Statistics and learning

Neural Networks

Emmanuel Rachelson and Matthieu Vignes

ISAE SupAero

14^{th} March 2013

Context

- branch of IA, which was motivated by a simulation of natural behaviours to model processes generating observed data sets.

Context

- branch of IA, which was motivated by a simulation of natural behaviours to model processes generating observed data sets.
- See: Ant colony algorithms, genetic algorithms, simulated annealing ...

Context

- branch of IA, which was motivated by a simulation of natural behaviours to model processes generating observed data sets.
- See: Ant colony algorithms, genetic algorithms, simulated annealing ...
- Artifical neurons were first introduced by W. McCulloch (a neurophysiologist) and W. Pitts (a logician). An artificial neural network was formally described in 1948 by Alan Turing. In 1959, Rosenblatt piled two layers (input and output) to form a network in order to simulate the retinal functioning for pattern recognition.

Context

- branch of IA, which was motivated by a simulation of natural behaviours to model processes generating observed data sets.
- See: Ant colony algorithms, genetic algorithms, simulated annealing ...
- Artifical neurons were first introduced by W. McCulloch (a neurophysiologist) and W. Pitts (a logician). An artificial neural network was formally described in 1948 by Alan Turing. In 1959, Rosenblatt piled two layers (input and output) to form a network in order to simulate the retinal functioning for pattern recognition.
- Computational developments had to wait until the 90s.

Context

- branch of IA, which was motivated by a simulation of natural behaviours to model processes generating observed data sets.
- See: Ant colony algorithms, genetic algorithms, simulated annealing ...
- Artifical neurons were first introduced by W. McCulloch (a neurophysiologist) and W. Pitts (a logician). An artificial neural network was formally described in 1948 by Alan Turing. In 1959, Rosenblatt piled two layers (input and output) to form a network in order to simulate the retinal functioning for pattern recognition.
- Computational developments had to wait until the 90s.
- Neural network theory benefited the study of the functioning of the brain and provided the basis to create artifical intelligence.

Neural networks

- (Artificial) **neural networks** are (more or less complex) assemblies of elementary components: **artificial neurons** (see next slide).

Neural networks

- (Artificial) **neural networks** are (more or less complex) assemblies of elementary components: **artificial neurons** (see next slide).
- Differ in organisation (architecture) complexity (size, feedbacks), neuron types but also the objectives: (un)supervised learning, optimisation, dynamical system representation.

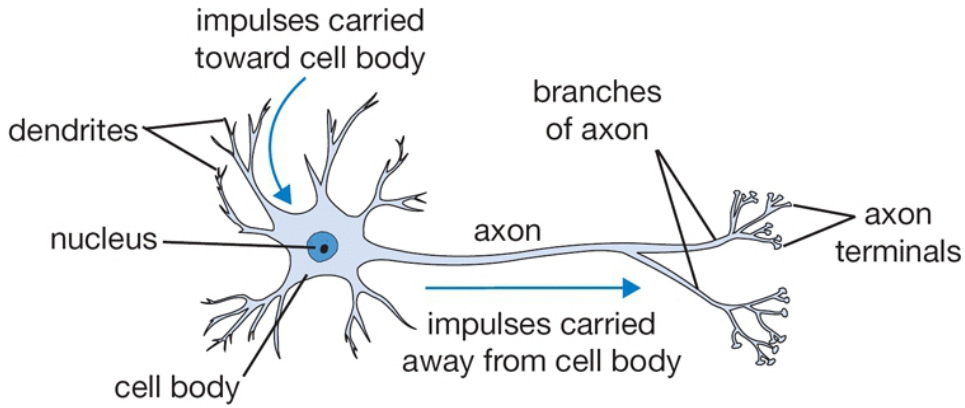
Neural networks

- (Artificial) **neural networks** are (more or less complex) assemblies of elementary components: **artificial neurons** (see next slide).
- Differ in organisation (architecture) complexity (size, feedbacks), neuron types but also the objectives: (un)supervised learning, optimisation, dynamical system representation.
- AI tasks concerned with neural networks: (i) function approximation (regression, time series pred. and model.), (ii) classification (pattern and sequence recognition, novelty detection and sequential decision making) and (iii) data processing (filtering, clustering, blind signal separation, compression).

