3.2 Linear Regression – Stochastic Gradient Descent

Lecture based on “Dive into Deep Learning” **http://D2L.AI** (Zhang et al., 2020)

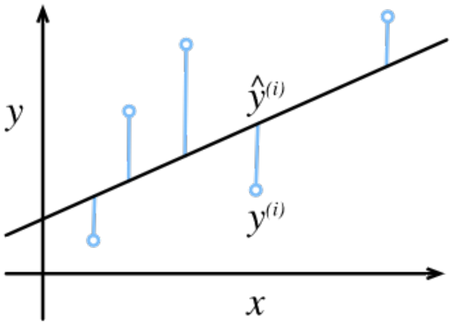
Prof. Dr. Christoph Lippert

Digital Health & Machine Learning

The **loss function** quantifies the distance between the **real** and **predicted** value of the target.

$$l^{(i)}(\mathbf{w}, b) = \frac{1}{2}\left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right)^2.$$

$$L(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n} l^{(i)}(\mathbf{w}, b)$$



$$(\mathbf{w}^*, b^*) = \underset{\mathbf{w}, b}{\operatorname{argmin}}\ L(\mathbf{w}, b)$$

- The **OLS** can be determined analytically.
- An alternative algorithm is **gradient descent**.

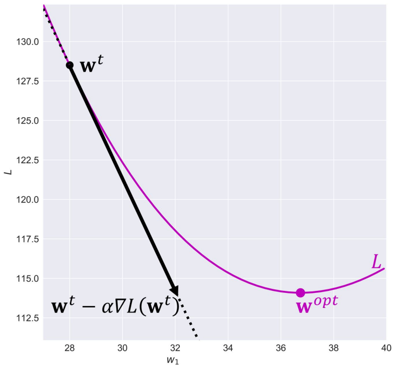
Optimization
Gradient descent

$$(\mathbf{w}^*, b^*) = \underset{\mathbf{w},b}{\operatorname{argmin}}\ L(\mathbf{w}, b)$$

$-\nabla L(\mathbf{w}, b)$ is the direction of **steepest descent**.
Gradient descent update rule:

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \eta \nabla L(\mathbf{w}, b),$$

for a small **stepsize** $\eta > 0$ (e.g., 10^{-3}).



Note:

- $O(n)$ cost for gradient computation.
- Can be very slow for some problems (oscillation).
- Given sufficient time, gradient descient will find the minimum
- The loss of linear regression loss is (strictly) **convex**.

Stochastic Gradient Descent

- Initialize the model parameters (randomly)
- Iterate:
  1. Uniformly sample a mini-batch $\mathcal{B}$ of training examples.
  2. Compute the gradient of the average loss on the mini batch.
  3. update.

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{(\mathbf{w}, b)} l^{(i)}(\mathbf{w}, b)$$

Note:

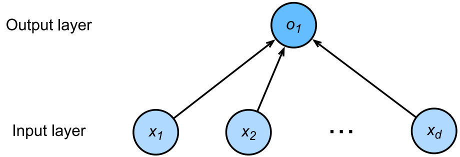
- The batch size and learning rate $\eta > 0$ are **hyper-parameters** that have to be **tuned** by the user.
- Avoids $O(n)$ cost for gradient computation.
- Can still oscillate.
- Never reaches the exact optimum.

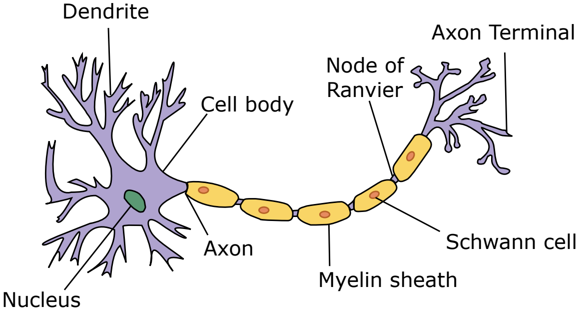
$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{(\mathbf{w}, b)} l^{(i)}(\mathbf{w}, b)$$

For quadratic losses and linear functions we can compute the updates explicitly:

$$\begin{aligned}
\mathbf{w} \leftarrow \mathbf{w} - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_{\mathbf{w}} l^{(i)}(\mathbf{w}, b) &= w - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathbf{x}^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right), \\
b \leftarrow b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \partial_b l^{(i)}(\mathbf{w}, b) &= b - \frac{\eta}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left(\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)}\right).
\end{aligned}$$

A Single-Layer Neural Network.





Linear models have a single neuron.

- d inputs $x_1, x_2, \dots x_d$
- 1 output

All inputs are connected to all outputs.

- **fully-connected layer** or **dense layer**.

Summary

- Linear regression with a squared loss has an analytic solution.
- Gradient descent is an iterative algorithm to minimize a differentiable loss function
  - Each step of gradient descent requires a pass over the whole data set.
  - Gradient descent can be very slow (e.g. oscillation)
- Stochastic gradient descent speeds up gradient calculation, by approximating the gradient on a small random subsample of the training data.
  - Requires hyperparameter tuning to perform well
  - Goto optimization algorithm in deep learning
- Linear models are single layer neural networks.