Contents

1 Introduction **2**

2 Neural Networks **3**
2.1 Notation . . . 3
2.2 Training . . . 5

3 Kernel Gradient Descent **15**
3.1 Functional Derivatives . . . 15
3.2 Kernel Gradient . . . 20
3.3 Kernel Approximation . . . 25
3.4 Neural Tangent Kernel . . . 27

4 Dynamic Linear Dimensionality Reduction **30**
4.1 Approach . . . 30
4.2 Theory . . . 31
4.2.1 Singular Value Decomposition . . . 31
4.2.2 Dimensionality Reduction . . . 32
4.3 Methodology . . . 36
4.3.1 Orthogonal Projections . . . 36
4.3.2 Principal Component Analysis . . . 37
4.3.3 Algorithm . . . 39

5 Numerical Experiments **40**

6 Conclusion **41**

7 Appendix **42**

1 Introduction

This thesis is about...

2 Neural Networks

This section introduces the basic concepts of neural networks and provides the necessary notation for further sections. We start with the formal definition of neural networks and explain the process of training afterwards. Finally we will introduce several training methods.

2.1 Notation

To start things off, we consider the simple model of a neuron.

Definition 2.1. *Let $n_x \in \mathbb{N}$, $\mathsf{w} \in \mathbb{R}^{n_x}$, $\mathsf{b} \in \mathbb{R}$ and $\sigma : \mathbb{R} \to \mathbb{R}$. A neuron is defined as*

$$\nu : \mathbb{R}^{n_x} \to \mathbb{R} : x \mapsto \sigma\Big(\mathsf{w}^\mathsf{T} x + \mathsf{b}\Big).$$

In this notion σ is called the activation function, w the weights and b the bias of ν.

As a generalization of this concept, one can use multiple neurons in parallel to form a layer.

Definition 2.2. *Let $n_x, n_y \in \mathbb{N}$, $W = [\mathsf{w}_1, \dots, \mathsf{w}_{n_y}] \in \mathbb{R}^{n_x \times n_y}$, $\mathsf{b} \in \mathbb{R}^{n_y}$ and $\sigma : \mathbb{R} \to \mathbb{R}$. Given neurons ν_i with weights w_i, bias b_i and activation function σ for $i = 1, \dots, n_y$, we define a layer as*

$$f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y} : x \mapsto \begin{bmatrix} \nu_1(x) \\ \vdots \\ \nu_{n_y}(x) \end{bmatrix} = \sigma\Big(W^\mathsf{T} x + \mathsf{b}\Big),$$

where the activation function σ is applied entry-wise.

To provide a better understanding, these basic concepts can be visualized as graphs.

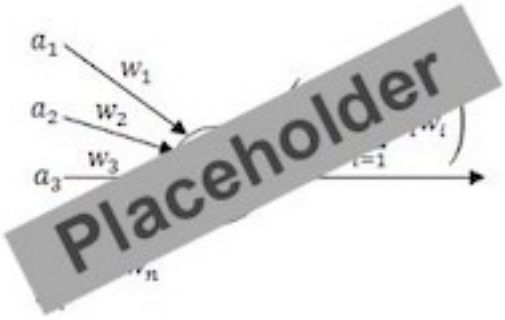

Figure 1: Illustration of a neuron as graph.

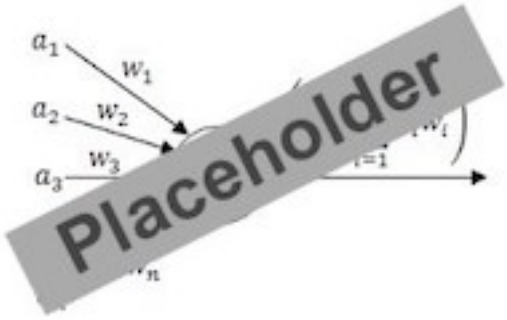

Figure 2: Illustration of a layer as graph.

In the following we will no longer distinguish between weights and biases and simply refer to them as parameters. Logically one can use the output of a layer as input for another layer. Iteratively doing so will construct specific functions known as neural networks. In order to formalize this idea we denote the following definition.

Definition 2.3. *Let $l \in \mathbb{N}$ and $n_0, \dots, n_l \in \mathbb{N}$. A neural network is defined as a function*

$$f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_l} : x \mapsto f_l \circ f_{l-1} \circ \cdots \circ f_1(x),$$

where f_i for $i = 1, \dots, l$ is a layer with input dimension n_{i-1} and output dimension n_i. Networks with a certain level of complexity, say $l > 2$, are referred to as deep neural networks.

Just like neurons and layers, neural networks can also be visualized as graphs.

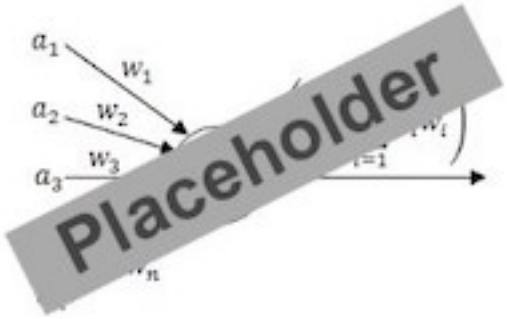

Figure 3: Illustration of a neural network as graph.

Neural networks find massive adoption in machine learning as they can model arbitrary complex functions. For example one could envision functions that predict the weather at a specific place and time or functions that determine if an image contains a cat or a dog. Clearly there is no simple mathematical formula to describe such problems.

A common practice to learn those functions is, to fix the number of layers, the number of neurons per layer as well as the activation functions, to choose some initial parameters and then iteratively adjust the parameters in order to approximate the unknown target function. In the notion of definition 2.3 one can determine the total number parameters in f as

$$n = \sum_{i=0}^{l-1} (n_i + 1) \cdot n_{i+1}.$$

Denoting these parameters as a vector $w \in \mathbb{R}^n$ enables us to define a function

$$\tilde{f} : \mathbb{R}^{n_0} \times \mathbb{R}^n \to \mathbb{R}^{n_l} : (x, w) \mapsto \tilde{f}(x, w),$$

such that for fixed $w_1, w_2 \in \mathbb{R}^n$ the functions $\tilde{f}(x, w_1)$ and $\tilde{f}(x, w_2)$ are neural networks matching in their number of layers, number of neurons as well as activation functions and differing only in their choice of parameters. One calls $\tilde{f}(x, w_1)$ and $\tilde{f}(x, w_2)$ realizations of the neural network architecture $\tilde{f}$.

In the following, the terms neural network and neural network architecture will be used synonymously if the meaning is clear by the context. Unless otherwise specified, $n_0 \in \mathbb{N}$ will denote the dimension of the input layer, $n_l \in \mathbb{N}$ will denote the dimension of the output layer and $n \in \mathbb{N}$ will denote the number of adjustable parameters.

2.2 Training

Training a neural network refers to finding the optimal parameters, given a fixed neural network architecture $\tilde{f} : \mathbb{R}^{n_0} \times \mathbb{R}^n \to \mathbb{R}^{n_l}$. In other words, finding a function $f \in \tilde{\mathcal{F}}$, where

$$\tilde{\mathcal{F}} = \Big\{ f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_l} : x \mapsto \tilde{f}(x, w) \mid w \in \mathbb{R}^n \Big\}$$

denotes the set of all neural networks with given architecture. In order to define a sense of optimality, we need a mechanism measuring the quality of network outputs. This is done by a loss function

$$\ell : \mathbb{R}^{n_l} \times \mathbb{R}^{n_l} \to \mathbb{R}_+ : \big(f(x, w), y\big) \mapsto \ell\big(f(x, w), y\big),$$

that should be chosen in such a way, that for some input $x \in \mathbb{R}^{n_0}$ with target output $y \in \mathbb{R}^{n_l}$ it associates some cost with the error between the prediction $f(x, w)$ and the true label y.

Under the assumption, that in reality there exists a probability distribution $\mathcal{D}$ on $\mathbb{R}^{n_0} \times \mathbb{R}^{n_l}$, describing the correlation between inputs $x \in \mathbb{R}^{n_0}$ and target values $y \in \mathbb{R}^{n_l}$, we call a parameter vector $w_* \in \mathbb{R}^n$ to be optimal, if

$$\forall w \in \mathbb{R}^n : R(w_*) \leqslant R(w),$$

where $R(w)$ denotes the so called risk or generalization error

$$R : \mathbb{R}^n \to \mathbb{R}_+ : w \mapsto \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\ell\big(f(x, w), y\big) \Big].$$

For later use we will denote the marginal distribution of the inputs as $\mathcal{D}_x$. At this point it should be noted, that in general the underlying probability distribution $\mathcal{D}$ is unknown. Hence we can not evaluate the expectation in previous expression.

For this reason, the training of neural networks requires labeled training data

$$\Big\{ (x_i, y_i) \in \mathbb{R}^{n_0} \times \mathbb{R}^{n_l} \mid i = 1, \dots, m \Big\}$$

for some $m \in \mathbb{N}$. Instead of working with the generalization error, the training data enables us to minimize the empirical error function

$$E : \mathbb{R}^n \to \mathbb{R}_+ : w \mapsto \frac{1}{m} \sum_{i=1}^{m} \ell\big(f(x_i, w), y_i\big) + \lambda \|w\|_2^2,$$

where $\lambda \geqslant 0$ is some regularization parameter used to control the complexity of w.

Intuitively minimization of the empirical error should lead to a minimization of the generalization error as well. Currently, a lot of research is being done on how these measures relate to each other. Nevertheless this thesis will focus on how to minimize the empirical error given labeled training data.

The problem of finding some parameter vector w that minimizes the empirical error function is usually hard to solve as the error may have highly nonlinear dependencies on the parameters. This is why in practice one often aims to compare several local minima and to settle for one that is sufficient enough, regardless of whether it is a global minimum or not.

Definition 2.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. The gradient of f is defined as*

$$\nabla f : \mathbb{R}^n \to \mathbb{R}^n : x = (x_1, \ldots, x_n) \mapsto \left[\frac{\partial}{\partial x_1} f(x), \cdots, \frac{\partial}{\partial x_n} f(x) \right]^\mathsf{T}.$$

Due to the complexity of the error function, in general there is no easy way to find an analytical solution to the problem $\nabla E(w) = 0$. Hence most of the techniques for error minimization are based on iterative approximation. One popular method is gradient descent, which was first raised by Cauchy in [2].

Algorithm 1 Gradient Descent

1: Requires differentiable $f : \mathbb{R}^n \to \mathbb{R}$, random $x_0 \in \mathbb{R}^n$ and $\epsilon \geqslant 0$.

2: Initialize $k \leftarrow 0$.

3: **while** $\|\nabla f(x_k)\|_2 > \epsilon$ **do**:

4: Choose a step size $\alpha_k > 0$.

5: Set $x_{k+1} \leftarrow x_k - \alpha_k \cdot \nabla f(x_k)$ and $k \leftarrow k + 1$.

6: **end while**

We will see, that in the case where f is bounded from below and ∇f is Lipschitz continuous, we can guarantee the convergence of gradient descent to a stationary point.

Definition 2.5. *Let $(\mathcal{X}, d_\mathcal{X})$ and $(\mathcal{Y}, d_\mathcal{Y})$ be two metric spaces. A function $f : \mathcal{X} \to \mathcal{Y}$ is called Lipschitz continuous, if there exists $L \geqslant 0$, such that*

$$\forall x, x' \in \mathcal{X} : d_\mathcal{Y}\big(f(x), f(x')\big) \leqslant L \cdot d_\mathcal{X}(x, x').$$

The smallest constant L, that suffices the previous condition is called Lipschitz constant of f.

Casually speaking, the Lipschitz continuity of ∇f ensures that the gradient does not change heavily for two points that are close to each other.

Definition 2.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. A stationary point of f is an element of*

$$\big\{x \in \mathbb{R}^n \mid \nabla f(x) = 0\big\}.$$

There a three kinds of stationary points: saddle points, local extrema and global extrema. We are especially interested in global minima as they provide the lowest possible error.

Definition 2.7. *Let $\mathcal{X}$ be a subset of some real vector space and $f : \mathcal{X} \to \mathbb{R}$ be bounded from below. We call $x_* \in \mathcal{X}$ a global minimizer of f, if*

$$\forall x \in \mathcal{X} : f(x_*) \leqslant f(x).$$

To prove convergence of the gradient descent algorithm for bounded f with Lipschitz continuous gradient, we need the following result which is formulated as lemma 1.2.3 in [3].

Lemma 2.8. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable, such that the gradient ∇f is Lipschitz continuous with Lipschitz constant $L \geqslant 0$, then*

$$\forall x, x' \in \mathbb{R}^n : f(x') \leqslant f(x) + \left\langle \nabla f(x), x' - x \right\rangle + \frac{L}{2}\|x' - x\|_2^2.$$

Proof. Let $x, x' \in \mathbb{R}^n$, define $z_\lambda \coloneqq x + \lambda(x' - x)$ for $\lambda \in \mathbb{R}$ and consider the function

$$\phi : \mathbb{R}^n \to \mathbb{R} : \lambda \mapsto f(z_\lambda).$$

Since f is differentiable we can apply the chain rule to derive

$$\phi' : \mathbb{R}^n \to \mathbb{R} : \lambda \mapsto (x' - x)^\mathsf{T} \cdot \nabla f(z_\lambda).$$

Integration over ϕ' from $\lambda = 0$ to $\lambda = 1$ yields

$$\int_0^1 \left\langle \nabla f(z_\lambda), x' - x \right\rangle d\lambda = \int_0^1 \phi'(\lambda)\ d\lambda = \phi(1) - \phi(0) = f(x') - f(x).$$

This equality can be rewritten as

$$\begin{aligned} f(x') - f(x) &= \int_0^1 \left\langle \nabla f(z_\lambda), x' - x \right\rangle d\lambda \\ &= \int_0^1 \left\langle \nabla f(z_\lambda) - \nabla f(x) + \nabla f(x), x' - x \right\rangle d\lambda \\ &= \left\langle \nabla f(x), x' - x \right\rangle + \int_0^1 \left\langle \nabla f(z_\lambda) - \nabla f(x), x' - x \right\rangle d\lambda. \end{aligned}$$

Using the Cauchy-Schwarz inequality we derive

$$\begin{aligned} \left| f(x') - f(x) - \left\langle \nabla f(x), x' - x \right\rangle \right| &= \left| \int_0^1 \left\langle \nabla f(z_\lambda) - \nabla f(x), x' - x \right\rangle d\lambda \right| \\ &\leqslant \int_0^1 \left| \left\langle \nabla f(z_\lambda) - \nabla f(x), x' - x \right\rangle \right| d\lambda \\ &\leqslant \int_0^1 \left\| \nabla f(z_\lambda) - \nabla f(x) \right\|_2 \cdot \left\| x' - x \right\|_2 d\lambda. \end{aligned}$$

Due to the Lipschitz continuity of ∇f we can use the upper bound

$$\left\| \nabla f(z_\lambda) - \nabla f(x) \right\|_2 \leqslant L \cdot \left\| z_\lambda - x \right\|_2 = L \cdot \left\| \lambda(x' - x) \right\|_2 = L\lambda \cdot \left\| x' - x \right\|_2$$

for $\lambda \in [0, 1]$, to conclude

$$\left| f(x') - f(x) - \left\langle \nabla f(x), x' - x \right\rangle \right| \leqslant \int_0^1 L\lambda \cdot \left\| x' - x \right\|_2^2 d\lambda = \frac{L}{2}\|x' - x\|_2^2.$$

Rearranging the inequality finally yields

$$f(x') \leqslant f(x) + \left\langle \nabla f(x), x' - x \right\rangle + \frac{L}{2}\|x' - x\|_2^2.$$