Neural networks

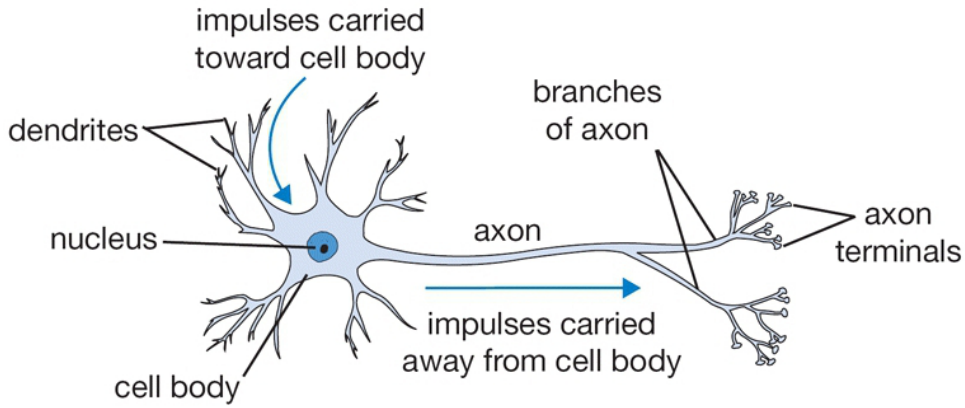
- (Artificial) **neural networks** are (more or less complex) assemblies of elementary components: **artificial neurons** (see next slide).
- Differ in organisation (architecture) complexity (size, feedbacks), neuron types but also the objectives: (un)supervised learning, optimisation, dynamical system representation.
- AI tasks concerned with neural networks: (i) function approximation (regression, time series pred. and model.), (ii) classification (pattern and sequence recognition, novelty detection and sequential decision making) and (iii) data processing (filtering, clustering, blind signal separation, compression).
- Application: system identification and control (vehicle or process control), game-playing with decision making (chess), pattern/sequence recognition (radar systems, face identification, object recognition, spam filters, speech or handwriting recognition), medical diagnosis, financial applications, data mining (knowledge discovery in databases), visualisation.

Biological vs artifical neuron



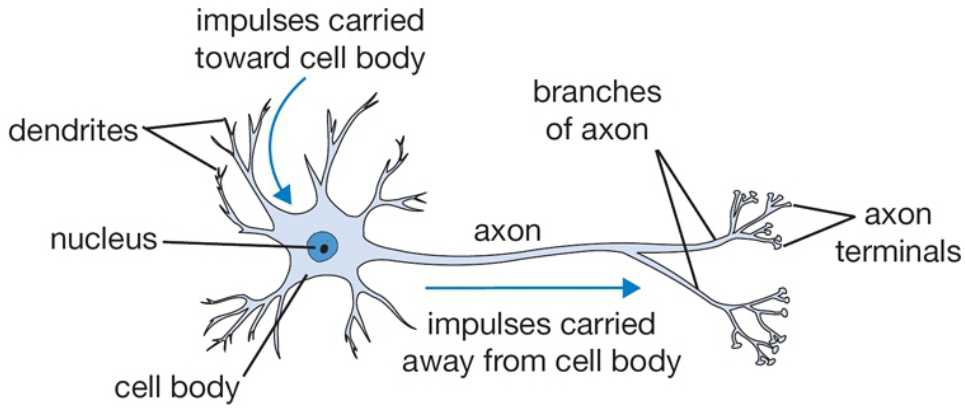
- Components for a (simplified) **biological neuron**:

