Talent course July 22

Classification & Logistic regression

$$\frac{\partial C(\beta)}{\partial \beta} = 0 = -X^T(y - p)$$

$$p(y_i \mid x_i \beta) = \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)}$$

$$y_i = 1$$

$$p(y_i = 0 \mid x_i \beta) = 1 - p(y_i = 1 \mid x_i \beta)$$

$$\boxed{\frac{\partial^2 C}{\partial \beta \partial \beta^T} > 0}$$, convex function

Iterative solver like Newton-Raphson:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$f(s) = 0 = f(x) + (s - x) f'(x) + (s - x)^2 f''(x) + \ldots$$

$$f(x) + (s-x) f'(x) \frac{2!}{2} \approx 0$$

$$s = x - \frac{f(x)}{f'(x)} \sim \frac{\partial C}{\partial \beta} = 0$$

$$|x_{n+1} - x_n| < \varepsilon \sim 10^{-10}$$

$$\beta^{new} = \beta^{old} - \frac{1}{\left(\frac{\partial^2 C}{\partial \beta \partial \beta^T}\right)_{\beta^{old}}} \times \left(\frac{\partial C}{\partial \beta}\right)_{\beta^{old}}$$

$$= \beta^{old} - \boxed{\underbrace{(X^T W X)^{-1}}_{\text{Hessian matrix}}} \left(\underline{-X^T (y-p)}\right)_{\beta^{old}}$$

Conjugate gradient method
(gradient + descent)

$$(X^T W X)^{-1} \equiv \gamma = \text{constant}$$

$$\beta^{new} = \beta^{old} - \gamma \nabla(\beta)$$

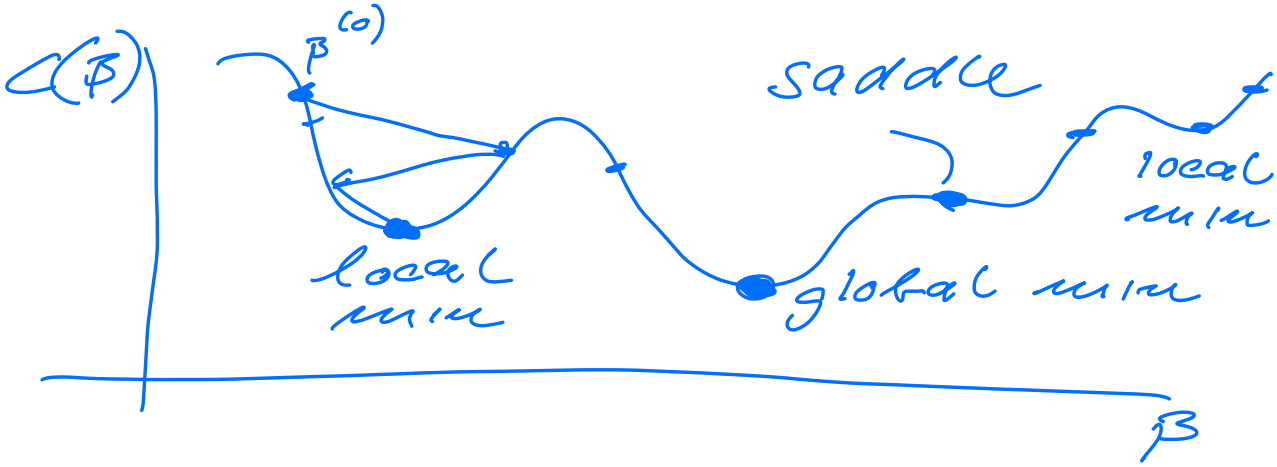
$$\beta = \beta - \gamma \nabla C(\beta)|_{\beta^{old}}$$

parameter in ML

$\gamma \in [10^{-5}, 10^{-4}, \ldots 10^{-1}]$

γ = LEARNING RATE

To avoid local minima one uses the family of stochastic gradient descent methods.



$\nabla C(\beta)$ using automatic differentiation (autograd)