This completes the proof, since x, x' were chosen arbitrary. □

Theorem 2.9. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable with a global minimizer $x_* \in \mathbb{R}^n$, such that the gradient ∇f is Lipschitz continuous with Lipschitz constant $L > 0$, then gradient descent with $\epsilon = 0$ and constant step size $\alpha_k = 1/L$ produces a sequence $(x_k)_{k=0}^{\infty}$, such that*

$$\forall m \in \mathbb{N} : \min_{0 \leqslant k \leqslant m} \big\|\nabla f(x_k)\big\|_2^2 \leqslant \frac{2L\big(f(x_0) - f(x_*)\big)}{m+1}.$$

Proof. Let $k \in \mathbb{N}$ be arbitrary. Since f is differentiable with Lipschitz continuous gradient, we can apply lemma 2.8 to derive

$$f(x_{k+1}) \leqslant f(x_k) + \big\langle \nabla f(x_k), x_{k+1} - x_k \big\rangle + \frac{L}{2}\|x_{k+1} - x_k\|_2^2.$$

By the definition of gradient descent, we can plug in $x_{k+1} - x_k = -1/L \cdot \nabla f(x_k)$ to conclude

$$\begin{aligned} f(x_{k+1}) &\leqslant f(x_k) + \big\langle \nabla f(x_k), -\frac{1}{L}\nabla f(x_k) \big\rangle + \frac{L}{2}\big\|\frac{1}{L}\nabla f(x_k)\big\|_2^2 \\ &= f(x_k) - \frac{1}{L}\big\|\nabla f(x_k)\big\|_2^2 + \frac{1}{2L}\big\|\nabla f(x_k)\big\|_2^2 \\ &= f(x_k) - \frac{1}{2L}\big\|\nabla f(x_k)\big\|_2^2. \end{aligned}$$

Hence the gradient descent algorithm with constant step size $\alpha_k = 1/L$ guarantees to make progress unless $\nabla f(x_k) \neq 0$. The previous expression is equivalent to

$$\big\|\nabla f(x_k)\big\|_2^2 \leqslant 2L\big(f(x_k) - f(x_{k+1})\big).$$

Summing up both sides from $k = 0$ to some $m \in \mathbb{N}$ and taking the average results in

$$\frac{1}{m+1}\sum_{k=0}^{m} \big\|\nabla f(x_k)\big\|_2^2 \leqslant \frac{2L}{m+1}\sum_{k=0}^{m} f(x_k) - f(x_{k+1}) = \frac{2L}{m+1}\big(f(x_0) - f(x_{m+1})\big).$$

Using $f(x_*) \leqslant f(x_k)$ for every $k \in \mathbb{N}_0$, we conclude

$$\min_{0 \leqslant k \leqslant m} \big\|\nabla f(x_k)\big\|_2^2 \leqslant \frac{1}{m+1}\sum_{k=0}^{m} \big\|\nabla f(x_k)\big\|_2^2 \leqslant \frac{2L}{m+1}\big(f(x_0) - f(x_*)\big).$$

This completes the proof, since m was chosen arbitrary. □

The theorem can be found as equation (1.2.15) in the beginning of section 1.2.3 of [3]. Note that in practice it is inefficient to use a constant step size of $1/L$. Nevertheless it is useful for theoretical analysis to guarantee convergence.

Theorem 2.9 states, that in the limit as $n \to \infty$, gradient descent converges to a stationary point $x \in \mathbb{R}^n$ with $\nabla f(x) = 0$. Since we have seen in the proof, that $f(x_k)$ is monotonically decreasing in k, gradient descent converges to the next stationary point in direction of descent, which is either a local minimum, a global minimum or a saddle point.

Next we will define a special class of functions having the nice property, that every stationary point is a global minimum and therefore gradient descent is an excellent tool for minimization.

Definition 2.10. *Let $\mathcal{X}$ be a subset of some real vector space. $\mathcal{X}$ is said to be convex, if*

$$\forall x, x' \in \mathcal{X} : \forall \lambda \in [0,1] : \lambda x + (1-\lambda)x' \in \mathcal{X}.$$

Casually speaking a set is convex if it contains the connection line between any two points in the set. Based on the notion of convex sets we can define convex functions.

Definition 2.11. *Let $\mathcal{X}$ be a convex subset of some real vector space. A function $f : \mathcal{X} \to \mathbb{R}$ is said to be convex, if*

$$\forall x, x' \in \mathcal{X} : \forall \lambda \in [0,1] : f\big(x + \lambda(x'-x)\big) \leqslant f(x) + \lambda\big(f(x') - f(x)\big).$$

For later use we denote the following property of convex functions.

Lemma 2.12. *Let $\mathcal{X}$ be a convex subset of some real vector space. For convex functions $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{X} \to \mathbb{R}$ the sum $f + g$ is convex as well.*

Proof. Let $x, x' \in \mathcal{X}$ and $\lambda \in [0,1]$ be arbitrary, then

$$\begin{aligned}\big(f+g\big)\big(x + \lambda(x'-x)\big) &= f\big(x + \lambda(x'-x)\big) + g\big(x + \lambda(x'-x)\big) \\ &\leqslant f(x) + \lambda\big(f(x') - f(x)\big) + g(x) + \lambda\big(g(x') - g(x)\big) \\ &= (f+g)(x) + \lambda\big((f+g)(x') - (f+g)(x)\big).\end{aligned}$$

This shows the convexity of $f + g$. □

Functions that are convex and continuous differentiable have the nice property, that every stationary point is a global minimum. Mathematically this can be formulated as follows.

Lemma 2.13. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex and continuous differentiable and $x \in \mathbb{R}^n$, then*

$$\nabla f(x) = 0 \;\Leftrightarrow\; \forall x' \in \mathbb{R}^n : f(x) \leqslant f(x').$$

Proof. Let $x \in \mathbb{R}^n$ with $\nabla f(x) = 0$. For arbitrary $x' \in \mathbb{R}^n$ we define the function

$$\phi : \mathbb{R} \to \mathbb{R} : \lambda \mapsto f(x) + \lambda\big(f(x') - f(x)\big) - f\big(x + \lambda(x'-x)\big),$$

which is non-negative on the interval $[0,1]$ by convexity of f. Denoting the derivative as

$$\phi' : \mathbb{R} \to \mathbb{R} : \lambda \mapsto f(x') - f(x) - (x'-x)^{\mathsf{T}} \cdot \nabla f\big(x + \lambda(x'-x)\big)$$

and using the non-negativity of ϕ together with $\phi(0) = 0$, we observe that

$$\phi'(0) = f(x') - f(x) - (x'-x)^{\mathsf{T}} \cdot \nabla f(x) \geqslant 0.$$

Since x was chosen such that $\nabla f(x) = 0$, we can rearrange terms to conclude

$$\forall x' \in \mathbb{R}^n : f(x) \leqslant f(x').$$

This completes the proof, since the backwards direction is trivial.

□

Thus for lower bounded functions $f : \mathbb{R}^n \to \mathbb{R}$, that are convex and differentiable with Lipschitz continuous gradient ∇f, we can guarantee the convergence of gradient descent to a global minimum due to theorem 2.9. Unfortunately this does not apply to the empirical error function used for training neural networks, but we will introduce a theoretical approach to prevent this problem in section 3. Next we will provide further iterative optimization methods.

The gradient descent method is based on the first order Taylor expansion. Intuitively, inclusion of second order information should speed up the minimization process. This idea lead to the famous Newton's method, which requires the objective function to be twice differentiable. Furthermore, to apply Newton's method, we have to compute the Hessian matrix, leading to way more computation effort needed in comparison to gradient descent. In order to reduce this additional effort, one can approximate the inverse Hessian matrix instead of computing it exactly. One popular method, that pursues this approach is the following from [4].

Algorithm 2 Broyden-Fletcher-Goldfarb-Shanno (BFGS)

1: Requires twice-differentiable $f : \mathbb{R}^n \to \mathbb{R}$, random $x_0 \in \mathbb{R}^n$ and $\epsilon \geqslant 0$.

2: Initialize $k \leftarrow 0$, $B_0 \leftarrow I_n$.

3: **while** $\|\nabla f(x_k)\|_2 > \epsilon$ **do**:

4: Choose a step size $\alpha_k > 0$.

5: Set $x_{k+1} \leftarrow x_k - \alpha_k B_k \nabla f(x_k)$.

6: Let $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.

7: Let $\rho_k = (y_k^\mathsf{T} s_k)^{-1}$ and $V_k = I_n - \rho_k y_k s_k^\mathsf{T}$.

8: Set $B_{k+1} \leftarrow V_k^\mathsf{T} B_k V_k + \rho_k s_k s_k^\mathsf{T}$ and $k \leftarrow k + 1$.

9: **end while**

With the initialization $B_0 = I_n$, the method starts with a simple gradient descent update and then iteratively minimizes the objective function by inverse Hessian matrix approximations.

Unfortunately, Hessian matrix computation or Hessian matrix approximation is infeasible for deep neural networks due to the large number of parameters. However, later on we will introduce an approach to train neural networks in tiny subspaces, such that second order methods become applicable.

At this stage we are restricted to training neural networks via gradient descent on the empirical error function. Recalling its definition

$$E : \mathbb{R}^n \to \mathbb{R} : w \mapsto \frac{1}{m} \sum_{i=1}^{m} \ell\big(f(x_i, w), y_i\big) + \lambda \|w\|_2^2,$$

we observe, that the computation effort needed to compute the gradient, depends on the number of training samples $m \in \mathbb{N}$. This is a severe problem for the gradient descent algorithm, since we have to train on a significant amount of data, to optimize the millions of

parameters in deep neural networks. Thus, gradient descent on the empirical error function is computationally and time expensive, that even with heavy computing power it can be unfeasible to find good solutions in a reasonable period of time. However, there exist several other optimization methods, that apply the same concept of minimizing a function along the slope of its surface. These algorithms make a trade-off between the accuracy and the time needed for gradient computations. In the following we will introduce two of them.

Stochastic gradient descent is a variation of gradient descent, that aims to approximate the gradient of the objective function instead of computing it exactly. The basic idea is to compute the gradient on a random fraction of the data set and use this as an estimation of the gradient on the whole data set. In each iteration $k \in \mathbb{N}$ one randomly draws a subset $\mathcal{I}_k \subset \{1, \dots, m\}$ of size $|\mathcal{I}_k| = b \in \mathbb{N}$ and uses

$$\nabla E(w_k, \mathcal{I}_k) \coloneqq \frac{1}{b} \sum_{i \in \mathcal{I}_k} \ell\big(f(x_i, w_k), y_i\big) + \lambda \|w_k\|_2^2$$

as an approximation of $\nabla E(w_k)$. Although only a small amount of data is used for one single parameter update, during training most of the data will be used to fit the model due to the large number of iterative updates. Formally stochastic gradient descent proceeds as follows.

Algorithm 3 Stochastic Gradient Descent (SGD)

1: Requires differentiable empirical error function $E : \mathbb{R}^n \to \mathbb{R}$ and random $w_0 \in \mathbb{R}^n$.

2: Requires a batch size $b \in \mathbb{N}$ and a number of epochs $K \in \mathbb{N}$.

3: **for** $k = 1, \dots, \frac{Km}{b}$ **do**:

4: Draw a random subset $\mathcal{I}_k \subset \{1, \dots, m\}$ of size $|\mathcal{I}_k| = b$.

5: Choose a step size $\alpha_k > 0$.

6: Set $w_{k+1} \leftarrow w_k - \alpha_k \cdot \nabla E(w_k, \mathcal{I}_k)$.

7: **end for**

In here, an epoch describes the use of a number of samples for parameter updates, that equals the size of the whole training dataset. Thus on average each individual training sample will contribute K times to the optimization process. In comparison to this, the classical gradient descent method uses the data of one epoch per iteration. A theoretical convergence analysis of stochastic gradient descent can be found in [5], but would exceed the scope of this thesis.

So far we have not discussed how to choose the step sizes α_k. To simplify things, in practice one usually chooses a step size $\alpha > 0$, that stays constant across each iteration. Unfortunately a constant step size does not adapt to the specific local properties of the error surface at iteration $k \in \mathbb{N}$. One method that aims to improve the stochastic gradient descent algorithm by including past gradients for the current parameter update is called adaptive moment estimation and was first raised in [6].

Algorithm 4 Adaptive Moment Estimation (Adam)

Requires differentiable empirical error function $E : \mathbb{R}^n \to \mathbb{R}$ and random $w_0 \in \mathbb{R}^n$.
Requires constant step size $\alpha > 0 \in \mathbb{R}$, batch size $b \in \mathbb{N}$ and number of epochs $K \in \mathbb{N}$.
Requires momentum factors $\beta_1, \beta_2 \in [0, 1)$ and a scalar $\epsilon > 0$.
for $k = 1, \ldots, \frac{Km}{b}$ **do**:
    Draw a random subset $\mathcal{I}_k \subset \{1, \ldots, m\}$ of size $|\mathcal{I}_k| = b$.
    Set $m_k \leftarrow \beta_1 m_{k-1} + (1 - \beta_1) \cdot \nabla E(w_k, \mathcal{I}_k)$ and let $\hat{m}_k \coloneqq \frac{m_k}{1-\beta_1^k}$.
    Set $v_k \leftarrow \beta_2 v_{k-1} + (1 - \beta_2) \cdot \nabla E(w_k, \mathcal{I}_k)^2$ and let $\hat{v}_k \coloneqq \frac{v_k}{1-\beta_2^k}$.
    Set $w_{k+1} \leftarrow w_k - \alpha \cdot \frac{\hat{m}_k}{\sqrt{\hat{v}_k}+\epsilon}$.
end for

In here, all operations on vectors are performed element-wise. The scalar ϵ is used to prevent division by zero. The default setup proposed by the authors is to choose $\epsilon = 10^{-8}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We will always apply this default setting when working with Adam.

This completes the necessary basics on notation and training of neural networks. Next we will investigate certain approaches on how to improve the training of neural networks in order prevent the curse of non-convexity and reduce the computation and time effort needed.

To Do: Cite 2.8 and 2.9
Exploration: Further second order method
Optional: Backpropagation

b : bias vector
w : weight vector
-
b : batch size
$d_{\mathcal{X}}$: metric on input space
$d_{\mathcal{Y}}$: metric on output space
f : real valued function
g : real valued function
$\tilde{f}$: neural network architecture
i : several indices
k : iteration index
l : number of layers in neural network
ℓ : loss function
m : real number / number of training samples
m_k : vector in Adam
n_0 : input dimension of neural network
n_l : output dimension of neural network
n_x : input dimension of layer
n_y : output dimension of layer
n : number of parameters in neural network
s : real valued vector in BFGS
v_k : vector in Adam
w : parameter vector in neural network
x : real valued vector (input)
x' : real valued vector (input)
y : real valued vector (output)
z_λ : real valued vector
-
α : step size
β_1 : momentum factor
β_2 : momentum factor
ϵ : convergence criterium
λ : real valued scalar / regularization parameter
ν : neuron
ϕ : real valued helper function
ρ : real value in BFGS
σ : activation function
-
B : inverse hessian matrix approximation
E : empirical error
I_n : identity matrix
K : number of epochs
L : lipschitz constant
R : generalization error
V : matrix in BFGS
W : weight matrix in layer

-
$\tilde{\mathcal{F}}$: set of neural networks with given architecture
$\mathcal{D}$: probability distribution
$\mathcal{D}_x$: marginal probability distribution
$\mathcal{I}$: set of indices $\mathcal{X}$: input space
$\mathcal{Y}$: output space
-
$\mathbb{E}$: expected value
$\mathbb{N}$: natural numbers
$\mathbb{R}$: real numbers

3 Kernel Gradient Descent

The goal of this section is to introduce and investigate the neural tangent kernel, which was analyzed by Arthur Jacot, Franck Gabriel and Clément Holger in [7]. We start by motivating and giving the definition of functional derivatives and functional gradient descent. This leads to the definition of multi-dimensional kernels, which enable us to generalize the functional gradient descent to so called kernel gradient descent. Having this concept in mind, we will prove that the training of neural networks in the parameter space is linked to kernel gradient descent in the function space with respect to a special kernel, the neural tangent kernel.

