NYILATKOZAT

Név: Mészáros Anna

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: GPWDPB

Szakdolgozat címe:
Inductive bias of Riemannian gradient flow in deep linear neural networks

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2022.05.31.

Mészáros Anna

a hallgató aláírása

Inductive bias of Riemannian gradient flow in deep linear neural networks

Author:

Mészáros Anna

Pure Mathematics BSc

External supervisor:

Dr. Huszár Ferenc

Associate Professor

University of Cambridge

Internal consultant:

Dr. Lukács András

Assistant Professor

Eötvös Loránd University

Eötvös Loránd University

Faculty of Science

Budapest, 2022

Contents

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Ferenc Huszár, for the continuous help he provided in the past two years. Not only did he introduce me to this topic, but he constantly has an eye on my studies towards deep learning. I am especially thankful for the research opportunities he provides, which resulted in my first paper. This evolved to be the base of my thesis. Furthermore, I want to thank András Lukács for accepting this thesis and providing the administrative background. Last but not least, I owe special thanks to my family and my friends for their support and motivation.

1 Introduction

Deep learning is a commonly used algorithm type, its key attribution is learning through examples. For instance, with an adequate amount of pictures of dogs and cats, the computer can learn to distinguish between them in the case of previously unseen images. This behavior is quite similar to human learning, moreover, with deep learning computers are able to achieve human-level performance, sometimes even exceed it (Alzubaidi et al., 2021).

Without deep learning, many digital developments would be unimaginable. It is the key technology behind certain medical research (Nagpal et al., 2019). Learning from examples, computers can detect brain tumors on MRI images with high accuracy, which can promote and accelerate healing. Another real-life example where deep learning is essential is marketing. Such algorithms analyze the individual's preferences through social media or search history with the aim of personalization. The result is accurate and successful advertising. Furthermore, deep learning brought the breakthroughs to self-driving cars, which includes recognizing traffic signs and identifying pedestrians. This technique is not only beneficial for self-controlling but also for preventing accidents. To illustrate the advantages listed above, He et al., 2020 presented a deep learning model called Mask R-CNN which is able to detect objects accurately such as cars, pedestrians and traffic lights, and simultaneously classify each pixel whether it is a part of a particular object or not.

Despite deep learning being able to perform surprisingly well, the mathematical explanation of the phenomenon is quite poor. In other words, we lack deep understanding of generalization, therefore it is unknown exactly in which situations will deep learning work and in which situations it won't. Since deep learning is based on optimization, specially on gradient descent related algorithms, which are geometrically motivated, it is justified to study the optimization algorithms from a geometrical point of view. This aspect could provide answers to the question of generalization. This thesis examines the possible reasons why deep learning works and investigates what role certain aspects play in it in a way was descried above. More detailed introduction can be found in Section 3 after introducing some essential concepts.

2 Preliminaries

Before the main topic, we review some relevant concepts of deep learning as the probabilistic model of deep learning, linear neural networks, generalization, and the definitions of inductive bias, explicit and implicit regularization.

2.1 Probabilistic model

Consider a problem with a given labeled data set $\mathcal{D} = \{(x_n, y_n)_{\mathbf{n=1}}^N\}$, where x_n is the nth data point and y_n is the corresponding label. We call $\mathcal{D}$ the **training set** and x_n a **training point**.

Denote the vector of x_n with $\boldsymbol{x}$ and the vector of y_n with $\boldsymbol{y}$, then the N pairs of samples $\boldsymbol{x}, \boldsymbol{y}$ come from an unknown distribution $Pr_{\mathcal{D}}(X, Y)$, and we use the assumption that (x_n, y_n) are independent and identically distributed for $n = 1, \ldots, N$. In order to predict the label y for a previously unseen x, we wish to estimate the distribution of y given x. Specially, consider an estimating **model** $Pr(Y|X = x; \theta)$, which is a family of probability distributions parametrized by θ. The goal is to fit the model to the training data modifying the parameter θ. **Fitting a probabilistic model to the data** means maximizing the log-likelihood of the dataset. Let $p(y|x; \theta)$ and $p(\boldsymbol{y}|\boldsymbol{x}; \theta)$ be the corresponding probability density functions for one and N samples respectively parametrized by θ. We now can write $p(\boldsymbol{y}|\boldsymbol{x}; \theta)$ as a product.

Statement 1. $p(\boldsymbol{y}|\boldsymbol{x}; \theta) = \Pi_{i=1}^N p(y_i|x_i; \theta)$

Proof.

$$p(\boldsymbol{y}|\boldsymbol{x}; \theta) = \frac{p(\boldsymbol{x}, \boldsymbol{y}; \theta)}{p(\boldsymbol{x}; \theta)} = \frac{p(\boldsymbol{x}, \boldsymbol{y}; \theta)}{\int \cdots \int p(\boldsymbol{x}, t_1, \ldots, t_N; \theta)\, dt_1 \ldots, dt_N} \overset{\text{iid}}{=}$$

$$\overset{\text{iid}}{=} \frac{p(\boldsymbol{x}, \boldsymbol{y}; \theta)}{\int \cdots \int \Pi_{i=1}^N p(x_i, t_i; \theta)\, dt_1 \ldots, dt_N} \overset{\text{iid}}{=} \frac{\Pi_{i=1}^N p(x_i, y_i; \theta)}{\Pi_{i=1}^N p(x_i; \theta)} = \Pi_{i=1}^N p(y_i|x_i; \theta)$$

□

Utilizing the previous statement we can derive the maximum log-likelihood

$$\begin{aligned}\underset{\theta}{\operatorname{argmax}} \log p(\boldsymbol{y}|\boldsymbol{x};\theta) = \underset{\theta}{\operatorname{argmax}} \log \Pi_{i=1}^{N} p(y_i|x_i;\theta) = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \log p(y_i|x_i;\theta) = \\ = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} -\log p(y_i|x_i;\theta)\end{aligned} \tag{2.1}$$

2.2 Gradient based learning

If we define the loss function of a single data point as the negative log-likelihood

$$\ell(\theta) = \ell(x, y, \theta)) := -\log p(y|x;\theta) \tag{2.2}$$

then the **loss function** of the whole data set is

$$\mathcal{L}(\theta) = \mathcal{L}(\boldsymbol{y}, \boldsymbol{x}, \theta) := \sum_{n=1}^{N} \ell(x_n, y_n, \theta) = \sum_{n=1}^{N} -\log p(y_n|x_n;\theta) \tag{2.3}$$

Therefore minimizing the prediction loss is equivalent to maximizing the log-likelihood. In the case of the examined problems, we also consider the previous model with appropriate estimating probability distributions.

Remark. We often modify θ with a neural network f that depends on the data and has own parameters ρ: $\theta = f(\boldsymbol{x}, \rho)$. In this case we adjust the parameters ρ through the loss function, therefore we rather consider the loss as a function of ρ. Thus Equation 2.3 becomes $\mathcal{L}(\rho) = \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{x}, \rho)) := \sum_{n=1}^{N} \ell(x_n, y_n, \rho) = \sum_{n=1}^{N} -\log p(y_n|x_n;\rho)$.

Minimizing the loss function often happens with (Euclidean) **gradient descent** (EGD). After initializing the parameter to θ_0 and picking an arbitrary **learning rate** $\eta \in \mathbb{R}$, in each step the parameter is updated as

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta), \tag{2.4}$$

where $\nabla_\theta \mathcal{L}(\theta)$ is the gradient of the loss w.r.t. θ. Consider the (Euclidean) **gradient flow** (EGF) the continuous time limit of EGD, defined as

$$\dot{\theta} = -\eta \nabla_\theta \mathcal{L}(\theta) \tag{2.5}$$

Remark. **Stochastic gradient descent** (SGD): let the training set be split perfectly by m minibatches such that the batch size is $B = N/m$, then each gradient descent step is evaluated only on a randomly chosen batch. This method is able to improve the generalization (Roberts, 2018).

Remark. When all the activation functions of the neural network are the identity, therefore f_ρ is a linear function, we call it **linear neural network**.

2.3 Generalization

After the training, we wish to evaluate the model's generalization, namely the performance on the unseen data, which is sampled from the same distribution as the training data. We call it the **test set**. In the evaluation, the key point is the **generalization error** (or test error), which is either the loss function calculated on the test set or it can be another function evaluated on the test data. Obviously, the lower the generalization error is, the better the model generalizes.

In spite of the low training error, high test error can occur. The reason behind this phenomenon is **overfitting**, which means there are too many parameters, the model is complex enough to 'memorize' the training data, but therefore it performs poorly on the test data. To avoid this scenario **regularization** techniques are used during the training, resulting in the prevention of overfitting, simpler models, and improved generalization.

2.4 Inductive bias

Examining the generalization performance of a neural network, an essential concept is **inductive bias**. Since infinitely many models can fit the training data, if the learning algorithm treated equally all the possible models, then resulting in an appropriate one, thus low generalization error would be impossible. All properties of the learning algorithm that influence which model will be selected, excluding the training set, are called the inductive bias. This could include the parametrization of the model, regularization terms, or it might be encoded in the architecture itself. Battaglia et al., 2018 say 'Ideally, inductive biases both improve the search for solutions without substantially diminishing performance, as well as help find solutions

which generalize in a desirable way; however, mismatched inductive biases can also lead to suboptimal performance by introducing constraints that are too strong.'

As mentioned above, inductive bias includes regularization techniques. Within this, explicit and implicit regularization are distinguished, but often exact definitions are not provided. In this thesis we consider the definition proposed by Hernández-García and König, 2018. Accordingly:

- **Explicit regularization** refers to those techniques which constrain the number of models that the learning algorithm can choose. Namely, the hypothesis set of the explicitly regularized neural network is a proper subset of the original hypothesis set. For instance, regularizing the ℓ_2-norm of the weights of the network, called weight decay reduces the achievable functions. So does the dropout technique, which resets some weights to 0.
- **Implicit regularization**, on the contrary, does not reduce the hypothesis set, but still lowers the generalization error, and prevents overfitting. In addition, this is rather an effect than a technique. One of the most common implicit regularization methods is stochastic gradient descent, which reduces the generalization error without explicitly reducing the achievable functions. Another common implicit regularization is early stopping.

This work will focus on the implicit regularization effects, especially the parametrization of the model.

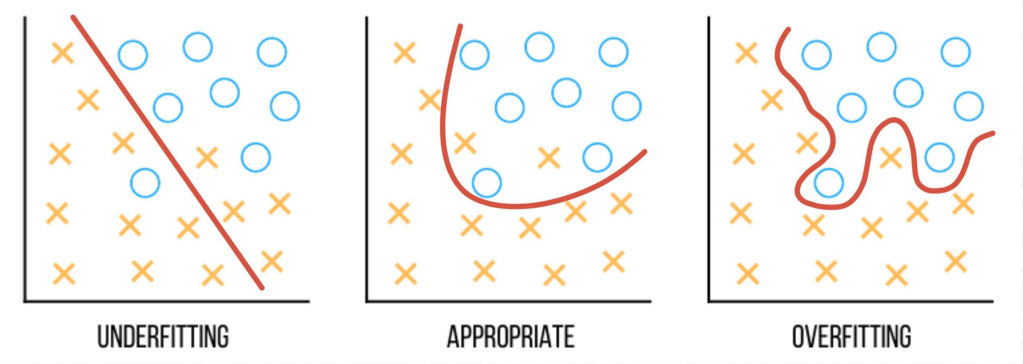

Figure 2.1: Illustration of the bias-variance tradeoff

The model should be complex enough to be able to express the training data, scoring low training error, but it should be simple enough to prevent overfitting. Regularization techniques are used to avoid overfitting in over-parametrized models. The figure on the left shows an underfitting model, while the model on the right is overfitting. In the figure in middle, an appropriate model can be seen.

3 Thesis overview

3.1 Problem statement

The main goal of learning problems is to fit a model to the training data which has small generalization error and performs well on the test data. However, several models can fit perfectly to the training points, it is up to the inductive bias to select one. In general, deep learning generalizes surprisingly well, e. g. Zawadzka-Gosk, Wołk, and Czarnowski, 2019 showed that state-of-the-art networks for image classification can exceed 99% accuracy, and in the Introduction we also listed examples of great performance. Therefore we can conclude that the inductive bias in most cases is able to select the right model from the interpolation regime, but the explanations for this phenomenon are quite poor. It is not exactly known why can deep learning generalize so well, better than we might expect. To answer this question further investigation is necessary.

Conventional wisdom attributed the success in generalization to the properties of the model family and to the regularization techniques used during training. But Zhang et al., 2017 have shown this is not exactly the case, these approaches fail to explain the generalization in deep learning, as they state "Explicit regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error". They showed counterexamples for both directions' statements: without explicit regularization techniques over-parametrized neural networks are still able to generalize (even though with classical theory the expectation is overfitting), and neural networks with architectures including explicit regularization can fit to randomly labeled data, therefore these completely fail to generalize.

Consequently, a new idea occurred: the answer might be found in the role of implicit regularization. There are three aspects of implicit regularization which are usually considered in the literature: initialization, built-in properties of stochastic gradient descent, and parameter-to-hypothesis mapping. This work focuses on the latter, we examine its role in order to gain a deeper understanding of the great generalization performance in deep learning.

The parameter-to-hypothesis mapping indeed has a role in inductive biases as several empirical and theoretical works suggested. Arora et al., 2019 showed that gradient descent is able to recover low-rank matrices from a few observations even without any regularization if the parametrization is matrix factorization. Furthermore, with appropriate parametrization, sparse solutions can be found effectively in separable classification problems with unregularized gradient descent (Gunasekar et al., 2018). Therefore one can claim that the parametrization of a problem influences the inductive bias, and as the previous results show, it could play a determining role in generalization.

Besides that, a new optimizing algorithm has become popular called natural gradient descent (NGD). Intuitively, this is similar to the original euclidean gradient descent (EGD) in the way that both algorithms take steps towards a minimum of a function, but while the gradient descent is considered in the Euclidean space with Euclidean distance, the other moves on a Riemannian manifold with adequate distance metric. Originally, the favorable feature of NGD is the fast convergence, but its exact computation could be intractable. From the perspective of parameter-to-hypothesis mapping, the key property of NGD is the fact that it is approximately invariant to reparametrization.

