Introduction to Neural Networks

First & Second lectures

E. Scornet

1. Neural Network architecture
   - Neurons
   - A historical model/algorithm - the perceptron
   - Going beyond perceptron - multilayer neural networks
   - Neural network training

2. Hyperparameters
   - How to choose the number of hidden layers/neurons?
   - Activation functions
   - Output units
   - Loss functions
   - Weight initialization

3. Regularization
   - Penalization
   - Dropout
   - Batch normalization
   - Early stopping

4. All in all

Outline

Outline

What is a neuron?

This is a real neuron!

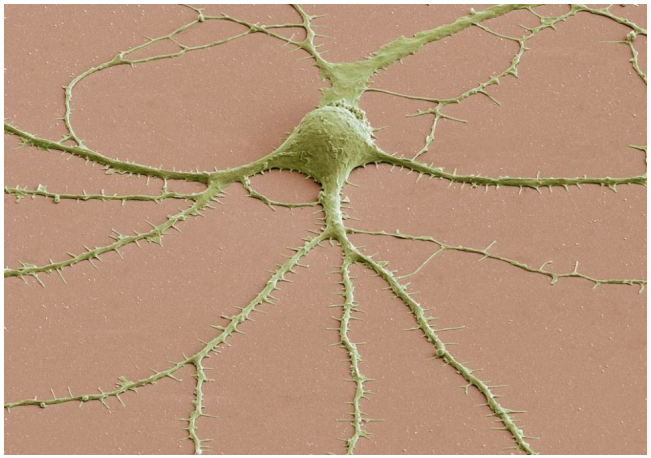


Figure: Real Neuron

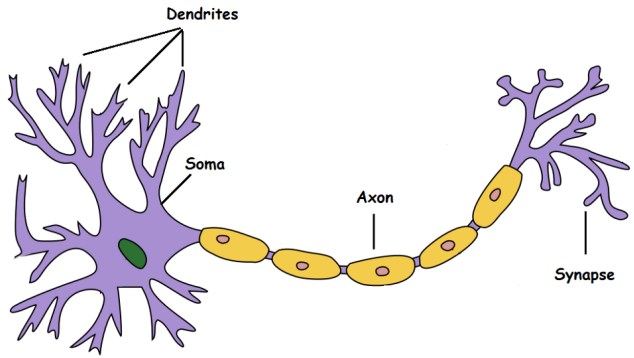


Figure: Real Neuron - diagram

What is a neuron?

This is a real neuron!

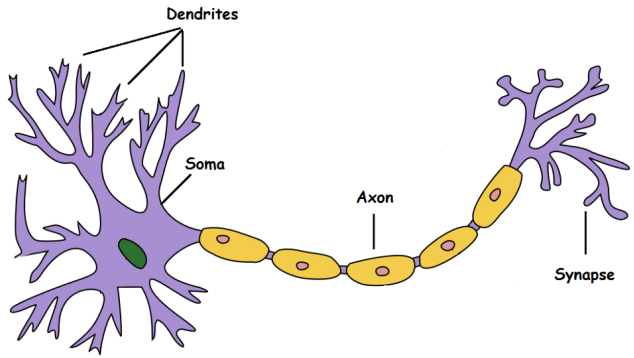


Figure: Real Neuron - diagram

Two main approaches to simulate intelligent behaviour:

- Connectionism: using connected circuits / neural networks.
- Symbolic AI / Old Good AI: combining human-readable representations of problems.

McCulloch and Pitts neuron - 1943

["A logical calculus of the ideas immanent in nervous activity", McCulloch and Pitts 1943]

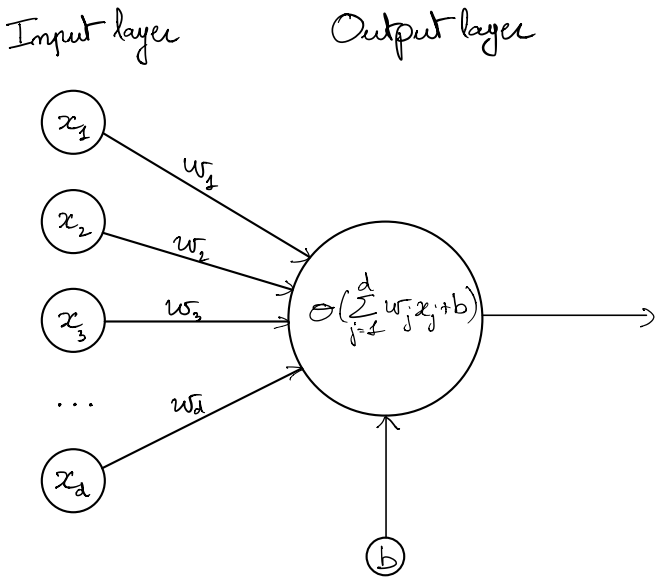
In 1943, portrayed with a simple electrical circuit by neurophysiologist Warren McCulloch and mathematician Walter Pitts.

McCulloch and Pitts neuron - 1943

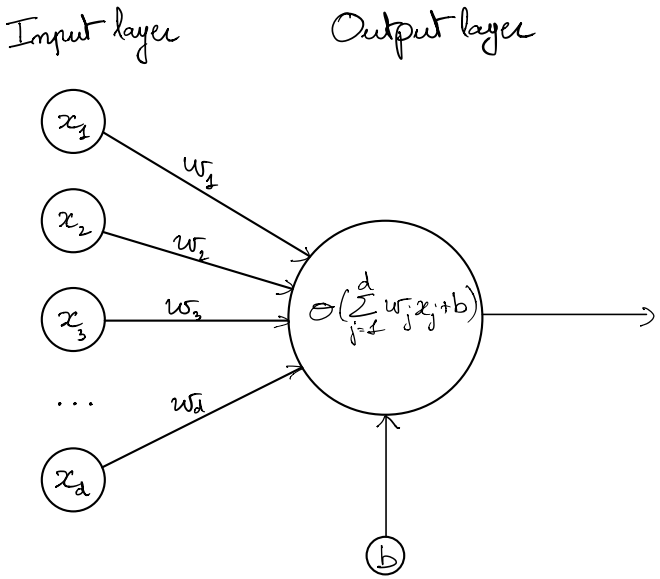
["A logical calculus of the ideas immanent in nervous activity", McCulloch and Pitts 1943]

In 1943, portrayed with a simple electrical circuit by neurophysiologist Warren McCulloch and mathematician Walter Pitts.