Biological vs artifical neuron



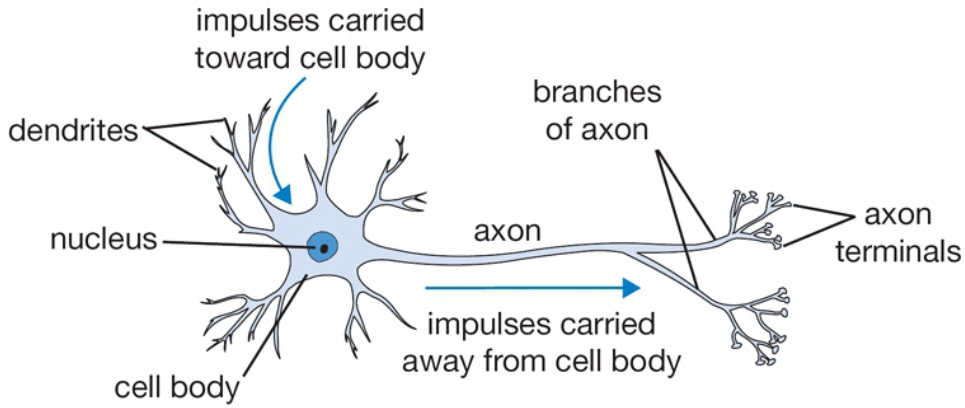
- Components for a (simplified) **biological neuron**:
  - synapses: connecting points to other neurons and nerf or muscle fibres,

Biological vs artifical neuron



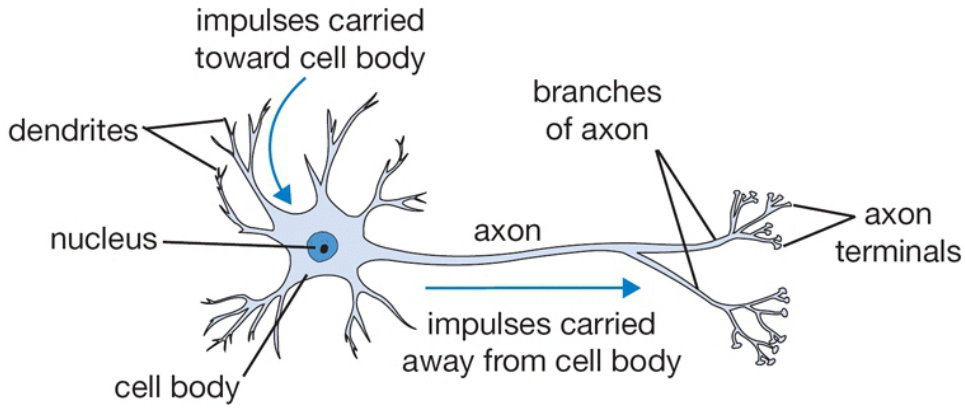
- Components for a (simplified) **biological neuron**:
  - synapses: connecting points to other neurons and nerf or muscle fibres,
  - dendrites: inputs for the neuron,

Biological vs artifical neuron



- Components for a (simplified) **biological neuron**:
  - synapses: connecting points to other neurons and nerf or muscle fibres,
  - dendrites: inputs for the neuron,
  - axon: output point of the neuron towards other neurons or towards fibres and

Biological vs artifical neuron



- Components for a (simplified) **biological neuron**:
  - synapses: connecting points to other neurons and nerf or muscle fibres,
  - dendrites: inputs for the neuron,
  - axon: output point of the neuron towards other neurons or towards fibres and
  - nucleus: activates the output as a function of input stimuli

Biological vs artifical neuron

- Components for a (simplified) **biological neuron**:
  - synapses: connecting points to other neurons and nerf or muscle fibres,
  - dendrites: inputs for the neuron,
  - axon: output point of the neuron towards other neurons or towards fibres and
  - nucleus: activates the output as a function of input stimuli
- Similarly, each **artifical neuron** is defined by an **internal state** $s \in \mathcal{S}$, **input signals** $x_1 \dots x_p$ and an **activation function**:

$$s = h(x_1 \dots x_p) = f\left(\alpha_0 + \sum_{j=1}^{p} \alpha_j x_j\right)$$

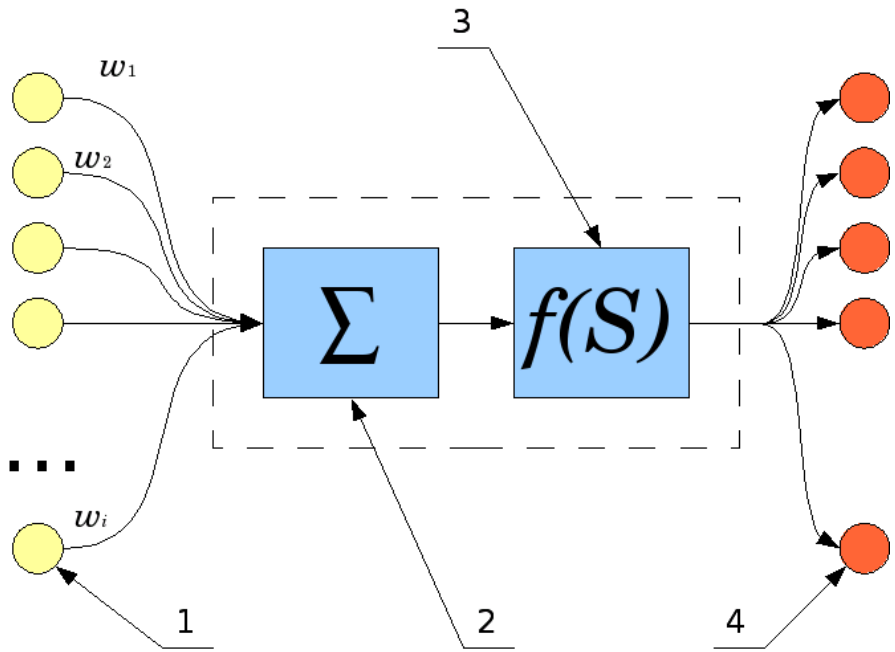
Biological vs artifical neuron

- Components for a (simplified) **biological neuron**:
  - synapses: connecting points to other neurons and nerf or muscle fibres,
  - dendrites: inputs for the neuron,
  - axon: output point of the neuron towards other neurons or towards fibres and
  - nucleus: activates the output as a function of input stimuli
- Similarly, each **artifical neuron** is defined by an **internal state** $s \in \mathcal{S}$, **input signals** $x_1 \dots x_p$ and an **activation function**:

$$s = h(x_1 \dots x_p) = f\left(\alpha_0 + \sum_{j=1}^{p} \alpha_j x_j\right)$$

- In brief, the activation function makes a transformation of a weighted linear combination of the inputs.

Artificial neuron (cont'd)



Artificial neuron (cont'd)

- α_0 is the bias of the neuron, α_j's are its weights and need to be estimated in a learning step. They are the **memory** or **distributed knwoledge** of the network.

Artificial neuron (cont'd)

- α_0 is the bias of the neuron, α_j's are its weights and need to be estimated in a learning step. They are the **memory** or **distributed knwoledge** of the network.
- Mostly used activation functions: linear, sigmoid, step or radial function, *etc.*. Can be deterministic or stochastic.

Artificial neuron (cont'd)

- α_0 is the bias of the neuron, α_j's are its weights and need to be estimated in a learning step. They are the **memory** or **distributed knwoledge** of the network.
- Mostly used activation functions: linear, sigmoid, step or radial function, *etc.*. Can be deterministic or stochastic.
- The choice of the activation function class is linked to that of the learning algorithm.

Artificial neuron (cont'd)

- α_0 is the bias of the neuron, α_j's are its weights and need to be estimated in a learning step. They are the **memory** or **distributed knwoledge** of the network.
- Mostly used activation functions: linear, sigmoid, step or radial function, *etc.*. Can be deterministic or stochastic.
- The choice of the activation function class is linked to that of the learning algorithm.
- We restrict this course to the elementary (feedforward and not recurrent) static structure of the network for supervised learning.

Multilayer perceptron

- a multilayer perceptron is composed of successive **layers**.

