

Introduction to Machine Learning

Chapter 4: Deep Learning- Training

Bernd Bischl

Department of Statistics – LMU Munich

Winter term 2021



GRADIENT DESCENT

- Let $\mathcal{R} : \mathbb{R}^p \rightarrow \mathbb{R}$ be an arbitrary, differentiable, unrestricted function (of $\theta \in \mathbb{R}^p$).

In the context of deep learning, \mathcal{R} represents the empirical risk function and θ represents the weights (and biases) of the network. For simplification, we will assume $\theta = (\theta_1, \dots, \theta_m)$.

- We want to minimize this function by gradient descent (GD).
- The negative gradient

$$-\mathbf{g} = -\nabla \mathcal{R}(\theta) = -\left(\frac{\partial \mathcal{R}}{\partial \theta_1}, \dots, \frac{\partial \mathcal{R}}{\partial \theta_m}\right)^\top$$

points in the direction of the **steepest descent**.

GRADIENT DESCENT

- “Standing” at a point $\theta^{[t]}$ during minimization, we improve by performing the following update:

$$\theta^{[t+1]} = \theta^{[t]} - \alpha \nabla \mathcal{R}(\theta^{[t]}),$$

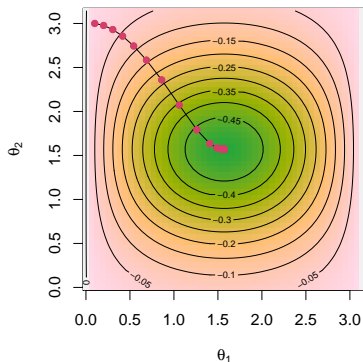
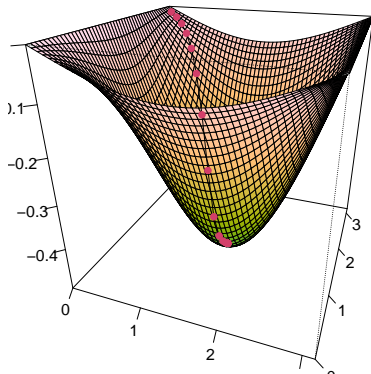
which implies (for sufficiently small α),

$$\mathcal{R}(\theta^{[t+1]}) \leq \mathcal{R}(\theta^{[t]})$$

- α determines the length of the step and is called **step size** or, in risk minimization, **learning rate**.

EXAMPLE: GRADIENT DESCENT

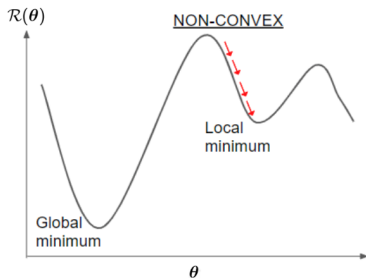
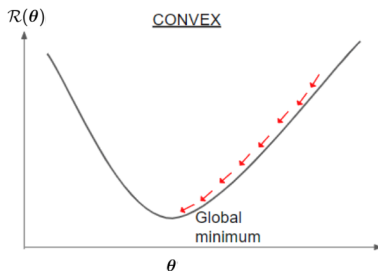
$$\mathcal{R}(\theta_1, \theta_2) = -\sin(\theta_1) \cdot \frac{1}{2\pi} \exp((\theta_2 - \pi/2)^2)$$



"Walking down the hill, towards the valley."

GRADIENT DESCENT AND OPTIMALITY

- GD is a greedy algorithm: In every iteration, it makes locally optimal moves.
- If $\mathcal{R}(\theta)$ is **convex** and **differentiable**, and its gradient is Lipschitz continuous, GD is guaranteed to converge to the global minimum (for small enough step-size).
- However, if $\mathcal{R}(\theta)$ has multiple local optima and/or saddle points, GD might only converge to a stationary point (other than the global optimum), depending on the starting point.



GRADIENT DESCENT AND OPTIMALITY

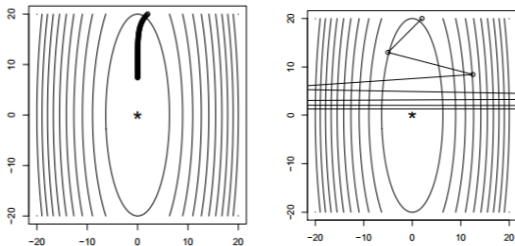
Note: It might not be that bad if we do not find the global optimum:

- We do not optimize the actual quantity of interest, i.e. the (theoretical) risk, but only an approximate version, i.e. the empirical risk.
- If the model class is very flexible, it might be disadvantageous to optimize too aggressively and increase the risk of overfitting.
- Early-stopping the optimization might even increase generalization performance.

LEARNING RATE

The step-size α plays a key role in the convergence of the algorithm. If

the step size is too small, the training process may converge **very** slowly (see left image). If the step size is too large, the process may not converge, because it **jumps** around the optimal point (see right image).