A McCulloch-Pitts neuron takes binary inputs, computes a weighted sum and returns 0 if the result is below threshold and 1 otherwise.



McCulloch and Pitts neuron - 1943



Donald Hebb took the idea further by proposing that neural pathways strengthen over each successive use, especially between neurons that tend to fire at the same time.

[*The organization of behavior: a neuropsychological theory*, Hebb 1949]

Outline

Perceptron - 1958

In the late 50s, Frank Rosenblatt, a psychologist at Cornell, worked on decision systems present in the eye of a fly, which determine its flee response.

In 1958, he proposed the idea of a Perceptron, calling it Mark I Perceptron. It was a system with a simple input-output relationship, modelled on a McCulloch-Pitts neuron.

["Perceptron simulation experiments", Rosenblatt 1960]

Perceptron diagram

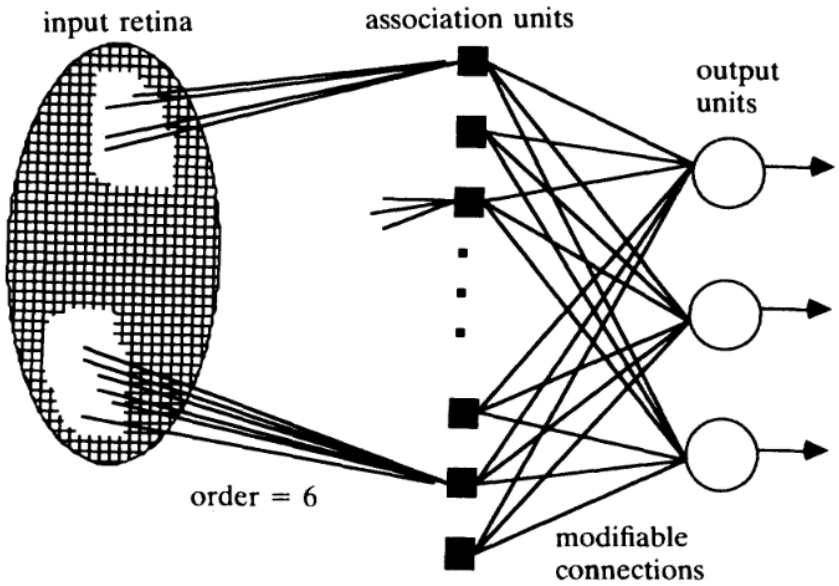
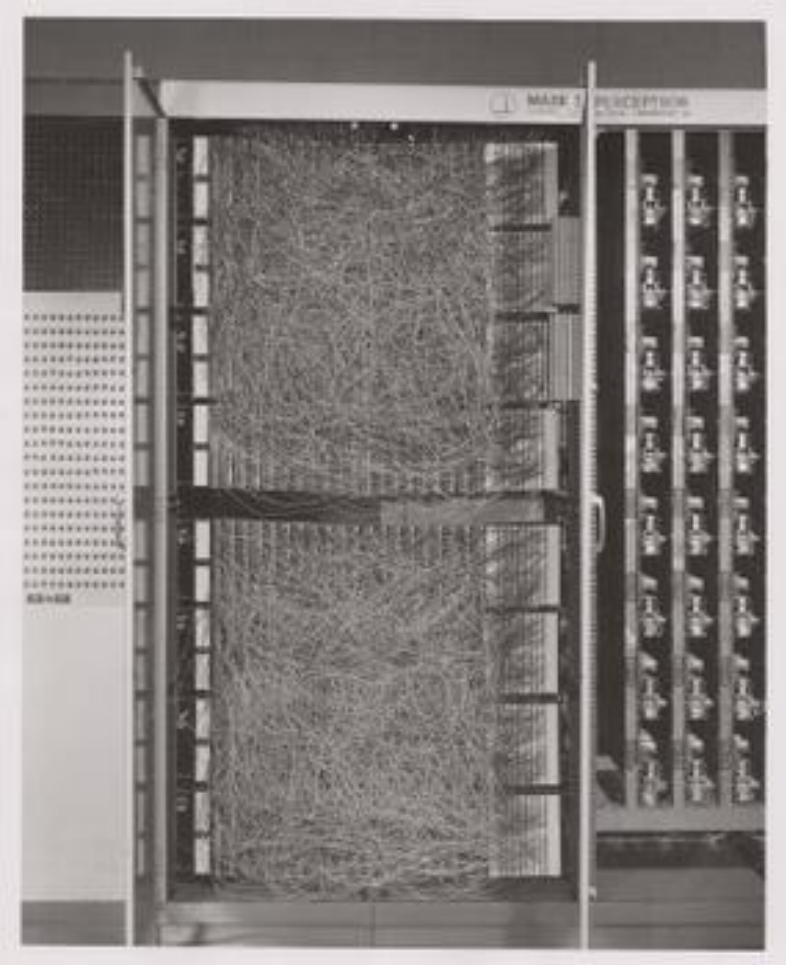


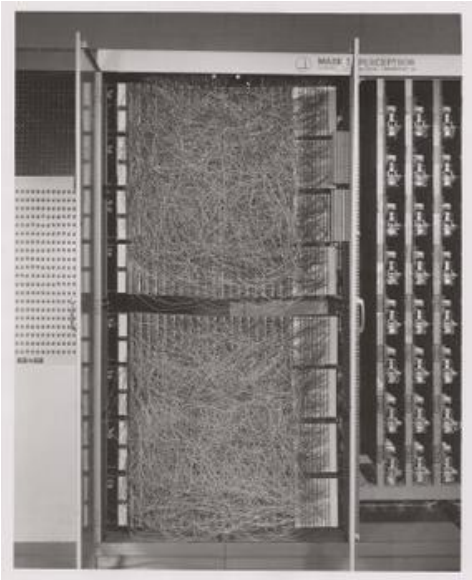
Figure: True representation of perceptron

The connections between the input and the first hidden layer cannot be optimized!