Multilayer perceptron

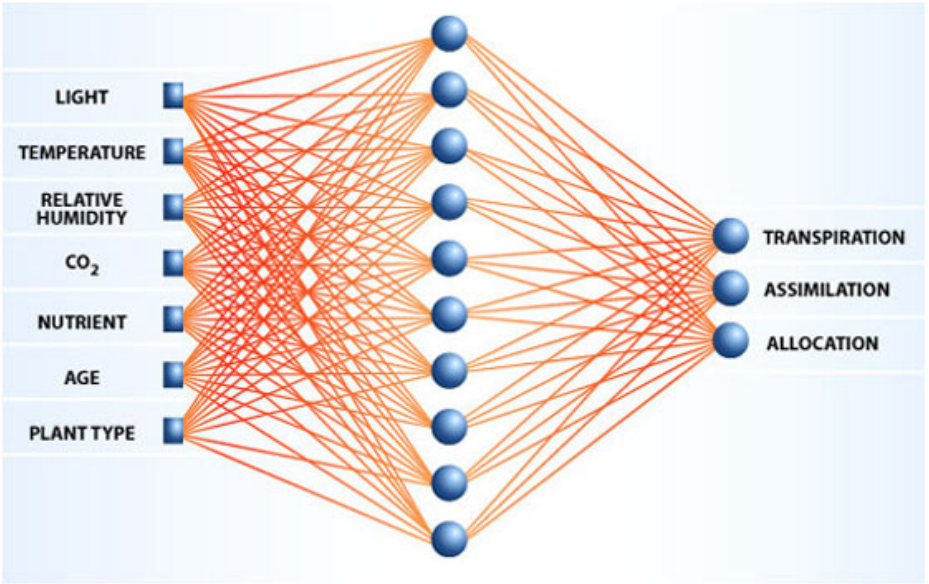
- a multilayer perceptron is composed of successive **layers**.
- A layer is a set of neurons which are not connected between them but can be connected to the previous and following layer.

Multilayer perceptron

- a multilayer perceptron is composed of successive **layers**.
- A layer is a set of neurons which are not connected between them but can be connected to the previous and following layer.
- An input layer reads input signals (one neuron/input) and an output layer gives the system response. In between several **hidden** layers transfer and transform the signal.

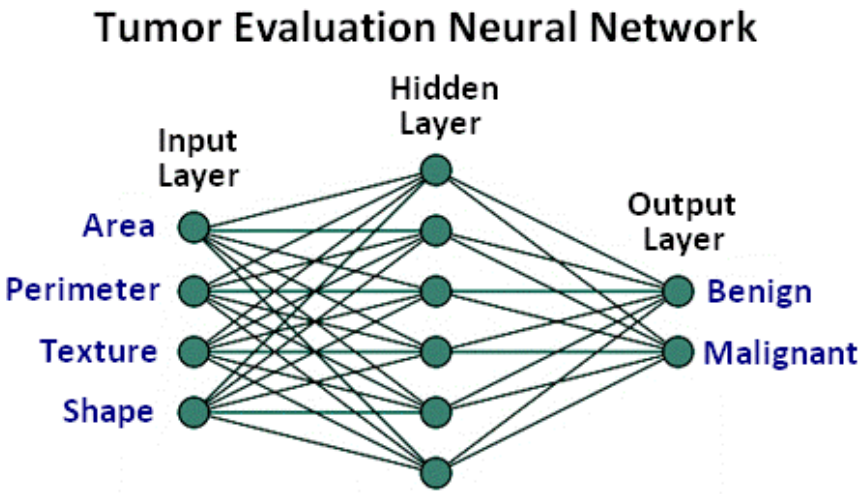
Multilayer perceptron

- a multilayer perceptron is composed of successive **layers**.
- A layer is a set of neurons which are not connected between them but can be connected to the previous and following layer.
- An input layer reads input signals (one neuron/input) and an output layer gives the system response. In between several **hidden** layers transfer and transform the signal.



Multilayer perceptron

- a multilayer perceptron is composed of successive **layers**.
- A layer is a set of neurons which are not connected between them but can be connected to the previous and following layer.
- An input layer reads input signals (one neuron/input) and an output layer gives the system response. In between several **hidden** layers transfer and transform the signal.



Neural networks

one layer is enough

Transfer function

A multilayer perceptron does a transformation of the input variables in the form:

$$Y = \varphi\left(X_1 \dots X_p; \alpha\right),$$

where α is the vector with elements α_{jkl}: parameter for entry j of neuron k of layer l. Note that the entry layer ($l = 0$) is not parameterised.

