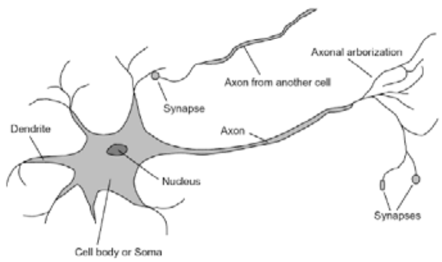
Linear Classification (Part II: Perceptron)

nanopoulos@ismll.de

Outline

- The perceptron model
- Minimization with gradient descent
- Solution for the perceptron model
- Convergence theorem
- Properties and limitations of perceptron

A Model for the Neuron



- Inputs are features
- Each feature has a weight
- Sum is the activation

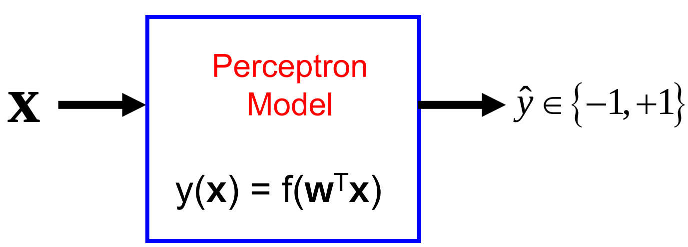
3

The Perceptron model

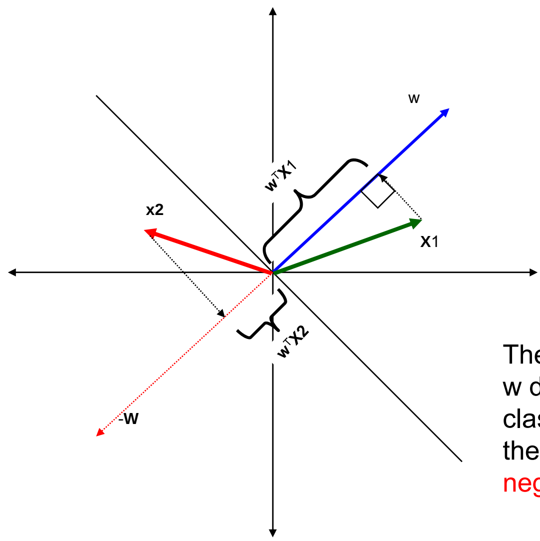
Two class targets: +1 for C_1, -1 for C_2

Training data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

Activation function: $f(a) = \begin{cases} +1, & a \geqslant 0 \\ -1, & a < 0. \end{cases}$ sign function



4

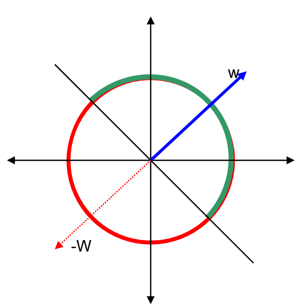


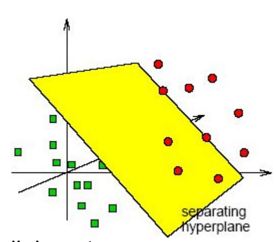
Visually, $\mathbf{w}^T\mathbf{x}$ is the distance you get if you "project **x** onto **w**"

In 3d: line→plane
In 4d: plane→hyperplane
…

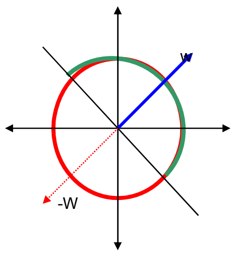
The line perpendicular to w divides the vectors classified as positive from the vectors classified as negative.