3.1 Functional Derivatives

As seen in section 2.2, the training of neural networks via parameter optimization is usually accomplished by minimizing the empirical error

$$E : \mathbb{R}^n \to \mathbb{R} : w \mapsto \frac{1}{m} \sum_{i=1}^{m} \ell\big(f(x_i, w), y_i\big) + \lambda \|w\|^2$$

on labeled training data $(x_i, y_i) \in \mathbb{R}^d \times \mathbb{R}^p$ for $i = 1, \dots, m$. Due to the non-convexity of the error function E it is in general hard to find a global minimum. However, we will see, that this problem can be prevented by moving from the parameter space $\mathbb{R}^n$ to the function space

$$\mathcal{F} \coloneqq \Big\{ f : \mathbb{R}^d \to \mathbb{R}^p \Big\}.$$

That is we are no longer optimizing the specific parameters of the unknown target function but rather the function itself. This requires the definition of a cost

$$C : \mathcal{F} \to \mathbb{R} : f \mapsto \frac{1}{m} \sum_{i=1}^{m} \ell\big(f(x_i), y_i\big),$$

associating some cost to each element of the function space $\mathcal{F}$. In this notion, we can write

$$E = C \circ (w \mapsto f(x, w)).$$

In contrast to the parameter optimization approach, the choice of a convex loss function ℓ provides a convex cost C, since by lemma 2.12 the sum of convex functions is convex again. We will introduce the functional gradient descent method, that enables us to minimize C in the function space $\mathcal{F}$ and is an analog to the gradient descent method in the parameter space $\mathbb{R}^n$. Since C is convex in the function space, convergence to a global minimum will be guaranteed in contrast to the non-convex parameter optimization problem.

Note that this approach is great for theoretical analysis due to the advantage of convexity. In practice however, we are still in need of the explicit parameters $w \in \mathbb{R}^n$, that describe the objective function. Nevertheless we can use the analysis of functional gradient descent to get a better understanding of gradient descent in the function space.

Let us start with a slight reformulation of well known results from functional analysis, that will enable us to define the functional derivative of the cost C.

Definition 3.1. *Given a real vector space $\mathcal{V}$, we define the dual space*

$$\mathcal{V}^* \coloneqq \Big\{\mu : \mathcal{V} \to \mathbb{R} \mid \mu \text{ linear and continuous}\Big\}.$$

The elements of $\mathcal{V}^*$ are referred to as linear functionals.

Until now it is unclear how all of the elements in $\mathcal{F}^*$ arise. We will see that they can be constructed by the elements of $\mathcal{F}$. This procedure is based on the following lemma.

Lemma 3.2. *Given a fixed probability distribution $\mathcal{D}_x$ on the input space $\mathbb{R}^d$, one can equip the function space $\mathcal{F}$ with the seminorm*

$$\|\cdot\|_{\mathcal{D}_x} : \mathcal{F} \to \mathbb{R} : f \mapsto \sqrt{\langle f, f\rangle_{\mathcal{D}_x}}$$

in terms of the symmetric positive semidefinite bilinear form

$$\langle\cdot,\cdot\rangle_{\mathcal{D}_x} : \mathcal{F} \times \mathcal{F} \to \mathbb{R} : (f, g) \mapsto \langle f, g\rangle_{\mathcal{D}_x} \coloneqq \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} g(x)\Big].$$

Proof. Let $f_1, f_2, f, g \in \mathcal{F}$ and $\lambda \in \mathbb{R}$. $\langle\cdot,\cdot\rangle_{\mathcal{D}_x}$ is indeed symmetric and bilinear since it suffices

(i) $\langle f_1 + f_2, g\rangle_{\mathcal{D}_x} = \langle f_1, g\rangle_{\mathcal{D}_x} + \langle f_2, g\rangle_{\mathcal{D}_x}$:

$$\begin{aligned}
\langle f_1 + f_2, g\rangle_{\mathcal{D}_x} &= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[\big(f_1(x) + f_2(x)\big)^\mathsf{T} g(x)\Big] \\
&= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[f_1(x)^\mathsf{T} g(x) + f_2(x)^\mathsf{T} g(x)\Big] \\
&= \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[f_1(x)^\mathsf{T} g(x)\Big] + \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[f_2(x)^\mathsf{T} g(x)\Big] \\
&= \langle f_1, g\rangle_{\mathcal{D}_x} + \langle f_2, g\rangle_{\mathcal{D}_x},
\end{aligned}$$

(ii) $\langle \lambda f, g\rangle_{\mathcal{D}_x} = \lambda\langle f, g\rangle_{\mathcal{D}_x}$:

$$\langle \lambda f, g\rangle_{\mathcal{D}_x} = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[\lambda f(x)^\mathsf{T} g(x)\Big] = \lambda \cdot \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} g(x)\Big] = \lambda\langle f, g\rangle_{\mathcal{D}_x},$$

(iii) $\langle f, g\rangle_{\mathcal{D}_x} = \langle g, f\rangle_{\mathcal{D}_x}$:

$$\langle f, g\rangle_{\mathcal{D}_x} = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} g(x)\Big] = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[g(x)^\mathsf{T} f(x)\Big] = \langle g, f\rangle_{\mathcal{D}_x}.$$

Furthermore $\langle\cdot,\cdot\rangle_{\mathcal{D}_x}$ is positive semidefinite, since for any $f \in \mathcal{F}$ it holds, that

$$\langle f, f\rangle_{\mathcal{D}_x} = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} f(x)\Big] = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[\sum_{i=1}^{d} f_i(x)^2\Big] \geqslant 0.$$

Thus $\|\cdot\|_{\mathcal{D}_x} = \sqrt{\langle\cdot,\cdot\rangle_{\mathcal{D}_x}}$ is a seminorm on the function space $\mathcal{F}$. □

Lemma 3.3. *The seminorm $\|\cdot\|_{\mathcal{D}_x}$ induces the pseudometric*

$$d : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_+ : (f, g) \mapsto \|f - g\|_{\mathcal{D}_x},$$

such that $(\mathcal{F}, d)$ is a complete pseudometric space.

Proof. Let $f, g, h \in \mathcal{F}$. d is indeed a pseudometric since it suffices

(i) $d(f,f) = 0$:

$$d(f,f) = \|f - f\|_{\mathcal{D}_x} = 0,$$

(ii) $d(f,g) = d(g,f)$:

$$d(f,g) = \|f - g\|_{\mathcal{D}_x} = \|g - f\|_{\mathcal{D}_x} = d(g,f),$$

(iii) $d(f,h) \leqslant d(f,g) + d(g,h)$:

$$\begin{aligned} d(f,h) &= \|f - h\|_{\mathcal{D}_x} \\ &= \|f - g + g - h\|_{\mathcal{D}_x} \\ &\leqslant \|f - g\|_{\mathcal{D}_x} + \|g - h\|_{\mathcal{D}_x} = d(f,g) + d(g,h). \end{aligned}$$

The completeness of $\mathcal{F}$ arises from the fact, that every Cauchy sequence in $\mathbb{R}$ converges. □

Based on the completeness of $\mathcal{F}$ one can introduce the concept of orthogonal projections.

Lemma 3.4. *Let $\mathcal{A} \subset \mathcal{F}$ be non-empty, closed and convex. For every $f \in \mathcal{F}$ there exists some $a^* \in \mathcal{A}$ such that*

$$\|f - a^*\|_{\mathcal{D}_x}^2 = \inf_{a \in \mathcal{A}} \|f - a\|_{\mathcal{D}_x}^2.$$

Such an element a^* is called orthogonal projection of f on $\mathcal{A}$.

Proof. Let $f \in \mathcal{F}$. There exists a sequence $(a_n)_{n\in\mathbb{N}} \subset \mathcal{A}$ such that

$$\lim_{n\to\infty} \|f - a_n\|_{\mathcal{D}_x}^2 = \inf_{a \in \mathcal{A}} \|f - a\|_{\mathcal{D}_x}^2 =: \delta.$$

For $n, m \in \mathbb{N}$ we can derive

$$\begin{aligned} \|a_n - a_m\|_{\mathcal{D}_x}^2 &= \|(f - a_n) - (f - a_m)\|_{\mathcal{D}_x}^2 \\ &= \|f - a_n\|_{\mathcal{D}_x}^2 + \|f - a_m\|_{\mathcal{D}_x}^2 - 2\big\langle f - a_n, f - a_m \big\rangle_{\mathcal{D}_x}. \end{aligned}$$

Furthermore we have

$$\|(f - a_n) + (f - a_m)\|_{\mathcal{D}_x}^2 = \|f - a_n\|_{\mathcal{D}_x}^2 + \|f - a_m\|_{\mathcal{D}_x}^2 + 2\big\langle f - a_n, f - a_m \big\rangle_{\mathcal{D}_x}.$$

Combining both expressions yields

$$\begin{aligned} \|a_n - a_m\|_{\mathcal{D}_x}^2 &= 2\|f - a_n\|_{\mathcal{D}_x}^2 + 2\|f - a_m\|_{\mathcal{D}_x}^2 - \|(f - a_n) + (f - a_m)\|_{\mathcal{D}_x}^2 \\ &= 2\|f - a_n\|_{\mathcal{D}_x}^2 + 2\|f - a_m\|_{\mathcal{D}_x}^2 - 4\|f - \frac{1}{2}(a_n + a_m)\|_{\mathcal{D}_x}^2. \end{aligned}$$

Since $\mathcal{A}$ is convex, $\frac{1}{2}(a_n + a_m) \in \mathcal{A}$, such that for arbitrary $\epsilon > 0$ there exists $N \in \mathbb{N}$ with

$$\|a_n - a_m\|_{\mathcal{D}_x}^2 \leqslant 2(\delta + \epsilon) + 2(\delta + \epsilon) - 4\delta = 4\epsilon$$

for any $n, m > N$. Thus $(a_n)_{n\in\mathbb{N}}$ is a Cauchy sequence. Since $\mathcal{F}$ is complete and $\mathcal{A}$ is closed, $(a_n)_{n\in\mathbb{N}}$ converges to some $a^* \in \mathcal{A}$ with $\|f - a^*\|_{\mathcal{D}_x}^2 = \delta$. □

Lemma 3.4 is a weakening of the Hilbert projection theorem stating that, given a Hilbert space, it even exists a unique orthogonal projection. In our case we lose the uniqueness, beacuse $\mathcal{F}$ is only a pseudometric space and not a metric space. However this is just a side note, since the existence of an orthogonal projection suffices for the further theory.

Corollary 3.5. *Let $f \in \mathcal{F}$ and $\mathcal{A} \subset \mathcal{F}$ be a non-empty and closed subspace. For every $a \in \mathcal{A}$ it holds, that $\langle f - a^*, a\rangle_{\mathcal{D}_x} = 0$, where $a^* \in \mathcal{A}$ denotes an orthogonal projection of f on $\mathcal{A}$.*

Proof. Let $a \in \mathcal{A}$ and $\lambda \in \mathbb{R}$. Since $\mathcal{A}$ is a subspace we can infer $a^* + \lambda a \in \mathcal{A}$. Thus we derive

$$\big\|(a^* + \lambda a) - f\big\|_{\mathcal{D}_x}^2 \geqslant \big\|a^* - f\big\|_{\mathcal{D}_x}^2$$

by the minimality of a^*. Rearranging the inequality yields

$$\big\|(a^* - f) + \lambda a\big\|_{\mathcal{D}_x}^2 - \big\|a^* - f\big\|_{\mathcal{D}_x}^2 = 2\big\langle a^* - f, \lambda a\big\rangle_{\mathcal{D}_x} + \big\|\lambda a\big\|_{\mathcal{D}_x}^2 \geqslant 0,$$

giving rise to the definition of the non-negative function

$$\phi : \mathbb{R} \to \mathbb{R} : \lambda \mapsto 2\lambda\big\langle a^* - f, a\big\rangle_{\mathcal{D}_x} + \lambda^2\big\|a\big\|_{\mathcal{D}_x}^2.$$

Under the assumption that $\big\langle a^* - f, a\big\rangle_{\mathcal{D}_x} \neq 0$ and $\|a\|_{\mathcal{D}_x}^2 \neq 0$, we derive

$$\phi\Big(-\frac{\big\langle a^* - f, a\big\rangle_{\mathcal{D}_x}}{\|a\|_{\mathcal{D}_x}^2}\Big) = -2 \cdot \frac{\big\langle a^* - f, a\big\rangle_{\mathcal{D}_x}^2}{\|a\|_{\mathcal{D}_x}^2} + \frac{\big\langle a^* - f, a\big\rangle_{\mathcal{D}_x}^2}{\|a\|_{\mathcal{D}_x}^2} = -\frac{\big\langle a^* - f, a\big\rangle_{\mathcal{D}_x}^2}{\|a\|_{\mathcal{D}_x}^2} < 0$$

and under the assumption that $\big\langle a^* - f, a\big\rangle_{\mathcal{D}_x} \neq 0$ and $\|a\|_{\mathcal{D}_x}^2 = 0$, we derive

$$\phi\Big(-\big\langle a^* - f, a\big\rangle_{\mathcal{D}_x}\Big) = -2 \cdot \big\langle a^* - f, a\big\rangle_{\mathcal{D}_x}^2 < 0,$$

which is a contradiction to $\phi(\lambda) \geqslant 0$ for every $\lambda \in \mathbb{R}$. Thus $\big\langle f - a^*, a\big\rangle_{\mathcal{D}_x} = 0$. □

Finally, the concept of orthogonal projections enables us to represent any functional $\mu \in \mathcal{F}^*$ based on a representer $d \in \mathcal{F}$.

Lemma 3.6. *The map*

$$J : \mathcal{F} \to \mathcal{F}^* : d \mapsto \langle d, \cdot\rangle_{\mathcal{D}_x}$$

is linear and surjective.

Proof. The linearity follows directly from the linearity of $\langle \cdot, \cdot\rangle_{\mathcal{D}_x}$. To prove that J is indeed surjective, we let $\mu \in \mathcal{F}^*$ and need to find some $d \in \mathcal{F}$ such that $\mu = J(d)$.

Case $\mu = 0$: We can choose $d = 0$, with $\mu = 0 = J(d)$.

Case $\mu \neq 0$: Due to the linearity of μ we can normalize and find some $e \in \mathcal{F}$ with $\mu[e] = 1$. Let $N \coloneqq \mu^{-1}(0)$ denote the kernel of μ, which is non-empty, closed and convex since μ is linear and continuous. Thus by Lemma 3.4 there exists some $p_e \in N$, such that p_e is an orthogonal projection of e on N. Define $f \coloneqq e - p_e$, then $f \notin N$ since

$$\mu[f] = \mu[e] - \mu[p_e] = \mu[e] = 1.$$

Based on the linearity of μ, we can derive for arbitrary $g \in \mathcal{F}$, that

$$\mu\big[g - \mu[g]f\big] = \mu[g] - \mu[g]\mu[f] = \mu[g] - \mu[g] \cdot 1 = 0,$$

which implies $h \coloneqq g - \mu[g]f \in N$. Thus $\langle f, h\rangle_{\mathcal{D}_x} = 0$ by corollary 3.5 and we conclude

$$\langle f, g\rangle_{\mathcal{D}_x} = \langle f, h\rangle_{\mathcal{D}_x} + \langle f, \mu[g]f\rangle_{\mathcal{D}_x} = 0 + \mu[g] \cdot \langle f, f\rangle_{\mathcal{D}_x} = \mu[g] \cdot \|f\|^2_{\mathcal{D}_x}.$$

Now we can assume that $\|f\|^2_{\mathcal{D}_x} \neq 0$. Otherwise we would have $\langle f, g\rangle_{\mathcal{D}_x} = 0$ for every $g \in \mathcal{F}$ and therefore $f = 0$ which is a contradiction to $f \notin N$. Rearranging terms results in

$$\mu[g] = \big\langle f/\|f\|^2_{\mathcal{D}_x}, g\big\rangle_{\mathcal{D}_x} = J\big(f/\|f\|^2_{\mathcal{D}_x}\big)[g].$$

Hence we have found $d = f/\|f\|^2_{\mathcal{D}_x}$ with $\mu = J(d)$. This shows surjectivity. □

Thus for each functional $\mu \in \mathcal{F}^*$ there exists $d \in \mathcal{F}$ such that $\mu = \langle d, \cdot\rangle_{\mathcal{D}_x}$. In other words

$$\mathcal{F}^* = \Big\{\mu : \mathcal{F} \to \mathbb{R} : f \mapsto \langle d, f\rangle_{\mathcal{D}_x} \mid d \in \mathcal{F}\Big\}.$$

Lemma 3.6 is a weakening of the Riesz representation theorem stating that any Hilbert space $\mathcal{H}$ is isometric and isomorphic to its dual space $\mathcal{H}^*$ via the map $J : \mathcal{H} \to \mathcal{H}^* : x \mapsto \langle x, \cdot\rangle_{\mathcal{H}}$.

