MLCC 2022
Neural networks

Simone Di Marino
Unige - DIMA, MaLGa

About this class

- Extend the nonlinear model to be able to learn also the *feature map*
- What is a neuron? The role of the nonlinearity
- Neural Networks: what are they and how to train them
- Some examples

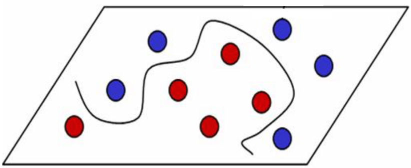
Supervised learning

Given

$$(x_1, y_1), \ldots, (x_n, y_n)$$

find f such that

$$f(x_{\text{new}}) = y_{\text{new}}$$



- $x \in \mathbb{R}^d$ input (for example a vectorization of an image)
- y output ($\{0, 1\}$ for classification (cat/dog or tumor/no tumor), price evaluation, wage decision)

Data representation and features

Consider as before

$$f(x) = w^\top \Phi(x).$$

Can we use a *meaningful* Φ?

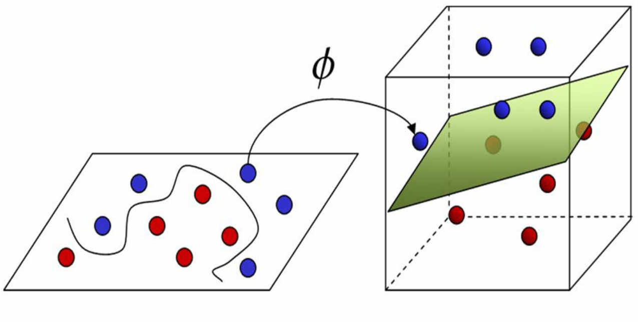
A two steps learning scheme is (was?) often considered

- *supervised* learning of w
- expert design or *unsupervised* learning of the **data representation** Φ

Data representation and features

$$\Phi : \mathbb{R}^d \to \mathbb{R}^D$$

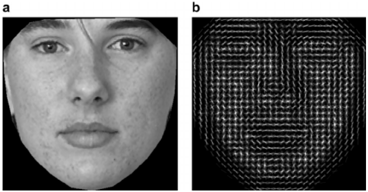
A mapping of data in a new **format** better suited for further processing



Data representation by design

Dictionaries of features (more meaningful, more ad hoc)

- Wavelet & friends.
- Bag of words.



Kernels (less meaningful, more universal)

- Classic: Gaussian $K(x, x') = e^{-\|x-x'\|^2\gamma}$, corresponds to $\Phi = \ldots$
- Structured input: kernels on histograms, graphs etc.

Data representation and features

Consider as before

$$f(x) = w^\top \Phi(x).$$

Can we *learn* also Φ?

- *supervised* learning of w AND Φ.

How to **parametrize** Φ?

Road Map

Part I: Basics neural networks

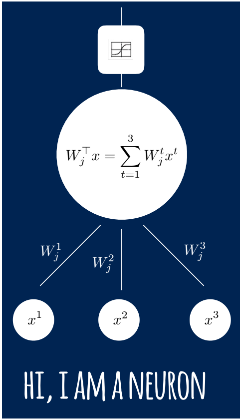
- Neural networks definition
- Optimization +approximation and statistics

Part II: One step beyond

- Auto-encoders
- Convolutional neural networks
- Tips and tricks

Part I: Basic Neural Networks

A neuron



A basic nonlinear function of x:

$$x \mapsto \sigma(W_j^1 x_1 + W_j^2 x_2 + W_j^3 x_3 - t_j)$$

Inspired from biology (t_j is a firing treshold).

How the brain works? Using many neurons connected to each other (network)?