$$\hat{y} = \text{sign}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$$





Mediaboost.com



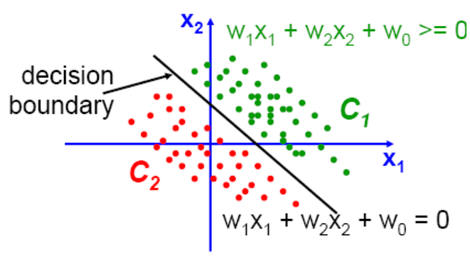
Notice that the <u>separating hyperplane</u> goes through the origin…if we don't want this we can preprocess our examples:

$$\cancel{\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle}$$

$$\mathbf{x} = \langle 1, x_1, x_2, \ldots, x_n \rangle$$

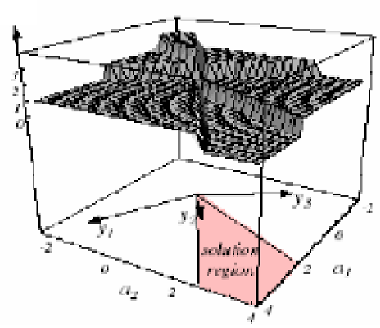
$$\cancel{\hat{y} = \mathrm{sign}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x})}$$

$$\hat{y} = \mathrm{sign}(w_0 1 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x})$$



Training the Perceptron

- Find **w** that minimizes an error function on all training points
- A possible error function is the number of misclassified points
  - Piecewise constant
  - Unsuitable for optimization



The Perceptron criterion

- We seek a vector **w** such that:
  - $\mathbf{w}^T\mathbf{x}_n > 0 \ \forall \ \mathbf{x}_n \in C_1 \quad (t_n = +1)$
  - $\mathbf{w}^T\mathbf{x}_n < 0 \ \forall \ \mathbf{x}_n \in C_2 \quad (t_n = -1)$
- Equivalently:
  - $\mathbf{w}^T\mathbf{x}_n \, t_n > 0 \ \forall \ \mathbf{x}_n$
- For each $\mathbf{x}_n$ associate error equal to:
  - 0, if $\mathbf{x}_n$ is classified correctly
  - $-\mathbf{w}^T\mathbf{x}_n \, t_n$, if $\mathbf{x}_n$ is classified incorrectly

9

Minimization function

$$E_P(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^T \mathbf{x}_n t_n$$

$\mathcal{M}$ denotes the set of all misclassified patterns

- How to minimize this function?
- We now a way: set derivative equal to 0
- Can we apply it here?
- No!
- Do we have other ways?

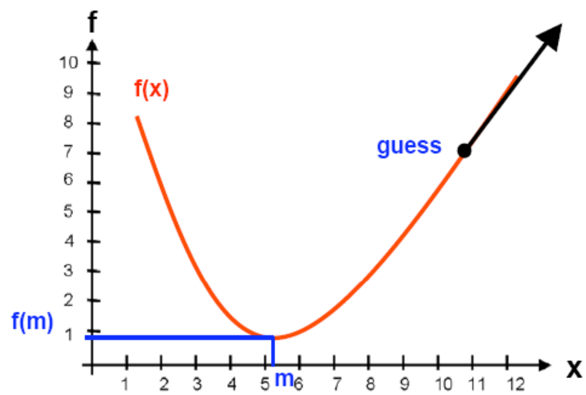
10

Outline

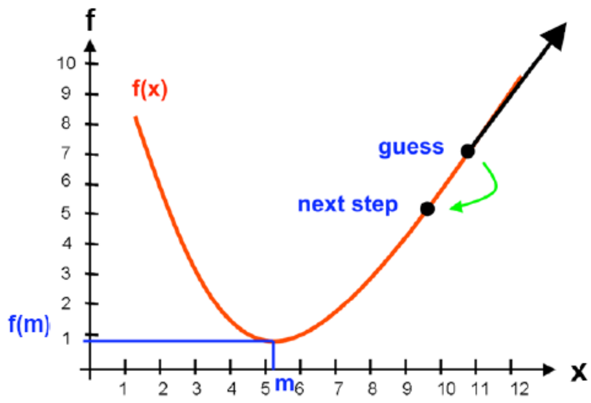
- The perceptron model
- Minimization with gradient descent
- Solution for the perceptron model
- Convergence theorem
- Properties and limitations of perceptron

Gradient descent (single variable)