Since $\langle \cdot, \cdot\rangle_{\mathcal{D}_x}$ is only positive semidefinite and not positive definite we lose the property of J being isometric and therefore injective. Again, this is just a side note, since surjectivity will be sufficient enough for the further theory. Note that, based on the definition of $\mathcal{D}_x$, J maps two elements of $\mathcal{F}$ to the same functional, if and only if they are equal on the data.

The representation theorem has great importance for the functional gradient descent method. Another crucial ingredient is the definition of functional derivatives and functional gradient.

Definition 3.7. *Let $\phi, f_0 \in \mathcal{F}$. A functional $\mu : \mathcal{F} \to \mathbb{R}$ is said to be differentiable at the point f_0 in direction ϕ, if*

$$\int \frac{\partial \mu}{\partial f_0(x)}(x)\phi(x)\, dx = \lim_{\epsilon \to 0} \frac{\mu[f_0 + \epsilon\phi] - \mu[f_0]}{\epsilon}$$

exists. The functional gradient of μ at f_0 is defined as

$$\nabla\mu|_{f_0} : \mathbb{R}^d \to \mathbb{R}^p : x \mapsto \frac{\partial \mu}{\partial f_0(x)}(x).$$

If the limit exists for every $\phi \in \mathcal{F}$ we define the functional derivative of μ at f_0 as

$$\partial_f \mu|_{f_0} : \mathcal{F} \to \mathbb{R} : \phi \mapsto \frac{\partial}{\partial \epsilon}\Big[\mu[f_0 + \epsilon\phi]\Big]_{\epsilon=0}.$$

Based on these definitions, we can iteratively minimize functionals $\mu : \mathcal{F} \to \mathbb{R}$ in a similar manner to the gradient descent method.
With a slight change in notation, the same argumentation that was used for convergence of gradient descent on functions, can be applied to the functional gradient descent method.

Let $\mu : \mathcal{F} \to \mathbb{R}$ be differentiable. Functional gradient descent proceeds as follows:

1) Initialize $k = 0$, choose $\epsilon \geqslant 0 \in \mathbb{R}$ and pick some random $f_0 \in \mathcal{F}$.

2) WHILE $\left\|\nabla\mu|_{f_k}\right\|_{\mathcal{D}_x} > \epsilon$ DO:

3) Choose a step size $\alpha_k > 0 \in \mathbb{R}$.

4) Set $f_{k+1} = f_k - \alpha_k \cdot \nabla\mu|_{f_k}$ and $k = k + 1$.

Until now we have a theoretical understanding of functional gradient descent in the function space $\mathcal{F}$. However there is not yet a connection to gradient descent in the parameter space. In theory, functional gradient descent is an excellent tool to guarantee convergence but in practice we need the explicit parameter vector $w \in \mathbb{R}^n$ to represent the optimal function due to functional gradient descent. This relationship between functional gradient descent and parameter gradient descent can be established by so called kernel gradient descent.

3.2 Kernel Gradient

We start by recalling the basic definition of one-dimensional kernels.

Definition 3.8. *Let X be an arbitrary set. A one-dimensional kernel over X is a map*

$$K : X \times X \to \mathbb{R} : (x, x') \mapsto K(x, x').$$

A kernel K is said to be positive definite symmetric if for any $m \in \mathbb{N}$ and $x \in X^m$, the matrix with entries $M_{i,j} = K(x_i, x_j)$ for $i, j = 1, \ldots, m$ is symmetric and positive semidefinite.

This thesis will work with kernels over the input space $\mathbb{R}^d$, not only with one-dimensional but with multi-dimensional kernels, which motivates the following definition.

Definition 3.9. *A multi-dimensional kernel over $\mathbb{R}^d$ is a function*

$$K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{p\times p} : (x, x') \mapsto K(x, x'),$$

such that $K(x, x') = K(x', x)^\mathsf{T}$ for any $x, x' \in \mathbb{R}^d$.

Based on the marginal distribution $\mathcal{D}_x$, kernels induce symmetric bilinear forms.

Lemma 3.10. *A kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{p\times p}$ induces the symmetric bilinear form*

$$\langle\cdot,\cdot\rangle_K : \mathcal{F} \times \mathcal{F} \to \mathbb{R} : (f, g) \mapsto \langle f, g\rangle_K \coloneqq \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} K(x, x')g(x')\Big],$$

where $x, x' \in \mathbb{R}^d$ are drawn independently according to $\mathcal{D}_x$.

Proof. Let $f_1, f_2, f, g \in \mathcal{F}$ and $\lambda \in \mathbb{R}$. $\langle\cdot,\cdot\rangle_K$ is indeed symmetric and bilinear since it suffices

(i) $\langle f_1 + f_2, g\rangle_K = \langle f_1, g\rangle_K + \langle f_2, g\rangle_K$:

$$\begin{aligned}\langle f_1 + f_2, g\rangle_K &= \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[\big(f_1(x) + f_2(x)\big)^\mathsf{T} K(x,x')g(x')\Big] \\ &= \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[f_1(x)^\mathsf{T} K(x,x')g(x') + f_2(x)^\mathsf{T} K(x,x')g(x')\Big] \\ &= \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[f_1(x)^\mathsf{T} K(x,x')g(x')\Big] + \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[f_2(x)^\mathsf{T} K(x,x')g(x')\Big] \\ &= \langle f_1, g\rangle_K + \langle f_2, g\rangle_K,\end{aligned}$$

(ii) $\langle \lambda f, g\rangle_K = \lambda\langle f, g\rangle_K$:

$$\begin{aligned}\langle \lambda f, g\rangle_K &= \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[\lambda f(x)^\mathsf{T} K(x,x')g(x')\Big] \\ &= \lambda \cdot \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} K(x,x')g(x')\Big] = \lambda\langle f, g\rangle_K,\end{aligned}$$

(iii) $\langle f, g\rangle_K = \langle g, f\rangle_K$:

$$\begin{aligned}\langle f, g\rangle_K &= \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} K(x,x')g(x')\Big] \\ &= \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} K(x',x)^\mathsf{T} g(x')\Big] \\ &= \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[g(x')^\mathsf{T} K(x',x)f(x)\Big] = \langle g, f\rangle_K.\end{aligned}$$

In contrast to $\langle\cdot,\cdot\rangle_{\mathcal{D}_x}$, the bilinear map $\langle\cdot,\cdot\rangle_K$ is not necessarily positive semidefinite. □

Using these bilinear forms, one can generalize the definition of positive definite kernels.

Definition 3.11. *A kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{p\times p}$ is called positive definite with respect to the seminorm $\|\cdot\|_{\mathcal{D}_x}$ if for every $f \in \mathcal{F}$ it holds, that $\|f\|_{\mathcal{D}_x} > 0$ implies $\langle f, f\rangle_K > 0$.*

We can use positive definite kernels to equip $\mathcal{F}$ with another seminorm.

Lemma 3.12. *Given a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{p\times p}$, that is positive definite with respect to the seminorm $\|\cdot\|_{\mathcal{D}_x}$ we can use the bilinear form $\langle\cdot,\cdot\rangle_K$ to define another seminorm*

$$\|\cdot\|_K : \mathcal{F} \to \mathbb{R} : f \mapsto \sqrt{\langle\cdot,\cdot\rangle_K}.$$

Proof. Since $\langle\cdot,\cdot\rangle_K$ is symmetric by lemma 3.10 it remains to show that $\langle\cdot,\cdot\rangle_K$ is positive semidefinite. For this we let $f \in \mathcal{F}$ be arbitrary and investigate the following cases.

Case $\|f\|_{\mathcal{D}_x} > 0$: Since K is positive definite with respect to $\|\cdot\|_{\mathcal{D}_x}$, we derive $\langle f, f\rangle_K > 0$.

Case $\|f\|_{\mathcal{D}_x} = 0$: By the definition of $\langle\cdot,\cdot\rangle_{\mathcal{D}_x}$, we derive

$$\|f\|_{\mathcal{D}_x}^2 = \langle f, f\rangle_{\mathcal{D}_x} = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} f(x)\Big] = \mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[\sum_{i=1}^{d} f_i(x)^2\Big] = 0.$$

Thus for every $x \sim \mathcal{D}_x$ it holds that $f(x) = 0$. Therefore we conclude

$$\langle f, f\rangle_K = \mathop{\mathbb{E}}_{x,x'\sim\mathcal{D}_x}\Big[f(x)^\mathsf{T} K(x,x')f(x')\Big] = 0.$$

In total we have shown that $\langle f, f\rangle_K > 0$ for every $f \in \mathcal{F}$, such that $\|\cdot\|_K$ is a seminorm. □

This concludes the necessary knowledge on kernels. Next we will introduce the kernel gradient of a functional, which is based on the partial application of kernels.

Definition 3.13. *Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{p\times p}$ be a kernel. For $j = 1, \dots, p$ one defines the partial application of K as the function*

$$K_{\cdot,j} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^p : (x, x') \mapsto K(x, x')_{\cdot,j},$$

where $K(x, x')_{\cdot,j}$ denotes the j-th column of the $p \times p$ matrix $K(x, x')$.

Fixing one argument of $K_{\cdot,j}$ results in a function in $\mathcal{F}$. This enables the following definition.

Definition 3.14. *Given a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{p\times p}$, we define the map*

$$\Phi_K : \mathcal{F}^* \to \mathcal{F} : \mu \mapsto f_\mu,$$

mapping a dual element $\mu \in \mathcal{F}$ to the function

$$f_\mu : \mathbb{R}^d \to \mathbb{R}^p : x \mapsto \begin{bmatrix} \mu[K_{\cdot,1}(\cdot, x)] \\ \vdots \\ \mu[K_{\cdot,p}(\cdot, x)] \end{bmatrix}.$$

Based on the definition of Φ_K we can define the kernel gradient of a functional.

Definition 3.15. *Let $f_0 \in \mathcal{F}$, $\mu : \mathcal{F} \to \mathbb{R}$ be a differentiable functional with $\partial_f \mu|_{f_0} \in \mathcal{F}^*$ and K be a kernel. The kernel gradient of μ at f_0 with respect to K is defined as*

$$\nabla_K \mu|_{f_0} : \mathbb{R}^d \to \mathbb{R}^p : x \mapsto \Phi_K\Big(\partial_f \mu|_{f_0}\Big)(x).$$

Using this definition, we can formulate the kernel gradient descent method.

Let K be a kernel and $\mu : \mathcal{F} \to \mathbb{R}$ be a differentiable functional with $\partial_f \mu|_{f_0} \in \mathcal{F}^*$. Kernel gradient descent with respect to the kernel K proceeds as follows:

1) Initialize $k = 0$, choose $\epsilon \geqslant 0 \in \mathbb{R}$ and pick some random $f_0 \in \mathcal{F}$.

2) WHILE $\left\|\nabla_K \mu|_{f_k}\right\|_{\mathcal{D}_x} > \epsilon$ DO:

   3) Choose a step size $\alpha_k > 0 \in \mathbb{R}$.

   4) Set $f_{k+1} = f_k - \alpha_k \cdot \nabla_K \mu|_{f_k}$ and $k = k + 1$.

In the following, we will apply kernel gradient descent to minimize the cost C. During this we are especially interested in the evolution of the parameters defining the sequence $(f_k)_{k\in\mathbb{N}_0}$.

With the choice of a differentiable loss function, the cost functional $C : \mathcal{F} \to \mathbb{R}$ is differentiable as well, such that the functional derivative $\partial_f C|_{f_0}$ is well-defined for every $f_0 \in \mathcal{F}$.

In the following we will make the restriction, that the marginal distribution $\mathcal{D}_x$ on the input space $\mathbb{R}^d$ is given by the empirical distribution on a finite subset of $\mathbb{R}^d$. This means there exist $m \in \mathbb{N}$ and $x_1, \dots, x_m \in \mathbb{R}^d$ such that

$$\mathcal{D}_x = \frac{1}{m} \sum_{i=1}^{m} \delta_{x_i},$$

where δ_{x_i} denotes the Dirac measure in x_i for $i = 1, \dots, m$. Under this assumption, the cost C depends only on the values of f at a finite dataset. Thus we conclude, that $\partial_f C|_{f_0}$ is linear and continuous and therefore $\partial_f C|_{f_0} \in \mathcal{F}^*$ for every $f_0 \in \mathcal{F}$. Hence by Lemma 3.6 there exists a representer $d|_{f_0} \in \mathcal{F}$, such that the functional derivative of C can be written as

$$\partial_f C|_{f_0} = \big\langle d|_{f_0}, \cdot \big\rangle_{\mathcal{D}_x}.$$

Next we will use this restriction on the distribution $\mathcal{D}_x$, to simplify the notation of Φ_K.

Lemma 3.16. *Let $\mu = \langle d, \cdot \rangle_{\mathcal{D}_x} \in \mathcal{F}^*$ and $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{p \times p}$ be a kernel, then*

$$\Phi_K(\mu) = \frac{1}{m} \sum_{i=1}^{m} K(\cdot, x_i) d(x_i),$$

where $x_1, \dots, x_m \in \mathbb{R}^d$ denotes the finite support of the empiricial distribution $\mathcal{D}_x$.

Proof. Let $x \in \mathbb{R}^d$ be arbitrary. By the definition of Φ_K we have

$$\Phi_K\big(\mu\big)(x) = \Phi_K\Big(\langle d, \cdot \rangle_{\mathcal{D}_x}\Big)(x) = \begin{bmatrix} \mu[K_{\cdot,1}(\cdot, x)] \\ \vdots \\ \mu[K_{\cdot,p}(\cdot, x)] \end{bmatrix} = \begin{bmatrix} \big\langle d, K_{\cdot,1}(\cdot, x) \big\rangle_{\mathcal{D}_x} \\ \vdots \\ \big\langle d, K_{\cdot,p}(\cdot, x) \big\rangle_{\mathcal{D}_x} \end{bmatrix}.$$

Recalling the definition of $\langle \cdot, \cdot \rangle_{\mathcal{D}_x}$, it holds that

$$\big\langle d, K_{\cdot,j}(\cdot, x) \big\rangle_{\mathcal{D}_x} = \mathop{\mathbb{E}}_{x' \sim \mathcal{D}_x} \Big[d(x')^\mathsf{T} K_{\cdot,j}(x', x) \Big]$$

for $j = 1, \dots, p$. Thus we derive

$$\Phi_K\big(\mu\big)(x) = \begin{bmatrix} \big\langle d, K_{\cdot,1}(\cdot, x) \big\rangle_{\mathcal{D}_x} \\ \vdots \\ \big\langle d, K_{\cdot,p}(\cdot, x) \big\rangle_{\mathcal{D}_x} \end{bmatrix} = \mathop{\mathbb{E}}_{x' \sim \mathcal{D}_x} \begin{bmatrix} d(x')^\mathsf{T} K_{\cdot,1}(x', x) \\ \vdots \\ d(x')^\mathsf{T} K_{\cdot,p}(x', x) \end{bmatrix} = \mathop{\mathbb{E}}_{x' \sim \mathcal{D}_x} \Big[d(x')^\mathsf{T} K(x', x) \Big]^\mathsf{T}.$$

With $K(x', x)^\mathsf{T} = K(x, x')$ for every $x, x' \in \mathbb{R}^d$ by definition, we conclude

$$\Phi_K\big(\mu\big)(x) = \mathop{\mathbb{E}}_{x' \sim \mathcal{D}_x} \Big[d(x')^\mathsf{T} K(x', x) \Big]^\mathsf{T} = \mathop{\mathbb{E}}_{x' \sim \mathcal{D}_x} \Big[K(x', x)^\mathsf{T} d(x') \Big] = \mathop{\mathbb{E}}_{x' \sim \mathcal{D}_x} \Big[K(x, x') d(x') \Big].$$

The last step consists of using the assumption, that $\mathcal{D}_x$ is the empirical distribution on a finite dataset $x_1, \dots, x_m \in \mathbb{R}^d$. This enables us to derive

$$\Phi_K\big(\mu\big)(x) = \mathop{\mathbb{E}}_{x' \sim \mathcal{D}_x} \Big[K(x, x') d(x') \Big] = \frac{1}{m} \sum_{i=1}^{m} K(x, x_i) d(x_i).$$

This proves the statement of the lemma. □

By lemma 3.16 we can write the kernel gradient of C at f_0 with respect to some kernel K as

$$\nabla_K C|_{f_0}(x) = \Phi_K\Big(\big\langle d|_{f_0}, \cdot\big\rangle_{\mathcal{D}_x}\Big)(x) = \frac{1}{m}\sum_{i=1}^{m} K(x, x_i) d|_{f_0}(x_i).$$

Due to the definition of the kernel, $\nabla_K C|_{f_0}$ has the advantage of being well-defined on the whole input space $\mathbb{R}^d$, where as the partial derivative $\partial_f C|_{f_0} = \langle d|_{f_0}, \cdot\rangle_{\mathcal{D}_x}$ is based on the bilinear form $\langle\cdot,\cdot\rangle_{\mathcal{D}_x}$ and therefore only defined on the finite dataset.

