UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2023-66
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 03/07/2023 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Alessandro TEMPERONI
Born on 1 May 1993 in Rome, (Italy)

ENHANCING DEEP LEARNING PERFORMANCE WITH SECOND-ORDER METHODS, RANDOM EMBEDDINGS, AND RELATION EXTRACTION

Abstract

Since the inception of civilization, the aspiration to create machines capable of thinking has persisted. Over the centuries, this dream has gradually come to fruition, with Artificial Intelligence now emerging as a field with numerous applications and promising research avenues. As a subfield, Deep Learning (DL) is dedicated to developing algorithms that can discern patterns in data, empowering machines to make predictions, draw conclusions, and carry out intricate tasks. This thesis delves into the enhancement of DL performance from various perspectives and vantage points, revealing the complexity of this field and the myriad ways it can be approached, while also highlighting the challenges of navigating through different levels of abstraction and maintaining the focus on the problem at hand.

In our exploration, we discuss second-order methods, random embeddings, and relation extraction. We address the initialization and optimization of neural networks (NNs) by introducing a new approximated chain rule, which aims to enable rapid and systematic training of NNs. Given that NNs are use-case sensitive, researchers and practitioners must undergo a series of laborious steps before deploying and ultimately releasing a functional model. Although the challenges of training and optimizing NNs are well understood, no single solution exists, and most contemporary approaches rely on simple and empirical heuristics. In Part II, our approximated chain rule for Hessian backpropagation transcends empirical first-order methods and lays a theoretical foundation for optimizing and training NNs. We systematically evaluate our approach through experiments, showcasing the superior efficiency of second-order methods across multiple datasets. In Part III, we shift our focus to analyzing the performance of random embeddings as a crucial tool for dimensionality reduction, with these embeddings playing a significant role in both Machine Learning (ML) and DL algorithms. Our research demonstrates improved bounds for sparse random embeddings compared to previous state-of-the-art techniques, with our bounds exhibiting considerable improvements across a range of real-world datasets. Our analysis strives to bridge the gap between theory and practice, providing robust and provable guarantees for sparse random embeddings while extending to Rademacher random embeddings and offering non-oblivious insights into input data. Lastly, in Part IV, we delve into information extraction, specifically targeting relation extraction. By examining the most advanced techniques and tools for extracting and analyzing textual data, we demonstrate their applicability in real-world scenarios. Our approach combines distant supervision, few-shot learning, OpenIE, and various language models to enhance the task of relation extraction, showcasing the capabilities of these methods through a simple and efficient approach to extracting relational labels from text. The diverse approaches and strategies discussed in this thesis collectively succeed in augmenting deep learning performance across various scenarios and applications.

Acknowledgements

Pursuing a Ph.D. has been an extraordinary adventure, filled with remarkable encounters and invaluable experiences. Throughout this incredible journey, I have had the privilege of meeting exceptional individuals whose inspiration and guidance have shaped my academic pursuits. While the path has not always been easy, the support I have received from both near and far has been truly heartening. I am immensely grateful and feel blessed to have such an outstanding network of people in my life.

I would like to express my deepest gratitude to my supervisor, Prof. Martin Theobald, for his unwavering support and guidance throughout my Ph.D. and beyond. Our paths first crossed in the autumn of 2017, when I was an Erasmus visiting student at the University of Luxembourg. Prof. Theobald graciously attended the first presentation in English I ever did in my life, which focused on the subject of maximum entropy inference processes. Despite having no obligation to do so, he approached me after the presentation to commend my efforts in distilling a complex topic into easily comprehensible language and relatable examples. This small act of kindness meant a great deal to me and set the tone for our working relationship.

I am grateful to Vinu, with whom I had the pleasure of sharing an office for two years. Your stimulating discussions truly made our shared space a welcoming and productive environment.

A special mention goes to Maria, not only for being my office mate during the latter half of my Ph.D. but also for being an exceptional project partner. Your support, enthusiasm, and dedication truly inspired me, I am deeply thankful.

I would also like to extend my appreciation to Mauro, a valued colleague and co-author. Your insights and expertise have been invaluable in our joint endeavors, and I am grateful for the opportunity to have worked alongside you.

Thanks also to Jingjing, a remarkable colleague whose presence has enriched my Ph.D. experience. Your professional support has made a significant impact on my journey.

In addition, I would like to extend my gratitude to Prof. Bouvry and Prof. Sorger for their contributions as members of my CET committee. I am truly appreciative of the time and effort they dedicated to providing guidance and constructive feedback throughout the process.

I am also grateful to Prof. Weikum and Prof. Merialdo for their roles as external reviewers, offering interesting perspectives and suggestions. In particular, I would like to acknowledge Prof. Merialdo for his enduring impact on my academic journey. Not only did he serve as my advisor during my Master's thesis, but he also introduced me to Matteo Cannaviccio, whose support and collaboration have been instrumental in my development as a researcher.

I would like to express my heartfelt appreciation to both Matteo Cannaviccio and Christian Hundt, who have been incredibly supportive during the most challenging moments of my Ph.D. journey. Engaging in research discussions with you has been an honor and a privilege, and I am deeply grateful for the opportunities to learn from your expertise and insights.

Finally, I would like to convey my sincere gratitude to my friends and family, whose indefatigable support, love, and motivation have been the foundation of my success. Your presence has been a beacon of light throughout this journey, and I am eternally grateful for the strength and inspiration you have provided. Thank you all!

Contents

List of Figures

List of Tables

[1]Here, for SETFIT, a subset of 15 relations is used similar to Tunstall et al., 2022

List of Abbreviations

AI Artificial Intelligence
DL Deep Learning
ML Machine Learning
NN NN
DNN Deep NN
GPU Graphic Processing Units
TPU Tensor Processing Units
RL Reinforcement Learning
SVM Support Vector Machine
SGD Stochastic gradient descent
ADAM Adaptive Moment Estimation
NLP Natural Language Processing
PCA Principal Components Analysis
SVD Singular Value Decomposition
IR Information Retrieval
BFGS Broyden–Fletcher–Goldfarb–Shanno
NED Named Entity Disambiguation
NER Named Entity Recognition
IE Information Extraction
OpenIE Open Information Extraction
RE Relation Extraction
QA Question Answering
KBP knowledge Base Population
LM Language Model
SRL Semantic Role Labeling
KN KnowledgeNet

Notation

In trying to find a notation which is internally consistent, I have adopted a number of general principles as follows.

a	A scalar
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
A	A tensor
$\boldsymbol{I}$	Identity matrix
$\text{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
X	A random variable
$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, ..., n\}$	The set containing all integers between 0 and n
$\mathcal{G}$	A graph
$\boldsymbol{A}^T$	Transpose of matrix A
$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$	Gradient of f with respect to $\boldsymbol{x}$
$\|x\|_p$	L^p norm of x
$\|x\|$	L^2 norm of x
$E(x)$	Error function
$\mathbb{E}\left[\|X\|\right]$	Expectation of a random variable X

This thesis is dedicated to my mother and my sister and to the memory of my father.

Part I

Introduction

Chapter 1

Introduction

1.1 Study Context

1.1.1 Artificial Neural Networks

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) that enables computers to learn from examples, much like humans do. It focuses on the development of algorithms capable of identifying patterns in large datasets, allowing machines to make predictions, draw conclusions, and perform complex tasks. In recent years, ML has gained immense popularity due to its ability to deliver remarkable results that were previously unattainable. It is now utilized in numerous disciplines, including Computer Science, Bioinformatics, Applied Statistics, Computational Physics, and many many more. Deep Learning (DL) models, often referred to as Deep Neural Networks (DNNs) consist of multiple interconnected layers. These layers, which include input, hidden, and output layers, process and transform data to ultimately produce a desired output. The term "deep" refers to the number of hidden layers in a neural network, which can range from 2-3 layers in traditional networks to hundreds or even thousands in more complex models.

One of the key strengths of DL models is their ability to handle unstructured data, such as images, text, and sound. They can perform tasks directly from raw data, without the need for manual feature extraction. This enables DL models to achieve state-of-the-art accuracy in various applications, sometimes even surpassing human-level performance.

The fundamental block of NNs, the perceptron, came out as a work by Rosenblatt, 1958 and it was in the news for weeks. At the time, researchers thought that the problems related to AI were solved entirely. This was far from being true and was pointed out by Minsky and Papert, 1972, the huge expectations around perceptron were shattered as it has been shown that the perceptron is, in reality, a linear classifier and cannot solve or learn XOR problems (non-linear problems). The generation of machine learners after Rosenblatt successfully extended linear models to represent nonlinear functions of x by applying the linear model to a transformed input $\phi(x)$, where ϕ is a nonlinear transformation. Further progress in DL was achieved in the 1980s when some of the earliest learning algorithms were intended to be computational models of biological learning (that is models of how learning happens in the brain), but we had to wait until recent years to see DL techniques become feasible and for its widespread adoption. There are two main reasons for this: the requirement for large amounts of labeled data and the need for substantial computing power. The advent of Big Data has provided researchers with vast datasets to train their models, while advances in hardware, such as Graphics Processing Units (GPUs) and specialized accelerators like Tensor Processing Units (TPUs), have made it possible to efficiently train large-scale DNNs.

DL is a powerful technique that has revolutionized the field of AI. By leveraging vast amounts of data and advanced NN architectures, DL models can perform a wide range of tasks with exceptional accuracy, making them indispensable tools in various disciplines and industries. The ongoing development of hardware and algorithms promises to further expand DL's potential applications and capabilities in the coming years.

Fundamental Problems

The types of learning are usually divided into two main categories: supervised learning and unsupervised learning.

Supervised learning involves training a model with labeled data, where inputs x are paired with corresponding outputs y. The goal is to learn a mapping from x to y, given a labeled set of input-output pairs $\{(x_i, y_i)\}_{i=1}^{n}$. This is called the training set, and n is the number of training examples. In its simplest form, each input x_i used for training is represented by a d-dimensional vector that represents the features of the data such as height and weight of a person. However, in more general cases, the input may consist of complex objects, such as a graph, an image, a time series, sentences, etc. The form of the output can vary, but many methods assume y_i to be a categorical variable from a finite set or a real-valued scalar. When the output is categorical, the problem is called classification problem, and when the output is a real-valued scalar, it is known as a regression problem.

Unsupervised learning involves finding intriguing patterns in data with no provided output to guide the learning process. In fact, here we only have given inputs $\{x_i\}_{i=1}^{N}$. This task is often referred to as knowledge discovery and is not as clearly defined as supervised learning because there are no prescribed patterns to search for, and no easily identifiable error metric to use.

A less frequently utilized form of learning is reinforcement learning (although recently has gained an incredible amount of attention in research), which involves learning how to act or behave in response to intermittent reward or penalty signals. However, this topic is beyond the scope of this thesis and it will not be discussed.

Supervised Learning

Classification involves learning a mapping from inputs x to outputs y, where $y \in \{1, ..., c\}$ with c being the number of classes. When there are two classes involved, it is referred to as binary classification. When more than two classes are involved, it is called multi-class classification. If the class labels are not mutually exclusive, it is referred to as multi-label classification. One way to approach this problem is to consider it as a function approximation. We assume $y = f(x)$, and the goal is to estimate the function f based on a labeled training set. The main objective is to make predictions on new inputs, which have not been previously encountered since making predictions on the training set is relatively easy. This ability to generalize is crucial, and the aim is to train NNs with strong generalization skills to tackle challenging classification problems. Some examples of classification problems in DL:

- **Image classification:** Image classification is a popular problem that involves identifying the object or scene depicted in a digital image. Images can be considered as high-dimensional vectors and classified into various categories such as cat, car, human, or airplane. A classification model can be developed to categorize images into different classes. For instance, DL techniques can be used to automate the classification of images of dogs and cats based on pre-classified images of these animals.

- **Customer behavior prediction:** Classification models can categorize customers based on their buying patterns and website browsing behavior. By using such models, it is possible to predict whether a customer is likely to purchase more items or not.

- **Spam filtering:** To identify spam emails, an algorithm is trained to learn the attributes that distinguish spam from non-spam emails. After training the classification model, it can be used to sort incoming emails as either spam or non-spam.

- **Document classification:** To categorize documents into various groups, a multinomial classification model can be employed. Such a classification model can be viewed as a function that assigns a category label to a given document.

- **Current trends:** One of the most notable trends in this domain is the advancement of transformer-based architectures, such as GPT-4 and its contemporaries. These models have demonstrated exceptional capabilities in generating high-quality and coherent text, images, and even audio. Researchers have also been focusing on improving the efficiency and scalability of these models by leveraging techniques like sparsity, model distillation, and low-precision training. Additionally, the exploration of semi-supervised learning methods has been gaining traction, as they facilitate the creation of more robust generative models with reduced dependence on labeled data. As we progress, the fusion of these advancements is expected to drive the development of increasingly sophisticated generative models with widespread applicability across diverse fields.

Regression is a technique that resembles classification in the sense that both are supervised ML methods used to make predictions based on input data. While classification focuses on categorizing data points into distinct classes or groups, regression deals with predicting continuous numerical values. A continuous response variable represents a measurable quantity that can assume any value within a certain range. This is in contrast to categorical outputs, where the response variable can only take on a limited number of discrete values, such as labels or classes.

One common application of regression is to predict the expected claim amount for an insured individual. In this scenario, an algorithm might be trained using historical data, including factors such as the individual's age, gender, and driving behavior. By identifying patterns and relationships, the regression model can make more accurate predictions about claim amounts, which can ultimately help insurance companies assess risk and set premiums accordingly. Another example of regression is forecasting future security prices in the financial sector. By analyzing historical price data, along with other relevant information like market trends and economic indicators, a regression model can be trained to make informed predictions about future security prices. These predictions can be invaluable for algorithmic trading, which involves leveraging computer algorithms to execute trade orders at high speeds and frequencies. By incorporating regression-based predictions into their strategies, traders can optimize their investment decisions and potentially generate higher returns.

Unsupervised Learning

Unsupervised learning is a subcategory of ML that involves the presentation of input data to the system without the accompanying output data. The system's primary objective is to identify and reveal hidden patterns, structures, or relationships within the data, which may not be immediately noticeable. This type of learning has the advantage of not requiring human experts to laboriously label data, a process that can be expensive, time-consuming, and demanding of specialized knowledge. In contrast to supervised learning, where the desired output for each input is explicitly provided, unsupervised learning does not offer direct guidance on what the system should be searching for. This makes the unsupervised learning process more challenging as the algorithm must explore and analyze the data independently in order to discern meaningful information.

Unsupervised learning is thought to more closely resemble the learning processes of humans and animals, where the majority of learning is self-directed and rooted in the recognition of patterns without explicit instruction. This enables the discovery of novel insights and the development of more generalized understanding based on the data.

The benefits of unsupervised learning extend to its applicability across a wide range of data types. Additionally, it can help circumvent issues arising from using labeled data that may not be representative of the true data distribution or contain enough information to accurately estimate the parameters of intricate models.

Some examples of unsupervised learning are:

- **Self-supervised learning**: Self-supervised learning is a subcategory of unsupervised learning that focuses on training models by generating auxiliary tasks, which do not require labeled data. This enables models to learn meaningful representations by solving these proxy tasks. In Natural Language Processing (NLP), self-supervised models like BERT have revolutionized the field by capturing rich linguistic structures through pretraining on massive text corpora. In computer vision, frameworks such as SimCLR and MoCo have demonstrated state-of-the-art performance in learning visual representations by predicting transformations or contrastive learning on unlabeled images.

  Clustering and dimensionality reduction: Unsupervised learning algorithms, like clustering and dimensionality reduction, are crucial for understanding complex data structures and discovering latent patterns. Recent advancements in this area include methods like UMAP and t-SNE, which have proven effective in visualizing high-dimensional data in lower-dimensional spaces, facilitating more interpretable representations. These techniques are widely employed in fields like Bioinformatics, Finance, and Social Sciences, where large, high-dimensional datasets are common.

- **Reinforcement Learning (RL)**: Integrating unsupervised learning with reinforcement learning has garnered significant interest, as it allows agents to autonomously learn from their environment without explicit supervision. Techniques like intrinsic motivation, curiosity-driven exploration, and unsupervised skill discovery are being investigated to enable agents to learn more effectively and generalize better across tasks. This research direction holds great potential for developing more intelligent and adaptable artificial systems.

- **Transfer learning and multitask learning:** One of the key challenges in unsupervised learning is leveraging the knowledge acquired from one domain or task to improve performance in another. Researchers are exploring transfer learning and multitask learning techniques that enable models to share and reuse learned representations, thereby reducing the need for extensive labeled data in new tasks. These approaches have demonstrated success in various applications, such as NLP, computer vision, and reinforcement learning.

1.1.2 Natural Language Processing

Over the past few decades, researchers have been addressing numerous challenges in the field of AI with the goal of enabling machines to mine knowledge from these unstructured sources. This is crucial as the volume of available online data continues to expand rapidly. Instead of reorganizing the Web to make it more machine-readable, the primary focus of research in these areas is to develop methods and algorithms that allow machines to automatically create, populate, and maintain extensive knowledge graphs or knowledge bases. Knowledge graphs are structured representations of information, linking entities and their relationships in a way that facilitates the extraction of valuable insights and information from diverse online data sources. By enabling machines to generate knowledge graphs automatically, researchers aim to improve the efficiency and effectiveness of information retrieval, recommendation systems, and various AI applications. This development will reduce the time and effort required by users to find the answers they need and will further advance the capabilities of AI systems in understanding and processing natural language data.

NLP is an interdisciplinary field that combines Computational Linguistics, AI, and Computer Science to enable machines to understand, interpret, and generate human language. As one of the most critical areas in AI research, NLP has witnessed significant progress and technological advancements in recent years. State-of-the-art NLP systems can now perform a myriad of complex tasks with remarkable accuracy. These tasks can include:

- **Sentiment Analysis:** NLP systems can gauge the sentiment behind a piece of text, such as determining if it is positive, negative, or neutral. This capability is crucial for businesses to analyze customer feedback or public opinion on social media platforms.
- **Machine Translation:** NLP has revolutionized the translation industry by enabling fast and accurate translations between different languages. Machine translation systems have become increasingly sophisticated, breaking down language barriers in real-time communication and easing access to information across linguistic boundaries.
- **Summarization:** Automatic summarization involves condensing lengthy pieces of text into shorter, more digestible summaries while retaining the essential information. NLP techniques have facilitated the development of summarization algorithms that can efficiently extract key points from documents, news articles, or scientific papers.
- **Question-Answering:** NLP systems can process and respond to natural language queries with relevant answers, drawing from vast repositories of knowledge. These systems can save time and effort in searching for information by providing direct answers to user inquiries.

- **Information Extraction:** NLP enables the automatic extraction of structured data from unstructured text sources. This process involves identifying entities, relationships, and events within the text, enabling the efficient organization and analysis of data.

However, despite these advances, NLP still faces numerous challenges that require further research and innovation.

Information Extraction (IE) is a subfield of NLP that focuses on automatically identifying and extracting structured information from unstructured text data. State-of-the-art IE systems have demonstrated significant improvements, but the field still grapples with various challenges. For example, IE systems often struggle with ambiguities arising from polysemy, homographs, and syntactic variations. Extracting meaningful information requires a deep understanding of context, which remains a challenge for current NLP models. While significant progress has been made, identifying and categorizing entities into fine-grained types (e.g., distinguishing between types of organizations, diseases, or events) remains a challenge. This is due to the difficulty of creating comprehensive ontologies and the scarcity of labeled data for such fine-grained distinctions. Also identifying and extracting complex relationships or events that span over multiple sentences or documents is a challenging task. Current models may not be well-equipped to handle such intricate structures and dependencies, especially when they require world knowledge or commonsense reasoning. Domain adaptation is another very important aspect of IE systems as they are usually tailored to specific domains, and their performance may degrade when applied to new domains or genres. Developing more robust models that can efficiently adapt to different settings remains an open research problem. Finally, real-world text data can be noisy, inconsistent, or incomplete, making it difficult for IE systems to extract accurate information. Developing models that are robust to these issues and can effectively handle uncertain or missing information is a key challenge in NLP.

To address these challenges, researchers and practitioners are exploring new approaches, including transfer learning, multitask learning, and unsupervised learning, to improve the generalization capabilities of IE systems. Additionally, incorporating external knowledge sources, such as knowledge graphs, and developing methods for commonsense reasoning can help in enhancing the performance of IE models. As the field of NLP continues to evolve, tackling these challenges in information extraction will play a crucial role in unlocking the full potential of AI systems to process and understand human language.

1.2 Terminology and Mathematical Concepts

In this section, we are going to provide the technical terminology and overview of the mathematics concepts this thesis is based on.

1.2.1 Tensors, Derivatives and Algebra

Tensors in DL are multi-dimensional arrays that serve as fundamental data structures for representing and manipulating data. They originate from various sources and play a crucial role in the functioning of models.

Tensors are used to represent input data, such as images, text, or audio, that is fed into a DL model. For example, an image can be represented as a 3-dimensional tensor with dimensions corresponding to width, height, and channels, where each element corresponds to a pixel value. DL models consist of layers with trainable parameters (weights and biases), which are adjusted during training to minimize a loss function. These parameters are stored as tensors, enabling efficient computation and updates during the training process. As data flows through a model, each layer transforms the input into an intermediate representation which is also represented as tensors and passed on to the subsequent layers. During the training process, gradients are computed using backpropagation to update the model parameters. Gradients are the partial derivatives of the loss function with respect to the model parameters, and they are also represented as tensors.

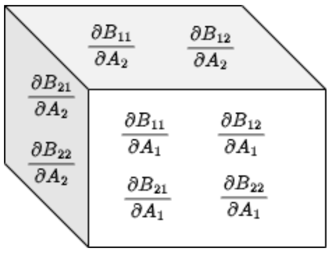
Tensor Derivatives. For two *tensors* $\mathsf{A} = \mathsf{A}_{j_1,\dots,j_q}$ and $\mathsf{B} = \mathsf{B}_{i_1,\dots,i_p}$, of rank q and p respectively, the *derivative* $D = D_{\mathsf{A}}\mathsf{B}$ is a tensor of rank $q+p$ with coordinates $D_{j_1,\dots,j_q,i_1,\dots,i_p} = \frac{\partial \mathsf{B}_{j_1,\dots,j_q}}{\partial \mathsf{A}_{i_1,\dots,i_p}}$.

For example, let

$$\mathsf{A} = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$$

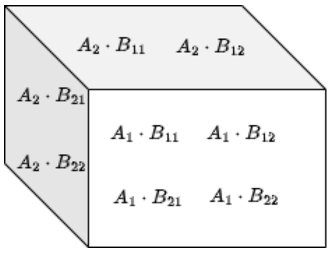
and

$$\mathsf{B} = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$


$\frac{\partial B_{11}}{\partial A_2}$
$\frac{\partial B_{12}}{\partial A_2}$
$\frac{\partial B_{21}}{\partial A_2}$
$\frac{\partial B_{22}}{\partial A_2}$
$\frac{\partial B_{11}}{\partial A_1}$
$\frac{\partial B_{12}}{\partial A_1}$
$\frac{\partial B_{21}}{\partial A_1}$
$\frac{\partial B_{22}}{\partial A_1}$

Tensor Products. *Contraction* sums over paired indices (axes), thus lowering the rank by 2 (or more when more pairs are specified). For example, contracting positions a and b in A produces the tensor $\sum_{i_a=i_b} \mathsf{A}_{i_1\dots i_a \dots i_b \dots i_p}$ with indices $\{i_1, \dots, i_p\} \setminus \{i_a, i_b\}$, where the dimensions of paired indices should match. A *full tensor product* combines tensors A and B by cross-multiplications $(\mathsf{A} \otimes \mathsf{B})_{i_1,\dots,i_p,j_1,\dots,j_q} = \mathsf{A}_{i_1,\dots,i_p} \cdot \mathsf{B}_{j_1,\dots,j_p}$, thereby producing a tensor of rank $p+q$. A *tensor dot-product* is the full tensor product followed by contraction of two compatible dimensions. For example, the standard matrix product of $\boldsymbol{A}_{i,j}$ and $\boldsymbol{B}_{k,l}$ is the tensor product followed by contraction of j and k. We denote the dot-product by $\bullet$, thereby omitting the contracted axes when this is clear from the context.

For example, given the same tensors A and B as above, we have:



Chain & Product Rules. Tensors obey similar chain and product rules as matrices. Specifically, we have $D_x(\mathsf{A} \bullet \mathsf{B}) = D_x\mathsf{A} \bullet \mathsf{B} + \mathsf{A} \bullet D_x\mathsf{B}$. Also, when $\mathsf{B} = f(\mathsf{A}(x))$ holds, we have $D_x\mathsf{B} = D_{\mathsf{A}}f \bullet D_x(\mathsf{A})$. The contraction is over all dimensions of A which match the arguments of f.
Spectral Norm. For any matrix $\boldsymbol{A}$, the *singular eigenvalues* are defined as the square roots of the eigenvalues of $\boldsymbol{A}^T\boldsymbol{A}$ (which is square symmetric and positive definite). The *spectral norm* then is the biggest singular eigenvalue of $\boldsymbol{A}$.
Neural Networks. From an algebraic perspective, we look at a *neural network* (NN) as a chain of mappings of the form:

$$\boldsymbol{z}^{(k+1)} = f^{(k)}\left(\boldsymbol{W}^{(k)} \cdot \boldsymbol{z}^{(k)} + \boldsymbol{b}^{(k)}\right)$$

which sequentially processes an *input vector* $\boldsymbol{x} = \boldsymbol{z}^0$ through a number of *layers* $k = 0 \ldots n-1$. We assume that $\boldsymbol{z}^{(k)}$ are real-valued vectors of shape $[d_k]$, *weights* $\boldsymbol{W}^{(k)}$ are matrices of shape $[d_{k+1}, d_k]$, *biases* $\boldsymbol{b}^{(k)}$ are of shape $[d_{k+1}]$, and $f^{(k)}$ are (possibly non-linear) *activation functions* which are applied element-wisely. The task of learning then is to minimize a given *loss function* $L(\boldsymbol{z}, t)$ where $\boldsymbol{z} = \boldsymbol{z}^n$ is the network output and t is the ground-truth, over the weights $\boldsymbol{W}^0, \ldots, \boldsymbol{W}^{n-1}$.

1.2.2 Probability Theory, Combinatorics and Optimization

Probability distributions and random variables play a significant role in many dimensionality reduction techniques, particularly those based on probabilistic models. They are used to model relationships between data points, estimate model parameters, and capture the uncertainty associated with the reduced representations. In dimensionality reduction, the goal is to find a lower-dimensional representation of high-dimensional data that preserves the underlying structure and relationships. Probabilistic models help capture these relationships by modeling the data as random variables following certain probability distributions. Probabilistic dimensionality reduction methods, assume that the data points are generated from an underlying probability distribution. These methods use random variables to model the observed data and the latent lower-dimensional space, and they learn the probability distributions that best explain the observed data. Some dimensionality reduction techniques use probability distributions to model pairwise similarities between data points in both the high-dimensional and low-dimensional spaces. The objective is to minimize the divergence between these two probability distributions so that the lower-dimensional representation preserves the original data structure.
Probability Distributions. Throughout the thesis we work on basic probability distributions: normal, binomial, and Rademacher. Given a vector $\boldsymbol{x}$ we denote by $\|\boldsymbol{x}\|$ its euclidean norm and by $\|\boldsymbol{x}\|_0$ the number of its non-zero components; we also say that $\boldsymbol{x}$ is ℓ-sparse with $\ell = \|\boldsymbol{x}\|_0$, for example $\boldsymbol{x} = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0\right)$ is 3-sparse.
Random Variables. The d-th *norm* of a vector $\boldsymbol{x}$ and a random variable X, respectively, are defined as:

$$\|\boldsymbol{x}\|_d = (\sum_i |\boldsymbol{x}_i|^d)^{\frac{1}{d}} \text{ and } \|X\|_d = \left(\mathbb{E}\left[|X|^d\right]\right)^{\frac{1}{d}}$$

we also define $\|\boldsymbol{x}\|_\infty = \max_i |\boldsymbol{x}_i|$ as usual. $\mathrm{Bern}(p)$ denotes the *Bernoulli distribution* with success probability p, while $\mathrm{Binom}(n, p)$ denotes the *binomial distribution* with n trials and success probability p. The *Rademacher distribution* takes values 1 and -1

with equal probabilities. Moreover, a random variable X is called *symmetric* when it has the same distribution as $-X$.
Majorization Order. We say that on n-dimensional vectors, majorization is defined as follows: x is dominated by y, denoted by $x \prec y$, if for their non-decreasing rearrangements $(x_i^{\downarrow})$ and $(y_i^{\downarrow})$ we have the inequality $\sum_{i=1}^{k} x_i^{\downarrow} \geqslant \sum_{i=1}^{k} y_i^{\downarrow}$ for $k = 1, \ldots, n$ with equality when $k = n$; equivalently, x can be produced from y by a sequence of *Robin-Hood operations* which replace

$$x_i > x_j,\ x_i \leftarrow x_i - \epsilon, x_j \leftarrow x_j + \epsilon \text{ for } \epsilon \in \left(0, \frac{x_i - x_j}{2}\right).$$

Intuitively, x being dominated by y means that x is more spread-out/dispersed than y. For example, we have $\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) \prec \left(\frac{1}{3}, \frac{1}{3}, 0, \frac{1}{3}\right)$ (it takes 3 Robin-Hood transfers).
Schur-convexity. Given a function $f : \mathbb{R}^n \to \mathbb{R}$ is the following property: $x \prec y$ implies $f(x) \leqslant f(y)$; we speak of Schur-concavity when the inequality is reversed. Schur-convex or Schur-concave functions are necessary symmetric; symmetric function is Schur-convex if

$$\left(\frac{\partial f}{\partial x_i} - \frac{\partial f}{\partial x_j}\right)(x_i - x_j) \geqslant 0 \text{ (Schur-Ostrowski criterion).}$$

For example, power sums $\sum_i x_i^q$ for $q \geqslant 1$ are Schur-convex.

We define the *moment domination* of a random variable Y over X, denoted as $X \prec_m Y$, by requiring $\mathbf{E}X^q \leqslant \mathbf{E}Y^q$ for all positive integers q. In particular, it implies that MGF of Y dominates the MGF of X, the majorization in the Lorentz stochastic order.

1.3 Structure of the Thesis

The structure of this thesis is organized into five distinct sections, each addressing a key aspect of the research. Part I sets the stage by introducing the study's context, motivation, and contributions that will be the main themes throughout the manuscript. This section provides a foundation for understanding the research's purpose and its significance within the field. In Part II, the focus shifts to second-order methods for DL. This section presents an overview of current state-of-the-art approaches, along with a discussion of their limitations and challenges. The purpose is to identify gaps in the existing methods and provide a comprehensive understanding of the current landscape. Following this analysis, the main contributions and novel approaches are proposed as solutions to the identified challenges, offering potential advancements in DL techniques. Part III delves into the performance of random embeddings as a critical tool for dimensionality reduction. These embeddings play a significant role in both ML and DL algorithms. This section examines the effectiveness of random embeddings in various contexts, comparing their performance to other dimensionality reduction techniques and assessing their potential benefits and drawbacks. In Part IV, the thesis narrows its focus to information extraction, specifically targeting relation extraction. This section explores the most advanced techniques and tools for extracting and analyzing textual data, demonstrating their application in real-world scenarios. The goal is to showcase the capabilities of these methods and provide insights into their potential impact on information extraction tasks. Lastly, Part V concludes the thesis by summarizing the key findings and contributions presented throughout the manuscript. This section also discusses the implications of the research and offers suggestions for future work in the field, providing a comprehensive wrap-up of the study's significance and potential impact on the research community.

The organization of the manuscript is as follows:

- **Chapter** 2 provides an introduction to second-order methods, which serve as a foundation for the subsequent chapters. These methods are essential to understanding the context and techniques employed in the research.

- **Chapter** 3 lays out the necessary background information for Part II of the thesis, offering context and setting the stage for the discussions and analyses that follow.

- In **Chapter** 4, the proposed solutions are presented, complete with theoretical foundations and experimental results from real-world datasets. This chapter demonstrates the practical implications of the proposed methods and draws upon the following publications.

  Maciej Skorski, Alessandro Temperoni and Martin Theobald (2021). "Revisiting Weight Initialization of Deep Neural Networks." In: *Proceedings of the Thirteenth Asian Conference of Machine Learning (ACML).* Online, pp. 1192-1207

  Alessandro Temperoni, Mauro Dalle Lucca Tosi and Martin Theobald (2023). "Efficient Hessian-based DNN Optimization via Chain-Rule Approximation." In: *Proceedings of the 6th Joint International Conference on Data Science & Management of Data (CODS-COMAD).* Mumbai, India, pp. 1192-1207

- **Chapter** 5 familiarizes the reader with dimensionality reduction techniques, which play a crucial role in ML and DL algorithms.

- **Chapter** 6 presents the key concepts required for Part III of the thesis, providing a foundation for understanding the material discussed in the subsequent chapters.

- In **Chapter** 7, the robustness of sparse random embeddings is explored. This chapter offers valuable insights into the efficacy of these embeddings in various contexts and draws upon the following publication.

  Maciej Skorski, Alessandro Temperoni and Martin Theobald (2022). "Robust and Provable Guarantees for Sparse Random Embeddings." In: *Advances in Knowledge Discovery and Data Mining: 26th Pacific-Asia Conference (PAKDD).* Online, pp. 211-223

- **Chapter** 8 proposes a non-oblivious analysis of Rademacher random embeddings with respect to the data. The work in this chapter is under review as

  Maciej Skorski, Alessandro Temperoni (2023). "Exact Non-Oblivous Performance of Rademacher Random Embeddings" In: (submitted)

- **Chapter** 9 introduces the topic of relation extraction, setting the stage for the exploration of advanced techniques and tools in the following chapters.

- In **Chapter 10**, an overview of the state-of-the-art in relation extraction is provided, offering a comprehensive understanding of the current landscape in this area.

- **Chapter 11** examines open information extraction as a potential solution for improving relation extraction, highlighting its capabilities and possible benefits. The work in this chapter is going to appear in the following publication.

  Alessandro Temperoni, Maria Biryukov and Martin Theobald (2023). "Enriching Relation Extraction with OpenIE." In: *Proceedings of the 12th International Conference on Data Science, Technology and Applications.*

- Finally, **Chapter 12** concludes the thesis by summarizing the key findings and contributions presented throughout the manuscript. This chapter also discusses the implications of the research and offers suggestions for future work in the field.

Part II

Second-order Methods for Deep Learning

Chapter 2

Introduction

2.1 Multilayer Perceptron

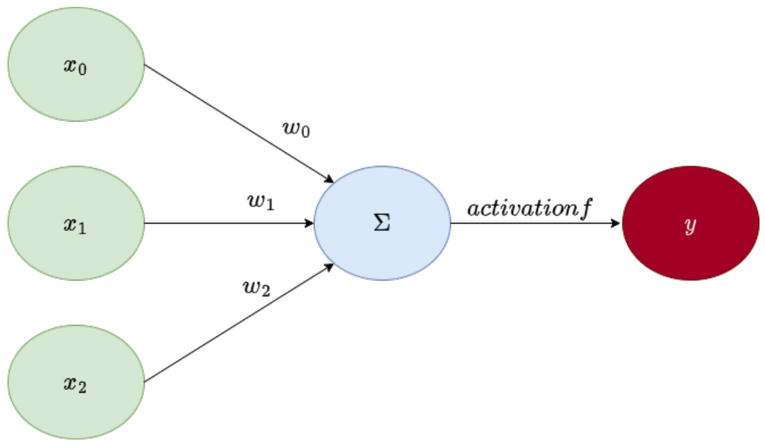
In this section, we are going to describe the perceptron, the building block of NNs, and we are going to see how it is possible to represent nonlinear functions of $\boldsymbol{x}$ by applying the linear model to a transformed input $\phi(\boldsymbol{x})$, where ϕ is a nonlinear transformation. We can think of ϕ as providing a set of features describing $\boldsymbol{x}$, or as providing a new representation of $\boldsymbol{x}$. The deep feedforward networks also called feedforward NNs or multilayer perceptrons come from this line of research. A feedforward network is a type of artificial NN that aims to approximate a specific function f^*, such as a classifier $y = f^*(\boldsymbol{x})$ that maps an input x to a category y. A feedforward network defines a mapping $y = f(\boldsymbol{x}; \boldsymbol{\theta})$ and learns the value of the parameters $\boldsymbol{\theta}$ that results in the best function approximation. The network operates by passing the input through a series of intermediate computations to ultimately produce an output. Unlike recurrent NNs, feedforward networks do not have feedback connections, meaning that the output does not influence the input.

Feedforward networks are crucial in ML and are often used as the basis for many commercial applications, such as convolutional networks (that are a specialized kind of feedforward networks) for object recognition. They are also a necessary step towards recurrent networks, which are commonly used for NLP.