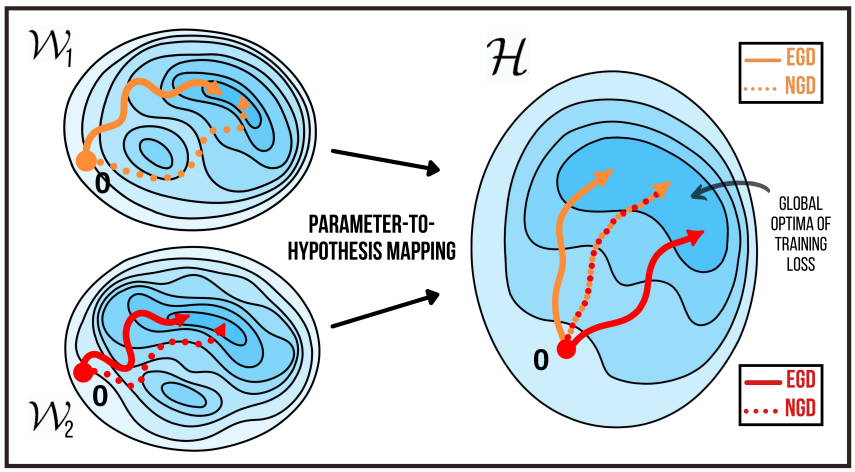

Figure 3.1: Parametrization-dependence of EGD and independence of NGD

In Figure 3.1, the illustration of the parametrization-dependence of EGD and independence of NGD can be found. If we consider two parameter spaces ($\mathcal{W}_1$, $\mathcal{W}_2$) and two optimization trajectories in each: one EGD, one NGD, and we map these into the hypothesis space ($\mathcal{H}$) then EGD finds different optima, but NGD finds the same.

Considering the previous two key statements together:

- Parameter-to-hypothesis mapping influences the inductive bias and the generalization in gradient-based learning
- Natural gradient descent eliminates the role of parametrization

These claims together naturally raise questions: what happens if we eliminate the role of parametrization and what can be said about the inductive bias of natural gradient descent. By investigating the behavior of NGD we are able to study the importance of parametrization. If the parameter-to-hypothesis mapping indeed plays a determining role in generalization then eliminating its effect completely may be undesirable and might question the necessity of the efforts put into the computation of NGD.

3.2 Contributions

This thesis examines the inductive bias of natural gradient descent in deep linear neural networks and simultaneously studies the role of parametrization. There are two reasons why we investigate linear neural networks: in the case of linear neural networks there are tasks where the inductive bias of EGD is well known (e. g. matrix completion and separable classification) and the natural gradient direction is more computable then in actual non-linear neural networks.

To put the contributions in context we review some essential background. In Section 2 we introduced some fundamental concepts of deep learning as the probabilistic model of learning problems, gradient descent, the method of generalization and the ideas behind inductive bias. Section 4 is about a generalization of the gradient descent algorithm, namely minimizing a function on a Riemannian manifold. We introduce its basic concepts, define the gradient on such a manifold then determine how to move "straight" from a point in direction of a vector. Furthermore, we review one special case the natural gradient descent and discuss its invariance property. In Section 5 there can be found the introductions of two commonly considered underdetermined problems: separable classification and matrix completion. We discuss which solutions are found regarding two different parametrizations with EGD in each case. Moreover, Sections 6 and 7 are about the findings related to NGD in separable classification and in matrix completion respectively. Here we make the fol-

lowing contributions based on the paper Kerekes, Mészáros, and Huszár, 2021. These results were obtained with equal contribution of the paper's additional author.

- We calculate the Fisher information matrix of the probabilistic models in separable classification and matrix completion,
- we prove an invariance property for separable classification with a linear model, namely that NGF is invariant under invertible transformations of the data,
- we show that with NGF the logits interpolate the training labels when the number of data points is less than the number of parameters, and we propose a conjecture that the separator converges in direction to the ordinary least squares solution, and
- in the case of matrix completion NGF filds the trivial solution, therefore it fails to generalize completely.

4 Riemannian gradient descent

Regarding optimization, one commonly used method is Euclidean gradient descent. In this case, the optimization trajectory lies in the Euclidean space equipped with the Euclidean metric. However, there exist scenarios when considering another space with another geometry is more beneficial. In the next section, we introduce a more general gradient descent algorithm defined on Riemannian manifolds based on the sources: Verhóczki, 2020; O'Neill, 1983; Absil, Mahony, and Sepulchre, 2008 and Kristiadi, 2019.

Generalizing the gradient descent to functions defined on a Riemannian manifold, two steps are necessary: defining the gradient, and moving in that direction from a point on the manifold.
First, our main interest is to define the gradient of a function on Riemannian manifolds. Before doing so, we review some essential definitions and statements related to differential geometry.

Definition 1. $\mathcal{M}$ topological space is an m-dimensional **differentiable manifold** if (1) $\mathcal{M}$ is Hausdorff,
(2) the topology of $\mathcal{M}$ has a countable base,
(3) an arbitrary point on $\mathcal{M}$ has an open neighborhood that is homeomorphic to an open subset of $\mathbb{R}^m$,
(4) and $\mathcal{M}$ is equipped with a maximal smooth atlas $\mathcal{A}$.

Definition 2. Let U be an open subset of the m-dimensional manifold $\mathcal{M}$. If the mapping $x : U \to \mathbb{R}^m$ is a homeomorphism between U and $x(U) \subset \mathbb{R}^m$, then the tuple (U, x) is a **map/local coordinate-system** of the manifold $\mathcal{M}$.

Consider a $(U, x) \in \mathcal{A}$ local coordiante-system on $\mathcal{M}$, and let $x^i = u^i \circ x$ $(i = 1, \ldots, m)$ be its coordinate functions, where $u^i : \mathbb{R}^m \to \mathbb{R}$, $u^i(a_1, \ldots, a_m) = a_i$. Thenceforth, when coordinate functions are involved in a calculation it is always considered locally. We now can define a smooth function on $\mathcal{M}$.

Definition 3. $f : \mathcal{M} \to \mathbb{R}$ function is **smooth**, if for an arbitrary $(U, x) \in \mathcal{A}$ map the $f \circ x^{-1} : x(U) \subset \mathbb{R}^m \to \mathbb{R}$ is a C^∞ function.

Remark. Denote $\mathcal{F}(\mathcal{M})$ the set of smooth functions defined on $\mathcal{M}$. If we consider addition and multiplication of functions as usual, then $\mathcal{F}$ is a unitary ring. Furthermore, if a multiplication with a scalar is specified then $\mathcal{F}$ is a vector field over $\mathbb{R}$.

Definition 4. At a fixed $p \in \mathcal{M}$ we can define **tangent vectors** $v : \mathcal{F}(\mathcal{M}) \to \mathbb{R}$ to $\mathcal{M}$ in p, which is a real-valued linear function satisfying the following equation for all $f, g \in \mathcal{F}$

$$v(fg) = v(f)g(p) + f(p)v(g)$$

The number $v(f)$ can be interpreted as the directional derivative of f along v at a given p.

Remark. At each $p \in \mathcal{M}$ let $T_p\mathcal{M}$ be the set of tangent vectors to $\mathcal{M}$. Since $T_p\mathcal{M}$ is a vector space over $\mathbb{R}$ with functional addition and scalar multiplication, $T_p\mathcal{M}$ is called the **tangent space to $\mathcal{M}$ at p**.

Since we often consider a local coordinate-system (U, x), it is essential to mention the connection between $T_p\mathcal{M}$ and T_pU.

Definition 5. Assume that $\mathcal{M}$ is a differentiable manifold with the atlas $\mathcal{A}$ and U is an open subset of $\mathcal{M}$. Equip U with the topology induced from that of $\mathcal{M}$, then U is also an m-dimensional differentiable manifold with the atlas $\mathcal{A}_U = \{(V, x) \in \mathcal{A} | V \subset U\}$. In this case U is called an **open submanifold** of $\mathcal{M}$.

Now, fix $p \in \mathcal{M}$, $v \in T_p\mathcal{M}$, and regard a function $g \in \mathcal{F}(U)$. We can construct a function $\tilde{g} \in \mathcal{M}$ such that $\tilde{g}|_V = g|_V$, where $V \subset U$ is an open neighborhood of p. Consider the mapping $\tilde{v} : \mathcal{F}(U) \to \mathbb{R}$, $\tilde{v}(g) = v(\tilde{g})$, where $\tilde{g}$ is defined above. It can be seen that $\tilde{v}$ is a tangent vector to the manifold U at p. Furthermore, the mapping $\phi : T_p\mathcal{M} \to T_U$, where $\phi(v) = \tilde{v}$ defines a linear isomorphism between $T_p\mathcal{M}$ and T_pU, therefore in the following we will identify the two tangent spaces.

Adapting the partial derivatives, for a point $p \in U$ let's define the mapping $\frac{\partial}{\partial x^i}(p) : \mathcal{F}(\mathcal{M}) \to \mathbb{R}$ with

$$\frac{\partial}{\partial x^i}(p)(f) = \partial_i(f \circ x^{-1})(x(p)). \tag{4.1}$$

It is verifiable that $\frac{\partial}{\partial x^i}(p)$ is a tangent vector to $\mathcal{M}$, and vectors $\frac{\partial}{\partial x^1}(p) \dots \frac{\partial}{\partial x^m}(p)$

form a basis for the tangent space $T_p\mathcal{M}$ and for all $v \in T_p\mathcal{M}$

$$v = \sum_{i=1}^{m} v(x^i) \frac{\partial}{\partial x^i}(p). \tag{4.2}$$

Definition 6. The tangent vectors $\frac{\partial}{\partial x^i}(p)$ $(i = 1, \ldots, m)$ are called the **coordinate basis vectors** of (U, x).

Now, after we discussed essential definitions related to smooth functions on $\mathcal{M}$, let's move on to smooth vector fields. First, denote $T\mathcal{M}$ the set of tangent vectors to $\mathcal{M}$.

Definition 7. $X : \mathcal{M} \to T\mathcal{M}$ is said to be a **vector field on** $\mathcal{M}$ if $X(p) \in T_p\mathcal{M}$ for all $p \in \mathcal{M}$, namely a vector field defines a tangent vector at each point.

Regarding an X vector field on $\mathcal{M}$ and $f \in \mathcal{F}(\mathcal{M})$, an $Xf : \mathcal{M} \to \mathbb{R}$ mapping can be introduced as

$$Xf(p) = X(p)(f). \tag{4.3}$$

Definition 8. An X vector field on $\mathcal{M}$ is smooth if the function Xf is differentiable for all $f \in \mathcal{F}(\mathcal{M})$.

Remark. Denote $\mathfrak{X}(\mathcal{M})$ the set of smooth vector fields on $\mathcal{M}$, it is an infinite-dimensional vector space with addition and scalar multiplication. Moreover, defining the product of $X \in \mathfrak{X}(\mathcal{M})$ and $f \in (\mathcal{M})$ as $fX : (\mathcal{M}) \to T(\mathcal{M})$, where $fX(p) = f(p)X(p)$ for all $p \in (\mathcal{M})$, $\mathfrak{X}(\mathcal{M})$ is a module over the unitary ring $\mathcal{F}(\mathcal{M})$.

Definition 9. Suppose (U, x) is a map of $\mathcal{M}$ and let $\frac{\partial}{\partial x^i} : U \to TU$ be a vector field that assigns $\frac{\partial}{\partial x^i}(p)$ to $p \in U$. Then $\frac{\partial}{\partial x^i}$ is called the ***i*th coordinate basis vector field** of (U, x).

Remark. At each $p \in U$ the value on a vector field X can be expressed by using Equation 4.2 as follows

$$X(p) = \sum_{i=1}^{m} \eta^i(p) \cdot \frac{\partial}{\partial x^i}(p), \tag{4.4}$$

where $\eta^i(p) = X(p)(x^i)$. The functions $\eta^i : U \to \mathbb{R}$ $(i = 1, \ldots, m)$ are called the coordinate function of X respecting the map (U, x).
Accordingly, any vector field can be written as

$$X = \sum_{i=1}^{m} \eta^i \cdot \frac{\partial}{\partial x^i} \quad \text{on } U. \tag{4.5}$$

Remark. If the consider the vector field $\frac{\partial}{\partial x^i}$ and a function f, then the notation of Equation 4.3 changes from $\frac{\partial}{\partial x^i} f$ to $\frac{\partial f}{\partial x^i}$.

Now, continue the discussion with the differential of a function.

Definition 10. Let $f \in \mathcal{F}(\mathcal{M})$ and $p \in \mathcal{M}$, then the mapping $df(p) : T_p\mathcal{M} \to \mathbb{R}$, defined as $df(p)(v) = v(f)$ $(v \in T_p\mathcal{M})$, is the **differential of the function** f **at** p.

Definition 11. The **differential** of a function $f \in \mathcal{F}(\mathcal{M})$ is $df : \mathfrak{X}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$, if for every $X \in \mathfrak{X}(\mathcal{M})$ $df(X) = Xf$.

Note, that $T_p^\star\mathcal{M} = \{f : T_p\mathcal{M} \to \mathbb{R} | f \text{ linear}\}$ is the (algebraic) dual space of $T_p\mathcal{M}$, its elements are called linear functionals, linear forms or covectors.
Obviously, $df(p)$ is a linear form. Now let's consider the coordiante functions of the local coordinate-system, and take their differentials at $p \in U$: $dx^i(p)$ $(i = 1, \ldots, m)$.

Statement 2. The vectors $dx^i(p) \in T_p^\star\mathcal{M}$ $(i = 1, \ldots, m)$ provide the dual basis to $\frac{\partial}{\partial x^i}(p)$.

Proof. At each point $p \in U$ $dx^i(\frac{\partial}{\partial x^j})(p) = \delta_{ij}$, since by Definition 11

$$dx^i(\frac{\partial}{\partial x^j}) = \frac{\partial x^i}{\partial x^j}.$$

Evaluating the equation above at p and using Equation 4.1 we get

$$\begin{aligned} dx^i(\frac{\partial}{\partial x^j})(p) = \frac{\partial x^i}{\partial x^j}(p) = \frac{\partial}{\partial x^j}(p)(x^i) = \partial_j(x^i \circ x^{-1})(x(p)) = \\ = \partial_j(u^i \circ x \circ x^{-1})(x(p)) = \partial_j u^i(x(p)) = \delta_{ij}, \end{aligned} \tag{4.6}$$

by using the associative property of composition. □

Accordingly, $dx^i(p)$ $(i = 1, \ldots, m)$ form a basis in $T_p^\star\mathcal{M}$.