Perceptron Machine



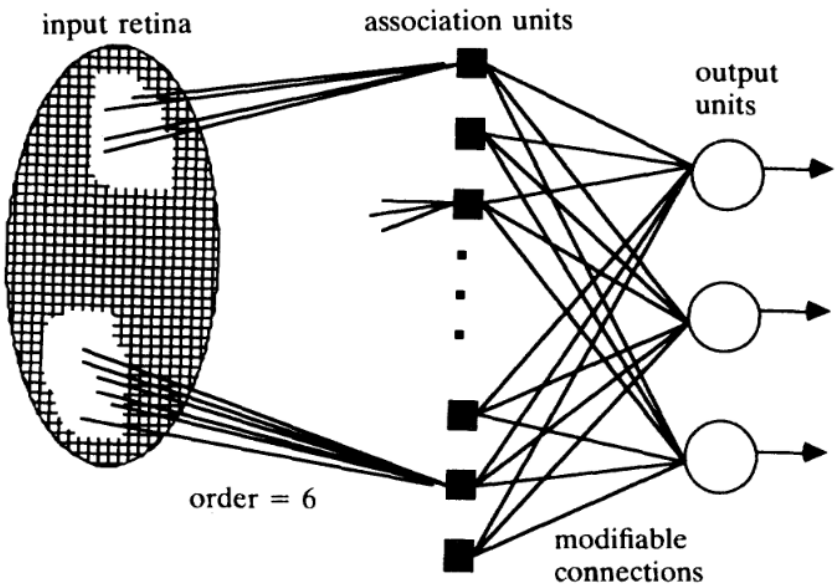
Perceptron Machine



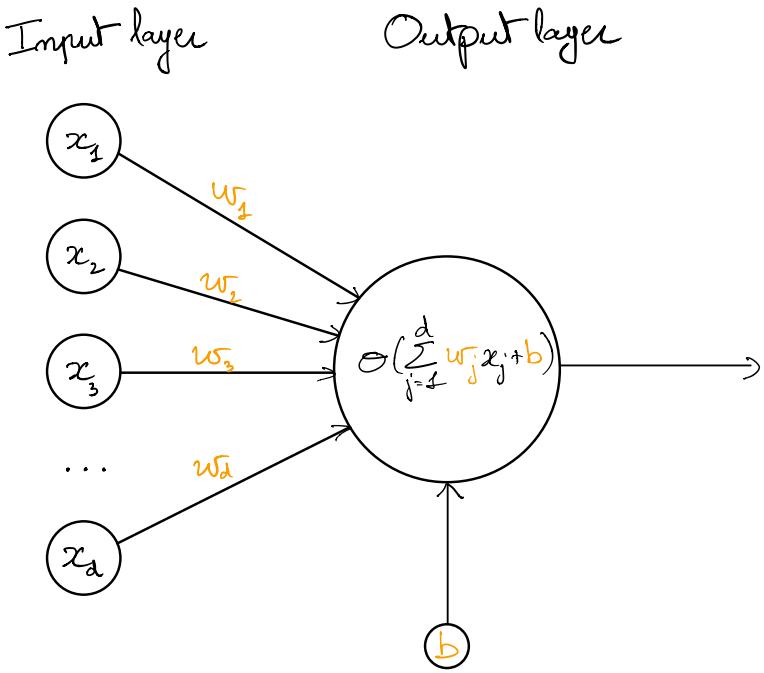
First implementation: Mark I Perceptron (1958).

- The machine was connected to a camera (20x20 photocells, 400-pixel image)
- Patchboard: allowed experimentation with different combinations of input features
- Potentiometers: that implement the adaptive weights

Parameters of the perceptron



Parameters of the perceptron



Activation function: $\sigma(z) = \mathbb{1}_{z>0}$

Learnable parameters:

- Weights $\mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{R}^d$
- Bias: b

How do we estimate $(\mathbf{w}, b)$?

The Perceptron Algorithm

We have $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, with $y_i \in \{-1, 1\}$. To ease notations, we put $\tilde{\mathbf{w}} = (w_1, \ldots, w_d, b)$ and $\tilde{\mathbf{x}}_i = (\mathbf{x}_i, 1)$

Perceptron Algorithm - first (iterative) learning algorithm

- Start with $\tilde{\mathbf{w}} = 0$.
- Repeat over all samples:
  - if $y_i < \tilde{\mathbf{w}}, \tilde{\mathbf{x}}_i > \leq 0$ modify $\tilde{\mathbf{w}}$ into $\tilde{\mathbf{w}} + y_i \tilde{\mathbf{x}}_i$,
  - otherwise do not modify $\tilde{\mathbf{w}}$.

The Perceptron Algorithm

We have $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, with $y_i \in \{-1, 1\}$. To ease notations, we put $\tilde{\mathbf{w}} = (w_1, \ldots, w_d, b)$ and $\tilde{\mathbf{x}}_i = (\mathbf{x}_i, 1)$

Perceptron Algorithm - first (iterative) learning algorithm

- Start with $\tilde{\mathbf{w}} = 0$.
- Repeat over all samples:
  - if $y_i < \tilde{\mathbf{w}}, \tilde{\mathbf{x}}_i > \leq 0$ modify $\tilde{\mathbf{w}}$ into $\tilde{\mathbf{w}} + y_i \tilde{\mathbf{x}}_i$,
  - otherwise do not modify $\tilde{\mathbf{w}}$.

Gradient descent:

1. Start with $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}_0$
2. Update $\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} - \eta \nabla \mathcal{L}(\tilde{\mathbf{w}})$, where $\mathcal{L}$ is the loss to be minimized.
3. Stop when $\tilde{\mathbf{w}}$ does not vary too much.

Perceptron as a stochastic gradient descent

Perceptron Algorithm

- Repeat over all samples:
  - if $y_i < \tilde{\mathbf{w}}, \tilde{\mathbf{x}}_i > \leq 0$ modify $\tilde{\mathbf{w}}$ into $\tilde{\mathbf{w}} + y_i \tilde{\mathbf{x}}_i$,
  - otherwise do not modify $\tilde{\mathbf{w}}$.