The network is composed of many functions that are combined in a directed acyclic graph, with the overall function being learned through the network's weights or parameters. For example, we might have three functions $f^{(0)}, f^{(1)}$, and $f^{(2)}$ connected in a chain, to form $f(\boldsymbol{x}) = f^{(2)}(f^{(1)}(f^{(0)}(\boldsymbol{x})))$. In this case, $f^{(0)}$ is called the first layer of the network, $f^{(1)}$ is called the second layer, and so on. In DL, the goal is to learn a parameterized representation, ϕ, that best approximates the function. The model includes both $\boldsymbol{\theta}$ and $\boldsymbol{W}$ parameters, where $\boldsymbol{\theta}$ is used to learn ϕ from a wide range of functions and $\boldsymbol{W}$ maps the learned representation to the desired output. This is an example of a deep feedforward network, with ϕ defining a hidden layer. Through optimization algorithms, the $\boldsymbol{\theta}$ parameter is found to correspond to a suitable representation. To achieve high generality, a broad family of functions $\phi(\boldsymbol{x}; \boldsymbol{\theta})$ can be used.

An example of Rosenblatt's perceptron is depicted in Figure 2.1. This is a simplified network having just one layer and that can be written as $y = f(\boldsymbol{x} \cdot \boldsymbol{W} + \boldsymbol{b})$. The issue pointed out by Minsky was solved by using a nonlinear function to describe the features of the network. Typically, NNs use an affine transformation that is controlled by learned parameters, followed by a fixed, nonlinear activation function that is applied element-wisely. The recommended default activation function for modern feedforward NNs is the rectified linear unit or ReLU, defined as $g(z) = max\{0, z\}$. This activation function, when applied to the output of a linear transformation, yields a nonlinear transformation that is still close to linear, as it is a

FIGURE 2.1: Example of a perceptron



piecewise linear function with two linear pieces. The benefit of using rectified linear units is that they maintain many of the properties that make linear models easy to optimize with gradient-based methods and generalize well. In computer science, it is a common practice to build complex systems from minimal components, and just as a Turing machine's memory only needs to store 0 or 1 states, a universal function approximator can be built from rectified linear functions.

FIGURE 2.2: Rectified Linear Unit

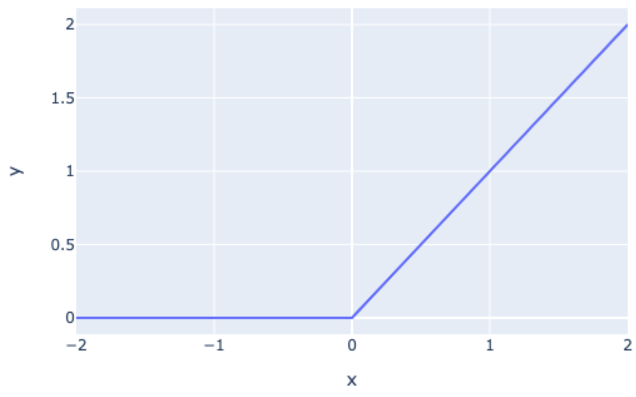
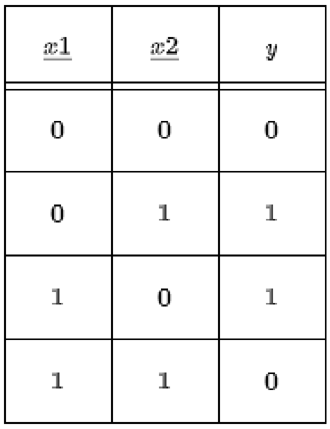
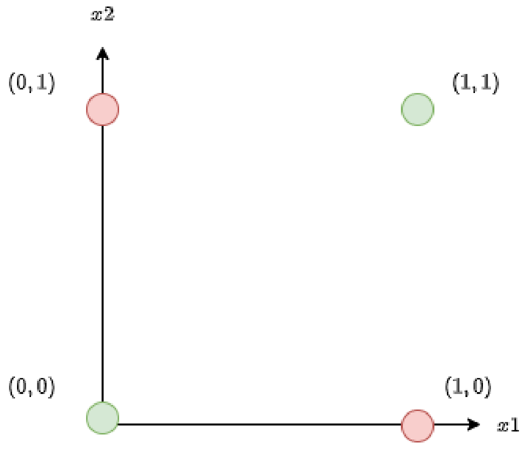


FIGURE 2.3: Illlustration of XOR



$x1$	$x2$	y
0	0	0
0	1	1
1	0	1
1	1	0



Example: learning the XOR function.

We can illustrate the concept of a feedforward network by presenting a practical example of a working feedforward network that can accomplish a straightforward task that linear classifiers cannot solve, which is learning the XOR function. For this purpose, we will demonstrate a fully functional feedforward network that is trained on a dataset of four points $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$, and the main objective will be to "only" fit the training set (as this is just a simple example and therefore we will not focus on statistical generalization). The "exclusive or" (XOR) function is an operation on two binary values x_1 and x_2. The XOR truth table and its 2D plot are shown in Figure 2.3. Let's then show a possible solution for the XOR problem. Let

$$\boldsymbol{W_1} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

$$\boldsymbol{b} = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

and

$$\boldsymbol{w_2} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}.$$

Let X be the input matrix including all four points represented in the truth table:

$$\boldsymbol{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Now, we walk through the network and, as a first step, we will multiply the input matrix by the first (hidden) layer's weight matrix:

$$\boldsymbol{XW_1} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

Then we add the bias vector b and obtain:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

We apply the rectified linear transformation to add non-linearity to this model

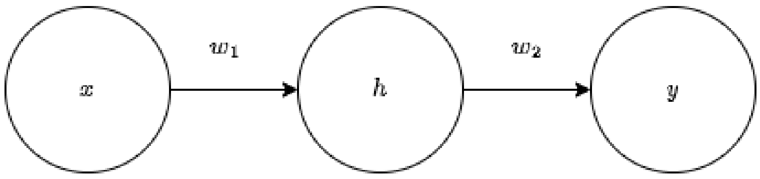
$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

This transformation has changed the relationship between the points. We finish by multiplying the matrix by the last's layer weight vector w_2

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

The NN has obtained the correct answer for every point. The solution was explicitly defined in this example, and it was demonstrated that it achieved zero error. However, in a realistic scenario, the number of parameters and training examples can reach billions, which means we cannot rely on guessing the solution as we did in this case.

FIGURE 2.4: XOR neural network



2.2 Gradient-based Learning

Training a NN is similar to training other ML models using *gradient descent* (Cauchy, 1874). Gradient descent is an iterative optimization algorithm that leverages first-order partial derivatives to minimize a cost function (it will be explained more in detail in section 2.4). However, the primary difference between linear models and NNs is the nonlinearity introduced by the latter, which causes many loss functions to become non-convex. As a result, NNs are typically trained using iterative, gradient-based optimization techniques, rather than the linear equation solvers used for linear regression models or convex optimization algorithms used for logistic regression or support vector machines (SVMs). Unlike convex optimization, stochastic gradient descent applied to non-convex loss functions lacks convergence guarantees and is highly sensitive to initial parameter values. To initialize feedforward NNs, it is crucial to set all weights to small random values and biases to zero or small positive values. It is worth noting that we can also train models such as linear regression and SVMs with gradient descent, especially when dealing with an extremely large training set. Therefore, training a NN is not very different from training any other

ML model. However, computing the gradient for a NN is slightly more complicated, but it can still be done efficiently and exactly.

2.2.1 Output Units

To apply gradient-based learning, we need to choose a loss function and determine how to represent the output of the model. Fortunately, the loss functions for DNNs are similar to those used for other parametric models. In most cases, we use the principle of maximum likelihood and the cross-entropy between the training data and the model's predictions as the loss function. The choice of the loss function is closely linked to the choice of the output unit used to represent the output. Any type of NN unit used as an output can also be used as a hidden unit. The role of the output layer is to provide some additional transformation from the features to complete the task that the network must perform. There are three main types of output units: linear, sigmoid, and softmax. Linear units are based on an affine transformation with no nonlinearity. Sigmoid units are used for binary classification problems where the variable y is either 0 or 1. Softmax units are used when we want to represent a probability distribution over a discrete variable with n possible values. The choice of the output unit affects the form of the cross-entropy function. For example sigmoid units, which have function

$$f(x) = \frac{1}{1 + e^{-x}}$$

have a saturation effect that can prevent gradient-based learning from making good progress, so the $-logP$ used in maximum likelihood undoes the exp of the sigmoid to avoid this issue. Softmax functions are most commonly used as the output of a classifier, to represent the probability distribution over n different classes.

Example Softmax

The softmax function can be considered as a (probabilistic) softer version of the argmax. Given a list of values from the last hidden layer of a network

$$[1, 3, 2]$$

applying argmax to this list returns

$$[0, 1, 0]$$

what if one is less sure about the outcome of the network and wants to express the argmax with likelihoods? This can be done by converting the values in the list into probabilities that sum up to 1.0. This is exactly what softmax does.

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where K is the number of classes. For example, every value in the list $[1, 3, 2]$ can be turned into a probability:

$$probability_1 = exp(1)/(exp(1) + exp(3) + exp(2)) = 0.091$$

$$probability_3 = exp(3)/(exp(1) + exp(3) + exp(2)) = 0.665$$

$$probability_2 = exp(2)/(exp(1) + exp(3) + exp(2)) = 0.244$$

2.2.2 Hidden Units

Selecting the appropriate type of hidden unit for a feedforward NN is a unique design challenge that sets it apart from most other DL models. This particular aspect remains an active area of research, with no definitive methodologies to address the issue. While rectified linear units (ReLU) serve as an excellent default option for hidden units, numerous alternatives exist. In this discussion, we will explore three widely-used hidden unit types in DL: rectified linear units, logistic sigmoid, and hyperbolic tangent. The process of determining which hidden unit type to use can be challenging, as it often involves a trial-and-error approach. ReLU, an activation function we have previously encountered, is defined as $g(z) = max\{0, z\}$. It is straightforward to optimize due to its linear-like behavior, and several generalizations of ReLU have been developed. These variations typically perform comparably well, and in some instances, even outperform the original ReLU. Prior to the rise of ReLU, NNs predominantly employed the logistic sigmoid activation function ($g(z) = \sigma(z)$) or the hyperbolic tangent activation function ($g(z) = tanh(z)$). These two activation functions share close relationships. We have previously discussed the sigmoid function and noted that it requires an appropriate cost function to be compatible with gradient-based learning. Interestingly, when it is necessary to use a sigmoidal activation function, the hyperbolic tangent function generally outperforms the logistic sigmoid.

Example ReLU

When ReLU activation function is applied to the matrix of the weights, every negative value is removed and replaced with 0. For example, the matrix $\boldsymbol{A}$ below

$$\boldsymbol{A} = \begin{bmatrix} 34 & 11 & -7 & 26 \\ 12 & -79 & -51 & 93 \\ -1 & 48 & 66 & 81 \\ 153 & -11 & 21 & -37 \\ 43 & -84 & 62 & 14 \end{bmatrix}$$

is transformed into

$$\boldsymbol{ReLU}(\boldsymbol{A}) = \begin{bmatrix} 34 & 11 & 0 & 26 \\ 12 & 0 & 0 & 93 \\ 0 & 48 & 66 & 81 \\ 153 & 0 & 21 & 0 \\ 43 & 0 & 62 & 14 \end{bmatrix}$$

2.2.3 Backpropagation

Backpropagation is a fundamental step for the optimization of gradient-based learning. It is an application of the chain rule from calculus, which efficiently computes gradients (first-order partial derivatives) of a loss function with respect to the parameters of the network. These gradients are then used in conjunction with an optimization algorithm, such as stochastic gradient descent, to update the model's parameters and minimize the loss function. Backpropagation works in two main steps: forward pass and backward pass. During the forward pass, the input data is passed through the NN, layer by layer, transforming it into intermediate activations using a combination of linear transformations and non-linear activation functions. The output of the last layer produces the network's predictions. The loss function is then computed. The goal is to minimize this loss function by adjusting the parameters of the neural network. The backward pass, where the backpropagation comes into play, computes the gradients of the loss function with respect to each parameter of the network. The process starts with calculating the gradient of the loss function with respect to the output layer's activations. This gradient is then propagated backward through the network, from the last layer to the first, by successively applying the chain rule to compute the gradients with respect to the parameters and activations of each layer. The key insight of backpropagation is that the gradient of the loss function with respect to a layer's input can be computed using the gradient with respect to its output, thus avoiding redundant calculations and making the computation efficient. Once the gradients have been computed for all the parameters, they are used to update the weights and biases in the network according to an optimization algorithm. The most common optimization method is stochastic gradient descent (SGD), which adjusts the parameters by taking a step in the opposite direction of the gradients, scaled by a learning rate. In practice, the backpropagation algorithm is performed iteratively over mini-batches of data, enabling the model to learn from multiple examples before updating the parameters. This approach, called mini-batch gradient descent, strikes a balance between computational efficiency and the quality of the gradient estimates. The entire process of forward pass, backward pass, and parameter updates is repeated for multiple epochs until the model converges to a solution with minimized loss.

2.3 Weight Initialization

DL training algorithms have an iterative nature and require the user to specify an initial point from which to begin the iterations. The choice of weight initialization can greatly affect the outcome of the training process. If the initial point is unstable, the algorithm may encounter numerical difficulties and fail altogether. Even when the algorithm does converge, the initial point can determine how quickly it converges and whether it converges to a point with high or low cost. Additionally, the initial point can have a significant impact on the generalization error of the model.

Currently, initialization strategies for DL models are simple and heuristic. However, designing improved strategies is challenging because the optimization of NNs is not yet well understood. Many initialization strategies aim to achieve certain properties when the network is first initialized, but it is unclear which of these properties are preserved during the training process. Additionally, some initial points may be beneficial for optimization but detrimental for generalization. Our understanding of how the initial point affects generalization is limited, offering little guidance on how to select the best initial point.

The only property that is known with certainty is that the initial parameters must "break symmetry" between different units. If two hidden units are connected to the same inputs, they must have different initial parameters in order to compute different functions. This is achieved by randomly initializing the parameters from a high-entropy distribution as it is computationally cheaper and unlikely to assign any units to compute the same function. Typically, NNs are almost always initialized to values drawn from a Gaussian or uniform distribution. The scale of the initial distribution can have a large effect on both the optimization procedure and the ability of the network to generalize. While the choice of Gaussian or uniform distribution does not seem to matter much but it has not been extensively studied.

Example Random Weight Initialization

Random weight initialization with small values from a high-entropy distribution. The value of the entropy is a function of the piece-wise entropy and of the size of the matrix. Usually, for DNN, the latter is a very big number.

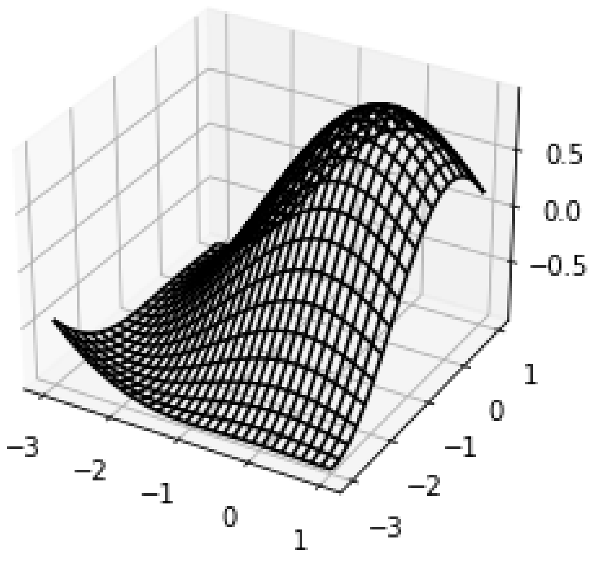
$$W = \begin{bmatrix} 0.65 & 0.68 & 0.49 & 0.5 & 0.38 & 0.42 & 0.68 & 0.48 & 0.73 & 0.49 \\ 0.54 & 0.32 & 0.58 & 0.53 & 0.46 & 0.34 & 0.36 & 0.54 & 0.33 & 0.5 \\ 0.65 & 0.52 & 0.52 & 0.52 & 0.59 & 0.51 & 0.6 & 0.47 & 0.63 & 0.72 \\ 0.57 & 0.5 & 0.52 & 0.41 & 0.45 & 0.39 & 0.55 & 0.62 & 0.54 & 0.52 \\ 0.4 & 0.4 & 0.51 & 0.73 & 0.47 & 0.63 & 0.77 & 0.28 & 0.77 & 0.48 \\ 0.61 & 0.53 & 0.69 & 0.52 & 0.49 & 0.68 & 0.59 & 0.57 & 0.69 & 0.32 \\ 0.2 & 0.6 & 0.43 & 0.73 & 0.54 & 0.52 & 0.44 & 0.36 & 0.44 & 0.45 \\ 0.3 & 0.53 & 0.68 & 0.65 & 0.67 & 0.53 & 0.56 & 0.35 & 0.66 & 0.49 \\ 0.6 & 0.5 & 0.56 & 0.44 & 0.69 & 0.48 & 0.59 & 0.6 & 0.64 & 0.44 \\ 0.54 & 0.6 & 0.44 & 0.46 & 0.52 & 0.73 & 0.39 & 0.23 & 0.52 & 0.53 \end{bmatrix}$$

2.4 Optimization

Optimization involves finding the maximum or minimum value of a function, $f(\boldsymbol{x})$, by changing the value of $\boldsymbol{x}$. Minimization is the most commonly used method and maximization can be achieved by minimizing the negative of the function, i.e. $-f(\boldsymbol{x})$. The function being optimized is referred to as the objective, cost, loss, or error function. The first derivative, $f'(x)$, of a function, $y = f(x)$, where x and y are real numbers, represents the slope of the function at a specific point, x. In other words, it specifies how to scale a small change in the input in order to obtain the corresponding change in the output: $f(x+\epsilon) \approx f(x) + \epsilon f'(x)$. The derivative is therefore useful for minimizing a function because it tells us how to change x in order to make a small improvement in y. For example, we know that $f(x - \epsilon sign(f'(x)))$ is less than $f(x)$ for small enough ϵ. We can thus reduce $f(x)$ by moving x in small steps with the opposite sign of the derivative. This technique is called gradient descent.

When $f'(x) = 0$, the derivative provides no information about which direction to move therefore such points where this happens are called critical points or stationary points. Points where $f(x)$ is lower than all neighboring points are called local minima, consequently, it is not possible to decrease the function by making infinitesimal steps. A local maximum is a point where $f(x)$ is higher than at all neighboring points, so it is not possible to increase $f(x)$ by making infinitesimal steps. Some critical points are neither maxima nor minima. These are known as saddle points. The lowest value of $f(x)$ is called a global minimum. There can be one or many global minima for a function. Local minima may not be globally optimal. In DL, optimizing functions with multiple local minima and saddle points in flat regions is challenging (i.e. the one in Figure 2.5), especially in high-dimensional input. Hence, often a low value of f is sought rather than the absolute minimum.

FIGURE 2.5: Optimization algorithms may fail to find a global minimum when there are multiple local minima or plateaus present. In the context of DL, we generally accept such solutions even though they are not truly minimal, as long as they correspond to significantly low values of the cost function.



To solve this issue, gradient descent looks for the best solution (rather the most acceptable, it is not in fact the true best solution) by proposing an update rule

$$x' = x - \eta \nabla_x f(x) \tag{2.1}$$

where $\nabla_x f(x)$ is the vector containing all the partial derivatives, the gradient of f, and η is the learning rate, a positive scalar determining the size of the step (as we often minimize functions with multiple inputs, we have to use the concept of partial derivatives). There are several ways to initialize the step size, a popular approach is to set it to a small constant but there are approaches that evaluate different values of η and choose the one that results in the smallest loss function value. The step size must be sufficiently small to prevent surpassing the minimum and ascending the loss landscape in areas with prominent positive curvature. Often, this implies that the step size is inadequate for making substantial advancements in other directions with lower curvature. Addressing this problem can be achieved by utilizing data from the Hessian matrix to direct the search. In the following section, we will discover that second-order techniques are crucial for determining the appropriate step size, as the second derivative provides insight into the expected performance of a gradient descent step.

2.5 Beyond Gradient Descent

In multiple dimensions, there is a different second derivative for each direction at a single point. Therefore, it can happen that, in one direction the derivative rapidly increases, while in another direction, it increases slowly. Gradient descent is unaware of this change in the derivative that multiple dimensions can cause during training, this makes also difficult to choose a good step size and can result in overshooting the minimum and therefore never converging. This issue can be solved by using information from the second-order derivatives to guide the search. In DL, second-order derivatives play a crucial role in optimization algorithms and can be collected together into a matrix called the Hessian matrix which can be considered the Jacobian of the gradient (The Jacobian is the matrix composed of all the partial derivatives of a multivariable function). These algorithms use the second-order information of the loss function, which is the function that measures the difference between the model's predicted output and the true output, to update the model's parameters. This allows for more efficient optimization, as the second-order information provides a better approximation of the local curvature of the loss function, compared to using only first-order derivatives. The Hessian matrix is a square matrix of second-order partial derivatives of a scalar-valued function, and it provides information about the curvature of a function at a specific point. The eigenvalues of the Hessian matrix can be used to determine whether the point is a local maximum, minimum, or saddle point. In DL, the Hessian matrix is typically computed for the parameters of the model, rather than for the input data. One of the key advantages of using second-order derivatives is that they can help to escape from poor local minima as second-order information can provide more insight into the loss landscape, and can help the optimization algorithm to find a better solution. Another advantage of using second-order derivatives in DL is that they can help to improve the stability of the optimization process. This is because second-order information can be used to adjust the step size of the optimization algorithm, which can help to prevent overshooting or undershooting the optimal solution. In addition, the Hessian can be used to analyze the generalization properties of NNs. The eigenvalues of the Hessian can indicate whether a network is overfitting or underfitting, and the distribution of eigenvalues can provide insight into the expressivity of the network. Researchers have also used the Hessian to identify areas of the network that are sensitive to small changes in the input, which can lead to new methods for interpretability and robustness. In this work, we explore the potential of second-order methods in DL, focusing on weight initialization and optimization, and showing how Hessian information can improve upon previous research on these two crucial tasks.

Example Jacobian

The formula for the Jacobian matrix is the following

$$f(x_1, x_2, \dots, x_n) = (f_1, f_2, \dots, f_m)$$

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

The Jacobian matrix is very expensive to calculate, its dimensions are determined by the number of output variables m and the number of input variables n.
Now, if we want to find the Jacobian matrix at the point (1,2) of the following function:

$$f(x, y) = (x^4 + 3y^2 x, 5y^2 - 2xy + 1)$$

We calculate all the first-order partial derivatives of the function:

$$\frac{\partial f_1}{\partial x} = 4x^3 + 3y^2$$

$$\frac{\partial f_1}{\partial y} = 6yx$$

$$\frac{\partial f_2}{\partial x} = -2y$$

$$\frac{\partial f_2}{\partial y} = 10y + 2x$$

Now we apply the formula for the Jacobian:

$$J_f = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} 4x^3 + 3y^2 & 6yx \\ -2y & 10y + 2x \end{bmatrix}$$

Once we have found the expression, we evaluated it at point (1,2):

$$J_f(1,2) = \begin{bmatrix} 4 \cdot 1^3 + 3 \cdot 2^2 & 6 \cdot 2 \cdot 1 \\ -2 \cdot 2 & 10 \cdot 2 - 2 \cdot 1 \end{bmatrix} = \begin{bmatrix} 16 & 12 \\ -4 & 1018 \end{bmatrix}$$

Example Hessian

The formula for the Hessian matrix is the following

$$\boldsymbol{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial^2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial^2 x_n} \end{bmatrix}$$

The Hessian matrix will always be a square matrix, with dimensions equal to the number of variables of the function. Therefore, its computation is generally more expensive than that of the Jacobian matrix, as the number of elements in the Hessian grows quadratically with the number of variables. If we want to find the Hessian matrix at the point (1,0) of the following function:

$$f(x,y) = (y^4 + x^3 + 3x^2 + 4y^2 - 4xy - 5y + 8)$$

We calculate all the first-order partial derivatives of the function:

$$\frac{\partial f}{\partial x} = 3x^2 + 6x - 4y$$

$$\frac{\partial f}{\partial y} = 4y^3 + 8y - 4x - 5$$

Once we have obtained the first derivatives, we need to calculate all the second-order partial d derivatives of the function:

$$\frac{\partial^2 f}{\partial^2 x} = 6x + 6$$

$$\frac{\partial^2 f}{\partial^2 y} = 12y^2 + 8$$

$$\frac{\partial^2 f}{\partial x \partial y} = -4$$

Now we are able to find the Hessian matrix using the formula:

$$\boldsymbol{H}_f(x,y) = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial^2 y} \end{bmatrix} = \begin{bmatrix} 6x+6 & -4 \\ -4 & 12y^2+8 \end{bmatrix}$$

So the Hessian matrix evaluated at the point (1,0) is:

$$\boldsymbol{H}_f(1,0) = \begin{bmatrix} 6 \cdot 1 + 6 & -4 \\ -4 & 8 \end{bmatrix} = \begin{bmatrix} 12 & -4 \\ -4 & 8 \end{bmatrix}$$

Chapter 3

Background and Related Work

In this chapter, we are going to provide the technical background and overview of the related work on second-order methods for DL.

The optimization algorithms commonly used in DL are versatile but lack guarantees due to the complexity of functions used, unlike other fields where optimization algorithms are tailored to specific (limited) function families. In this section, while discussing the most common practices and the previous works, we are going to explain the source of this deficiency.

Initialization Based on Variance Flow Analysis. Glorot and Bengio, 2010 proposed a framework for balancing the trade-off between representational power and generalization in NNs. Representational power refers to the ability of a NN to accurately model the input data, while generalization refers to the ability of the network to make accurate predictions on unseen data. The proposed framework aims to balance this trade-off by estimating the variance at different layers of the NN. By estimating the variance at different layers, the framework can identify where the network has the most representational power and where it is overfitting the training data. The approach is based on the assumption that the activation functions used in the network, which introduce non-linearity, approximately behave like the identity function around zero i.e., $f(u) \approx u$ for small u. This means that the activation functions do not have a significant impact on the variance of the network's outputs when the inputs are close to zero. By using this assumption, the framework can identify the layer where the activation function starts to have a significant impact on the variance of the network's outputs and use this information to balance the trade-off between representational power and generalization. This approach has been generalized to other nearly linear functions in follow-up works (He et al., 2015; Xu, Huang, and Li, 2016). This means that the framework can be applied to a wider range of activation functions and has expanded the applicability of the framework to a wider range of NN architectures. By linearization, we then obtain:

$$z_i^{(k+1)} \approx \sum_{j=1\ldots d_k} \boldsymbol{W}_{i,j}^{(k)} \cdot z_j^{(k)} + \boldsymbol{b}_i^k \tag{3.1}$$

In the forward pass, we require $\mathbf{Var}[z^{(k+1)}] \approx \mathbf{Var}[z^{(k)}]$ to maintain the magnitude of inputs until the last layer. In the backward pass, we compute the gradients by recursively applying the chain rule

$$\partial_{z_i^{(k)}} L = \sum_{j=1\ldots d_{k+1}} \partial_{z_j^{(k+1)}} L \cdot \partial_{z_i^{(k)}} z_j^{(k+1)} \tag{3.2}$$

$$\approx \sum_{j=1\ldots d_{k+1}} \partial_{z_j^{(k+1)}} L \cdot \boldsymbol{W}_{j,i}^{(k)} \tag{3.3}$$

while we aim to keep their magnitude, i.e., $\mathbf{Var}\left[\partial_{\mathbf{z}^{(k-1)}} L\right] \approx \mathbf{Var}\left[\partial_{\mathbf{z}^{(k)}} L\right]$.

Looking at Equations 3.1 and 3.2, we see that the weights $\boldsymbol{W}^{(k)}$ interact with the previous layer during the forward pass and with the following layer during the backward pass. The first action is multiplying along the input dimension d_k, while the second action is multiplying along the output dimension d_{k+1}. One can prove that, in general, taking the dot-product with an *independently* centered random matrix along dimension d scales the variance by the factor d (Glorot and Bengio, 2010; Xu, Huang, and Li, 2016). Thus, to balance the two actions during the forward and backward pass, one usually chooses the $\boldsymbol{W}^{(k)}$ as i.i.d. samples from a normal distribution $N(\mu, \sigma^2)$ with mean $\mu = 0$ and standard deviation

$$\sigma[\boldsymbol{W}^{(k)}] = \sqrt{\frac{2}{d_k + d_{k+1}}}. \tag{3.4}$$

Variance-based initialization schemes (Arpit, Campos, and Bengio, 2019; Glorot and Bengio, 2010; He et al., 2015; Hendrycks and Gimpel, 2016; Xu, Huang, and Li, 2016), however, implicitly assume *independence* of weight gradients across layers. This is *not true already in first pass*, since back-propagated gradients depend on weights used during the forward pass and also on the input data. More specifically, during the forward pass, the input data is transformed by the network's weights, which produces a prediction. Then, during the backpropagation step, the gradient of the loss function with respect to the weights is computed and used to update the weights. However, this gradient depends not only on the current weights but also on the input data that was used during the forward pass. Therefore, the gradient of the loss function with respect to the weights is not entirely independent across layers. This means that variance-based initialization schemes may not provide optimal initializations, especially for DNN with many layers.

Example dependency of weight gradients across layers

Consider a regression setting with two layers and a linear activation function, such that

$$L = (z - t)^2, z = \boldsymbol{W}_2 \boldsymbol{W}_1 \boldsymbol{x}.$$

Note that

$$\partial_z L = 2(z - t) = -2(\boldsymbol{W}_2 \boldsymbol{W}_1 - t)$$

depends on both $\boldsymbol{W}_2$ and $\boldsymbol{W}_1$. To see correlations with the input vector, consider a one-dimensional regression

$$L = (z - t)^2,$$

where

$$z = \boldsymbol{W}\boldsymbol{x}.$$

From Equation 5 in Glorot and Bengio, 2010, we should have

$$\mathbf{Var}[\partial_{\boldsymbol{W}} L] = \mathbf{Var}[\partial_{\boldsymbol{W}} z] \cdot \mathbf{Var}[\partial_z L]$$

for $\boldsymbol{W}$ with unit variance, but this gives

$$\mathbf{Var}[2(\boldsymbol{W}\boldsymbol{x} - t)\boldsymbol{x}] = \mathbf{Var}[\boldsymbol{x}] \cdot \mathbf{Var}[2(\boldsymbol{W}\boldsymbol{x} - t)].$$

Not only two sides can be a factor away but also the target t can be correlated to the input $\boldsymbol{x}$.

In addition to this lack of correlations, the above kinds of variance analysis do not reveal any correlations between different variables, which could provide important insights into how the optimization is behaving. Additionally, the methods only provide qualitative insights, meaning that they do not directly connect the variance estimation to the underlying optimization problem. Therefore, it is not possible to get more quantitative insights, such as estimating the step size, from these first-order methods. Estimating the step size would be useful because it would provide more precise control over the optimization process. This could help to improve the stability and efficiency of the optimization algorithm, but first-order methods are not able to provide this information.

Other First-Order Approximations. To address the aforementioned lack of guarantees in DL, we sometimes restrict ourselves to functions that are either *Lipschitz* continuous or have *Lipschitz* continuous derivatives. The stability of the linearization in 3.1 has also been studied using the *Lipschitz property* as the sensitivity metric (Szegedy et al., 2013; Virmaux and Scaman, 2018) as well as *mean-field theory* to investigate the dynamic of the forward-pass for very deep networks in Xiao et al., 2018. This property is useful because it allows quantifying the gradient descent assumption that a small change in the input will have a small change in the output. The *Lipschitz continuity* is a fairly weak constraint and many optimization problems in DL can be made *Lipschitz* continuous without major modifications.

Some generalizations to the classical schemes discussed above have been also proposed. For example Balduzzi et al., 2017 generalizes the variance analyses to

ResNets, while Bachlechner et al., 2020 introduces extra parameters at the activation layers.

Adaptive First Order Methods. Adaptive first-order methods are a class of optimization techniques that are commonly used to train DL models. These methods adapt the learning rate during the training process to improve the convergence of the model. Adaptive first-order methods have been shown to be computationally efficient and have been used successfully in many DL tasks. These methods can improve the performance of the model and can also help to prevent the model from getting stuck in suboptimal solutions. There are multiple variations, but these methods can be represented using the following general update formula:

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta_t \frac{\boldsymbol{m}_t}{\boldsymbol{v}_t} \tag{3.5}$$

where η_t is the learning rate, and $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$ denote the so-called first and second moment terms, respectively. AdaGrad, RMSProp, and Adam are very popular adaptive learning rate optimization algorithms commonly used in training DNNs. The AdaGrad algorithm (Duchi, Hazan, and Singer, 2011) adjusts the learning rate for each parameter based on the historical gradient of the loss. The parameters with larger gradient values have a larger decrease in learning rate, whereas the parameters with smaller gradient values have a smaller decrease in learning rate. This results in faster convergence in the gentler slopes of the parameter space. However, when used to train DNNs, AdaGrad can suffer from a premature decrease in learning rate due to the accumulation of squared gradients from the beginning of training.

The RMSProp algorithm (Tieleman and Hinton, 2012) improves upon AdaGrad by replacing the accumulation of squared gradients with an exponential weighted moving average. This allows RMSProp to converge more rapidly after finding a convex structure in the non-convex optimization problem of training NNs. The moving average introduces a new hyperparameter, ρ, which controls the length scale of the average. Empirically, RMSProp has proven to be an effective optimization algorithm for DNNs.

Adam, short for Adaptive Moment Estimation (Kingma and Ba, 2015), is a variant of RMSProp and momentum, with some important differences. Unlike RMSProp, Adam incorporates momentum directly into its first-order moment estimate (with exponential weighting) of the gradient. Adam also includes bias corrections to its estimates of both the first-order and second-order moments, while RMSProp only includes an estimate of the second-order moment without a correction factor. Adam is generally considered to be robust to hyperparameter choices, although the learning rate may need to be adjusted from the default value.

In summary, AdaGrad, RMSProp, and Adam are three popular optimization algorithms used in DL, with each algorithm offering its own strengths and weaknesses. The choice of algorithm will depend on the specific task at hand and the desired trade-off between convergence speed and optimization performance.

Despite all these attempts, learning non-use-case specific models has been shown to be a challenging task as it is still not clear which optimizer should work for a new task as even the choice of the optimizer has effectively become a hyperparameter. To address this challenge, many attempts have been made to understand the source of the use-case specificity that distinguishes DL problems. Researchers have sought to identify the factors that make certain problems more challenging than others and to develop methods for improving the generalization performance of DL models. To this date, second-order optimization methods have been partially shown to be effective in some cases but have not been sufficiently investigated in the context of learning and optimization.

Hessian Approaches to Neural Nets. Considering the Hessian is ubiquitous in the field of optimization (Salvatier, Wiecki, and Fonnesbeck, 2016), and has been also applied in the context of DNNs (Dauphin and Schoenholz, 2019; Ghorbani, Krishnan, and Xiao, 2019; Sagun et al., 2017). Since the exact calculation of the Hessian is not feasible for larger networks (which is quadratic in the total, usually already large, number of parameters), efficient approximation schemes have been proposed.

Newton's method (Nocedal and Wright, 2006) is the most widely used second-order optimization technique. It is based on second-order Taylor series expansion to improve an objective function. This involves two steps: updating the inverse of the Hessian and adjusting the network parameters. However, Newton's method only works well when the Hessian is positive definite, and is not suitable for non-convex objective functions in DL, as they often have challenging features such as saddle points. Additionally, training large NNs using Newton's method is computationally demanding, limiting its practical use to smaller networks. The Broyden–Fletcher–Goldfarb–Shanno (BFGS) (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970) algorithm attempts to bring some of the advantages of Newton's method without the computational burden. However, also this method has to store the inverse of the Hessian matrix, making it impractical for most modern DL models. A memory-cost-efficient version of BFGS (the Limited Memory BFGS or L-BFGS Liu and Nocedal, 1989) that avoids storing the complete inverse Hessian exists. Like BFGS, L-BFGS uses an estimate of the inverse of the Hessian matrix, but where BFGS stores a dense nxn approximation to the inverse Hessian (where n is the number of variables in the problem), L-BFGS stores only a few vectors that represent the approximation implicitly.

Block-diagonal approximations of the Hessian tensor were considered in TONGA (Le Roux et al., 2008 and Martens and Grosse, 2015), but were not extensively evaluated (very small networks, not-real world, visual not a quantitative comparison).

Some Hessian characteristics such as the eigenvalues or the trace (Ghorbani, Krishnan, and Xiao, 2019, Jacot, Gabriel, and Hongler, 2020), have been proposed. However, these are simplified and do not give insights into the Hessian tensor structure (such as extracting the diagonal or the block). Furthermore, these approximations are probabilistic with no clear guarantee on the error. Finally, Yao et al., 2021 proposed a new approach for approximating the calculation of the diagonal of the Hessian matrix. The approach utilizes techniques from random linear algebra, which are a set of mathematical techniques used to solve problems involving large, non-convex, and high-dimensional data sets. Despite the use of this new approach, the results obtained were not found to be any better than those obtained using previous state-of-the-art methods but just comparable. This suggests that while the proposed approach may have some potential, it may not be as effective as other existing methods for approximating the diagonal of the Hessian matrix.