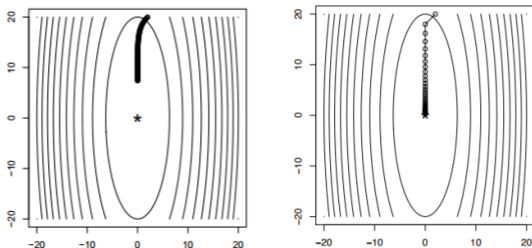


LEARNING RATE

So far we have assumed a fixed value of α in every iteration:

$$\alpha^{[t]} = \alpha \quad \forall t = \{1, \dots, T\}$$

However, it makes sense to adapt α in every iteration:



Steps of gradient descent for $\mathcal{R}(\theta) = 10\theta_1^2 + 0.5\theta_2^2$. Left: 100 steps for with a fixed learning rate. Right: 40 steps with an adaptive learning rate.

STOCHASTIC GRADIENT DESCENT

Let us consider GD for empirical risk minimization. The updates are:

$$\boldsymbol{\theta}^{[t+1]} = \boldsymbol{\theta}^{[t]} - \alpha \cdot \frac{1}{n} \cdot \sum_{i=1}^n \nabla_{\boldsymbol{\theta}} L\left(y^{(i)}, f(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}^{[t]})\right)$$

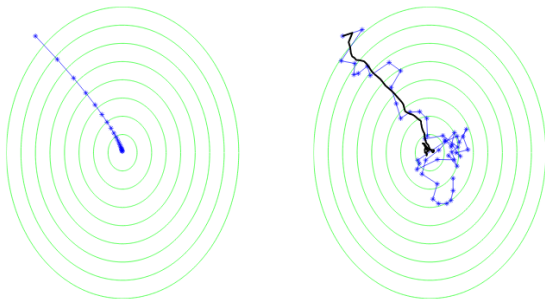
- Optimization algorithms that use the entire training set to compute updates in one huge step are called **batch** or **deterministic**. This is computationally very costly or often impossible.
 - **Idea:** Instead of letting the sum run over the whole dataset (**batch mode**) one can also let it run only over small subsets (**minibatches**), or only over a single example i .
 - If the index i of the training example is a random variable with uniform distribution, then its expectation is the batch gradient $\nabla_{\boldsymbol{\theta}} \mathcal{R}_{\text{emp}}(\boldsymbol{\theta})$
- We have a **stochastic**, noisy version of the batch gradient

STOCHASTIC GRADIENT DESCENT

- The gradient w.r.t. a single training observation is fast to compute but not reliable. It can be used simply as a computational trick to deal with large data or to operate on real streams of online data in online learning.
- In contrast, the full batch gradient is costly (or even impossible, e.g., when data does not even fit into memory) to compute, particularly in DL, but it averages out all the noise from sub-sampling.
- Minibatches are in between. The batch size decides upon the compromise between speed and averaging (smoothing).
- In summary: SGD computes an unbiased estimate of the gradient by taking the average gradient over a minibatch (or one sample) to update the parameter θ in this direction.

STOCHASTIC GRADIENT DESCENT

An illustration of the SGD algorithm (to minimize the function $1.25(x_1 + 6)^2 + (x_2 - 8)^2$).



source : Shalev-Shwartz and Ben-David. Understanding machine learning: From theory to algorithms. Cambridge University Press, 2014.

Figure: On the left is GD and on the right is SGD. The black line depicts the averaged value of θ .

STOCHASTIC GRADIENT DESCENT

Algorithm Basic SGD pseudo code

```
1: Initialize parameter vector  $\theta^{[0]}$ 
2:  $t \leftarrow 0$ 
3: while stopping criterion not met do
4:   Randomly shuffle data and partition into minibatches  $J_1, \dots, J_K$  of size  $m$ 
5:   for  $k \in \{1, \dots, K\}$  do
6:      $t \leftarrow t + 1$ 
7:     Compute gradient estimate with  $J_k$ :  $\hat{g}^{[t]} \leftarrow \frac{1}{m} \sum_{i \in J_k} \nabla_{\theta} L(y^{(i)}, f(\mathbf{x}^{(i)} | \theta^{[t-1]}))$ 
8:     Apply update:  $\theta^{[t]} \leftarrow \theta^{[t-1]} - \alpha \hat{g}^{[t]}$ 
9:   end for
10: end while
```

STOCHASTIC GRADIENT DESCENT

- With minibatches of size m , a full pass over the training set (called an **epoch**) consists of $\frac{n}{m}$ gradient updates.
- SGD and its modifications are the most used optimization algorithms for ML in general and for deep learning in particular.
- SGD (with one or a few samples per batch) updates have a high variance, even though they are unbiased. Because of this variance, the learning rate α is typically much smaller than in the full-batch scenario.

STOCHASTIC GRADIENT DESCENT

- When the learning rate is slowly decreased, SGD converges to a local minimum.
- SGD with minibatches reduces the variance of the parameter updates and utilizes highly optimized matrix operations to efficiently compute gradients.
- Minibatch sizes are typically between 50 and 1000.
- Recent results indicate, that SGD often leads to better generalizing models than GD, and thus may perform some kind of indirect regularization.