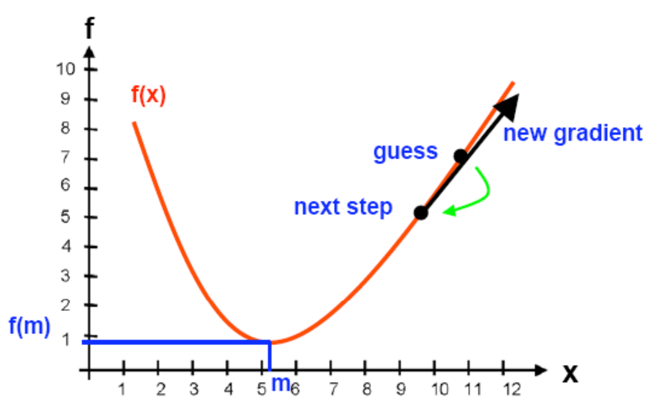
Minimum of a function is found by following the slope of the function



Gradient descent (single variable)



Gradient descent (single variable)



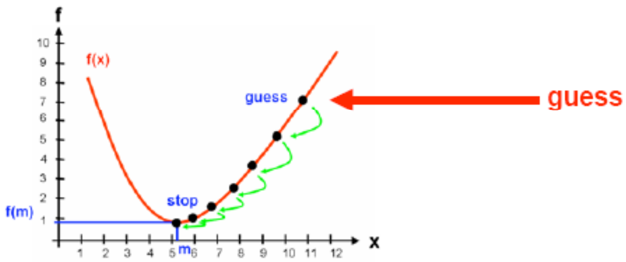
Gradient descent: algorithm

Start with a point (guess)

Repeat

- Determine a descent direction
- Choose a step
- Update

Until stopping criterion is satisfied



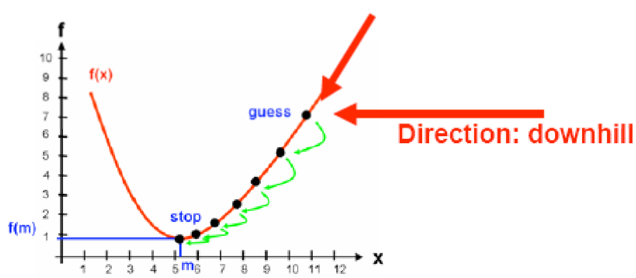
Gradient descent: algorithm

Start with a point (guess)

Repeat

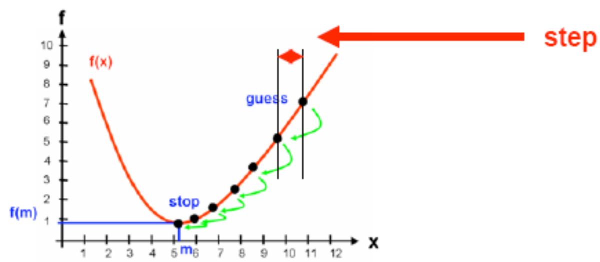
- Determine a descent direction
- Choose a step
- Update

Until stopping criterion is satisfied



Gradient descent: algorithm

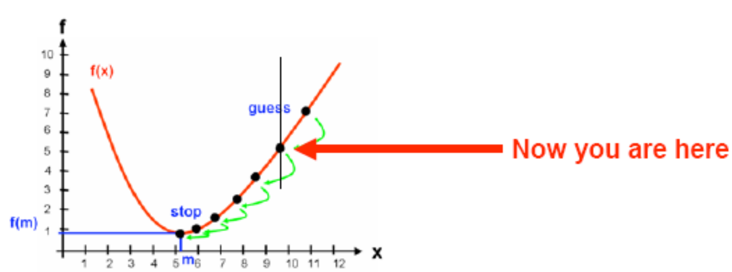
Start with a point (guess)
Repeat
  Determine a descent direction
  Choose a step
  Update
Until stopping criterion is satisfied