A sample is misclassified if $y_i < \tilde{\mathbf{w}}, \tilde{\mathbf{x}}_i > \leq 0$.
Thus we want to minimize the loss

$$\mathcal{L}(\tilde{\mathbf{w}}) = -\sum_{i \in \mathcal{M}_{\tilde{\mathbf{w}}}} y_i < \tilde{\mathbf{w}}, \tilde{\mathbf{x}}_i >,$$

where $\mathcal{M}_{\tilde{\mathbf{w}}}$ is the set of indices misclassified by the hyperplane $\tilde{\mathbf{w}}$.

Stochastic Gradient Descent:

1. Select randomly $i \in \mathcal{M}_{\tilde{\mathbf{w}}}$
2. Update $\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} - \eta \nabla \mathcal{L}_i(\tilde{\mathbf{w}}) = \tilde{\mathbf{w}} + \eta y_i \tilde{\mathbf{x}}_i$

Perceptron algorithm: $\eta = 1$.

Convergence of the Perceptron algorithm

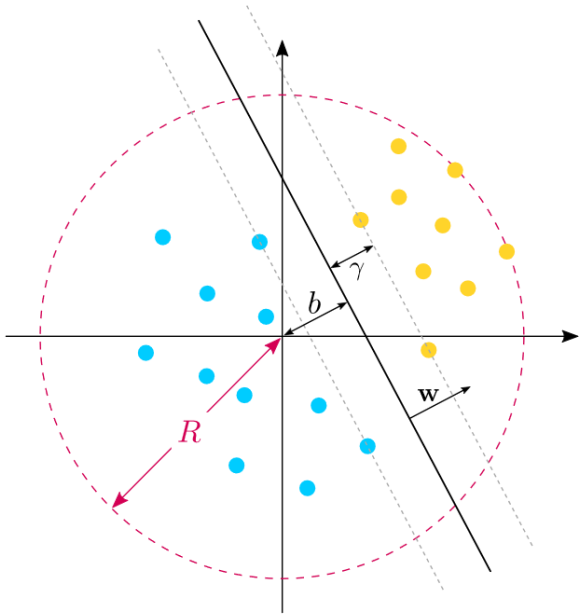


Figure: From `http://image.diku.dk/kstensbo/notes/perceptron.pdf`

Let $R = \max_i \|\mathbf{x}_i\|$. Let $\tilde{\mathbf{w}}^\star$ be the optimal hyperplane of margin

$$\gamma = \min_i y_i \langle \tilde{\mathbf{w}}^\star, \tilde{\mathbf{x}}_i \rangle,$$

with

$$\|\tilde{\mathbf{w}}^\star\| = 1.$$

Convergence of the Perceptron algorithm

Let $R = \max_i \|\mathbf{x}_i\|$. Let $\tilde{\mathbf{w}}^\star$ be the optimal hyperplane of margin

$$\gamma = \min_i y_i \langle \tilde{\mathbf{w}}^\star, \tilde{\mathbf{x}}_i \rangle,$$

with

$$\|\tilde{\mathbf{w}}^\star\| = 1.$$

Theorem (Block 1962; Novikoff 1963)

Assume that the training set $\mathcal{D}_n = \{(\boldsymbol{x}_1, y_i), \ldots, (\boldsymbol{x}_n, y_n)\}$ *is linearly separable* ($\gamma > 0$). *Start with* $\tilde{\boldsymbol{w}}_0 = 0$. *Then the number of updates* k *of the perceptron algorithm is bounded by*

$$k + 1 \leq \frac{1 + R^2}{\gamma^2}.$$

Perceptron - Summary and drawbacks

Perceptron algorithm

- We have a data set $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
- We use the perceptron algorithm to learn the weight vector $\mathbf{w}$ and the bias b.
- We predict using $f_{(\mathbf{w},b)}(\mathbf{x}) = \mathbb{1}_{\langle \mathbf{w}, \mathbf{x} \rangle + b > 0}$.

Limitations

Perceptron - Summary and drawbacks

Perceptron algorithm

- We have a data set $\mathcal{D}_n = \{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$
- We use the perceptron algorithm to learn the weight vector $\mathbf{w}$ and the bias b.
- We predict using $f_{(\mathbf{w},b)}(\mathbf{x}) = \mathbb{1}_{\langle \mathbf{w}, \mathbf{x} \rangle + b > 0}$.

Limitations

- The decision frontier is linear! Too simple model.
- The perceptron algorithm does not converge if data are not linearly separable: in this case, the algorithm must not be used.
- In practice, we do not know if data are linearly separable... Perceptron should never be used!

Perceptron will make machine intelligent... or not!

The Perceptron project led by Rosenblatt was funded by the US Office of Naval Research.

The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. Later perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech and writing in another language [...]

Press conference, 7 July 1958, New York Times.

For an extensive study of the perceptron, [*Principles of neurodynamics. perceptrons and the theory of brain mechanisms*, Rosenblatt 1961]

AI winter

In 1969, Minsky and Papert exhibit the fact that it was difficult for perceptron to

- detect parity (number of activated pixels)
- detect connectedness (are the pixels connected?)
- represent simple non linear function like XOR

There is no reason to suppose that any of [the virtue of perceptrons] carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension is sterile. Perhaps some powerful convergence theorem will be discovered, or some profound reason for the failure to produce an interesting "learning theorem" for the multilayered machine will be found.

["Perceptrons.", Minsky and Papert 1969]

This book is the starting point of the period known as "AI winter", a significant decline in funding of neural network research.

Controversy between Rosenblatt, Minsky, Papert:

["A sociological study of the official history of the perceptrons controversy", Olazaran 1996]

Outline

Moving forward - ADALINE, MADALINE

In 1959 at Stanford, Bernard Widrow and Marcian Hoff developed AdaLinE (ADAptive LINear Elements) and **MAdaLinE** (Multiple AdaLinE) the latter being the first network successfully applied to a real world problem.