Neural networks

one layer is enough

Transfer function

A multilayer perceptron does a transformation of the input variables in the form:

$$Y = \varphi\left(X_1 \dots X_p; \alpha\right),$$

where α is the vector with elements α_{jkl}: parameter for entry j of neuron k of layer l. Note that the entry layer ($l = 0$) is not parameterised.

Theorem (of "universal approximation")

A not overwhelming structure with one hidden layer is enough to account for most classical problems in statistics modelling or learning. Stems from the approximation of any **regular** *function with arbitrary accuracy in a finite domain of variable space by a neural networks with a finite number of neurons in a unique hidden layer and a linear output neuron.*

Learning

- We have a n-learning sample: $(x_1^{(i)}, \ldots, x_p^{(i)}, y^{(i)})$ $(i = 1 \ldots n)$, for variables X and Y.

Learning

- We have a n-learning sample: $(x_1^{(i)}, \ldots, x_p^{(i)}, y^{(i)})$ $(i = 1 \ldots n)$, for variables X and Y.
- In the case of linear regression with a single output: $y = \varphi(x; \alpha, \beta) = \beta_0 + {}^\top \beta z$, with $z_k = f(\alpha_{k,0} + {}^\top \alpha x)$, $\forall k = 1 \ldots q$.

Learning

- We have a n-learning sample: $(x_1^{(i)}, \ldots, x_p^{(i)}, y^{(i)})$ $(i = 1 \ldots n)$, for variables X and Y.
- In the case of linear regression with a single output: $y = \varphi(x; \alpha, \beta) = \beta_0 + {}^\top \beta z$, with $z_k = f(\alpha_{k,0} + {}^\top \alpha x)$, $\forall k = 1 \ldots q$.
- Parameters can be estimated using a least square criterion by minimising the quadratic loss $Q(\alpha, \beta) = \sum_i Q^{(i)} = \sum_i \left[y^{(i)} - \varphi(x^{(i)}; \alpha, \beta) \right]^2$.

Learning

- We have a n-learning sample: $(x_1^{(i)}, \ldots, x_p^{(i)}, y^{(i)})$ $(i = 1 \ldots n)$, for variables X and Y.
- In the case of linear regression with a single output: $y = \varphi(x; \alpha, \beta) = \beta_0 + {}^\top \beta z$, with $z_k = f(\alpha_{k,0} + {}^\top \alpha x)$, $\forall k = 1 \ldots q$.
- Parameters can be estimated using a least square criterion by minimising the quadratic loss $Q(\alpha, \beta) = \sum_i Q^{(i)} = \sum_i \left[y^{(i)} - \varphi(x^{(i)}; \alpha, \beta)\right]^2$.
- for classification, this can be generalised to any differentiable loss function in particular entropy.

Learning

- We have a n-learning sample: $(x_1^{(i)}, \ldots, x_p^{(i)}, y^{(i)})$ $(i = 1 \ldots n)$, for variables X and Y.
- In the case of linear regression with a single output: $y = \varphi(x; \alpha, \beta) = \beta_0 + {}^\top \beta z$, with $z_k = f(\alpha_{k,0} + {}^\top \alpha x)$, $\forall k = 1 \ldots q$.
- Parameters can be estimated using a least square criterion by minimising the quadratic loss $Q(\alpha, \beta) = \sum_i Q^{(i)} = \sum_i \left[y^{(i)} - \varphi(x^{(i)}; \alpha, \beta) \right]^2$.
- for classification, this can be generalised to any differentiable loss function in particular entropy.
- Several classical algorithms exist and are generally based on an evaluation of the gradient by retro-propagation.

Error retropropagation step

- Consists in evaluating the derivative of the cost function in the direction of all parameters.