Remark. An arbitrary $\Theta : \mathfrak{X}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$ can be expressed as

$$\Theta = \sum_{i=1}^{m} \Theta(\frac{\partial}{\partial x^i}) dx^i$$

Therefore the differential of a function f has the form

$$df = \sum_{i=1}^{m} \frac{\partial f}{\partial x^i} dx^i, \tag{4.7}$$

since $df(\frac{\partial}{\partial x^i}) = \frac{\partial f}{\partial x^i}$.

The following definitions are about tensor fields and their product, which are essential for the definition of the Riemannian metric.

Definition 12. On a manifold $\mathcal{M}$, the mapping $Q : \mathfrak{X}(\mathcal{M})^r \to \mathcal{F}(\mathcal{M})$ is a **type** $(0, r)$ **tensor field** ($r \in \mathbb{N}$), if it is r-linear over the ring $\mathcal{F}(\mathcal{M})$, i. e. for all $f \in \mathcal{F}(\mathcal{M})$ function and $X_1, \ldots, X_i, X_i', \ldots, X_r$ $(1 \leq i \leq r)$ vector fields

$$\begin{aligned} Q(X_1, \ldots, fX_i, \ldots, X_r) &= f \cdot Q(X_1, \ldots, X_i, \ldots, X_r), \\ Q(X_1, \ldots, X_i + X_i', \ldots, X_r) &= Q(X_1, \ldots, X_i, \ldots, X_r) + Q(X_1, \ldots, X_i', \ldots, X_r). \end{aligned} \tag{4.8}$$

Remark. A type $(0, r)$ tensor field is also called **covariant r-tensor field**.

Furthermore, denote $\mathcal{T}_r^0(\mathcal{M})$ the set of type $(0, r)$ tensor fields. $\mathcal{T}_r^0(\mathcal{M})$ is a module over $\mathcal{F}(\mathcal{M})$ commutative ring.

The next step is defining the value of Q type $(0, r)$ tensor field at a point $p \in \mathcal{M}$.

Definition 13. A Q type $(0, r)$ tensor field defines a $Q_p : (T_p\mathcal{M})^r \to \mathbb{R}$ mapping at each p which is interpreted as follows. Consider r vector fields $X_1, \ldots, X_r \in \mathfrak{X}(\mathcal{M})$ for r particular tangent vectors $v_1, \ldots, v_r \in T_p\mathcal{M}$ such that $X_k(p) = v_k$ $(k = 1, \ldots, r)$ are satisfied. Then $Q_p(v_1, \ldots, v_r) = Q(X_1, \ldots, X_r)(p)$.

Remark. The value of a tensor field Q does not depend on the choice of the representing vector fields.

Remark. Q_p is an r-linear mapping on the vector space $T_p\mathcal{M}$ respecting $\mathbb{R}$.

Remark. The differential of a function f $df : \mathfrak{X}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$ is a covariant 1-tensor field.

Definition 14. Let $Q_1 \in \mathcal{T}_r^0(\mathcal{M})$ and $Q_2 \in \mathcal{T}_s^0(\mathcal{M})$ be covariant tensor fields. The product of the tensor fields is $Q_1 \otimes Q_2 : \mathfrak{X}(\mathcal{M})^{r+s} \to \mathcal{F}(\mathcal{M})$, where

$$Q_1 \otimes Q_2(X_1, \ldots, X_{r+s}) = Q_1(X_1, \ldots, X_r) \cdot Q_2(X_{r+1}, \ldots, X_{r+s}).$$

$Q_1 \otimes Q_2$ is (r+s)-linear over $\mathcal{F}(\mathcal{M})$, therefore $Q_1 \otimes Q_2 \in \mathcal{T}_{r+s}^0(\mathcal{M})$, thus the type (0, r+s) tensor field is the **tensor product** of Q_1 and Q_2.

We now discuss the Riemannian manifold and generalized gradient on it, which is essential to Riemannian gradient descent.

Definition 15. Let $\mathcal{M}$ be a differentiable manifold, and $g : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathcal{F}(\mathcal{M})$ is a symmetric covariant 2-tensor field on it. Furthermore, on each $T_p\mathcal{M}$ $(p \in \mathcal{M})$ g defines a $g_p : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$ symmetric bilinear form. If g_p is positive definit for all $p \in \mathcal{M}$, then g is a **Riemannian metric** on the $\mathcal{M}$ manifold, and the $(\mathcal{M}, g)$ tuple is called **Riemannian manifold**.

Remark. The metric tensor g defines an inner product on each tangent space: $\langle v, w \rangle_g = g_p(v, w)$. Moreover, use the notation $g(X, Y) = \langle X, Y \rangle$.

Consider a map (U, x) of $\mathcal{M}$.

Definition 16. The **component functions of the metric tensor** g are the smooth functions $g_{ij} : U \to \mathbb{R}$ $(i, j = 1, \ldots, m)$, where $g_{ij}(p) = g_p(\frac{\partial}{\partial x^i}(p), \frac{\partial}{\partial x^j}(p))$ for all $p \in U$.

Remark.

$$g_{ij} = g(\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}) \tag{4.9}$$

Remark. For an arbitrary $p \in U$ the matrix representation of g_p in the basis $\{\frac{\partial}{\partial x^1}(p), \ldots, \frac{\partial}{\partial x^m}(p)\}$ is an $m \times m$ matrix $G(p)$, whose elements are $G(p)_{ij} = g_{ij}(p)$. Since $g_{ij} = g_{ji}$, $G(p)$ is symmetric, and because of the positive definitness $det\ G(p) > 0$, so $G(p)$ is invertible. Denote the elements of $G(p)^{-1}$ by $g^{ij}(p)$ $(i, j = 1, \ldots, m)$, and let $g^{ij} : U \to \mathbb{R}$ be a function which assigns $g^{ij}(p)$ to $p \in U$. Thus one can define $G, G^{-1} : U \to End(\mathbb{R}^m)$.

Remark.

$$g|_U = \sum_{i=1}^{m} \sum_{j=1}^{m} g_{ij} dx^i \otimes dx^j \tag{4.10}$$

4.1 Riemannian gradient

Now let $\mathcal{M}$ be a Riemannian manifold and $f : \mathcal{M} \to \mathbb{R}$ be a real valued function on $\mathcal{M}$. With this notation the optimization problem is

$$\min_{p \in \mathcal{M}} f(p).$$

Adapting the gradient descent method the gradient of a function should be defined. The definition is motivated by the one on the Euclidean space.

Definition 17. The **gradient** of a function $f \in \mathcal{F}(\mathcal{M})$ is the unique vector field $grad\ f \in \mathfrak{X}(\mathcal{M})$, such that for all $X \in \mathfrak{X}(\mathcal{M})$

$$\langle grad\ f, X\rangle = df(X). \tag{4.11}$$

Remark. At a point p g defines the inner product g_p, thus 4.11 modifies to $\langle grad\ f(p), v\rangle_p = df(X)(p) = Xf(p) = X(p)(f) = v(f)$ using $X(p) = v$, therefore

$$\langle grad\ f(p), v\rangle_p = v(f). \tag{4.12}$$

The equation above can be seen as at each point the inner product of the gradient and a tangent vector v is the directional derivative along v, as it is expected.

We now express the gradient of f in term of local coordinates according to Jäckel, 2017, namely in the basis formed by the vector fields $\frac{\partial}{\partial x^1}, \ldots, \frac{\partial}{\partial x^m}$ respecting the map (U, x). Thus there exist coefficients $f^j : U \to \mathbb{R}$ $(j = 1, \ldots, m)$ such that

$$grad\ f = \sum_{j=1}^{m} f^j \frac{\partial}{\partial x^j} \tag{4.13}$$

Calculating the coefficients, first use Definition 11 on the vector field $\frac{\partial}{\partial x^i}$ and Equation 4.11 in connection of the gradient

$$\frac{\partial f}{\partial x^i} = df(\frac{\partial}{\partial x^i}) = \langle grad\ f, \frac{\partial}{\partial x^i}\rangle = g(grad\ f, \frac{\partial}{\partial x^i}) = g(\sum_{k=1}^{m} f^k \frac{\partial}{\partial x^k}, \frac{\partial}{\partial x^i}) =$$

beacuse g is bilinear over $\mathcal{F}(\mathcal{M})$ and from Equation 4.9

$$= \sum_{k=1}^{m} f^k g(\frac{\partial}{\partial x^k}, \frac{\partial}{\partial x^i}) = \sum_{k=1}^{m} f^k g_{ki}.$$

For a fixed j multiply the equation above by g^{ij} and sum over the index i

$$\sum_{i=1}^{m} \frac{\partial f}{\partial x^i} g^{ij} = \sum_{i=1}^{m}\sum_{k=1}^{m} f^k g_{ki} g^{ij} = \sum_{k=1}^{m} f^k \sum_{i=1}^{m} g_{ki} g^{ij} = \sum_{k=1}^{m} f^k \delta_{kj} = f^j,$$

where δ_{kj} is a function which assigns the Kronecker-delta to all $p \in U$. The result of

the derivation above is the following if we substitute back to Equation 4.13

$$grad\ f = \sum_{i=1}^{m}\sum_{j=1}^{m} g^{ij}\frac{\partial f}{\partial x^i}\frac{\partial}{\partial x^j}, \tag{4.14}$$

and at a point $p \in U$

$$grad\ f(p) = \sum_{i=1}^{m}\sum_{j=1}^{m} g^{ij}(p)\frac{\partial f}{\partial x^i}(p)\frac{\partial}{\partial x^j}(p) \in T_p\mathcal{M}. \tag{4.15}$$

Instead of the summations one can use vectors and matrices regarding the coordiante basis vector fields $\frac{\partial}{\partial x^i}$ $(i = 1, \ldots, m)$. The differential of a function f can be repesented with a row vector d containing the partial derivatives of f according to Equation 4.7: $d = (\frac{\partial f}{\partial x^1}, \ldots, \frac{\partial f}{\partial x^m})$. Then the gradient of f can be expressed with a column vector h, such that

$$h = G^{-1}d^\top. \tag{4.16}$$

Statement 3. Let f be a real valued function of $\mathcal{M}$ and $p \in \mathcal{M}$ is a fixed point. Then among all unit vector $v \in T_p\mathcal{M}$, the gradient $grad\ f(p)$ is the direction in which the directional derivative $v(f)$ is the greatest. Moreover, $\|grad\ f(p)\|_g$ equals to the value of the directional derivative in that direction.

Proof. From Equation 4.12 and because of $\|v\|_g = 1$

$$v(f) = \langle grad\ f(p), v\rangle_p = \|grad\ f(p)\|_g\|v\|_g\cos\theta = \|grad\ f(p)\|_g\cos\theta.$$

This expression is maximized when $\cos\theta = 1$, therefore the vector v' which maximizes the directional derivative has the same direction as $grad\ f(p)$, and the directional derivative in that direction equals to $\|grad\ f(p)\|_g$. □

Obtaining the result above, the gradient that is defined on Riemannian manifolds indeed has the same essential property as the Euclidean gradient has in connection with the gradient descent algorithm, namely it is the direction in which the directional derivative is the greatest. Regarding the Euclidean gradient descent after calculating the gradient, the algorithm moves in the opposite direction along a straight line obtaining a new point. Our next interest is to define moving "straight" from a point in a direction on a manifold.

4.2 Exponential map and retraction

Beforehand, on a manifold we were able to define the derivation of a smooth function along a tangent vector, hence the motivation behind covariant derivative is defining the derivative of a vector field along a tangent vector, but in the following, only its general form will be used, namely the derivative of a vector field with respect to another vector field.

Definition 18. The mapping $\nabla : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$ is a **covariant derivation** if for all $X, Y, Z \in \mathfrak{X}(\mathcal{M})$ and $f \in \mathcal{F}(\mathcal{M})$

(1) $\nabla(X + Y, Z) = \nabla(X, Z) + \nabla(Y, Z)$,

(2) $\nabla(fX, Y) = f \cdot \nabla(X, Y)$,

(3) $\nabla(X, Y + Z) = \nabla(X, Y) + \nabla(X, Z)$,

(4) $\nabla(X, fY) = f \cdot \nabla(X, Y) + (Xf) \cdot Y$.

Remark. Because of (4) ∇ is not a type $(1, 2)$ tensor field.

Remark. $\nabla(X, Y)$ vector field is also denoted by $\nabla_X Y$.

Now let (U, x) be a map of $\mathcal{M}$ and denote its coordiante basis vector fields with $X_i = \frac{\partial}{\partial x^i}$ $(i = 1, \dots, m)$. Then the vector field $\nabla_{X_i} X_j \in \mathfrak{X}(\mathcal{M})$ can be expressed in the basis of coordiante basis vector fields in the form $\nabla_{X_i} X_j = \sum_{k=1}^{m} \Gamma_{ij}^k \cdot X_k$ with appropriate functions $\Gamma_{ij}^k \in \mathcal{F}(\mathcal{M})$ $(i, j, k = 1, \dots, m)$.

Definition 19. The differentiable functions $\Gamma_{ij}^k : U \to \mathbb{R}$ are called the **Christoffel symbols** of the covariant derivative ∇ respecting the map (U, x).

The previous two definitions can be considered on a general manifold, but on Riemannian manifolds we consider a unique covariant derivative having certain properties which is called the Levi-Civita connection. Now, the Christoffel symbols of the Levi-Civita connection can be written as the following using the metric tensor g and its inverse.

$$\Gamma_{ij}^k = \frac{1}{2} \sum_{l=1}^{m} g^{kl} \left(\frac{\partial g_{jl}}{\partial x^i} + \frac{\partial g_{li}}{\partial x^j} - \frac{\partial g_{ij}}{\partial x^k} \right) \tag{4.17}$$

for all $i, j, k = 1, \dots, m$. From now on, we consider the covariant derivative and Christoffel symbols above, but the following definitions hold for general covariant derivatives as well.

As the aim is to define moving "straight" on a manifold, let's regard the curves of the manifold.

Definition 20. The $\gamma : I \to \mathcal{M}$ smooth mapping is a **smooth curve** of the differentiable manifold $\mathcal{M}$.

Regarding the 1-dimensional $\mathbb{R}$ Euclidean space, it is a differentiable manifold and respecting a map $(\mathbb{R}, id)$ at a point $t \in \mathbb{R}$ denote $\frac{\partial}{\partial u}(t)$ the coordiante basis vector. If $f : \mathbb{R} \to \mathbb{R}$ is a smooth function, then $\frac{\partial}{\partial u}(t)(f) = f'(t)$.