[*Adaptive switching circuits*, Bernard Widrow and Hoff 1960]

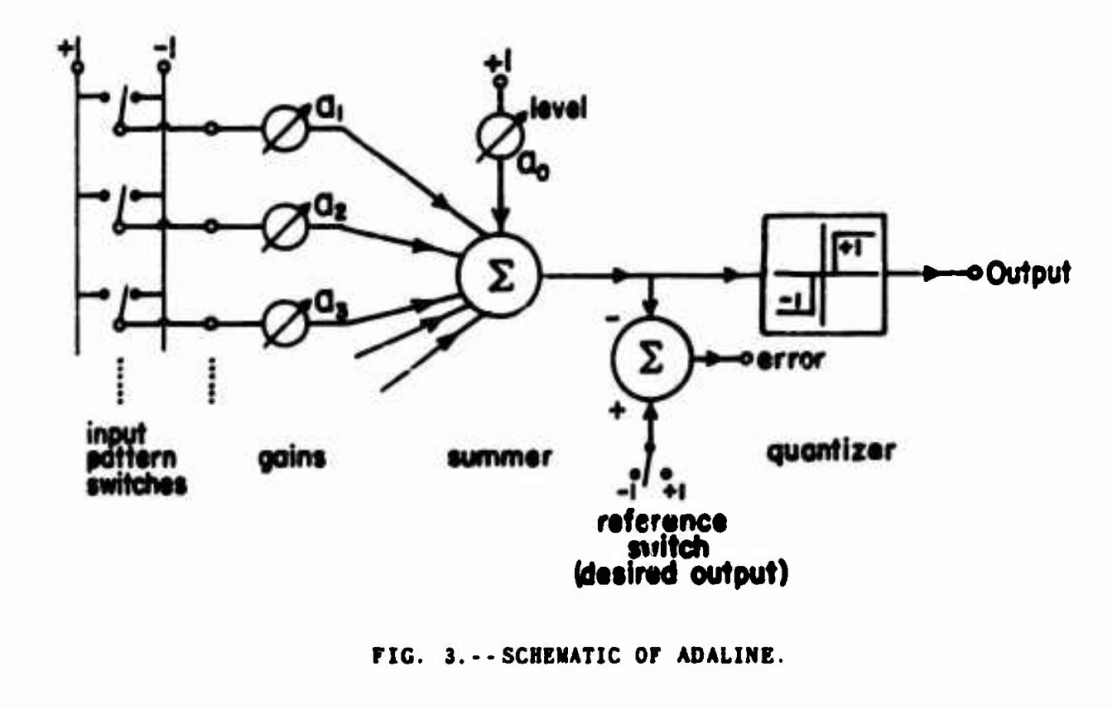


FIG. 3.--SCHEMATIC OF ADALINE.

Moving forward - ADALINE, MADALINE

[*Adaptive switching circuits*, Bernard Widrow and Hoff 1960]

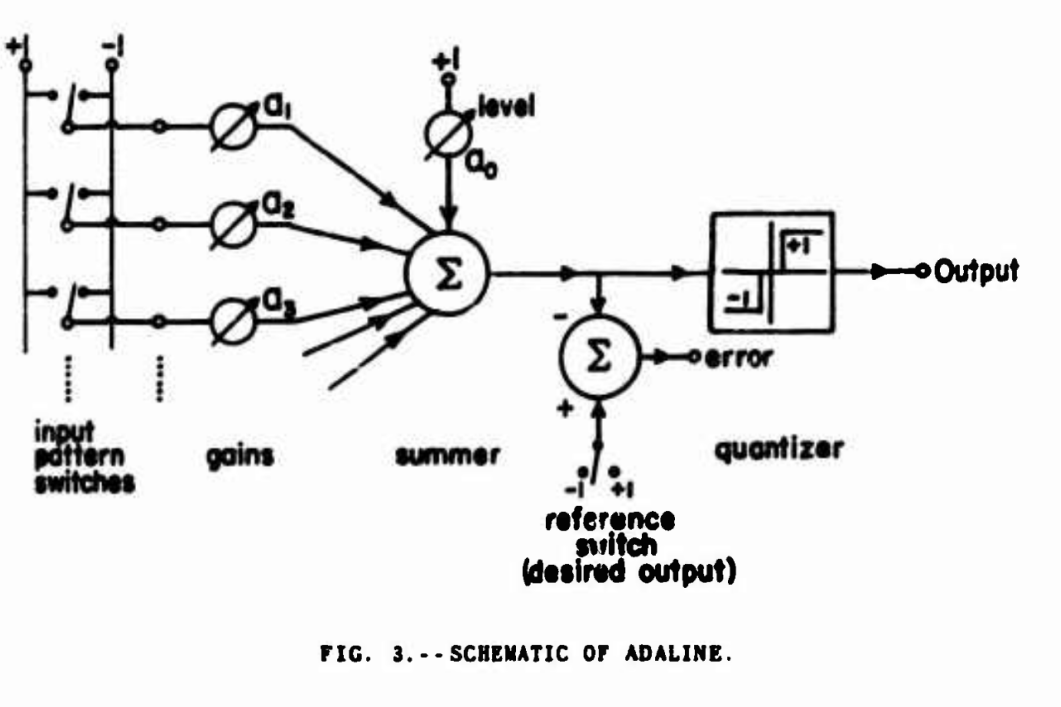


FIG. 3.--SCHEMATIC OF ADALINE.

- Loss: square difference between a weighted sum of inputs and the output
- Optimization procedure: trivial gradient descent

MADALINE

Many Adalines: network with one hidden layer composed of many Adaline units.

["Madaline Rule II: a training algorithm for neural networks", Winter and Widrow 1988]

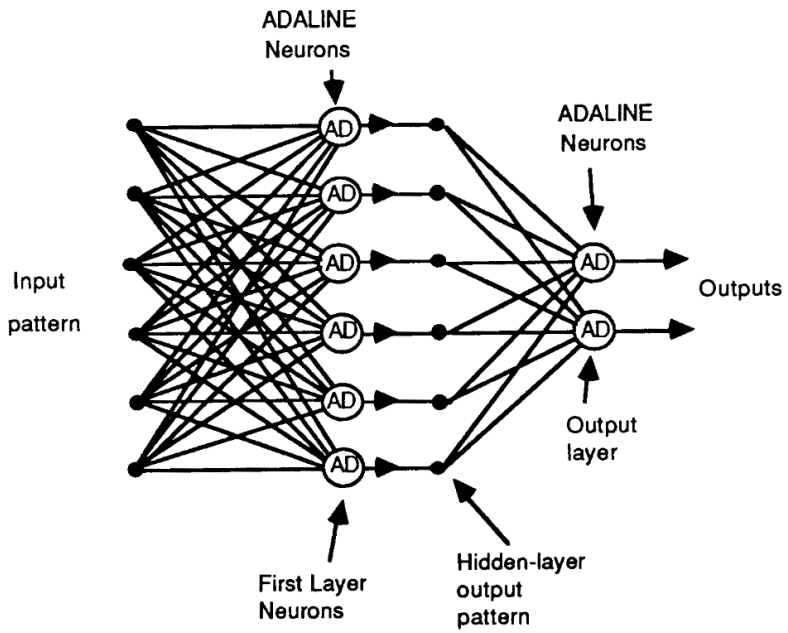


Figure 1: Layered feed-forward ADALINE network.