Error retropropagation step

- Consists in evaluating the derivative of the cost function in the direction of all parameters.
- If we write $z_k^{(i)} = f(\alpha_{k,0} + {}^\top \alpha x^{(i)})$ and $z^{(i)} = (z_1^{(i)}, \ldots, z_q^{(i)}$, then:

$$\begin{aligned}
\frac{\partial Q^{(i)}}{\partial \beta_k} &= -2(y^{(i)} - \varphi(x^{(i)}))({}^\top \beta z^{(i)}) z_k^{(i)} = \delta_i z_k^{(i)} \\
\frac{\partial Q^{(i)}}{\partial \alpha_{kj}} &= -2(y^{(i)} - \varphi(x^{(i)}))({}^\top \beta z^{(i)}) \beta_k f'({}^\top \alpha_k x^{(i)}) x_p^{(i)} = s_{ki} x_p^{(i)},
\end{aligned}$$

Error retropropagation step

- Consists in evaluating the derivative of the cost function in the direction of all parameters.
- If we write $z_k^{(i)} = f(\alpha_{k,0} + {}^\top \alpha x^{(i)})$ and $z^{(i)} = (z_1^{(i)}, \ldots, z_q^{(i)}$, then:

$$\begin{aligned}
\frac{\partial Q^{(i)}}{\partial \beta_k} &= -2(y^{(i)} - \varphi(x^{(i)}))({}^\top \beta z^{(i)}) z_k^{(i)} = \delta_i z_k^{(i)} \\
\frac{\partial Q^{(i)}}{\partial \alpha_{kj}} &= -2(y^{(i)} - \varphi(x^{(i)}))({}^\top \beta z^{(i)}) \beta_k f'({}^\top \alpha_k x^{(i)}) x_p^{(i)} = s_{ki} x_p^{(i)},
\end{aligned}$$

- where δ_i (resp. s_{ki}) is the current error term for output (resp. hidden layer neuron). They verify:

$$s_{ki} = f'({}^\top \alpha_k x^{(i)}) \beta_k \delta_i.$$

Error retropropagation step

- Consists in evaluating the derivative of the cost function in the direction of all parameters.
- If we write $z_k^{(i)} = f(\alpha_{k,0} + {}^\top \alpha x^{(i)})$ and $z^{(i)} = (z_1^{(i)}, \ldots, z_q^{(i)}$, then:

$$\begin{aligned}
\frac{\partial Q^{(i)}}{\partial \beta_k} &= -2(y^{(i)} - \varphi(x^{(i)}))({}^\top \beta z^{(i)}) z_k^{(i)} = \delta_i z_k^{(i)} \\
\frac{\partial Q^{(i)}}{\partial \alpha_{kj}} &= -2(y^{(i)} - \varphi(x^{(i)}))({}^\top \beta z^{(i)}) \beta_k f'({}^\top \alpha_k x^{(i)}) x_p^{(i)} = s_{ki} x_p^{(i)},
\end{aligned}$$

- where δ_i (resp. s_{ki}) is the current error term for output (resp. hidden layer neuron). They verify:

$$s_{ki} = f'({}^\top \alpha_k x^{(i)}) \beta_k \delta_i.$$

- They are evaluated in 2 steps: "before" with current weight values, input values are applied to get evaluations $\hat{\varphi}(x^{(i)})$ and "after" to determine δ_i which are back-propagated to compute s_{ki} and hence access gradient evaluations.

Optimisation algorithm

- Once gradient computation is possible, several alogorithms can be used.

Optimisation algorithm

- Once gradient computation is possible, several alogorithms can be used.
- An elementary one is an iterative update in the steepest direction. Since $\nabla.Q$ points in the largest increasing error direction, moving in the alternative direction allows us to decrease Q:

Optimisation algorithm

- Once gradient computation is possible, several alogorithms can be used.
- An elementary one is an iterative update in the steepest direction. Since $\nabla.Q$ points in the largest increasing error direction, moving in the alternative direction allows us to decrease Q:
-

$$\begin{aligned}
\beta_k^{(r+1)} &= \beta_k^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q^{(i)}}{\partial \beta_k^{(r)}} \\
\alpha_{kp}^{(r+1)} &= \alpha_{kp}^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q^{(i)}}{\partial \alpha_{kp}^{(r)}}
\end{aligned}$$

Optimisation algorithm

- Once gradient computation is possible, several alogorithms can be used.
- An elementary one is an iterative update in the steepest direction. Since $\nabla.Q$ points in the largest increasing error direction, moving in the alternative direction allows us to decrease Q:
- $$\begin{aligned} \beta_k^{(r+1)} &= \beta_k^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q^{(i)}}{\partial \beta_k^{(r)}} \\ \alpha_{kp}^{(r+1)} &= \alpha_{kp}^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q^{(i)}}{\partial \alpha_{kp}^{(r)}} \end{aligned}$$
- τ is the **learning rate**. It can be kept to a default value, determined bu the user or varied along the iterations.

