Introduction to neural networks. Perceptron algorithm.

Kairit Sirts

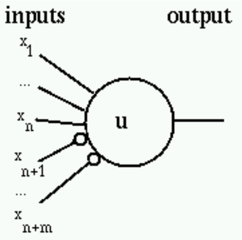
07.03.2014

Biologically inspired learning

- Our brain is made of neurons that send electrical signals to each other.
- Signal emitted by a single neuron depends on the signals of its incoming neurons and the strengths of the connections.
- Learning in brain happens by neurons becoming connected to other neurons ...
- ... and the strengths of the connections becoming adapted over time.

1943 - McCulloch and Pitts

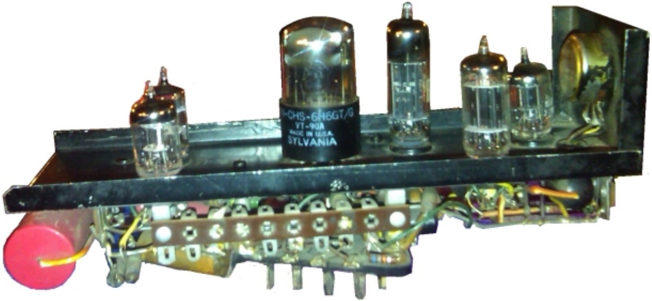
- Proposed a model of artificial neurons:
  - Each neuron is either on or off.
  - Binary inputs and outputs.
  - Inhibitory and exhibitory inputs.
  - Sufficient number of neighboring neurons can influence to switch the neuron on.
  - Any computable function could be computed by some network.



http://osp.mans.edu.eg/rehan/ann/McCulloch-PittsNeuronApplet.htm

1950 - SNARC

- First neural network computer, was built in Harvard (Minsky and Edmonds).
  - 3000 vacuum tubes
  - automatic pilot mechanism from a B-24 bomber
  - 40 neurons
  - Simulated a rat finding its way in a maze.

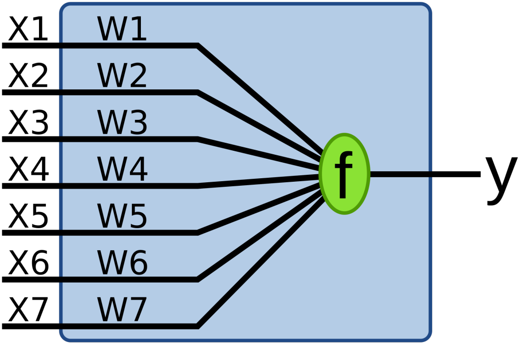


http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minsky-edmonds-american/

1957 - Perceptron

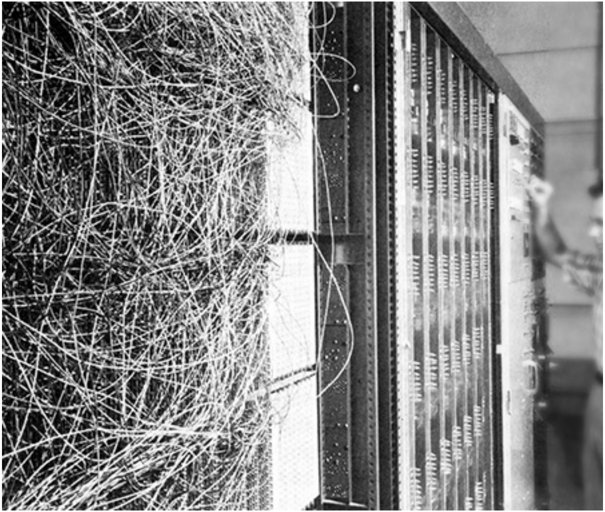
- Developed by Frank Rosenblatt 1957.
- Model of a single neuron network.
- Important innovation was the addition of input weights.
- Learns a linear decision boundary between points of different classes.
- First simulated on an IBM computer.
- On 1960s built on special purpose hardware.

