Compressed Sensing and Neural Networks

Jan Vybíral

(Charles University & Czech Technical University
Prague, Czech Republic)

NOMAD Summer

Berlin, September 25-29, 2017

Outline

Part I

Lasso & Compressed Sensing

- Least squares & Regularization
- Convexity, P vs. NP
- Sparsity & ℓ_1-minimization
- Compressed Sensing

Neural Networks

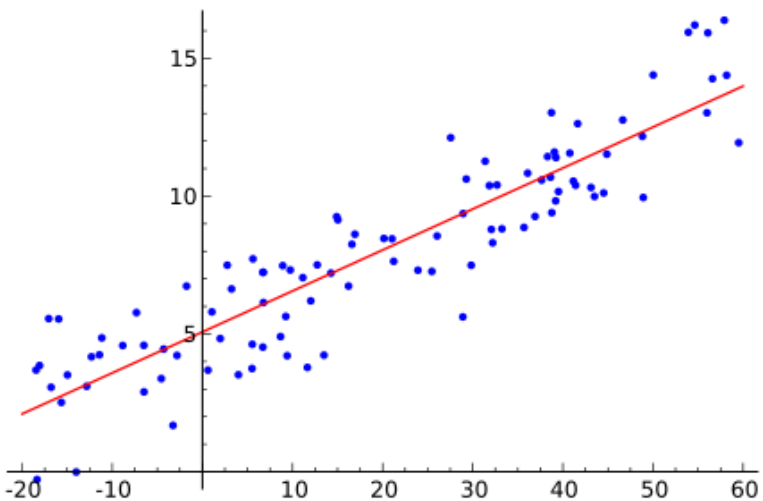
- Introduction
- Notation
- Training the network
- Applications

Least squares

Fitting a cloud of points by a linear hyperplane

Considered already by Gauss and Legendre in 18th century

In 2D:



Least squares

Objects (=points) described by Ω real numbers:

$$\mathbf{d}_1 = (d_{1,1}, \ldots, d_{1,\Omega}) \in \mathbb{R}^{\Omega}$$

$$\vdots$$

$$\mathbf{d}_N = (d_{N,1}, \ldots, d_{N,\Omega}) \in \mathbb{R}^{\Omega}$$

N - number of objects; $\mathbf{D}$ - $N \times \Omega$ matrix with rows $\mathbf{d}_1, \ldots, \mathbf{d}_N$

Least squares

Objects (=points) described by Ω real numbers:

$$\mathbf{d}_1 = (d_{1,1}, \ldots, d_{1,\Omega}) \in \mathbb{R}^\Omega$$

$$\vdots$$

$$\mathbf{d}_N = (d_{N,1}, \ldots, d_{N,\Omega}) \in \mathbb{R}^\Omega$$

N - number of objects; $\mathbf{D}$ - $N \times \Omega$ matrix with rows $\mathbf{d}_1, \ldots, \mathbf{d}_N$

$\mathbf{P} = (P_1, \ldots, P_N)$ are properties of interest

We look for a linear dependence $P = f(\mathbf{d})$ with a linear f, i.e.

$$P_i = \sum_{j=1}^{\Omega} c_j d_{i,j} \qquad \text{or} \qquad \mathbf{P} = \mathbf{D}\mathbf{c}$$

Least squares

The solution is found by minimizing the least-square error:

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathbb{R}^{\Omega}} \sum_{i=1}^{N} \left(P_i - \sum_{j=1}^{\Omega} c_j d_{i,j} \right)^2 = \arg\min_{\mathbf{c}\in\mathbb{R}^{\Omega}} \|\mathbf{P} - \mathbf{D}\mathbf{c}\|_2^2$$

Least squares

The solution is found by minimizing the least-square error:

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathbb{R}^{\Omega}} \sum_{i=1}^{N} \left(P_i - \sum_{j=1}^{\Omega} c_j d_{i,j} \right)^2 = \arg\min_{\mathbf{c}\in\mathbb{R}^{\Omega}} \|\mathbf{P} - \mathbf{D}\mathbf{c}\|_2^2$$

- Closed formula exists
- Convex objective function
- $\hat{\mathbf{c}}$ with all coordinates occupied
- Absolute term incorporated by an additional column full of ones

Regularization

How to include preknowledge on $\mathbf{c}$?

Say, we prefer linear fit with small coefficients. We just weight the error of the fit against the size of the coefficient!

