# Gradient Descent

Johanni Brea

Introduction to Machine Learning

# Optimization in Machine Learning

- For linear regression there is an analytical solution that minimizes the RMSE.
- For logistic regression (and most other methods) there is no analytical solution.
- For many models there are specialized optimizers.
- There is a course at EPFL on Optimization for machine learning
  `https://edu.epfl.ch/coursebook/en/optimization-for-machine-learning-CS-439`
- A simple optimizer that works usually well for parametric models is

  **gradient descent**.

# Table of Contents

# Gradient Descent

1. Input: loss function $L$, initial guess $\beta^{(0)} = \left(\beta_0^{(0)}, \ldots, \beta_p^{(0)}\right)$ learning rate $\eta$, maximal number of steps $T$.
2. For $t = 1, \ldots, T$
   - $\delta_i = \eta \dfrac{\partial L}{\partial \beta_i}\left(\beta^{(t-1)}\right)$
   - $\beta_i^{(t)} = \beta_i^{(t-1)} - \delta_i$
3. Return $\beta^{(T)}$

Automatic Differentiation software uses the chain rule and symbolic derivatives for primitive functions, to compute the derivative of almost any code we write.

# Practical Considerations

- Choosing a good learning rate can be tricky.
- Scaling the loss function has an impact on gradient descent.
  It is e.g. advisable to have $L$ independent of the size of the data set, e.g. replace $L = \sum_{i=1}^{n}(y_i - \beta x_i)^2$ by $L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta x_i)^2$.
- Additive constants in the loss function $L$ that do not depend on the parameters have no impact on gradient descent; they are often removed from the loss function.
- Preprocessing the input and output may have a strong effect on gradient descent. There are domain-specific “best preprocessing practices” (e.g. for images or audio). Standardizing inputs (and outputs in the case of regression) is an option.

# Table of Contents

# Convex and Non-Convex Loss Functions

**Globally Convex Loss Function**

Loss function has a unique global minimum

From any initial condition there is a path towards the global minimum along which the loss is monotonically decreasing. The same solution is found by gradient descent independently of the initial condition.

The loss function of (multiple) (logistic) linear regression (with L1 or L2 regularization) is globally convex.

**Non-Convex Loss Function**

Loss function has multiple local minima

The solution of gradient descent depends on the initial condition.

# Table of Contents

# Stochastic Gradient Descent (SGD)

Computing the loss over all samples $1, \ldots, n$ can be computationally costly.
A subset of the training data may be sufficient to estimate the gradient direction.

1. Input: loss function $L$, initial guess
   $\beta^{(0)} = \left(\beta_0^{(0)}, \ldots, \beta_p^{(0)}\right)$
   learning rate $\eta$, maximal number of steps $T$,
   batch size $B$.
2. For $t = 1, \ldots, T$
   - Determine batch of training indices $\mathcal{I}$
   - $\delta_i = \eta \frac{\partial L}{\partial \beta_i}\left(\beta^{(t-1)}; \mathcal{I}\right)$
   - $\beta_i^{(t)} = \beta_i^{(t-1)} - \delta_i$
3. Return $\beta^{(T)}$

where $L(\beta; \mathcal{I})$ is the loss function evaluated on the training samples with indices in $\mathcal{I}$, e.g.

$$L(\beta; \mathcal{I}) = \frac{1}{B} \sum_{i \in \mathcal{I}} \left(y_i - x_i^T \beta\right)^2$$

**Example** $B = 5$

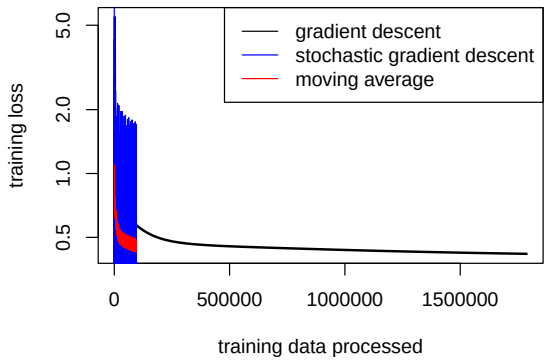
| | batch 1 | batch 2 | batch 3 | b |
|---|---|---|---|---|
| $\mathcal{I}$ | 1 8 3 13 93 | 9 14 2 26 31 | ⋯ | |

# Terminology

- **batchsize** $B$ (hyper-parameter): number of samples in each batch. Sometimes this is also called **minibatch size**.
- **epoch**: collection of update steps that go once through the entire dataset.
- **number of epochs** (hyper-parameter): number of times to go through the entire dataset.

Example: If we choose for a dataset of size $n = 1000$ a batchsize of $B = 20$ and we train for 30 epochs, there are in total $T = n/B \times 30 = 50 \times 30 = 1500$ gradient descent updated steps and each data point contributes 30 times to the computation of partial derivatives.

# Example Learning Curve

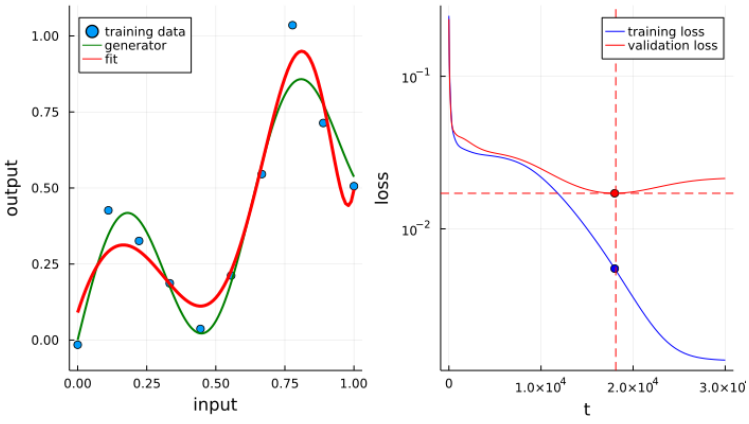



The training loss on batches of size 32 is very variable. But if we look at the moving average over the training loss of 50 subsequent batches, we see that stochastic gradient descent drops to a fairly low loss after processing far less training data than gradient descent.

# Table of Contents

# Early Stopping

Start with small weights and stop gradient descent when validation loss starts to increase.




- At the beginning of gradient descent all parameter values are small.
- Typically, the norm of the parameters increases during gradient descent.
- The parameter values found at early stopping have usually a smaller norm than the parameter values with the lowest training error.

# Quiz

- With a constant learning rate and a non-zero gradient, the training loss in gradient decent is decreasing in every step.
- In every step of gradient descent one can choose a learning rate larger than zero such that the training loss is decreasing, unless the gradient is zero.
- In every step of stochastic gradient descent one can choose a learning rate larger than zero such that the training loss is decreasing, unless the gradient is zero.
- For a training set of size $n$, computing the gradient in (standard) gradient descent takes $n/B$ times longer than computing the gradient in stochastic gradient descent with batch size $B$.
- Models found with early stopping tend to have a lower bias but a higher variance (when fitted on different training sets from the same data generator) than models found without early stopping.

# Suggested Reading

- 10.7 Fitting a Neural Network (skip parts 10.7.1, 10.7.3 and 10.7.4)
- 10.7.2 Regularization and Stochastic Gradient Descent