Chapter 4

Approximated Chain Rule for Hessian Backpropagation

The chain rule is a fundamental concept in calculus that is used to calculate the derivative of composite functions. It is a powerful tool for optimizing complex functions and is essential for training ML models. In ML, it is used to calculate the gradient of a complex function, which is used in the optimization process to update the model's parameters. In this context, the chain rule is often referred to as the backpropagation algorithm. The backpropagation algorithm is used to calculate the gradient of the loss function with respect to the parameters. It starts by calculating the derivative of the loss function with respect to the model's output. Then, it calculates the derivative of the model's output with respect to the model's parameters. The final step is to multiply the two derivatives to get the gradient of the loss function with respect to the parameters.

$$D_{\mathsf{W}} L(\boldsymbol{z}, t) \;=\; D_{\mathsf{W}} \boldsymbol{z} \bullet D_{\boldsymbol{z}} L(\boldsymbol{z}, t) \tag{4.1}$$

First-order methods have been successfully utilized to optimize NNs for years now. This method is relatively simple to implement, but it has some limitations. One of the main limitations is that it only takes into account the slope of the function, without considering the entire loss landscape of the optimization problem. This means that first-order methods may converge to a suboptimal solution. In this work of thesis, we are going to present a new second-order-based approximated chain rule, a more powerful tool for calculating the gradient of a multi-variable function. This method takes into account the entire landscape of the loss function and therefore is more accurate in determining the direction of the steepest increase. Additionally, the second-order information is more robust to the presence of local minima and maxima, as it can distinguish between them and the global optimum.

We are going to show two applications of our Hessian approximated chain rule: (i) weight initialization and (ii) optimizer's update rule.

Before presenting our new chain rule, we need to introduce some more notation. Let $\boldsymbol{z}^{(k)}$ be the input of the k-th layer. Let $\mathsf{A}^{(k)} = D_{u^{(k)}} \boldsymbol{z}^{(k+1)}$ be the derivative of the forward activation at the k-th layer, with respect to the output before activation $u^{(k)} = \mathsf{W}^{(k)} \cdot \boldsymbol{z}^{(k)} + \boldsymbol{b}^{(k)}$. Let $\mathsf{B}^{k+1} = D_{\boldsymbol{z}^{(k+1)}} \boldsymbol{z}^{(n)}$ be the output derivative backpropagated to the input of the $(k+1)$-th layer. Let $\mathbf{H}_{\boldsymbol{z}} = D^2_{\boldsymbol{z}} L(\boldsymbol{z}, t)$ be the loss Hessian with respect to the predicted value $\boldsymbol{z}$. Finally, let $\mathbf{H}_{\mathsf{W}} = D^2_{\mathsf{W}} L(\boldsymbol{z}^{(n)}, t)$ be the loss Hessian with respect to the weights W.

The Hessian of a loss function is a matrix that describes the curvature of a function over its domain. In general, the Hessian of a simple loss function has nice properties, such as being positive definite, which makes it easy to optimize. However, when a neural network is used to reparameterize the problem, the dependency of the output z on the weights W becomes more complex, which can make the Hessian difficult to analyze. One important question that arises when using NNs is how the dependency of the network output on the weights affects the loss curvature. This is because the Hessian of the loss function can change significantly depending on the specific architecture and parameters of the NN. For example, if the network is over-parameterized, the Hessian can be ill-conditioned, which can make optimization difficult. On the other hand, if the network is under-parameterized, the Hessian can be too simple, which can make optimization easy but can lead to overfitting. Therefore, when using NNs, it is important to carefully analyze the properties of the Hessian and how they are affected by the specific architecture and parameters of the network. This can help to choose an appropriate optimization algorithm, prevent overfitting, and ensure that the network is able to generalize well to unseen data.

We introduce a novel approximation of the chain rule for backpropagation, which uses second-order derivatives to enhance the efficiency and performance of gradient-based optimization in DL.

$$\underbrace{D^2_{\mathrm{W}}L(z,t)}_{\textbf{reparameterized Hessian}} = D^2_z L(z,t) \underbrace{\bullet D_{\mathrm{W}}z \bullet D_{\mathrm{W}}z}_{\textbf{linearization effect}} + D_z L(z,t) \underbrace{\bullet D^2_{\mathrm{W}}z}_{\textbf{curvature effect}} \tag{4.2}$$

where bullets denote tensor dot-products along the appropriate dimensions. This is more subtle than traditional backpropagation of first derivatives because both first- and second-order effects have to be captured.

4.1 Revisited Weight Initialization Scheme

We now present and discuss a new weight-initialization scheme by applying our Hessian chain rule across the (hidden) layers $k = 0 \dots n-1$ of a NN. In general, for training NNs, variants of gradient-descent (as described in Robbins, 1951) are applied in order to update the model parameters W iteratively towards the gradient $g = D_{\mathrm{W}}L$ of the loss function. In order to quantify this decrease, we need first to consider the *second-order* Taylor series *approximation* to the function $f(x)$ around the current point $x^{(0)}$:

$$f(x) \approx f(x^{(0)}) + (x - x^{(0)})^T g + \frac{1}{2}(x - x^{(0)})^T \mathbf{H}(x - x^{(0)}).$$

Substituting this into our approximation, we obtain,

$$L(\mathrm{W} - \gamma g) \approx L(\mathrm{W}) - \gamma g^T \cdot g + \frac{\gamma^2}{2} g^T \cdot \mathbf{H} \cdot g$$

where $\cdot$ stands for the matrix (or more generally tensor) dot product. There are three terms here: the original value of the loss of the weights, the expected improvement due to the gradient, and the correction we must apply to account for the Hessian. The maximal step size γ^* guarantees that the decrease equals $\gamma^* = \|\mathbf{H}\|^{-1}$ (Goodfellow, Bengio, and Courville, 2016) where $\|\mathbf{H}\|$ is the Hessian norm, i.e., its maximal eigenvalue. In other words, if we want to train with a constant step size, then we need to control the norm of the Hessian. We therefore propose the following paradigm:

> *Good weight initialization controls the Hessian:* we initialize the weights $\mathsf{W}^{(k)}$ such that $\|\mathbf{H}_{\mathsf{W}^{(k)}}\| \approx 1$.

Moreover, we only make the following mild assumption about the loss functions, which requires the activation function to be *nearly linear around 0*:

> *Admissible activation functions:* the loss function must satisfy $f(0) = 0$ and $f''(0) = 0$. We note that this condition holds for all standard activation functions, such as linear, sigmoid, tanh or ReLU.

Finally, our techniques aim to approximate the global curvature of weights up to leading terms. These approximations are accurate under the following mild assumption:

> *Relatively small inputs:* we have $\|z^{(k)}\| \leqslant c$ for all layers k, for a small constant c (e.g., $c = 0.5$).

Note that the latter is necessary to ensure the stability of the forward pass and is implicitly assumed so also in the variance flow analyses.

From 4.2 we derive,

Theorem 1 (Approximated Hessian chain rule for neural networks)**.** *With notation as above, the loss Hessian $\mathbf{H}_{\mathsf{W}^{(k)}}$ with respect to the weights $\mathsf{W}^{(k)}$ satisfies*

$$\mathbf{H}_{\mathsf{W}^{(k)}}[\mathbf{g},\mathbf{g}] \approx \mathbf{v}^T \cdot \mathbf{H}_{\mathbf{z}} \cdot \mathbf{v} \quad \text{with} \quad \mathbf{v} = \mathsf{B}^{(k)} \cdot \mathsf{A}^{(k)} \cdot \mathbf{g} \cdot \mathsf{F}^{(k)} \tag{4.3}$$

where products are standard matrix products. More precisely, the approximation holds up to a third-order *error term $\sim f'''c^3 \cdot \|\mathbf{g}\|^2$ where f''' is the bound on the third derivative of the activation functions and c is the bound on the inputs $\mathbf{x}$. The leading term then is of order $\sim c^2 \cdot \|\mathbf{g}\|^2$.*

This theorem in its essence states that the only Hessian that is needed is the Hessian of the output layer, which is $D^2_z L(z,t)$ in our approximated chain rule 4.2.

We observe the following important properties.

Remark 1 (Beyond block-diagonal Hessian)**.** *The approximation above computes the main blocks of the Hessian, namely Hessians with respect to each layer. In practice, the cross-layer components are of smaller order and can be omitted (Botev, Ritter, and Barber, 2017). However, our approximation extends to this case by using two different* **v***'s instead of one.*

Remark 2 (Low computational complexity)**.** *Computing the Hessian approximation is of cost comparable to backpropagation. The only Hessian we need is the final loss/output Hessian, which is usually small (K^2 for the classification of K classes).*

Remark 3 (Beyond MLP model)**.** *We formulated the result for densely-connected networks but the approximation holds in general with $\mathbf{v} = D_{\mathsf{W}^{(k)}}\mathbf{z}^{(n)} \bullet \mathbf{g}$ (as we will see in empirical evaluation).*

Remark 4 (Perfect approximation for ReLU networks)**.** *We have exact equality for activations with $f'' = 0$ such as variants of ReLU (see Section 4.3.3).*

Remark 5 (Good approximation up to leading terms)**.** *Regardless of the activation function, the error term is of smaller order (under our assumption of bounded inputs).*

Remark 1 involves a hidden constant which we demonstrate to be small in practice (see our experiments in section 7.5).

4.1.1 Approximation via Jacobian Products

From the previous subsection, we are left with the linearization effect of the chain rule. The linearization effect refers to the ability to approximate a non-linear function using a linear function, which can make the calculation of derivatives simpler. By factorizing the linearization effect of the chain rule, we can further simplify the problem of controlling the *products of the hidden layers' Jacobians*. This means that by breaking down the chain rule into simpler components, it becomes easier to understand and control the impact of the hidden layers on the overall function.

Theorem 2 (Hessian factorized into Jacobians)**.** *Up to third-order terms in $z^{(i)}$, we can factorize v from Theorem 1 into*

$$\mathbf{v} \approx \mathsf{J}^{(n-1)} \cdot \ldots \mathsf{J}^{(k+1)} \cdot \mathsf{A} \cdot \mathbf{g} \cdot \mathsf{J}^{(k-1)} \cdot \ldots \mathsf{J}^{(0)} \cdot \mathbf{z}^{(0)} \tag{4.4}$$

where $\mathbf{J}^k = D_{z^{(k)}} z^{(k+1)}$ is the derivative of the output with respect to the input at the k-th layer. In particular, the Hessian's dominant eigenvalue scales by a factor of at most $\|v\|^2$, where it holds that:

$$\|\mathbf{v}\| \leqslant \|\underbrace{\mathsf{J}^{(n-1)} \cdot \ldots \cdot \mathsf{J}^{(k+1)}}_{\textit{backward product}}\| \cdot \|\mathsf{A}\| \cdot \|\underbrace{\mathsf{J}^{(k-1)} \cdot \ldots \cdot \mathsf{J}^{(0)}}_{\textit{forward product}}\| \cdot \|\mathbf{z}^{(0)}\| \tag{4.5}$$

The norm of the matrix product $\|\mathsf{J}_k \ldots \mathsf{J}_1\|$ then is computed as the maximum of the vector norm $\|\mathsf{J}_k \cdots \mathsf{J}_1 \cdot v\|$ over vectors v with unit norm. Given this result, a good weight initialization thus aims to make the backward and forward products having a norm close to 1.

Remark 6 (Connection to products of random matrices)**.** *Note that our problem closely resembles the problem of* random matrix products *(Kargin et al., 2010). This is because Jacobians for smooth activation functions are simply random-weight matrices.*

Remark 7 (Connection to spectral norms)**.** *Further, it is possible to estimate the product of random matrices by the product of their spectral norms. In particular, the spectral norm of a random $m \times n$ matrix with zero-mean and unit-variance entries is $\frac{1}{\sqrt{m}+\sqrt{n}}$ on average (Silverstein, 1994). For the Gaussian case, this can be found precisely by Wishart matrices (Edelman, 1988). This however is overly pessimistic for long products, for similar reasons as overestimating of Lipschitz constants (Szegedy et al., 2013; Virmaux and Scaman, 2018).*

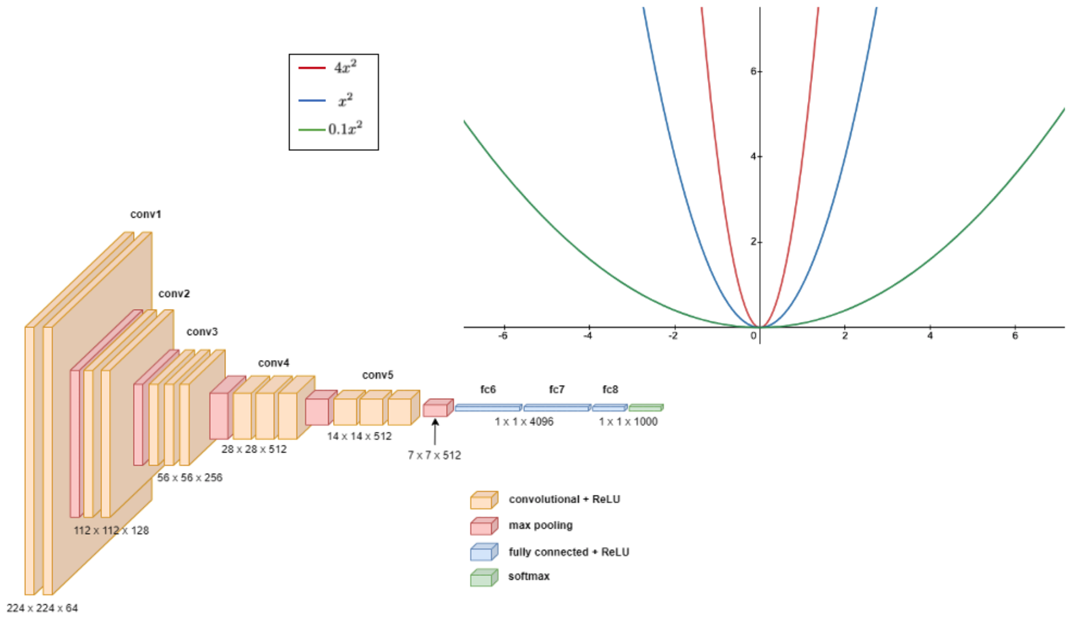
4.2 Second-order Optimizer

The idea behind our second-order optimizer is to enable efficient and systematic training of NNs as learning non-use-case specific models has been shown to be a challenging task in DL and even the choice of the optimizer has become a hyperparameter. For example, SGD with momentum is typically used in Computer Vision; Adam is used for training transformer models for NLP; and Adagrad is used for Recommendation Systems. Many attempts have been made to justify and understand the source of use-case specificity that distinguishes DL problems. To this date, second-order optimization methods have been partially shown to be effective in some cases but have not been sufficiently investigated in the context of learning and optimization.

4.2.1 What is the Problem with SGD?

If we look at what happens when training with SGD, we have a loss function and we are trying to do descent but what actually happens is that it uses the same learning rate across all the layers of a NN. The problem is that not all the layers have the same loss landscape. One way to look at this is if, with the network in figure 4.1, we wanted to fit the family of quadratic functions (let's not forget that in ML the optimization algorithm tries to fit in the best possible way a family of functions, this problem has been formalized as a problem of minimum. For this example we have the quadratic family of functions). The last layer looks like $y = x^2$, the 4th layer has a loss landscape of $y = 4x^2$ and the first layer has $y = 0.1x^2$. If we have a flat loss landscape, like in $0.1x^2$, we want to take a bigger step size because the gradient is small and if we use the same step size when we have a sharper loss landscape, as in $4x^2$, this is going to be suboptimal. SGD does not see this in fact, at the origin, the first derivative of $y = x^2$, $y = 4x^2$, $y = 0.1x^2$ is all the same: 0. So the concept behind our second-order optimizer is to have a learning process that adapts depending on the loss landscape of every layer.

FIGURE 4.1: Different functions of the same family across NN layers



Let's recall the general update formula,

$$W_{t+1} = W_t - \eta \Delta W_t \tag{4.6}$$

we can read this formula as the next parameter is equal to the old parameter minus the learning rate times the direction. This direction for first-order methods is the gradient. What second-order methods do, is to normalize these curvature differences through the Hessian information.

4.2.2 AdaPrHess: Adaptive Approximated Hessian

Adam is an attempt to fix SGD's problems by developing an Adaptive Moment Estimation optimizer. Adam combines the ideas of momentum and adaptive learning rate to improve the stability and speed of convergence of the NNs. If momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction which prefers flat minima in the error surface. Adaptive learning rates, on the other hand, help the algorithm to adjust the step size automatically based on the characteristics of the error surface. Adam's update formula is

$$\mathsf{W}_{t+1} = \mathsf{W}_t - \eta_t \frac{\boldsymbol{m}_t}{\boldsymbol{v}_t} \tag{4.7}$$

where $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$ are decaying averages of past and past squared gradients

$$\boldsymbol{m}_t = \frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} \boldsymbol{g}_i}{(1-\beta_1^t)}$$

$$\boldsymbol{v}_t = \sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t} \beta_1^{t-i} \boldsymbol{g}_i^2}{(1-\beta_2^t)}}$$

They represent the first moment (the mean) and the second moment (the uncentered variance) of the gradients, hence the name of the method. Even if Adam is a great attempt to address the issue with SGD, the problem remains the same as the nature of the information exploited does not change: first-order derivates.

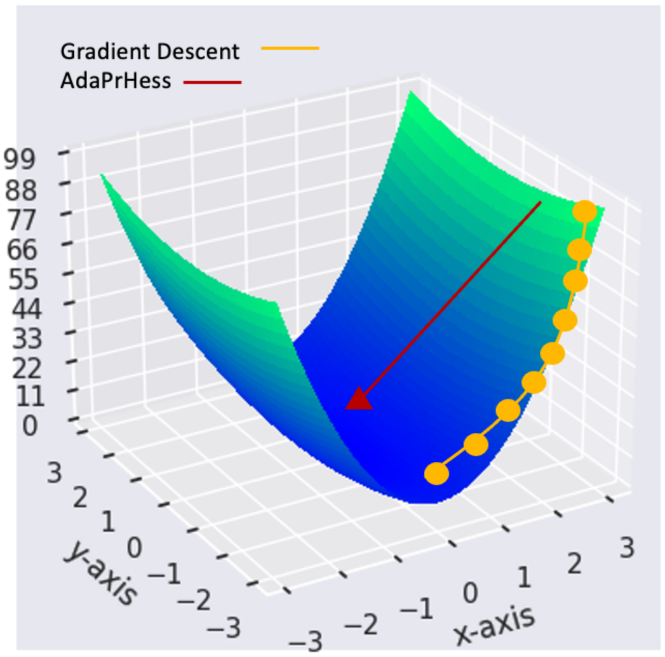
We propose AdaPrHess, an optimizer that uses second-order information in the update rule. We calculate $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$ as follows:

$$\boldsymbol{m}_t = \frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} \boldsymbol{g}_i}{(1-\beta_1^t)}$$

$$\boldsymbol{v}_t = \sqrt{\frac{(1-\beta_2)\sum_{i=1}^{t} \beta_1^{t-i} \boldsymbol{d}_i^2}{(1-\beta_2^t)}}$$

The first momentum is as before, compute the momentum gradient. In the second momentum, the denominator $\boldsymbol{v}_t$, we are using the diagonal of the Hessian instead of the gradient, referred to as $\boldsymbol{d}_i^2$. This small difference has a powerful effect as we dynamically incorporated both kinds of knowledge in order to approximate the loss function: gradient and Hessian. Utilizing this information improves the efficiency of the optimization process and helps converge to a solution more rapidly as the second-order derivatives are used to determine the nature of a critical point, such as whether it is a local maximum, local minimum, or a saddle point (an example of this concept is illustrated in Figure 4.2).

FIGURE 4.2: The trajectory of our AdaPrHess compared with the traditional gradient descent. Gradient descent converges very slowly whereas AdaPrHess converges to the optimum way more quickly because of the directional second-order derivative



4.3 Relationship to Popular Activation Functions

We next discuss the relationship of our Hessian-based weight initialization scheme to a number of previous schemes, namely *smooth activations*, *dropout*, and *ReLU* (Glorot and Bengio, 2010, Hendrycks and Gimpel, 2016 Krizhevsky, Sutskever, and Hinton, 2012).

4.3.1 Smooth Activations

We first formulate the following lemma.

Lemma 1 (Dot-product by random matrices). *Let* **W** *be a random matrix of shape* $[n, m]$*, with zero-mean entries and a variance of* σ^2*. Let* **z**, **z**′ *be independent vectors of shape* $[m]$ *and* $[n]$*, respectively. It then holds that:*

$$\mathbf{E}\|\mathbf{W}\cdot\mathbf{z}\|^2 = n\sigma^2\cdot\mathbf{E}\|\mathbf{z}\|^2 \tag{4.8}$$

$$\mathbf{E}\|\mathbf{z}'\cdot\mathbf{W}\|^2 = m\sigma^2\cdot\mathbf{E}\|\mathbf{z}'\|^2 \tag{4.9}$$

Using this, we can estimate the growth of the Jacobian products in Theorem 2 as follows.

Corollary 1 (Smooth activations). *Consider activation functions such that* $f'(0) = 1$*. Then* $\mathsf{J}^{(k)} \approx \mathbf{W}^{(k)}$ *(up to leading terms) and the norm of the forward product is stable when*

$$\mathbf{Var}[\mathbf{W}^{(k)}] = \frac{1}{d_{k+1}}, \tag{4.10}$$

while the norm of the backward product is stable when

$$\mathbf{Var}[\mathbf{W}^{(k)}] = \frac{1}{d_k}. \tag{4.11}$$

As a compromise, we can choose $\mathbf{Var}[\mathbf{W}^{(k)}] = \frac{2}{d_{k+1}+d_k}$.

Note that we exploit the fact that (up to leading terms) Jacobians of smooth activation functions are independent from any other components.

4.3.2 Dropouts

Dropouts (i.e., inactive neurons) can be described by a *randomized function* f_p which, for a certain dropout probability p, multiplies the input by $B_{1-p} \cdot \frac{1}{1-p}$, where B_{1-p} is a Bernoulli random variable with parameter $1-p$. The Jacobian then is precisely given by:

$$\mathsf{J} = \mathsf{W} = \mathsf{diag}(B_1, \ldots, B_d), \quad B_i \sim \mathsf{Bern}(1-p) \tag{4.12}$$

When multiplying from left or right, this scales the norm square by $(1-p)^{-2} \cdot \mathbf{E}[\mathsf{Bern}(1-p)]^2 = 1-p$. Thus, we obtain the following corollary.

Corollary 2 (Initialization for dropout)**.** *Let $1-p$ be the* keep rate *of a dropout. Let σ^2 be the initialization variance without dropouts, then it should be corrected as:*

$$\sigma' = \sigma / \sqrt{1-p} \tag{4.13}$$

This corresponds to the analysis in Hendrycks and Gimpel, 2016, except that they suggested a different correction factor for the backpropagation phase.

4.3.3 ReLU

Rectified Linear Unit (ReLU) (Krizhevsky, Sutskever, and Hinton, 2012) is a non-linear activation function given by $f(\boldsymbol{u}) = \max(\boldsymbol{u}, 0)$. Consider a layer such that $\boldsymbol{z}' = f(\boldsymbol{u})$, $\boldsymbol{u} = \mathsf{W} \cdot \boldsymbol{z}$, where W is zero-centered with a variance of σ^2 and again of shape $[n, m]$, while $\boldsymbol{z}$ is of shape m. We then have $\mathsf{J} = D_{\boldsymbol{z}} \boldsymbol{z}' = \mathsf{diag}(f'(u)) \cdot \mathsf{W}$.

For the forward product, we therefore consider $\mathsf{J} \cdot \boldsymbol{z} = \mathsf{diag}(f'(\boldsymbol{u})) \cdot \boldsymbol{u}$. This scales the norm of $\boldsymbol{u}$ by $\frac{1}{2}$ when $\boldsymbol{u}$ is symmetric and zero-centered, which is true when also W is symmetric and zero-centered. The norm of z is thus changed by $\frac{n\sigma^2}{2}$.

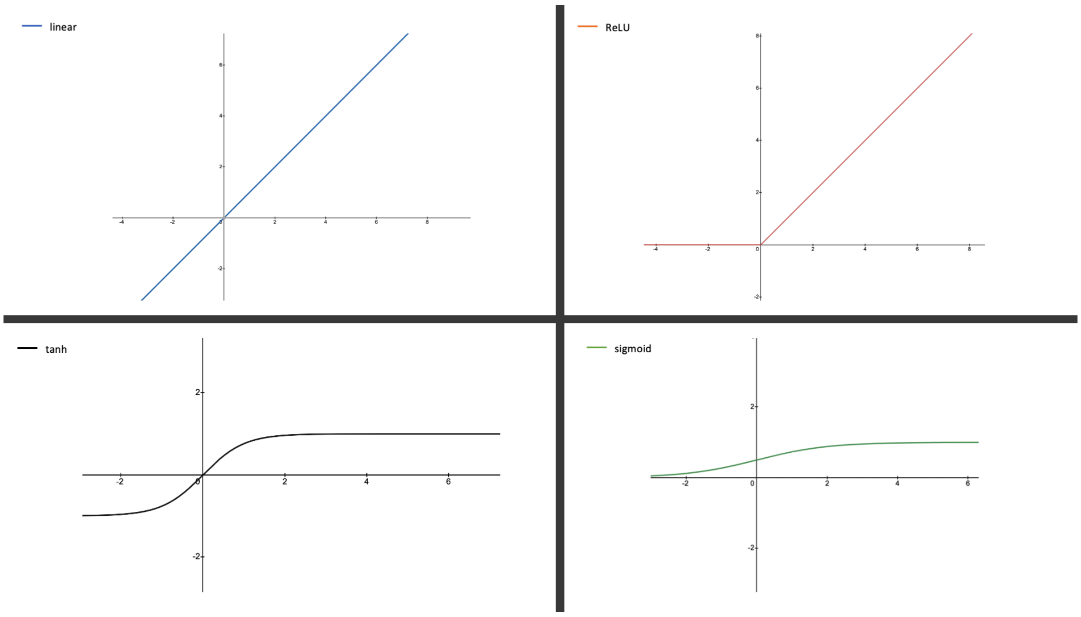
For the backward product, on the other hand, we have to consider $\boldsymbol{v} \cdot \mathsf{J} \cdot \mathsf{J}$, where J is the Jacobian product for the subsequent layers and possibly depends on $\boldsymbol{u}$. However, if the next layer is initialized with i.i.d. samples, the output distribution only depends on the number of active neurons $r = \#\{i : u_i = 1\}$. Conditioned on this information, the following layers are independent from J. Given r, the squared backward product norm thus changes by the factor $r/n \cdot m\sigma^2$. Since $\mathbf{E}[r] = n/2$, the scaling factor is $\frac{m\sigma^2}{2}$.

Corollary 3 (ReLU intialization)**.** *The initialization variance σ^2 in the presence of ReLU should thus be corrected as:*

$$\sigma' = \frac{\sigma}{\sqrt{2}} \tag{4.14}$$

We remark that similar techniques can also be used to derive formulas for weighted ReLU (Arpit, Campos, and Bengio, 2019).

FIGURE 4.3: Behaviour of activation functions around 0



4.4 Experiments

We conducted the following experiments to support our theoretical findings stated in the previous sections. To cover a broad and diverse range of experiments, we trained our models on the MNIST, FashionMNIST, CIFAR10, Google SVHN, and Flowers image datasets and the Tiny Shakespeare dataset.

Implementation. All models were coded in Python using the `Tensorflow 2.4` library (A. et al., 2015) .

Hessian Calculation. The Tensorflow API (in the release 2.4.0, used for the experiments) does not have good support for Hessian calculations. The default implementation under `tf.hessians` is not compatible with fused operations, which are operations that are combined together for more efficient computation. This means that certain loss functions, such as the sparse cross-entropy used in classification, cannot be used as described in GitHub, 2019a. Additionally, the evaluation of Hessian products is not allowed without creating the entire Hessian, which can cause memory issues. This is problematic because creating the entire Hessian requires a lot of memory, especially for large datasets. Batch mode is also not supported, which means that the default implementation does not allow for calculating gradients for multiple samples at once. Parallel computation of components for Jacobians has only recently been added (Agarwal, 2019), but it does not work when combining higher-order derivatives as discussed in the issue GitHub, 2019b. This means that efficient calculation of higher-order derivatives is not possible at the moment. Therefore, when implementing an approximation, a hybrid solution of expressing Hessians as a composition of sequential gradients followed by parallelized computation of the Jacobians is used. This approach allows for a more efficient calculation of Hessians. The fast Hessian-vector algorithm is utilized, this algorithm is an efficient way to compute the action of the Hessian of a scalar-valued function on a given vector. Using this algorithm makes it possible to approximate the Hessian without calculating it explicitly, which can save both time and memory.

4.4.1 Correlations between Loss and Layer Gradients

In the variance analysis one assumes (Glorot and Bengio, 2010; He et al., 2015; Xu, Huang, and Li, 2016; Hendrycks and Gimpel, 2016) that the gradient components of the chain rule in 3.2 are independent, and each has i.i.d. entries. As previously argued, we expect the gradient of the loss with respect to the output $\frac{\partial L}{\partial z^{(n)}}$ and the layer-to-layer gradients $\frac{\partial z^{(i)}}{z^{(i-1)}}$ to be correlated. We therefore prepared an experiment to demonstrate these correlations based on a simple 3-layer NN with Glorot's initialization scheme over the MNIST dataset. We re-ran the initialization a large number of times and used *Pearson's r correlation test* to estimate the dependencies, each using different random seeds. Our findings confirm (i) *very significant correlations* among components of the loss gradient, as well as (ii) *significant correlations* between the loss/output gradient and the output/layer gradients. The experiment is summarized in Table 4.1.

#Samples	**Tested Gradients**	**Dependencies**	**Test Result**
10^4	Loss/Output (diff. components)	10/10 times	avg. p-value $\approx 10^{-5}$ (strong evidence)
10^5	Loss/Output6 vs. Output6/Output5	9/10 times	avg. p-value $\approx 2 \cdot 10^{-2}$ (strong evidence)

TABLE 4.1: Correlations among the components of the backpropagation equation for a simple 3-layer NN. Dependencies tested with *Pearson's r* at 95% significance, using 10 seeds each.

4.4.2 Hessian at Initialization and Convergence

Activation	**Initialization**	**Loss after 2 Epochs**
ReLU	Hessian-driven stddev (1)	0.181294
ReLU	He Normal	0.307550
ReLU	Orthogonal	0.373200
tanh	Glorot	0.356311
tanh	Orthogonal	0.375280

TABLE 4.2: Our initialization approach compared with other methods. The architecture is a 3-layer network trained on MNIST data.

In this experiment, we compare a) the initialization approach using Theorem 1, picking the weights according to the Hessian values, and b) different standard deviations.

The model is a dense 3-layer network on MNIST, with layers of 128, 128 and 10 output units, respectively. We train for two epochs, using SGD with a `learning_rate = 0.1`. Our approach gives much better results for the loss after 2 training epochs, as depicted in Table 4.2.

To make the experiment more challenging, we trained a model with ResNet18 on Cifar-10 with 10 output units on the last dense layer. We train for 10 epochs, using SGD with a `learning_rate = 0.01`, activation function set to SeLU (except for the last dense layer), and initializer set to random normal distribution (with different standard deviations). The estimation of the Hessians is evaluated on 5 convolutional layers and on the last dense layer. It is possible to look at the Hessian values and make the right choice on which standard deviation to use. This is possible because the Hessian carries second-order information that captures the loss landscape's complexity. More detailed results of this setup, including the estimated Hessians and the loss after different steps, are presented in Figure 4.4. The initial Hessian values are evaluated on the normalized gradients, and weights are re-randomized to obtain the density plots. Compared to our approach, other schemes are underestimating the weights. The Hessian values explode or vanish for bad standard deviation

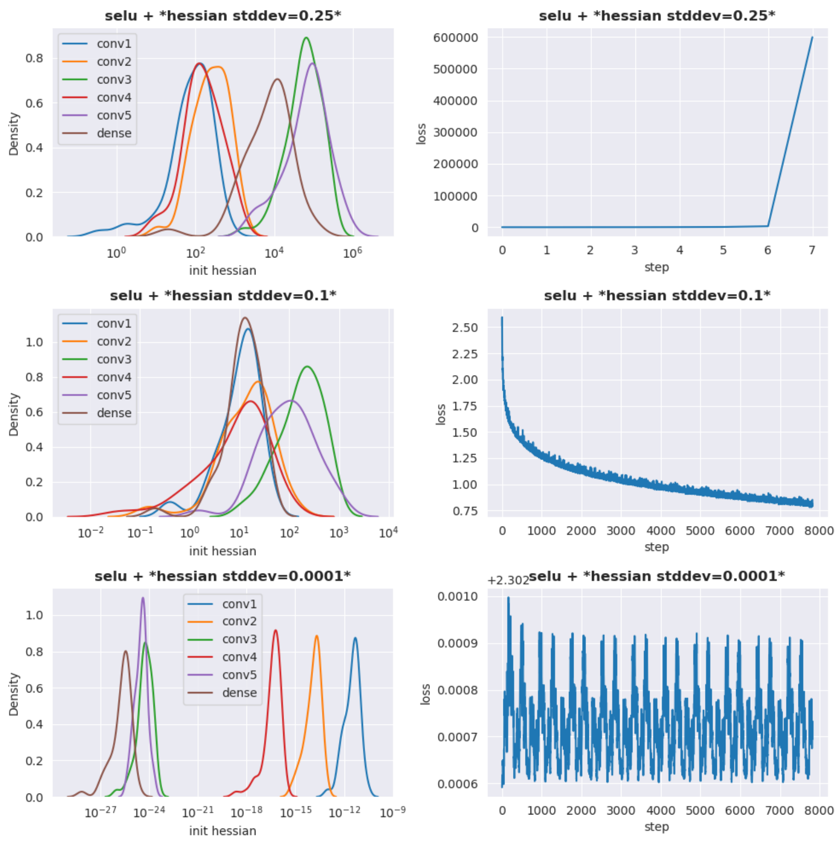


FIGURE 4.4: Hessian at initialization (distributions under initialization randomness) compared with the training loss progress.

whereas for good standard deviations the density plots are almost aligned (concentrated between 10^0 and 10^1). To summarize, our approach gives a diagnostic tool that provides guidance on how to tune the initialization schemes and allows faster convergence.

4.4.3 Good Initializers Nearly Stabilize the Hessian

In this experiment, we demonstrate that under popular initialization schemes, Hessian norms are relatively small at the initialization and during training, and also when estimated on batches. However, often for some of the layers, Hessian norms are *significantly* smaller than 1, which signals that there is still room for improvement (as discussed in the previous section) for these schemes. To illustrate this claim, we use the LeNet5 architecture on the CIFAR10 dataset (depicted in Figure 4.5) and EfficientNet architecture for other datasets: FashionMNIST (Xiao, Rasul, and Vollgraf, 2017) and SVHN (all shown in Figure 4.6), using tanh as activation functions and Glorot initialization scheme. The reported Hessians' values are evaluated on normalized gradients, as before.

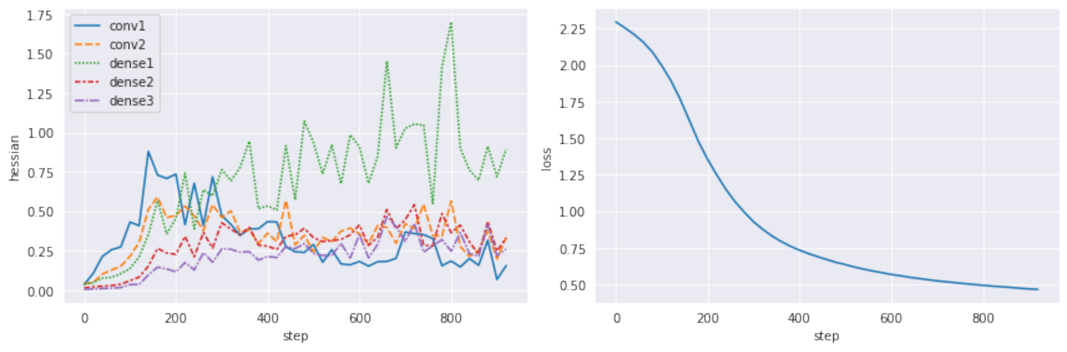


FIGURE 4.5: Hessian and loss values during training (batch estimates).

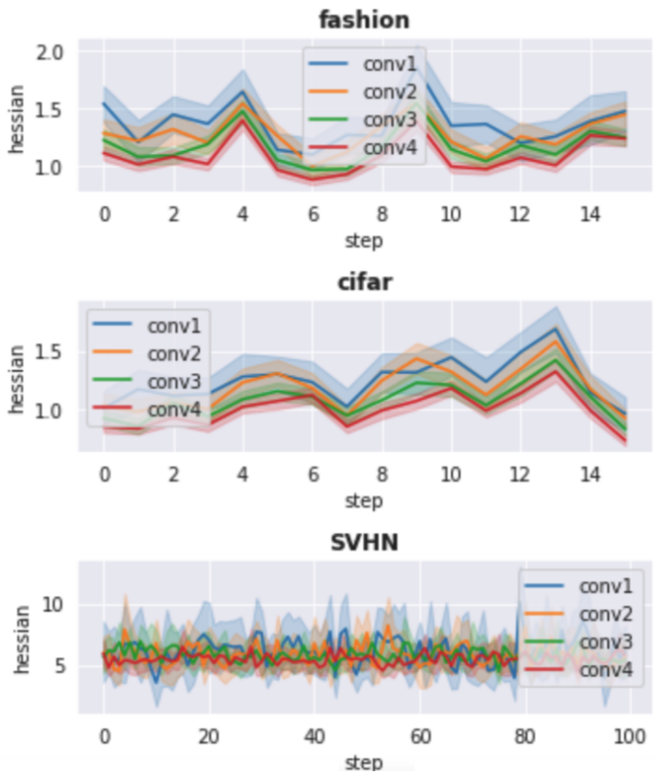


FIGURE 4.6: Hessian norm during training.

