Introduction to Deep Learning (I2DL)

Exercise 5: Two Layer Network, SGD and optimization

Today's Outline

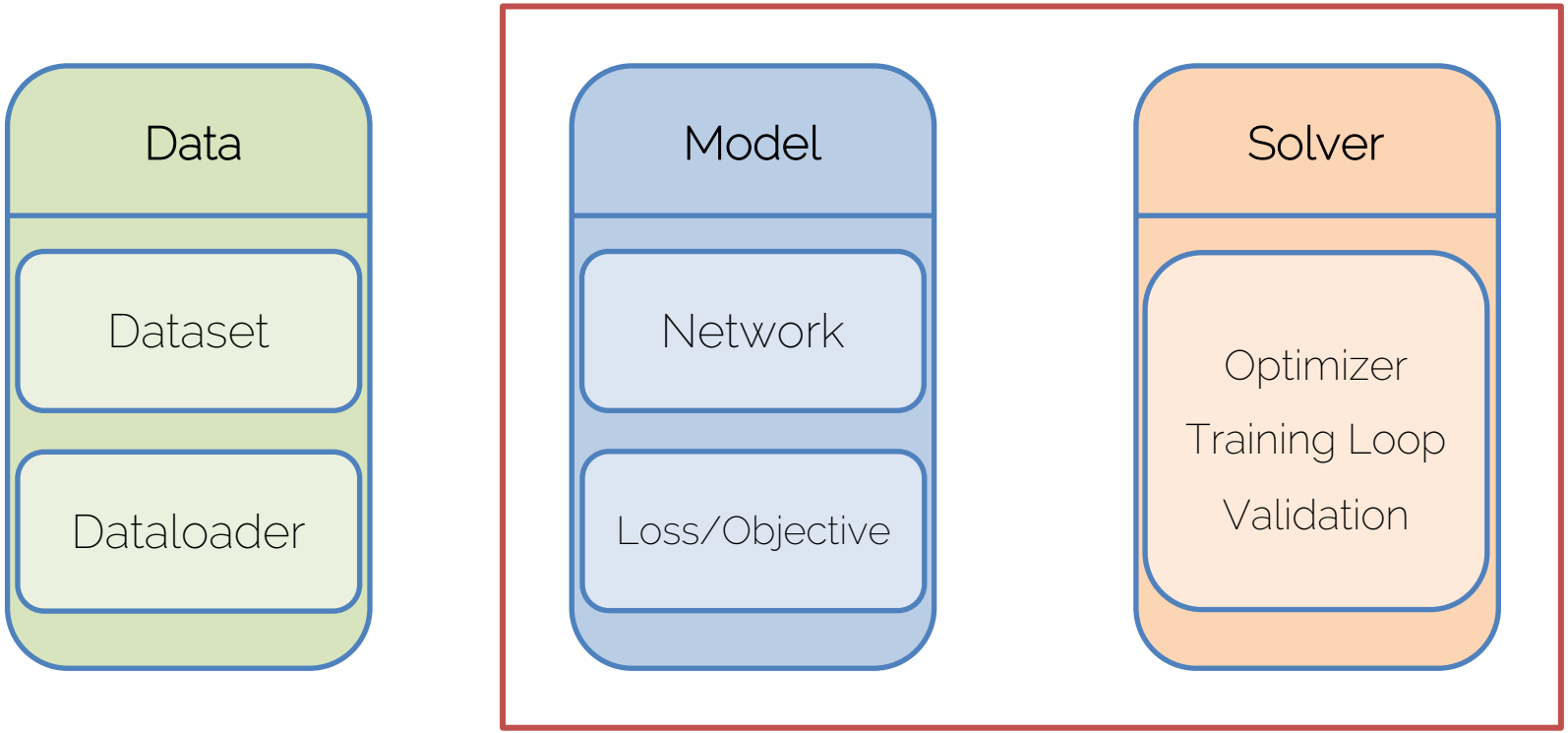
- Submission System Updates
- Two layer neural network
- Submission 5: Two layer network for regression
  - Start: **May 22, 2020 12.00**
  - End: **May 27, 2020 23.59**
- Stochastic Gradient Descent
- Optimization