Definition 3.17. *Let K be a kernel. A time dependent function $f : \mathbb{R} \to \mathcal{F}$ is said to follow the kernel gradient descent on C with respect to K if it satisfies the differential equation*

$$\partial_t f(t) = -\nabla_K C|_{f(t)}.$$

We can think of such a function as a continuous extension to the sequence produced by kernel gradient descent on C, which can be used to investigate the change in C.

Lemma 3.18. *Let K be a positive definite kernel with respect to $\|\cdot\|_{\mathcal{D}_x}$ and $f : \mathbb{R} \to \mathcal{F}$ be a time dependent function, that follows the kernel gradient descent on C with respect to K, then the evolution of $C \circ f$ during kernel gradient descent is expressed by*

$$\partial_t C|_{f(t)} = -\big\|d|_{f(t)}\big\|_K^2,$$

where $d|_{f(t)} \in \mathcal{F}$ denotes the representer of $\partial_f C|_{f(t)} = \big\langle d|_{f(t)}, \cdot\big\rangle_{\mathcal{D}_x} \in \mathcal{F}^*$ for $t \in \mathbb{R}$.

Proof. Using the notation $C|_{f(t)} = C\big[f(t)\big]$, we derive

$$\partial_t C|_{f(t)} = \partial_f C|_{f(t)}\big[\partial_t f(t)\big] = \big\langle d|_{f(t)}, \cdot\big\rangle_{\mathcal{D}_x}\big[\partial_t f(t)\big] = \big\langle d|_{f(t)}, \partial_t f(t)\big\rangle_{\mathcal{D}_x}.$$

Since f follows the kernel gradient descent on C with respect to K this can be rewritten as

$$\partial_t C|_{f(t)} = \big\langle d|_{f(t)}, \partial_t f(t)\big\rangle_{\mathcal{D}_x} = \big\langle d|_{f(t)}, -\nabla_K C|_{f(t)}\big\rangle_{\mathcal{D}_x}.$$

Recalling the definition of $\nabla_K C|_{f(t)}$, which was given by

$$\nabla_K C|_{f(t)} = \frac{1}{m}\sum_{i=1}^{m} K(\cdot, x_i) d|_{f(t)}(x_i),$$

we derive by the linearity of $\langle\cdot,\cdot\rangle_{\mathcal{D}_x}$, that

$$\begin{aligned}
\big\langle d|_{f(t)}, -\nabla_K C|_{f(t)}\big\rangle_{\mathcal{D}_x} &= -\frac{1}{m}\sum_{i=1}^{m}\big\langle d|_{f(t)}, K(\cdot, x_i) d|_{f(t)}(x_i)\big\rangle_{\mathcal{D}_x} \\
&= -\frac{1}{m}\sum_{i=1}^{m}\mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[d|_{f(t)}(x)^\mathsf{T} K(x, x_i) d|_{f(t)}(x_i)\Big] \\
&= -\mathop{\mathbb{E}}_{x'\sim\mathcal{D}_x}\mathop{\mathbb{E}}_{x\sim\mathcal{D}_x}\Big[d|_{f(t)}(x)^\mathsf{T} K(x, x') d|_{f(t)}(x')\Big] = -\big\langle d|_{f(t)}, d|_{f(t)}\big\rangle_K.
\end{aligned}$$

Finally we conclude, that

$$\partial_t C|_{f(t)} = \big\langle d|_{f(t)}, -\nabla_K C|_{f(t)}\big\rangle_{\mathcal{D}_x} = -\big\langle d|_{f(t)}, d|_{f(t)}\big\rangle_K = -\big\|d|_{f(t)}\big\|_K^2.$$

This completes the proof. □

The lemma guarantees the convergence of C to a critical point, as $\partial_t C|_{f(t)} \leqslant 0$. In the case where C is convex and bounded from below, this implies the convergence of $f(t)$ to a global minimum $f^* \in \mathcal{F}$ for $t \to \infty$.

3.3 Kernel Approximation

This subsection illustrates the relationship between gradient descent in the parameter space and kernel gradient descent in the function space under the assumption, that the objective function depends linear on its parameters. The example given in here, will be generalized later on, leading to the definition of the neural tangent kernel.

To start things off, we let $\mathcal{D}_{\mathcal{F}}$ be any probability distribution on the function space $\mathcal{F}$ and construct a kernel K, such that for $x, x' \in \mathbb{R}^d$ the image $K(x, x')$ is a $p \times p$ matrix defined by

$$K_{i,j}(x, x') \coloneqq \mathop{\mathbb{E}}_{f \sim \mathcal{D}_{\mathcal{F}}} \Big[f_i(x) \cdot f_j(x) \Big], \quad i, j = 1, \dots, p.$$

We will see, that this kernel can be approximated by random functions. For this we choose some $n \in \mathbb{N}$ and draw n random functions $f^{(k)} : \mathbb{R}^d \to \mathbb{R}^p$ for $k = 1, \dots, n$ independently from the distribution $\mathcal{D}_{\mathcal{F}}$. Using these, we can define the linear map

$$F^{lin} : \mathbb{R}^n \to \mathcal{F} : w \mapsto f_w \coloneqq \frac{1}{\sqrt{n}} \sum_{k=1}^{n} w_k f^{(k)},$$

whose partial derivatives for $k = 1, \dots, n$ are given by

$$\partial_{w_k} F^{lin} : \mathbb{R}^n \to \mathcal{F} : w \mapsto \frac{1}{\sqrt{n}} f^{(k)}.$$

Based on F^{lin} we can define another kernel, that will be used for the approximation of K.

Definition 3.19. *For $x \in \mathbb{R}^p$ we define the symmetric matrix $x \otimes x \coloneqq M \in \mathbb{R}^{p \times p}$, where*

$$M_{i,j} \coloneqq x_i \cdot x_j, \quad i, j = 1, \dots, p.$$

Definition 3.20. *Given n random functions $f^{(k)} : \mathbb{R}^d \to \mathbb{R}^p$ for $k = 1, \dots, n$ drawn i.i.d. according to some distribution $\mathcal{D}_{\mathcal{F}}$ on the function space $\mathcal{F}$, the tangent kernel is defined as*

$$\tilde{K} \coloneqq \sum_{k=1}^{n} \partial_{w_k} F^{lin}(w) \otimes \partial_{w_k} F^{lin}(w) = \frac{1}{n} \sum_{k=1}^{n} f^{(k)} \otimes f^{(k)}.$$

In other words, $\tilde{K}$ maps a tuple $(x, x') \in \mathbb{R}^d \times \mathbb{R}^d$ to the matrix $\tilde{K} \in \mathbb{R}^{p \times p}$ with entries

$$\tilde{K}_{i,j}(x, x') = \frac{1}{n} \sum_{k=1}^{n} f_i^{(k)}(x) \cdot f_j^{(k)}(x'), \quad i, j = 1, \dots, p.$$

Similar to the definition of time dependent functions, that follow the kernel gradient descent on C with respect to some kernel K we can define time dependent functions, that follow the gradient descent on $C \circ F^{lin}$. This is done by the following definition.

Definition 3.21. *A time dependent function $w : \mathbb{R} \to \mathbb{R}^n$ is said to follow the gradient descent on $C \circ F^{lin}$, if it satisfies the differential equation*

$$\partial_t w(t) = -\nabla \big(C \circ F^{lin} \big)(w(t)) = -\nabla C|_{f_{w(t)}},$$

where $-\nabla C|_{f_{w(t)}}$ denotes the gradient of $C \circ F^{lin}$ at $w(t)$.

Again, we can think of such a function as a continuous extension to the sequence produced by gradient descent on $C \circ F^{lin}$.

Lemma 3.22. *Let $w : \mathbb{R} \to \mathbb{R}^n$ be a time dependent function, that follows the gradient descent on $C \circ F^{lin}$. For $k = 1, \dots, n$, the change in w_k during gradient descent is given by*

$$\partial_t w_k(t) = -\frac{1}{\sqrt{n}} \big\langle d|_{f_{w(t)}}, f^{(k)} \big\rangle_{\mathcal{D}_x},$$

where $d|_{f_{w(t)}} \in \mathcal{F}$ denotes the representer of $\partial_f C|_{f_{w(t)}} = \big\langle d|_{f_{w(t)}}, \cdot \big\rangle_{\mathcal{D}_x} \in \mathcal{F}^*$ for $t \in \mathbb{R}$.

Proof. Using the notation in definition 3.21, we derive

$$\partial_t w_k(t) = -\partial_{w_k}\big(C \circ F\big)(w(t)) = \partial_f C|_{f_{w(t)}}\Big[-\frac{1}{\sqrt{n}} f^{(k)} \Big] = -\frac{1}{\sqrt{n}} \big\langle d|_{f_{w(t)}}, f^{(k)} \big\rangle_{\mathcal{D}_x}$$

for $k = 1, \dots, n$. This completes the proof. □

Based on this lemma we can establish a relationship between time dependent functions that follow the gradient descent on $C \circ F^{lin}$ and time dependent functions that follow the kernel gradient descent on C with respect to the tangent kernel.

Lemma 3.23. *Let $w : \mathbb{R} \to \mathbb{R}^n$ be a time dependent function, that follows the gradient descent on $C \circ F^{lin}$, then the time dependent function $F^{lin} \circ w : \mathbb{R} \to \mathcal{F} : t \mapsto f_{w(t)}$ follows the kernel gradient descent on C with respect to the tangent kernel $\tilde{K}$. Formally this is*

$$\partial_t\big(F^{lin} \circ w\big)(t) = \partial_t f_{w(t)} = -\nabla_{\tilde{K}} C|_{f_{w(t)}}.$$

Proof. By lemma 3.22 and the definition of F^{lin} it holds, that

$$\partial_t f_{w(t)} = \frac{1}{\sqrt{n}} \sum_{k=1}^{n} \partial_t w_k(t) \cdot f^{(k)} = -\frac{1}{n} \sum_{k=1}^{n} \big\langle d|_{f_{w(t)}}, f^{(k)} \big\rangle_{\mathcal{D}_x} \cdot f^{(k)}.$$

Furthermore, we can write the kernel gradient of C at $f_{w(t)}$ with respect to $\tilde{K}$ as

$$\begin{aligned}
\nabla_{\tilde{K}} C|_{f_{w(t)}} &= \frac{1}{m} \sum_{i=1}^{m} K(\cdot, x_i) \cdot d|_{f_{w(t)}}(x_i) \\
&= \frac{1}{m} \sum_{i=1}^{m} \left[\frac{1}{n} \sum_{k=1}^{n} f^{(k)} \otimes f^{(k)} \right](\cdot, x_i) \cdot d|_{f_{w(t)}}(x_i) \\
&= \frac{1}{n} \sum_{k=1}^{n} \left[\frac{1}{m} \sum_{i=1}^{m} f^{(k)}(x_i)^{\mathsf{T}} d|_{f_{w(t)}}(x_i) \right] \cdot f^{(k)} \\
&= \frac{1}{n} \sum_{k=1}^{n} \mathop{\mathbb{E}}_{x \sim \mathcal{D}_x} \Big[f^{(k)}(x)^{\mathsf{T}} d|_{f_{w(t)}}(x) \Big] \cdot f^{(k)} \\
&= \frac{1}{n} \sum_{k=1}^{n} \big\langle f^{(k)}, d|_{f_{w(t)}} \big\rangle_{\mathcal{D}_x} \cdot f^{(k)}.
\end{aligned}$$

Comparison of the previous two expressions provides

$$\partial_t f_{w(t)} = -\frac{1}{n} \sum_{k=1}^{n} \big\langle d|_{f_{w(t)}}, f^{(k)} \big\rangle_{\mathcal{D}_x} \cdot f^{(k)} = -\nabla_{\tilde{K}} C|_{f_{w(t)}}.$$

This proves, that $F^{lin} \circ w$ follows the kernel gradient descent on C with respect to $\tilde{K}$. □

Thus kernel gradient descent on C with respect to the tangent kernel $\tilde{K}$ is equivalent to gradient descent on $C \circ F^{lin}$. For $i, j = 1, \dots, p$ the law of large numbers yields

$$\forall x, x' \in \mathbb{R}^d : \lim_{n\to\infty} \tilde{K}_{i,j}(x, x') = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} f_i^{(k)}(x) \cdot f_j^{(k)}(x') = \mathop{\mathbb{E}}_{f\sim\mathcal{D}_{\mathcal{F}}} \Big[f_i(x) \cdot f_j(x) \Big].$$

Hence the random tangent kernel $\tilde{K}$ is an approximation of the constant limiting kernel K.

3.4 Neural Tangent Kernel

The training of neural networks behaves quite similar to the kernel gradient descent with respect to the tangent kernel. However, in contrast to the situation in section 3.3, we consider the realization function

$$F : \mathbb{R}^n \to \tilde{\mathcal{F}} : w \mapsto f(x, w),$$

which is no longer linear due to the nested construction of neural networks. Therefore, the partial derivatives $\partial_{w_k} F$ are no longer independent of the parameters w, which motivates the definition of a new kernel, specifically related to neural networks.

Definition 3.24. *The neural tangent kernel is defined as*

$$\Theta_w \coloneqq \sum_{k=1}^{n} \partial_{w_k} F(w) \otimes \partial_{w_k} F(w).$$

Thus, in contrast to the tangent kernel, which is constant during training since it is independent on the parameters w, the neural tangent kernel is random at initialization and varies during training. Therefore the relationship between gradient descent in the parameter space and kernel gradient descent with respect to the neural tangent kernel is more complex.

We will see, that in the infinite-width limit, that is the number of neurons per layer tends to infinity, the neural tangent kernel becomes deterministic at initialization and stays constant during training.

In order to simplify the further notation, we have to make some adjustments to the notion of neural networks in section 2.1. We still consider neural networks for $l \in \mathbb{N}$ and $d_0, \dots, d_l \in \mathbb{N}$ as functions

$$f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_l} : x \mapsto f_l \circ f_{l-1} \circ \cdots \circ f_1(x),$$

where each f_i for $i = 1, \dots, l$ is a layer with input dimension d_{i-1} and output dimension d_i. However, in contrast to the notation in section 2.1 we denote these layers as functions

$$f_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i} : x \mapsto \sigma\Big(\frac{1}{\sqrt{d_{i-1}}} W^{(i-1)} x + \beta b^{(i-1)} \Big),$$

where the activation function σ is applied entrywise. This notation arises by formulating definition 2.2 in terms of vectors and matrices, and introducing the additional factors $1/\sqrt{d_{i-1}}$ and $\beta > 0$. This procedure is important for later results.