$\lambda > 0$ - *regularization parameter*

$$\hat{\mathbf{c}} = \underset{\mathbf{c} \in \mathbb{R}^{\Omega}}{\arg\min} \|\mathbf{P} - \mathbf{D}\mathbf{c}\|_2^2 + \lambda \|\mathbf{c}\|_2^2$$

- $\lambda \to 0$: least squares
- $\lambda \to \infty$: $\mathbf{c} = 0$

Tractability

Convexity

- The minimizer is unique
- Local minimum of a convex function is also a global one
- Many effective methods exist (*convex optimization*)

Tractability

Convexity

- The minimizer is unique
- Local minimum of a convex function is also a global one
- Many effective methods exist (*convex optimization*)

P vs. NP

- P-problems: solvable in polynomial time (in dependence on the size of the input)
- NP-problems: solution verifiable in polynomial time; P$\subset$NP
- One million dollar problem: P=NP?
- *Computational Complexity*

Sparsity

If Ω is large (especially $\Omega \gg N$), we are often interested in "selecting features", i.e. in $\mathbf{c}$ with many coordinates equal to zero.

$\|\mathbf{c}\|_0 := \#\{i : c_i \neq 0\}$ - the number of non-zero coordinates of $\mathbf{c}$

Looking for a linear fit using only two features:

$$\hat{\mathbf{c}} = \underset{\mathbf{c}\in\mathbb{R}^\Omega, \|\mathbf{c}\|_0 \leq 2}{\arg\min} \|\mathbf{P} - \mathbf{D}\mathbf{c}\|_2^2$$

Regularized version:

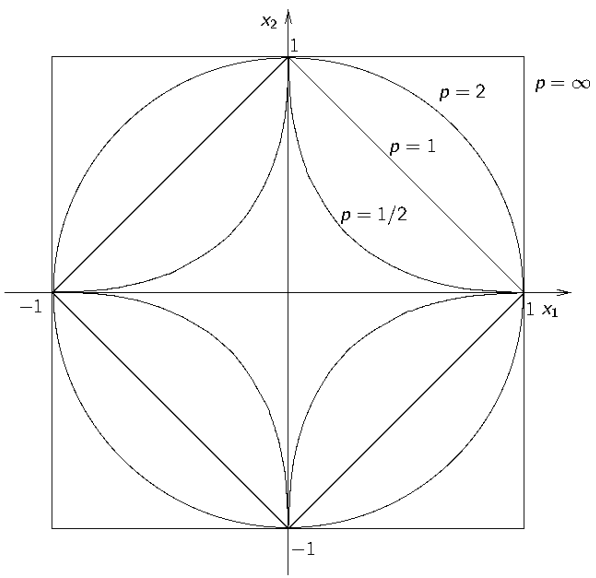
$$\hat{\mathbf{c}} = \underset{\mathbf{c}\in\mathbb{R}^\Omega}{\arg\min} \|\mathbf{P} - \mathbf{D}\mathbf{c}\|_2^2 + \lambda\|\mathbf{c}\|_0$$

NP-hard!

ℓ_1-minimization

Other ways to measure the size of $\mathbf{c}$: the ℓ_p-norms

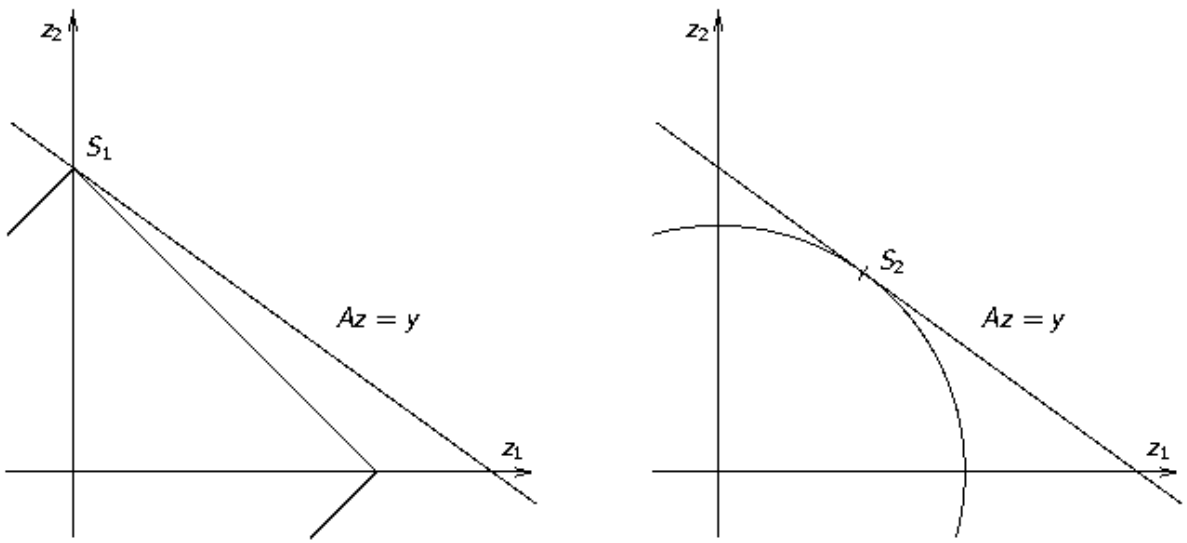
$$\|\mathbf{c}\|_p = \left(\sum_{j=1}^{\Omega} |c_j|^p\right)^{1/p}$$