Submission System

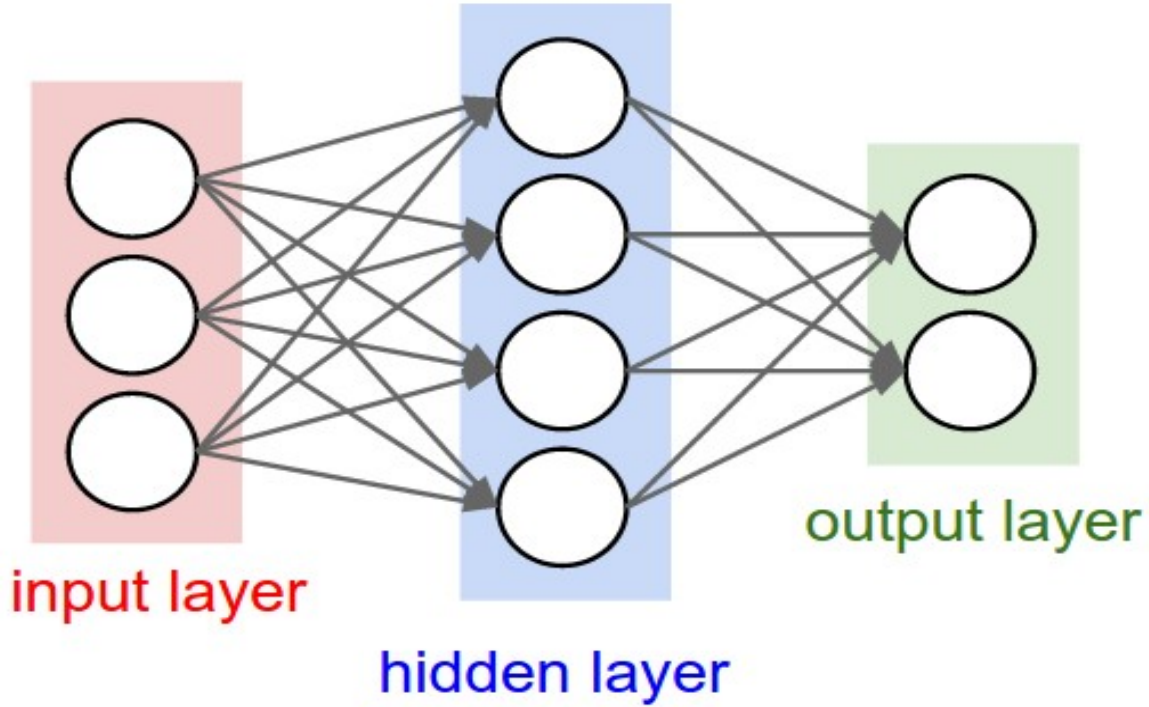
- No submission after deadline
- ~~Submission Limit: max. 5 successful submissions~~
  - (no submission limit!)
- Threshold bug Exercise 4:
  - You receive bonus when you pass 80%

Two Layer Neural Network (NN)

The Pillars of Deep Learning



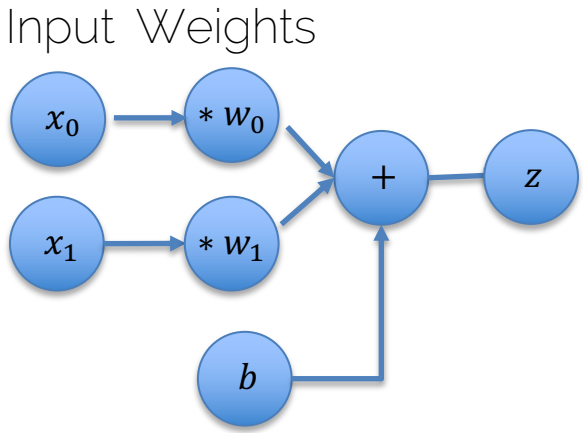
You will build your first NN



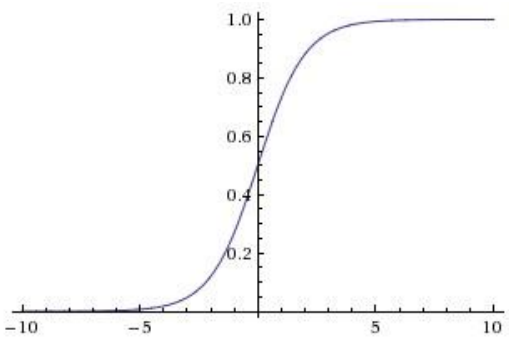
Source: http://cs231n.github.io/neural-networks-1/