2-layers Neural Networks

$$\phi_j(x) = \sigma\left(\sum_{i=1}^{d} W_j^i x_i + b_j\right), \qquad \Phi(x) = (\phi_1(x), \ldots, \phi_D(x))$$

$$f_{w,W,b}(x) = w^\top \sigma(Wx + b), \quad \underbrace{x \mapsto Wx + b}_{\textbf{Affine}}$$

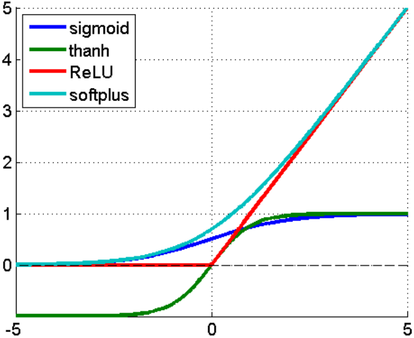
Properties

- $\sigma : \mathbb{R} \to \mathbb{R}$ is called activation function;
- σ acts component-wise $\sigma(a_1, \ldots, a_n) = (\sigma(a_1), \ldots, \sigma(a_n))$;
- σ **has to be nonlinear**

Activation functions

For $\alpha \in \mathbb{R}$ consider,

- **sigmoid** $s(\alpha) = 1/(1+e^{-\alpha})t$,
- **hyperbolic tangent** $s(\alpha) = (e^{\alpha} - e^{-\alpha})/(e^{\alpha} + e^{-\alpha})$,
- **ReLU** $s(\alpha) = |\alpha|_+$ (aka ramp, hinge),
- **Softplus** $s(\alpha) = \log(1 + e^{\alpha})$.

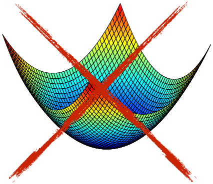


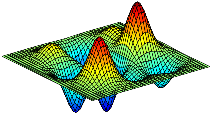
Main difference: nonconvexity!

$$f_w(x) = w^\top \Phi(x), \qquad f_{w,W,b}(x) = w^\top \sigma(Wx + b)$$

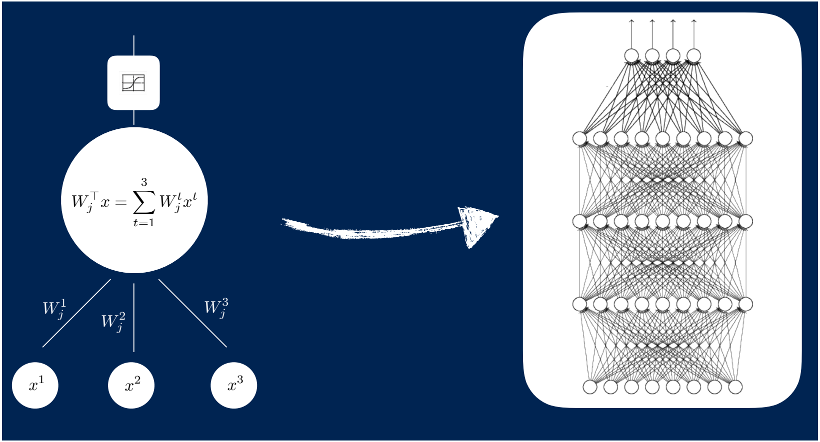
- The first one is linear in w
- The second one is not linear in w, W, b (although can be separately convex) : no guarantees of convergence for ERM since the empirical risk is **non convex**

$$(w, W, b)^* = \operatorname{argmin}_{w,W,b} \sum_{i=1}^{n} (y_i - w^\top \sigma(Wx_i + b))^2 + \lambda \|(w, W, b)\|_2^2$$