4.4.4 Error in Hessian Approximation

Theorem 1 shows that, up to terms of smaller orders, the curvature effect can be neglected. Below, we experimentally verify that the constant under the $O()$ term is indeed small. We estimated a) the relative error in Theorem 1 at **initialization**, using several initialization restarts, as shown in Figure 4.7 and b) the relative error in 1 during **training** (computed on the batch), shown in Figure 4.8 For the first experiment, we used both ResNet and EfficientNet architectures on MNIST, CIFAR10 and Flowers dataset.

For the second experiment, we used ResNet, EfficientNet and a simplified version of GoogleNet on CIFAR10. We trained the models for 5 epochs. For both experiments, we sampled 5 layers (4 convolutional and 1 dense) to show how the Hessian changes (with respect to the error and the iterations), the models are initialized with

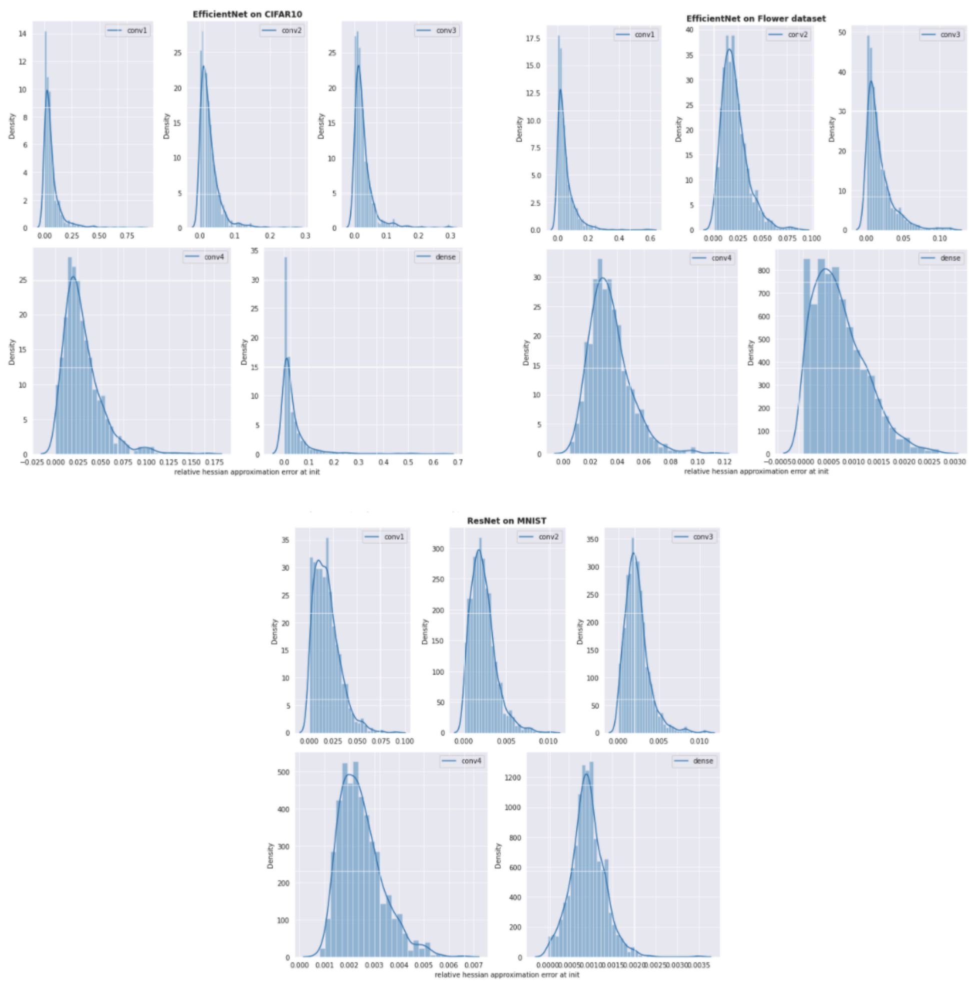


FIGURE 4.7: Our Hessian approximation at initialization.

SeLU activation function and with random distribution. The Hessian was evaluated on the corresponding gradients. We find that the approximation is accurate at initialization, and works remarkably well during training on small batches (the relative error smaller than a small constant ensures that we capture the right order of magnitude).

4.4.5 Training with the Hessian

In this experiment, we show the benefits of plugging our Hessian approximation into a new update rule based on Adam (Kingma and Ba, 2015). We trained a Word2Vec[1] model using the Tiny Shakespeare dataset (Karpathy, 2015) in three settings: (1) using SGD; (2) using the standard Adam optimizer; (3) using Adam adapted with our novel Hessian approximation.

The Tiny Shakespeare dataset (Karpathy, 2015) is composed of 40,000 lines of varied Shakespeare plays. We chose it due to its limited size, which should impact the convergence of such a high-dimensional problem and make it more challenging. Thus, this limitation should accentuate the difference of how different optimizers estimate the global curvature of the loss landscape. We used 85% of the dataset for training and 15% for validation of the model. For the SGD setting, we

[1]https://www.tensorflow.org/tutorials/text/word2vec

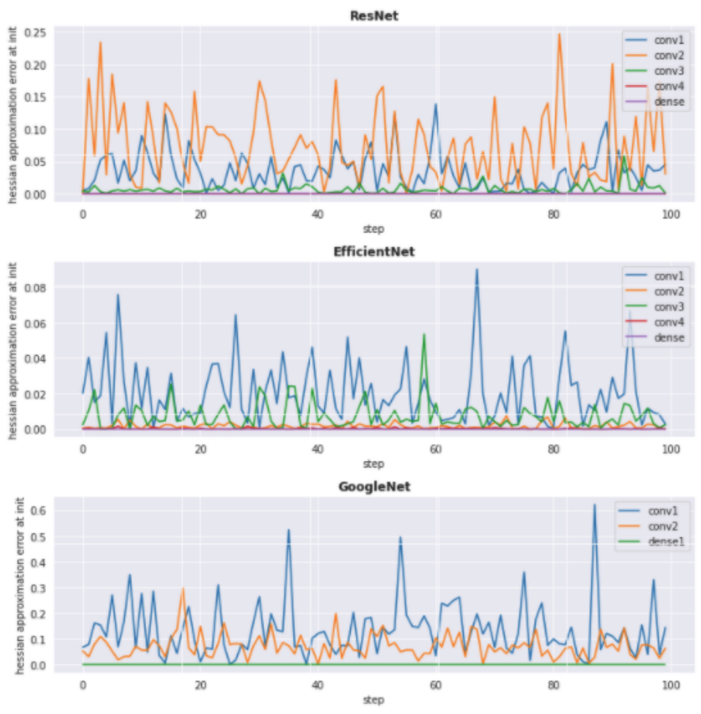


FIGURE 4.8: Our Hessian approximation during training.

used `learning_rate = 0.1`. Regarding the Adam based settings, we used mostly the same hyperparameters on both of them: `CategoricalCrossentropy` to calculate the loss, `mini_batch_size = 512`, `learning_rate = 0.001`, `beta_1 = 0.99` and `beta_2 = 0.999`. Differently, we used `epsilon = 1e-4` for our setting, while we used `epsilon = 1e-7` for standard Adam. The increase in the `epsilon` for our setting was simply to avoid gradient vanishing, which seems to happen at different values due to the Hessian approximation. We note that all the values of the other hyperparameters are the standard ones for Adam when using TensorFlow.

Figure 4.9 illustrates the results of our experiments. We can compare the difference of the *validation loss* when using different optimizers, with respect to the number of epochs used to train the Word2Vec models. There are two results that we would like to highlight: (1) our Adam adaptation achieved a smaller validation loss during the whole training; (2) our Adam adaptation was impacted less by overfitting than Adam and SGD.

Considering the computational time to train the Word2Vec model, our approach took 38% longer than Adam and was over 4 times faster than SGD. It took 229s to reach minimum validation loss (3 epochs), while Adam took 165s (3 epochs), and SGD took 999s (333 epochs). Those results are indications that the Hessian approximation may improve standard Adam convergence in exchange for some computational efficiency. However, we note that the complexity of using our Hessian approximation still is orders of magnitude smaller than using the exact Hessian. Even so, it is still possible to observe the benefits of the second-order derivatives when comparing SGD, standard Adam, and our approach with respect to the validation loss curves.

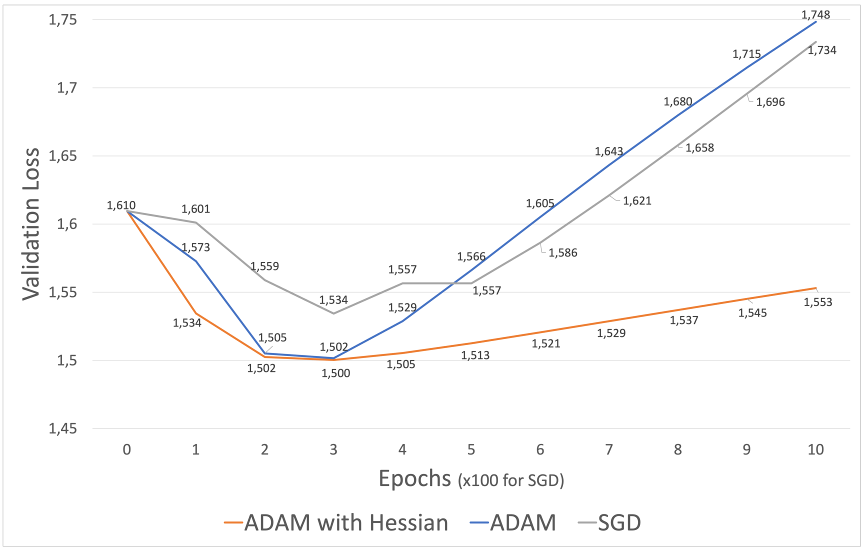


FIGURE 4.9: Comparison of SGD, standard Adam, and our Adam adapted with the Hessian approximation.

4.5 Final Remarks

We discussed how to approximate Hessians of loss functions for NNs, and devised how to use them to gain better insights into weight initialization and optimization of NNs. The main theoretical finding is our approximated chain rule for Hessian backpropagation which goes beyond first-order derivatives and is able to capture second-order effects. Besides our theoretical results, we provide a detailed empirical validation of these ideas, which demonstrates that (i) considering the Hessian norm leads to faster convergence of the learning loss than existing initialization schemes under various activation functions, (ii) we can plug Hessian information into Adam's update rule with remarkable results and (iii) our approximation of the chain rule allows for fast computations without a noticeable impact on the training time.

Part III

Improved Bounds for Random Embeddings

Chapter 5

Introduction

We now move on to the next part of this thesis, discussing improved performance of random embeddings. While we have been exploring second-order methods to optimize model parameters and potentially speed up convergence in DL, another fascinating area of research that complements these techniques is random embeddings. As we shift our focus from calculating second-order derivatives to discovering alternative approaches, random embeddings offer a way to reduce the dimensionality and complexity of data in DL models. By leveraging random projections and preserving essential structure within the data, these embeddings allow for more efficient training and improved generalization in various tasks, such as classification and clustering. In the following discussion, we will delve into the theoretical properties of random embeddings and detail on how we improved their bounds.

5.1 Dimensionality Reduction

Dimensionality reduction is a technique used in ML and data analysis to reduce the number of variables (features or dimensions) in a high-dimensional dataset. The goal is to identify a lower-dimensional subspace that still retains the important information present in the original dataset. This is accomplished by projecting the data points from the high dimensional space to the lower dimensional space in such a way that the maximum amount of information is preserved.

The main benefits of dimensionality reduction include reduced computational cost, improved visualization of the data, and improved performance of DL algorithms. It can also help in identifying patterns and relationships in the data that might not be easily noticeable in the high dimensional space. An example of this concept is shown in Figure 5.1, where we project 3d data down to a 2d plane. The 2d approximation is quite good since most points lie close to this subspace. The underlying idea behind this technique is to recognize that even though the data may seem to have a large number of dimensions, there could actually be only a limited number of factors that contribute to the variability of the data. In other words, there may only be a few latent variables that explain most of the variation in the data. For instance, when modeling the appearance of facial images, there might be only a few factors that describe most of the variability in the data. These factors could include lighting conditions, the pose of the face, the identity of the person, and so on.

There are several techniques for dimensionality reduction, including random embeddings, principal component analysis (PCA), singular value decomposition (SVD), and many more. The choice of technique depends on the specific requirements of the problem and the type of data being analyzed. A computationally simple method of dimensionality reduction that does not introduce significant distortion in the dataset would be thus desirable.

Random embeddings are a powerful tool widely used in many fields to reduce the dimensions of the input data in a computationally feasible way. This technique projects high-dimensional data into a lower-dimensional space using a random matrix while approximately preserving the main characteristics of the original data.

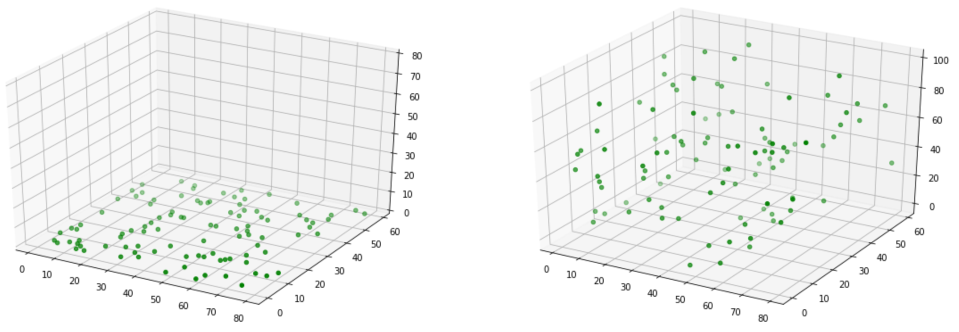


FIGURE 5.1: An example of dimensionality reduction. A set of points embedded in 3d can be represented in 2d with minimal loss of information.

This Part III studies the theoretical properties of random embeddings and their application to several classes of optimization problems and it will provide a deeper understanding of the technique and its potential applications. Specifically, we address two topics related to random embeddings. First, we analyze and improve upon the guarantees for sparse random embeddings. We show that our bounds are explicit as opposed to the asymptotic guarantees provided in the previous state of the art. Our bounds are guaranteed to be sharper by significant constants across a wide range of data-driven parameters, including dimensionality, sparsity, and dispersion of the data. This means that our approach can be used to generate random embeddings that are more accurate and efficient than traditional methods. We empirically demonstrate that our bounds significantly outperform prior works on a wide range of real-world datasets, such as collections of images, text documents represented as bags of words, and text sequences vectorized by neural embeddings. Behind our numerical improvements are techniques of broader interest, which improve upon key steps of previous analysis in terms of tighter estimates for certain types of quadratic chaos, establishing extreme properties of sparse linear forms, and improvements on bounds for the estimation of sums of independent random variables. Second, we revisit the performance of Rademacher random embeddings and establish novel statistical guarantees that are numerically sharp and non-oblivious with respect to the input data. More specifically, our result is the Schur-concavity property of Rademacher random embeddings with respect to the inputs. This offers a novel geometric perspective on the performance of random projections while improving quantitatively on bounds from previous works. This means that our methods can provide better statistical guarantees than traditional methods, even when working with non-oblivious data. This non-oblivious analysis is a novelty compared to the techniques from the previous work and bridges the frequently observed gap between theory and practice.

We start by doing an example and show how random embeddings can be used as a dimensionality reduction tool and then delve deeper into the technical details of our research and the results we have obtained. This includes a detailed explanation of the mathematical foundations of our methods, as well as numerical experiments that demonstrate the efficacy of our approach in addressing the curse of dimensionality in numerical optimization.

Example Random Embeddings

Let's examine the example below. Suppose we have the matrices as follows: X is a 10×10 matrix representing our input data. The random matrix A serves as our random embedding with sparsity $s = 0.66$, and has dimensions 10×3. Lastly, our projected data is of dimensions 10×3.
It is evident that the final projected matrix has significantly lower dimensions than the original matrix X: 10×10 compared to 10×3. If we were to increase the input data matrix by 3 or 4 orders of magnitude, the advantages of random embeddings become even more apparent. Although this example is simplistic, it effectively conveys the core concept: random embeddings offer a powerful approach for handling high-dimensional data by reducing its dimensions while preserving the essential features.

$$X = \begin{bmatrix} 6 & 4 & 5 & 1 & 8 & 1 & 7 & 5 & 6 & 8 \\ 4 & 5 & 6 & 8 & 0 & 2 & 0 & 1 & 7 & 2 \\ 0 & 5 & 0 & 7 & 7 & 0 & 0 & 8 & 8 & 5 \\ 1 & 8 & 2 & 4 & 5 & 5 & 3 & 2 & 4 & 1 \\ 0 & 1 & 4 & 1 & 8 & 7 & 5 & 5 & 4 & 1 \\ 4 & 6 & 2 & 4 & 8 & 7 & 2 & 6 & 3 & 4 \\ 3 & 1 & 8 & 4 & 7 & 4 & 2 & 5 & 1 & 6 \\ 5 & 1 & 6 & 2 & 7 & 3 & 3 & 0 & 6 & 0 \\ 4 & 5 & 3 & 2 & 4 & 3 & 8 & 7 & 1 & 1 \\ 4 & 1 & 4 & 2 & 1 & 8 & 1 & 4 & 7 & 3 \end{bmatrix}$$

$$A^T = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$A^T X = \begin{bmatrix} 32 & 12 & 17 \\ 16 & 7 & 17 \\ 28 & 5 & 8 \\ 14 & 16 & 7 \\ 22 & 13 & 8 \\ 23 & 15 & 9 \\ 27 & 7 & 12 \\ 19 & 7 & 17 \\ 16 & 16 & 8 \\ 19 & 10 & 15 \end{bmatrix}$$

Next, we are going to describe the most used dimensionality reduction techniques: random embeddings, principal component analysis (PCA) and singular value decomposition (SVD).

5.2 Methods for Dimensionality Reduction

5.2.1 Random Embeddings

This technique is based on the Johnson-Lindenstrauss lemma, which states that any high-dimensional data can be embedded in a lower-dimensional space while approximately preserving the pairwise distances between the points. This lemma provides a theoretical foundation for the technique and makes it more reliable. By showing that it is possible to approximate the Euclidean distances between data points in a lower-dimensional space, it provides a way to embed high-dimensional data in a computationally feasible way. We are going to discuss further this lemma in the next chapter, at the moment this intuition is sufficient to describe random embedding with the following notation,

$$\boldsymbol{X}_{d\times n}^{Rand.Emb.} = \boldsymbol{A}_{d\times m}\boldsymbol{X}_{m\times n}$$

given that $\boldsymbol{X}_{m\times n}$ is the original data that we want to project, $\boldsymbol{A}_{d\times m}$ is the random matrix used for the projection and $\boldsymbol{X}_{d\times n}^{Rand.Emb.}$ represents the projected data to a d-dimensional subspace. Random embeddings (also known as random projections) can be faster and more effective than most dimensionality reduction techniques. Looking at the time complexity, it is $O(ndm)$ when projecting on a subspace of size d (and even faster if the matrix is sparse).

5.2.2 Principal Components Analysis

Like random embeddings, also PCA defines a projection from a high-dimensional space into a low-dimensional space picking a small set of "direction" vectors that can be used as a basis to explain the data in the low-dimensional space. The major difference is that PCA is trying hard to pick the best basis vectors by looking for directions in which the original data varies the most (as the first principal component can equivalently be defined as a direction that maximizes the variance of the projected data) while random embeddings are picking the directions randomly.

$$\boldsymbol{X}^{PCA} = \boldsymbol{Eigen}_{matrix}\boldsymbol{X}$$

For PCA the "directions" are the eigenvectors of the data covariance matrix Eigen$_{matrix}\{\boldsymbol{X}\boldsymbol{X}^T\}$, then if dimensionality reduction of the given dataset is desired, the data can be projected onto a subspace spanned by the most important eigenvectors. When working with very high dimensional data, PCA can become slow with time complexity of $O(n^2 \times m + n^3)$ on a matrix of size $m \times n$. PCA maintains the best possible projection, however, if you want to explore variability in order to gain stability and precision from aggregation, there is only one set of principal components (i.e., only one best possible projection) and PCA cannot be used in this situation.

Example PCA

Given the initial multi-dimensional array:

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

1. Standardize the data (mean = 0, standard deviation = 1):
Calculate the mean for each column:

$$\mu_1 = \frac{1+4+7}{3} = 4, \quad \mu_2 = \frac{2+5+8}{3} = 5, \quad \mu_3 = \frac{3+6+9}{3} = 6$$

Subtract the mean from each value in the corresponding column and divide by the standard deviation (std) for each column:

$$\sigma_1 = 2.45, \quad \sigma_2 = 2.45, \quad \sigma_3 = 2.45$$

$$X_{\text{std}} = \begin{bmatrix} -1.22 & -1.22 & -1.22 \\ 0.00 & 0.00 & 0.00 \\ 1.22 & 1.22 & 1.22 \end{bmatrix}$$

2. Compute the covariance matrix:

$$Cov_{\text{matrix}} = \begin{bmatrix} 0.75 & 0.75 & 0.75 \\ 0.75 & 0.75 & 0.75 \\ 0.75 & 0.75 & 0.75 \end{bmatrix}$$

3. Calculate the eigenvalues and eigenvectors of the covariance matrix:
Eigenvalues: $\lambda_1 = 2.25$, $\lambda_2 = 0$, $\lambda_3 = 0$

Eigenvectors: $v_1 = \begin{bmatrix} 0.577 \\ 0.577 \\ 0.577 \end{bmatrix}, v_2 = \begin{bmatrix} -0.707 \\ 0.000 \\ 0.707 \end{bmatrix}, v_3 = \begin{bmatrix} 0.408 \\ -0.816 \\ 0.408 \end{bmatrix}$

4. Sort the eigenvalues in descending order and select the d eigenvectors corresponding to the d largest eigenvalues. Since we want to reduce the dimensions to 2, we will choose the eigenvectors corresponding to the 2 largest eigenvalues (λ_1 and λ_2):

$$Eigen_{\text{matrix}} = \begin{bmatrix} 0.577 & -0.707 \\ 0.577 & 0.000 \\ 0.577 & 0.707 \end{bmatrix}$$

5. Project the standardized data onto the new eigenspace (multiply the standardized data with the eigenvector matrix):

$$X_{\text{projected}} = \begin{bmatrix} -2.11 & -0.87 \\ 0.00 & 0.00 \\ 2.11 & 0.87 \end{bmatrix}$$

By applying PCA, we have reduced the dimensions of the original 3x3 array to a 3x2 array:

$$X_{\text{projected}} = \begin{bmatrix} -2.11 & -0.87 \\ 0.00 & 0.00 \\ 2.11 & 0.87 \end{bmatrix}$$

5.2.3 Singular Value Decomposition

A closely related method to PCA is SVD,

$$\boldsymbol{X}_{m\times n} = \boldsymbol{U}_{m\times m}\boldsymbol{S}_{m\times n}\boldsymbol{V}^T_{n\times n}$$

where $\boldsymbol{U}$ is an orthogonal matrix whose columns are orthonormal eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$, $\boldsymbol{S}$ is a diagonal matrix containing the square roots of the eigenvalues of $\boldsymbol{U}$ and $\boldsymbol{V}$ and $\boldsymbol{V}^T$ is the transpose of an orthogonal matrix whose rows are orthonormal eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$. However, by reducing $\boldsymbol{S}$ to its d largest singular values, we may achieve the desired dimensionality reduction also for $\boldsymbol{U}$ and $\boldsymbol{V}^T$:

$$\boldsymbol{X}_{m\times n} \approx \boldsymbol{U}_{m\times d}\boldsymbol{S}_{d\times d}\boldsymbol{V}^T_{d\times n}$$

Using SVD, the dimensionality of the data can be reduced by projecting the data onto the reduced-dimensional space $\boldsymbol{V}^T_d$ corresponding to the d largest singular values:

$$\boldsymbol{X}^{SVD} = \boldsymbol{V}^T_{d\times n}\boldsymbol{X}$$

Like PCA, SVD is expensive to compute. It is more efficient (even if still expensive) for sparse data, as the time complexity becomes dependent on the c non-zero entries per column $O(c \times m \times n)$, which is why it is used for text processing documents in NLP use cases.

Example SVD

Let's perform SVD on the same 3x3 multi-dimensional array we used for the PCA example. Given the initial multi-dimensional array:

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

1. Perform SVD on X:

$$U = \begin{bmatrix} -0.215 & 0.884 & 0.415 \\ -0.520 & 0.240 & -0.823 \\ -0.826 & -0.404 & 0.408 \end{bmatrix}$$

$$S = \begin{bmatrix} 16.848 & 0.000 & 0.000 \\ 0.000 & 1.528 & 0.000 \\ 0.000 & 0.000 & 0.000 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.479 & -0.576 & -0.674 \\ 0.776 & 0.076 & -0.624 \\ 0.408 & -0.816 & 0.408 \end{bmatrix}$$

2. Reduce dimensions by selecting the d largest singular values (d=2 in our case) and their corresponding columns in U and V^T:

$$U_d = \begin{bmatrix} -0.215 & 0.884 \\ -0.520 & 0.240 \\ -0.826 & -0.404 \end{bmatrix}$$

$$S_d = \begin{bmatrix} 16.848 & 0.000 \\ 0.000 & 1.528 \end{bmatrix}$$

$$V_d^T = \begin{bmatrix} -0.479 & -0.576 & -0.674 \\ 0.776 & 0.076 & -0.624 \end{bmatrix}$$

3. Reconstruct the matrix X with reduced dimensions by multiplying U_d, S_d, and V_d^T:

$$X_{\text{reduced}} = U_d \times S_d \times V_d^T = \begin{bmatrix} -0.846 & 1.945 & 1.111 \\ -4.683 & 4.946 & 4.111 \\ -8.520 & 7.946 & 7.111 \end{bmatrix}$$

Keep in mind that this reconstructed matrix is an approximation of the original matrix with reduced dimensions.

4. Project the original data onto the reduced-dimensional space by multiplying the original matrix X by the reduced V_d^T:

$$X_{\text{projected}} = X \times V_d^T = \begin{bmatrix} -2.689 & 1.483 \\ -6.142 & 0.444 \\ -9.594 & -0.594 \end{bmatrix}$$

By applying SVD, we've reduced the dimensions of the original 3x3 array to a 3x2 array:

$$X_{\text{projected}} = \begin{bmatrix} -2.689 & 1.483 \\ -6.142 & 0.444 \\ -9.594 & -0.594 \end{bmatrix}$$

Although PCA and SVD are powerful dimensionality reduction techniques, they can be computationally expensive, particularly when dealing with large, high dimensional datasets. This computational burden often poses challenges for real-world applications that require fast and efficient data processing. However, there exist numerical approximations that alleviate this burden by providing approximate solutions, without calculating PCA or SVD exactly. These approximate methods, such as Randomized PCA and Randomized SVD, utilize randomized algorithms to significantly reduce computational complexity while still preserving the essential structure of the data. By employing these approximations, one can achieve substantial speedup and memory efficiency in comparison to the traditional PCA and SVD techniques, making them more feasible for large-scale data analysis tasks. In light of these challenges and the potential benefits of randomized algorithms, we investigated random embeddings as an alternative approach for dimensionality reduction. The next chapter will introduce the background and state-of-the-art of random embeddings, providing a comprehensive overview of this promising technique for efficient data processing and analysis.

Chapter 6

Background and Related Work

The seminal result of Johnson and Lindenstrauss, 1984 states that *random linear mappings* have nearly isometric properties, and hence are well-suited for embeddings: they *nearly preserve distances* when projecting high-dimensional data into a *lower-dimensional space*. Formally, for an error parameter $\epsilon > 0$, an $m \times n$ matrix $\boldsymbol{A}$ appropriately sampled (e.g., using appropriately scaled Gaussian entries), and any input vector $\boldsymbol{x} \in \mathbb{R}^n$, it holds that

$$1 - \epsilon \leqslant \|\boldsymbol{A}\boldsymbol{x}\|_2 / \|\boldsymbol{x}\|_2 \leqslant 1 + \epsilon \quad \text{with probability } 1 - \delta \tag{6.1}$$

if the embedding dimension is $m = \Theta\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$.

This bound on the dimension m has been shown to be *asymptotically optimal* (Jayram and Woodruff, 2013), while the assumptions made on the Gaussian distribution of matrix A can be further replaced by the Rademacher distribution, or relaxed even further by only requiring the sub-Gaussian condition to hold for the construction of the projection matrix $\boldsymbol{A}$.

The result is a typical *dimension-distortion tradeoff*: one aims to minimize $m \ll n$, while keeping ϵ and δ possibly small. Smaller dimensions m allow for efficient processing of large, high-dimensional datasets, while a small distortion guarantees that analytical tasks can be performed with a similar effect over the embedded data as it is the case for the original data.

Over the past years, variants of the aforementioned *Johnson-Lindenstrauss Lemma* have found important applications to text mining and image processing (Bingham and Mannila, 2001), approximate nearest-neighbor search (Ailon and Chazelle, 2006; Indyk and Motwani, 1998), approximation algorithms for clustering high-dimensional data (Makarychev, Makarychev, and Razenshteyn, 2019), speeding up computations in linear algebra (Nelson and Nguyên, 2013; Sarlos, 2006; Clarkson and Woodruff, 2017), analyzing graphs (Frankl and Maehara, 1988; Linial, London, and Rabinovich, 1995), and even to hypothesis testing (Lopes, Jacob, and Wainwright, 2011; Shi, Lu, and Song, 2020) and privacy (Blocki et al., 2012; Kenthapadi et al., 2013). It is also worth mentioning, from a theoretical perspective, the importance of understanding Hilbert spaces in functional analysis (Johnson and Naor, 2010). Finally, we note that, while 6.1 gives high-probability guarantees for the sampled matrix, it is also possible to find the concrete matrix A which (surely) satisfies this inequality in randomized polynomial time (Dasgupta and Gupta, 1999) or by means of derandomization (Kane and Nelson, 2010). The focus of this paper is on *linear sparse random embeddings*, where $\boldsymbol{A}$ in 6.1 has at most s non-zero entries in each column, which allows for faster computation of the embedded vectors. This setup has been covered by a substantial line of recent research (Ailon and Chazelle, 2006; Kane and Nelson, 2014; Li, Hastie, and Church, 2006), which established that, for the optimal dimension m, one may set $s = \Theta(m\,\epsilon)$, thereby gaining a factor of ϵ in matrix sparsity[1]. This can be further improved by exploiting *structural properties* of the input data: as shown in recent works (Kane and Nelson, 2014; Freksen, Kamma, and Larsen, 2018; Freksen, Kamma, and Larsen, 2019), with $v \triangleq \|\boldsymbol{x}\|_\infty / \|\boldsymbol{x}\|_2$, one may set the sparsity to

$$s = \Theta\left(\frac{v^2}{\epsilon}\max\left(\log\frac{1}{\delta}, \frac{\log^2\frac{1}{\delta}}{\log\frac{1}{\epsilon}}\right)\right) \tag{6.2}$$

while keeping the optimal choice of dimension $m = \Theta\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$. This shows that a better sparsity s is feasible when the data-dependent parameter v is small. The parameter v should thus be understood as the *dispersion* of the input vector $\boldsymbol{x}$, i.e., v is small when the components of $\boldsymbol{x}$ are of comparable magnitude, and it is larger when there are dominating components. Empirically, random embeddings work much better than predicted by their theoretical bounds. The main goal of this work thus is to bridge this frequently observed gap between theory and practice and thereby develop both *robust* and *provable guarantees* for sparse random embeddings.

Recent state-of-the-art analyses (Freksen, Kamma, and Larsen, 2018; Freksen, Kamma, and Larsen, 2019) provide only asymptotic bounds which tend to disguise dependencies on rather large constants (Freksen, Kamma, and Larsen, 2018). In practice, they often yield trivial results which however limits their usability. Moreover, real-data evaluations from prior works are mostly of qualitative nature: they analyze trends in parameter tradeoffs (Freksen, Kamma, and Larsen, 2019) rather than provable guarantees. Regarding the dispersion measure $v = \|\boldsymbol{x}\|_\infty / \|\boldsymbol{x}\|_2$, which is the key ingredient of recent improvements, no study has evaluated its typical behavior on real-world data to our knowledge so far. The typical value of the dispersion v may also depend on the type of the data (text, images, etc.), which in turn makes the findings harder to generalize.

To date, the literature offers no satisfactory treatment of this prevalent gap between provable and practically meaningful guarantees. Some authors (Freksen, Kamma, and Larsen, 2018; V. and W., 2011) suggested that very good empirical performance may be an evidence for small constants, but it may well be the case that sparse random embeddings work better than predicted by the underlying theory due to other data properties, not present in any of the analyses. Indeed, while one can expect a low data dispersion to help increasing sparsity, the proposed dispersion measure v is very crude and does not capture this aspect well in a quantitative sense.

Theory of sparse random embeddings. Our work improves directly upon Freksen, Kamma, and Larsen, 2018 (case $s = 1$) and Freksen, Kamma, and Larsen, 2019 (general s). These works determine the relation between sparsity and data dispersion, thereby building on a long line of earlier works on variants of the Johnson-Lindenstrauss Lemma (Ailon and Chazelle, 2006; Kane and Nelson, 2014; Li, Hastie, and Church, 2006). The provided bounds, albeit proven to be asymptotically optimal, suffer from a *lack of explicit constants* which cannot be easily extracted from the previous estimates.

Empirical evaluation of random embeddings. The good empirical performance of random linear embeddings, including sparse variants, has been confirmed many

[1]One may in fact further reduce the sparsity s by a factor of $B > 1$, however at the cost of increasing the dimension m by a factor of $2^{\Theta(B)}$ (i.e., exponentially).

times (Bingham and Mannila, 2001; V. and W., 2011). These works point out the gap between provable and observed performance, which we are addressing in this work.

Estimation of quadratic chaos. Technically speaking, the analysis of errors in random projections can be reduced to the more general problem of estimating quadratic forms of random variables, also called *quadratic chaos*. The literature offers a variety of tools, from variants of the well-known Hanson-Wright Lemma (Hanson, 1971; Zhou, 2019) to more specialized bounds (Boucheron et al., 2005). However, they would produce worse constants than our direct approach.

Embeddings of large collections/subspaces. Orthogonal to obtaining bounds for a single vector x is the question of how to extend such bounds to hold simultaneously for all x from a finite collection or an entire subspace of input data. This can be done by a black-box reduction using ϵ-net arguments and is solved by a reduction to the single vector case by means of ϵ-net arguments. Such bounds can be also obtained in our case.

Rademacher random projections. As recently Burr, Gao, and Knoll, 2018 showed, Rademacher random projections are asymptotically *dimension-optimal with exact constant*; this result improves upon a previous suboptimal bound of Kane and Nelson, 2014. The statistical performance of Rademacher projections is superior to the sparse ones. Furthermore, the theoretical bounds for Rademacher random projections are much better than those available for sparse analogues (Cohen, Jayram, and Nelson, 2018). The best, prior to this paper, analysis of

$$\Phi_{k,j} = \frac{1}{\sqrt{m}} r_{k,i}, \quad r_{k,i} \sim^{IID} \{-1,+1\}. \tag{6.3}$$

is given by Achlioptas, 2003. It is worth noting that Rademacher projections are also superior to their Gaussian counterparts; indeed, we know that they are dominated by the gaussian-based projections (Achlioptas, 2003). The relation of statistical performance and input structure has not been understood in-depth yet; as for conceptually similar research, we note that recent results show that for sparse data one can improve the sparsity of random projections, gaining in computing time (Freksen, Kamma, and Larsen, 2019).

Chapter 7

Robust and Provable Guarantees for Sparse Random Embeddings

In this chapter, our contribution that focuses on sparse random embeddings is presented. We show that our bounds are explicit as opposed to the asymptotic guarantees provided in the previous state of the art and our bounds are guaranteed to be sharper by significant constants across a wide range of data-driven parameters, including dimensionality, sparsity, and dispersion of the data. Our approach can be used to generate random embeddings that are more accurate and efficient than traditional methods. We empirically demonstrate that our bounds significantly outperform prior works on a wide range of real-world datasets, such as collections of images, text documents represented as bags-of-words, and text sequences vectorized by neural embeddings. Behind our numerical improvements are techniques of broader interest, which improve upon key steps of previous analysis in terms of tighter estimates for certain types of quadratic chaos, establishing extreme properties of sparse linear forms, and improvements on bounds for the estimation of sums of independent random variables.