- Unit balls in ℓ_p in $\mathbb{R}^2$
- $p = \infty$: $\|\mathbf{c}\|_\infty = \max_{j=1,\ldots,\Omega} |c_j|$
- $p \geq 1$ - convex problem
- $p \leq 1$ - promotes sparsity

ℓ_1-minimization

$p \leq 1$ - promotes sparsity



Solution of $S_p = \underset{z \in \mathbb{R}^2}{\arg\min} \|z\|_p \quad \text{s.t.} \quad Az = y$ for $p = 1,\ p = 2$

ℓ_1-minimization

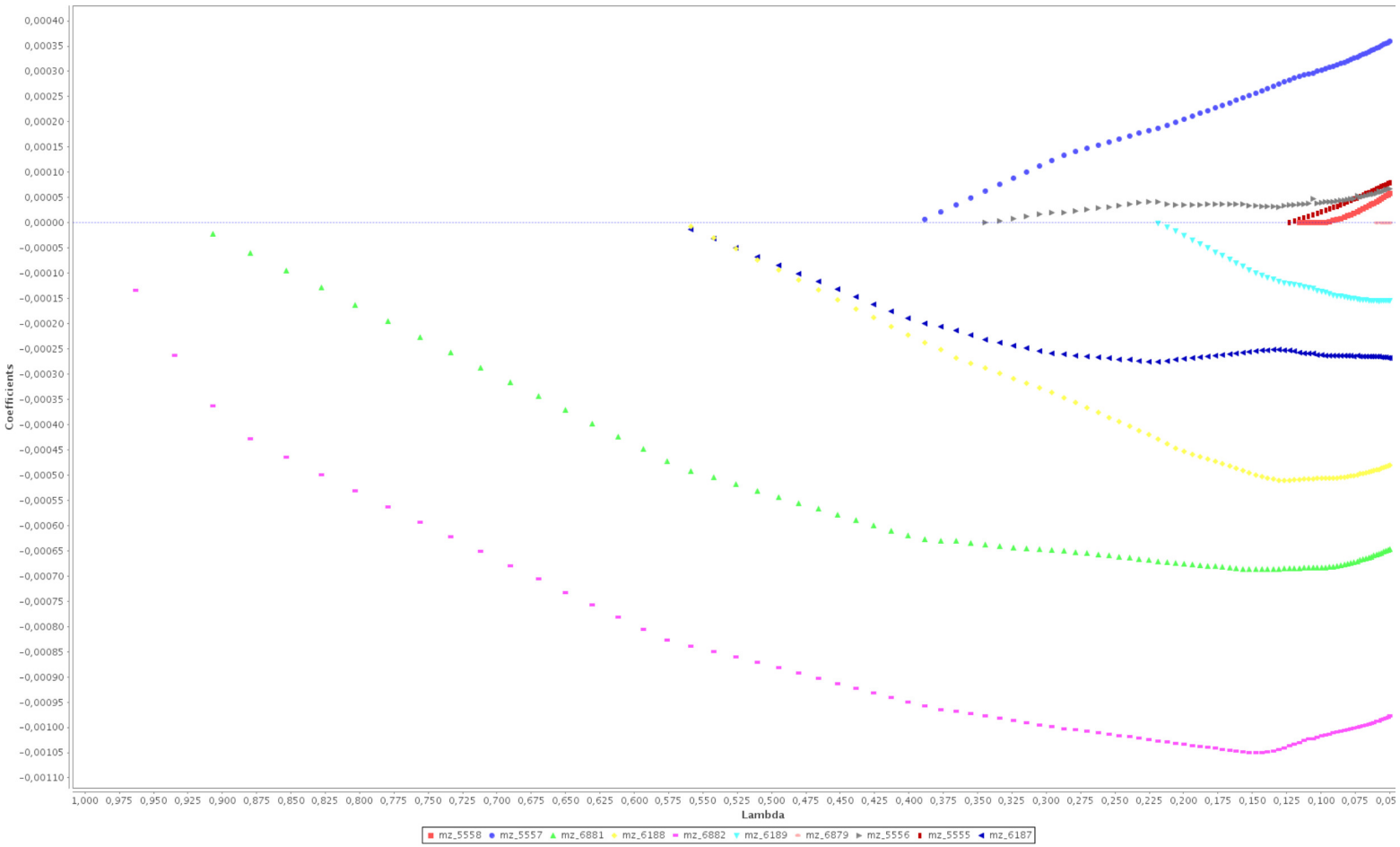
Take $p = 1$ (Lasso - Tibshirani, 1996)