You will build your first NN

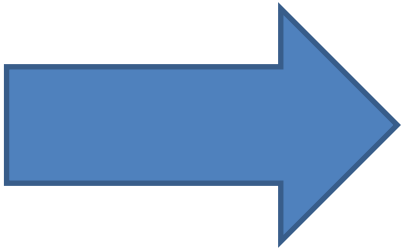
- Affine Layer: $\mathbf{z} = W \cdot x + b$

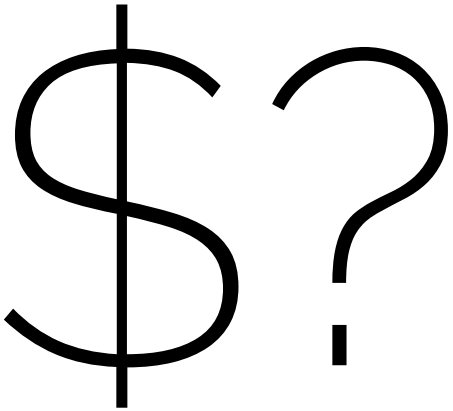


- Activation Function:
  - Sigmoid: $\sigma(z) = \frac{1}{(1+e^{-z})}$



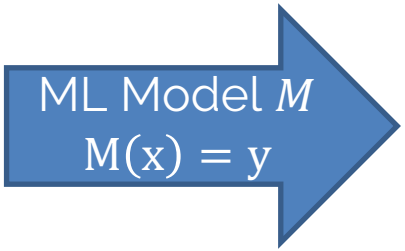
Regression of House Prices

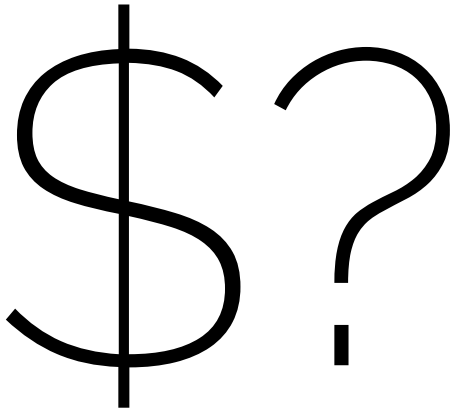




Regression of House Prices

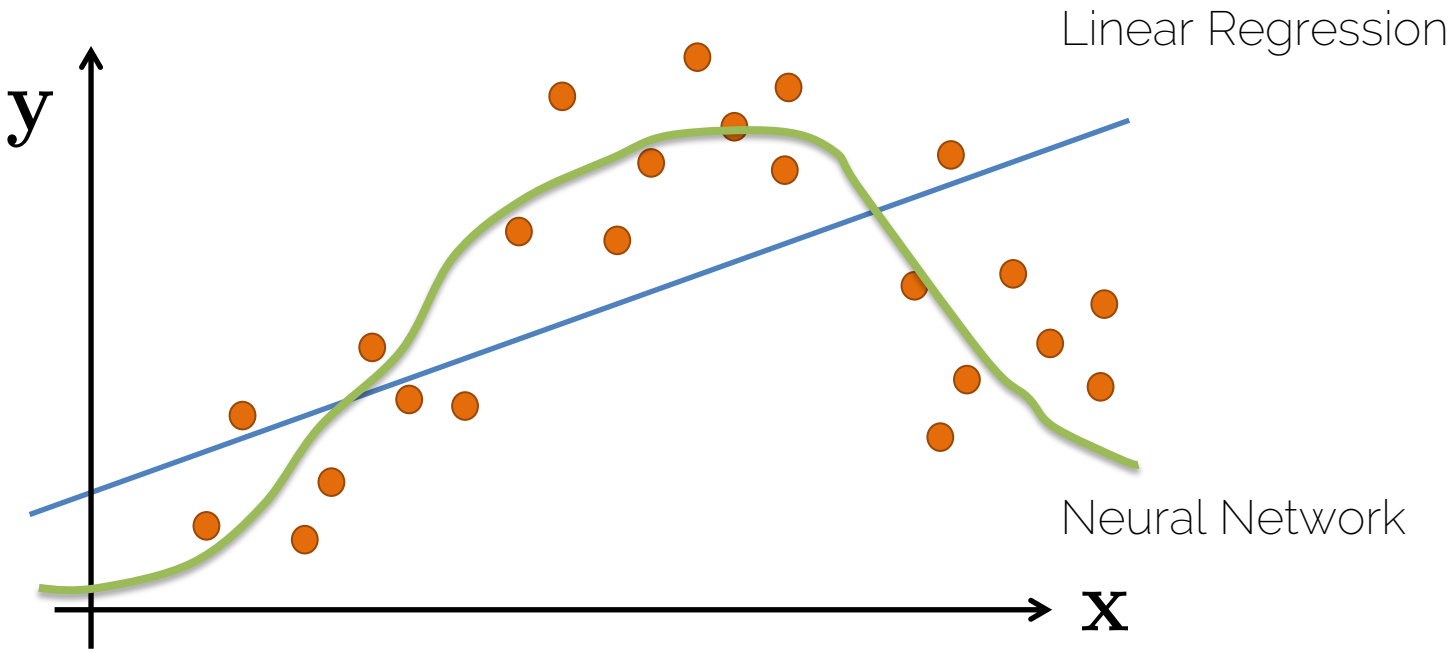
$$\begin{pmatrix} \text{LivingArea} \\ \text{YearBuilt} \\ \text{OutdoorArea} \\ \dots \end{pmatrix}$$