7.1 Construction of the Embeddings

Let $\boldsymbol{A}$ be an $m \times n$ matrix which is sampled as follows:

(1) Fix a positive integer $s \leqslant m$, the *column sparsity* of $\boldsymbol{A}$.

(2) For each column, select s row positions at random (without replacement), place ± 1 uniform-randomly at these positions and 0 at the remaining positions.

(3) Finally, scale all entries of A by $\frac{1}{\sqrt{s}}$.

Remark 8 (Alternative Constructions)**.** *The above construction of* **A** *is as in Freksen, Kamma, and Larsen, 2018; Freksen, Kamma, and Larsen, 2019, but our analysis works also when sampling is done with replacement.*

To analyze the error obtained from the respective projection of $\boldsymbol{x}$ by $\boldsymbol{A}$, we define as in Freksen, Kamma, and Larsen, 2019

$$E(\boldsymbol{x}) \triangleq \|\boldsymbol{A}\boldsymbol{x}\|_2^2 - \|\boldsymbol{x}\|_2^2 = \sum_{r=1}^{m} \sum_{1 \leqslant i \neq j \leqslant n} \boldsymbol{A}_{r,i} \boldsymbol{A}_{r,j} \boldsymbol{x}_i \boldsymbol{x}_j \tag{7.1}$$

which is then analyzed by looking into individual "row" contributions, namely $E(\boldsymbol{x}) = \frac{1}{s}\sum_{r=1}^{m} E_r(\boldsymbol{x})$ with

$$E_r(\boldsymbol{x}) \triangleq s \sum_{1\leqslant i\neq j\leqslant n} \boldsymbol{A}_{r,i}\boldsymbol{A}_{r,j}\boldsymbol{x}_i\boldsymbol{x}_j\,. \tag{7.2}$$

The goal is to identify conditions, such that $\Pr_{\boldsymbol{A}}[|E(\boldsymbol{x})| > \epsilon\|\boldsymbol{x}\|_2^2] \leqslant \delta$, as this implies 6.1. By scaling, we can assume $\|\boldsymbol{x}\|_2 = 1$. Throughout the paper, we denote $p = \frac{s}{m}$.

7.2 Key Techniques for the Analysis

For the following steps, we leverage two techniques which were not used in prior work, namely (a) *careful use of symmetry properties* and (b) *majorization*.

7.2.1 Quadratic Chaos Estimation

Studying the error $E(\boldsymbol{x})$, due to pairwise terms, requires the estimation of quadratic forms $\sum_{i\neq j} Z_i Z_j$, with $Z_i = \boldsymbol{A}_{r,i}\boldsymbol{x}_i$. To this end, we develop a useful general inequality, which reduces the problem to (simpler) linear forms.

Lemma 2. *For symmetric and independent random variables Z_i and any positive even d, we have:*

$$\|\sum_{i\neq j} Z_i Z_j\|_d \leqslant 4\,\|\sum_i Z_i\|_d^2 \tag{7.3}$$

Remark 9. *The proof establishes more, namely that for a positive integer d (odd or even), we have $\|\sum_{i\neq j} Z_i Z_j\|_d \leqslant 4\,\|\sum_{i\neq j} Z_i Z_j'\|_d$ where Z_i' are independent copies of Z_i.*

The constant $C = 4$ in Lemma 2 can be further improved. For example, it is easily seen that for $d = 2$ one may choose $C = \sqrt{2}$. For a general d, the use of hypercontractive inequalities may give furthers refinements.

7.2.2 Extremal Properties of Linear Chaos

We now move on to deriving bounds for linear forms of symmetric random variables, which bound quadratic forms. The following lemma gives a geometric insight into their behavior with respect to the input weights.

Lemma 3. *For $\mathbf{x} \in \mathbb{R}^n$, define $S(\mathbf{x}) = \sum_i \mathbf{x}_i Y_i$ where $Y_i \sim^{\text{i.i.d.}} Y$ with $Y \in \{-1, 0, 1\}$ taking values ± 1 each with probability $p/2$ and 0 with probability $1-p$. Then, for every pair of vectors $\mathbf{x}, \mathbf{x}'$, such that $(\mathbf{x}_i^2)_i \succ (\mathbf{x}_i'^2)_i$, and positive even integer d, the following inequality holds:*

$$\|S(\mathbf{x})\|_d \leqslant \|S(\mathbf{x}')\|_d \tag{7.4}$$

The lemma yields the following corollary.

Corollary 4. *Let Y_i be as in Lemma 3. For $v \in (0,1)$, consider all vectors $\mathbf{x} \in \mathbb{R}^n$, such that $\|\mathbf{x}\|_2 = 1$ and $\|\mathbf{x}\|_\infty = v$. Then, $\|\sum_i \mathbf{x}_i Y_i\|_d$ for an even $d > 0$ is maximized at $\mathbf{x} = \mathbf{x}^*$ where:*

$$\mathbf{x}_i^* = \begin{cases} v & i = 1 \\ \sqrt{\frac{1-v^2}{n-1}} & i = 2\ldots n \end{cases} \tag{7.5}$$

7.2.3 Estimation of I.I.D. Sums

The techniques outlined above allow us to bound the row-wise error contributions $E_r(\boldsymbol{x})$. In order to assemble them into a bound on the overall error $E(\boldsymbol{x})$, we prove the following lemma.

Lemma 4. *Let $Z_1, \ldots, Z_m \sim^{i.i.d.} Z$, where Z is symmetric, and let d be positive and even. Then:*

$$\|\sum_{i=1}^{m} Z_i\|_d \leqslant \min\left\{t > 0 : \mathbb{E}(1+Z/t)^d \leqslant \mathrm{e}^{\frac{d}{2m}}\right\} \tag{7.6}$$

This improves the constant provided in the seminal result of Latala, 1997 by a factor of $\mathrm{e}^{1/2}$.

7.3 Bounds Based on Error Moments

We first bound the row-wise error contributions $E_r(\boldsymbol{x})$, defined in 7.2, as follows.

Lemma 5. *Suppose that $\|\mathbf{x}\|_2 = 1$ and $\|\mathbf{x}\|_\infty = v$, then we have $\|E_r(\mathbf{x})\|_d \leqslant T_{n,p,d}(v)$ for any positive and even d, where we define*

$$T_{n,p,d}(v) \triangleq 4\left(\sum_{k=0}^{\frac{d}{2}} \binom{d}{2k} p^{\mathbb{I}(k>0)}\, v^{2k}(1-v^2)^{\frac{d-2k}{2}} \cdot \mathbb{E}(B'-B'')^{d-2k}\right)^{\frac{2}{d}} \tag{7.7}$$

and $B', B'' \sim^{i.i.d.} \frac{1}{\sqrt{n-1}} \cdot \mathrm{Binom}(n-1, \frac{1-\sqrt{1-2p}}{2})$.

To show this result, we combine Lemma 2 and Lemma 3. When explicitly evaluating $\|\sum_i \boldsymbol{x}_i^* Y_i\|_d$, we thereby arrive at the expression given by 7.7.

The following theorem is the main result of our work.

Theorem 3 (Error Moments). *If $\|\mathbf{x}\|_2 = 1$ and $\|\mathbf{x}\|_\infty = v$, then for any positive even d, we have that*

$$\|E(\mathbf{x})\|_d \leqslant s^{-1} \cdot Q_{n,p,d}(v),$$

where $Q = Q_{n,p,d}(v)$ solves the equation

$$\sum_{k=0}^{\frac{d}{2}} \binom{d}{2k} (T_{n,p,2k}(v)/Q)^{2k} = \mathrm{e}^{\frac{d}{2m}} \tag{7.8}$$

and $T_{n,p,2k}$ is as in Lemma 5 (with d replaced by $2k$).

The detailed proof (see appendix B) starts with $E(\boldsymbol{x}) = \frac{1}{s}\sum_{r=1}^{m} E_r(\boldsymbol{x})$, applies Lemma 4 with $Z_r = E_r(\boldsymbol{x})$, and finally uses Lemma 5 (similarly to Freksen, Kamma, and Larsen, 2019). The subtle points of the proof are summarized below.

- *Correlation of $E_r(\mathbf{x})$ for different r:* fortunately (due to sampling without replacement), this is a *negative dependency*. Thus, the same moment bounds as for independent random variables can be applied also here (Shao, 2000).

- *Non-symmetric distribution of $E_r(\mathbf{x})$:* we compare the moments of $E_r(x)$ with the moments of a random variable which is symmetric; this allows for applying moment bounds for the sums of symmetric random variables. We remark that this argument also fills a gap in Freksen, Kamma, and Larsen, 2019.

Corollary 5 (Error Confidence)**.** *For the error*

$$\epsilon = \mathrm{e}\, s^{-1} \cdot Q_{n,p,\lceil \log(1/\delta) \rceil}(v),$$

we have $\Pr[|E(\mathbf{x})| > \epsilon] \leqslant 1 - \delta$ *and* (6.1) *holds.*

The corollary is a direct application of Markov's inequality $\Pr[|E(x)| > \epsilon] \leqslant (s^{-1}Q_{n,p,d}(v)/\epsilon)^d$.

7.4 Discussion

Remark 10 (Computational Efficiency)**.** *The time of evaluating the distortion ϵ in 5 is in*

$$\mathrm{TIME} = O(\log^4(1/\delta)\log(m\log(1/\delta))). \tag{7.9}$$

This is because $T_{n,p,d}(v)$ can be evaluated with $O(d^3)$ operations, utilizing the combinatorial formulas for binomial moments. In turn, $Q_{n,p,d}(v)$ inverts a monotone function which can be computed by bisection in $O(\log(m\,d))$ steps.

Remark 11 (Comparison to State-of-the-Art)**.** *The approach of Freksen, Kamma, and Larsen, 2019 follows the same roadmap, but the critical steps in that work are estimated in a weaker way than in our approach, namely:*

1. *a weaker analogue of our Lemma 2 is used,*
2. *in place of our sharp Corollary 4, an overestimation of $\| \sum_i \mathbf{x}_i Y_i \|_d$ is obtained,*
3. *bounds on $E_r(\mathbf{x})$ are assembled to bound $E(\mathbf{x})$ via a weaker variant of Latala, 1997, which is further weaker than our Lemma 4.*

Thus, our bounds are guaranteed to be tighter for all parameter regimes.

Remark 12 (Dependency on n)**.** *Although our dependency on n is only asymptotically bounded, we find that—interestingly—it indeed helps improving the bounds on real-world datasets and use-cases, as shown in the next section.*

7.5 Empirical Evaluation

In this section, the present the detailed results of our experimental evaluation. We implemented the bound provided by Theorem 4 in Python 3.6 and tested it in the Google Colab environment using an Intel(R) Xeon(R) CPU @ 2.20GHz and the default RAM configuration of 13GB.

Best Bounds in Prior Works.

To give a clear and fair comparison, we analyze the best constants in the previous asymptotic analysis (Freksen, Kamma, and Larsen, 2019). The in-depth analysis gives the value of "optimistic" constants necessary to avoid breaking down the proof (while the actual constants are likely worse).

Remark 13 (Optimistic Constants in Prior Works)**.** *The bound of Freksen, Kamma, and Larsen, 2019 uses the better of the following two lemmas (Lemmas D.1 and D.2, respectively):*

1. $\|E_r(\mathbf{x})\|_d \leqslant 2\,C_1 \cdot \left(\sup_{1\leqslant t \leqslant \frac{d}{2}} \left[\frac{dv}{t}\left(\frac{p}{dv^2}\right)^{\frac{1}{2t}}\right]\right)^2$
2. $\|E_r(\mathbf{x})\|_d \leqslant 2\,C_2 \cdot \frac{d}{\log(1/p)}$,

where d is assumed positive and even.

Here, the extra factor of 2 appears as the effect of symmetrization (the random variable $E_r(x)$ must be dominated by a symmetric random variable to conclude the bound on $E(x)$). The best constants satisfy $C_1 \geqslant 4\mathrm{e}$ and $C_2 \geqslant 8$, as it is implied by the analysis of their proof technique.

7.5.1 Synthetic Benchmark

Setup. The key ingredient of our improvements is the sharper bound on the row-wise error contributions $E_r(x)$ from Lemma 5. In this experiment, we compare this bound (referred to as T_{new}) with its analogue from Freksen, Kamma, and Larsen, 2019 with the "optimistic" constants as discussed in Remark 13 (referred to as T_{old}). Figures 7.1 and 7.2 illustrate the respective ratios of T_{new} and T_{old} with respect to the error contributions $E_r(x)$ for $n = 10^4$ and various ranges of d, v and $p = \frac{s}{m}$. Points with non-even d are interpolated.

Results. Our bounds are better by up to an order of magnitude across a wide range of parameters. Therefore, we should expect similar improvements for our bounds on the overall error $E(x)$ (recall that $E_r(x)$ are aggregated into $E(x)$ using Lemma 4).

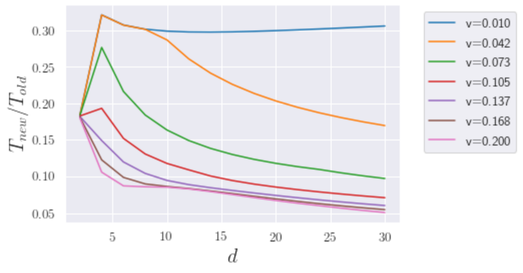


FIGURE 7.1: T_{new}/T_{old} for $n = 10^4$, $p = 10^{-3}$

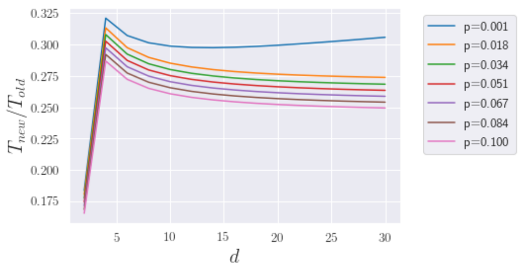


FIGURE 7.2: T_{new}/T_{old} for $n = 10^4$, $v = 10^{-2}$

7.5.2 Real-World Datasets

Setup. We next consider various real-world datasets of different content types, sizes and numbers of features—as summarized in Table 7.1.

Dispersion. Since sparsity s depends on the data-dependent dispersion v, results obtained in prior work may be of limited applicability in practice when v is not small. To understand the behavior of v, we evaluate its distribution on our datasets. We conclude that, indeed, the value of v may be quite large, even when n is big; in such cases, using a very small sparsity s is not theoretically justified. Density plots on Figures 7.3, 7.4, 7.5, and 7.6 illustrate the distribution of the dispersion $v = \|x\|_\infty / \|x\|_2$ for vectors $x = x_1 - x_2$ over all pairs $x_1, x_2 \in \mathcal{X}$ from a subsample $\mathcal{X}$ of the dataset. Evaluating the dispersion on pairwise differences corresponds to the intended usage of random projections: preserving pairwise distances within a dataset. We used $|\mathcal{X}| = 250$, such that v is estimated based on $\approx 5 \cdot 10^4$ samples.

Dataset	Content	Comments
`NIPS`	text	13,000 words
`Word2Vec/Wiki`	text	5M lines / 48M words of English Wikipedia articles
`News20`	text	20,000 documents / 34,000 words of English news (Ken, 1995)
`MNIST`	images	60,000 images with 28x28 pixels (LeCun and Cortes, 2010)
`CIFAR100`	images	60,000 images with 32x32 pixels (Krizhevsky, 2012)
`SVHN`	images	600,000 images with 32x32 pixels (Netzer et al., 2011)
`Caltech101`	images	9,000 images with 300x200 pixels (Fei-Fei, Fergus, and Perona, 2004)
`Cars`	images	16,000 images with 500x500 pixels (Deng, Krause, and Fei-Fei, 2013)
`Goodwin040`	fluid dynamics	18,000 columns / 18,000 rows (Davis and Hu, 2011)
`Mycieliskian17`	undir. graph	98,000 columns / 98,000 rows (Davis and Hu, 2011)

TABLE 7.1: Summary of real-world datasets used in our experiments

We generally find that, for each dataset, v is sharply concentrated around a "typical" value, whose magnitude is data-dependent.

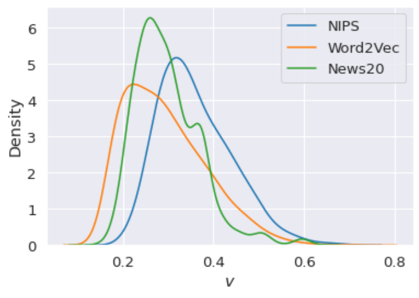


FIGURE 7.3: v on text data

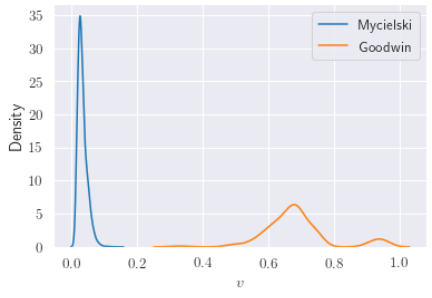


FIGURE 7.4: v on sparse-matrix

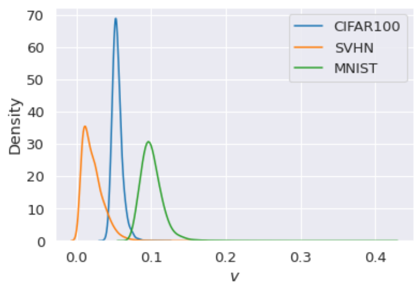


FIGURE 7.5: v on small images

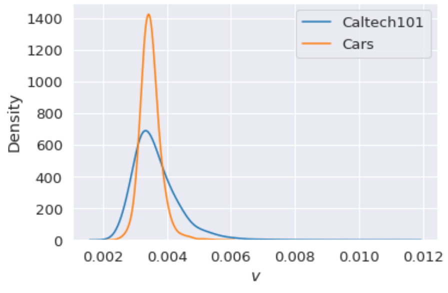


FIGURE 7.6: v on large images

Distortion. The next experiment analyzes the confidence $1 - \delta$ as a function of the distortion ϵ of our and previous bounds. We assume $\frac{m}{n} = 0.1$, $\frac{s}{m} = 0.01$. The dispersion v is chosen at the typical most likely value for each dataset. The confidence follows from Theorem 4 by Markov's inequality. The results are illustrated on 7.7. Our bounds produce very good results for all datasets with large n, thus outperforming the previous approach by several orders of magnitude in terms of confidence. Remarkably, we also obtain non-trivial bounds when n is small as opposed to the bounds from previous works.

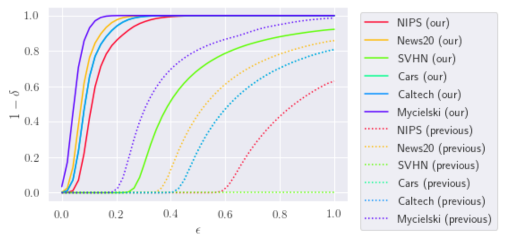


FIGURE 7.7: Confidence $1-\delta$ vs. distortion ϵ

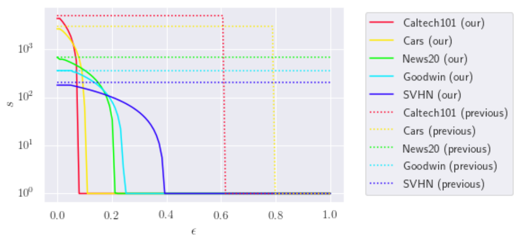


FIGURE 7.8: Sparsity s vs. distortion ϵ

Sparsity. In this experiment, we evaluate the critical value of distortion ϵ, which allows for using non-trivial sparsity $s < m$ such that the confidence $1-\delta$ is at least $\frac{3}{4}$. For each dataset, we choose as before its typical value v and fix the dimension reduction factor $\frac{m}{n} = 0.1$. The results are summarized in 7.8. Note that, for smaller values of ϵ, no $s < m$ can work, which produces flat segments $s = m$. Our bounds offer a non-trivial sparsity s for much smaller distortions, and quickly achieve $s = 1$.

Dimensionality. In the last experiment, we evaluate the minimal non-trivial dimension m. We again consider a fixed sparsity of $\frac{s}{m} = 0.1$ and choose the typical dispersion v for each dataset. Then, for various values of ϵ, we compute the smallest m which still yields a confidence of $1-\delta$ of $\frac{3}{4}$. The results, illustrated in 7.9, show that our bounds are better by 10 times or more.

Multiple data points. So far the experiments covered the performance on one input vector at a time only; the case of multiple data points reduces to the former one by scaling the confidence accordingly (union bound), where we again compute the smallest m which still yields a confidence $1-\delta$ of $\frac{3}{4}$ over all points. The result shows the expected logarithmic dependency of the dimensionality m with respect to the data size, as shown in 7.10.

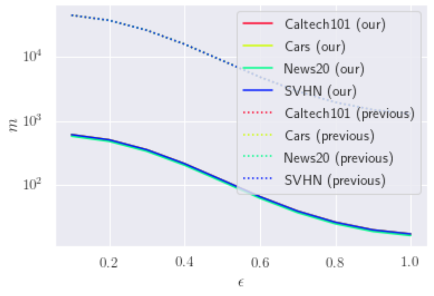


FIGURE 7.9: Dimensionality m vs. distortion

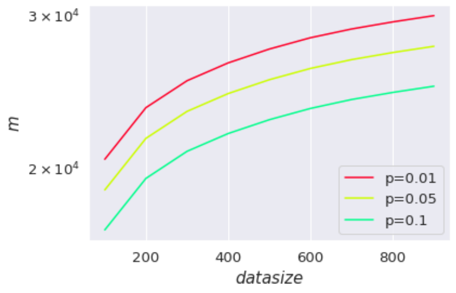


FIGURE 7.10: Dimensionality m vs. data size

7.6 Final Remarks

We have developed a framework for sparse random projections that offers provable guarantees and exhibits significant numerical enhancements over existing methods. Our superiority compared to previous approaches has been demonstrated across a diverse range of synthetic and real-world datasets. The intuition underlying our improved bounds represents a novel contribution, as it enables the generation of random embeddings that are more accurate and efficient than conventional techniques. Furthermore, the innovative inequalities driving our advancements have broader implications and are of substantial interest for various statistical inference applications. This framework not only advances the state of the art in sparse random projections but also provides a foundation for further exploration and development in the field of dimensionality reduction and related applications.

Chapter 8

Exact Non-Oblivious Performance of Rademacher Random Embeddings

In this chapter, we revisit the performance of Rademacher random embeddings and establish novel statistical guarantees that are numerically sharp and non-oblivious with respect to the input data. More specifically, our result is the Schur-concavity property of Rademacher random embeddings with respect to the input data. This offers a novel geometric perspective on the performance of random projections while improving quantitatively on bounds from previous works. This means that our methods can provide better statistical guarantees than traditional methods, even when working with non-oblivious data. This non-oblivious analysis is a novelty compared to the techniques from the previous work and bridges the frequently observed gap between theory and practice.

8.1 Main Result

In the following result, we provide the promised numerically sharp, non-oblivious and geometrically insightful bounds for Rademacher random projections. In the (particularly interesting) case of sparse input, we obtain more explicit formulas involving binomial distributions.

Theorem 4 (Sharp Moment Bounds for Rademacher Random Projections). *Let Φ be sampled according to the Rademacher scheme, and define the distortion as*

$$E(x) \triangleq \frac{\|\Phi x\|^2}{\|x\|^2} - 1. \tag{8.1}$$

Then the following holds true:

(a) *$E(x)$ has moments that are Schur-concave polynomials in (x_i^2)*

(b) *$E(x)$ is moment-dominated by E_* defined as*

$$E_* = \frac{1}{m}\sum_{i=1}^{m}(Z_i^2 - 1) \tag{8.2}$$

where Z_i are standardized binomial r.v.s.:

$$Z_i \sim^{IID} \frac{B - \mathbf{E}B}{\sqrt{\mathbf{Var}[B]}},\ B \sim \mathsf{Binom}\left(\|x\|_0, \frac{1}{2}\right). \tag{8.3}$$

Equivalently,

$$\mathbf{E}E(x)^q \leqslant \mathbf{E}E_*^q \tag{8.4}$$

holds for $q = 2, 3, \ldots$ with equality when all components of the input x are equal.

We briefly overview the proof of Theorem 4 (see Equation 8.1): it starts by a reduction to the dimension $m = 1$, and writing the distortion as a Rademacher chaos of order 2; we then find extreme values of its moments geometrically, by means of *Schur optimization*. Finally, these extreme values can be found explicitly and efficiently by linking them to binomial moments.

8.2 Proving Schur Convexity

We present a useful framework for proving Schur convexity properties. It makes repeated use of few auxiliary facts to eventually reduce the task to a 2-dimensional problem. This is often easier than the classical approach of evaluating derivative tests.

Lemma 6. *The class of* non-negative *Schur-convex (or concave) functions forms a semi-ring.*

Lemma 7. *A multivariate function is Schur-convex (respectively, Schur-concave) if and only if it is symmetric and Schur-convex (respectively, Schur-concave) with respect to each pair of variables.*

To demonstrate the usefulness of these facts, we sketch an alternative proof of a refined version of celebrated Khintchine's Inequality, due to Efron. This refinement plays an important role in statistics, namely in proving properties of popular Student-t tests.

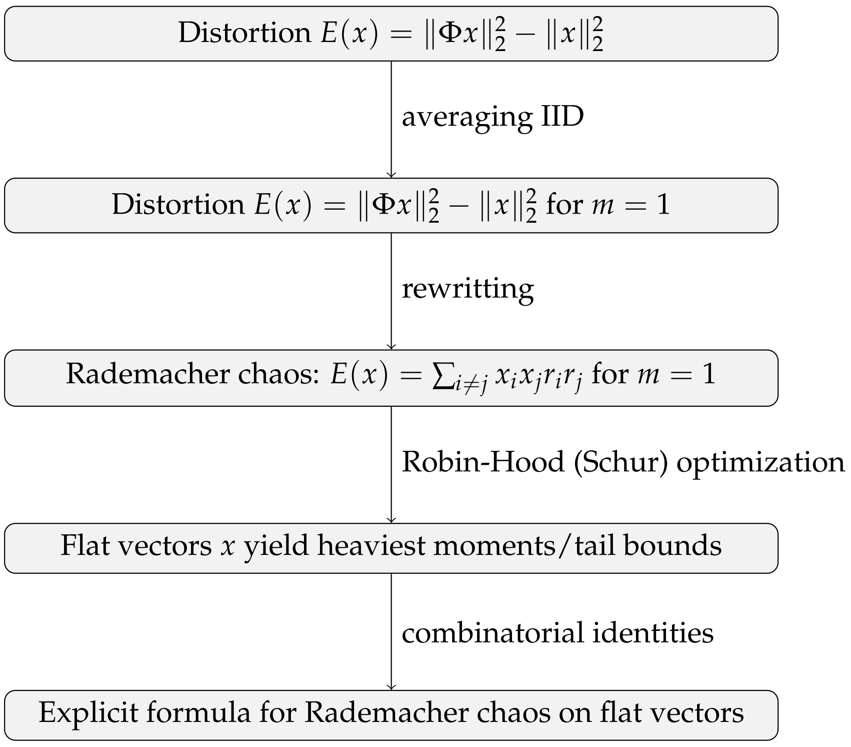

Distortion $E(x) = \|\Phi x\|_2^2 - \|x\|_2^2$
averaging IID
Distortion $E(x) = \|\Phi x\|_2^2 - \|x\|_2^2$ for $m = 1$
rewritting
Rademacher chaos: $E(x) = \sum_{i\neq j} x_i x_j r_i r_j$ for $m = 1$
Robin-Hood (Schur) optimization
Flat vectors x yield heaviest moments/tail bounds
combinatorial identities
Explicit formula for Rademacher chaos on flat vectors

FIGURE 8.1: The proof roadmap for Theorem 4.

Corollary 6 (Refined Khintchine Inequality Efron, 1968)**.** *The mapping*

$$x \to \mathbf{E}(\sum_i x_i r_i)^q$$

is a Schur-concave function of (x_i^2). *As a consequence for* $\sigma = \|x\|_2$ *we have*

$$\mathbf{E}(\sum_{i=1}^n x_i r_i)^q \leqslant \mathbf{E}\left(\frac{\sigma}{\sqrt{n}}\sum_{i=1}^n r_i\right)^q \leqslant \mathbf{E}\mathsf{Norm}(0,\sigma^2)^q.$$

Proof. The symmetry with respect to (x_i) is obvious. Applying the multinomial expansion to $(\sum_i x_i r_i)^q$, taking the expectation and using the symmetry of Rademacher random variables, we conclude that $\mathbf{E}(\sum_i x_i r_i)^q$ is polynomial in variables (x_i^2). By Lemma 7, it suffices to prove the Schur-concavity property for x_1^2, x_2^2. By the binomial formula and the independence of r_1, r_2 from $(r_i)_{i>2}$, we see that

$$\mathbf{E}(\sum_i x_i r_i)^q = \sum_k \binom{q}{k}\mathbf{E}(\sum_{i>2} x_i r_i)^{q-k}\cdot \mathbf{E}(x_1 r_1 + x_2 r_2)^k$$

is a combination of expressions $\mathbf{E}(x_1 r_1 + x_2 r_2)^k$ with coefficients $c_k = \binom{q}{k}\mathbf{E}(\sum_{i>2} x_i r_i)^{q-k}$ that are independent of x_1, x_2 and non-negative due to the symmetry of r_i. By Lemma 6, it suffices to prove that $F_k \triangleq \mathbf{E}(x_1 r_1 + x_2 r_2)^k$ is a Schur-concave function of x_1^2, x_2^2. Define

$$G_k \triangleq \mathbf{E}(x_1 r_1 + x_2 r_2)^k x_1 x_2 r_1 r_2$$

By

$$(x_1 r_1 + x_2 r_2)^k = (x_1 r_1 + x_2 r_2)^{k-2}(x_1^2 + x_2^2 + 2x_1 x_2 r_1 r_2)$$

we have

$$F_k = (x_1^2 + x_2^2)F_{k-2} + 2G_{k-2}$$

and

$$G_k = (x_1^2 + x_2^2)G_{k-2} + 2x_1^2x_2^2F_{k-2}.$$

Since $x_1^2 + x_2^2$ and $x_1^2x_2^2$ are both Schur-concave in x_1^2, x_2^2, the Schur-concavity property of F_k, G_k is proven when it is proven for $k := k-2$. By mathematical induction, it suffices to realize that $F_0 = 1, F_1 = 0, G_1 = 1, G_2 = x_1^2x_2^2$ are Schur-concave in x_1^2, x_2^2.

Let $\mathbf{1}_n$ be the vector of n ones. The first inequality follows then by

$$\frac{\sum_{i=1}^n x_i^2}{n} \cdot \mathbf{1}_n \prec (x_1^2, \ldots, x_n^2),$$

and is clearly sharp. Since $\frac{1}{n+1}\mathbf{1}_{n+1} \prec \frac{1}{n}\mathbf{1}_n$ and the Schur-concavity implies that $\mathbf{E}(\sum_{i=1}^n r_i/\sqrt{n})^q$ increases with n, the second inequality follows by the CLT. □

8.3 Rademacher Chaoses

Of independent interests are the techniques used in this work. The first result analyzes the quadratic Rademacher chaos geometrically. It is similar in the spirit of the results of Efron, 1968 and Eaton, 1970, which however concern only a first-order Rademacher chaos.

Theorem 5 (Schur-concavity of Rademacher chaoses)**.** *Let (r_i) be a sequence of independent Rademacher random variables. Then the Rademacher chaos moment*

$$\mathbf{R}_q(x) \triangleq \mathbf{E}\left(\sum_{i\neq j} x_i x_j r_i r_j\right)^q \tag{8.5}$$

is a Schur-concave function of (x_i^2) for every positive integer q.

The second result is a recipe for explicitly computing the extreme moment values:

Theorem 6 (Extreme Moments of Rademacher Chaos)**.** *For any x and $k = \|x\|_0$ the following holds:*

$$\mathbf{R}_q(x) \leqslant \mathbf{R}_q(x^*), \quad x^* = \underbrace{\left(\frac{\|x\|_2}{\sqrt{K}}, \ldots, \frac{\|x\|_2}{\sqrt{K}}\right)}_{K \text{ times}}, \tag{8.6}$$

and furthermore the explicit value of this bound equals

$$\mathbf{R}_q(x^*) = \|x\|_2^{2q} \cdot \mathbf{E}_{\bar{B}}(\bar{B}^2 - 1)^q, \tag{8.7}$$

where $\bar{B} = \frac{B-K/2}{\sqrt{K/4}}$ standardizes the symmetric binomial distribution with k trials B.

8.4 Numerical Comparison

The presented result is *numerically optimal* and *captures input sparsity*. It should be compared against the bounds from Achlioptas, 2001 and the no-go result from Burr, Gao, and Knoll, 2018, as illustrated in Table 8.1. To see that our bounds are better

Author	Result
Burr, Gao, and Knoll, 2018	$\max_x \mathbf{P}[\lvert E(x)\rvert > \epsilon] \geqslant 2\exp\left(-\frac{m\epsilon^2(1+o(1))}{4})\right)$ when $m \gg \epsilon^{-2}, n \gg 1$
Achlioptas, 2001	$\mathbf{P}[\lvert E(x)\rvert > \epsilon] \leqslant 2\exp\left(-\frac{m\epsilon^2}{4}\left(1-\frac{2}{3}\epsilon\right)\right)$
this paper	$E(x) \prec_m \frac{\sum_{i=1}^m Z_i^2 - 1}{m}$, $Z_i \sim^{IID} \frac{B-\mathbf{E}B}{\sqrt{\mathbf{Var}[B}}$, $B \sim \text{Binom}\left(\lVert x\rVert_0, \frac{1}{2}\right)$

TABLE 8.1: Bounds from this work from Theorem 4 compared with the best prior bounds (Achlioptas, 2001) and the sharp no-go results (Burr, Gao, and Knoll, 2018). Our bounds imply those from prior work by "normal majorization" arguments (see the supplementary material).

than those in Achlioptas, 2001, it suffices to use the Gaussian majorization argument to obtain a weaker bound

$$E(x) \prec_m \frac{\sum_{i=1}^m (N_i^2 - 1)}{m}$$

where N_i are independent standard normal random variables, and use known sub-gamma tail bounds for chi-square distributions (for example, those developed in the monograph on concentration inequalities (Boucheron, Lugosi, and Bousquet, 2003)).

To validate our findings, we performed the following experiments on both synthetic and real-world datasets.

8.4.1 Synthetic Dataset

Figures 8.2 and 8.3 demonstrate numerical improvements. In Figure 8.2 we can notice that the more spread-out the input (controlled by sparsity $\lVert x\rVert_0$), the more distorted the projected output (captured by $\mathbf{R}_q(x)$, the Rademacher chaos moment). Utilizing the input dispersion improves probability bounds by orders of magnitude. Note: for normalization purposes, we assume $\lVert x\rVert_2 = 1$. In Figure 8.3 capturing input-sparsity ($\ell = \lVert x\rVert_0$) improves the bounds on Rademacher random projections, as demonstrated by distortion probability tails, the input sparsity is the key: random projections are seen *less distorted when input data is sparse*.

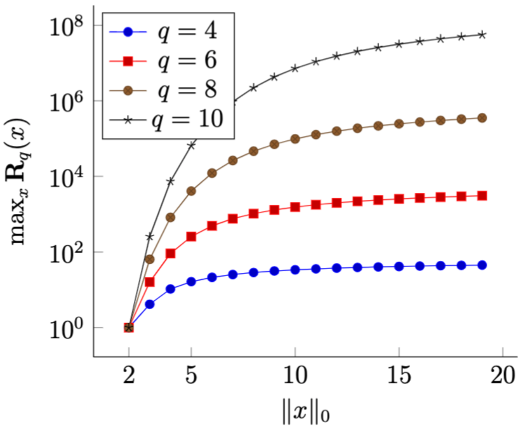


FIGURE 8.2: The input dispersion improves bounds.

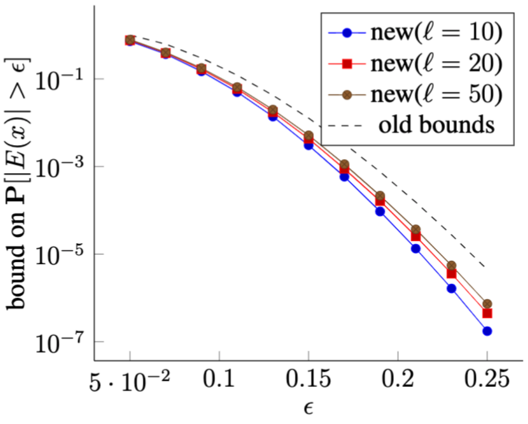


FIGURE 8.3: Capturing input-sparsity ($\ell = \lVert x\rVert_0$).

8.4.2 Real-world Datasets

Our results are validated with experiments performed on real-world datasets from the SuiteSparse Matrix Collection (formerly the University of Florida Sparse Matrix Collection) available at `https://sparse.tamu.edu/`. For these experiments, we selected matrices from ML datasets. In figure 8.4 the distortion measured w.r.t. the density of the embeddings shows that sparse data result in better bounds and proves that Rademacher projections are superior to sparse ones. In Figure 8.5 measuring the distortion tail probability on real-world datasets confirms our theoretical findings: capturing the input sparsity improves the bounds on Rademacher random projections. The datasets, from left to right, are displayed from the most sparse (mnist) to the least one (glass).