Perceptron model



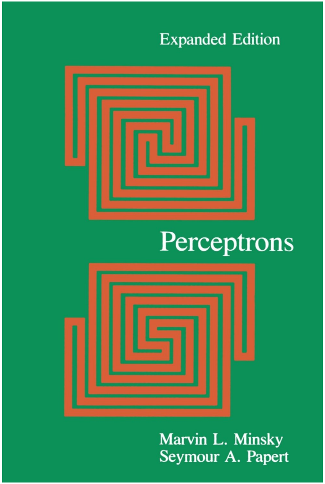
http://en.wikipedia.org/wiki/Perceptron

Mark I Perceptron



http://www.rutherfordjournal.org/article040101.html

"Perceptrons", 1962

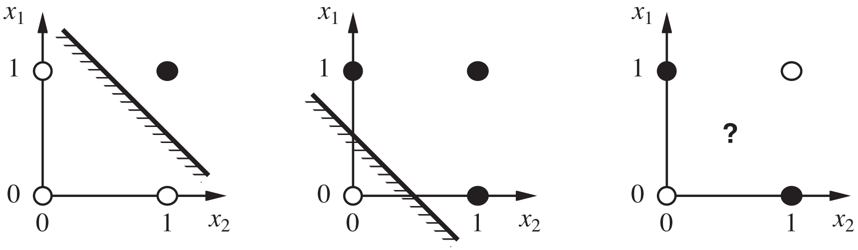


http://www.amazon.co.uk

"Perceptrons", 1962

- Misinterpreted to show that neural networks were fatally flawed.
- It actually only showed the limitations of perceptron model.
  - Perceptron cannot learn parity function.
  - Perceptron cannot learn XOR.
- As a result the money was cut from the whole AI research leading to **AI winter**.
- The situation improved only in the middle of 1980s.

AND,OR and XOR



(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

1980s - new rise

- Neural network training reinvented at least by four different groups.
- Increased computational power provides new opportunities.
- Neural networks become fashionable again.
- But more on this next week.

Model of a single neuron

- Mathematically:
  - input vector $\mathbf{x} \in \mathbb{R}^d$ arrives
  - the neuron has d weights
  - neuron computes the sum:

$$a = \sum_{j=1}^{d} w_j x_j = \mathbf{w}^T \mathbf{x}$$

  - neuron outputs $f(a)$ that is the activation function of the form:

$$f(a) = \begin{cases} +1 & \text{if } a \geq 0 \\ -1 & \text{if } a < 0 \end{cases}$$

  - Often the **bias** or **intercept** term is added:

$$a = \sum_{j=1}^{d} w_j x_j + b = \mathbf{w}^T \mathbf{x} + b$$

Interpretation of weights

- Features with 0 weight are ignored.
- Features with positive weight indicate positive examples.
- Features with negative weight indicate negative examples.
- Bias term will set the threshold different than 0.

Perceptron criterion

- We are seeking a weight vector $\mathbf{w}$ such that:
  - Inputs $\mathbf{x}_{+1}$ with positive label will have $\mathbf{w}^T\mathbf{x}_{+1} + b > 0$;
  - Inputs $\mathbf{x}_{-1}$ with negative label will have $\mathbf{w}^T\mathbf{x}_{-1} + b < 0$;
- When labels are denoted by y then all inputs must satisfy:

$$(\mathbf{w}^T\mathbf{x} + b)y > 0$$

Perceptron criterion

- With correctly classified examples associates zero cost.
- With incorrectly classified examples tries to minimize $-(\mathbf{w}^T\mathbf{x} + b)y$.
- The perceptron criterion is thus given as:

$$E_p(\mathbf{w}) = -\sum_{i=1}^{n} (\mathbf{w}^T\mathbf{x} + b)y$$

- Total cost function is piecewise linear.

Algorithm outlook

- Cycle through training data.
- For each input evaluate the perpectron function.
- If it is correctly classified then the weight vector remains the same.
- If it is incorrectly classified then:
  - in case of positive label add the $\mathbf{x}$ to the weight vector, add one to bias term;
  - in case of negative label subtract the $\mathbf{x}$ from the weight vector, subtract one from bias term.
- This is **online** learning algorithm, only looks at a single item at a time.
- It is **error-driven** algorithm, only changes the weights if there is an error.

Perceptron algorithm

Input: data set $\mathbf{x}_i \in \mathbb{D}$, $y_i \in \{+1, -1\}$, for $i = 1 \dots n$, MaxIter;
$\mathbf{w} \leftarrow \mathbf{0}$;
$b \leftarrow 0$;
MaxIter $\leftarrow 0$
repeat
    for all $\mathbf{x} \in \mathbb{D}$ **do**
        $a = \mathbf{w}^T \mathbf{x} + b$
        if $ya \leq 0$ **then**
            $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}y$
            $b \leftarrow b + y$
        end if
    end for
    MaxIter $\leftarrow$ MaxIter $+1$
until no changes in inner loop or MaxIter reached;

