Stochastic Machine Learning
04 - Recap on Deep Learning

Thorsten Schmidt

Abteilung für Mathematische Stochastik

www.stochastik.uni-freiburg.de
thorsten.schmidt@stochastik.uni-freiburg.de

WS 2020/21

We recall.

Definition (Deep network)

A **neural network** is an n-fold composition of simple functions

$$f(x) = f^n \circ \cdots \circ f^1(x) = f^n(f^{n-1}(\cdots f^2(f^1(x))\cdots)).$$

It is called **deep**, if $n \geq 2$. f^k is called the k-the layer of the network.
Each layer is a composition of a non-linear **activation function** σ and an affine function $a + Bx$,

$$f^k(\cdot) = \sigma^k(a^k + B^k\cdot)$$

In this case the network has one input layer, $n-1$ hidden layers and one (f^n) output layers.

Generic Learning algorithm

- We begin with the forward pass: given weights and activation functions, we compute
$$f(x, \theta) = f(x, a^1, B^1, \ldots, a^n, B^n).$$
Activation functions are not optimized, so they do not arise here.
- Compute all partial gradients in the backward-pass and optimize with regard to the loss function: the loss function given target y is denoted by $L(x, \theta) = L(y - f(x, \theta))$ and we compute
$$\partial_{a^1} L, \ldots, \partial_{B^n} L.$$
- Then we update the weights and stop if the target precision is achieved.

Learning vs. pure optimization

- In optimization we are simply interested in minimizing the loss function.
- In learning, we rather want to achieve a good generalization, thus we want to minimize the loss on a data set which we de not have at hand!

Definition

Gradient descent For a generic function $F : \mathbb{R}^n \to \mathbb{R}$, the **gradient descent algorithm** with learning rate α proceeds via

$$\theta_{n+1} = \theta_n - \alpha \nabla F(\theta_n).$$

The sequence $\theta_0, \theta_1, \ldots$ convergence to a local minimum. If F is convex, this is also a global minimum. This is why convex optimization is much easier compared to more general problems.

Ill-conditioned

- Define the condition number $\kappa(Q)$ of a matrix as the quotient of the largest over the smallest eigenvalue.
- Considering $F(\theta) = \frac{1}{2}\theta^\top Q\theta$, the **contraction rate** of gradient descent is
$$\| \theta_{n+1} - \theta^* \| \leq \frac{\kappa(Q) - 1}{\kappa(Q) + 1} \| \theta_n - \theta^* \|$$
when using the optimal learning rate $\alpha^* = 2/(\lambda_{\max} - \lambda_{\min})$.
- If the problem is ill-conditioned, a zig-zag behaviour occurs, which can be improved by
- Momentum:
$$\theta_{n+1} = \theta_n - \alpha \nabla f(\theta_n) - \beta(\theta_n - \theta_{n-1}).$$
The contraction rate is
$$\| \theta_{n+1} - \theta^* \| \leq \frac{\sqrt{\kappa(Q)} - 1}{\sqrt{\kappa(Q)} + 1} \| \theta_n - \theta^* \|,$$
using optimal α and β.
- This can be improved using **pre-conditioning**,
$$\theta_{n+1} = \theta_n - \alpha D_n \nabla F(\theta_n)$$
with optimal $D_n = \nabla^2 F(\theta_n)^{-1} = Q^{-1}$.

Stochastic gradient descent

- However, in deep learning many problems arise which brings stochastic gradient descent (SGD) on the plan.
- Goal: minimize the empirical loss

$$L(\theta) := \frac{1}{n} \sum_{i=1}^{n} L(f(x^i, \theta), y^i)$$

- In SGD, we sample a **mini-batch** $(\tilde{x}^1, \tilde{y}^1), \dots, (\tilde{x}^m, \tilde{y}^m)$ from the data (x^i, y^i), $i = 1, \dots, n$ and update as

$$\theta_{n+1} = \theta_n - \alpha \nabla_\theta \frac{1}{m} \sum_{i=1}^{m} L(f(\tilde{x}^i, \theta), \tilde{y}^i).$$

  Each of this step is called an **epoch**. In stochastic gradient descent with $m = 1$, epoch is used differently: here an epoch is one full sweep through the data $1, \dots, n$.
- Recall: do not choose the batch size too small, rather as large as possible. And: smaller batchsize requires lower learning rates.
- How do we pick the best learning rate in practice?
- Now it is time to experiment with different adaptive gradient descent algorithms.

Quick questions

- What is gradient descent?
- What problems may we experience during gradient descent?
- What can you say about convergence rates?
- Why do we use stochastic gradient descent ?
- What are mini-batches and how should the learning rate be chosen?
- Why should we use adaptive gradient descent algorithms?

What to do today?

- Look a little bit around what optimizers can be used and how they perform.
- We have used `rmsprop` - compare this to `Adam`. Also look up `Adagrad`.
- What are differences and when should the one preferred over the other?
- Try to find better result for the MNIST database.
- Try the NIST database or other sources (see the first lecture for references and links).