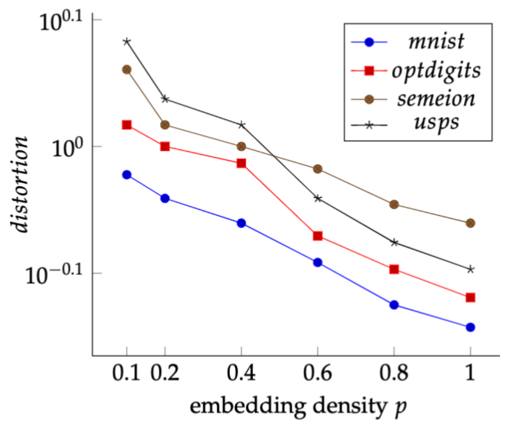


FIGURE 8.4: The distortion measured w.r.t. the density

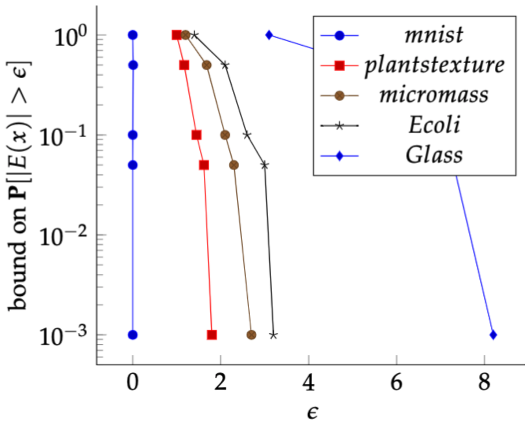


FIGURE 8.5: The distortion tail probability on real-world datasets

8.5 Final Remarks

We revisited the performance of Rademacher random projections, connecting the statistical guarantees with the input structure: for spreadness and, a special case, sparsity. The main result of this paper proves Schur-concavity properties, which makes the bounds numerically sharp and data aware (non-obliviuos) while giving a geometric perspective to the performance of the projections. We benchmarked our bounds both theoretically and empirically by measuring the distortion of the projected vectors against the original input data. As a result, dense projections are preferred, and they work incredibly well with sparse input data. We believe that our findings are of broader interest for a variety of statistical-inference applications.

Part IV

Relation Extraction

Chapter 9

Introduction

Open Information Extraction (OpenIE) is a specific approach to IE that aims to identify and extract relations and their corresponding arguments from text without relying on predefined schemas or domain-specific templates. OpenIE has a wide range of applications, including but not limited to knowledge base construction, text analytics, question answering (QA), and semantic search. It can be used to automatically construct and expand large-scale knowledge bases, such as knowledge graphs, by extracting structured information from vast amounts of unstructured text. OpenIE aids in transforming unstructured text into structured data, facilitating advanced analytics and visualization for better decision-making. Key aspects of OpenIE include named entity recognition (NER), named entity disambiguation (NED), relation extraction (RE), co-reference resolution, scalability, and knowledge representation.

NER is a fundamental task in OpenIE that involves identifying and classifying named entities, such as people, organizations, locations, and dates, within the text. NER is crucial for understanding the context and extracting relationships between entities. NED is the process of resolving the ambiguity surrounding named entities, especially when they have multiple possible meanings or referents. This task helps ensure the correct interpretation of entities and their relationships in the text. Co-reference resolution is the process of identifying and linking different textual expressions that refer to the same entity or concept. This task is essential for accurately understanding and connecting relationships between entities, even when they are mentioned in different parts of the text.

RE is a technique that focuses on structuring natural-language text by detecting potential semantic connections between two or more real-world concepts, typically referred to as "entities". Relations are assumed to fall into predefined categories and to hold between entities of specific types. This process is vital for understanding the relationships and context within a given text and plays a crucial role in various IE tasks. For example, the SPOUSE relation may exist between two entities of type "Person", while instances of the CEO relation would link entities of type "Person" and "Organization", respectively. These relations allow for a more structured representation of the information within the text, making it easier to analyze and query.

Being a sub-discipline of IE, extracting labeled relations can also help boost the performance of various IE downstream tasks, such as knowledge-base population (KBP) (Trisedya et al., 2019; Gardner and Mitchell, 2015) and QA (Wang et al., 2012; Xu et al., 2016). In the context of KBP, RE serves as a vital component in the automatic construction and expansion of knowledge bases. By identifying and categorizing the relationships between entities, RE enables the enrichment of existing knowledge bases with new facts and connections. This process significantly contributes to the development of large-scale knowledge graphs and semantic networks, which are essential for various AI applications, such as semantic search and recommendation systems. For QA systems, RE is crucial for providing relevant and accurate answers to users' queries. By understanding the relationships between entities in a text, QA systems can efficiently retrieve the most pertinent information to address a specific question. This capability is particularly important when dealing with complex queries that involve multiple entities and relationships, as it enables the system to provide a more comprehensive and contextually accurate response.

In Part IV of this thesis, we conduct a comprehensive analysis of different design combinations for the task of RE. Our main focus is on integrating OpenIE with two types of embeddings, namely context-free and context-sensitive, to examine the strengths and limitations of each combination. Our primary hypothesis is that OpenIE can enhance even context-sensitive language models like BERT by breaking down complex sentences into multiple clauses, each representing the target relation more precisely than the original sentence. Our objective is to make significant advancements in Web-scale RE. To achieve this, we utilize the OpenIE approach to model and classify relational phrases, taking advantage of shorter clauses that more accurately capture the target relation compared to potentially lengthy and complicated input sentences. We employ distant supervision to transfer labels from annotated corpora to OpenIE extractions, thereby minimizing manual labeling efforts needed for training and fine-tuning the underlying models. Additionally, we explore few-shot training, which can further decrease the amount of labeled training examples to fewer than 20 per relation, while still delivering satisfactory results in many instances. Our detailed experiments are conducted on two annotated RE corpora, *KnowledgeNet* (Mesquita et al., 2019) and *FewRel* (Han et al., 2018), using Wikidata as a backend knowledge base (KB) in conjunction with various state-of-the-art (both context-free and context-sensitive) language models. These models include a basic Word2Vec model with annotated and disambiguated Named Entities (NEs), BERT, RoBERTa, AlBERT, SETFIT, and their "distilled" versions. The results show that many of our combined approaches significantly improve upon the best-known results for both KnowledgeNet and FewRel, demonstrating the potential of our method for advancing the field of RE.

Chapter 10

Background and Related Work

Distant Supervision vs. Few-Shot Learning. Extracting labeled relations from previously unseen domains usually requires large amounts of training data. Manually annotated corpora are relatively small due to the amount of work involved in their construction. To this end, *distant supervision* (Mintz et al., 2009) may help to alleviate the manual labeling effort but training data, which may serve as the basis for distant supervision, is only available for relations covered by an already-existing KB such as Yago (Suchanek, Kasneci, and Weikum, 2007), DBpedia (Lehmann et al., 2015) or Wikidata (Vrandečić and Krötzsch, 2014). To overcome this limitation, Gashteovski et al., 2020 manually evaluates the semantics of alignments between OpenIE triples and the facts in DBpedia. Distant supervision is used as a first step to compare facts that have the same (or at least similar) arguments. We, in contrast, use distant supervision to transfer the labels from the annotated corpora to the OpenIE extractions, thereby creating an annotated set of clauses which can then be used for training. Moreover, for cold-start KBP settings (KBP, 2017), *few-shot learning* (Wang et al., 2021b) has recently evolved as an interesting alternative to distant supervision. In few-shot-training for KBP (or more classical tasks like text classification), an underlying language model such as BERT or SBERT (Devlin et al., 2019; Reimers and Gurevych, 2019) is augmented by an additional prediction layer for the given labeling task which is then retrained by very few samples. Here, often 20–50 examples for each label are sufficient to achieve decent results. However, all of these approaches for KBP focus on labeling and training trade-offs for the given input text, while other–perhaps more obvious—options, namely to exploit syntactic and other structural clues based on OpenIE, NER and NED, are at least as promising as these training aspects in order to further improve prediction accuracy. These additional bits and pieces of information can be integrated into the learning process to develop more accurate and efficient models for extracting relevant information from textual data. For example, incorporating syntactic parsing, entity recognition, and disambiguation could provide a more comprehensive understanding of the relationships between entities and the context in which they appear. This enriched information may help improve the overall performance of RE tasks and further advance the capabilities of ML models in understanding and processing natural language. By exploring both distant supervision and few-shot learning techniques and integrating additional structural information from OpenIE, NER, and NED, there is potential to develop more robust and accurate models for extracting valuable insights from unstructured textual data. This will, in turn, contribute to the broader field of IE and NLP.

Domain-Oriented vs. Open Information Extraction. OpenIE (Carlson et al., 2010; Etzioni et al., 2004; Banko et al., 2007; Fader, Soderland, and Etzioni, 2011) expresses an alternative text-structuring paradigm compared to the more classical, domain-oriented IE techniques (Trisedya et al., 2019; Gardner and Mitchell, 2015): it transforms sentences into a set of *⟨arguments – relational phrase⟩* tuples without labeling the relational phrases explicitly or requiring its arguments to be of particular entity types. Consider, for instance, the sentence: *"In 2008 Bridget Harrison married Dimitri Doganis"*. From an RE perspective, it would be represented as: ⟨Bridget Harrison; SPOUSE; Dimitri Doganis⟩. Its OpenIE[1] counterpart would decompose the input sentence into two tuples: ⟨Bridget Harrison; married; Dimitri Doganis⟩ and ⟨Bridget Harrison; married Dimitri Doganis In; 2008⟩. Intuitively, the two representations capture the same semantic message of a marriage relationship between Bridget Harrison and Dimitri Doganis. Furthermore, OpenIE produces additional informative tuples describing, e.g., temporal or collocational aspects of the relation via adverbial phrases, which however may not necessarily have a corresponding canonicalized form. Importantly, OpenIE extracts relational phrases along with the original sentences' arguments, thus structuring the input text without loss of information. All these characteristics make OpenIE a useful intermediate representation for a number of downstream IE tasks that impose further structuring or normalization (Mausam, 2016; Martínez-Rodríguez, López-Arévalo, and Ríos-Alvarado, 2018; Lockard, Shiralkar, and Dong, 2019). Moreover, OpenIE's adaptability proves particularly advantageous when applied to large-scale, diverse datasets where domain-specific knowledge may be limited or hard to define. Its ability to handle ambiguity and provide richer contextual information is crucial for complex tasks, leading to more robust and accurate IE.

Word Embeddings vs. Language Models. In the past few years, *word embeddings* (Bojanowski et al., 2017; Mikolov et al., 2013; Pennington, Socher, and Manning, 2014) found their applications and proved to be efficient in a wide range of IE tasks. Word embeddings represent text as dense vectors in a continuous vector space. Traditional word embeddings, such as Word2Vec (Mikolov et al., 2013) and FastText (Bojanowski et al., 2017), are lightweight and conveniently fast at training and inference time. However, being static (each word in the corpus is represented by the same vector, regardless of its context), these embeddings have limited ability to capture a word's changing meaning with respect to different contexts. On the contrary, recently trained, large-scale *language models* (LMs), such as BERT (Devlin et al., 2019), ELMO (Peters et al., 2018) or GPT-3 (Radford et al., 2019), extend the approach by generating dynamic embeddings, where each word's representation depends on its surrounding context, thus pinning down particular meanings of polysemic words and entire phrases. Despite the differences, both types of embeddings allow to quantitatively express semantic similarities between words and phrases based on the closeness of their respective vectors in the vector space. Furthermore, other linguistic components such as syntactic dependency trees or OpenIE-style tuples can be used to train or fine-tune various embedding models with positive impact on more advanced IE tasks such as text comprehension, similarity and analogy (Levy and Goldberg, 2014; Stanovsky, Dagan, and Mausam, 2015), semantic role labeling (SRL) (Marcheggiani et al., 2017), as well as RE and QA (Sachan et al., 2021).

[1]Based on OpenIE 5.1: `https://github.com/dair-iitd/OpenIE-standalone`

Chapter 11

Enriching Relation Extraction with OpenIE

In this chapter, we present our three principal strategies for classifying relational paraphrases (and entire clauses) obtained from OpenIE into canonical relations over a predefined KB schema. We next provide a brief overview of the three approaches, before we describe them in more detail in the following sections.

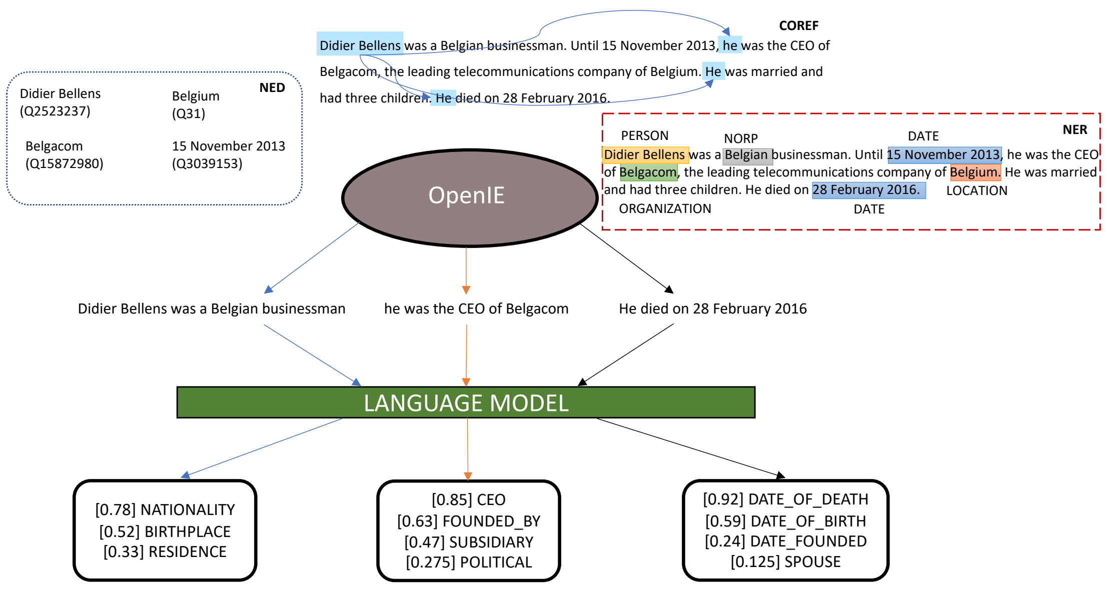
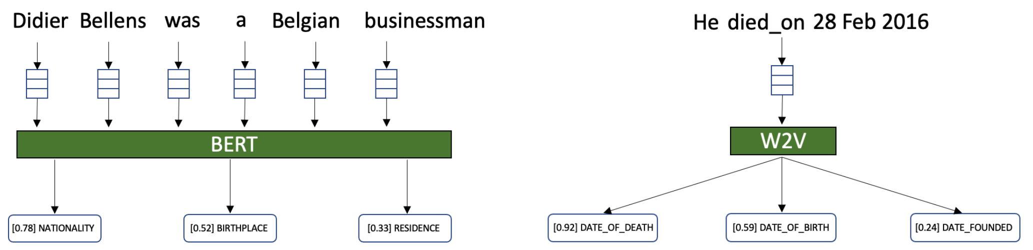

COREF
Didier Bellens was a Belgian businessman. Until 15 November 2013, he was the CEO of Belgacom, the leading telecommunications company of Belgium. He was married and had three children. He died on 28 February 2016.
NED
Didier Bellens (Q2523237)
Belgium (Q31)
Belgacom (Q15872980)
15 November 2013 (Q3039153)
NER
PERSON
NORP
DATE
Didier Bellens was a Belgian businessman. Until 15 November 2013, he was the CEO of Belgacom, the leading telecommunications company of Belgium. He was married and had three children. He died on 28 February 2016.
LOCATION
ORGANIZATION
DATE
OpenIE
Didier Bellens was a Belgian businessman
he was the CEO of Belgacom
He died on 28 February 2016
LANGUAGE MODEL
[0.78] NATIONALITY
[0.52] BIRTHPLACE
[0.33] RESIDENCE
[0.85] CEO
[0.63] FOUNDED_BY
[0.47] SUBSIDIARY
[0.275] POLITICAL
[0.92] DATE_OF_DEATH
[0.59] DATE_OF_BIRTH
[0.24] DATE_FOUNDED
[0.125] SPOUSE

FIGURE 11.1: System overview.

Fine-Tuning Language Models. Our primary strategy for achieving this goal involves developing a dedicated RE model by training it on a corpus of annotated sentences. Subsequently, we will utilize this RE model to accurately predict the relations present within previously unseen sentences and clauses. To accomplish this, we begin by employing a large-scale, pretrained LM, such as BERT or one of its variants, which has demonstrated remarkable performance in various NLP tasks. We then augment the BERT model with an additional classification layer, enabling us to fine-tune the model specifically for the RE classification task. The fine-tuning process essentially involves adapting the pretrained model to better identify and classify relations among entities in a given context. BERT, being a general-purpose and context-sensitive LM, has been trained on an extensive dataset comprising billions of input sentences, thereby providing a robust foundation for our RE model. By leveraging BERT's comprehensive pretraining, we anticipate that our approach will yield remarkable results in RE tasks, even with a relatively small amount of annotated sentences required for *fine-tuning* the classification layer. This is primarily due to BERT's inherent ability to understand complex language patterns and dependencies, which enables it to adapt and learn the nuances of semantic meaning more efficiently.



Context-Free Relation Signatures. As an alternative and more straightforward approach, we also explore the implementation of a context-free baseline for RE using a clause-based Word2Vec model. This method eliminates the need for annotated sentences during the training phase, making it a more accessible and less resource-intensive option. To accomplish this, we train the Word2Vec model on a domain-specific corpus, such as Wikipedia articles, using an unsupervised learning approach. This enables the model to learn semantic relationships between words based on their co-occurrence patterns in the text. To apply the Word2Vec model to RE, we first aggregate individual word vectors to create *relation signatures* for a predefined set of target relations. Relation signatures essentially capture the semantic representation of the relations we aim to identify. Next, we employ OpenIE techniques to obtain relational paraphrases from the input text. We then quantitatively evaluate the vector similarities between the generated relation signatures and the relational paraphrases obtained from OpenIE. By comparing these similarities, we can assess the performance of the clause-based Word2Vec model in identifying and classifying the target relations within the given text. High vector similarity scores suggest that the Word2Vec model has effectively captured the semantic essence of the target relations.

Though this method may lack the contextual understanding offered by more advanced models like BERT, it presents a viable, less complex alternative for RE. Furthermore, the unsupervised nature of the Word2Vec training process makes it particularly well-suited for scenarios where access to annotated data is limited or unavailable. To enhance the performance of this approach, we can consider incorporating additional techniques, such as clustering algorithms or dimensionality reduction methods, to refine the extraction and classification of relations.

Contextualized Relation Signatures. Our third and final approach integrates the concepts from the previous two methods by exploring the use of BERT-like models in a feature-based manner for RE. In this hybrid approach, we harness the power of contextualized embeddings derived from large-scale pretrained models, such as BERT, and combine them with a *contextualized form of relation signatures*. To create these contextualized relation signatures, we first manually provide a few training sentences as input for each target relation. These sentences serve as examples that help the model understand the characteristics and nuances of the target relations.

Next, we utilize the pretrained BERT-like model to generate contextualized embeddings for the words in these input sentences. These embeddings capture the rich semantic and syntactic information inherent in the text, taking into account the context in which the words appear. Once we have obtained the contextualized embeddings, we aggregate them to form relation signatures that are representative of the target relations in a context-sensitive manner. By combining the strengths of BERT-like models and the concept of relation signatures, this approach aims to achieve improved performance in RE tasks. The contextualized embeddings offer a more in-depth understanding of the text, while the relation signatures provide a systematic way to represent and identify the target relations.

11.1 Fine-Tuning Language Models for Relation Extraction

In the context-aware approach, we add a single fully connected layer for the classification task on top of the last layer of an otherwise task-agnostic pre-trained LM such as BERT or one of its variants (see below for details). The size of the added layer is equal to the number of classification labels. Fine-tuning the model then consists of training the new layer's weights over a task-specific annotated dataset. For our RE task, a typical annotated example would consist of (1) an *input sentence*, (2) the *entity pair* corresponding to the sentence's subject and object, and (3) the *target relation* as label. For example, the sentence *"After five successful albums and extensive touring, they disbanded after lead vocalist Sandman died of a heart attack onstage in Palestrina, Italy, on July 3, 1999."* would then be encoded into the clause (amongst others) ⟨ {Sandman; July 3, 1999}, DATE_OF_DEATH ⟩. Note, however, that our approach to the relation classification differs from the established setup in one important way: while many works on the topic capitalize on the importance of relational argument (entity) representation (Soares et al., 2019; Zhou and Muhao, 2021; Boros, Moreno, and Doucet, 2021; Zhang et al., 2019a), we completely exclude entity-related information (obtained from common NER/NED toolkits) during the training, thereby delegating the task of extracting the relational argument to the OpenIE step. Therefore, an adjusted input for tuning the model is reduced to pairs made of (1) *input clause* and (2) *target relation*. This streamlined input structure allows our model to focus on identifying and classifying relations without being influenced by entity-specific information. By excluding entity details, we aim to create a more flexible and versatile RE model that can effectively generalize across various input types and domains.

We utilized a selection of pretrained LMs from the BERT family for fine-tuning our RE task. Here, we briefly introduce each model and provide the rationale behind our choices:

- *bert-base-uncased* (Devlin et al., 2019) is a **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (BERT) model consisting of 12 layers of transformers and is case-insensitive. BERT has become a default "baseline" for numerous NLP tasks involving general-purpose pretrained models.
- *distilbert-base-uncased* (Sanh et al., 2020) is a variant of BERT pretrained on the knowledge distillation principle, which involves transferring knowledge from one or more large models to a single smaller one. DistilBERT has demonstrated performance nearly on par with the full-size BERT, while using fewer resources and being faster during training and inference.
- *xlnet-base-cased* (Yang et al., 2019) is an autoregressive model that improves upon BERT's ability to learn semantic dependencies between sentence components. This

Model	Layers	Hidden	#Parameters
bert-base-uncased	12	768	110M
distilbert-base-uncased	6	768	66M
xlnet-base-cased	12	768	110M
roberta-base	12	768	125M
distillroberta-base	6	768	82M
albert-base-v1	12	768	11M
setfit	12	768	110M

TABLE 11.1: Overview of models used in the context-aware RE setting.

feature may provide an advantage during inference when using a model fine-tuned on entire sentences to classify OpenIE-style clauses.

- *roberta-base* (Liu et al., 2019) has been trained on a much larger corpus than BERT and is further optimized. It also incorporates a dynamic token-masking objective in the model's training to enhance its sensitivity and robustness to context. It has exhibited the highest accuracy (compared to BERT and XLNet) on the Recognizing Textual Entailment (RTE) task (Liu et al., 2019; Wand et al., 2019), which is closely related to RE. This motivates our interest in using this model.
- *distilroberta-base* is a distilled version of the RoBERTa-base model that follows the same training procedure as DistilBERT. On average, DistilRoBERTa is twice as fast as Roberta-base. It is primarily aimed at being fine-tuned on tasks that use the whole sentence (potentially masked) to make decisions, such as sequence classification, token classification, or question answering.
- *albert-base-v1* (Lan et al., 2020) introduces a sentence-order prediction (SOP) training objective, primarily focusing on inter-sentence coherence—a property we expect to benefit from when transferring knowledge learned from entire sentences to OpenIE clauses.
- *setfit* (Tunstall et al., 2022) stands for **Se**ntence **T**ransformer **Fi**ne-**t**uning and is a recent model designed for few-shot text classification. It is trained on a small number of text pairs (8, to be precise) in a contrastive Siamese manner. The resulting model is then employed to generate rich text embeddings, which are used for training a classification task.

By leveraging the unique strengths of each of these models, we aim to create a robust and accurate RE system capable of effectively identifying and classifying relations in various contexts and domains. In doing so, we hope to enhance the overall performance and versatility of our approach, enabling it to excel in a wide range of RE tasks.

11.2 Using Context-Free Relation Signatures for Relation Extraction

For the context-free approach, we start from a large dump of English Wikipedia articles which we process with a pipeline consisting of ClausIE (Corro and Gemulla, 2013) for clause decomposition, Stanford CoreNLP (Manning et al., 2014) and AIDA-light (Nguyen et al., 2014) for NER and NED, respectively. This pipeline yields an initial amount of 190 million clauses, from which we distill 13.5M binary relations of the form ⟨*subject; relational phrase; object*⟩ (thereby keeping only clauses that have exactly one named entity both as subject and object as well as a verbal phrase as predicate).

Following (Fader, Soderland, and Etzioni, 2011), we apply regular expressions on the verbal phrases to identify patterns of the form *verb* | *verb + particle* which should cover $\approx$ 85% of the verb-based relations in English. To further normalize the extracted verbal phrases, we use part-of-speech (POS) and lemmatization information: for all but passive verbs, we substitute their inflections with the respective lemma. This way, we are able to distinguish between active and passive voices of otherwise identical verbs (which usually indicate inverse relations of each other). After the above cleaning and normalization steps, our overall representation of a clause is of the form: $\langle entity_1, verb + particle, entity_2 \rangle$ with the additional condition that $entity_1$ and $entity_2$ should not be equal[1].

We next embed the clauses into their word vector representations. Specifically, we consider two encoding schemes:

(i) by exploiting the *compositionality* of word vectors:

$$\vec{V}_{verb} + \vec{V}_{particle}$$

(ii) by creating *bigrams* of verbs and particles for the most frequent relational paraphrases in the corpus (e.g., *work_at, graduate_from, born_in*):

$$\vec{V}_{verb\_particle}$$

For the latter bigram-based encoding, we treat bigrams for the prepositional verbs as additional dictionary entries before a Word2Vec model is trained on the clauses. This choice is motivated by considering that, particularly for knowledge discovery, particles (i.e., prepositions) may give a crucial insight on the possible type of the entities involved in the relation. As we will see in Section 11.5, we leverage both aforementioned techniques in comparison.

To train the models under (i) and (ii), we use Word2Vec "skip-gram model" implementation provided by the Gensim (Rehurek and Sojka, 2011), with the window size 2, and negative sampling as loss function.

In this context-free approach, we further aggregate the vector representation of each target relation by including also synonyms for these relations provided by an additional backend KB. As an example, let

$$S_{\texttt{P571}} = \{\text{“date founded”}, \text{“date created”}, \ldots, \text{“established”}\}$$

denote the *Wikidata*[2] *synonyms* provided for the relation `P571`. Then, the vector for its corresponding *relation signature* is computed as follows

$$\vec{V}_{\texttt{P571}} = \frac{1}{|S_{\texttt{P571}}|} \sum_{synonym \in S_{\texttt{P571}}} \vec{V}_{synonym}$$

[1]We found about 5% of clauses in the corpus to represent reflexive relations, i.e., with the subject and object referring to the same entity. From our observations, we could not derive meaningful KB facts from these relations. We therefore removed such reflexive relations from the corpus.

[2]`www.wikidata.org`

where we use the *artihmetic mean* (in analogy to Gensim[3]'s `w2v.most_similar` function to retrieve similar vectors for a set of positive examples) in order to aggregate a set of such synonyms into a single vector.

Since the target relations considered in our experiments correspond to Wikidata (Vrandečić and Krötzsch, 2014) properties, we use Wikidata as backend KB and consider the English parts of the *"Also known as"* sections of the respective properties as source for the synonymous relational phrases. To leverage our Word2Vec model, we again normalise the property name and its synonyms by following the steps described above (before vectorization). By default, we then use bigrams of verb lemmas and their particles for the aggregation of the vectors into relation signatures. However, if a bigram is not found in the model's vocabulary, we fall back to our compositional encoding also for the respective synonyms.

11.3 Using Contextualized Relation Signatures for Relation Extraction

For our third approach, we further explore the concept of using relation signatures to represent relations. However, we now generate *contextualized relation signatures* in a slightly different manner. We model a relation using a small set of structured natural language units that convey self-contained meaning, such as sentences or clauses. Consequently, we modify the procedure of signature generation as follows: for each relation, (i) 5 units (clauses or sentences) are manually sampled from an underlying labeled corpus; (ii) units are embedded into a LM [4]; (iii) a normalized average of the embeddings represents the signature for the respective relation label. Analogous to the context-free approach, during testing, units to be labeled are embedded into the same model, and the resulting vectors are compared to the vectors of relation signatures using cosine similarity. Unlike the context-free approach, here, clauses are not reduced to their relational phrase components but are instead considered as whole units.

This heuristic is purposefully implemented to resemble few-shot learning techniques in a feature-based manner. It offers two major advantages for our goal of scaling RE: it requires a minimal amount of additional labeling effort, and it enables the addition of new target relations on-the-fly. By leveraging these strengths, we aim to create a more efficient and adaptable RE system that can effectively identify and classify relations with minimal manual intervention, thereby facilitating a more scalable and flexible approach.

11.4 Text-Processing Pipeline

Before we apply the methods described above to an actual corpus, we first preprocess the corpus with a typical NLP pipeline consisting of 5 steps: sentence splitting, tokenization and part of speech tagging[5], NER (Schweter and Akbik, 2020) and NED (Li et al., 2020) (both optional[6]), coreference resolution (CR) (Joshi et al., 2020) and OpenIE (Mausam, 2016). Once all the steps are performed, we align the obtained

[3]https://radimrehurek.com/gensim/

[4]https://github.com/cyk1337/embedding4bert

[5]https://spacy.io/

[6]In the scope of this work we focus on the relational predicates and use named entity recognition (NER) to ensure that the subject and object entity types are compatible with the backend KB property.

annotations with the original sentences to obtain a corpus representation that is of the form shown in Figure 11.1.

Since we aim at leveraging OpenIE for RE, a natural choice for tuning a language model would be to use a set of labeled clauses as input, where the labels correspond to the target relations for a given dataset. To the best of our knowledge, such datasets are not available at clause level. We therefore create one in a distant-supervision manner as follows: given a sentence from the training corpus with the relation specified as a pair of ⟨*subject*, *object*⟩ entities and a label, we label the clauses obtained from that sentence provided that their subject and object correspond to the subject and object entities from the labeled sentence.

> PLACE_OF_BIRTH(<u>Barack Obama</u>, <u>Honolulu</u>)
> OpenIE clause: *<u>Barack Obama</u> was born in <u>Honolulu</u>*
> (*<u>Barack Obama</u> was born in <u>Honolulu</u>*, PLACE_OF_BIRTH)

Note that we can only safely align clauses to labeled relations when the former correspond to unambiguous extractions with exactly one entity in the subject and object, respectively.

It is generally believed that a clause's predicate is the main carrier of the relation type (Fader, Soderland, and Etzioni, 2011; Farima, Bhutani, and Jagadish, 2022; Lockard, Shiralkar, and Dong, 2019). We observe though that it is not necessarily the case. Consider the following clauses: (a) ⟨John Deane Spence; was; a *British Conservative Party politician*⟩ and (b) ⟨Eccles; served as; a *Labour Party member* of Manchester City Council⟩. In these examples, the clause's object contains both – the cue of the relation type (*Political Affiliation*) and the relational argument. In other cases, the predicate contains both – a relation type indicator and relational argument: (c)⟨ he; *joined the Communist Party* In; 1923.⟩ Note that the clause's object here is irrelevant for the relation. These examples show that there is no "consistency" in the OpenIE format that we could rely on when translating OpenIE extractions to RE. Gashteovski, Gemulla, and Corro, 2017 addresses the problem of clause normalization by applying a series of transformations driven by a morpho-syntactic analysis of the produced clauses.

On the contrary, we consider the clauses produced by an OpenIE system as potentially noisy relations. This is the reason why we consider each OpenIE extraction in its entirety, as a short sentence, and use each for the relation prediction individually. In this way, we exploit an implicit semantic connection between the clause's elements that synergically express the relational meaning of a clause. As such, our approach does not suffer from falling into one of the two extremes as indicated in (Zhang et al., 2019a): neither performing an instance-level inference relying on embedding for ⟨*subject*, *object*⟩ pairs thus being unable to handle unseen entities; nor performing a predicate-level mapping thus ignoring background evidence from individual entities. Rather, we examine to what extent task-agnostic pretrained LMs are able to transfer learned signals from longer sentences to short facts. We therefore use a portion of labeled sentences to tune the model and use it for clause classification.

TABLE 11.2: Properties.

Property	Entity Type Signature	Facts
DATE_OF_BIRTH	PER-DATE	761
DATE_OF_DEATH	PER-DATE	664
RESIDENCE	PER-LOC	664
BIRTHPLACE	PER-LOC	1137
NATIONALITY	PER-LOC	639
EMPLOYEE_OF	PER-ORG	1625
EDUCATED_AT	PER-ORG	951
POLITICAL_AFF.	PER-ORG	635
/ CHILD_OF	PER-PER	888
SPOUSE	PER-PER	1338
DATE_FOUNDED	ORG-DATE	500
HEADQUARTERS	ORG-LOC	880
SUBSIDIARY_OF	ORG-ORG	544
FOUNDED_BY	ORG-PER	764
CEO	ORG-PER	643

11.5 Experiments

In this section, we describe the experiments and datasets we used to evaluate our proposed methods, which are based on two commonly used RE benchmarks: *KnowledgeNet* and *FewRel*. Both collections were preprocessed by our NLP pipeline described in section 11.4.

KnowledgeNet (KN) (Mesquita et al., 2019) is a dataset for populating a KB (here: Wikidata) with facts expressed in natural language on the Web. We selected KN as our primary benchmark because it provides facts in the form of ⟨*subject*, *property*, *object*⟩ triplets as sentence labels. The documents in the first release of KN are either DBpedia abstracts (i.e., the first paragraph of a Wikipedia page) or short biographical texts about a person or organization from the Web. 9,073 sentences from 4,991 documents were chosen to be annotated with facts corresponding to 15 properties about a particular property of interest (see Table 11.2 for the detailed list). In total, KN comprises 13,425 facts from 15 properties. Only 23% of the facts annotated in their release are actually present in Wikidata.

FewRel (Han et al., 2018) is a popular benchmark for few-shot RE, consisting of 70,000 sentences over 100 relations (divided in 64 for training and 36 for testing purposes). This dataset is meant to be extremely competitive even for the most advanced models for RE, and for this reason we employed it also in our work. However, we did not use FewRel as it was originally conceived in its typical few-shot setting, but we randomly split the sentences per relation into separate training (75%) and testing (25%) sets.

11.5.1 Baseline Approaches

We now evaluate the three different approaches of sections 11.1, 11.2, and 11.3. Particularly, for the fine-tuned BERT models, we created different combinations of training and testing sets as follows.

- **Baseline 1: Clauses + LM.** We use OpenIE to extract clauses from sentences. Both fine-tuning and prediction of the LM were then performed on *clauses*.

- **Baseline 2: Mixed + LM.** Fine-tuning of the LM was performed on *sentences*, while prediction was then performed on *clauses*.
- **Baseline 3: Sentences + LM.** Both fine-tuning of the LM and inference were performed on *sentences*.
- **Baseline 4: Clauses + W2V.** We use OpenIE to extract clauses from sentences. Context-free relation signatures (as described in section 11.2) based on the simple Word2Vec model were then used to infer the target relation.
- **Baseline 5: Clauses + feature-based BERT**. For the feature-based approaches, we applied the same three combinations as for Baselines 1, 2, and 3. Thus swapping the fine-tuning phase with the *relation signature* construction which was generated by using only 5 randomly drawn samples per relation. For this baseline, both *relation signature* construction and inference were performed on *clauses*.
- **Baseline 6: Mixed + feature-based BERT**. The *relation signature* construction was generated using 5 randomly drawn *sentences* per relation, while prediction was performed on *clauses*.
- **Baseline 7: Sentences + feature-based BERT**. Both the *relation signature* construction and prediction were performed on *sentences*.

11.5.2 Evaluation

RE inherently resembles a *multi-class prediction* task. For KN, a particularity of the benchmark is that sentences may also have multiple labels, i.e., we need to consider and evaluate a *multi-label prediction* setting. Moreover, since OpenIE may turn each input sentence into multiple clauses, we define the following variants of the three classes of *true positives* (TPs), *false positives* (FPs) and *false negatives* (FNs) needed to compute precision, recall and F1, and with respect to whether the unit of prediction is either a *sentence* or a *clause*.

Prediction Unit: Sentence. The unit of prediction is a sentence. Under a *single-label prediction* setting, TPs, FPs and FNs can be computed in the standard way by considering also a single (i.e., the "best") predicted label per sentence. However, under a *multi-label prediction* setting, we predict as many labels as were given for the KN sentence, and then consider how many of the predicted labels also match the given labels as the TPs (and vice versa for the FPs and FNs).

Prediction Unit: Clause. The unit of prediction is a clause. Under a *single-label prediction* setting, this means that we also predict one label per clause, but since OpenIE may extract multiple clauses from the given KN sentence, we then still need to compare multiple labels obtained from the clauses with the single, given label of the KN sentence. We therefore define the following two variants for TPs and FPs (FNs again follow similarly): ANY and ALL.