$$\hat{\mathbf{c}} = \arg\min_{\mathbf{c}\in\mathbb{R}^{\Omega}} \|\mathbf{P} - \mathbf{D}\mathbf{c}\|_2^2 + \lambda\|\mathbf{c}\|_1$$

- Chen, Donoho, Saunders: Basis pursuit (1998)
- $\lambda \to 0$: least squares
- $\lambda \to \infty$: $\hat{\mathbf{c}} = 0$
- In between: λ selects sparsity

ℓ_1-minimization

Effect of $\lambda > 0$ on the support of ω



Compressed Sensing (aka Compressive Sensing, Compressive Sampling)

Theorem: Let $\mathbf{D} \in \mathbb{R}^{N \times \Omega}$ with **independent gaussian entries!** Let $0 < \varepsilon < 1$, s a natural number and

$$N \geq C\Big(s \log(\Omega) + \log(1/\varepsilon)\Big), \quad C \text{ a universal constant.}$$

If $\mathbf{c} \in \mathbb{R}^{\Omega}$ is s-sparse, $\mathbf{P} = \mathbf{D}\mathbf{c}$ and $\hat{\mathbf{c}}$ is the minimizer of

$$\hat{\mathbf{c}} = \underset{\mathbf{u} \in \mathbb{R}^{\Omega}}{\arg\min} \|\mathbf{u}\|_1, \quad \text{s.t. } \mathbf{P} = \mathbf{D}\mathbf{u},$$

then $\mathbf{c} = \hat{\mathbf{c}}$ with prob. at least $1 - \varepsilon$.

Compressed Sensing (aka Compressive Sensing, Compressive Sampling)

- Candés, Romberg, Tao (2006); Donoho (2006)
- Extensive theory of recovery of sparse vectors from linear measurements
- Optimal conditions on the number of measurements (i.e. data points) $N \approx Cs \log \Omega$
- Only true, if most of the features (i.e. the columns of $\mathbf{D}$) are incoherent with the majority of the others (if two features are very similar, it is difficult to distinguish between them)
- H. Boche, R. Calderbank, G. Kutyniok, J.V.,
  A Survey of Compressed Sensing,
  First chapter in *Compressed Sensing and its Applications*,
  Birkhäuser, Springer, 2015

Dictionaries

Real-life signals are (almost) never sparse in the canonical basis of $\mathbb{R}^\Omega$, more often they are sparse in some orthonormal basis, i.e.

$$\mathbf{x} = \mathbf{B}\mathbf{c},$$

where $\mathbf{c} \in \mathbb{R}^\Omega$ is sparse and columns (and rows) of $\mathbf{B} \in \mathbb{R}^{\Omega\times\Omega}$ are orthonormal vectors - wavelets, Fourier basis, etc.

Compressed Sensing applies then without any essential change!
...just replace $\mathbf{D}$ with $\mathbf{DB}$... i.e. you rotate the problem...

Dictionaries

Even more often, the signal is represented in an overcomplete dictionary/lexicon:

$$\mathbf{x} = \mathbf{L}\mathbf{c},$$

where $\mathbf{c} \in \mathbb{R}^{\ell}$ is sparse and columns of $\mathbf{L} \in \mathbb{R}^{\Omega \times \ell}$ is the dictionary/lexicon - its columns form an overcomplete system ($\ell > \Omega$)

$\mathbf{x}$ is a sparse combination of non-orthogonal vectors - the columns of $\mathbf{L}$.