If $\mathcal{M}$ and $\mathcal{N}$ are differentiable manifolds and $\mu : \mathcal{M} \to \mathcal{N}$ is a mapping between them, then to a tangent vector $v \in T_p\mathcal{M}$ assign the mapping $v_\mu : \mathcal{F}(\mathcal{M}) \to \mathbb{R}$, where $v_\mu(f) = v(f \circ \mu)$ for all $f \in \mathcal{F}(\mathcal{M})$. The mapping v_μ is a tangent vector on the manifold $\mathcal{N}$ at the point $\mu(p)$.

Definition 21. The mapping $T_p\mu : T_p\mathcal{M} \to T_{\mu(p)}\mathcal{N}$ is called the **tangent map or differential** of μ at $p \in \mathcal{M}$ if $T_p\mu(v) = v_\mu$ for all $v \in T_p\mathcal{M}$. Another notation for the differential of μ at p is $D\mu(p)$.

Definition 22. The **tangent vector of a curve** $\gamma : I \to \mathcal{M}$ at point $t \in I$ is the tangent vector $T_t\gamma(\frac{\partial}{\partial u}(t))$ at the point $\gamma(t)$. We denote the tangent vector of γ at t with $\dot{\gamma}(t)$.

In the definition above the tangent map of γ is used between the manifolds $\mathbb{R}$ and $\mathcal{M}$.

Statement 4. The tangent vector of γ at the point $t \in I$ can be expressed as

$$\dot{\gamma}(t) = \sum_{i=1}^{m} (x^i \circ \gamma)'(t) \cdot \frac{\partial}{\partial x^i}(\gamma(t)). \tag{4.18}$$

Definition 23. Suppose $\gamma : I \to \mathcal{M}$ is a smooth curve. The **vector field along** γ is a mapping $Z : I \to T\mathcal{M}$, where $Z(t) \in T_{\gamma(t)}\mathcal{M}$ for all $t \in I$.

Statement 5. $Z(t)$ can be written as

$$Z(t) = \sum_{i=1}^{m} \zeta^i(t) \cdot (\frac{\partial}{\partial x^i} \circ \gamma)(t),$$

where $\zeta^i : I \to \mathbb{R}$ $(i = 1, \ldots, m)$ are real functions.

Definition 24. Z smooth if the functions ζ^i $(i = 1, \ldots, m)$ are differentiable.

Remark. The vector field $\dot{\gamma} : I \to T\mathcal{M}$, where $\dot{\gamma}(t)$ is the tangent vector of γ at $t \in I$, is a smooth vector field along γ.

Consider a map (U, x) at the point $\gamma(t)$. Let $J \subset I$ be a real interval, where $\gamma(J) \subset U$ and let $Z|_J$ be a vector field along $\gamma|_J$. Then $Z|_J = \sum_{i=1}^{m} \zeta^i \cdot (\frac{\partial}{\partial x^i} \circ \gamma|_J)$, where $\zeta^i \in \mathcal{F}(J)$. Furthermore, let $\gamma^i = x^i \circ \gamma|_J$ be a real function.

Definition 25. The covariant derivative of the vector field Z along γ at $t \in J$ is the vector

$$Z'(t) = \sum_{k=1}^{m} \{(\zeta^k)'(t) + \sum_{i=1}^{m} \sum_{j=1}^{m} \Gamma_{ij}^k(\gamma(t)) \cdot (\gamma^i)'(t) \cdot \zeta^j(t)\} \cdot \frac{\partial}{\partial x^k}(\gamma(t)). \tag{4.19}$$

Remark. The vector $Z'(t) \in T_{\gamma(t)}\mathcal{M}$ is not depend on the choice of the map (U, x) at $\gamma(t)$.

Definition 26. On a Riemannian manifold $(\mathcal{M}, g)$ consider a map (U, x) and a smooth curve $\gamma : I \to \mathcal{M}$, where $\gamma(I) \subset U$. γ is a **geodesic** if $(\dot{\gamma})'(t) = 0$ for all $t \in I$.

From Equation 4.18 and Equation 4.19, the following statement can be derived.

Statement 6. The smooth curve γ is geodesic if and only if $y_k = x^k \circ \gamma$ $(k = 1, \ldots, m)$ satisfy

$$y_k''(t) + \sum_{i=1}^{m} \sum_{j=1}^{m} \Gamma_{ij}^k(\gamma(t)) \cdot y_i'(t) \cdot y_j'(t) = 0. \tag{4.20}$$

system of second-order ordinary differential equations.

Definition 27. Suppose that $[a, b] \subset I$ is a closed subinterval, then the **length** of the curve segment $\gamma|_{[a,b]}$ is

$$l(\gamma|_{[a,b]}) = \int_a^b \|\dot{\gamma}(t)\|_g$$

Using the length of curve segments we can define the Riemannian distance of $p, q \in \mathcal{M}$.

Definition 28. The **Riemannian distance** of $p, q \in \mathcal{M}$ is

$$d_g(p, q) = \inf\{l(\gamma) | \gamma : [a, b] \to \mathcal{M}, \gamma(a) = p, \gamma(b) = q\}.$$

Remark. It can be shown that the Riemannian distance is indeed a metric, using the fact that $\mathcal{M}$ is Hausdorff.

The γ, such that $l(\gamma) = d_g(p, q)$, is a **length-minimizing curve**. It can be shown that the length-minimizing curves are geodesics and that geodesics are locally length-minimizing (Carmo, 1992).

We now can see that between two points of the manifold there exists a special curve that are "straight" between the two points, this curve is a geodesic. Since geodesics are curves such that the covariant derivative of the velocity vector (tangent vector) along the curve itself is 0 at any time, intuitively, the velocity vector's direction and length is constant along γ, therefore geodesics are indeed the generalization in a certain way of straight lines.

Definition 29. $\gamma : I \to \mathcal{M}$ is a **maximal** geodesic, if there exists no interval J and $\sigma : J \to \mathcal{M}$ geodesic such that $I \subset J$, $I \neq J$, $\sigma|_I = \gamma$.

To compute a geodesic staring from a specific $p \in \mathcal{M}$ in direction $v \in T_p\mathcal{M}$, we need to solve the system of second-order ODE 4.20 with initial conditions

$$\gamma(0) = p, \dot{\gamma}(0) = v.$$

The following Statement illustrates the uniqueness of such maximal geodesic.

Statement 7. If $p \in \mathcal{M}$ and $v \in T_p\mathcal{M}$ are given, then there exists a unique maximal geodesic $\gamma_v : I \to \mathcal{M}$ on the Riemannian manifold $(\mathcal{M}, g)$, such that $\gamma(0) = p$ and $\dot{\gamma}(0) = v$.

Remark. Importantly, observe that if $\gamma_v(t)$ is a maximal geodesic on interval I satisfying the initial conditions $\gamma(0) = p$, $\dot{\gamma}(0) = v$ then $\gamma_v(ct)$ $(c \neq 0)$ is also a maximal geodesic on $\frac{I}{c}$ satisfying the system 4.20 with initial conditions $\gamma(0) = p$, $\dot{\gamma}(0) = cv$. Therefore, from the statement above

$$\gamma_{cv}(t) = \gamma_v(ct) \quad ct \in I. \tag{4.21}$$

Using the fact above, we are able to define the exponential map.

Definition 30. For each $p \in \mathcal{M}$ let

$$\mathcal{D}(p) := \{v \in T_p\mathcal{M} : \gamma_v(1) \text{ is defined}\},$$

where γ_v is the unique maximal geodesic satisfies the initial conditions $\gamma(0) = p$, $\dot{\gamma}(0) = v$. $\mathcal{D}(p)$ is called the **domain of the exponential map**.

Definition 31. The **exponential map** is the map $\exp_p : \mathcal{D}(p) \to \mathcal{M}$ given by

$$\exp_p(v) = \gamma_v(1).$$

The exponential map can be interpreted as we move along a geodesic (generalization of a "straight" line on the manifold) starting from a point p in the direction of the velocity vector v.

Computing the exponential map can be challenging since to evaluate the geodesic, a system of non-linear differential equations needs to be solved. Therefore, we would like to introduce an approximation of the exponential map, which is a computationally more efficient alternative. This method is the retraction, but before its discussion, we review some claims related to a vector space and its tangent spaces.

Statement 8. An m-dimensional vector space $\mathcal{V}$ over $\mathbb{R}$ is an m-dimesional differentiable manifold.

Proof sketch. Let $(e_i)_{i=1,\dots,m}$ be a basis of $\mathcal{V}$. If $p = \sum_{i=1}^{m} p^i e_i$, then

$$x : \mathcal{V} \to \mathbb{R}^m, \quad p \mapsto \begin{bmatrix} p^1 \\ p^2 \\ \vdots \\ p^m \end{bmatrix}$$

is a global chart of $\mathcal{V}$, and $(\mathcal{V}, x)$ is a global map, which defines a maximal smooth atlas. Furthermore, x is an isomorphism between $\mathcal{V}$ and $\mathbb{R}^m$, therefore in Definition 1 all the conditions are satisfied.

Statement 9. If $\mathcal{V}$ is a vector space, then

$$\mathcal{V} \simeq T_p\mathcal{V} \tag{4.22}$$

for an arbitrary $p \in \mathcal{V}$.

Proof sketch. Let $x^1, \dots, x^m$ be the coordiante functions of x; $p \in \mathcal{V}$, and consider the basis $(e_i)_{i=1,\dots,m}$ of $\mathcal{V}$ and the basis $(\frac{\partial}{\partial x^i}(p))_{i=1,\dots,m}$ of $T_p\mathcal{V}$. Then regard the mapping

$\phi : \mathcal{V} \to T_p\mathcal{V}$ given by $v \mapsto w$, where $v = \sum_{i=1}^m v^i e_i$ and $w = \sum_{i=1}^m v^i \frac{\partial}{\partial x^i}(p)$. Then ϕ is an isomorphism.

Therefore, we can identify $\mathcal{V}$ with its tangent space at a point.

The remainder of the section is about the approximation of the exponential map, namely about retractions.

Definition 32. Let $R : T\mathcal{M} \to \mathcal{M}$ be a smooth mapping, and denote R_p the restriction of R to $T_p\mathcal{M}$ for a $p \in \mathcal{M}$. R and R_p is a **retraction** if
(1) $R_p(0_p) = p$, where 0_p is the zero element of $T_p\mathcal{M}$
(2) $DR_p(0_p) = \mathrm{id}_{T_p\mathcal{M}}$, where D denotes the differential of R_p according to Definition 21 and $\mathrm{id}_{T_p\mathcal{M}}$ denotes the identity mapping on $T_p\mathcal{M}$. Furthermore, we use the identification $T_{0_p}T_p\mathcal{M} \simeq T_p\mathcal{M}$, since $T_p\mathcal{M}$ is a vector space.

In terms of the Riemannian gradient descent the most important property of retraction is that it is a first order approximation of the exponential map, namely

$$d_g(\exp_p(tv), R_p(tv)) = \mathcal{O}(t^2), \quad t \in \mathbb{R}.$$

Before introducing a retraction that is well defined and computationally efficient, we review the embedded submanifolds.

Definition 33. A P k-dimensional differentiable manifold is an **embedded submanifold** of $\mathcal{M}$ if
(1) P is a subset of $\mathcal{M}$ and its topology is the same as the subspace topology inherited from $\mathcal{M}$,
(2) and the tangent map of $\iota : P \hookrightarrow \mathcal{M}$ canonical injection $T_p\iota$ is injective for all $p \in \mathcal{M}$.

Remark. If $k = m$ then such P is an open submanifold of $\mathcal{M}$, and $T_pP \simeq T_p\mathcal{M}$ as discussed above. While $k < m$, T_pP can be identified with a k-dimension subspace of $T_p\mathcal{M}$ by the mapping $T_p\iota$.

We now use the assumption that the manifold $\mathcal{M}$ is an embedded submanifold of a vector space $\mathcal{E}$. Abusing notation, define the sum of $p + v$, where $p \in \mathcal{M}$ can be viewed as a point on $\mathcal{E}$, and $v \in T_p\mathcal{M}$ viewed as an element of $T_p\mathcal{E} \simeq \mathcal{E}$. Therefore in the vector space $\mathcal{E}$ we are able to do the addition $p + v$.
Using the facts above, a particular retraction can be defined for a given $p \in \mathcal{M}$ and

$v \in T_p\mathcal{M}$ which is described intuitively as:
(1) moving along v to get the point $p + v$ in $\mathcal{E}$,
(2) projecting the point $p + v$ back to $\mathcal{M}$.
In the following, we state the retraction above formally.

Statement 10. Let $\mathcal{N}$ be a manifold such that $\dim(\mathcal{M}) + \dim(\mathcal{N}) = \dim(\mathcal{E})$. Additionally assume that there exists a diffeomorphism (a mapping that smooth, bijective, and its inverse is differentiable) $\phi : \mathcal{M} \times \mathcal{N} \to \mathcal{E}'$, where $\mathcal{E}'$ is an open submanifold of $\mathcal{E}$ and there is an element $i \in \mathcal{N}$ such that $\phi(p, i) = p$, $\forall p \in \mathcal{M}$.
Under these assumptions the mapping

$$R_p(v) := \pi_1(\phi^{-1}(p + v)),$$

where $\pi_1 : \mathcal{M} \times \mathcal{N} \to \mathcal{M}$ such that $\pi(p, q) = p$, defines a retraction on $\mathcal{M}$.

Example. Let $\mathcal{M} = S^{n-1}$ be a unit sphere embedded in $\mathcal{E} = \mathbb{R}^n$ and let $\mathcal{N} = \{x \in \mathbb{R} : x > 0\}$. Consider the diffeomorphism $\phi : \mathcal{M} \times \mathcal{N} \to \mathbb{R}^n_*$ such that $\phi(x, r) = rx$. By the Statement above the retraction is $R_p(v) = \frac{x+v}{\|x+v\|}$.

Previously, we defined the gradient of a function on a Riemannian manifold, and discussed how to move from a point in direction of the gradient viewed as a tangent vector. That was the exponential map. Then we introduced a first-order approximation of the exponential map, called retraction. Moreover, because of computational reasons a particular retraction was described which is essential for defining the Riemannian gradient descent algorithm.