ANY	TP:	*any* of the clauses' labels match the single given label of the KN sentence.
	FP:	*none* of the clauses' labels match the single given label of the KN sentence.
ALL	TP:	*all* of the clauses' labels match the single given label of the KN sentence.
	FP:	*not all* of the clauses' labels match the single given label of the KN sentence.

However, under a *multi-label prediction* setting, when using clauses as prediction unit, ANY and ALL would be too extreme to give a fair estimate of the prediction quality. We therefore introduce a third variant, UNION, as follows.

TABLE 11.3: The performance of all our approaches using KnowledgeNet.

Method	P	R	F1
Human	0.88	0.88	0.88
Diffbot Joint Model	0.81	0.81	0.81
KnowledgeNet Baseline 5 (BERT)	0.67	0.69	0.68
Clauses + BERT (ALL)	0.86	0.86	0.86
Clauses + BERT (ANY)	0.90	0.92	0.91
Clauses + BERT (UNION)	0.89	0.89	0.89
Clauses + distillBERT (ALL)	0.86	0.86	0.86
Clauses + distillBERT (ANY)	**0.92**	**0.92**	**0.92**
Clauses + distillBERT (UNION)	0.91	0.91	0.91
Clauses + feature-based-BERT (ALL)	0.86	0.74	0.79
Clauses + feature-based-BERT (ANY)	0.91	0.91	0.91
Clauses + feature-based-BERT (UNION)	0.91	0.87	0.89
Clauses + SETFIT(ANY)	0.85	0.83	0.84
Mixed + BERT (ALL)	0.87	0.75	0.80
Mixed + BERT (ANY)	0.93	0.84	0.89
Mixed + BERT (UNION)	**0.91**	**0.93**	**0.92**
Mixed + distillBERT (ALL)	0.85	0.70	0.77
Mixed + distillBERT (ANY)	0.91	0.80	0.85
Mixed + distillBERT (UNION)	0.90	0.92	0.91
Mixed + feature-based-BERT (ALL)	0.85	0.69	0.76
Mixed + feature-based-BERT (ANY)	0.85	0.83	0.84
Mixed + feature-based-BERT (UNION)	0.88	0.83	0.85
Mixed + SETFIT (ANY)	0.82	0.77	0.79
Sentences + BERT	0.86	0.78	0.82
Sentences + distillBERT	**0.87**	**0.79**	**0.83**
Clauses + Word2Vec (ALL)	0.71	0.62	0.66
Clauses + Word2Vec (ANY)	0.77	0.58	0.67
Clauses + Word2Vec (UNION)	**0.83**	**0.66**	**0.66**

UNION	TPs:	the *union* of the clauses' labels that match the given set of labels of the KN sentence.
	FPs:	the *union* of the clauses' labels that do not match the given set of labels of the KN sentence.

That is, under a multi-label prediction setting (both when using sentences and clauses as prediction units), multiple TPs, FPs and FNs may be produced per KN sentence. FewRel, on the other hand, is a single-labeled dataset and provides property annotations at the fact level. There, ALL and ANY collapse into the same (standard) case, while UNION is not present at all. Based on the afore-defined variants of TPs, FPs and FNs, we then compute *precision* (P), *recall* (R) and F1 in the standard way for both KN and FewRel.

11.5.3 Results

We now present the results of the seven baseline approaches outlined in section 11.5.1. We performed detailed experiments to demonstrate the effectiveness of our method and show how OpenIE improves RE. We tested multiple pre-trained LMs

TABLE 11.4: The performance of all our approaches using FewRel[a].

Method	P	R	F1
ERNIE	0.88	0.88	0.88
DeepEx	—	—	0.48
Clauses + BERT	**0.71**	**0.71**	**0.71**
Clauses + distillBERT	0.68	0.68	0.68
Clauses + SETFIT	0.68	0.68	0.68
Clauses + roBERTa	0.68	0.68	0.68
Clauses + distillroBERTa	0.66	0.67	0.66
Clauses + feature-based-BERT	0.75	0.59	0.66
Clauses + alBERT	0.65	0.66	0.65
Mixed + BERT	**0.66**	**0.67**	**0.66**
Mixed + distillBERT	0.65	0.66	0.65
Mixed + roBERTa	0.65	0.67	0.65
Mixed + distillroBERTa	0.65	0.67	0.65
Mixed + SETFIT	0.66	0.64	0.65
Mixed + alBERT	0.62	0.63	0.62
Mixed + feature-based-BERT	0.64	0.59	0.61
Sentences + BERT	**0.65**	**0.66**	**0.65**
Sentences + roBERTa	0.65	0.66	0.65
Sentences + distillroBERTa	0.65	0.66	0.65
Sentences + alBERT	0.63	0.65	0.64
Sentences + distillBERT	0.64	0.65	0.64
Sentences + SETFIT	0.64	0.63	0.63
Clauses + Word2Vec	0.61	0.52	0.56

[a]Here, for SETFIT, a subset of 15 relations is used similar to Tunstall et al., 2022

and report the best results in Tables 11.3 and 11.4. For all the baselines except Baseline 3 and 7, we consider candidate clauses when their subject and object text spans overlap with the text spans of the subject and object of the ground truth relations. For KN[7], the results are averaged after performing a 4-fold cross-validation on the 4 folders into which it is divided by default. For FewRel[8], we averaged over 10 runs with random splits (each by dividing the dataset in 75% for training and 25% for testing purposes) to shuffle as much as possible the data and have significant changes in the distribution of the text during training and testing time.

KN Results. We motivated our choice for the LMs in section 11.1, however, the experimental results do not suggest a clear suitability of a specific model for all RE settings. We notice that BERT and distillBERT performed best on KN, while RoBERTa and SETFIT were also useful in some settings applied to FewRel. For KN, our best baseline (Baseline 1, Clauses + distillBERT) significantly outperforms the previous work (Diffbot Joint Model and KN Baseline 5, reported on top of Table 11.3). The most important improvements (also in comparison to our other baselines) are due to (1) using clauses as a unit of prediction, (2) incorporating clauses during fine-tuning, and (3) allowing any of the OpenIE clauses to match the single KN label (as described in the ANY evaluation mode).

[7]`https://github.com/diffbot/knowledge-net`
[8]`http://zhuhao.me/fewrel`

FewRel Results. For FewRel, we compare our results against Matching the Blanks (MTB) (Soares et al., 2019), ERNIE (Zhang et al., 2019b) and DeepEx (Wang et al., 2021a). Being the board leader on FewRel (with an accuracy of 93.86), Matching the Blanks classifies the relations relying solely on the text input. It however employs additional entity markers, which we deliberately omit in favor of taking advantage of the OpenIE-based sentence decomposition and the LMs ability to interpret the arguments. While our strategy proves effective for KN, explicit entity markers may still be lacking for FewRel which represents a much more fine-grained set of 100 relations (compared to the 15 relations of KN). ERNIE is different from MTB and our system in that it uses knowledge graphs to enrich pre-trained LM. It shows good performance on FewRel (with an F1-score of 88.32), but the robustness of the system may be questioned due to inherent incompleteness of the knowledge graphs which typically limits the system's ability to generalize. We, on the other hand, want to demonstrate how a fast and simple approach can be successful even on such a competitive dataset while not suffering from unseen relational components. DeepEx offers an interesting comparative scenario because it formulates the RE task as an extension to OpenIE. While DeepEx outscores many state of the art OpenIE systems, we outperform it on the task of RE by large margin, including the few-shot setting. We attribute this result to the way OpenIE clauses are translated into relations: DeepEx essentially maps relational phrases from clauses to a knowledge graph property label or its aliases but does not take the signal from the entire clause into account. We notice that this approach roughly corresponds to our context-free "Clauses + Word2Vec" baseline which generally achieves weaker results on both datasets (especially compromising the recall), as there seem to be a plethora of relation types, specially in FewRel, that cannot be captured well just by considering the clauses' predicates.

Few-Shot Results. Figure 11.2 shows the best performing models in few-shot setting. We notice that for KN, 8 samples are sufficient for feature-based BERT to achieve about 85% F1-score. The other two models require many more samples yet do not reach the same result. On the contrary, all the tree models demonstrate similar behaviour on FewRel data. BERT has a slight advantage, however, it needs at least 30 samples to achieve above 50% F1-score. We hypothesize that the overall number and diversity of relations in FewRel contribute to this phenomenon.

Summary. We clearly notice that the best performances are obtained on the baselines where we use OpenIE extractions for inference and fine-tuning the LMs, which motivates our choice and design for the experiments. Our experiments confirm that decomposing a sentence into clauses, which are designed to express a compact, more "focused" unit of information, allows us to distill various aspects of a sentence's meaning and to represent it as (a set of) labeled relation instances in a concise manner.

11.6 Final Remarks

We proposed and evaluated a variety of simple and direct strategies to combine OpenIE with Language Models for the task of Relation Extraction. We explored how OpenIE may serve as an intermediate way of extracting concise factual information from natural-language input sentences, and we combined the obtained clauses with both context-free and contextual LMs, as both are widely used in research and known for their strengths and weaknesses. For our experiments, we utilized the KnowledgeNet dataset with 15 properties as well as the well-known FewRel dataset

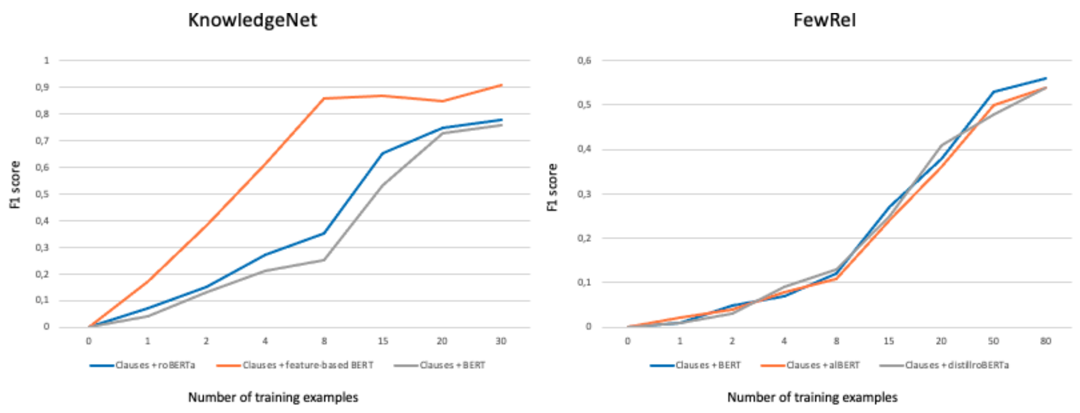


FIGURE 11.2: Change in the F1 score in a FewShot setting.

containing 100 relations. Both datasets use Wikidata as underlying KB, which constitutes a valuable resource as demonstrated by the increased number of scientific publications and applications both in academia and industry in recent years. We presented detailed experiments on Word2Vec, BERT, RoBERTa, AlBERT, SETFIT and their further distilled versions with a range of baselines that achieve up to 92% and 71% of F1 score for KnowledgeNet and FewRel, respectively. In the future, we intend to apply our work also toward various downstreams tasks such as sentiment analysis, question answering, knowledge base population, and further knowledge graph aspects.

Part V

Conclusions

Chapter 12

Conclusions

This thesis delved into the multifaceted world of DL, examining its intricacies and complexities from various angles and perspectives. Our exploration covered second-order methods, random embeddings, and RE, aiming to improve the performance of DL techniques across different use cases.

In Part II, we focused on the initialization and optimization of neural networks by introducing a new approximated chain rule for Hessian backpropagation. This method aimed to facilitate fast and systematic training of NNs, addressing the limitations of traditional empirical heuristics. Our approximated chain rule for Hessian backpropagation provided a theoretical foundation for the optimization and training of NNs, demonstrating the superior efficiency of second-order methods across multiple datasets.

In Part III, we shifted our focus to the analysis of random embeddings, which serve as a crucial tool for dimensionality reduction in ML and DL algorithms. Our research yielded improved bounds for sparse random embeddings, outperforming the previous state of the art and providing robust, provable guarantees for real-world datasets. Furthermore, our analysis extended to Rademacher random embeddings, offering non-oblivious insights into input data.

In Part IV, we ventured into IR, specifically targeting RE. By examining advanced techniques and tools for extracting and analyzing textual data, we demonstrated their applicability in real-world scenarios. Our approach combined distant supervision, few-shot learning, OpenIE, and various LMs to enhance the task of RE, showcasing the capabilities of these methods in a fast and efficient manner.

In conclusion, this thesis presented a diverse array of approaches and strategies, each contributing to the enhancement of DL performance across a range of scenarios and applications. By investigating the nuances of DL from multiple angles, we have provided valuable insights and innovative solutions that can be leveraged by researchers and practitioners in the field.

For future directions, the following list presents some possibilities that can build upon and extend our work:

- **Second-order methods:** As hardware continues to advance at a remarkable pace, it will be intriguing to investigate the feasibility of calculating the full Hessian that considers the second partial derivative of the weights matrix (quadratic with respect to the number of weights per layer). Although currently infeasible, such advancements could lead to significant improvements in the training and optimization of neural networks, leveraging the power of second-order methods more effectively.

- **Sparse Random Embeddings:** In our research, we introduced the dispersion parameter, v, which is data-dependent. While our improved bounds were experimentally tested on a range of datasets with varying values for v, future work could involve testing these bounds on an even broader array of datasets from diverse use cases. By doing so, we can further validate the theoretical guarantees of our sparse random embeddings and better understand their applicability across different domains.

- **Relation Extraction:** A notable challenge in RE is the presence of noisy extractions from OpenIE. To address this issue, future work could involve implementing an intermediate post-processing step for OpenIE output before inputting it into language models for label inference. This additional step would not only help mitigate noise in the data but also allow for the retention of more OpenIE extractions, ultimately yielding a larger sample for the relational labeling process. By refining the RE pipeline, we can expect to see improvements in the accuracy and reliability of extracted information from textual data.

Appendix A

Approximated Chain Rule for Hessian Backpropagation

A.1 Proof of Hessian Chain Rule

Our goal is to compute the Hessian with respect to the parameters in the layer k. By the chain rule

$$D_{\mathsf{W}^{(k)}} L(z,t) = D_{z^{(n)}} L(z,t) \bullet D_{\mathsf{W}^{(k)}} \boldsymbol{z}^{(n)} \tag{A.1}$$

Note that the second tensor is of shape $[d_n, d_{k+1}, d_k]$ (rank 3!), the contraction is over the dimension of $z^{(n)}$. Again by the chain and product rules

$$D^2_{\mathsf{W}^{(k)}} L = \underbrace{D^2_{\boldsymbol{z}^{(n)}} L \bullet D_{\mathsf{W}^{(k')}} \boldsymbol{z}^{(n)} \bullet D_{\mathsf{W}^{(k)}} \boldsymbol{z}^{(n)}}_{\mathsf{H}_1} + \underbrace{D_{\boldsymbol{z}^{(n)}} L \bullet D^2_{\mathsf{W}^{(k)}} \boldsymbol{z}^{(n)}}_{\mathsf{H}_2} \tag{A.2}$$

In the component H_1 the dot-products contract indices $\boldsymbol{z}^{(n)}$ (note that D^2 is symmetric and the terms D are same, hence the order of pairing dimensions of $\mathsf{W}^{(k)}$ does not matter). As for the second component H_2, it is a product of tensors of rank 1 and 5. In order to further simplify, we are going to show that H_2 is negligible compared to H_1. the intuition is as follows: in H_1 the contribution comes from gradients $D_{\mathsf{W}^{(k)}}$ while in H_2 from second-order derivatives $D^2_{\mathsf{W}^{(k)}}$; we consider activations such that $f(u) = au + O(u^3)$ and therefore for small u second-derivatives are near zero but first derivatives are not, and their contributions dominate.

In the analysis below we assume that weights are sufficiently small, and biasses are zero (or of much smaller variance compared to weights). Let $\boldsymbol{u}^{(k)} = \mathsf{W}^{(k)} \cdot \boldsymbol{z}^{(k)} + \boldsymbol{b}^{(k)}$ be the output before activation at the k-th layer.

Due to A.2, our goal is to evaluate first and second derivatives of $\boldsymbol{z}^{(n)}$ with respect to weights $\mathsf{W}^{(k)}$, under the assumption that inputs $\boldsymbol{z}^{(i)}$ are sufficiently small. Consider how $\boldsymbol{z}^{(k+1)}$ depends on $\mathsf{W}^{(k)}$. By the chain rules

$$D_{\mathsf{W}^{(k)}} \boldsymbol{z}^{(k+1)} = D_{\boldsymbol{u}^{(k)}} f(\boldsymbol{u}^{(k)}) \bullet D_{\mathsf{W}^{(k)}} \boldsymbol{u}^{(k)} \tag{A.3}$$

$$D^2_{\mathsf{W}^{(k)}} \boldsymbol{z}^{(k+1)} = D^2_{\boldsymbol{u}^{(k)}} f(\boldsymbol{u}^{(k)}) \bullet D_{\mathsf{W}^{(k)}} \boldsymbol{u}^{(k)} \bullet D_{\mathsf{W}^{(k)}} \boldsymbol{u}^{(k)} \tag{A.4}$$

Note that $D_{\boldsymbol{u}^{(k)}} f(\boldsymbol{u}^{(k)})$ and $D^2_{\boldsymbol{u}^{(k)}} f(\boldsymbol{u}^{(k)})$ are diagonal tensors because f is applied element-wise. More precisely

$$\left[D^2_{\boldsymbol{u}^{(k)}} f(\boldsymbol{u}^{(k)}) \right]_{i,j,j'} = \delta_{i,j} \delta_{i,j'} \cdot f''(\boldsymbol{u}^{(k)}_i) \tag{A.5}$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta which is one where indices match and zero otherwise. Moreover,

$$\left[D_{\mathsf{W}^{(k)}}\boldsymbol{u}^{(k)}\right]_{j,p,q} = \frac{\partial}{\partial \mathsf{W}^{(k)}_{p,q}}(\mathsf{W}^{(k)}\cdot \boldsymbol{z}^{(k)} + \boldsymbol{b}^{(k)})_j = \delta_{j,p}\cdot z^{(k)}_q \tag{A.6}$$

Thus

$$\left[D^2_{\mathsf{W}^{(k)}}\boldsymbol{z}^{(k+1)}\right]_{i,p,q,p',q'} = \delta_{i,p}\delta_{i,p'}\cdot f''(\boldsymbol{u}^{(k)}_i)\cdot \boldsymbol{z}^{(k)}_q\cdot \boldsymbol{z}^{(k)}_{q'} \tag{A.7}$$

When this tensor acts, as a bi-linear form, on a tensor $g = g_{p,q}$ we therefore obtain

$$\left[D^2_{\mathsf{W}^{(k)}}\boldsymbol{z}^{(k+1)}\bullet g\bullet g\right]_i = f''(\boldsymbol{u}^{(k)}_i)\sum_{q,q'}\boldsymbol{z}^{(k)}_q\boldsymbol{z}^{(k)}_{q'}g_{i,q}g_{i,q'} \tag{A.8}$$

$$= f''(\boldsymbol{u}^{(k)}_i)\left(\sum_q \boldsymbol{z}^{(k)}_q g_{i,q}\right)^2 \tag{A.9}$$

Since our assumption on activations implies $f''(\boldsymbol{u}) = O(f'''\cdot\boldsymbol{u})$ for real $\boldsymbol{u}$, we see this is of order $O(f'''\|\boldsymbol{u}^{(k)}\|\|\boldsymbol{z}^{(k)}\|^2)\cdot\|g\|^2$.

Claim 1 (Magnitude of second derivative of weights)**.**

$$D^2_{\mathsf{W}^{(k)}}\mathbf{z}^{(k+1)}\bullet g\bullet g = O(f'''\|\mathbf{u}^{(k)}\|\|\mathbf{z}^{(k)}\|^2)\cdot\|g\|^2. \tag{A.10}$$

which is of order $O(f'''\cdot c^3)$ where c is the constant from our 'relatively small inputs' assumption.

Next, observe that the roles of $\boldsymbol{z}^{(k)}$ and $\mathsf{W}^{(k)}$ in $\boldsymbol{u}^{(k)}$ are symmetric. Thus we have a similar result with respect to $\boldsymbol{z}^{(k)}$.

Claim 2 (Magnitude of second derivative of inputs)**.**

$$D^2_{\mathbf{z}^{(k)}}\mathbf{z}^{(k+1)}\bullet g\bullet g = O(f'''\|\mathbf{u}^{(k)}\|\|\mathsf{W}^{(k)}\|^2)\cdot\|g\|^2 \tag{A.11}$$

which is of order $O(f'''\cdot c)$ where c is the constant from our 'relatively small inputs' assumption.

We need to prove that this propagates to higher-level outputs z^i, where $i > k$. This is intuitive, considering now that $\boldsymbol{z}^{(i)}$ is a function of $\boldsymbol{z}^{(k+1)}$ with no dependencies on $\mathsf{W}^{(k)}$. To prove it formally look at the second-order chain rule

$$D^2_{\mathsf{W}^{(k)}}\boldsymbol{z}^{(i)} = D^2_{\boldsymbol{z}^{(k+1)}}\boldsymbol{z}^{(i)}\bullet D_{\mathsf{W}^{(k)}}\boldsymbol{z}^{(k+1)}\bullet D_{\mathsf{W}^{(k)}}\boldsymbol{z}^{(k+1)} + D_{\boldsymbol{z}^{(k+1)}}\boldsymbol{z}^{(i)}\bullet D^2_{\mathsf{W}^{(k)}}\boldsymbol{z}^{(k+1)} \tag{A.12}$$

Now the second term is clearly $O(f'''c^3)$ by the first claim. As for the first term, the first tensor is of order $O(f'''c)$ while the two others are $O(f'c)$. The dot-product gives the bound $O(f'''c^3)$.

Claim 3. *For every $i > k$ it holds $D^2_{\mathsf{W}^{(k)}}\mathbf{z}^{(i)} = O(f''c^3)$.*

Summing up, we can ignore second-derivatives with respect to weights, and this is accurate except third-order terms in the magnitude of $\boldsymbol{z}^{(i)}$. In particular, we can ignore the effect of H_2.

A.2 Factorizing the Hessian Quadratic Form

Consider any potential update vector g for weights $\mathsf{W}^{(k)}$, it has to be of same shape as $D_{\mathsf{W}^{(k)}}L$ or equivalently $\mathsf{W}^{(k)}$, that is $[d_{k+1}, d_k]$. Our goal is to evaluate the hessian quadratic form on g. Ignoring the smaller part H_2 we are left with H_1 which gives

$$D^2_{\mathsf{W}^{(k)}} L \bullet g \bullet g = D^2_{z^{(n)}} L \bullet D_{\mathsf{W}^{(k)}} z^{(n)} \bullet D_{\mathsf{W}^{(k)}} z^{(n)} \bullet g \bullet g \tag{A.13}$$

where g is contracted on all indices together with $\mathsf{W}^{(k)}$. To emphasize this we can regroup, obtaining

Claim 4 (Hessian quadratic form)**.** *The hessian quadratic form for an update g equals*

$$D^2_{\mathsf{W}^{(k)}} L \bullet g \bullet g = D^2_{\mathbf{z}^{(n)}} L \bullet \left(D_{\mathsf{W}^{(k)}} \mathbf{z}^{(n)} \bullet g \right) \bullet \left(D_{\mathsf{W}^{(k)}} \mathbf{z}^{(n)} \bullet g \right) \tag{A.14}$$

We work further to simplify rank-3 tensors. By the chain rule

$$D_{\mathsf{W}^{(k)}} z^{(n)} \bullet g = D_{z^{(k+1)}} z^{(n)} \bullet D_{\mathsf{W}^{(k)}} z^{(k+1)} \bullet g \tag{A.15}$$

Let $u^{(k)} = \mathsf{W}^{(k)} \cdot z^{(k)} + b^{(k)}$ be the output before activation. By the chain rule

$$D_{\mathsf{W}^{(k)}} z^{(k+1)} = D_{u^{(k)}} z^{(k+1)} \bullet D_{\mathsf{W}^{(k)}} u^{(k)} \tag{A.16}$$

Note that $D_{\mathsf{W}^{(k)}} u^{(k)}$ is a third-order tensor of shape $[d_{k+1}, d_{k+1}, d_k]$. Denote its elements by $M_{i',i,j}$. We have

$$\left[D_{\mathsf{W}^{(k)}} u^{(k)} \right]_{i',i,j} = [i' = i] \cdot z^{(k)}_j \tag{A.17}$$

and we compute the dot product

$$\left[D_{\mathsf{W}^{(k)}} u^{(k)} \bullet g \right]_{i'} = \sum_{i,j} \left[D_{\mathsf{W}^{(k)}} u^{(k)} \right]_{i',i,j} g_{i,j} = \sum_j z^{(k)}_j g_{i',j} \tag{A.18}$$

which can be expressed compactly in terms of matrix multiplication as

$$D_{\mathsf{W}^{(k)}} u^{(k)} \bullet g = g \cdot z^{(k)} \tag{A.19}$$

which is a vector of shape $[d_{k+1}]$.

Using this in A.16 we obtain, in terms of matrix products

$$D_{\mathsf{W}^{(k)}} z^{(k+1)} \bullet g = D_{u^{(k)}} z^{(k+1)} \cdot \left(D_{\mathsf{W}^{(k)}} z^{(k+1)} \bullet g \right) = D_{u^{(k)}} z^{(k+1)} \cdot g \cdot z^{(k)} \tag{A.20}$$

Now, in terms of matrix products, A.15 becomes

$$D_{\mathsf{W}^{(k)}} z^{(n)} \bullet g = D_{z^{(k+1)}} z^{(n)} \cdot D_{\mathsf{W}^{(k)}} z^{(k+1)} \cdot g \cdot z^{(k)} \tag{A.21}$$

which is a vector of shape $[d_n]$. Finally note that $\mathsf{H} \bullet v \bullet v = v^T \cdot \mathsf{H} \cdot v$ where H is a symmetric matrix and v is vector. This proves

Claim 5 (Approximated hessian form)**.** *For sufficiently small inputs, the hessian quadratic form can be approximated as*

$$\mathsf{H}_{\mathsf{W}^{(k)}}[g, g] \approx \mathbf{v}^T \cdot \mathsf{H}_{\mathbf{z}} \cdot \mathbf{v} \tag{A.22}$$

where

$$\mathbf{v} = D_{\mathbf{z}^{(k+1)}}\mathbf{z}^{(n)} \cdot D_{\mathsf{W}^{(k)}}\mathbf{z}^{(k+1)} \cdot g \cdot \mathbf{z}^{(k)}. \tag{A.23}$$

This claim implies the first part of Remark 1. The error estimate follows by the discussion in the previous section.

A.3 Further Factorization

We have seen that the quadratic effects of $\mathsf{W}^{(k)}$ can be ignored, thus it is enough to consider the simplified recursion

$$z^{(k+1)} \approx a \cdot \left(\mathsf{W}^{(k)} \cdot z^{(k)} + b^{(k)}\right) \tag{A.24}$$

for a diagonal matrix a (which captures first-derivatives of activations), or equivalently:

Claim 6 (Second-order recursion for small inputs)**.** *For relatively small inputs the hessian can be computed under the simplified recursion*

$$\mathbf{z}^{(k+1)} \approx \mathbf{J}^{(k)} \cdot \mathbf{z}^{(k)} \tag{A.25}$$

where $\mathbf{J}^{k} = D_{\mathbf{z}^{(k)}}\mathbf{z}^{(k+1)}$*. In particular, the term* H_2 *can be ignored.*

We now proceed to further factorize v. By linearization we obtain

$$z^{(k)} \approx D_{z^{(k-1)}}z^{(k)} \cdot z^{(k-1)} = \ldots = D_{z^{(k-1)}}z^{(k)} \tag{A.26}$$

Moreover, by the chain rule, output gradient facatorizes as

$$D_{z^{(k+1)}}z^{(n)} = D_{z^{(n-1)}}z^{(n)} \cdot z^{(k-1)} \cdot D_{z^{(n-2)}}z^{(n-1)} \cdot \ldots \cdot D_{z^{(k+1)}}z^{(k+2)} \tag{A.27}$$

regardless of linearizing assumptions. Combining these two observations gives

Claim 7 (Factorizing under linearization)**.** *Up to terms linear in* $\mathbf{z}^{(i)}$ *for* $i = k-1, \ldots, 1$ *we have*

$$\mathbf{v} \approx D_{\mathbf{z}^{(n-1)}}\mathbf{z}^{(n)} \cdot D_{\mathbf{z}^{(n-2)}}\mathbf{z}^{(n-1)} \cdot \ldots \cdot D_{\mathbf{z}^{(k+1)}}\mathbf{z}^{(k+2)} \cdot A \cdot g \\ \cdot D_{\mathbf{z}_{(k-1)}}\mathbf{z}^{(k)} \cdot \ldots D_{\mathbf{z}^{(0)}}\mathbf{z}^{(1)} \cdot \mathbf{z}^{(0)} \tag{A.28}$$

This proves Equation 4.4.

Appendix B

Sparse Random Embeddings

B.1 Proof of Theorem 1

Recall that we have $E(x) = s^{-1}\sum_{r=1}^{m} E_r(x)$. Let $E'_r(x)$ be independent copies of $E_r(x)$ for $r = 1 \ldots m$ and define $E''_r(x) = E'_r(x)\sigma_r$ where σ_r are iid Rademacher random variables.

The random variables $(E_r(x))_r$ are negatively dependent , and thus moments of their sum are not bigger than if they were independent, that is

$$\|\sum_r E_r(x)\|_d \leqslant \|\sum_r E'_r(x)$$

as observed in Freksen, Kamma, and Larsen, 2019 (see also a more general argument in Shao, 2000). By the symmetrization

$$\|\sum_r E'_r(x)\|_d \leqslant 2\|\sum_r E''_r(x)\|_d$$

This discussion shows that

$$\|E(x)\|_d \leqslant 2s^{-1}\|\sum_r E''_r(x)\|_d.$$

We now apply our bounds from Lemma 4 to the symmetrized and independent random variables $(E''_r(x))_r$; their even moments are as those of $E_r(x)$ and bounded as in Lemma 5. The result follows (note that we lost the constant 2 which appears in the theorem, due to symmetrization).

B.2 Analysis of Freksen, Kamma, and Larsen, 2019

B.2.1 Proof of Remark 12

Inspecting the proof we find that best constants are

1. $C_1 = 2eC$ where C is a constant which satisfies $\binom{2d}{2d_1 \ldots 2d_n} \leqslant C^d \frac{(2d)^{2d}}{\prod_i (2d_i)^{2d_i}}$ for all $(d_i)_i$ with sum d such that $d_i \leqslant d/2$. Specializing to $d_1 = d/2, d_2 = d/2$ and $d_i = 0$ for $i > 2$ we see that C must satisfy $\binom{2d}{d} \leqslant C^d \frac{(2d)^{2d}}{d^{2d}} = C^d 2^{2d}$. But $\binom{2d}{d} = \Theta(2^{2d}/\sqrt{d})$ and taking large d we get $C \geqslant 1$.

2. The proof starts with the bound $\|E_r\|_d^{1/2} \leqslant K^{1/2} + C_1 \cdot K^{-1/2} \sup_{1 \leqslant t \leqslant d} d/t \cdot (Kp/d)^{1/2t}$ for any integer K and $C_1 \geqslant 2e$ as in the discussion above. The goal is to choose K so that the bound becomes $\sqrt{C_2 d/\log(1/p)}$.

Choosing $t = \frac{1}{2}\log(d/Kp)$ we see that the bound gives at least $R = K^{1/2} + \frac{2d}{K^{1/2}\log(d/Kp)}$, when $2 \leqslant \log(d/Kp) \leqslant 2d$; this condition is indeed satisfied: the right-side follows because $p/K < 1$, and the left-side follows because otherwise $\log(d/Kp) < 2$ so the supremum is at $t = 1$ giving the bound $\Omega(\sqrt{d/p})$, worse than our goal for small p. For the same reason we can assume $K \leqslant d$. Now by AM-GM inequality we obtain $R \geqslant 2\sqrt{2d/\log(d/Kp)} \geqslant 2\sqrt{2d/\log(1/p)}$.

Thus, the proof implies only $C_2 \geqslant 8$.

B.3 Proof of Lemma 2

Let $S = \sum_{i=1}^{n} Y_i$. Then

$$\mathbb{E}|\sum_{i\neq j} Y_i Y_j|^d = \mathbb{E}(\sum_i Y_i(S-Y_i))^d$$

$$\sum_{\|(d_i)_i\|_1=d} \binom{d}{d_1,\ldots,d_n} \mathbb{E}\left[\prod_i Y_i^{d_i}(S-Y_i)^{d_i}\right]$$

We first prove that

$$\mathbb{E}\left[\prod_i Y_i^{d_i}(S-Y_i)^{d_i}\right] \leqslant \mathbb{E}\left[S^d \prod_i Y_i^{d_i}\right]. \tag{B.1}$$

We have $(S-Y_i)^{d_i} = \sum_{d_{i,j}} \binom{d_i}{d_{i,1},\ldots,d_{i,n-1}} \prod_{j\neq i}^{n-1} Y_j^{d_{i,j}}$, due to $S - Y_i = \sum_{j\neq i} Y_j$. This can be written as

$$(S-Y_i)^{d_i} = \sum_{d_{i,j}:d_{i,i}=0} \binom{d_i}{d_{i,1},\ldots,d_{i,n}} \prod_j^n Y_j^{d_{i,j}}$$

We now have

$$\begin{aligned} L &= \mathbb{E}\left[\prod_i Y_i^{d_i}(S-Y_i)^{d_i}\right] \\ &= \mathbb{E}\left[\prod_i Y_i^{d_i} \sum_{d_{i,j}:d_{i,i}=0} \binom{d_i}{d_{i,1},\ldots d_{i,n}} \prod_j^n Y_j^{d_{i,j}}\right] \\ &= \sum_{d_{i,j}:d_{i,i}=0} \mathbb{E}\left[\prod_i \binom{d_i}{d_{i,1},\ldots d_{i,n}} Y_i^{d_i} \prod_j^n Y_j^{d_{i,j}}\right] \end{aligned}$$

All expectations terms are non-negative due to the symmetry of Y_i. Thus the whole sum does not decrease if we ignore the condition $d_{i,i} = 0$ (as this will produce only more terms). Therefore

$$L \leqslant \mathbb{E}\left[\sum_{d_{i,j}}\prod_i \binom{d_i}{d_{i,1},\ldots d_{i,n}} Y_i^{d_i}\prod_j^n Y_j^{d_{i,j}}\right]$$
$$= \mathbb{E}\left[\prod_i Y_i^{d_i}\left(\sum_j Y_j\right)^{d_i}\right]$$
$$= \mathbb{E}\left[\prod_i Y_i^{d_i} S^d\right]$$

which completes the proof of B.1.

We now claim that $Y_1,\ldots,Y_n$ conditionally on $S = s$ are negatively dependent. To this end it suffices to establish that $\mathbb{E}[\phi(Y_1,\ldots,Y_k)|Y_1+\ldots+Y_k = s]$ is non-decreasing in s for coordinate-wise non-decreasing functions ϕ and $k = 1,\ldots,n-1$. Conditioning on $Y_3\ldots,Y_k$ we see that it suffices to prove that for $k = 1,2$. The case of $k = 1$ is obvious. For $k = 2$ we use the observation which states that it suffices to check that for functions ϕ of one argument; since $\mathbb{E}[\phi(Y_1)|Y_1+Y_2 = s] = \mathbb{E}\phi(s-Y_2)$ is indeed non-decreasing in s when ϕ is non-decreasing, the result follows.

By the negative dependence we thus obtain

$$\mathbb{E}\left[\prod_i Y_i^{d_i}(S-Y_i)^{d_i}\right] \leqslant \mathbb{E}\left[S^d\right]\mathbb{E}\left[\prod_i Y_i^{d_i}\right] \tag{B.2}$$

which, coupled with the multinomial expansion, finishes the proof.

B.4 Proof of Lemma 3

Proof. It suffices to prove that

$$u \to \mathbb{E}f\left(\sum_i Y_i u_i^{1/2}\right)$$

is Schur-concave in u, where $f(t) = t^d$. Indeed, we have $\mathbb{E}(\sum_i Y_i x_i)^d = \mathbb{E}(\sum_i Y_i|x_i|)^d$ (follows by raising to the power of d and applying the multinomial expansion, then only even powers contribute to the expectation), and the claim follows by denoting $|x_i| = u_i$.

Since g is symmetric it suffices to check the Schur-Ostrowski criterion for u_1 and u_2. Let $W = \sum_{i>2}\eta_i\sigma_i u_i^{1/2}$, then

$$\frac{\partial g}{\partial u_1} - \frac{\partial g}{\partial u_2} = \frac{u_2^{1/2}X_1 - u_1^{1/2}X_2}{2(u_1u_2)^{1/2}}\cdot f'\left(\sum_{i=1}^2 Y_i u_i^{1/2} + W\right)$$

Thus it remains to prove that the expectation of

$$Q \triangleq (u_2^{1/2}X_1 - u_1^{1/2}X_2)\cdot f'\left(\sum_{i=1}^2 Y_i u_i^{1/2} + W\right) \tag{B.3}$$

is negative when $u_1 < u_2$. Recall that Y_i are symmetric and take three values $\{-1,0,1\}$. We condition on two cases: a) $X_1, X_2 \neq 0$ and b) one of X_1, X_2 is zero.