Deep neural networks



Neural Nets: going deeper

Basic idea: **compose** simply **parameterized** representations

$$\Phi = \Phi_L \circ \cdots \circ \Phi_2 \circ \Phi_1$$

Let $d_0 = d$ and

$$\Phi_\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}, \quad \ell = 1, \dots, L$$

and in particular

$$\Phi_\ell = \sigma \circ W_\ell, \quad \ell = 1, \dots, L$$

where

$$W_\ell : \mathbb{R}^{d_{\ell-1}} \to \mathbb{R}^{d_\ell}, \quad \ell = 1, \dots, L$$

linear/affine and σ is the activation function acting component-wise

$$\sigma : \mathbb{R} \to \mathbb{R}.$$

Deep neural nets

$$f(x) = w^\top \Phi_L(x), \qquad \underbrace{\Phi_L = \overline{\Phi}_L \circ \cdots \circ \overline{\Phi}_1}_{\text{compositional representation}}$$

$$\overline{\Phi}_1 = \sigma \circ W_1 \quad \dots \quad \overline{\Phi}_L = \sigma \circ W_L$$

ERM

$$\min_{w,(W_j)_j} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^\top \Phi_L(x_i))^2$$

Neural networks jargoon

$$\Phi_L(x) = \sigma(W_L \dots \sigma(W_2 \sigma(W_1 x)))$$

- L is the number of **layers**
- Each intermediate representation corresponds to a **(hidden) layer**
- The dimensionalities $(d_\ell)_\ell$ correspond to the number of **hidden units**

Some questions

$$f_{w,(W_\ell)_\ell}(x) = w^\top \Phi_{(W_\ell)_\ell}(x), \qquad \Phi_{(W_\ell)_\ell} = \sigma(W_L \ldots \sigma(W_2 \sigma(W_1 x)))$$

We have our model but:

- **Optimization:** Can we **train** efficiently?
- **Approximation:** Are we dealing with **rich** models?

Computing the gradient: chain rule

$$f_{w,(W_\ell)_\ell}(x) = w^\top \Phi_{(W_\ell)_\ell}(x), \qquad \Phi_{(W_\ell)_\ell} = \sigma(W_L \dots \sigma(W_2 \sigma(W_1 x)))$$

and ERM again

$$\sum_{i=1}^n (y_i - f_{w,(W_\ell)_\ell}(x_i))^2,$$

How can we compute the gradient with respect to w, W_ℓ?

chain rule!

$$(f \circ g)'(a) = f'(g(a)) \cdot g'(a)$$

$$(f \circ g \circ h)'(a) = f'(g(h(a))) \cdot g'(h(a)) \cdot h'(a)$$

Computations

Consider

$$\min_{w,W} \widehat{\mathcal{E}}(w,W), \qquad \widehat{\mathcal{E}}(w,W) = \sum_{i=1}^{n} (y_i - f_{(w,W)}(x_i)))^2.$$

Back-propagation & GD

Empirical risk minimization,

$$\min_{w,W} \widehat{\mathcal{E}}(w,W), \qquad \widehat{\mathcal{E}}(w,W) = \sum_{i=1}^{n} (y_i - f_{(w,W)}(x_i)))^2.$$

An approximate minimizer is computed via the following **gradient** method

$$\begin{aligned} w_j^{t+1} &= w_j^t - \gamma_t \frac{\partial \widehat{\mathcal{E}}}{\partial w_j}(w^t, W^t) \\ W_{j,k}^{t+1} &= W_{j,k}^t - \gamma_t \frac{\partial \widehat{\mathcal{E}}}{\partial W_{j,k}}(w^{t+1}, W^t) \end{aligned}$$

where the step-size $(\gamma_t)_t$ is often called learning rate.

Back-propagation & chain rule

Direct computations show that:

$$
\begin{aligned}
\frac{\partial \widehat{\mathcal{E}}}{\partial w_j}(w, W) &= -2 \sum_{i=1}^{n} \underbrace{(y_i - f_{(w,W)}(x_i)))}_{\Delta_{j,i}} h_{j,i} \\
\frac{\partial \widehat{\mathcal{E}}}{\partial W_{j,k}}(w, W) &= -2 \sum_{i=1}^{n} \underbrace{(y_i - f_{(w,W)}(x_i)))w_j \sigma'(w_j^\top x)}_{\eta_{i,k}} x_i^k
\end{aligned}
$$