Definition 3.25. *Let $T \subset \mathbb{R}$ be an index set. A family of random variables $\{X_t \mid t \in T\}$ is called a Gaussian process, if and only if for every finite set $\{t_1, \dots, t_k\} \subset T$, the random variable $(X_{t_1}, \dots, X_{t_k})$ is multivariate Gaussian distributed.*

Lemma 3.26. *Let $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_l}$ be a neural network of depth l, with a Lipschitz continuous non-linear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ at each neuron. In the limit as $d_1, \dots, d_{l-1} \to \infty$ sequentially, the output functions $f_i(x, w)$ for $i = 1, \dots, d_l$, tend to i.i.d. centered Gaussian processes of covariance $\Sigma^{(l)}$, where $\Sigma^{(l)}$ is defined recursively by*

$$\Sigma^{(l)}(x, x') = \frac{1}{d_0} x^\mathsf{T} x' + \beta^2$$
$$\Sigma^{(l+1)}(x, x') = \mathop{\mathbb{E}}_{f \sim \mathcal{N}(0, \Sigma^{(l)})} \Big[\sigma\big(f(x)\big) \sigma\big(f(x')\big) \Big] + \beta^2,$$

taking the expectation with respect to a centered Gaussian process f of covariance $\Sigma^{(l)}$.

Proof. **TO DO** □

Theorem 3.27. *Let $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_l}$ be a neural network of depth l, with a Lipschitz continuous non-linear activation function $\sigma : \mathbb{R} \to \mathbb{R}$ at each neuron. In the limit as $d_1, \dots, d_{l-1} \to \infty$ sequentially, the neural tangent kernel Θ_w converges in probability to a deterministic limiting kernel, this is*

$$\Theta_w \to \Theta_\infty \otimes I_{d_0}.$$

The scalar kernel $\Theta_\infty : \mathbb{R}^{d_0} \times \mathbb{R}^{d_0} \to \mathbb{R}$ is defined recursively by

$$\Theta_\infty^{(1)}(x, x') = \Sigma^{(1)}(x, x')$$
$$\Theta_\infty^{(l+1)}(x, x') = \Theta_\infty^{(l)}(x, x') \dot{\Sigma}^{(l+1)}(x, x') + \Sigma^{(l+1)}(x, x'),$$

where

$$\dot{\Sigma}^{(l+1)}(x, x') \coloneqq \mathop{\mathbb{E}}_{f \sim \mathcal{N}(0, \Sigma^{(l)})} \Big[\sigma'\big(f(x)\big) \sigma'\big(f(x')\big) \Big],$$

taking the expectation with respect to a centered Gaussian process f of covariance $\Sigma^{(l)}$.

Proof. **TO DO** □

Theorem 3.28. *Assume that $\sigma : \mathbb{R} \to \mathbb{R}$ is a Lipschitz continuous and twice differentiable non-linear function with bounded second derivative. For any T such that the integral $\int_0^T \|d_t\|_{\mathcal{D}_x} \, dt$ stays stochastically bounded, as $d_1, \dots, d_{l-1} \to \infty$ sequentially, we have, uniformly for $t \in [0, T]$, that*

$$\Theta_w \to \Theta_\infty \otimes I_{d_l}.$$

As a consequence, in this limit, the dynamics of f_w is described by the differential equation

$$\partial_t f_{w(t)} = \Phi_{\Theta_\infty \otimes I_{d_l}} \Big(\big\langle d_t, \cdot \big\rangle_{\mathcal{D}_x} \Big).$$

Proof. **TO DO** □

Theorem 3.29. *Theorem 3.7 from [8]*

Proof. **TO DO** □

Theorem 3.30. *Theorem 2.1 from [9]*

Proof. **TO DO** □

TO DO - Section 3.2
LEONARDO: Why is $\partial_f C|_{f_0}$ linear and continuous?
LEONARDO: Why does the first equality in proof of lemma 3.20 holds?

TO DO - Section 3.3
LEONARDO: Why does the first equality in proof of lemma 3.24 holds?
LEONARDO: Is the equivalence between the gradient methods clear?

TO DO
References (E-Mail + 3.7 wikipedia references)
Notation (double variables)
Notation 3.7: d instead of μ

TO DO MORE DETAILED
Completeness argument
Critical point of functionals
Convergence of kernel gradient descent
More general words on why we need theorems and definitions

TO DO POTENTIALLY
Proofs 3.29 - 3.31
Backpropagation
Convergence of functional gradient descent

TIPS
libgen.rs for sources

4 Dynamic Linear Dimensionality Reduction

This section is dedicated to the theoretical considerations on how to efficiently reduce the computation effort needed, to train deep neural networks. If not otherwise specified, the following results stem from [1]. The authors approach is based on the hypothesis, that deep neural networks can be trained in low-dimensional subspaces. They argue, that the parameter optimization trajectory could be embedded in a tiny subspace. We will start by motivating this hypothesis based on the previous seen properties of the neural tangent kernel. Afterwards we will introduce an efficient method for the subspace extraction, that enables us to train the network in a tiny subspace. Finally we can construct a second order optimization method training the parameters in the lower-dimensional space.

4.1 Approach

In the following we will denote the parameter training sequence by $(w_k)_{k=0,\dots,K}$ after $K \in \mathbb{N}$ iterations. This sequence naturally arises during any of the proposed optimization methods in section 2.2. The low-dimensional landscape hypothesis from [1] assumes that, there exists a low-dimensional affine set which approximately contains the optimization trajectory. This can be simply illustrated by the following example.

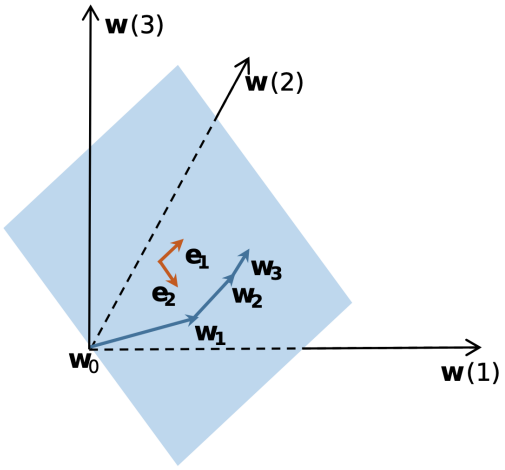

Figure 4: Low-dimensional parameter trajectory

Figure 4 visualizes a three dimensional space containing the three optimizable parameters $w(1), w(2), w(3)$. However, the training trajectory $(w_k)_{k=0,\dots,3}$ could be embedded in a two-dimensional hyperplane, spanned by the orthogonal vectors e_1 and e_2. Note that none of the optimizable variables could be left out to reduce the dimension, but rather one has to construct new independent variables to obtain the lower-dimensional subspace that approximately contains the optimization trajectory.

In the following we will investigate this hypothesis from a theoretical background and conduct several experiments indicating that deep neural networks can be well trained in tiny subspaces of approximately dimension 50. This enables us to efficiently use second order information in the training process and improves robustness against label noise.

4.2 Theory

To effectively reduce the dimensionality we need some background knowledge on the singular value decomposition (SVD) and low-rank approximations.

4.2.1 Singular Value Decomposition

The following result is denoted and proven as theorem 2.5.2 in [10].

Theorem 4.1 (Singular Value Decomposition)**.** *For every matrix $A \in \mathbb{R}^{m\times n}$ there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ and a diagonal matrix*

$$\Sigma \coloneqq diag(\sigma_1, \dots, \sigma_p) \in \mathbb{R}^{m\times n}$$

with $p \coloneqq min\{m, n\}$ and singular values $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_p \geqslant 0$, such that

$$A = U\Sigma V^{\mathsf{T}}.$$

The SVD decomposes any real valued matrix in a product of three special matrices. This representation yields the singular values which, like eigenvalues, characterize important properties of a matrix. For the further course we need the following definition.

Definition 4.2. *The columns of the matrix $U = [u_1, \dots, u_m]$ are referred to as left singular vectors and the columns of the matrix $V = [v_1, \dots, v_n]$ are referred to as right singular vectors.*

Theorem 4.1 shows that the sequence of singular values is unique, since they are sorted in descending order. However, the matrices U and V do not have to be unique. For example, the corresponding singular vectors can be interchanged if there are two identical singular values. Therefore, a matrix can have several singular value decompositions.

As a simple implication of theorem 4.1 we denote the following corollary.

Corollary 4.3. *Let $A \in \mathbb{R}^{m\times n}$ be a matrix with singular value decomposition $A = U\Sigma V^{\mathsf{T}}$, such that $p \coloneqq min\{m, n\}$ and $\sigma_1 \geqslant \cdots \geqslant \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0$ for $r \leqslant p$.*

1. *Denoting the columns of U and V with u_i and v_i, we have*

$$Av_i = \sigma_i u_i, \quad A^{\mathsf{T}}u_i = \sigma_i v_i, \quad i = 1, \dots, p.$$

2. *The squares of the singular values $\sigma_1^2, \dots, \sigma_r^2$ are the eigenvalues of $A^{\mathsf{T}}A$ to the corresponding eigenvectors $v_1, \dots, v_r$.*

Proof.

1. Using $A = U\Sigma V^{\mathsf{T}}$ and the orthogonality of V we derive

$$Av_i = U\Sigma V^{\mathsf{T}}v_i = U\Sigma e_i = U\sigma_i e_i = \sigma_i u_i$$

for $i = 1, \dots, p$. Analogously we conclude the second statement

$$A^{\mathsf{T}}u_i = V\Sigma U^{\mathsf{T}}u_i = V\Sigma e_i = V\sigma_i e_i = \sigma_i v_i.$$

2. Since $A^\mathsf{T}A$ is symmetric, its spectral decomposition is given by

$$A^\mathsf{T}A = V\Sigma^\mathsf{T}U^\mathsf{T}U\Sigma V^\mathsf{T} = V\Sigma^\mathsf{T}\Sigma V^\mathsf{T} = V\Sigma^\mathsf{T}\Sigma V^{-1}.$$

Using $\Sigma^\mathsf{T}\Sigma = \operatorname{diag}(\sigma_1^2, \dots, \sigma_r^2, 0, \dots, 0) \in \mathbb{R}^{n\times n}$ we conclude the corollary. □

The SVD is an excellent tool to find low-rank approximations of a matrix $A \in \mathbb{R}^{m\times n}$, that is to find a matrix $B \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(B) < \operatorname{rank}(A)$ such that $\|A - B\|$ is small for some given matrix norm. We can generate such a low-rank approximation by simply reducing the number of singular values in the singular value decomposition.

Definition 4.4. *Let $A \in \mathbb{R}^{m\times n}$ be a matrix with singular value decomposition $A = U\Sigma V^\mathsf{T}$. For $k < r = \operatorname{rank}(A)$ we define an approximation of A as*

$$A^k \coloneqq \sum_{i=1}^{k} u_i\sigma_i v_i^\mathsf{T} = \tilde{U}\tilde{\Sigma}\tilde{V}^\mathsf{T},$$

where $\tilde{U} \coloneqq [u_1, \dots, u_k]$, $\tilde{V} \coloneqq [v_1, \dots, v_k]$ and $\tilde{\Sigma} \coloneqq \operatorname{diag}(\sigma_1, \dots, \sigma_k)$.

In terms of the spectral norm $\|\cdot\|_2$, the matrix A^k is the best possible rank-k approximation. The underlying theorem is the following, which is denoted and proven as theorem 2.5.3 in [10].

Theorem 4.5 (Eckart-Young-Mirsky)**.** *Let $A \in \mathbb{R}^{m\times n}$ be a matrix with singular value decomposition $A = U\Sigma V^\mathsf{T}$. For every $k < r = \operatorname{rank}(A)$ it holds, that*

$$\min_{\operatorname{rank}(B)=k} \|A - B\|_2 = \|A - A^k\|_2 = \sigma_{k+1}.$$

This concludes the necessary background knowledge on singular value decompositions and low-rank approximations, which will be useful for the subspace extraction.

4.2.2 Dimensionality Reduction

Next, we investigate a single output neural network architecture

$$f : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R} : (x, w) \mapsto f(x, w).$$

As seen in section 2.2 the neural network architecture f induces a function space

$$\mathcal{F} \coloneqq \Big\{ f_w : \mathbb{R}^d \to \mathbb{R} : x \mapsto f(x, w) \mid w \in \mathbb{R}^n \Big\}.$$

In order to train the neural network we let

$$\mathcal{X} = \big\{(x_i, y_i) \mid i = 1, \dots, m\big\} \subset \mathbb{R}^d \times \mathbb{R}$$

denote a training set of size $m \in \mathbb{N}$. Recall that training of f refers to finding the optimal parameter vector w, which is accomplished by minimizing the empirical error function

$$E : \mathbb{R}^n \to \mathbb{R} : w \mapsto \sum_{i=1}^{m} \ell\big(f(x_i, w), y_i\big).$$

Note that in comparison to the formulation given in section 2.2, we removed the constant factor $1/m$ in front of the sum, which has no influence on the minimum. Furthermore we assumed $\lambda = 0$, that is we do not penalize the complexity of w.

Under the assumption, that all activation functions in f are differentiable, thus f is differentiable itself, the choice of a differentiable loss function ℓ enables us to minimize E via gradient descent. In order to analyze this optimization process we define the cost functionals

$$\mathcal{L}_i : \mathcal{F} \to \mathbb{R} : f_w \mapsto \ell(f_w(x_i), y_i)$$

for $i = 1, \dots, m$, such that the empirical error E can be written as

$$E(w) = \sum_{i=1}^{m} \ell\big(f(x_i, w), y_i\big) = \sum_{i=1}^{m} \ell\big(f_w(x_i), y_i\big) = \sum_{i=1}^{m} \mathcal{L}_i(f_w).$$

To investigate the parameter evolution during gradient descent on E we will use the concept of gradient flows.

Definition 4.6. *The gradient flow according to E is a time dependent function*

$$w : [0, \infty) \to \mathbb{R}^n : t \mapsto w_t,$$

that suffices the ordinary differential equation

$$\partial_t w_t = -\nabla E(w_t).$$

Since gradient descent performs parameter updates with regard to the update rule

$$w_{k+1} \leftarrow w_k - \alpha \cdot \nabla E(w_k),$$

we can let $\alpha \to 0$, to derive the equation

$$\lim_{\alpha \to 0} \frac{w_{k+1} - w_k}{\alpha} = -\nabla E(w_k).$$

Thus, the gradient flow w covers the parameter dynamics during gradient descent on E.

Lemma 4.7. *In the case of a single output neural network architecture, the gradient flow according to E suffices the ordinary differential equation*

$$\partial_t w_t = -\nabla_w f(\mathcal{X}, w_t)^\mathsf{T} \cdot \nabla_{f(\mathcal{X},w)} \mathcal{L}|_{f_{w_t}},$$

where

$$\nabla_w f(\mathcal{X}, w_t)^\mathsf{T} \coloneqq \Big[\nabla_w f(x_1, w_t), \dots, \nabla_w f(x_m, w_t)\Big] \in \mathbb{R}^{n \times m}$$

denotes the gradients of f at w_t on the training data set $\mathcal{X}$ and

$$\nabla_{f(\mathcal{X},w)} \mathcal{L}|_{f_{w_t}} \coloneqq \Big[\nabla \mathcal{L}_1|_{f_{w_t}}(x_1), \dots, \nabla \mathcal{L}_m|_{f_{w_t}}(x_m)\Big]^\mathsf{T} \in \mathbb{R}^m$$

denotes the functional gradients of the costs $\mathcal{L}_i$ at f_{w_t} evaluated on x_i for $i = 1, \dots, m$.