Algorithm 1 Riemannian Gradient Descent

```
Initialize p_0 arbitrary
Initialize η ∈ ℝ, η > 0
repeat
    Compute the gradient of f at p_t, i. e. h_t := grad f(p_t)
    Move in direction − h_t, i. e. p_{t+1} = R_{p_t}(−η h_t)
    t = t + 1
until stopping criterion is not satisfied
Return p_t
```

4.3 Natural gradient descent

In this section, we discuss the algorithm natural gradient descent (Amari, 1997). One of the most important special cases of Riemannian gradient descent regarding machine learning is optimizing a function on a statistical manifold $(\mathbb{R}^D, g)$. Specially, $\mathbb{R}^D$ is the parameter space of statistical models $p(\boldsymbol{y}|\boldsymbol{x};\theta)$, where θ is an element of the parameter space. Equip the manifold with a metric tensor g called the Fisher information metric, defined as

$$g_{ij}(\theta) = \mathbb{E}_X \left[\mathbb{E}_{Y|X;\theta} \left[\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x};\theta)}{\partial \theta^i} \frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x};\theta)}{\partial \theta^j} \right] \right]. \tag{4.23}$$

In the definition above, the expectation $\mathbb{E}_{Y|X;\theta}$ is taken over the distribution specified by θ, and $\mathbb{E}_X$ is calculated over the empirical distribution of the training data, however, it is possible to choose any other distribution.

As we discussed in Section 2, in case of a probabilistic model $p(\boldsymbol{y}|\boldsymbol{x};\theta)$ in order to maximize the log-likelihood we wish to minimize the loss function $\mathcal{L}(\theta) = \sum_{i=1}^{N} l(\theta, x_i, y_i)$, where $l(\theta, x, y) = -\log p(y|x;\theta)$. Then metric tensor g can be represented by an $N \times N$ matrix $F(\theta)$ at a point, given by

$$F(\theta) = \mathbb{E}_X[\mathbb{E}_{Y|X;\theta}[\nabla_\theta \mathcal{L}(\theta) \nabla_\theta^\top \mathcal{L}(\theta)]]. \tag{4.24}$$

The function on the manifold which should be minimized is $\mathcal{L}(\theta)$, and its gradient is

$$grad\ \mathcal{L}(\theta) = F^{-1}(\theta) d^\top(\theta) = F^{-1}(\theta) \nabla_\theta \mathcal{L}(\theta) =: \tilde{\nabla}_\theta \mathcal{L}(\theta) \tag{4.25}$$

using Equation 4.16 and the fact that on $\mathbb{R}^D$ the column vector $d^\top$ containing the partial derivatives of $\mathcal{L}$ is denoted by $\nabla_\theta \mathcal{L}$. In this setup the gradient of $\mathcal{L}$ on the particular Riemannian manifold is called the **natural gradient** of $\mathcal{L}$ and it is denoted by $\tilde{\nabla}_\theta \mathcal{L}(\theta)$.

Furthermore, consider the next step of the Riemannian gradient descent, namely moving in the direction of the gradient. The manifold $\mathbb{R}^D$ is embedded in itself and $\mathbb{R}^D \simeq T_p \mathbb{R}^D$, therefore the addition $p + v$ simply can be done in $\mathbb{R}^d$, and further projection is not necessary. This method results in the **natural gradient descent** (NGD), given by

$$\theta(t+1) = \theta(t) - \eta F^{-1}(\theta) \nabla_\theta \mathcal{L}(\theta), \tag{4.26}$$

where F is the Fisher information matrix and η is the step size.

In section 2 gradient flow, the continuous time limit of gradient descent was defined, analogously, **natural gradient flow** (NGF) is given by

$$\dot{\theta} = -F^{-1}(\theta)\nabla_\theta \mathcal{L}(\theta). \tag{4.27}$$

Note, that the Fisher information matrix $F(\theta)$ is not always invertible in this case the parameters of the neural network do not form a Riemannian manifold. However, in the problems we consider there always is a parametrization with respect to which the Fisher information matrix is invertible. Nevertheless, the natural gradient descent algorithm is reasonable and works efficiently even when the space is not a Riemannian manifold as the approach and derivation show in Pascanu and Bengio, 2013.

When the Fisher is not invertible we rather consider NGF as any trajectory θ_t which satisfies

$$F(\theta)\dot{\theta} = -\nabla_\theta \mathcal{L}(\theta). \tag{4.28}$$

Furthermore, it is possible to use a generalized inverse (i. e. A^g s.t. $AA^gA = A$) to choose from the NGF trajectories. One commonly used generalized inverse is the Moore-Penrose pseudoinverse (A^+) whose important property is that if $A \in \mathbb{R}^{m\times n}, m < n$ then it gives the smallest ℓ_2 norm solution among the vectors $\mathbf{x}$ which satisfies $A\mathbf{x} = \mathbf{y}$. Using the Moore-Penrose pseudoinverse the NGF equation is

$$\dot{\theta} = -F^{+}(\theta)\nabla_\theta \mathcal{L}(\theta). \tag{4.29}$$

Remark. In machine learning the loss function is optimized through a feedforward neural network $f(x, \beta)$, whose parameters are β. In this case, the parameter space of neural networks can only form a Riemannian manifold if the neural networks are smooth enough (Daróczy, Aleksziev, and Benczúr, 2019). In this thesis only linear networks are considered, where the corresponding parameter space is indeed a differentiable manifold ($\mathbb{R}^n$).

4.3.1 Parametrization invariance of NGF

The most important property of natural gradient descent is the fact that NGD with infinitesimally small step size (i. e. NGF) is invariant to reparametrization. Below we formally state this property from Amari, 1997.

Statement 11. Let $\mathbf{w}$ and θ be two parameter vectors such that $\dim \mathbf{w} \leq \dim \theta$, and let $\mathcal{P}$ be a mapping such that $\mathbf{w} = \mathcal{P}(\theta)$. Consider a natural gradient flow in θ starting from θ_0. With the assumption that the Jacobian $J = \frac{\partial \mathbf{w}_t}{\partial \theta_t}$ and the Fisher $F(\mathbf{w}_t)$ are both full rank for all t, $\mathbf{w}_t = \mathcal{P}(\theta_t)$ has the exact same trajectory as if the natural gradient flow was considered in $\mathbf{w}$ starting from $\mathbf{w}_0 = \mathcal{P}(\theta_0)$ i. e. $\mathbf{w}_t = \mathcal{P}(\theta_t)$ solves the differential equation $\dot{\mathbf{w}}_t = -F(\mathbf{w}_t)^{-1} \nabla_{\mathbf{w}_t} \mathcal{L}(\mathbf{w}_t)$ for all t.

5 Inductive bias of gradient descent

5.1 Separable classification

The first underdetermined problem considered is separable classification. Regard a binary classification dataset $\{(\mathbf{x}_n, y_n), n = 1, \dots, N\}$ with $\mathbf{x}_n \in \mathbb{R}^D$ and $y_n \in \{-1, 1\}$. Assume that it is separable by a homogeneous linear classifier with a positive margin, i. e. $\exists \boldsymbol{\beta}^*$ s.t. $y_n \mathbf{x}_n^\top \boldsymbol{\beta}^* \geq 1 \ \forall n$, where $\boldsymbol{\beta}^*$ corresponds to the normal vector of the separating hyperplane. The task is to predict the label y for an unknown new data point.

Remark. Note that $X = (\mathbf{x}_1 \cdots \mathbf{x}_N)^\top$ denotes the matrix of the data where each row corresponds to a data point, therefore X is an $N \times D$ matrix. Additionally, the logistic function is denoted by $\phi(a) = \frac{1}{1+e^{-a}}$.

According to Section 2.1 we define an estimating probabilistic model parametrized by $\mathbf{s}$. Specially, consider a Bernoulli distribution with the parameter $\phi(\mathbf{s})$ which indeed depends on $\mathbf{s}$. By modifying $\mathbf{s}$ we wish to maximize the log-likelihood, equivalently minimize the loss defined below using neural networks. Furthermore, we consider $\mathbf{s}$ as the output of the network whose parameters are $\mathbf{w} \in \mathbb{R}^P$: $\mathbf{s} = f(X, \mathbf{w})$, therefore we actually update $\mathbf{w}$. The most simple case is when a certain linear neural network is considered, namely $\mathbf{s} = X\mathbf{w}$, $P = D$. In this case, we use the notation $\boldsymbol{\beta}$ instead of $\mathbf{w}$ to coincide with the problem described above. In the following, we define the probabilistic model using the linear network approach, but the non-linear case is greatly similar. Moreover, $\mathbf{x}$ denotes one data point and $s = \mathbf{x}^\top \boldsymbol{\beta}$, if there are several data points we consider them per coordinate.

$$\begin{aligned} P(y = 1 | X = \mathbf{x}; \boldsymbol{\beta}) &= \phi(s) = \frac{1}{1 + e^{-s}} = \frac{1}{1 + e^{-\mathbf{x}^\top \boldsymbol{\beta}}} = \phi(\mathbf{x}^\top \boldsymbol{\beta}) \\ P(y = -1 | X = \mathbf{x}; \boldsymbol{\beta}) &= 1 - \phi(s) = 1 - \frac{1}{1 + e^{-\mathbf{x}^\top \boldsymbol{\beta}}} = 1 - \phi(\mathbf{x}^\top \boldsymbol{\beta}) = \phi(-\mathbf{x}^\top \boldsymbol{\beta}) \end{aligned} \tag{5.1}$$

The aim is to calculate the maximum log-likelihood estimation of $\boldsymbol{\beta}$. Therefore, as

it was derived before, let the loss function be

$$\ell(\boldsymbol{\beta}) = -\log p(y|\mathbf{x};\boldsymbol{\beta}) = -\log \phi(y\mathbf{x}^\top\boldsymbol{\beta}) = \log(1+e^{-y\mathbf{x}^\top\boldsymbol{\beta}}) = \log(1+e^{-ys}) \quad (5.2)$$

for one data point, called the logistic loss, and for the whole data set

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{n=1}^{N} \log(1+e^{-y_n\mathbf{x}_n^\top\boldsymbol{\beta}}) = \sum_{n=1}^{N} \log(1+e^{-y_n\mathbf{s}_n}). \quad (5.3)$$

In the setting described above, there may be several perfectly separating hyperplanes, therefore it depends on the inductive bias to select one. In the following, we consider two different parametrizations and discuss that in the two cases the unregularized Euclidean gradient descent results in two entirely different solutions.

Remark. The length of $\boldsymbol{\beta}$ will converge to infinity as the loss is being minimized, therefore we only have interest in its direction. It is parallel to the fact that the hyperplane is homogeneous.

Firstly, consider the so-called **direct parametrization** where the separator vector $\boldsymbol{\beta}$ is simply parametrized by its elements. The corresponding neural network is quite primitive, it only has one layer with weights $\boldsymbol{\beta}$, we referred to this previously as a kind of linear neural network. Studying the direct parametrization Soudry et al. (2017) found that $\boldsymbol{\beta}_t$ converges in direction to the ℓ_2 large margin classifier, given by

$$\lim_{t\to\infty} \frac{\boldsymbol{\beta}_t}{|\boldsymbol{\beta}_t|} = \frac{\boldsymbol{\beta}^*_{\ell_2}}{|\boldsymbol{\beta}^*_{\ell_2}|} \text{ where } \boldsymbol{\beta}^*_{\ell_2} = \underset{\boldsymbol{\beta}\in\mathbb{R}^D}{\arg\min} ||\boldsymbol{\beta}||_2 \text{ s.t. } y_n\mathbf{x}_n^\top\boldsymbol{\beta} \geq 1 \quad \forall n.$$

The second parametrization regarded is the **diagonal parametrization**, whose name comes from the corresponding neural network. As illustrated in Figure 5.1, the weights between $i-1$th and ith layer can be written in the form of a diagonal matrix whose entries on the main diagonal are represented by the vector $\mathbf{w_i}$ and there is a weight vector $\mathbf{1}$ between the last and the output layer.

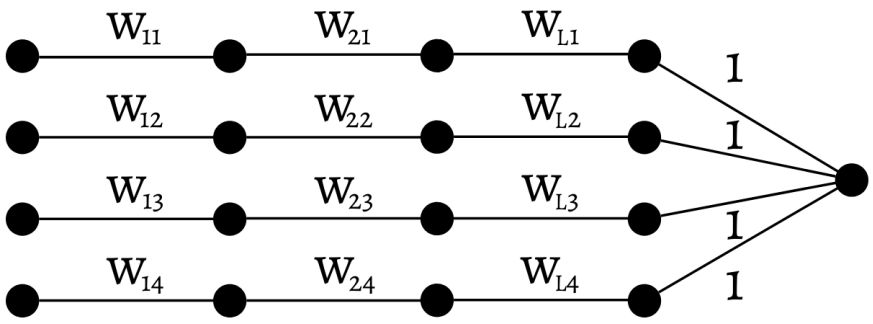

Figure 5.1: Feedforward neural network which corresponds to the diagonal parametrization in separable classification

The weights of the L-layer linear diagonal network above can be rewritten such as $\boldsymbol{\beta} = \mathbf{w_1} \odot \mathbf{w_2} \odot \ldots \odot \mathbf{w_L}$, where $\odot$ denotes elementwise product.

Considering the parametrization above Gunasekar et al., 2018 showed that if the parameters $\mathbf{w_1}, \ldots, \mathbf{w_L}$ are updated through EGD then $\boldsymbol{\beta}_t$ converges in direction to the $\ell_{\frac{2}{L}}$ large margin classifier, defined as

$$\lim_{t\to\infty} \frac{\boldsymbol{\beta}_t}{|\boldsymbol{\beta}_t|} = \frac{\boldsymbol{\beta}^*_{diag}}{|\boldsymbol{\beta}^*_{diag}|} \text{ where } \boldsymbol{\beta}^*_{diag} = \underset{\boldsymbol{\beta}\in\mathbb{R}^D}{\arg\min} ||\boldsymbol{\beta}||_{\frac{2}{L}} \text{ s.t. } y_n \mathbf{x}_n^\top \boldsymbol{\beta} \geq 1 \quad \forall n.$$

Remark. The ℓ_p $(0 < p < 1)$ quasi-norm is defined the same as for $p \geq 1$, i. e. $\|x\|_p = (|x_1|^p + \ldots + |x_D|^p)^{1/p}$.

The classifier $\boldsymbol{\beta}$ which is found in the situation above has a remarkable property: it is significantly sparse (has several 0 elements and just a few non-zero) as Tibshirani, 1996 suggested. Therefore, the inductive bias of the diagonal parametrization is sparsity-seeking.

The sparsity-seeking property is useful for example when to each human attribute we have to assign genes having contribution. In this task the attributes might be affected by only a few genes, therefore sparse solutions are advantageous.