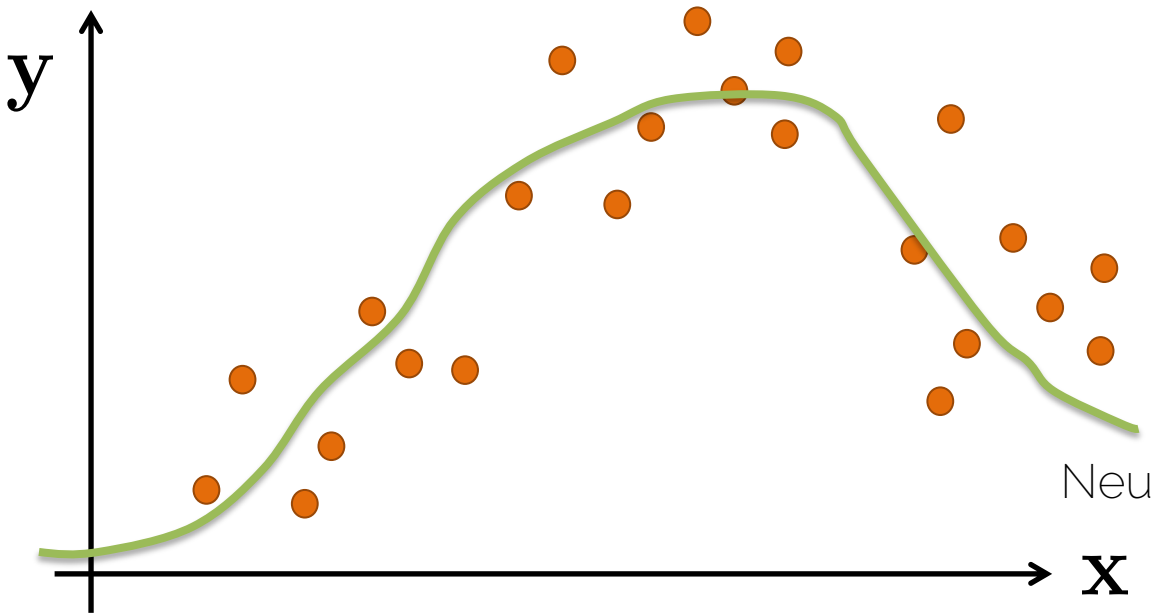


Regression for House Prices

- Find a model that explains a target $\mathbf{y}$ given the inputs $\mathbf{X}$



Regression for House Prices

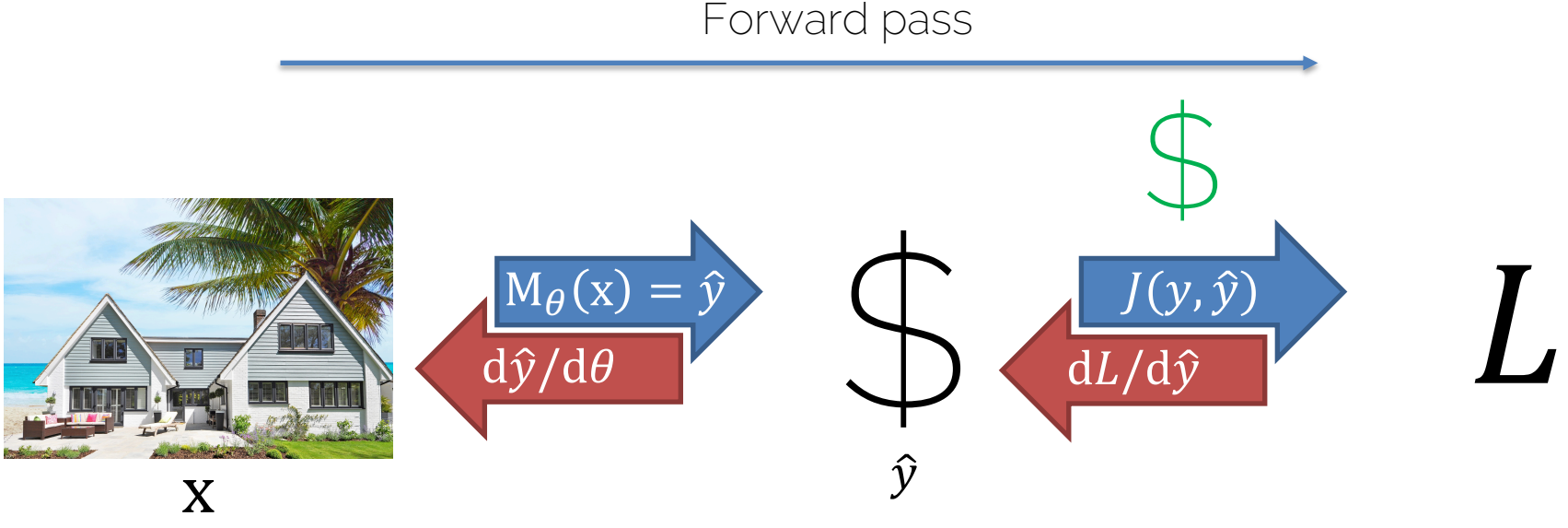


Minimizing

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

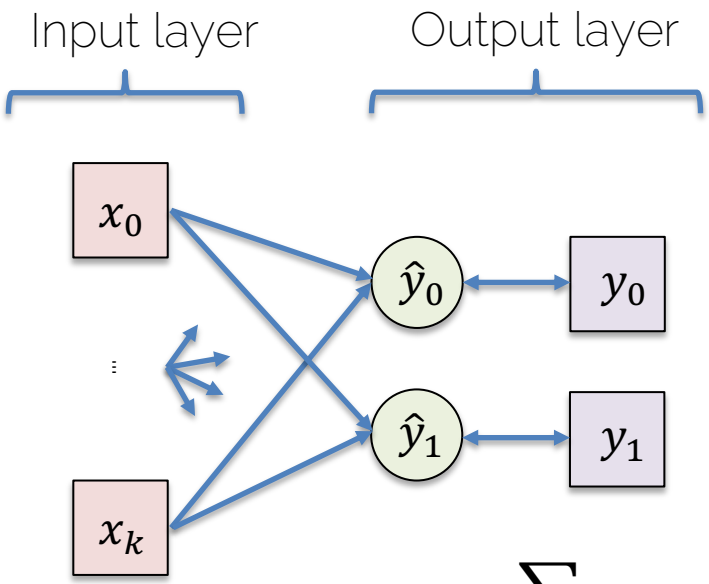
Objective function
Energy
Cost function

Training your Network



Optimization with gradient descent: $\theta_{t+1} = \theta_t - \lambda \cdot \nabla_\theta \mathrm{L}$

Compute Graphs → Neural Networks



$$\hat{y}_i = A(b_i + \sum_k x_k w_{i,k})$$

Activation function

bias

Goal: We want to compute gradients of the loss function L w.r.t. all weights $\boldsymbol{w}$

$$L = \sum_i L_i$$

L: sum over loss per sample, e.g. L2 loss → simply sum up squares:

$$L_i = (\hat{y}_i - y_i)^2$$

→ use chain rule to compute partials

$$\frac{\partial L}{\partial w_{i,k}} = \frac{\partial L}{\partial \hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial w_{i,k}}$$

We want to compute gradients w.r.t. all weights $\boldsymbol{W}$ AND all biases $\boldsymbol{b}$

Submission 5: Two Layer NN

- Submit your Two Layer network
- You have to succeed **forward and backward pass** test to get bonus
- Test your network in the notebook -> Testing is same on the server
- Have fun coding:)