Examples: Unions of two or more orthonormal bases, each capturing different features

Dictionaries

- Compressed sensing can be adapted also to this situation
- Optimization:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{u}\in\mathbb{R}^{\Omega}} \|\mathbf{L}^*\mathbf{u}\|_1, \quad \text{s.t. } \mathbf{P} = \mathbf{D}\mathbf{u}$$

- We do not recover the (non-unique!) sparse coefficients $\mathbf{c}$, but the (approximation of) the signal $\mathbf{x}$.
- Error bound involves $\mathbf{L}^*\mathbf{x}$, is reasonably small for example when $\mathbf{L}^*\mathbf{L}$ is nearly diagonal . . . not too many features in the dictionary are too correlated. . .

ℓ_1-based optimalization

- ℓ_1-**SVM**: Support vector machines are a standard tool for classification problems. ℓ_1-penalty term leads to *sparse classifiers*.
- **Nuclear norm**: Minimizing nuclear norm (=sum of absolute values of eigenvalues) of a matrix leads to *low-rank matrices*.
- **TV(=total variation)-norm**: Minimizing $\sum_{i,j} |u_{i,j+1} - u_{i,j}|$ over images **u** gives images with edges and flat parts.
- L_1: Minimizing the L_1-norm (=integral of the absolute value) of a function leads to functions with small support
- **TV-norm of** f: Minimizing $\int |\nabla f|$ leads to functions with jumps along curves.

Part II

Lasso & Compressed Sensing

- Least squares & Regularization
- Convexity, P vs. NP
- Sparsity & ℓ_1-minimization
- Compressed Sensing

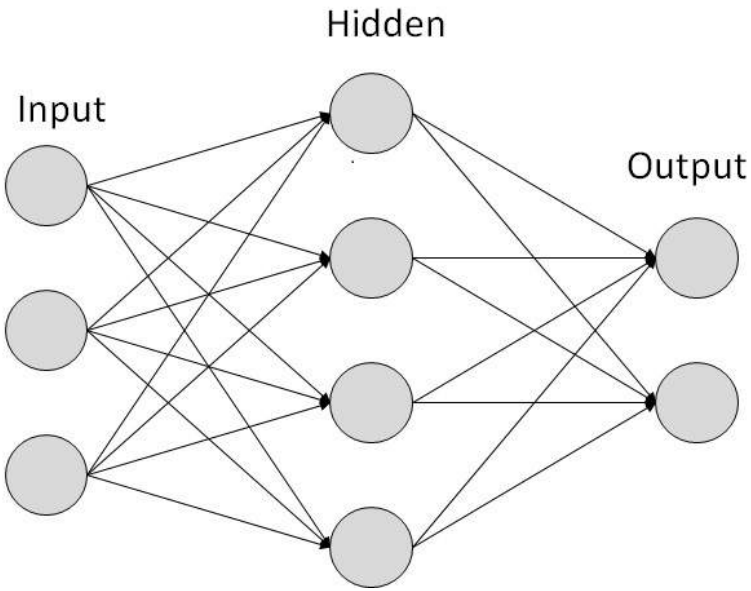
Neural Networks

- Introduction
- Notation
- Training the network
- Applications

Neural Networks

W. McCulloch, W. Pitts (1943)
Motivated by biological research on human brain and neurons

Neural network is a graph of nodes, partially connected. Nodes represents neurons, oriented connections between the nodes represent the transfer of outputs of some neurons to inputs of other neurons.



Neural Networks

- In 70's and 80's a number of obstacles appeared - insufficient computer power to train large neural networks, theoretical problems of processing exclusive-or, etc.
- Support vector machines (and other simple algorithms) took over the field of machine learning
- 2010's: Algorithmic advances and higher computational power allowed to train large neural networks to human (and superhuman) performance in pattern recognition
- Large neural networks (a.k.a. deep learning) used successfully in many tasks

Neural Networks: Artificial Neuron

Artificial Neuron:
. . . gets activated if a linear combination of its inputs grows over a certain threshold. . .