Optimisation algorithm

- Once gradient computation is possible, several alogorithms can be used.
- An elementary one is an iterative update in the steepest direction. Since $\nabla.Q$ points in the largest increasing error direction, moving in the alternative direction allows us to decrease Q:
- $$\begin{aligned} \beta_k^{(r+1)} &= \beta_k^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q^{(i)}}{\partial \beta_k^{(r)}} \\ \alpha_{kp}^{(r+1)} &= \alpha_{kp}^{(r)} - \tau \sum_{i=1}^{n} \frac{\partial Q^{(i)}}{\partial \alpha_{kp}^{(r)}} \end{aligned}$$
- τ is the **learning rate**. It can be kept to a default value, determined bu the user or varied along the iterations.
- Refinements include second order expansion, stochastic modifications for not being trapped in a local minimum, adding an inertia term to avoid oscillations...Be aware of the algorithm you use and be critical

Controling for complexity

- Parameter choice:

Controling for complexity

- Parameter choice:
  1. number of hidden layers (1 or 2 in general) and number of neurons per layer. They lead the bias/variance compromise and hence the learning vs prediction quality.

Controling for complexity

- Parameter choice:
  1. number of hidden layers (1 or 2 in general) and number of neurons per layer. They lead the bias/variance compromise and hence the learning vs prediction quality.
  2. iteration number, threshold for error and decay regularisation term

Controling for complexity

- Parameter choice:
  1. number of hidden layers (1 or 2 in general) and number of neurons per layer. They lead the bias/variance compromise and hence the learning vs prediction quality.
  2. iteration number, threshold for error and decay regularisation term
  3. learning rate τ.

Controling for complexity

- Parameter choice:
  1. number of hidden layers (1 or 2 in general) and number of neurons per layer. They lead the bias/variance compromise and hence the learning vs prediction quality.
  2. iteration number, threshold for error and decay regularisation term
  3. learning rate τ.
- It's mainly about avoiding over-fitting: learning and test samples, CV or bootstraps.

Controling for complexity

- Parameter choice:
  1. number of hidden layers (1 or 2 in general) and number of neurons per layer. They lead the bias/variance compromise and hence the learning vs prediction quality.
  2. iteration number, threshold for error and decay regularisation term
  3. learning rate τ.
- It's mainly about avoiding over-fitting: learning and test samples, CV or bootstraps.
- Computation can be stopped when error on validation sample gets worse, while error on learning sample still improves.

Controling for complexity

- Parameter choice:
  1. number of hidden layers (1 or 2 in general) and number of neurons per layer. They lead the bias/variance compromise and hence the learning vs prediction quality.
  2. iteration number, threshold for error and decay regularisation term
  3. learning rate τ.
- It's mainly about avoiding over-fitting: learning and test samples, CV or bootstraps.
- Computation can be stopped when error on validation sample gets worse, while error on learning sample still improves.
- the neuron number (per layer) can be chosen by CV.

Controling for complexity

- Parameter choice:
  1. number of hidden layers (1 or 2 in general) and number of neurons per layer. They lead the bias/variance compromise and hence the learning vs prediction quality.
  2. iteration number, threshold for error and decay regularisation term
  3. learning rate τ.
- It's mainly about avoiding over-fitting: learning and test samples, CV or bootstraps.
- Computation can be stopped when error on validation sample gets worse, while error on learning sample still improves.
- the neuron number (per layer) can be chosen by CV.
- a **decay** (penalty term of the form $Q(\theta) + \gamma \parallel \theta \parallel$) can avoid any question about the number of neurons by choosing it large and then restricting active parts of the activation functions by choosing γ only by CV.

Let's take a break before the practical sesson

Only 2 sessions left: regression trees and MCMC !