Optional Notebooks

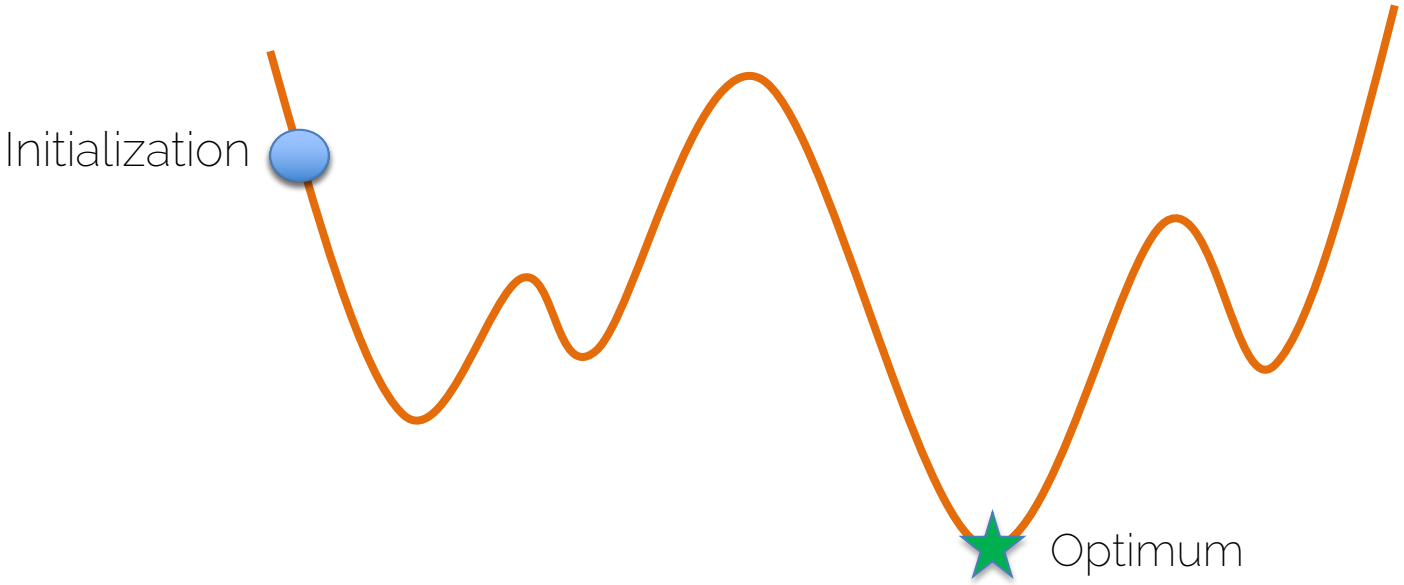
Optional Notebooks

- Stochastic Gradient Descent
  - 2a_StochasticGradientDescent.ipynb
- Optimization
  - 2b_Optimization.ipynb

Stochastic Gradient Descent (SGD)

Gradient Descent

$$x^* = \arg\min f(x)$$



GD and SGD

- Gradient Descent (GD):
  - $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} - \alpha \frac{1}{n} \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} L_i\left(\boldsymbol{\theta}^{k}, \boldsymbol{x}_i, \boldsymbol{y}_i\right)$
  - Iteration with all data

- Stochastic Gradient Descent (SGD)
  - $\frac{1}{n}\left(\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{y}_i)\right) = \mathbb{E}_{i \sim [1,\dots,n]}[\nabla_{\boldsymbol{\theta}} L_i(\boldsymbol{\theta}, \boldsymbol{x}_i, \boldsymbol{y}_i)]$
    - $\approx \frac{1}{|S|} \sum_{j \in S} \left(\nabla_{\boldsymbol{\theta}} L_j\left(\boldsymbol{\theta}, \boldsymbol{x}_j, \boldsymbol{y}_j\right)\right)$ with $S \subseteq \{1, \dots, n\}$
  - Iteration with **minibatches** (subsets of whole data)

GD and SGD

- Explore **2a_StochasticGradientDescent.ipynb**

- Observe differences between GD and SGD
  - Memory need for forward and backward pass
  - Nr. of computations per iteration
  - Stability of convergence
  - Speed of convergence
  - …

Optimizer

Optimzer

- Explore **2b_Optimization.ipynb**

- Compare implementation and performance between different optimizer:
  - SGD (Stochastic Gradient Descent)
  - SGD + Momentum
  - Adam

Submission 5: Two layer network for regression

Start: May 22, 2020 12.00

End: May 27, 2020 23.59

See you next week