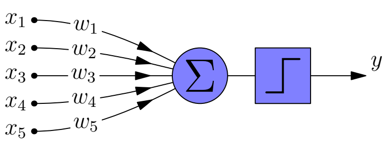
Error driven Learning

1. Thresholded Single-Layer Networks
   - Classification Capabilities
   - Limitations
2. Learning
   - Perceptron Learning
   - Delta Rule

1. Thresholded Single-Layer Networks
   - Classification Capabilities
   - Limitations
2. Learning
   - Perceptron Learning
   - Delta Rule

Thresholded Neurons

TLU — Threshold Logic Unit

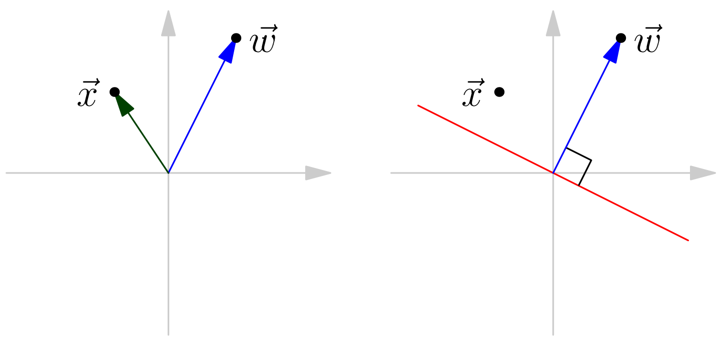


$$y = \begin{cases} 1 & \text{when } \sum_i w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

- Binary output
- Classifies input patterns

Classification Capabilities

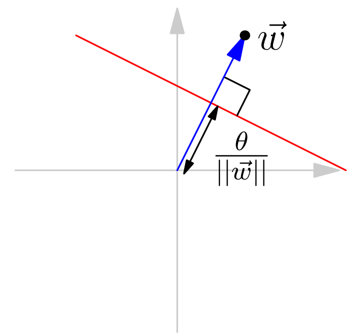
How does the classification work?



Linear Separation

Classification Capabilities

Regulation of the threshold θ



$$y = \begin{cases} 1 & \text{when } \sum_i w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

The separating hyper-plane can be arbitrarily positioned

Classification Capabilities

Important trick: The variable threshold can be substituted by an extra weight from a constant input

$$\sum_i w_i x_i > \theta$$

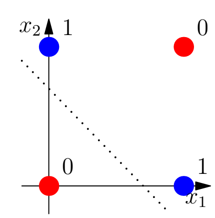
$$w_0 \cdot 1 + \sum_i w_i x_i > 0 \qquad w_0 = -\theta$$

Why? Regulation of the threshold does not have to be treated as a special case

Limitations

Can all groups of patterns be separated?
Classical counter-example: Exclusive OR (XOR)

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \to 0 \qquad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \to 1 \qquad \begin{bmatrix} 1 \\ 0 \end{bmatrix} \to 1 \qquad \begin{bmatrix} 1 \\ 1 \end{bmatrix} \to 0$$



Not linearly separable!

Perceptron Learning

Training of a Thresholded Network: Perceptron Learning
Basic Principle: Weights are changed whenever a pattern is erroneously classified

When the result $= 0$, should be $= 1$

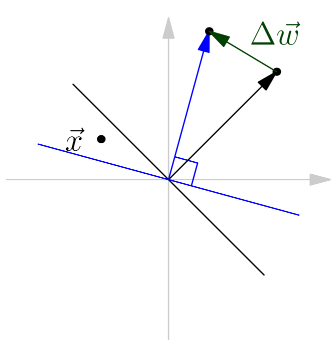
$$\Delta\vec{w} = \eta\vec{x}$$

When the result $= 1$, should be $= 0$

$$\Delta\vec{w} = -\eta\vec{x}$$

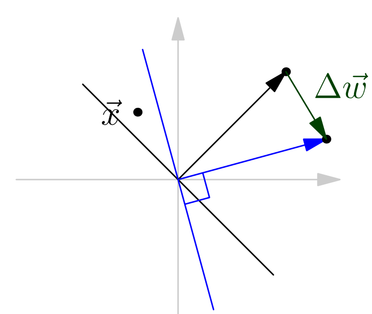
Perceptron Learning

When the result $= 0$, should be $= 1$: $\Delta\vec{w} = \eta\vec{x}$



Perceptron Learning

When the result $= 1$, should be $= 0$: $\Delta\vec{w} = -\eta\vec{x}$



Perceptron Learning

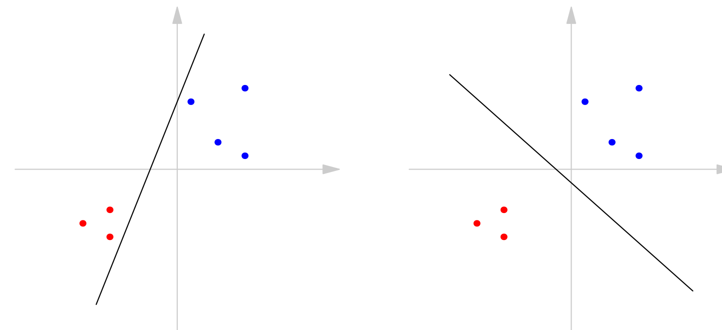
Convergence Theorem

If a solution exists for a finite training dataset then perceptron learning always converges after a finite number of steps

Independent of step size (η)

Perceptron Learning

Problem: Learning terminates unnecessarily early



Bad when patterns are only approximately similar to those used during training

Delta Rule

Delta-rule (Widrow-Hoff rule)
Use symmetric target values $\{-1, 1\}$
Measure the error before thresholding

$$e = t - \vec{w}^T \vec{x}$$

Find the weights which minimize the cost-function

$$\mathcal{E} = \frac{e^2}{2}$$

Delta Rule

Minimize the cost-function

$$\mathcal{E} = \frac{e^2}{2}$$

Simple algorithm: Steepest Decent
Gradient = direction in which the error increases most
Steepest Decent ⇒ Move in the opposite direction
Gradient direction:

$$\frac{\partial \mathcal{E}}{\partial \vec{w}} = e\frac{\partial e}{\partial \vec{w}} = e\frac{\partial (t - \vec{w}^T \vec{x})}{\partial \vec{w}} = -e\vec{x}$$

Delta Rule:

$$\Delta \vec{w} = \eta e \vec{x}$$

Training of Thresholded Single-Layer Networks

- Perceptron Learning

$$\Delta \vec{w} = \eta e \vec{x} \qquad \text{where } e = t - y$$

- Delta Rule

$$\Delta \vec{w} = \eta e \vec{x} \qquad \text{where } e = t - \vec{w}^T \vec{x}$$