MADALINE

Many Adalines: network with one hidden layer composed of many Adaline units.

["Madaline Rule II: a training algorithm for neural networks", Winter and Widrow 1988]

Applications:

- Speech and pattern recognition

  ["Real-Time Adaptive Speech-Recognition System", Talbert et al. 1963]

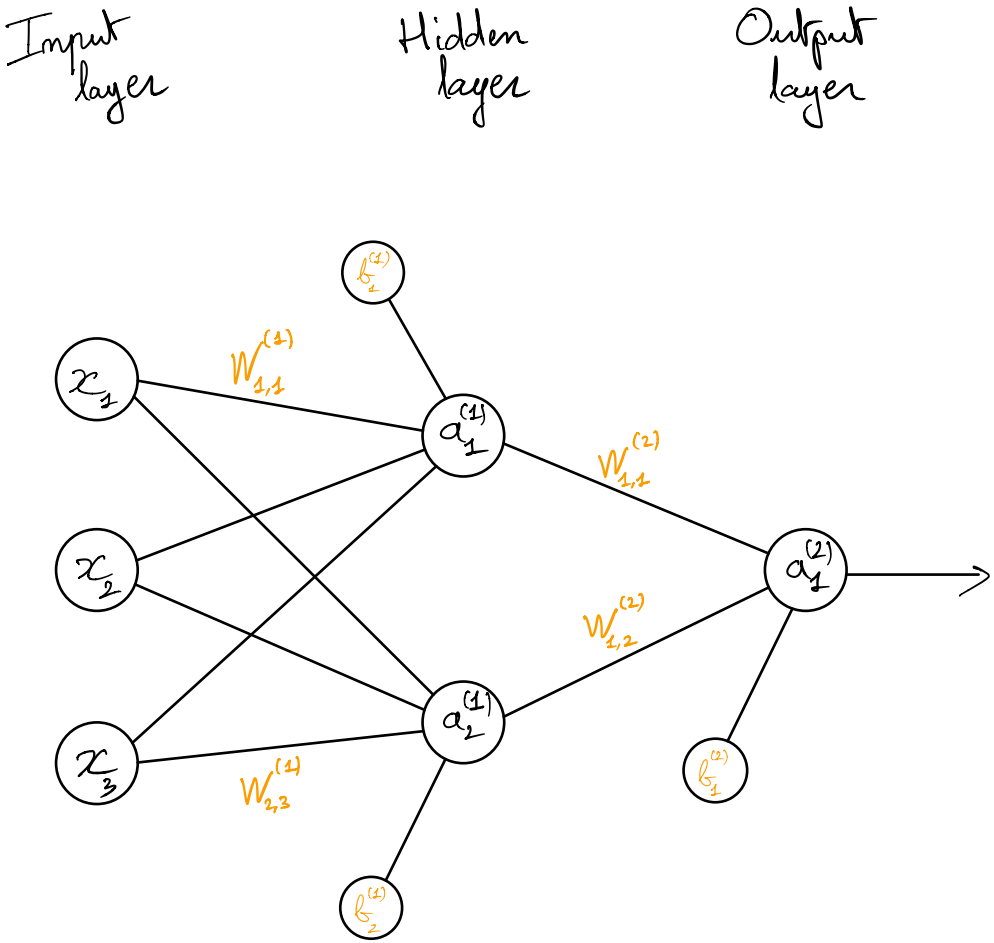
- Weather forecasting

  ["Application of the adaline system to weather forecasting", Hu 1964]

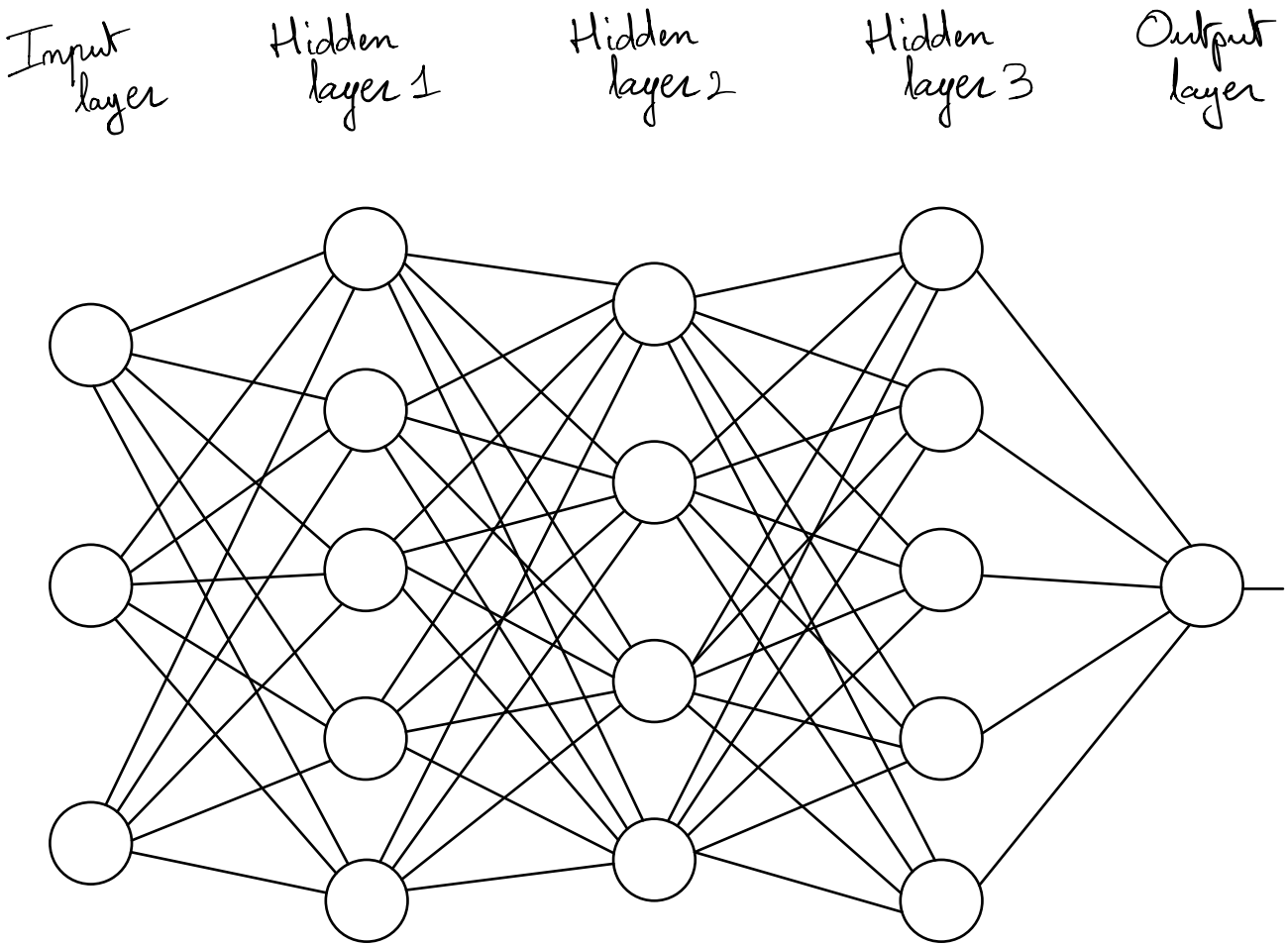
- Adaptive filtering and adaptive signal processing