Perceptron update

- The effect of a single update is:

$$
\begin{aligned}
-(\mathbf{w}^{(k+1)T}\mathbf{x} + b^{k+1})y &= -(\mathbf{w}^{(k)T}\mathbf{x} + b^{k})y - (\mathbf{x}y)^T(\mathbf{x}y) - yy \\
&< -(\mathbf{w}^{(k)T}\mathbf{x} + b^{k})y,
\end{aligned}
$$

- because $\|\mathbf{x}y\|^2 > 0$.
- Some previously correctly classified inputs might now be wrong.
- Only the contribution of error of the current input is guaranteed to be reduced.
- The total error is not guaranteed to be reduced at each step.

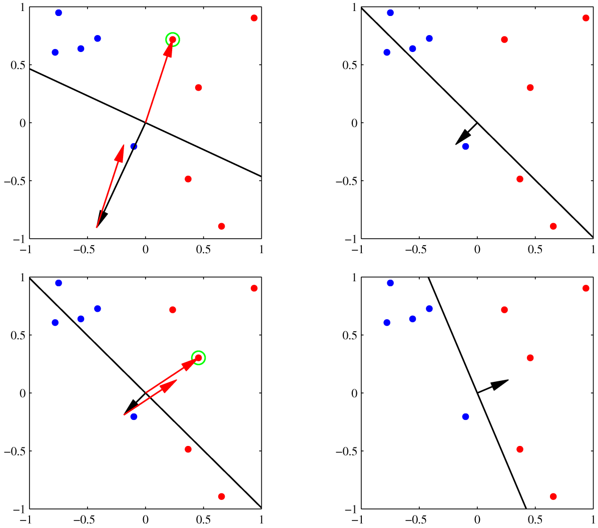
Interpretation of decision boundary

- Decision boundary is formed from a set of points $\mathbf{x}$ for which activation is 0.

$$\mathbb{B} = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} = 0\}$$

- Two vectors have zero dot-product when they are **perpendicular**.
- Thus the desision boundary is a plane perpendicular to the weight vector $\mathbf{w}$.

Perceptron example



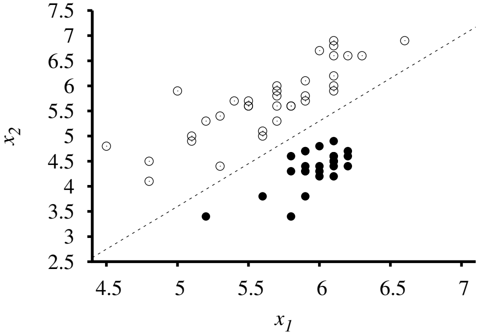
Perceptron converence theorem

Theorem

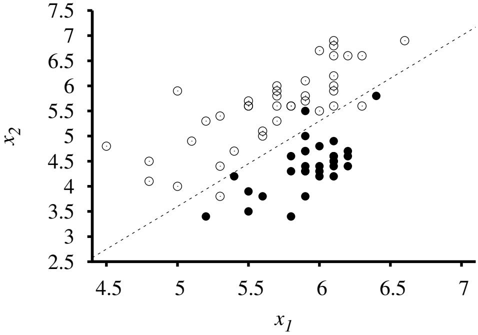
If the data is linearly separable then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.

- The number of steps required might be substantial.
- Before convergence is achieved it is not possible to distuingish between a slowly convergent and nonseparable problem.
- Linearly separable data has many solutions.
- The specific solution found will depend on the initialization of weights and the ordering of data.
- With nonseparable data the algorithm will never converge.

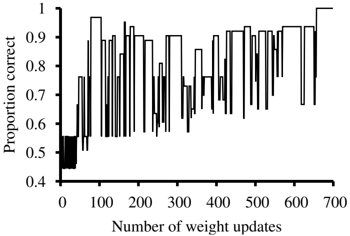
Linearly separable data

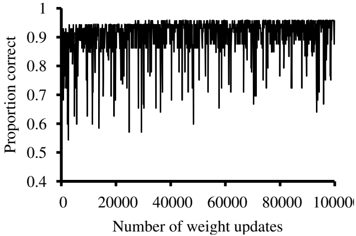


Linearly nonseparable data



Learning curve with separable and nonseparable data





Limitations of perceptron algorithm

- It does not provide probabilistic outputs.
- It does not generalize easily to more than two classes.
- It can only learn linear decision boundaries.
- All these problems will be solved with neural networks.