17

Gradient descent: algorithm

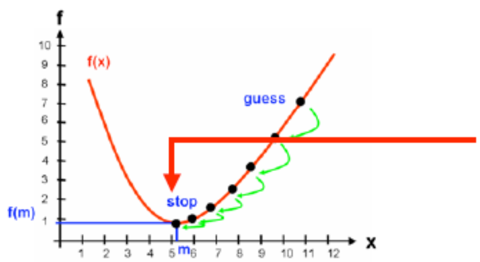
Start with a point (guess)
Repeat
  Determine a descent direction
  Choose a step
  Update
Until stopping criterion is satisfied



18

Gradient descent: algorithm

Start with a point (guess)
Repeat
  Determine a descent direction
  Choose a step
  Update
Until stopping criterion is satisfied



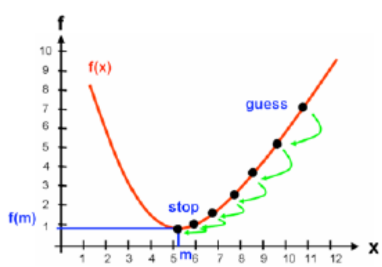
Stop when "close" from minimum

Gradient descent: algorithm

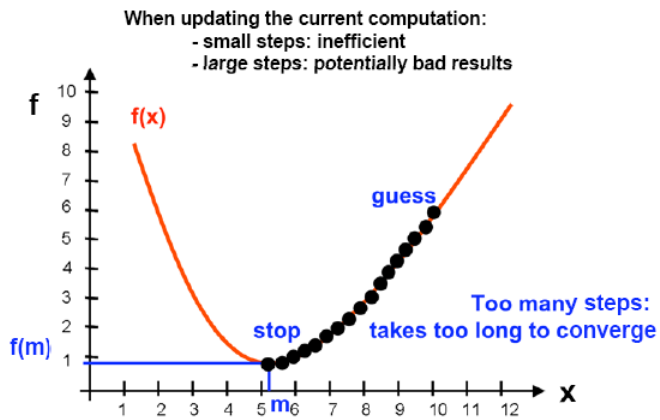
Start with a point (guess)
Repeat
  Determine a descent direction
  Choose a step
  Update
Until stopping criterion is satisfied

guess = x

direction = -f'(x)
step = h > 0
x:=x–hf'(x)
f'(x)~0

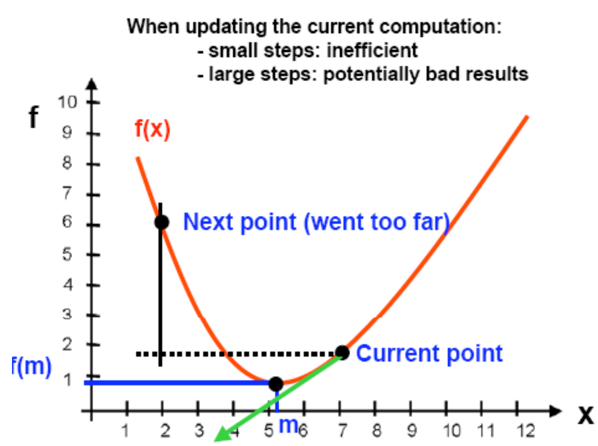


Learning rate



21

Learning rate



22

Gradient operator for high dimensions

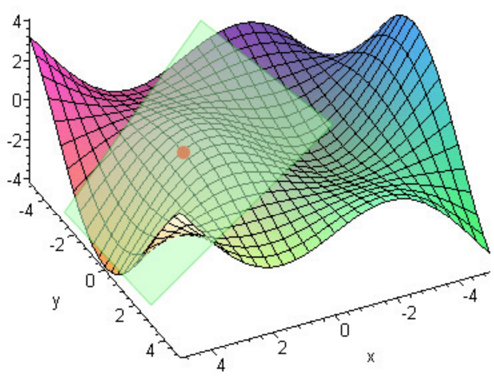
$$f : R^2 \to R \qquad \nabla f(x,y) := \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{pmatrix}$$

This is just a genaralization of the derivative in two dimensions.
This can be generalized to any dimension.

$$f : R^n \to R \qquad \nabla f(x_1,\dots,x_n) := \begin{pmatrix} \frac{\partial f}{\partial x_1}, \dots, & \frac{\partial f}{\partial x_n} \end{pmatrix}$$