5.2 Matrix completion

The next underdetermined problem we deal with is matrix completion. In this task, we have randomly chosen observations $A_1, A_2, \ldots, A_N$ of an unknown matrix $\boldsymbol{\beta}^* \in \mathbb{R}^{D\times D}$, where $A_i \in \mathbb{R}^{D\times D}$ is a matrix which has 1 in the place of the observed entry and 0 everywhere else. The value of the observed entries are $y_1, y_2, \ldots, y_N$. From this

data, the aim is to recover the matrix $\boldsymbol{\beta}^*$. As several matrices can fit perfectly to the data the problem is underdetermined and it is up to the inductive bias to select one solution.

Remark. We only consider squared matrices but the arguments hold more generally.

According to Section 2 we define an estimating probabilistic model parametrized by the matrix $\boldsymbol{\beta}$. Specially, consider a normal distribution with mean $\langle A, \boldsymbol{\beta} \rangle_F$ and variance σ^2, where $\langle ., . \rangle_F$ denotes the Frobenius product, A is an observation and σ^2 is fixed.

$$p(y|X = A; \boldsymbol{\beta}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y - \langle A, \boldsymbol{\beta} \rangle_F)^2}{2\sigma^2}} \tag{5.4}$$

To determine the maximum log-likelihood estimation of $\boldsymbol{\beta}$ we use the loss function below

$$\ell(\boldsymbol{\beta}) = -\log p(y|A; \boldsymbol{\beta}) = \log(\sigma\sqrt{2\pi}) + \frac{(y - \langle A, \boldsymbol{\beta} \rangle_F)^2}{2\sigma^2} \tag{5.5}$$

for one data point, and for the whole data set

$$\mathcal{L}(\boldsymbol{\beta}) = \sum_{n=1}^{N} \log(\sigma\sqrt{2\pi}) + \frac{(y_n - \langle A_n, \boldsymbol{\beta} \rangle_F)^2}{2\sigma^2}. \tag{5.6}$$

Now we consider two different parametrizations and we discuss that in the two cases the unregularized Euclidean gradient descent results in two entirely different solutions.

Firstly, consider the **direct parametrization** where the matrix $\boldsymbol{\beta}$ is simply parametrized by its elements. The corresponding neural network is primitive, it only has one fully connected layer with weights $\boldsymbol{\beta}$. The Statement below describes the solution EGD finds in this case.

Statement 12. In the setting discussed above EGF finds the "trivial" solution, namely in the place where was observation the entry converges to the observed value, but everywhere else the value won't change from the initialization.

Proof. To find the solution we have to solve the differential equation

$$\dot{\boldsymbol{\beta}} = -\nabla_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}). \tag{5.7}$$

Considering each entry, this defines D^2 independent differential equations. Where

there was no observation the gradient is 0, therefore the differential equation is

$$\dot{\boldsymbol{\beta}}_{ij} = 0, \tag{5.8}$$

hence the solution is a constant with the initialized value

$$\boldsymbol{\beta}_{ij}(t) = \boldsymbol{\beta}_{ij}(0). \tag{5.9}$$

Where we have an observation the differential equation has the form

$$\dot{\boldsymbol{\beta}}_{ij} = \frac{y_n}{\sigma^2} - \frac{\boldsymbol{\beta}_{ij}}{\sigma^2}. \tag{5.10}$$

Its solution is

$$\boldsymbol{\beta}_{ij}(t) = y_n + ce^{-\frac{1}{\sigma^2}t}, \tag{5.11}$$

where c is a constant. Hence

$$\lim_{t\to\infty} \boldsymbol{\beta}_{ij}(t) = y_n. \tag{5.12}$$

□

Another significant parametrization we consider is **matrix factorization**.

Definition 34. The deep matrix factorization of $\boldsymbol{\beta} \in \mathbb{R}^{d,d'}$, with hidden dimensions $d_1, d_2, \ldots, d_{L-1} \in \mathbb{N}$, is the parametrization $\boldsymbol{\beta} = W_1 W_2 \ldots W_L$, where $W_j \in \mathbb{R}^{d_j, d_{j-1}}, j = 1, 2, \ldots, L$, with $d_L := d, d_0 := d'$. L is the depth of the factorization, the matrices $W_1, \ldots, W_L$ are the factors, and $\boldsymbol{\beta}$ is the product matrix.

Remark. We only use the case when $d = d_1 = \ldots = d_L = d' = D$. Furthermore, this over-parametrization allows us to constrain the rank of the product matrix if we limit the shared dimensions.

The matrix factorization can be considered as a linear neural network, whose weights are illustrated in Figure 5.2 below.

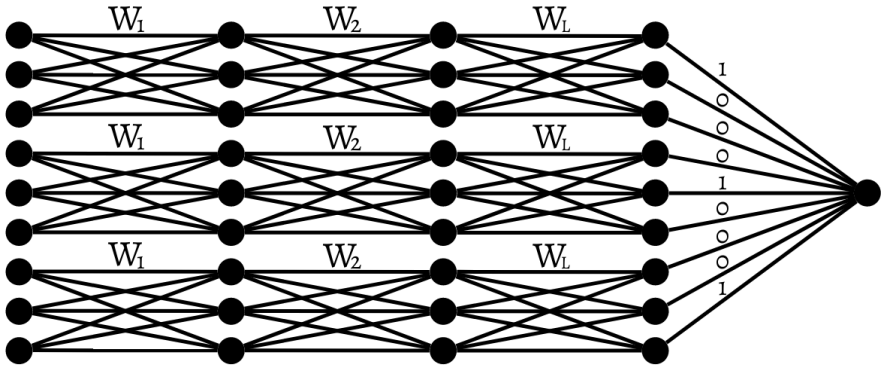

Figure 5.2: Feedforward neural network that corresponds to the matrix factorization parametrization

Arora et al., 2019 showed that if the factors in matrix factorization are updated through EGD then the product matrix $\boldsymbol{\beta}$ tends to converge to low-rank solutions without any explicit regularization. This inductive bias is useful for instance in the *Netflix problem* which is the following. The Netflix users typically rate limited number of movies and the Netflix wants to recommend titles, based on the ratings, that any particular user is likely to be willing to watch. In the data matrix the rows are the users, the columns are the movies and the aim is to complete the data matrix given these rating observations. In reality, one can assume the user-rating matrix to be low-rank because the individual's preferences might be influenced by only a few factors. Therefore, if we have the assumption that the matrix we want to complete is low-rank then perfect recovery is possible without any further regularization.

6 Separable classification with natural gradient descent

In Section 5.1 we discussed which solutions are found with EGD for two different parametrizations, now we examine the behavior of the parametrization independent algorithm NGD instead of the parametrization dependent EGD on separable classification. To perform NGD, calculating the Fisher information matrix is essential.

6.1 Fisher information matrix

We now compute the Fisher of the direct parametrization. However, from one parametrization's Fisher it is straightforward to get the Fisher of another parametrization as the statement below claims.

Statement 13. Let $\mathbf{w}$ and θ be two parameter vectors, and let $\mathcal{P}$ be a mapping such that $\mathbf{w} = \mathcal{P}(\theta)$. If the Fisher of $\mathbf{w}$ is $F(\mathbf{w})$ then the Fisher of θ is $F(\theta) = J^\top F(\mathbf{w}) J$, where $J = \frac{\partial \mathcal{P}(\theta)}{\partial \theta}$ is the Jacobian.

Proof. According to the chain rule:

$$\nabla_\theta \mathcal{L}(\theta)^\top = \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w})^\top J, \quad \text{so} \quad \nabla_\theta \mathcal{L}(\theta) = J^\top \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}),$$

hence

$$F(\theta) = \mathbb{E}_X \mathbb{E}_{Y|X}[\nabla_\theta \mathcal{L}(\theta) \nabla_\theta^\top \mathcal{L}(\theta)] = \mathbb{E}_X \mathbb{E}_{Y|X}[J^\top \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \nabla_{\mathbf{w}}^\top \mathcal{L}(\mathbf{w}) J] = J^\top F(\mathbf{w}) J.$$

□

To begin the computation of the Fisher information matrix of $\boldsymbol{\beta}$, first, calculate the gradient of the loss of one sample

$$\nabla_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta}) = \nabla_{\boldsymbol{\beta}} \log(1 + e^{-y\mathbf{x}^\top \boldsymbol{\beta}}) = \frac{-y\mathbf{x} e^{-y\mathbf{x}^\top \boldsymbol{\beta}}}{1 + e^{-y\mathbf{x}^\top \boldsymbol{\beta}}} = \frac{-y\mathbf{x}}{1 + e^{y\mathbf{x}^\top \boldsymbol{\beta}}} = -y\mathbf{x}\phi(-y\mathbf{x}^\top \boldsymbol{\beta}), \quad (6.1)$$

and the following:

$$\begin{aligned}\mathbb{E}_X[\mathbb{E}_{Y|X;\boldsymbol{\beta}}[-\nabla_{\boldsymbol{\beta}}\ell(\boldsymbol{\beta})]] &= \mathbb{E}_X[\mathbb{E}_{Y|X;\boldsymbol{\beta}}[YX\phi(-YX^\top\boldsymbol{\beta})]] = \\ &= \mathbb{E}_X[X\phi(-X^\top\boldsymbol{\beta})\phi(X^\top\boldsymbol{\beta}) - X\phi(X^\top\boldsymbol{\beta})\phi(-X^\top\boldsymbol{\beta})] = 0.\end{aligned} \tag{6.2}$$

Because of the equation above the Fisher information of the N sample equals N times the Fisher of one sample, hence compute the Fisher of a single sample

$$\begin{aligned}F_1(\boldsymbol{\beta}) &= \mathbb{E}_X[\mathbb{E}_{Y|X;\boldsymbol{\beta}}[\nabla_{\boldsymbol{\beta}}\ell(\boldsymbol{\beta})\nabla_{\boldsymbol{\beta}}^\top\ell(\boldsymbol{\beta})]] = \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{Y|X}[\mathbf{x}_n\mathbf{x}_n^\top\phi(-Y_n\mathbf{x}_n^\top\boldsymbol{\beta})^2] = \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\top(\phi(-\mathbf{x}_n^\top\boldsymbol{\beta})^2\phi(\mathbf{x}_n^\top\boldsymbol{\beta}) + \phi(x_n^\top\boldsymbol{\beta})^2\phi(-\mathbf{x}_n^\top\boldsymbol{\beta})) = \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\top\phi(\mathbf{x}_n^\top\boldsymbol{\beta})\phi(-\mathbf{x}_n^\top\boldsymbol{\beta}).\end{aligned} \tag{6.3}$$

Therefore the Fisher for the N sample is

$$F(\boldsymbol{\beta}) = \sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\top\phi(\mathbf{x}_n^\top\boldsymbol{\beta})\phi(-\mathbf{x}_n^\top\boldsymbol{\beta}). \tag{6.4}$$

Denoting it more compactly, we can write

$$F(\boldsymbol{\beta}) = X^\top\mathrm{diag}[\phi(X\boldsymbol{\beta})\odot\phi(-X\boldsymbol{\beta})]X, \tag{6.5}$$

where diag stands for a diagonal matrix whose entries on the main diagonal are represented by the vector $\phi(X\boldsymbol{\beta})\odot\phi(-X\boldsymbol{\beta})$, $\phi(X\boldsymbol{\beta})$ denotes the application of the function ϕ elementwise, and $\odot$ is for elementwise product.

In the next section we wish to consider the output of the linear neural network, namely $\mathbf{s} = X\boldsymbol{\beta}$ (or $\mathbf{s} = f(X, \mathbf{w})$) as a parametrization of the probabilistic model, therefore in the following, we also calculate its gradient and Fisher information matrix.

$$[\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})]_i = \frac{\partial\mathcal{L}(\mathbf{s})}{\partial s_i} = \frac{\partial}{\partial s_i}\sum_{n=1}^{N}\log(1+e^{-y_n s_n}) = \frac{-y_i e^{-y_i s_i}}{1+e^{-y_i s_i}} = -y_i\phi(-y_i s_i) \tag{6.6}$$

and

$$[F(\mathbf{s})]_{i,j} = [\mathbb{E}_X[\mathbb{E}_{Y|X}[\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})\nabla_{\mathbf{s}}^{\top}\mathcal{L}(\mathbf{s})]]]_{i,j} = \mathbb{E}_X[\mathbb{E}_{Y|X}[Y_i\phi(-Y_iS_i)Y_j\phi(-Y_jS_j)]] = \\ = \begin{cases} \mathbb{E}_Y[(\phi(-Y_is_i)^2] & \text{if} \quad i = j \\ \mathbb{E}_Y[Y_i\phi(-Y_is_i))]\mathbb{E}_Y[Y_j\phi(-Y_js_j))] & \text{if} \quad i \neq j \end{cases} \tag{6.7}$$

From there

$$\mathbb{E}_Y[\phi(-Y_is_i)^2] = \phi(-s_i)^2\phi(s_i) + \phi(s_i)^2\phi(-s_i) = \phi(s_i)\phi(-s_i), \tag{6.8}$$

$$\mathbb{E}_Y[Y_i\phi(-Y_is_i)] = \phi(-s_i)\phi(s_i) - \phi(s_i)\phi(-s_i) = 0 \tag{6.9}$$

Hence the Fisher of $\mathbf{s}$ is:

$$[F(\mathbf{s})]_{i,j} = \delta_{i,j}\phi(s_i)\phi(-s_i). \tag{6.10}$$

6.2 Results

Before determining the convergence behavior of the separator $\boldsymbol{\beta}_t$ we first prove an invariance property related to the transformation of the data, through which we can conclude the impossibility of large margin behavior which was the case considering EGD. We state this property separately for $N < D$ and $N \geq D$.

Theorem 1. Assume that $N < D$, X is full rank and let A be an invertible $D \times D$ matrix. Denote $\boldsymbol{\beta}_t$ and $\boldsymbol{\beta}'_t$ the trajectory of NGF on data X and $XA^{\top}$ respectively (so the transformed data points are $A\mathbf{x}_n$ with labels y_n). Then $X\boldsymbol{\beta}_t = XA^{\top}\boldsymbol{\beta}'_t$ with the assumption that $\boldsymbol{\beta}$ and $\boldsymbol{\beta}'$ have equivalent initial conditions (i. e. $X\boldsymbol{\beta}_0 = XA^{\top}\boldsymbol{\beta}'_0$).