Back-prop equations: $\eta_{i,k} = \Delta_{j,i} c_j \sigma'(w_j^\top x)$

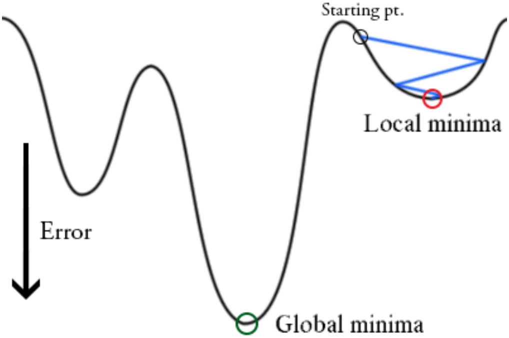
Using above equations, the updates are performed in two steps:

- **Forward pass** compute function values keeping weights fixed,
- **Backward pass** compute errors and propagate
- Hence the weights are updated.

SGD is typically preferred

$$
\begin{aligned}
w_j^{t+1} &= w_j^t - \gamma_t 2(y_t - f_{(w_t, W_t)}(x_t)) h_{j,t} \\
W_{j,k}^{t+1} &= W_{j,k}^t - \gamma_t 2(y_t - f_{(w_{t+1}, W_t)}(x_t)) w_j \sigma'(w_j^\top x) x_t^k
\end{aligned}
$$

Non convexity and SGD



Few remarks

- Optimization by **gradient methods**– typically SGD
- **Online** update rules are potentially biologically plausible– **Hebbian learning** rules describing neuron **plasticity**
- **Multiple layers** can be analogously considered
- **Multiple step-size per layers** can be considered
- **Initialization** is tricky
- **NO** convergence guarantees

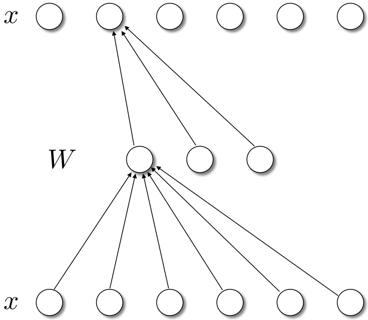
Some questions

- What is the benefit of multiple layers?
- Why does stochastic gradient seem to work?

Unsupervised learning with neural networks

- Because unlabeled data abound
- Because one could use obtained weight for initialize supervised learning (pre-training)

Auto-encoders



- A neural network with **one input layer, one output layer and one (or more) hidden layers** connecting them.
- The output layer has **equally** many nodes as the input layer,
- It is trained to **predict the input** rather than some target output.

Auto-encoders (cont.)

An auto encoder with one hidden layer of k units, can be seen as a **representation-reconstruction** pair:

$$\Phi : \mathbb{R}^D \to \mathcal{F}_k, \quad \Phi(x) = \sigma\left(Wx\right), \quad \forall x \in \mathbb{R}^D$$

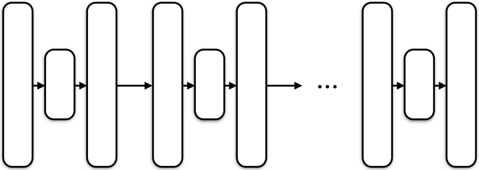
with $\mathcal{F}_k = \mathbb{R}^k$, $k < d$ and

$$\Psi : \mathcal{F}_k \to \mathbb{R}^D, \quad \Psi(\beta) = \sigma\left(W'\beta\right), \quad \forall \beta \in \mathcal{F}_k.$$

Stacked auto-encoders

Multiple layers of auto-encoders can be **stacked** [Hinton et al '06]...

$$\underbrace{(\Phi_1 \circ \Psi_1)}_{\text{Autoencoder}} \circ (\Phi_2 \circ \Psi_2) \cdots \circ (\Phi_\ell \circ \Psi_\ell)$$



...with the potential of obtaining **richer** representations.

Next lecture

Sparsity in linear model and interpretability...