  ["Adaptive signal processing", Bernard and Samuel 1985]

Neural network with one hidden layer



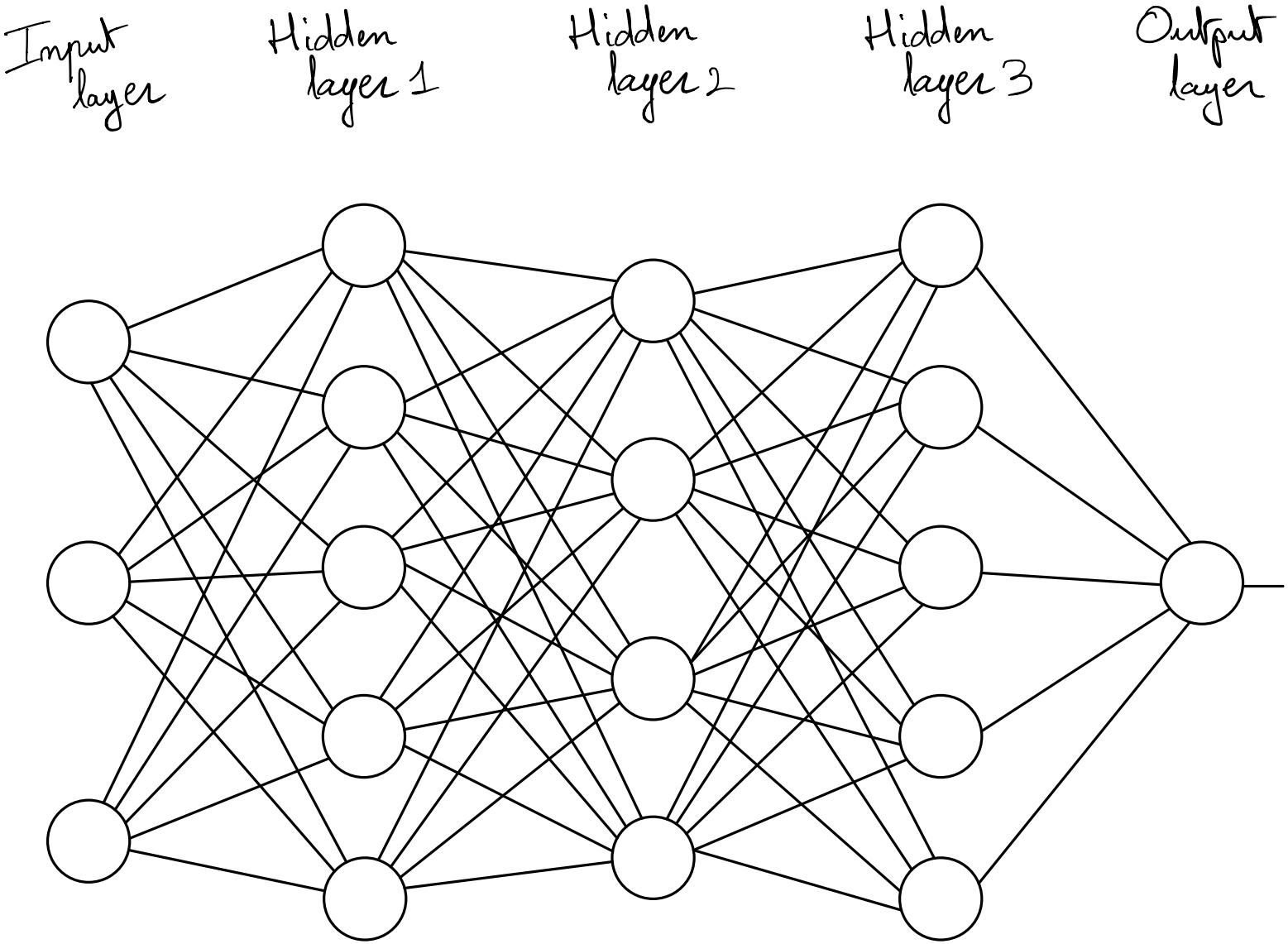
How to find weights and bias?



Perceptron algorithm does not work anymore!

Outline

How to find weights and bias?