- Inputs $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$
- Weights $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$
- Comparing $\langle w, x \rangle$ with a threshold $b \in \mathbb{R}$
- Plugging the result into the "activation function" - jump (or smoothed jump) function σ

Artificial neuron is a function

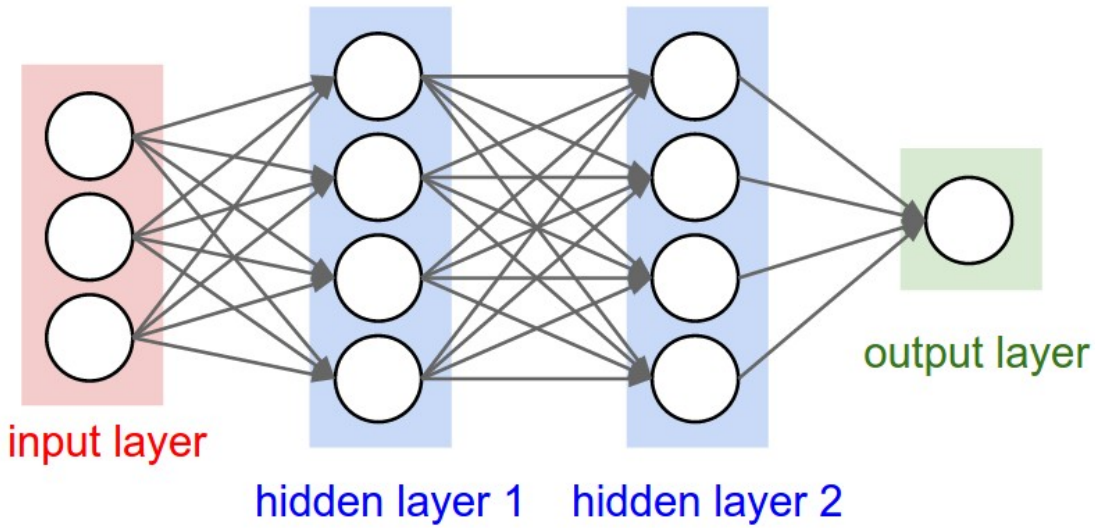
$$x \to \sigma(\langle x, w \rangle - b),$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ might be $\sigma(x) = \operatorname{sgn}(x)$ or $\sigma(x) = e^x/(1 + e^x)$, etc.

Neural Networks: Layers

Artificial neural network is a directed, acyclic graph of artificial neurons

The neurons are grouped by their distance to the input into layers



Neural Networks: Layers

- Input: $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$
- First layer of neurons:
  $y_1 = \sigma(\langle x, w_1^1 \rangle - b_1^1), \ldots, y_{n_1} = \sigma(\langle x, w_{n_1}^1 \rangle - b_{n_1}^1)$
- The outputs $y = (y_1, \ldots, y_{n_1})$ become inputs for the next layer $\ldots$; last layer outputs $y \in \mathbb{R}$
- Training the network: given inputs $x^1, \ldots, x^N$ and outputs $y^1, \ldots, y^N$ and optimize over weights w's and b's

Neural Networks: Training

- The parameters p of the network are initialized (for example in a random way) $\Longrightarrow \mathcal{N}_p$
- For a set of pairs input/output (x^i, y^i) we calculate the output of the neural network with current parameters $\Longrightarrow z^i = \mathcal{N}_p(x^i)$.
- In an optimal case, $z^i = y^i$ for all inputs
- Update the parameters of the neural networks to minimize/decrease the loss function, i.e.

$$\sum_i |y^i - z^i|^2$$

- . . . and repeat . . .

Neural Networks: Training

- Non-convex minimization over a huge space!
- Huge number of local minimizers exist
- Initialization of the minimization algorithm is important
- Backpropagation algorithm: the error at the output is redistributed to the neurons of the last hidden layer, then to the previous one, etc.
- The error is distributed back through the network and used to update the parameters of each neuron by a gradient descent method

Neural Networks: Training

- Discovered in 1960's
- Applied to neural networks 1970's
- Theoretical progress in 1980's and 1990's
- Profited from increased computational power in 2010's, which allowed applications to large data sets and neural networks of tens or hundreds of layers
- Achieved human and super-human powers in pattern recognition and later on in many other applications

Neural Networks: Deep learning

- Training of a layer with large number (~ 100) layers
- Made possible by the use of GPU's (Nvidia), which accelerated the speed of deep learning by ca. 100times
- Use of many parameters makes it sensitive to overfitting (=too exact adaptation to the training data, not observed in other data from the same area)
- Overfitting reduced by regularization methods: ℓ_2 (decay) or ℓ_1 (sparsity) of weights
- Further tricks used to accelerate the learning algorithm

Applications

- Pattern recognition
- Computer vision
- Speech recognition
- Social network filtering
- Recommendation systems
- Bioinformatics
- AlphaGo
- ...

Thank you for your attention!