Introduction to neural networks

Models and back propagation

Outline

Motivation in Machine Learning

Neural networks

Backpropagation

General optimization problem

Parameter inference in machine learning often boils down to solving

$$\operatorname*{argmin}_{w \in \mathbb{R}^d} \ell_n(w) + g(w)\,,$$

with ℓ_n a **goodness-of-fit functio based on a loss ℓ**,

$$\ell_n(w) = \frac{1}{n} \sum_{i=1}^{n} \ell(w, y_i, x_i \rangle)$$

and

$$g(w) = \lambda \mathrm{pen}(w)\,,$$

where $\lambda > 0$ and **$\mathrm{pen}(\cdot)$ is some penalization function**.

$\rightharpoonup$ $\mathrm{pen}(w) = \|w\|_2^2$ (Ridge).

$\rightharpoonup$ $\mathrm{pen}(w) = \|w\|_1$ (Lasso).

General optimization problem - Regression

In a regression setting, for all $1 \leqslant i \leqslant n$,

$$Y_i = f_\star(X_i) + \varepsilon_i \,,$$

where the $(\varepsilon_i)_{1\leqslant i\leqslant n}$ are i.i.d. centered random variables in $\mathbb{R}^M$, $X_i \in \mathbb{R}^d$ and $f_\star$ **is an unknown function**.

The standard approach to estimate the parameters is by **minimizing the mean square error**:

$$\ell_n : \theta \mapsto \frac{1}{n}\sum_{i=1}^{n} \left\| f_\theta(X_i) - Y_i \right\|^2 \,,$$

where f_θ **is a nonlinear parametric function** used to estimate the unknown function $f_\star$.

Gradient descent algorithm

Gradient descent

Input: Function ℓ_n to minimize, initial vector $\theta^{(0)}$, $k = 0$.

Parameters: step size $\eta > 0$.

While *not converge* do

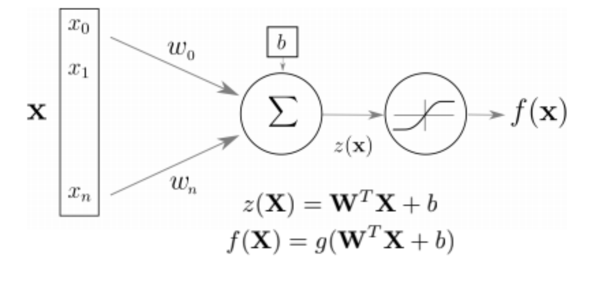
$$\rightharpoondown \theta^{(k+1)} = \theta^{(k)} - \eta_{k+1} \nabla \ell_n(\theta^{(k)}).$$

$$\rightharpoondown k = k + 1.$$

Output: $\theta^{(n*)}$ where n_* is the last iteration.

Outline

A neuron



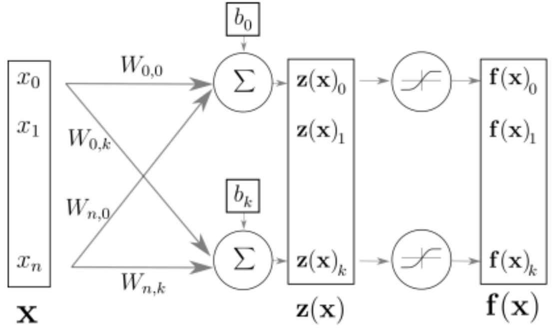
→ X **input in** $\mathbb{R}^d$.

→ $z(X)$ **pre-activation in** $\mathbb{R}^M$, with **weight** $W \in \mathbb{R}^{dxM}$ and **bias** $b \in \mathbb{R}^M$.

→ g **softmax function**.

One neuron is a multi-class extension of the logistic regression model.

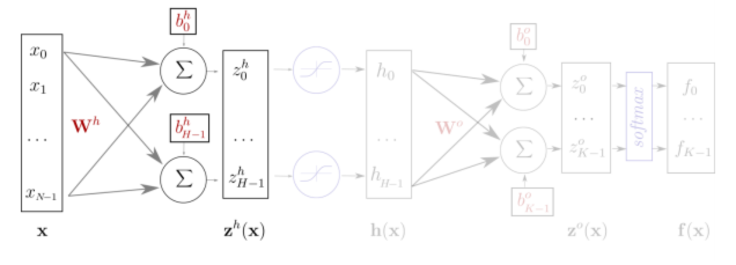
Layer of neurons and hidden states



$$f(\mathbf{X}) = g(\mathbf{z}(\mathbf{X})) = g(\mathbf{WX} + \mathbf{b})$$

- → X **input in** $\mathbb{R}^d$.
- → $z(X)$ **pre-activation in** $\mathbb{R}^k$, with **weight** $W \in \mathbb{R}^{dxk}$ and **bias** $b \in \mathbb{R}^k$.
- → g **any activation function** (nonlinear & nondecreasing function).
- → $f(X)$ **hidden state** in $\mathbb{R}^k$ which may be used as input of a new neuron...