Proof. Denote $\mathbf{s}_t = X\boldsymbol{\beta}_t$ and $\mathbf{s}'_t = XA^{\top}\boldsymbol{\beta}'_t$. Furthermore, from Equation 6.6 and 6.10 the gradient and the Fisher are the following

$$[\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})]_i = -y_i\phi(-y_is_i)$$
$$[F(\mathbf{s})]_{i,j} = \delta_{i,j}\phi(s_i)\phi(-s_i)$$

The exact same can be said about $\mathbf{s}'$, hence $\mathbf{s}$ and $\mathbf{s}'$ are the solutions of the same first-order ordinary differential equation $\dot{s} = -F^{-1}(s)\nabla_s\mathcal{L}(s)$ so with same initialization $s_0 = s'_0$ the solution is unique (its solution is determinable), thus $\mathbf{s}_t = \mathbf{s}'_t$. □

Theorem 2. Use the same assumptions as Theorem 1 but consider the case $N \geq D$ instead of $N < D$. Then $A^\top \boldsymbol{\beta}'_t = \boldsymbol{\beta}_t$.

Proof.

$$\begin{aligned}&\mathbb{E}_X[\mathbb{E}_{Y|X}[-\nabla_{\boldsymbol{\beta}'}\ell(\boldsymbol{\beta}')]] = \mathbb{E}_X[\mathbb{E}_{Y|X}[YAX\phi(-YX^\top A^\top \boldsymbol{\beta}')]] = \\ &= \mathbb{E}_X[AX\phi(-X^\top A^\top \boldsymbol{\beta}')\phi(X^\top A^\top \boldsymbol{\beta}') - AX\phi(X^\top A^\top \boldsymbol{\beta}')\phi(-X^\top A^\top \boldsymbol{\beta}')] = 0\end{aligned} \tag{6.11}$$

Because of the equation above the Fisher information of the N sample equals N times the Fisher of one sample, hence compute the Fisher of $\boldsymbol{\beta}'$ of a single sample

$$\begin{aligned}F_1(\boldsymbol{\beta}') &= \mathbb{E}_X[\mathbb{E}_{Y|X}[\nabla_{\boldsymbol{\beta}'}\ell(\boldsymbol{\beta}', XA^\top, \mathbf{y})\nabla^\top_{\boldsymbol{\beta}'}\ell(\boldsymbol{\beta}', XA^\top, \mathbf{y})]] = \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{Y|X}[\nabla_{\boldsymbol{\beta}'}\log(1+e^{-Y_n\mathbf{x}_n^\top A^\top \boldsymbol{\beta}'})\nabla^\top_{\boldsymbol{\beta}'}\log(1+e^{-Y_n\mathbf{x}_n^\top A^\top \boldsymbol{\beta}'})] = \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{Y|X}[(-Y_nA\mathbf{x}_n\phi(-Y_n\mathbf{x}_n^\top A^\top \boldsymbol{\beta}'))(-Y_nA\mathbf{x}_n\phi(-Y_n\mathbf{x}_n^\top A^\top \boldsymbol{\beta}'))^\top] = \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{Y|X}[A\mathbf{x}_n\mathbf{x}_n^\top\phi(-Y_n\mathbf{x}_n^\top A^\top \boldsymbol{\beta}')^2A^\top] =\end{aligned}$$

using the notation $\boldsymbol{v} = A^\top \boldsymbol{\beta}'$ we get

$$\begin{aligned}&= A(\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{Y|X}[\mathbf{x}_n\mathbf{x}_n^\top\phi(-Y_n\mathbf{x}_n^\top \boldsymbol{v})^2])A^\top = \\ &= A(\frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{Y|X}[\nabla_{\boldsymbol{v}}\ell(\boldsymbol{v})\nabla^\top_{\boldsymbol{v}}\ell(\boldsymbol{v})])A^\top = AF_1(\boldsymbol{v})A^\top.\end{aligned}$$

Therefore for N samples:

$$F(\boldsymbol{\beta}') = AF(\boldsymbol{v})A^\top. \tag{6.12}$$

$$F(\boldsymbol{v}) = X^\top \mathrm{diag}[\phi(X\boldsymbol{v}) \odot \phi(-X\boldsymbol{v})]X \tag{6.13}$$

as we have seen before in Equation 6.5. Note, that the Fisher of $\boldsymbol{v}$ must be invertible because its size is $D \times D$ and its rank is D as well. Hence, $F(\boldsymbol{\beta}')$ is also invertible.

Specify $\nabla_{\boldsymbol{\beta}'}\ell(\boldsymbol{\beta}')$ and let $J = \frac{\partial v}{\partial \boldsymbol{\beta}'}$:

$$\nabla_{\boldsymbol{\beta}'}\ell(\boldsymbol{\beta}', XA^\top, \mathbf{y}) = J^\top\nabla_{\boldsymbol{v}}\ell(\boldsymbol{v}, X, \mathbf{y}) \tag{6.14}$$

$$J_{i,j} = \frac{\partial \boldsymbol{v}_i}{\partial \boldsymbol{\beta}'_j} = \frac{\partial \sum_{k=1}^{D}\boldsymbol{\beta}'_k A_{k,i}}{\partial \boldsymbol{\beta}'_j} = A_{j,i} \tag{6.15}$$

Therefore $J = A^\top$ so $J^\top = A$,

$$\nabla_{\boldsymbol{\beta}'}\ell(\boldsymbol{\beta}') = A\nabla_{\boldsymbol{v}}\ell(\boldsymbol{v}). \tag{6.16}$$

Now consider NGF on $\boldsymbol{\beta}'$:

$$\begin{aligned}\dot{\boldsymbol{\beta}}' &= -F(\boldsymbol{\beta}')^{-1}\nabla_{\boldsymbol{\beta}'}\mathcal{L}(\boldsymbol{\beta}') = -(AF(\boldsymbol{v})A^\top)^{-1}\sum_{n=1}^{N}\nabla_{\boldsymbol{\beta}'}\ell(\boldsymbol{\beta}') = \\ &= -(A^\top)^{-1}F(\boldsymbol{v})^{-1}A^{-1}A\sum_{n=1}^{N}\nabla_{\boldsymbol{v}}\ell(\boldsymbol{v}) = -(A^\top)^{-1}F(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v})\end{aligned} \tag{6.17}$$

by the chain rule we have

$$\dot{\boldsymbol{v}} = J\dot{\boldsymbol{\beta}}' = A^\top\dot{\boldsymbol{\beta}}'. \tag{6.18}$$

From Equation (6.17) and (6.18) we get:

$$\dot{\boldsymbol{v}} = F(\boldsymbol{v})^{-1}\nabla_{\boldsymbol{v}}\mathcal{L}(\boldsymbol{v}). \tag{6.19}$$

Therefore $\boldsymbol{v}_t$ is a solution for the same first-order differential equation as $\boldsymbol{\beta}$, and because of the same initialization ($X\boldsymbol{\beta}_0 = XA^\top\boldsymbol{\beta}'_0 = X\boldsymbol{v}_0 \to \boldsymbol{\beta}_0 = A^\top\boldsymbol{\beta}'_0 = \boldsymbol{v}_0$) $\boldsymbol{\beta}_t = \boldsymbol{v}_t = A^\top\boldsymbol{\beta}'_t$. □

Conclusion. Denote $s_t(X, \mathbf{y})$ the trajectory of $X\boldsymbol{\beta}_t$, which is the function $\boldsymbol{\beta}_t^\top\mathbf{x}$ evaluated at each data points $\mathbf{x}_n$. Then $s_t(XA^\top, \mathbf{y}) = s_t(X, \mathbf{y})$.

Proof. $s_t(XA^\top, \mathbf{y}) = XA^\top\boldsymbol{\beta}_t(XA^\top, \mathbf{y}) = X\boldsymbol{\beta}_t(X, \mathbf{y}) = s_t(X, \mathbf{y})$ □

A special case of this invariance property is when $A = aI$ diagonal matrix. In this case Theorem 1, 2 means that the coordinates of $\boldsymbol{\beta}'_t$ is the one of $\boldsymbol{\beta}_t$ multiplied by $\frac{1}{a}$. This behavior rules out implicit regularization as a non-data-dependent norm of $\boldsymbol{\beta}_t$, specially the ℓ_p large margin behavior. Accordingly, it is natural to consider a method that indeed has the invariance property stated above.

Regard the ordinary least squares regression (OLS) which gives a solution for the problem $X\boldsymbol{\beta} = \mathbf{y}$. The particular solution it finds is $\boldsymbol{\beta}_{OLS} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\|\mathbf{y} - X\boldsymbol{\beta}\|^2$ and if the columns of the matrix X are independent, then $\boldsymbol{\beta}_{\text{OLS}} = (X^\top X)^{-1}X^\top\mathbf{y}$. We now show the invariance property for OLS that was stated in Theorem 1,2. Again, we split the problem into two cases: $N < D$ and $N \geq D$.

Statement 14. Assume that $N < D$, X is full rank and let A be an invertible $D \times D$ matrix. Denote $\boldsymbol{\beta}_{OLS}$ and $\boldsymbol{\beta}'_{OLS}$ the ordinary least squares solution for the matrix X and $XA^\top$ respectively and label $\mathbf{y}$ in both cases. Then $X\boldsymbol{\beta}_{OLS} = XA^\top\boldsymbol{\beta}'_{OLS}$.

Proof. Immediately follows from the definition of the problems: $X\boldsymbol{\beta}_{OLS} = \mathbf{y}$ and $XA^\top\boldsymbol{\beta}'_{OLS} = \mathbf{y}$. □

Statement 15. Use the same assumptions as Statement 14 but consider the case $N \geq D$ instead of $N < D$. Then $A^\top\boldsymbol{\beta}'_{OLS} = \boldsymbol{\beta}_{OLS}$.

Proof. Due to the assumption that X is full rank in this case the columns of X are independent, therefore

$$A^\top\boldsymbol{\beta}'_{OLS} = A^\top((XA^\top)^\top XA^\top)^{-1}(XA^\top)^\top y = A^\top A^{\top -1}(X^\top X)^{-1}A^{-1}AX^\top y = \boldsymbol{\beta}_{OLS}$$

□

The next theorems illustrate the further possible connection between the solution that NGD finds and the OLS solution.

Theorem 3. Assume that $N \leq D$, X is full rank and the parameters $\boldsymbol{\beta}_t$ of a linear model follow NGF, then the output of the network $\mathbf{s}_t = X\boldsymbol{\beta}_t$ follows an asymptotically linear trajectory with direction vector $\mathbf{y}$.

This theorem is a special case of Theorem 4 which is stated later.

Remark. The reason why we consider $\mathbf{s}$ instead of $\boldsymbol{\beta}$ is the following. Since the Fisher information matrix of $\boldsymbol{\beta}$ is $F(\boldsymbol{\beta}) = X^\top \mathrm{diag}[\phi(X\boldsymbol{\beta}) \odot \phi(-X\boldsymbol{\beta})]X$, it can be seen that $\mathrm{rank}(F(\boldsymbol{\beta})) = \mathrm{rank}(X) \leq N < D$, so $F(\boldsymbol{\beta})$ is not invertible, hence several NGF paths are possible. Intuitively, when $\mathrm{rank}(X) = N$ $\boldsymbol{\beta}$ has $D - N$ degrees of freedom and with $\mathbf{s}$ $\boldsymbol{\beta}$ is indeed described on N dimensions.

When there are more parameters than data points, with NGD the output of the linear network is a solution that interpolates the labels $\mathbf{y}$ perfectly just as OLS does. Furthermore, OLS finds the minimum ℓ_2 norm solution to the problem $X\boldsymbol{\beta} = \mathbf{y}$ and the Moore-Penrose pseudoinverse $\boldsymbol{\beta} = X^+\mathbf{y}$ also yields the smallest ℓ_2 norm solution. This fact invited the following conjecture.

Conjecture. If the Moore-Penrose pseudoinverse is used to calculate the natural gradient descent direction, i. e. Eqn. (4.29), then $\boldsymbol{\beta}_t$ converges in direction to the OLS solution.

Theorem 3 only refers to a linear model, specially the simple $X\boldsymbol{\beta}$ model, but actually, the statement of the theorem holds in a more general case, namely for arbitrary non-linear over-parametrized models as well. Let $\theta \in \mathbb{R}^P$ be the parameters of a neural network $\mathbf{s} = f(X;\theta) \in \mathbb{R}^N$ whose output define the probabilistic model discussed in Section 5.1.

Theorem 4. Assume that $N \leq P$, the parameter θ_t follows NGF and the Jacobian $J_t = \frac{\partial \mathbf{s}_t}{\partial \theta_t}$ is full rank. Then $\mathbf{s}_t$ follows asymptotically linear trajectory with direction vector $\mathbf{y}$.

Proof. Note that the assumptions of Statement 11 are satisfied, because $\dim \mathbf{s} \leq \dim \theta$, the Jacobian is full rank, and the Fisher $F(\mathbf{s}_t)$ is invertible since it is the exact same as in the $\mathbf{s} = X\boldsymbol{\beta}$ case (see Equation 6.10). Therefore the trajectory of $\mathbf{s}$ is the same as the trajectory of θ, if $\mathbf{s}_0 = f(X, \theta_0)$. The path of $\mathbf{s}$ is defined by

$$\dot{\mathbf{s}} = -F^{-1}(\mathbf{s})\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s}). \tag{6.20}$$

First assume $\mathbf{s}$ is 1-dimensional and use the notation $s = \mathbf{s}$, $\mathbf{x} = \mathbf{x}_n$ and $y = \mathbf{y}$. To solve Equation 6.20 it requires the gradient and the Fisher information matrix of s which are the following (from Equation 6.6 and 6.10)

$$\nabla_s \mathcal{L}(s) = -y\phi(-ys) \tag{6.21}$$

and

$$F(s) = \phi(s)\phi(-s). \tag{6.22}$$

Then Equation 6.20 can be written as

$$\dot{s} = \frac{y\phi(-ys)}{\phi(s)\phi(-s)}. \tag{6.23}$$

Now multiply both sides of the equation above by y

$$y\dot{s} = \frac{\phi(-ys)}{\phi(ys)\phi(-ys)} \tag{6.24}$$

and substitute $\tilde{s} = ys$

$$\frac{\partial \tilde{s}}{\partial t} = \frac{1}{\phi(\tilde{s})}. \tag{6.25}$$

The solution to this differential equation is

$$\tilde{s} = \log(e^{t+c} - 1), \tag{6.26}$$

where c is a constant. From Equation 6.26 we get the asymptotic behaviour

$$\lim_{t\to\infty} \frac{\tilde{s}}{t+c} = \lim_{t\to\infty} \frac{\log(e^{t+c}-1)}{t+c} = \lim_{t\to\infty} \frac{e^{t+c}}{e^{t+c}-1} = \lim_{t\to\infty} \frac{1}{1-e^{-(t+c)}} = 1 \tag{6.27}$$

using the L'Hopital Rule.