In case a) the result reduces to the case of Rademacher variables, solved already by Eaton. We are left with case b). If $X_1 = X_2$ the expression is zero. We further assume $X_1 \neq X_2$. Consider the two disjoint events: $\mathcal{E}_1$ is that $X_2 = 0$ and $X_1 = \pm 1$ and $\mathcal{E}_2$ is that $X_1 = 0$ and $X_2 = \pm 1$. Then we have that

$$\mathbb{E}\left[Q|\mathcal{E}_1, W\right] = u_2^{1/2}\left(f'\left(u_1^{1/2}+W\right) - f'\left(-u_1^{1/2}+W\right)\right)$$
$$\mathbb{E}\left[Q|\mathcal{E}_2, W\right] = -u_1^{1/2}\left(f'\left(u_2^{1/2}+W\right) - f'\left(-u_2^{1/2}+W\right)\right)$$

For $t > 0$ we consider the the auxiliary function

$$g_w(t) = t^{-1}\left(f'\left(t+w\right) - f'\left(-t+w\right)\right)$$

We have $\mathbb{E}\left[Q|\mathcal{E}_1, W\right] = (u_1u_2)^{1/2} g_W(u_1^{1/2})$ and $\mathbb{E}\left[Q|\mathcal{E}_2, W\right] = -(u_1u_2)^{1/2} g_W(u_2^{1/2})$, and therefore $\mathbb{E}[Q|\mathcal{E}_1 \cup \mathcal{E}_2] = (u_1u_2)\mathbb{E}_W[g_W(u_1^{1/2}) - g_W(u_2^{1/2})]$. If we prove that $g_W(t)$ increases in t, the proof is complete.

Since in our case $f(t) = t^d$, we find that $g_w(t) = d \cdot \frac{(w+t)^{d-1}-(w-u)^{d-1}}{t}$. Since d is even $g_w(t) = dt^{-1}((w+t)^{d-1} + (t-w)^{d-1}) = d \cdot \sum_{0 \leqslant k < \frac{d-1}{2}} \binom{d}{2k} t^{d-2-2k} w^{2k}$, indeed is increasing in t regardless of w. □

B.5 Intuitions about Corollary 4

The result follows because x^* is majorized by every other vector which satisfies the constraints.

However one may wonder why is not the flat vector x^{flat}, with all non-zero entries equal to v, the worst case? Observe that already for the case of $d = 2$ this gives the norm of $\sqrt{v^2 p}$ while our construction gives the biggeer value $\sqrt{pv^2 + p\frac{1-v^2}{n-1}}$.

B.6 Proof of Lemma 5

Proof. Due to Lemma 2 applied to $Z_i \sim A_{r,i} \cdot x_i$ and the definition of A, it suffices to show that $\| \sum_i x_i Y_i \|_d \leqslant T_{n,p,d}(v)$. Consider x^* as in Corollary 4. Define $W = \frac{1-v^2}{\sqrt{n-1}} \sum_{i=2}^{n} Y_i$. Using the independence and symmetry of Y_1 and W, we obtain

$$\| \sum_i x_i^* Y_i \|_d \leqslant \mathbb{E}\left(Y_1 v + W\right)^d$$
$$= \sum_{k=0}^{d/2} \binom{d}{2k} v^{2k} \mathbb{E} Y_1^{2k} \mathbb{E} W^{d-2k}$$

Let B_1, B_2 be Bernoulli with parameter σ such that $\sigma^2 + (1-\sigma^2) = 1-p$, then $Y_i \sim B_1 - B_2$, and thus $W \sim B' - B''$ where $B, B'' \sim^{iid} \mathrm{Binom}(n-1, \sigma)$. Also, we have $\mathbb{E} Y_1^{2k} = p^{\mathbb{I}(k>0)}$. We have $T_{n,p,d}(v) \leqslant \| \sum_i x_i^* Y_i \|_d$, which combined with the bound above, completes the proof. □

B.7 Proof of Lemma 4

We have

$$\mathbb{E}(\sum_i Z_i)^d = \sum_{d=(d_i)_i} \binom{d}{d_1 \ldots d_n} \prod_i \mathbb{E} Z_i^{d_i}$$

We need the following

Proposition 1. *We have* $\binom{d-y}{x} \leqslant c \cdot \binom{d}{x}$ *for* $0 \leqslant x, y,\ x+y \leqslant d$ *where* $c = \mathrm{e}^{-\frac{xy}{d}}$.

Proof. Note that c satisfies $\prod_{i=0}^{x-1} \left(1 - \frac{y+i}{d+i}\right) \leqslant c$. The left-hand side is at most $(1 - y/d)^x \leqslant \mathrm{e}^{-yx/d}$ □

We conclude that

$$\binom{d}{d_1 \ldots d_n} \leqslant \mathrm{e}^{-\frac{d}{2}} \prod_{i=1}^{k} \binom{d}{d_i}.$$

To see this we assume without losing generality that d_i is sorted in descending order. Since $\binom{d}{d_1 \ldots d_n} = \binom{d}{d_1}\binom{d-d_1}{d_2}\binom{d-d_1-d_2}{d_3}\ldots$ by 1 the above holds with constant e^{-c} where $c = d^{-1}\sum_{1 \leqslant j \leqslant i \leqslant d}^{k} d_i d_j \geqslant \frac{(\sum_i d_i)^2}{2d} = d/2$.

Using the above bound we get

$$\mathbb{E}(\sum_i Z_i)^d \leqslant \mathrm{e}^{-d/2} \sum_{d=(d_i)_i} \prod_i \binom{d}{d_i} \mathbb{E} Z_i^{d_i}$$
$$= \mathrm{e}^{-d/2} \prod_i \sum_k \binom{d}{k} \mathbb{E} Z_i^{k}$$

Substituting $Z_i := Z_i/t$ we obtain

$$\mathbb{E}(t^{-1}\sum_i Z_i)^d \leqslant \mathrm{e}^{-d/2} \prod_i \sum_k \binom{d}{k} \mathbb{E} Z_i^{k}/t^k$$

Now if $Z_1, \ldots, Z_n \sim^{iid} Z$ and t is such that $(\sum_k \binom{d}{k} \mathbb{E} Z^k / t^k)^n = \mathrm{e}^{d/2}$ we obtain $\mathbb{E}(t^{-1}\sum_i Z_i)^d \leqslant 1$ which is equivalent to $\|\sum_i Z_i\|_d \leqslant t$.

Appendix C

Rademacher Random Embeddings

C.1 Proof of Lemma 6

Consider two non-negative functions f, g and inputs $x \prec y$. Consider the identity

$$f(y)g(y) - f(x)g(x) \quad = \quad (f(y) - f(x)) \cdot g(y) + f(x) \cdot (g(y) - g(x)). \tag{C.1}$$

If f, g are Schur-convex then $f(y) - g(x) \geqslant 0$ and $g(y) - g(x) \geqslant 0$ and the whole expression is non-negative when f, g are non-negative. This shows that $f \cdot g$ is also Schur-convex. The claim for Schur-concave functions follows analogously (the expression is then non-positive).

C.2 Proof of Lemma 7

The proof follows from the fact that x is dominated by y if and only if x can be produced from y by a sequence of *Robin-Hood operations*, and the fact that Robin-Hood operations change only two fixed components of vectors.

C.3 Proof of Theorem 5

Proof. Note that $\mathbf{R}_q$ is a polynomial in x_i^2 with integer coefficients, and thus a well-defined function of (x_i^2). This follows by applying the multinomial expansion and noticing that monomials with odd exponents have expectation zero due to the symmetry of Rademacher distribution. $\mathbf{R}_q$ is obviously symmetric. By Lemma 7 it now suffices to validate the Schur-concativity for x_1^2, x_2^2 and any fixed choice of $(x_i)_{j>2}$. Define the following expressions

$$\begin{aligned} P &= \sum_{i \notin \{1,2\}} x_i r_i \\ R &= \sum_{i,j \notin \{1,2\}} x_i x_j r_i r_j, \end{aligned} \tag{C.2}$$

then our task is to prove the Schur-concativity of the function

$$\mathbf{R}_q \triangleq \mathbf{E} \left(P(x_1 r_1 + x_2 r_2) + x_1 x_2 r_1 r_2 + R \right)^q, \tag{C.3}$$

with respect to x_1^2, x_2^2.

By the multinomial expansion we find that

$$\mathbf{R}_q \triangleq \sum_{q_1+q_2+q_3=q} \binom{q}{q_1, q_2, q_3} \left[\mathbf{E} \left[P^{q_1} R^{q_3} \right] \quad \mathbf{E} \left[(x_1 r_1 + x_2 r_2)^{q_1} (x_1 x_2 r_1 r_2)^{q_2} \right] \right], \tag{C.4}$$

where we used the independence of r_1, r_2 on $(r_i)_{i>2}$ and thus also on P, R. Observe that $P^{q_1}R^{q_3}$ is, by definition and our assumption $x_i \geqslant 0$, a polynomial in symmetric random variables r_i with non-negative coefficients. This observation shows that

$$\mathbf{E}\left[P^{q_1}R^{q_3}\right] \geqslant 0, \tag{C.5}$$

and by Lemma 6 it suffices to prove that

$$F \triangleq \mathbf{E}\left[(x_1r_1 + x_2r_2)^{q_1}(x_1x_2r_1r_2)^{q_2}\right] \tag{C.6}$$

is Schur-concave as a function of x_1^2, x_2^2 for any non-negative integers q_1, q_2.

To see that F is indeed a well-defined function of x_1^2, x_2^2, note that it equals the expectation of a polynomial in the symmetric random variables $y_i = x_ir_i$; thus only monomials with even-degrees contribute, and the result is a polynomial in $y_i^2 = x_i^2$. In fact, F equals the sum of even-degree monomials in the expanded polynomial $(x_1 + x_2)^{q_1}(x_1x_2)^{q_2}$.

We next observe that

$$F = \begin{cases} (x_1x_2)^{q_2}\mathbf{E}\left[(x_1r_1 + x_2r_2)^{q_1}\right] & q_2 \text{ even} \\ (x_1x_2)^{q_2-1}\mathbf{E}\left[(x_1r_1 + x_2r_2)^{q_1}x_1x_2r_1r_2\right] & q_2 \text{ odd} \end{cases}. \tag{C.7}$$

Note that x_1x_2 is Schur-concave in non-negative x_1, x_2; indeed, the identity $(x_1 + \epsilon)(x_2 - \epsilon) = x_1x_2 + \epsilon(x_2 - x_1 - \epsilon)$ shows that Robin-Hood transfers increase the value. By Lemma 6 we conclude that $(x_1x_2)^k$ is Schur concave in x_1^2, x_2^2 for non-negative even k. Thus, by C.7 and Lemma 6, we conclude that it suffices to consider the case $q_2 = 1$, that is, to prove the Schur-concavity of the following two functions:

$$G_k \triangleq \mathbf{E}\left[(x_1r_1 + x_2r_2)^k\right] \tag{C.8}$$

$$H_k \triangleq \mathbf{E}\left[(x_1r_1 + x_2r_2)^k x_1x_2r_1r_2\right]. \tag{C.9}$$

with respect to x_1^2, x_2^2 for any non-negative integer k.

Using the identity $(x_1r_1 + x_2r_2)^k = (x_1r_1 + x_2r_2)^{k-2}(x_1^2 + x_2^2 + 2x_1x_2r_1r_2)$, we find the following recurrence relation

$$G_k = (x_1^2 + x_2^2)G_{k-2} + 2H_{k-2} \tag{C.10}$$

$$H_k = 2x_1^2x_2^2G_{k-2} + (x_1^2 + x_2^2)H_{k-1}, \tag{C.11}$$

valid for $k \geqslant 2$. Since $x_1^2 + x_2^2$ and $x_1^2x_2^2$ are Schur-concave as functions of x_1^2, x_2^2, by Lemma 6 the concavity property proven for $k - 2$ implies that it is valid also for k. By induction, it suffices to verify the case $k = 0$ and $k = 1$. But we see that

$$\begin{aligned} G_0 &= 1 \\ G_1 &= 0 \\ H_0 &= 1 \\ H_1 &= 2x_1^2x_2^2 \end{aligned} \tag{C.12}$$

are all Schur-concave as functions of x_1^2, x_2^2. This completes the proof. □

C.4 Proof of Theorem 6

Without loss of generality, we assume that $\|x\|_2 = 1$. From Theorem 5 and the fact that (x_i^2) majorizes $({x_i^*}^2)$ we obtain

$$\max_{\|x\|_0 \leqslant K} \mathbf{E}\left(\sum_{i<j} x_i x_i r_i r_j\right)^q = \mathbf{E}\left(\sum_{i<j} x_i^* x_i^* r_i r_j\right)^q = \mathbf{E}\left(\frac{1}{K}\sum_{1\leqslant i<j\leqslant K} r_i r_j\right)^q, \tag{C.13}$$

Observe that $r_i = 1 - 2b_i$ where (b_i) is a sequence of independent Bernoulli random variables with parameter $\frac{1}{2}$. Therefore,

$$\begin{aligned} \mathbf{E}\left(\sum_{i=1}^K r_i\right)^q = \\ &=^{(a)} \sum_{k\in\mathbb{Z}} k^q \cdot \mathbf{P}\left\{\sum_{i=1}^K r_i = k\right\} \\ &=^{(b)} \sum_k k^q \cdot \mathbf{P}\left\{\sum_{i=1}^K b_i = \frac{K-k}{2}\right\} \\ &=^{(c)} \sum_{i=0}^K (K-2i)^q \cdot \mathbf{P}\left\{\mathsf{Binom}\left(K, \frac{1}{2}\right) = i\right\} \\ &=^{(d)} \frac{1}{2^K}\sum_{i=0}^K \binom{K}{i}(K-2i)^q \\ &=^{(e)} \frac{1}{2^K}\sum_i \binom{K}{i}(-K+2i)^q, \end{aligned} \tag{C.14}$$

where in (a) we use the fact that $\sum_i r_i$ takes integer values, (b) follows by the identity $r_i = 1 - 2b_i$, (c) follows by $\mathsf{Binom}(K, 1/2) \sim \sum_{i=1}^K b_i$, (d) uses the explicit formula on the binomial probability mass function, and finally in (e) we substitute $i := K - i$ and use the symmetry of binomial coefficients $\binom{K}{i} = \binom{K}{K-i}$.

Using the above formula, we further calculate

$$\begin{aligned} \mathbf{E}\left(\sum_{1\leqslant i\neq j\leqslant K} r_i r_j\right)^q = \\ &=^{(a)} \mathbf{E}\left(\left(\sum_{i=1}^K r_i\right)^2 - \sum_{i=1}^K r_i^2\right)^q \\ &=^{(b)} \sum_j \binom{q}{j}(-K)^{q-j}\mathbf{E}\left(\sum_{i=1}^K r_i\right)^{2j} \\ &=^{(c)} \frac{1}{2^K}\sum_{i,j}\binom{q}{j}\binom{K}{i}(-K+2i)^{2j}(-K)^{q-j} \\ &=^{(d)} \frac{(-K)^q}{2^K}\sum_i \binom{K}{i}\left(1 - \frac{(-K+2i)^2}{K}\right)^q, \end{aligned} \tag{C.15}$$

where (a) follows by the square sum completion, (b) follows by the binomial formula and $r_i^2 = 1$, (c) follows directly by C.14, and (d) is obtained by algebraic rearrangements.

Inserting C.14 into C.13, we arrive at

$$\max_{x:\|x\|_0\leqslant K} \mathbf{E}\left(\sum_{1\leqslant i\neq j\leqslant K} x_i x_j r_i r_j\right)^q \quad = \quad \frac{1}{2^K}\sum_{i=0}^{K}\binom{K}{i}\left(\frac{(-K+2i)^2}{K}-1\right)^q. \tag{C.16}$$

To simplify further, let $Z \sim \frac{\mathsf{Binom}\left(K,\frac{1}{2}\right)-\frac{K}{2}}{\sqrt{\frac{K}{4}}}$ be the standardization of the symmetric binomial distribution. Denoting $i \sim \mathsf{Binom}\left(K,\frac{1}{2}\right)$ we have $Z^2 \sim \frac{\left(i-\frac{K}{2}\right)^2}{\frac{K}{4}} = \frac{(-K+2i)^2}{K}$, and we can rewrite C.16 as follows:

$$\max_{x:\|x\|_0\leqslant K} \mathbf{E}\left(\sum_{1\leqslant i\neq j\leqslant K} x_i x_j r_i r_j\right)^q = \mathbf{E}_Z\left(Z^2-1\right)^q, \tag{C.17}$$

which finishes the proof.

C.5 Proof of Theorem 4

We have to prove that for the distortion $E(\cdot)$ defined as in

$$E(x) \triangleq \frac{\|\Phi x\|^2}{\|x\|^2} - 1. \tag{C.18}$$

the following inequality holds true:

$$E(x) \leqslant E(y), \quad (y_i^2) \prec (x_i^2). \tag{C.19}$$

The proof goes through several reduction steps until Schur-concavity of few simple functions.

We first observe that it suffices to prove that the moments of the expression

$$x \to \|\Phi x\|^2 - \|x\|^2, \tag{C.20}$$

are Schur-concavity with respect to (x_i^2). Indeed, since $(y_i^2) \prec (x_i^2)$ implies $\|x\|^2 = \sum_i x_i^2 = \sum_i y_i^2 = \|y\|^2$ we have $\mathbf{E}E(x)^q \leqslant \mathbf{E}E(y)^q$ if and only if $\mathbf{E}(\|\Phi x\|^2 - \|x\|^2)^q \leqslant \mathbf{E}(\|\Phi y\|^2 - \|y\|^2)^q$, by the definition of E.

We first prove that the distortion of m-dimensional projections is the average of m IID distortions of 1-D projections. Observe that

$$\|\Phi x\|^2 - \|x\|^2 = \sum_{k=1}^{m}\left((\Phi_k x)^2 - \mathbf{E}(\Phi_k x)^2\right), \tag{C.21}$$

where Φ_k is the k-th row of Φ; this follows by $\mathbf{E}(\Phi_k x)^2 = \sum_i x_i^2 \mathbf{Var}[\Phi_{k,i}] = \frac{1}{m}\|x\|^2$. Furthermore, the summands in (C.21) are independent and identically distributed:

$$(\Phi_k x)^2 - \mathbf{E}(\Phi_k x)^2 \sim \frac{1}{m}\sum_{i\neq j} x_i x_j r_i r_j. \tag{C.22}$$

Then we note that the Schur-concavity test can be done on the 1-D case. This follows because, due to the multinomial expansion applied to C.22, the q-th moment of m-dimensional distortion is a multivariate polynomial in 1-D distortion moments of order $k \leqslant q$, with non-negative coefficients; the distortion moments are themselves non-negative, and by Lemma 6 and Theorem 5 we obtain the first part of the theorem.

Finally, applying Theorem 6 proves the second part.

Bibliography

A., Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*.

Achlioptas, Dimitris (2001). "Database-friendly random projections". In: *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 274–281.

— (2003). "Database-friendly random projections: Johnson-Lindenstrauss with binary coins". In: *Journal of computer and System Sciences* 66.4, pp. 671–687.

Agarwal, Ashish (2019). "Static Automatic Batching In TensorFlow". In: *ICML*, pp. 92–101.

Ailon, Nir and Bernard Chazelle (2006). "Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform." In: *STOC*, pp. 557–563.

Arpit, Devansh, Víctor Campos, and Yoshua Bengio (2019). "How to Initialize your Network? Robust Initialization for WeightNorm & ResNets". In: *NeurIPS*, pp. 10900–10909.

Bachlechner, Thomas et al. (2020). "ReZero is All You Need: Fast Convergence at Large Depth". In: *CoRR* abs/2003.04887.

Balduzzi, David et al. (2017). "The Shattered Gradients Problem: If resnets are the answer, then what is the question?" In: *ICML*, pp. 342–350.

Banko, M. et al. (2007). "Open Information Extraction from the Web." In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

Bingham, Ella and Heikki Mannila (2001). "Random projection in dimensionality reduction: applications to image and text data." In: *SIGKDD*, pp. 245–250.

Blocki, J. et al. (2012). "The Johnson-Lindenstrauss transform itself preserves differential privacy." In: *Proceedings of the 2013 IEEE 53td Annual Symposium on Foundations of Computer Science*.

Bojanowski, P. et al. (2017). "Enriching Word Vectors with Subword Information." In: *Trans. Assoc. Comput. Linguistics*.

Boros, E., J. Moreno, and A. Doucet (2021). "Event Detection with Entity Markers." In: *Proceedings of the 43rd European Conference on IR Research*.

Botev, Aleksandar, Hippolyt Ritter, and David Barber (2017). "Practical Gauss-Newton optimisation for deep learning". In: *ICML*. PMLR, pp. 557–565.

Boucheron, S. et al. (2005). "Moment inequalities for functions of independent random variables." In: *Ann. Probab.* 33.2, pp. 514–560.

Boucheron, Stéphane, Gábor Lugosi, and Olivier Bousquet (2003). "Concentration inequalities". In: *Summer school on machine learning*, pp. 208–240.

Broyden, Charles George (1970). "The convergence of a class of double-rank minimization algorithms". In: *Journal of the Institute of Mathematics and Its Applications* 6, pp. 76–90.

Burr, Michael, Shuhong Gao, and Fiona Knoll (2018). "Optimal bounds for Johnson-Lindenstrauss transformations." In: *The Journal of Machine Learning Research* 19.1, pp. 2920–2941.

Carlson, A. et al. (2010). “Toward an architecture for never-ending language learning.” In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. ACM.

Cauchy, Louis Augustin (1874). “Compte Rendu à l’Académie des Sciences”. In.

Clarkson, K. L. and D. P. Woodruff (2017). “Low-rank approximation and regression in input sparsity time.” In: *Journal of the ACM* 63.6, pp. 1–45.

Cohen, Michael B, TS Jayram, and Jelani Nelson (2018). “Simple analyses of the sparse Johnson-Lindenstrauss transform.” In: *1st Symposium on Simplicity in Algorithms (SOSA 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Corro, L. Del and R. Gemulla (2013). “ClausIE: Clause-based Open Information Extraction”. In: *22nd International World Wide Web Conference*.

Dasgupta, S. and A. Gupta (1999). “An elementary proof of the Johnson-Lindenstrauss lemma.” In: *International Computer Science Institute, Technical Report* 22.1, pp. 1–5.

Dauphin, Yann N. and Samuel S. Schoenholz (2019). “MetaInit: Initializing learning by learning to initialize”. In: *NeurIPS*. Ed. by Hanna M. Wallach et al., pp. 12624–12636.

Davis, T. A. and Y. Hu (2011). “The University of Florida sparse matrix collection.” In: *Annals of Probability* 38.1, pp. 1–25.

Deng, J., J. Krause, and L. Fei-Fei (2013). “Fine-Grained Crowdsourcing for Fine-Grained Recognition.” In: *CVPR*.

Devlin, J. et al. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Duchi, J., E. Hazan, and Y. Singer (2011). “Adaptive Subgradient methods for online learning and stochastic optimization”. In: *Journal of Machine Learning Research* 12, pp. 2121–2159.

Eaton, Morris L (1970). “A note on symmetric Bernoulli random variables”. In: *The annals of mathematical statistics* 41.4, pp. 1223–1226.

Edelman, Alan (1988). “Eigenvalues and condition numbers of random matrices”. In: *SIAM Journal on Matrix Analysis and Applications* 9.4, pp. 543–560.

Efron, Bradley (1968). *Student’s t-test under non-normal conditions*. Tech. rep. HARVARD UNIV CAMBRIDGE MA DEPT OF STATISTICS.

Etzioni, O. et al. (2004). “Web-scale information extraction in knowitall.” In: *International World Wide Web Conference*. ACM.

Fader, A., S. Soderland, and O. Etzioni (2011). “Identifying Relations for Open Information Extraction.” In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. ACL.

Farima, F. Bayat, N. Bhutani, and H. Jagadish (2022). “CompactIE: Compact Facts in Open Information Extraction.” In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL.

Fei-Fei, L., R. Fergus, and P. Perona (2004). “Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories.” In: *CVPR*.

Fletcher, Roger (1970). “A new approach to variable metric algorithms”. In: *Computer journal* 13(3), pp. 317–322.

Frankl, P. and H. Maehara (1988). “The Johnson-Lindenstrauss lemma and the sphericity of some graphs.” In: *Journal of Combinatorial Theory, Series B* 44.3, pp. 355–362.

Freksen, C. B., L. Kamma, and K. G. Larsen (2018). "Fully understanding the hashing trick." In: *Proceedings of the thirty-first Advances in Neural Information Processing Systems*.

— (2019). "Understanding Sparse JL for Feature Hashing." In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*.

Gardner, M. and T. Mitchell (2015). "Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. ACL.

Gashteovski, K., R. Gemulla, and L. Del Corro (2017). "MinIE: Minimizing Facts in Open Information Extraction." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. ACL.

Gashteovski, K. et al. (2020). "On Aligning OpenIE Extractions with Knowledge Bases: A Case Study." In: *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*. ACL.

Ghorbani, Behrooz, Shankar Krishnan, and Ying Xiao (2019). "An investigation into neural net optimization via Hessian eigenvalue density". In: *ICML*. PMLR, pp. 2232–2241.

GitHub (2019a).

— (2019b). https://github.com/tensorflow/tensorflow/issues/675.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *AISTATS*, pp. 249–256.

Goldfarb, Donal (1970). "A family of variable metric updates derived by variational means". In: *Mathematics of Computation* 24(109), pp. 23–26.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press.

Han, X. et al. (2018). "FewRel: A Large-Scale Supervised Few-shot Relation Classification Dataset with State-of-the-Art Evaluation." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. ACL.

Hanson D. L. Wright, F. T. (1971). "A bound on tail probabilities for quadratic forms in independent random variables." In: *Ann. math. stat.* 42.3, pp. 1079–1083.

He, Kaiming et al. (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *ICCV*, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

Hendrycks, Dan and Kevin Gimpel (2016). "Adjusting for dropout variance in batch normalization and weight initialization". In: *CoRR* abs/1607.02488.

Indyk, P. and R. Motwani (1998). "Approximate nearest neighbors: towards removing the curse of dimensionality." In: *The 30th ACM Symposium on Theory of Computing*.

Jacot, Arthur, Franck Gabriel, and Clément Hongler (2020). "The asymptotic spectrum of the Hessian of DNN throughout training". In: *ICLR*. OpenReview.net.

Jayram, T. S. and David Woodruff (2013). "Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error." In: *Transactions on Algorithms* 9.3, pp. 1–17.

Johnson, W. B. and A. Naor (2010). "The Johnson-Lindenstrauss lemma almost characterizes Hilbert space, but not quite." In: *Discrete & Computational Geometry* 43.3, pp. 542–553.

Johnson, William B. and Joram Lindenstrauss (1984). "Extensions of Lipschitz mappings into a Hilbert space." In: *Contemporary Mathematics* 26.1, pp. 189–206.

Joshi, M. et al. (2020). "SpanBERT: Improving Pre-training by Representing and Predicting Spans." In: *Trans. Assoc. Comput. Linguistics*.

Kane, D. M. and J. Nelson (2010). "Derandomized sparse Johnson-Lindenstrauss transform." In: *CoRR* abs/1006.3585.

— (2014). "Sparser Johnson-Lindenstrauss transforms." In: *Journal of the ACM* 61.1, pp. 1–23.

Kargin, Vladislav et al. (2010). "Products of random matrices: Dimension and growth in norm". In: *The Annals of Applied Probability* 20.3, pp. 890–906.

Karpathy, Andrej (2015). *char-rnn*. `https://github.com/karpathy/char-rnn`.

KBP (2017). "https://tac.nist.gov/2017/KBP/ColdStart/". In.

Ken, L. (1995). "Newsweeder: learning to filter netnews." In: *ICML*.

Kenthapadi, K. et al. (2013). "Privacy via the Johnson-Lindenstrauss Transform." In: *Journal of Privacy and Confidentiality* 5.1, pp. 39–71.

Kingma, Diederik P. and Jimmy Lei Ba (2015). "ADAM: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*.

Krizhevsky, A. (2012). "Learning Multiple Layers of Features from Tiny Images." In: *University of Toronto (Technical Report)*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*, pp. 1106–1114.

Lan, Z. et al. (2020). "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations." In: *8th International Conference on Learning Representationss*.

Latala R., et al. (1997). "Estimation of moments of sums of independent real random variables." In: *Annals of Probability* 25.3, pp. 1502–1513.

LeCun, Y. and C. Cortes (2010). "MNIST handwritten digit database". In.

Lehmann, J. et al. (2015). "DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia." In: *Semantic Web Journal* 6(2), pp. 167–195.

Levy, O. and Y. Goldberg (2014). "Dependency-Based Word Embeddings." In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. ACL.

Li, B. et al. (2020). "Efficient One-Pass End-to-End Entity Linking for Questions." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. ACL.

Li, P., T. J. Hastie, and K. W. Church (2006). "Very sparse random projections." In: *Proceedings of the twelfth SIGKDD international conference on Knowledge Discovery and Data Mining*.

Linial, N., E. London, and Y. Rabinovich (1995). "The geometry of graphs and some of its algorithmic applications." In: *Combinatorica* 86.15, pp. 215–245.

Liu, Donc C. and Jorge Nocedal (1989). "On the limited memory method for large scale optimization". In: *Mathematical Programming B*. 45(3), pp. 503–528.

Liu, Y. et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach." In: *CoRR, vol.abs/1907.11692*.

Lockard, C., P. Shiralkar, and X. L. Dong (2019). "When Open Information Extraction Meets the Semi-Structured Web." In: *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Lopes, M., L. Jacob, and M. J. Wainwright (2011). "A more powerful two-sample test in high dimensions using random projection." In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*.

Makarychev, K., Y. Makarychev, and I Razenshteyn (2019). "Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering." In: *The 51st ACM Symposium on Theory of Computing*.

Manning, C. D. et al. (2014). "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Marcheggiani, D. et al. (2017). "Semantic Role Labeling." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Martens, James and Roger Grosse (2015). "Optimizing neural networks with kronecker-factored approximate curvature". In: *ICML*. PMLR, pp. 2408–2417.

Martínez-Rodríguez, J. L., I. López-Arévalo, and A. B. Ríos-Alvarado (2018). "OpenIE-based approach for Knowledge Graph construction from text." In: *Expert Syst. Appl.*

Mausam (2016). "Open Information Extraction Systems and Downstream Applications." In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*.

Mesquita, F. et al. (2019). "KnowledgeNet: A Benchmark Dataset for Knowledge Base Population." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. ACL.

Mikolov, T. et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality." In: *27th Annual Conference on Neural Information Processing Systems*.

Minsky, M. and S. Papert (1972). *Perceptrons: An Introduction to Computational Geometry*. The MIT Press.

Mintz, M. et al. (2009). "Distant supervision for relation extraction without labeled data." In: *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNL*. ACL.

Nelson, J. and H. L. Nguyên (2013). "OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings." In: *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*.

Netzer, Y. et al. (2011). "Reading Digits in Natural Images with Unsupervised Feature Learning." In.

Nguyen, D.B. et al. (2014). "AIDA-light: High-Throughput Named-Entity Disambiguation." In: *Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference*.

Nocedal, Jorge and Stephen J. Wright (2006). *numerical optimization*. Springer.

Pennington, J., Socher, and D. C. Manning (2014). "GloVe: global vectors for word representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Peters, M. E. et al. (2018). "Deep Contextualized Word Representations." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. ACL.

Radford, A. et al. (2019). "Language Models are Unsupervised Multitask Learners." In.

Rehurek, R. and P. Sojka (2011). "Gensim–python framework for vector space modelling." In: *NLP Centre, Faculty of Informatics*.

Reimers, N. and I. Gurevych (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Proceedings of the 2019 Conference on Natural Language Processing*.

Robbins, Herbert E. (1951). "A Stochastic Approximation Method". In: *Annals of Mathematical Statistics*.

Rosenblatt, Frank (1958). “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65.6, 386–408.

Sachan, D. S. et al. (2021). “Do Syntax Trees Help Pre-trained Transformers Extract Information?” In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*. ACL.

Sagun, Levent et al. (2017). “Empirical Analysis of the Hessian of Over-Parametrized Neural Networks”. In: *CoRR* abs/1706.04454. arXiv: 1706.04454.

Salvatier, John, Thomas V. Wiecki, and Christopher Fonnesbeck (2016). “Probabilistic programming in Python using PyMC3”. In: *PeerJ Comput. Sci.* 2, e55. DOI: 10.7717/peerj-cs.55.

Sanh, V. et al. (2020). “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.” In: *CoRR, vol. abs/1910.01108*.

Sarlos, T. (2006). “Improved approximation algorithms for large matrices via random projections.” In: *Proceedings of the 2013 IEEE 47th Annual Symposium on Foundations of Computer Science*.

Schweter, S. and A. Akbik (2020). “FLERT: Document-Level Features for Named Entity Recognition.” In: *CoRR abs/2011.06993*.

Shanno, David F. (1970). “Conditioning of quasi-Newton methods for function minimization”. In: *Mathematics of Computation* 24(111), pp. 647–656.

Shao, Q. (2000). “A comparison theorem on moment inequalities between negatively associated and independent random variables.” In: *Journal of Theor. Probab.* 13.2, pp. 343–356.

Shi, C., W. Lu, and R. Song (2020). “A sparse random projection-based test for overall qualitative treatment effects.” In: *Journal of the American Statistical Association* 115.531, pp. 1201–1213.

Silverstein, Jack W (1994). “The spectral radii and norms of large dimensional noncentral random matrices”. In: *Stochastic Models* 10.3, pp. 525–532.

Soares, L.B. et al. (2019). “Matching the Blanks: Distributional Similarity for Relation Learning.” In: *Proceedings of the 57th Conference of the Association for Computational Linguistics*. ACL.

Stanovsky, G., I. Dagan, and Mausam (2015). “Open IE as an Intermediate Structure for Semantic Tasks.” In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*. ACL.

Suchanek, F. M., G. Kasneci, and G. Weikum (2007). “Yago: a core of semantic knowledge.” In: *Proceedings of the 16th international conference on World Wide Web*.

Szegedy, Christian et al. (2013). “Intriguing properties of neural networks”. In: *CoRR* abs/1312.6199.

Tieleman, T. and G. Hinton (2012). “Divide the gradient by running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2, pp. 26–31.

Trisedya, B. D. et al. (2019). “Neural Relation Extraction for Knowledge Base Enrichment.” In: *Proceedings of the 57th Conference of the Association for Computational Linguistics*. ACL.

Tunstall, L. et al. (2022). “Efficient Few-Shot Learning Without Prompts.” In: *CoRR abs/2209.11055*.

V., Suresh and Qiushi W. (2011). “The Johnson-Lindenstrauss Transform: An Empirical Study.” In: *Proceedings of the workshop on Algorithm Engineerging and Experiments*.

Virmaux, Aladin and Kevin Scaman (2018). “Lipschitz regularity of deep neural networks: analysis and efficient estimation”. In: *NeurIPS*, pp. 3839–3848.

Vrandečić, D. and M. Krötzsch (2014). “Wikidata: a free collaborative knowledgebase.” In: *Communications of the ACM*.

Wand, A. et al. (2019). “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.” In: *7th International Conference on Learning Representations*.

Wang, C. et al. (2012). “Relation extraction and scoring in DeepQA.” In: *J. Res. Dev.*

Wang, C. et al. (2021a). “Zero-Shot Information Extraction as a Unified Text-to-Triple Translation.” In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. ACL.

Wang, Y. et al. (2021b). “Generalizing from a Few Examples: A Survey on Few-shot Learning”. In: *ACM Computing Surveys* 53(3), pp. 1–34.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms”. In: *CoRR* abs/1708.07747. arXiv: 1708.07747.

Xiao, Lechao et al. (2018). “Dynamical Isometry and a Mean Field Theory of CNNs: How to Train 10, 000-Layer Vanilla Convolutional Neural Networks”. In: *ICML*, pp. 5389–5398.

Xu, Bing, Ruitong Huang, and Mu Li (2016). “Revise Saturated Activation Functions”. In: *CoRR* abs/1602.05980.

Xu, K. et al. (2016). “Enhancing Freebase Question Answering Using Textual Evidence”. In: *CoRR*.

Yang, Z. et al. (2019). “XLNet: Generalized Autoregressive Pretraining for Language Understanding.” In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*.

Yao, Zhewei et al. (2021). “ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, D. et al. (2019a). “OpenKI: Integrating Open Information Extraction and Knowledge Bases with Relation Inference.” In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Zhang, Z. et al. (2019b). “ERNIE: Enhanced Language Representation with Informative Entities.” In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. ACL.

Zhou, S. (2019). “Sparse Hanson-Wright inequalities for subgaussian quadratic forms.” In: *Bernoulli Society for Mathematical Statistics and Probability* 25.3, pp. 1603–1639.

Zhou, W. and C. Muhao (2021). “An Improved Baseline for Sentence-level Relation Extraction.” In: *CoRR, vol. abs/2102.01373*. ACL.