Proof. By definition 4.6 we need to show, that

$$\partial_t w_t = -\nabla E(w_t) = -\nabla_w f(\mathcal{X}, w_t)^\mathsf{T} \cdot \nabla_{f(\mathcal{X},w)} \mathcal{L}|_{f_{w_t}}.$$

For arbitrary $w \in \mathbb{R}^n$ we can apply the chain rule to derive the gradient

$$\nabla E(w) = \sum_{i=1}^{m} \nabla_w \ell\big(f(x_i, w), y_i\big) = \sum_{i=1}^{m} \nabla_w f(x_i, w) \cdot \nabla \mathcal{L}_i|_{f_w}(x_i).$$

Using the definitions from the lemma, this is equivalent to

$$\nabla E(w) = \sum_{i=1}^{m} \nabla_w f(x_i, w) \cdot \nabla \mathcal{L}_i|_{f_w}(x_i) = \nabla_w f(\mathcal{X}, w_t)^\mathsf{T} \cdot \nabla_{f(\mathcal{X},w)} \mathcal{L}|_{f_{w_t}}.$$

This completes the proof, since w was chosen arbitrary. □

In the infinite width limit, we can apply theorem 3.30 to approximate the neural network architecture f as a linear model

$$f^{lin}(x, w_t) \approx f(x, w_0) + \nabla_w f(\mathcal{X}, w_0)(w_t - w_0).$$

This is nothing else than a first order Taylor expansion. However, the key observation is, that in the infinite width limit, the error term converges to zero. Based on this, the differential equation in lemma 4.7 can be rewritten as

$$\partial_t w_t = -\nabla_w f(\mathcal{X}, w_0)^\mathsf{T} \cdot \nabla_{f(\mathcal{X},w)} \mathcal{L}|_{f_{w_t}},$$

where the first factor is a constant matrix and the second factor changes over time t.

At this stage we can explain the main idea of how to reduce the dimensionality of the parameter space. If $\nabla_w f(\mathcal{X}, w_0)$ can be approximated by a low-rank matrix, the training trajectory $(w_k)_{k=0,\dots,K}$ would only depend on this low-rank constant matrix and the functional gradient of C, which will enable us to effectively reduce the dimensionality. To clarify this, we apply singular value decomposition on $\nabla_w f(\mathcal{X}, w_0)$ and derive

$$\nabla_w f(\mathcal{X}, w_0) = U_0 \Sigma_0 V_0^\mathsf{T},$$

where $U_0 \in \mathbb{R}^{m \times m}$ and $V_0 \in \mathbb{R}^{n \times n}$ are orthogonal and $\Sigma_0 = \operatorname{diag}(\sigma_1, \dots, \sigma_m)$ is positive semi-definite. Using the notation

$$\tilde{f} : \mathbb{R}^n \to \mathbb{R}^m : w \mapsto \Big[f(x_1, w), \dots, f(x_m, w)\Big]^\mathsf{T},$$

we can apply definition 3.24 to derive the neural tangent kernel

$$\Theta_{w_0} = \sum_{k=1}^{n} \partial_{w_k} \tilde{f}(w_0) \otimes \partial_{w_k} \tilde{f}(w_0) = \nabla_w f(\mathcal{X}, w_0) \cdot \nabla_w f(\mathcal{X}, w_0)^\mathsf{T} = U_0 \Sigma_0 \Sigma_0^\mathsf{T} U_0^\mathsf{T}.$$

This is nothing else than the spectral decomposition of Θ_0. By theorem 3.29 $\cdots$. Thus by theorem 4.5 we can approximate Σ_0 by a low-rank matrix $\tilde{\Sigma}_0 = \operatorname{diag}(\sigma_1, \dots, \sigma_d)$, such that

$$\Sigma_0 \approx \tilde{U}_0 \tilde{\Sigma}_0 \tilde{V}_0^\mathsf{T},$$

where $\tilde{U}_0 \in \mathbb{R}^{m \times d}$ contains the first d columns of an $m-$dimensional identity matrix and $\tilde{V}_0 \in \mathbb{R}^{n \times d}$ contains the first d columns of an $n-$dimensional identity matrix. Using this approximation we can derive

$$\nabla_w f(\mathcal{X}, w_0) = U_0 \Sigma_0 V_0^\mathsf{T} \approx U_0 \tilde{U}_0 \tilde{\Sigma}_0 \tilde{V}_0^\mathsf{T} V_0^\mathsf{T}.$$

Thus, the dynamics of gradient flow are approximately governed by

$$\partial_t w_t \approx -\nabla_w f(\mathcal{X}, w_0)^\mathsf{T} \cdot \nabla_{f(\mathcal{X},w)} \mathcal{L}|_{f_{w_t}} \approx -V_0 \tilde{V}_0 \Big(\tilde{\Sigma}_0 \tilde{U}_0^\mathsf{T} U_0^\mathsf{T} \cdot \nabla_{f(\mathcal{X},w)} \mathcal{L}|_{f_{w_t}} \Big)$$

The clasped part yields the projected gradient in the $d-$dimensional subspace and the variables $V_0 \tilde{V}_0$ establish a projection from the lower-dimensional space back to the $n-$dimensional parameter space.

This concludes the theoretical motivation of the subspace extraction. Based on this approach we could effectively train neural networks in the lower-dimensional subspace. However, the previous discussion holds only in the infinite width limit for training in the lazy regime. The rest of this thesis is dedicated to numerical experiments indicating that deep neural networks indeed can be trained in low-dimensional subspaces, although the theoretical conditions do not hold in practice.

Problem: Chain rule for functionals.
Problem: Reference and existence of gradient flow

4.3 Methodology

The key issue of dimensionality reduction consist of finding a suitable dimension $d \in \mathbb{N}$ and a $d-$dimensional subspace that approximately covers the optimization trajectory. We will address this problem via principal component analysis (PCA), which is based on the concept of orthogonal projections.

4.3.1 Orthogonal Projections

The results of this subsection are denoted in section 2.6.1 in [10].

Lemma 4.8. *Let $S \subseteq \mathbb{R}^n$ be a subspace. There exists a unique matrix $P \in \mathbb{R}^{n\times n}$, such that*

$$Im(P) = S \ \wedge \ P^2 = P = P^\mathsf{T}.$$

The linear map $\tilde{P} : \mathbb{R}^n \to \mathbb{R}^n : x \mapsto Px$ is called the orthogonal projection onto S.

Proof. To prove the existence of an orthogonal projection, we let $d \coloneqq \dim(S)$ denote the dimension of S and $V := [v_1, \dots, v_d] \in \mathbb{R}^{n\times d}$ denote an orthonormal basis of S. We define

$$P \coloneqq VV^\mathsf{T} \in \mathbb{R}^{n\times n}$$

and observe by the orthonormality of V, that $V^\mathsf{T}V = I_d$. Thus P is idempotent, this is

$$P^2 = VV^\mathsf{T}VV^\mathsf{T} = VI_dV^\mathsf{T} = VV^\mathsf{T} = P.$$

Furthermore P is symmetric, since

$$P^\mathsf{T} = (VV^\mathsf{T})^\mathsf{T} = VV^\mathsf{T} = P.$$

By the orthonormality of V, for arbitrary $s \in S$ there exists $a = (a_1, \dots, a_d)^\mathsf{T} \in \mathbb{R}^d$ with

$$s = a_1v_1 + \cdots + a_dv_d = Va.$$

Based on this we derive $S \subseteq \mathrm{Im}(P)$ since

$$Ps = VV^\mathsf{T}s = VV^\mathsf{T}Va = VI_da = Va = s.$$

Additionally $\mathrm{Im}(P) \subseteq S$ due to the fact, that for $y \in \mathbb{R}^n$, we have $\tilde{a} \coloneqq V^\mathsf{T}y \in \mathbb{R}^d$, so

$$Py = VV^\mathsf{T}y = V\tilde{a} \in S.$$

Thus $\mathrm{Im}(P) = S$. This proves the existence of an orthogonal projection P onto S. To prove uniqueness of P, we should first note that based on $P^2 = P = P^\mathsf{T}$ we have

$$\forall x, y \in \mathbb{R}^n : (Px)^\mathsf{T}(y - Py) = x^\mathsf{T}P^\mathsf{T}y - x^\mathsf{T}P^\mathsf{T}Py = x^\mathsf{T}Py - x^\mathsf{T}Py = 0.$$

Using $Px \in S$ for every $x \in \mathbb{R}^n$, we can conclude that

$$\forall y \in \mathbb{R}^n : y - Py \in S^\perp := \big\{a \in \mathbb{R}^n \mid \forall b \in S : a^\mathsf{T}b = 0\big\}.$$

Next we let P_1 and P_2 denote orthogonal projections onto S, then for any $z \in \mathbb{R}^n$ we derive

$$\begin{aligned}\|(P_1 - P_2)z\|_2^2 &= \|P_1 z - P_2 z\|_2^2 \\ &= z^\mathsf{T} P_1^\mathsf{T} P_1 z - z^\mathsf{T} P_2^\mathsf{T} P_1 z - z^\mathsf{T} P_1^\mathsf{T} P_2 z + z^\mathsf{T} P_2^\mathsf{T} P_2 z \\ &= z^\mathsf{T} P_1 z - z^\mathsf{T} P_2^\mathsf{T} P_1 z - z^\mathsf{T} P_1^\mathsf{T} P_2 z + z^\mathsf{T} P_2 z,\end{aligned}$$

where we have used the property $P_i^2 = P_i = P_i^\mathsf{T}$ for $i = 1, 2$. Since each term on the right-hand side is a real number, we can transpose each term to derive

$$\begin{aligned}\|(P_1 - P_2)z\|_2^2 &= z^\mathsf{T} P_1 z - z^\mathsf{T} P_2^\mathsf{T} P_1 z - z^\mathsf{T} P_1^\mathsf{T} P_2 z + z^\mathsf{T} P_2 z \\ &= (P_1 z)^\mathsf{T} z - (P_1 z)^\mathsf{T} P_2 z - (P_2 z)^\mathsf{T} P_1 z + (P_2 z)^\mathsf{T} z \\ &= (P_1 z)^\mathsf{T}(z - P_2 z) + (P_2 z)^\mathsf{T}(z - P_1 z).\end{aligned}$$

Using $\mathrm{Im}(P_1) = S = \mathrm{Im}(P_2)$, we observe that

$$P_1 z \in S \ \wedge \ z - P_2 z \in S^\perp \ \wedge \ P_2 z \in S \ \wedge \ z - P_1 z \in S^\perp.$$

Thus the right-hand side of the equality is zero and by the positive definiteness of the norm we conclude $P_1 - P_2 = 0$. This implies $P_1 = P_2$ showing the uniqueness of P. □

In our setting, the d-dimensional subspace $S \subseteq \mathbb{R}^n$ is unknown, which is why the following definition will be needed.

Definition 4.9. *For $d \in \mathbb{N}$ we define the set of rank-d orthogonal projection matrices*

$$\mathcal{P}_d \coloneqq \left\{ P \in \mathbb{R}^{n \times n} \mid \mathit{rank}(P) = d \ \wedge \ P^2 = P = P^\mathsf{T} \right\}.$$

Having this background knowledge on orthogonal projections in mind, we can proceed with the theory on principal component analysis.

4.3.2 Principal Component Analysis

The results of this subsection can be found in chapter 15.1 of [11]. In the following we fix $d \leqslant n$ and let $W = [w_1, \ldots, w_t] \in \mathbb{R}^{n \times t}$ be a mean-centered data matrix, that is $\sum_{i=1}^t w_i = 0$.

Principal component analysis consists of finding the orthogonal projection matrix $P^* \in \mathcal{P}_d$, that minimizes the reconstruction error

$$\mathcal{P}_d \to \mathbb{R} : P \mapsto \sum_{i=1}^t \|P w_i - w_i\|_2^2.$$

Equivalently PCA can be formulated as finding the solution $P^* \in \mathcal{P}_d$, such that

$$P^* \coloneqq \operatorname*{arg\,min}_{P \in \mathcal{P}_d} \|PW - W\|_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Casually speaking, principal component analysis aims to project the original data to a lower-dimensional space while preserving as much information as possible. The following theorem provides a method on solving the PCA problem via singular value decomposition.

Theorem 4.10. *The PCA solution $P^* \in \mathcal{P}_d$ can be decomposed as*

$$P^* = U_d U_d^\mathsf{T},$$

where $U_d \in \mathbb{R}^{n\times d}$ is the matrix formed by the first d left singular vectors of W.

Proof. Let $P \in \mathcal{P}_d$ be arbitrary. By the linearity of the trace, we observe that

$$\begin{aligned}\|PW - W\|_F^2 &= \mathrm{Tr}\big[(PW-W)^\mathsf{T}(PW-W)\big] \\ &= \mathrm{Tr}\big[W^\mathsf{T}P^2W - 2W^\mathsf{T}PW + W^\mathsf{T}W\big] \\ &= \mathrm{Tr}\big[W^\mathsf{T}W\big] - \mathrm{Tr}\big[W^\mathsf{T}PW\big],\end{aligned}$$

since $P^\mathsf{T} = P = P^2$ by definition. Using that $\mathrm{Tr}\big[W^\mathsf{T}W\big]$ is independent of P, we derive

$$\operatorname*{arg\,min}_{P\in\mathcal{P}_d} = \|PW - W\|_F^2 = \operatorname*{arg\,max}_{P\in\mathcal{P}_d} \mathrm{Tr}\big[W^\mathsf{T}PW\big].$$

By the proof of lemma 4.8, there exists an orthonormal basis $Q = [q_1, \dots, q_d] \in \mathbb{R}^{n\times d}$, such that P can be decomposed as $P = QQ^\mathsf{T}$. Using the invariance of the trace under cyclic permutation and the orthonormality of Q, we have

$$\mathrm{Tr}\big[W^\mathsf{T}PW\big] = \mathrm{Tr}\big[W^\mathsf{T}QQ^\mathsf{T}W\big] = \mathrm{Tr}\big[Q^\mathsf{T}WW^\mathsf{T}Q\big] = \sum_{i=1}^{d} q_i^\mathsf{T}WW^\mathsf{T}q_i.$$

To maximize the rightmost expression we perform SVD on $W = U\Sigma V^\mathsf{T}$, such that

$$WW^\mathsf{T} = U\Sigma V^\mathsf{T}V\Sigma^\mathsf{T}U^\mathsf{T} = U\Sigma\Sigma^\mathsf{T}U^\mathsf{T}.$$

Using the notation $Z = U^\mathsf{T}QQ^\mathsf{T}U$ we derive by the invariance under cyclic permutation, that

$$\mathrm{Tr}\big[Q^\mathsf{T}WW^\mathsf{T}Q\big] = \mathrm{Tr}\big[Q^\mathsf{T}U\Sigma\Sigma^\mathsf{T}U^\mathsf{T}Q\big] = \mathrm{Tr}\big[U^\mathsf{T}QQ^\mathsf{T}U\Sigma\Sigma^\mathsf{T}\big] = \mathrm{Tr}\big[Z\Sigma\Sigma^\mathsf{T}\big].$$

As Z is orthogonal as a product of three orthogonal matrices we conclude (why orthogonal?)

$$\mathrm{Tr}\big[W^\mathsf{T}PW\big] = \mathrm{Tr}\big[Z\Sigma\Sigma^\mathsf{T}\big] = \sum_{i=1}^{d} z_{i,i}\sigma_i^2 \leqslant \sum_{i=1}^{d}\sigma_i^2.$$

The upper bound is attained in the case where $Q = U_d = [u_1, \dots, u_d]$ contains the first d columns of U, such that based on corollary 4.3 we can use $W^\mathsf{T}u_i = \sigma_i v_i$ to conclude

$$\mathrm{Tr}\big[W^\mathsf{T}PW\big] = \sum_{i=1}^{d} q_i^\mathsf{T}WW^\mathsf{T}q_i = \sum_{i=1}^{d} u_i^\mathsf{T}WW^\mathsf{T}u_i = \sum_{i=1}^{d} u_i^\mathsf{T}Wv_i\sigma_i = \sum_{i=1}^{d}\sigma_i^2.$$

This completes the proof. □

Using principal component analysis we can finally construct an effective method for dimensionality reduction.