We proved Theorem 4 for $N = 1$, now consider $N > 1$ case. Now the gradient and the Fisher are

$$[\nabla_{\mathbf{s}} \mathcal{L}(\mathbf{s})]_i = -y_i \phi(-y_i s_i) \tag{6.28}$$

And the Fisher information matrix:

$$[F(\mathbf{s})]_{i,j} = \delta_{i,j} \phi(s_i)\phi(-s_i). \tag{6.29}$$

Substituting Equation 6.28 and 6.29 in Equation 6.20 we get

$$\begin{pmatrix} \dot{s_1} \\ \dot{s_2} \\ \vdots \\ \dot{s_N} \end{pmatrix} = - \begin{pmatrix} \frac{1}{\phi(s_1)\phi(-s_1)} & 0 & \cdots & 0 \\ 0 & \frac{1}{\phi(s_2)\phi(-s_2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\phi(s_N)\phi(-s_N)} \end{pmatrix} \begin{pmatrix} -y_1\phi(-y_1 s_1) \\ -y_2\phi(-y_2 s_2) \\ \vdots \\ -y_N\phi(-y_N s_N) \end{pmatrix} = \begin{pmatrix} \frac{y_1\phi(-y_1 s_1)}{\phi(s_1)\phi(-s_1)} \\ \frac{y_2\phi(-y_2 s_2)}{\phi(s_2)\phi(-s_2)} \\ \vdots \\ \frac{y_N\phi(-y_N s_N)}{\phi(s_N)\phi(-s_N)} \end{pmatrix}$$

These are N independent differential equations and each is the same as Euquation 6.23 in the $N = 1$ case. Then multiply the nth equation by y_n and substitute $\tilde{s}_n = y_n s_n$ as we did before. As a result in each dimension $\tilde{\mathbf{s}}$ is asymptotically $t + c_n$ for some constant c_n. Hence $\tilde{\mathbf{s}} \approx t\mathbb{1} + \tilde{\boldsymbol{c}}$, $\tilde{\boldsymbol{c}} \in \mathbb{R}^D$ and $\mathbf{s} \approx t\mathbf{y} + \boldsymbol{c}$, where $\boldsymbol{c} \in \mathbb{R}^D$ is a constant. □

Remark. If our network is the linear $\mathbf{s} = X\mathbf{w}$ then $J = X$, so Theorem 3 is a special case of Theorem 4 indeed.

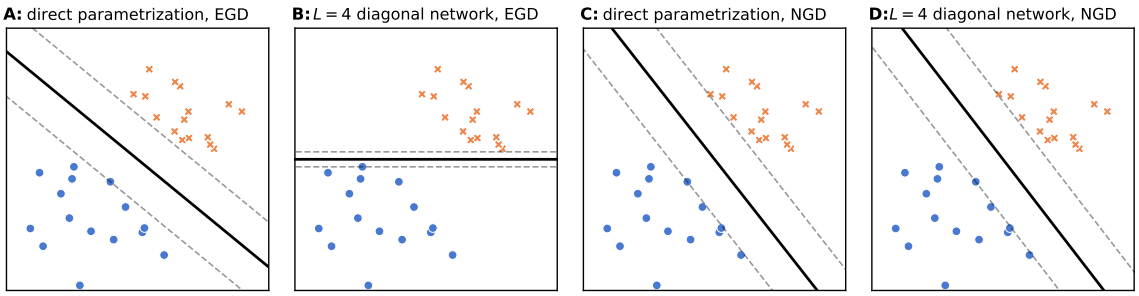

Figure 6.1: Inductive bias of EGD and NGD in separable classification

Figure 6.1 from Kerekes, Mészáros, and Huszár, 2021 illustrates the parametrization dependence with EGD and independence with NGD. It shows which solution is found with EGD and NGD using direct and diagonal parametrizations on a separable classification data set. Clearly, EGD leads to different separator with different parametrization, while NGD finds the same solution.

7 Matrix completion with natural gradient descent

In Section 5.2 we discussed which solutions are found with EGD for two different parametrizations, now we examine the behavior of the parametrization independent algorithm NGD instead of the parametrization dependent EGD on matrix completion. To perform NGD, calculating the Fisher information matrix is essential.

7.1 Fisher information matrix

We now compute the Fisher of the direct parametrization. However, from one parametrization's Fisher it is straightforward to get the Fisher of another parametrization as the Statement 13 claims.

To begin the computation of the Fisher information matrix of $\boldsymbol{\beta}$, first, calculate the gradient of the loss. The gradient is a matrix where it contains 0 in the place of unobserved entries, and has $\frac{-y_n+\langle A_n,\boldsymbol{\beta}\rangle}{\sigma^2}$ where there was an observation. The Fisher is defined as

$$F(\boldsymbol{\beta}) = \mathbb{E}_X[\mathbb{E}_{Y|X}[\nabla_{\boldsymbol{\beta}}\mathcal{L}(\boldsymbol{\beta})\nabla_{\boldsymbol{\beta}}^{\top}\mathcal{L}(\boldsymbol{\beta})]]. \tag{7.1}$$

In the case of separable classification we computed the expectation $\mathbb{E}_X$ over the empirical distribution, but now we chose another: $B_1, B_2, \ldots, B_D$, where B_i has an entry 1 in the ith row ith column and 0 everywhere else (the loss is obviously summed over B_is). Now the Fisher is

$$F(\boldsymbol{\beta}) = \mathbb{E}_{Y|X}\begin{pmatrix} \frac{(y_1-\langle B_1,\boldsymbol{\beta}\rangle)^2}{\sigma^4} & 0 & \cdots & 0 \\ 0 & \frac{(y_2-\langle B_2,\boldsymbol{\beta}\rangle)^2}{\sigma^4} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{(y_D-\langle B_D,\boldsymbol{\beta}\rangle)^2}{\sigma^4} \end{pmatrix} = \frac{\sigma^2}{\sigma^4}I = \frac{1}{\sigma^2}I. \tag{7.2}$$

7.2 Results

The theorem below states that natural gradient descent in the case of matrix completion finds the trivial solution.

Theorem 5. The solution of NGF converges to the trivial solution under any overparametrization θ for which $\dim \boldsymbol{\beta} \leq \dim \theta$, $J = \frac{\partial \boldsymbol{\beta}_t}{\partial \theta_t}$ is full rank (where $\boldsymbol{\beta}$ is the direct parametrization) and we use the same initialization.

Proof. Consider a parametrization θ such that $\boldsymbol{\beta} = \mathcal{P}(\theta)$ and let θ_t be the solution of NGF in θ. Futhermore, we know that $J = \frac{\partial \boldsymbol{\beta}_t}{\partial \theta_t}$ is full rank. The Fisher of $\boldsymbol{\beta}$ is invertible ($F(\boldsymbol{\beta}) = \frac{1}{\sigma^2} I$), therefore from Statement 11 $\boldsymbol{\beta}_t = \mathcal{P}(\theta_t)$ solves the differential equation

$$\dot{\boldsymbol{\beta}} = -F(\boldsymbol{\beta})^{-1} \nabla_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}) \tag{7.3}$$

which is

$$\dot{\boldsymbol{\beta}} = -\sigma^2 \nabla_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta}). \tag{7.4}$$

Its solution is

$$\boldsymbol{\beta}_{ij}(t) = \boldsymbol{\beta}_{ij}(0) \tag{7.5}$$

when $\boldsymbol{\beta}_{ij}$ is unobserved and

$$\boldsymbol{\beta}_{ij}(t) = y_n + ce^{-t} \tag{7.6}$$

when $\boldsymbol{\beta}_{ij}$ is observed.

$$\lim_{t \to \infty} \boldsymbol{\beta}_{ij}(t) = y_n. \tag{7.7}$$

□

The theorem implies that NGF finds a solution in which observed entries converge to the observed value, and everywhere else the values do not change from the initialization. While Euclidean gradient descent generalizes well in matrix factorization with the low-rank assumption and has an implicit regularization towards low rank, natural gradient descent fails to generalize and finds the trivial solution.

8 Summary

In gradient descent how we parametrize the hypothesis plays a significant role in the underdetermined problems where the inductive bias has an important role. EGD in separable classification is able to recover sparse separators, and in matrix completion it can identify low-rank matrices with the right parametrization. These inductive biases have an essential connection with good generalization. However, there exist algorithms that are irrespective of parametrization.

We examined the behavior of natural gradient descent, which finds the same solution with whichever parametrization. We were interested in characterizing this particular solution in order to understand the importance of parametrization. We found that in the case of separable classification the output of the network with NGD converges to the labels $\mathbf{y}$ and we suspect that the separator converges to the ordinary least squares solution if the problem is over-parametrized. Where, in contrast, EGD has a large margin behavior. Considering matrix completion NGD finds the trivial solution, namely the observed entries converge to the observed value while others do not move from the initialization. We see how it fails to generalize completely.

Even though NGD in these cases did not perform desirably, it does not mean that NGD is always useless. As this algorithm is often used because of its fast convergence, it might be the case that NGD minimizes the training loss effectively at the cost of poorer generalization.

Through these results, the role of parameter-to-hypothesis mapping can be understood better. Besides the role of initialization and properties of stochastic gradient descent, parameter-to-hypothesis mapping is an important aspect of implicit regularization. Consequently, now we also have deeper knowledge about implicit regularization. As it was shown, explicit regularization is neither necessary nor by itself sufficient for controlling generalization error. Therefore, implicit regularization has a significant role in understanding the way of generalization in deep learning and understanding how the inductive bias selects a model from the several functions fitting to the training data. This knowledge is essential in designing architectures that can generalize significantly well in different deep learning problems.

9 Bibliography

Alzubaidi, Laith et al. (2021). "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions". In: URL: https://link.springer.com/article/10.1186/s40537-021-00444-8.

Nagpal, Shaveta et al. (2019). "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning". In: *Archives of Computational Methods in Engineering*.

He, Kaiming et al. (2020). "Mask R-CNN". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2. ISSN: 19393539. DOI: 10.1109/TPAMI.2018.2844175.

Roberts, Daniel A (2018). "SGD implicitly regularizes generalization error". In: *In Integration of Deep Learning Theories Workshop*. URL: https://arxiv.org/abs/2104.04874.

Battaglia, Peter W et al. (2018). "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261*.

Hernández-García, Alex and Peter König (2018). "Data augmentation instead of explicit regularization". In: *arXiv preprint arXiv:1806.03852*.

Zawadzka-Gosk, Emilia, Krzysztof Wołk, and Wojciech Czarnowski (2019). "Deep learning in State-of-The-Art image classification exceeding 99% accuracy". In.

Zhang, Chiyuan et al. (2017). "Understanding deep learning requires rethinking generalization". In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Arora, Sanjeev et al. (May 2019). "Implicit Regularization in Deep Matrix Factorization". In: *Advances in Neural Information Processing Systems* 32. URL: http://arxiv.org/abs/1905.13655.

Gunasekar, Suriya et al. (2018). "Implicit Bias of Gradient Descent on Linear Convolutional Networks". In: *Advances in Neural Information Processing Systems*. URL: http://arxiv.org/abs/1806.00468.

Kerekes, Anna, Anna Mészáros, and Ferenc Huszár (2021). *Depth Without the Magic: Inductive Bias of Natural Gradient Descent*. DOI: 10.48550/ARXIV.2111.11542. URL: https://arxiv.org/abs/2111.11542.

Verhóczki, László (2020). "A sokaságok differenciálgeometriája". In: URL: https://web.cs.elte.hu/geometry/vl/SokasagokJegyzetVL.pdf.

O'Neill, Barrett (1983). "Semi-Riemannian Geometry With Applications to Relativity". In: URL: https://books.google.hu/books?id=CGk1eRSjFIIC&pg=PA54&hl=hu&source=gbs_toc_r&cad=4#v=onepage&q&f=false.

Absil, P.-A., R. Mahony, and R. Sepulchre (2008). "Optimization Algorithms on Matrix Manifolds". In: URL: https://www.scribd.com/book/233094479/Optimization-Algorithms-on-Matrix-Manifolds?utm_medium=cpc&utm_source=google_search&utm_campaign=3Q_Google_DSA_NB_RoW&utm_device=c&gclid=CjwKCAjwur-SBhB6EiwA5sKtjuU5_JNIyv0WcuFWrai-EcY_-fIb02MEYwyWy4xjViKMEzUJjWTHORoCWdsQAvD_BwE.

Kristiadi, Agustinus (2019). "Optimization and Gradient Descent on Riemannian Manifolds". In: URL: https://agustinus.kristia.de/techblog/2019/02/22/optimization-riemannian-manifolds/.

Jäckel, Frieder (2017). *Definition of gradient of a function f in Riemannian manifold*. Mathematics Stack Exchange. URL: https://math.stackexchange.com/q/2250437.

Carmo, Manfredo Do (1992). "Riemannian Geometry". In: URL: https://blackwells.co.uk/bookshop/product/Riemannian-Geometry-by-Manfredo-Perdigo-do-Carmo/9780817634902.

Amari, Shun-ichi (1997). "Neural Learning in Structured Parameter Spaces - Natural Riemannian Gradient". In: URL: https://proceedings.neurips.cc/paper/1996/file/39e4973ba3321b80f37d9b55f63ed8b8-Paper.pdf.

Pascanu, Razvan and Yoshua Bengio (2013). "Revisiting Natural Gradient for Deep Networks". In: URL: https://arxiv.org/abs/1301.3584v7.

Daróczy, Bálint, Rita Aleksziev, and András Benczúr (2019). "Tangent Space Separability in Feedforward Neural Networks". In: URL: https://eprints.sztaki.hu/9954/1/Daroczy_1_31319087_ny.pdf.

Soudry, Daniel et al. (2017). "The Implicit Bias of Gradient Descent on Separable Data". In: 19. URL: https://arxiv.org/abs/1710.10345v4.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection Via the Lasso". In: URL: https://onlinelibrary.wiley.com/doi/full/10.1111/j.2517-6161.1996.tb02080.x.