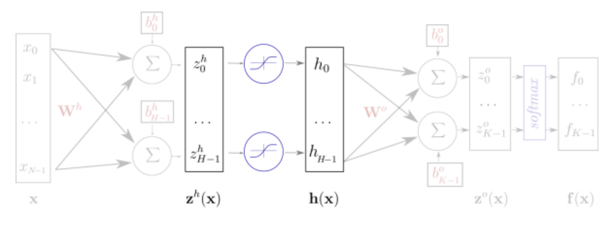
Feed Forward Network



→ X **input in** $\mathbb{R}^d$.

→ $z^h(X)$ **pre-activation in** $\mathbb{R}^H$, with **weight** $W^h \in \mathbb{R}^{dxH}$ and **bias** $b^h \in \mathbb{R}^H$.

Feed Forward Network

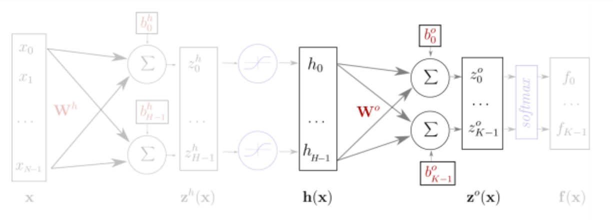


→ X **input in** $\mathbb{R}^d$.

→ $z^h(X)$ **pre-activation in** $\mathbb{R}^H$, with **weight** $W^h \in \mathbb{R}^{dxH}$ and **bias** $b^h \in \mathbb{R}^H$.

→ g **any activation function** to produce $h \in \mathbb{R}^H$.

Feed Forward Network



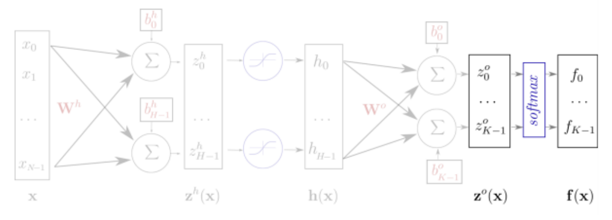
→ X **input in** $\mathbb{R}^d$.

→ $z^h(X)$ **pre-activation in** $\mathbb{R}^H$, with **weight** $W^h \in \mathbb{R}^{dxH}$ and **bias** $b^h \in \mathbb{R}^H$.

→ g **any activation function** to produce $h \in \mathbb{R}^H$.

→ $z^o(X)$ **pre-activation in** $\mathbb{R}^M$, with **weight** $W^o \in \mathbb{R}^{HxM}$ and **bias** $b^o \in \mathbb{R}^M$.

Feed Forward Network



→ X **input in** $\mathbb{R}^d$.

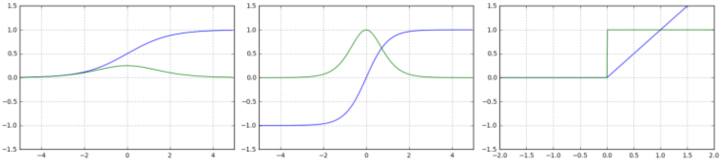
→ $z^h(X)$ **pre-activation in** $\mathbb{R}^H$, with **weight** $W^h \in \mathbb{R}^{dxH}$ and **bias** $b^h \in \mathbb{R}^H$.

→ g **any activation function** to produce $h \in \mathbb{R}^H$.

→ $z^o(X)$ **pre-activation in** $\mathbb{R}^M$, with **weight** $W^o \in \mathbb{R}^{HxM}$ and **bias** $b^o \in \mathbb{R}^M$.

→ Apply the **softmax function to produce the output**, i.e. $\mathbb{P}(Y = m|X)$ for $1 \leqslant m \leqslant M$.

Activation functions



$$\mathrm{sigm}(x) = \frac{1}{1+e^{-x}} \qquad \tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \qquad \mathrm{relu}(x) = \max(0, x)$$

$$\mathrm{sigm}'(x) = \mathrm{sigm}(x)(1-\mathrm{sigm}(x)) \qquad \tanh'(x) = 1 - \tanh(x)^2 \qquad \mathrm{relu}'(x) = 1_{x>0}$$

$\rightarrow$ As there is no modelling assumptions anymore, **virtually any activation function** may be used.

$\rightarrow$ The rectified linear unit (RELU) activation function $\sigma(x) = \max(0, x)$ and its extensions are the default recommendation in modern implementations (Jarrettet al., 2009; Nair and Hinton, 2010; Glorot et al., 2011a), (Maas et al.,2013), (He et al., 2015). One of the major motivations arise from the **gradient based parameter optimization which is numerically more stable with this choice**.

Outline