4.3.3 Algorithm

To find the lower-dimensional subspace, that approximately covers the optimization trajectory we first need to sample $t \in \mathbb{N}$ parameter vectors along the optimization trajectory, which can be achieved by training the network with one of the proposed algorithms in section 2.2.

The idea of dimensionality reduction consists of applying PCA to the parameters $\{w_1, \dots, w_t\}$. For this we need to centralize these samples as $\bar{w} \coloneqq \sum_{i=1}^{t} w_i$ and define

$$W \coloneqq \big[w_1 - \bar{w}, \dots, w_t - \bar{w}\big] \in \mathbb{R}^{n \times t}.$$

This enables us to apply PCA on the mean-centered data matrix W. As seen in section 4.3.2, for $d \in \mathbb{N}$, the orthogonal projection on the $d-$dimensional subspace, that covers the parameter trajectory best, is induced by

$$P^* \coloneqq \operatorname*{arg\,min}_{P \in \mathcal{P}_d} \big\|PW - W\big\|_F^2.$$

By theorem 4.10 we can construct P^* from the left singular vectors of W. For this we apply corollary 4.3 to compute the right singular vectors v_i for $i = 1, \dots, d$ by the spectral decomposition of $W^\mathsf{T} W \in \mathbb{R}^{t \times t}$. These can then be used to derive the left singular vectors

$$u_i = \frac{1}{\sigma_i} W v_i, \quad i = 1, \dots, d.$$

Thus we have found an orthonormal basis $U_d = [u_1, \dots, u_d]$ of the d-dimensional subspace, that covers the optimization trajectory best. The orthogonal projection onto this subspace is given by

$$P : \mathbb{R}^n \to \mathbb{R}^n : w \mapsto U_d U_d^\mathsf{T} w.$$

In summary, the algorithm for dimensionality reduction can be notated as follows.

Algorithm 5 Dynamic Linear Dimensionality Reduction (DLDR)

1: Sample parameter trajectory $\{w_1, \dots, w_t\}$ along the training.

2: Compute the mean $\bar{w} \coloneqq \sum_{i=1}^{t} w_i$.

3: Centralize the samples as $W = [w_1 - \bar{w}, \dots, w_t - \bar{w}]$.

4: Perform spectral decomposition such that $W^\mathsf{T} W = V \Sigma^2 V^{-1}$.

5: Choose suitable subspace dimension $d \in \mathbb{N}$.

6: Obtain the d largest eigenvalues $\big[\sigma_1^2, \dots, \sigma_d^2\big]$ with eigenvectors $\big[v_1, \dots, v_d\big]$.

7: Determine the singular vectors $u_i = 1/\sigma_i W v_i$ for $i = 1, \dots, d$.

8: Return the orthonormal basis $\big[u_1, \dots, u_d\big]$.

For complexity ...

5 Numerical Experiments

6 Conclusion

In conclusion...

7 Appendix

Definition 7.1. *A real vector space $\mathcal{H}$ is called a Hilbert space, if there exists an inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$, such that H is complete with respect to the norm $\| \cdot \| \coloneqq \sqrt{\langle \cdot, \cdot \rangle}$.*

Theorem 7.2. *Every Hilbert space $\mathcal{H}$ is isometric and isomorphic to its dual space $\mathcal{H}^*$ via*

$$J : \mathcal{H} \to \mathcal{H}^* : x \mapsto \langle x, \cdot \rangle.$$

Therefore each $\hat{x} \in \mathcal{H}^*$ can be represented as $\hat{x} = J(x)$ for some $x \in \mathcal{H}$.

Proof. Let $f, g \in \mathcal{F}$ and $\mu \coloneqq J(f)$. By the Cauchy-Schwarz inequality it holds that

$$|\mu[g]| = \langle f, g \rangle \leqslant \|f\|_{\mathcal{D}_x} \|g\|_{\mathcal{D}_x} \implies \|\mu\|_{\mathcal{F}^*} \leqslant \|f\|_{D_x}.$$

Thus $\mu \in \mathcal{F}^*$ and therefore J is well defined. Furthermore it holds that

$$|\mu[f]| = \langle f, f \rangle = \|f\|_{\mathcal{D}_x}^2 \implies \|\mu\|_{\mathcal{F}^*} \geqslant \|f\|_{D_x}.$$

The combination of both inequalities yields

$$\|J(f)\|_{\mathcal{F}^*} = \|\mu\|_{\mathcal{F}^*} = \|f\|_{D_x},$$

proving that J is indeed an isometry between $\mathcal{F}$ and its dual $\mathcal{F}^*$. Therefore J is injective and it only remains to prove the surjectivity. □

Corollary 7.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable with global minimizer $x_* \in \mathbb{R}^n$, such that the gradient ∇f is Lipschitz continuous with Lipschitz constant $L > 0$, then*

$$\forall x \in \mathbb{R}^n : f(x) - f(x_*) \geqslant \frac{1}{2L} \|\nabla f(x)\|_2^2.$$

Proof. Let $x \in \mathbb{R}^n$ be arbitrary. Using $y = x - 1/L \cdot \nabla f(x)$ in lemma 2.8 yields

$$\begin{aligned} f(y) &\leqslant f(x) + \left\langle \nabla f(x), -\frac{1}{L} \nabla f(x) \right\rangle + \frac{L}{2} \left\| \frac{1}{L} \nabla f(x) \right\|_2^2 \\ &= f(x) - \frac{1}{L} \|\nabla f(x)\|_2^2 + \frac{1}{2L} \|\nabla f(x)\|_2^2 \\ &= f(x) - \frac{1}{2L} \|\nabla f(x)\|_2^2. \end{aligned}$$

Taking the infimum on the lefthand side results in

$$f(x_*) = \inf_{y \in \mathbb{R}^n} f(y) \leqslant f(x) - \frac{1}{2L} \|\nabla f(x)\|_2^2.$$

Rearranging the inequality completes the proof. □

Lemma 7.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable, such that the gradient ∇f is Lipschitz continuous with Lipschitz constant $L > 0$, then*

$$\forall x, y \in \mathbb{R}^n : \left\langle \nabla f(y) - \nabla f(x), y - x \right\rangle \geqslant \frac{1}{L} \|\nabla f(y) - \nabla f(x)\|_2^2.$$

Proof. Let $x, y \in \mathbb{R}^n$ be arbitrary. Define the differentiable functions

$$g_x : \mathbb{R}^n \to \mathbb{R} : z \mapsto f(z) - \big\langle \nabla f(x), z \big\rangle,$$
$$g_y : \mathbb{R}^n \to \mathbb{R} : z \mapsto f(z) - \big\langle \nabla f(y), z \big\rangle,$$

then g_x is minimized by x and g_y is minimized by y since the gradients are given by

$$\nabla g_x(z) = \nabla f(z) - \nabla f(x),$$
$$\nabla g_y(z) = \nabla f(z) - \nabla f(y).$$

Thus we can apply corollary 7.3 to g_x and derive

$$f(y) - f(x) - \big\langle \nabla f(x), y - x \big\rangle = g_x(y) - g_x(x) \geqslant \frac{1}{2L} \big\| \nabla g_x(y) \big\|_2^2.$$

Similarly, applying corollary 7.3 to g_y results in

$$f(x) - f(y) - \big\langle \nabla f(y), x - y \big\rangle = g_y(x) - g_y(y) \geqslant \frac{1}{2L} \big\| \nabla g_y(x) \big\|_2^2.$$

Since $\big\| \nabla g_x(y) \big\|_2^2 = \big\| \nabla f(y) - \nabla f(x) \big\|_2^2 = \big\| \nabla g_y(x) \big\|_2^2$, summation of the inequalities yields

$$\big\langle \nabla f(y) - \nabla f(x), y - x \big\rangle \geqslant \frac{1}{L} \big\| \nabla f(y) - \nabla f(x) \big\|_2^2,$$

where we have used $-\big\langle \nabla f(y), x - y \big\rangle = \big\langle \nabla f(y), y - x \big\rangle$. This completes the proof. □

Theorem 7.5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex and differentiable. Under the assumption that ∇f is Lipschitz continuous with Lipschitz constant $L > 0$, the gradient descent algorithm with $\epsilon = 0$ and $\alpha_k = 1/L$ for every $k \in \mathbb{N}_0$ produces a sequence $(x_k)_{k \in \mathbb{N}_0} \in \mathbb{R}^n$, such that*

$$\forall k \in \mathbb{N} : f(x_k) - f(x_*) \leqslant \frac{2L \big\| x_0 - x_* \big\|_2^2}{k}.$$

Proof. Let $k \in \mathbb{N}$ be arbitrary. Applying lemma 7.4 to x_k and x_* provides

$$\big\langle \nabla f(x_k) - \nabla f(x_*), x_k - x_* \big\rangle \geqslant \frac{1}{L} \big\| \nabla f(x_k) - \nabla f(x_*) \big\|_2^2.$$

Using $\nabla f(x_*) = 0$ and the symmetry of the scalar product, this is equivalent to

$$\big\langle x_k - x_*, \nabla f(x_k) \big\rangle \geqslant \frac{1}{L} \big\| \nabla f(x_k) \big\|_2^2.$$

Together with $x_{k+1} = x_k - 1/L \cdot \nabla f(x_k)$ by gradient descent, this can be used to show

$$\begin{aligned} \big\| x_{k+1} - x_* \big\|_2^2 &= \big\| x_k - x_* - \frac{1}{L} \nabla f(x_k) \big\|_2^2 \\ &= \big\| x_k - x_* \big\|_2^2 - 2 \frac{1}{L} \big\langle x_k - x_*, \nabla f(x_k) \big\rangle + \frac{1}{L^2} \big\| \nabla f(x_k) \big\|_2^2 \\ &\leqslant \big\| x_k - x_* \big\|_2^2 - \frac{2}{L} \cdot \frac{1}{L} \big\| \nabla f(x_k) \big\|_2^2 + \frac{1}{L^2} \big\| \nabla f(x_k) \big\|_2^2 \\ &= \big\| x_k - x_* \big\|_2^2 - \frac{1}{L^2} \big\| \nabla f(x_k) \big\|_2^2, \end{aligned} \tag{1}$$

which implies $\|x_k - x_*\|_2^2$ is monotonically decreasing in k. Furthermore lemma 2.8 yields

$$f(x_{k+1}) \leqslant f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2}\|x_{k+1} - x_k\|_2^2.$$

By the definition of gradient descent, we can plug in $x_{k+1} - x_k = -1/L \cdot \nabla f(x_k)$ to derive

$$\begin{aligned} f(x_{k+1}) &\leqslant f(x_k) + \langle \nabla f(x_k), -\frac{1}{L}\nabla f(x_k) \rangle + \frac{L}{2}\|\frac{1}{L}\nabla f(x_k)\|_2^2 \\ &= f(x_k) - \frac{1}{L}\|\nabla f(x_k)\|_2^2 + \frac{1}{2L}\|\nabla f(x_k)\|_2^2 \\ &= f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|_2^2. \end{aligned} \tag{2}$$

Hence the gradient descent algorithm with constant step size $\alpha_k = 1/L$ guarantees to make progress unless $\nabla f(x_k) \neq 0$. On top of this we can infer by the convexity of f, that

$$f(x_k) - f(x_*) \leqslant \langle \nabla f(x_k), x_k - x_* \rangle.$$

Using the Cauchy-Schwarz inequality and the monotonicity of $\|x_k - x_*\|_2^2$ by (1) yields

$$f(x_k) - f(x_*) \leqslant \langle \nabla f(x_k), x_k - x_* \rangle \leqslant \|\nabla f(x_k)\|_2 \cdot \|x_k - x_*\|_2 \leqslant \|\nabla f(x_k)\|_2 \cdot \|x_0 - x_*\|_2,$$

which is equivalent to

$$\|\nabla f(x_k)\|_2 \geqslant \frac{1}{\|x_0 - x_*\|_2}\big(f(x_k) - f(x_*)\big). \tag{3}$$

Inserting the bound (3) in (2) results in

$$f(x_{k+1}) \leqslant f(x_k) - \frac{1}{2L}\frac{1}{\|x_0 - x_*\|_2^2}\big(f(x_k) - f(x_*)\big)^2.$$

Subtracting $f(x_*)$ on both sides and using the notation $\Delta_k \coloneqq f(x_k) - f(x_*)$ yields

$$\Delta_{k+1} \leqslant \Delta_k - \frac{1}{2L}\frac{1}{\|x_0 - x_*\|_2^2}\Delta_k^2 \;\Leftrightarrow\; \frac{1}{2L}\frac{1}{\|x_0 - x_*\|_2^2}\Delta_k^2 \leqslant \Delta_k - \Delta_{k+1}.$$

Since Δ_k is monotonically decreasing in k by (2), we derive $1 \leqslant \Delta_k/\Delta_{k+1}$. Expanding the inequality with the positive factor $1/\Delta_k\Delta_{k+1}$ therefore provides

$$\frac{1}{2L}\frac{1}{\|x_0 - x_*\|_2^2} \leqslant \frac{1}{2L}\frac{1}{\|x_0 - x_*\|_2^2} \cdot \frac{\Delta_k}{\Delta_{k+1}} \leqslant \frac{1}{\Delta_{k+1}} - \frac{1}{\Delta_k}.$$

Summing up both sides from $i = 0, \ldots, k-1$ yields

$$k \cdot \frac{1}{2L}\frac{1}{\|x_0 - x_*\|_2^2} \leqslant \sum_{i=0}^{k-1}\left[\frac{1}{\Delta_{k+1}} - \frac{1}{\Delta_k}\right] = \frac{1}{\Delta_k} - \frac{1}{\Delta_0} \leqslant \frac{1}{\Delta_k}.$$

Finally we can use $\Delta_k = f(x_k) - f(x_*)$ to conclude

$$\Delta_k = f(x_k) - f(x_*) \leqslant \frac{2L\|x_0 - x_*\|_2^2}{k}.$$

This completes the proof. □

Problem: Maxima in lemma 7.4

References

[1] Tao Li, Lei Tan, Qinghua Tao, Yipeng Liu, Xiaolin Huang. *Low Dimensional Landscape Hypothesis is True: DNNs can be Trained in Tiny Subspaces.* arXiv preprint arXiv:2103.11154, 2021.

[2] Augustin Cauchy. *Méthode générale pour la résolution des systemes d'équations simultanées.* Comptes rendus de l'Académie des sciences, volume 25, pages 536-538, 1847.

[3] Yurii Nesterov. *Introductory Lectures on Convex Optimization.* Applied Optimization, volume 87, 2004.

[4] Richard H. Byrd, Jorge Nocedal, Robert B. Schnabel. *Representations of Quasi-Newton Matrices and their Use in Limited Memory Methods.* Mathematical Programming, volume 60(1), pages 129-156, 1994

[5] Léon Bottou, Frank E. Curtis, Jorge Nocedal. *Optimization Methods for Large-Scale Machine Learning.* SIAM Review, volume 60(2), pages 223-311, 2018.

[6] Diederik P. Kingma, Jimmy Lei Ba. *Adam: A Method for Stochastic Optimization.* International Conference on Learning Representations (ICLR), 2015.

[7] Arthur Jacot, Franck Gabriel, Clément Holger. *Neural Tangent Kernel: Convergence and Generalization in Neural Networks.* Advances in Neural Information Processing Systems, volume 31, pages 8571-8580, 2018.

[8] Zhou Fan, Zhichao Wang. *Spectra of the Conjugate Kernel and Neural Tangent Kernel for Linear-Width Neural Networks.* Advances in Neural Information Processing Systems, volume 33, pages 7710-7721, 2020.

[9] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, Jeffrey Pennington. *Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent.* Journal of Statistical Mechanics: Theory and Experiment. 2020(12):124002, 2020.

[10] Gene H. Golub, Charles F. Van Loan. *Matrix Computations.* Third edition, 1996.

[11] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar. *Foundations of Machine Learning.* Second edition, 2018.