$$\nabla C(\beta) = -v^T(u - \beta)$$

$$X \in \mathbb{R}^{n \times p} \qquad y \in \mathbb{R}^n$$

$$p \in \mathbb{R}^n$$

$$\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = X^T W X$$

$$W \in \mathbb{R}^{n \times n}$$

$$\hookrightarrow \in \mathbb{R}^{p \times p}$$

$$\nabla \ell(\beta) \in \mathbb{R}^p$$

$n \gg p$, $n \geq p$

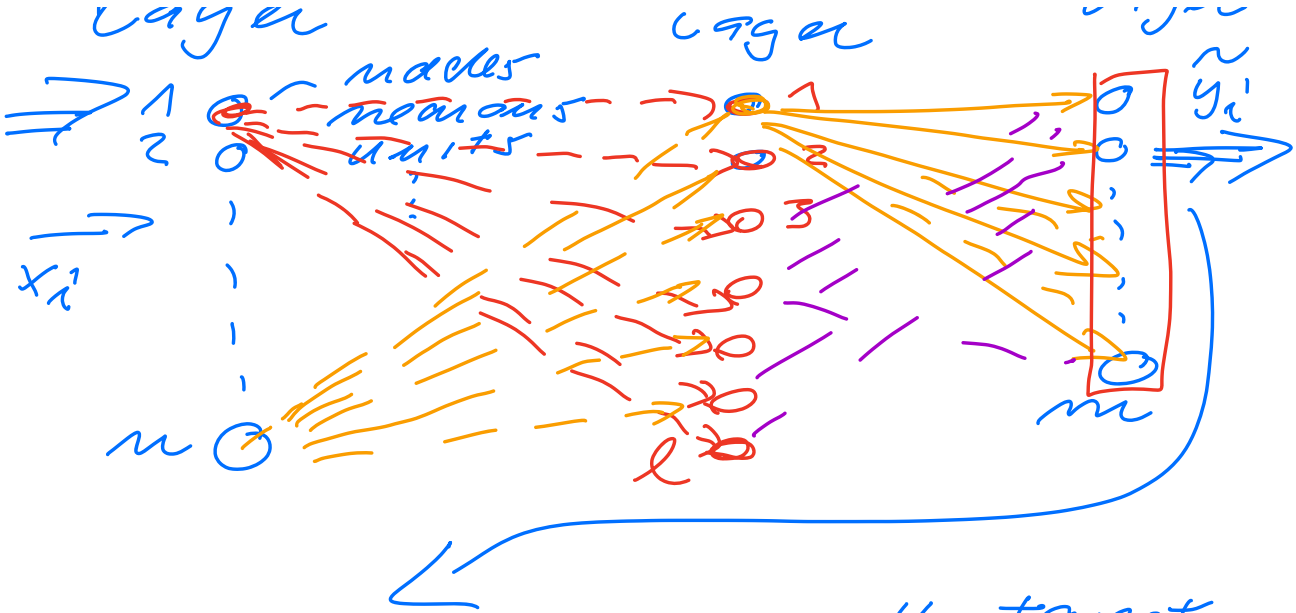
$p \gg n$

$n \sim 10^3 - 10^5$ $\qquad p \sim 10^1 - 10^2$

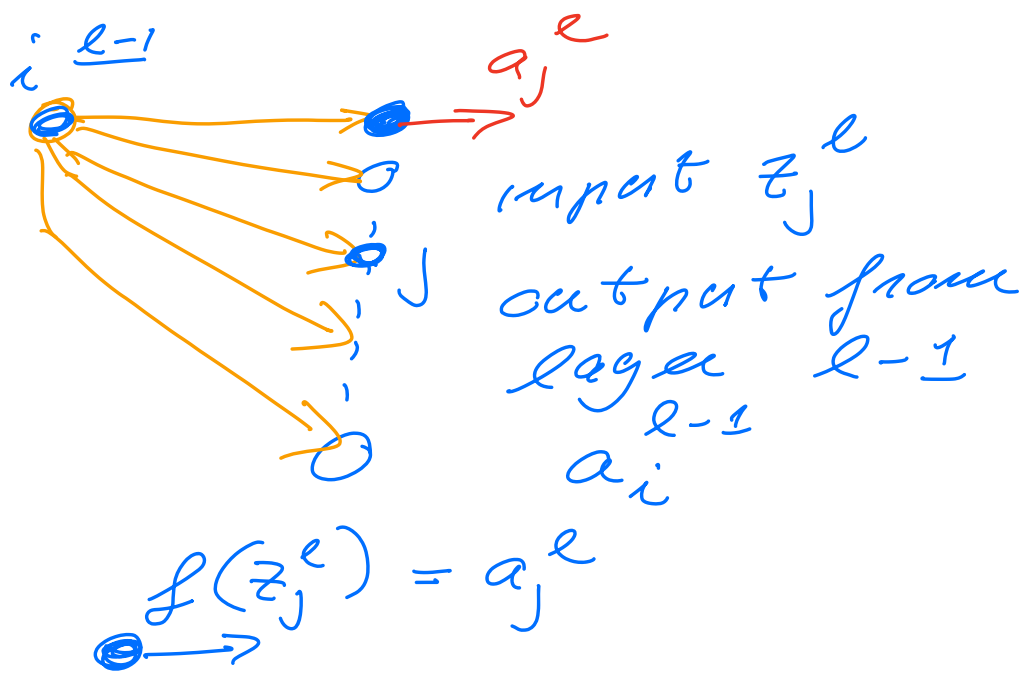
Neural Network (NN)

Feed Forward NN

input layer $\qquad$ hidden $\qquad$ output layer



Basic parameters



$$z_j^{\ell} = \sum_{i=1}^{M_{\ell-1}} w_{ij}^{\ell} a_i^{\ell-1} + b_j^{\ell}$$

weights — biases — parameters to fit in optimization.

z_j^{ℓ} produces the output from a node, a_j^{ℓ}

$$a_j^{\ell} = f(z_j^{\ell}) = \frac{1}{1+e^{-z_j^{\ell}}}$$

f: activation function

Gradients of the cost function to optimize weights w_{ij} and b_j

Algorithm to compute the gradients =

Back propagation algo (chain rule),

Parameters:

- w_{ij}^{ℓ} for each layer,
- b_j^{ℓ} for each layer
- activation function
  - logit
  - tanh
  - RELU
  - ELU
  - ...
- Number of hidden layers
- Number of nodes (except input and output)
- Learning rate γ (grid search)

- hyperparameter λ
  norm-2, norm-1 - ..

- choice of gradient descent method.