Gradient Descent Algorithm

- The prediction of the network is given by $f_\theta(\mathbf{x})$.
- Empirical risk minimization:

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\ell(Y_i, f_\theta(\mathbf{X}_i))}_{\ell_i(\theta)}.$$

Stochastic Gradient descent rule:

While $|\theta_t - \theta_{t-1}| \geq \varepsilon$ do

- Sample $I_t \subset \{1, \ldots, n\}$
- $$\theta_{t+1} = \theta_t - \eta_t \left(\frac{1}{|I_t|} \sum_{i \in I_t} \nabla_\theta \ell_i(\theta_t) \right)$$

Gradient Descent Algorithm

- The prediction of the network is given by $f_\theta(\mathbf{x})$.
- Empirical risk minimization:

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\ell(Y_i, f_\theta(\mathbf{X}_i))}_{\ell_i(\theta)}.$$

Stochastic Gradient descent rule:

While $|\theta_t - \theta_{t-1}| \geq \varepsilon$ do

- Sample $I_t \subset \{1, \ldots, n\}$
- $$\theta_{t+1} = \theta_t - \eta_t \left(\frac{1}{|I_t|} \sum_{i \in I_t} \nabla_\theta \ell_i(\theta_t) \right)$$

How to compute $\nabla_\theta \ell_i$ efficiently?

Backprop Algorithm

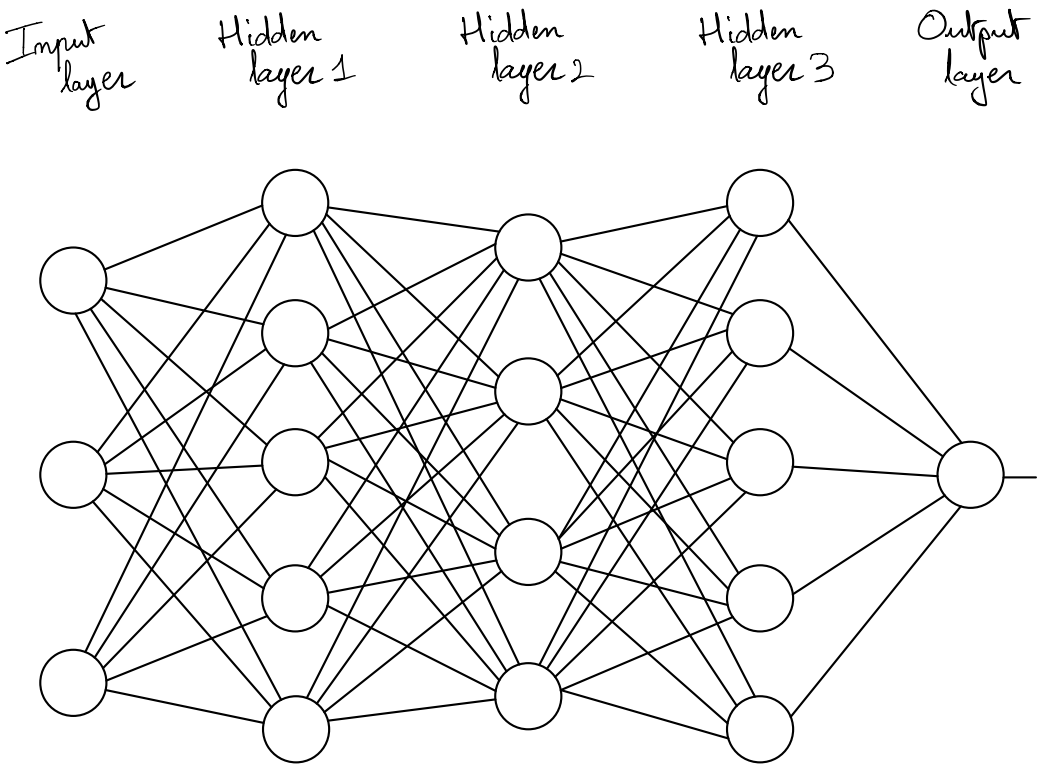
How to compute $\nabla_\theta \ell_i$ efficiently?

A Clever Gradient Descent Implementation

- Popularized by Rumelhart, McClelland, Hinton in 1986.
- Can be traced back to Werbos in 1974.
- Nothing but the use of chain rule derivation with a touch of dynamic programing.

- Key ingredient to make the Neural Networks work!
- Still at the core of Deep Learning algorithm.

Backpropagation idea



Backpropagation equations

Neural network with L layers, with vector output, with quadratic cost

$$C = \frac{1}{2}\|y - a^{(L)}\|^2.$$

By definition,

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial z_j^{(\ell)}}.$$

The four fundamental equations of backpropagation are given by

$$\delta^{(L)} = \nabla_a C \odot \sigma'(z^{(L)}), \tag{1}$$

$$\delta^{(\ell)} = ((w^{(\ell+1)})^T \delta^{(\ell+1)}) \odot \sigma'(z^{(\ell)}) \tag{2}$$

$$\frac{\partial C}{\partial b_j^{(\ell)}} = \delta_j^{(\ell)} \tag{3}$$

$$\frac{\partial C}{\partial w_{j,k}^{(\ell)}} = a_k^{(\ell-1)} \delta_j^{(\ell)}. \tag{4}$$

Backpropagation Algorithm

Let

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial z_j^{(\ell)}},$$

where $z_j^{(\ell)}$ is the entry of the neuron j of the layer ℓ.

Neural network training

(a) Initialize randomly the weights and biases in the network.

(b) For all training samples $(x_i)_{i \in B}$ in the batch B,

1. **Feedforward:** Send all samples of the batch through the network and store the values of activation function and its derivative, for each neuron.
2. **Output loss:** Compute the neural network loss average on all samples of the batch.
3. **Backpropagation (BP):** Compute recursively the vectors $\delta^{(\ell)}$ starting from $\ell = L$ to $\ell = 1$ with BP equations (1) and (2). Compute the gradient with BP equations (3) and (4).
4. **Optimization:** Update the weights and biases using a gradient-based optimization procedure, using the gradient previously computed.

(c) Repeat step (b) until some convergence criterion is reached.

Neural Network terminology

- (Mini) Batch size: number of training examples in one forward/backward pass. The higher the batch size is, the more memory space you'll need.
- Number of iterations: Number of parameters updates during the whole training.
- Epoch: number of iterations required for the network to have seen the whole training set.

For example, for a data set with 12800 training examples, if you set the batch size at 128, it will take 100 iterations to complete 1 epoch.