The Gradient Properties

The gradient defines (hyper) plane approximating the function infinitesimally



$$\Delta z = \frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y$$

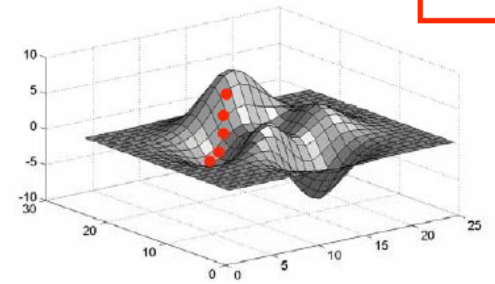
(intuitive: the gradient points to the greatest change direction)

Gradient descent: algorithm for high dim

Start with a point (guess)
Repeat
    Determine a descent direction
    Choose a step
    Update
Until stopping criterion is satisfied

guess = x

direction = -f(x)
step = h > 0
x:=x–h ∇f'(x)
∇f'(x)~0



Stochastic gradient descent

$$E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w})$$

On-line gradient descent, also known as *sequential gradient descent* or *stochastic gradient descent*, makes an update to the weight vector based on one data point at a time, so that

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)})$$

And now, back to Perceptron

Outline

- The perceptron model
- Minimization with gradient descent
- Solution for the perceptron model
- Convergence theorem
- Properties and limitations of perceptron

Minimize error with stochastic gradient descent

$$E_{\mathrm{P}}(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^{\mathrm{T}} \mathbf{x}_n t_n$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_{\mathrm{P}}(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \mathbf{x}_n t_n$$

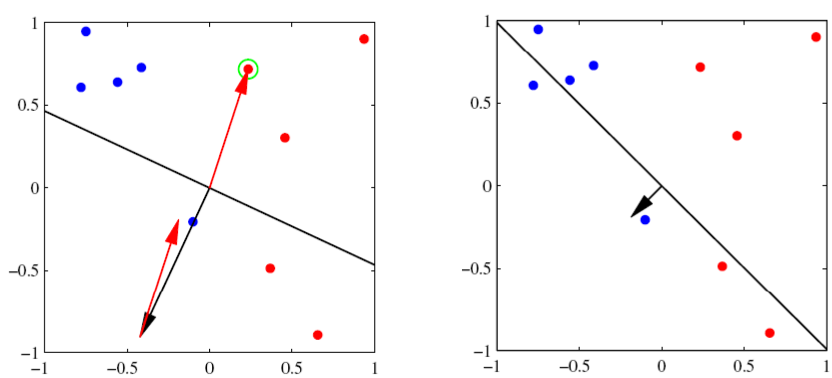
For simplicity, set η = 1 (learning rate)

Intuitive explanation

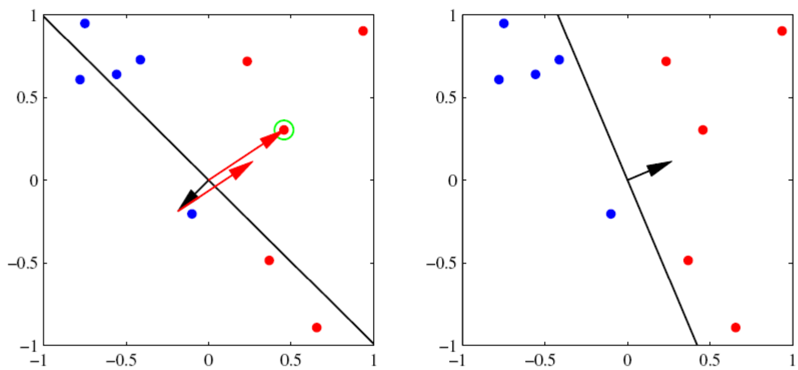
We cycle through the training patterns in turn, and for each pattern $\mathbf{x}_n$:

- if it is correctly classified, then **w** remains unchanged
- if it is incorrectly classified, then
  - for class *C1* we add $\mathbf{x}_n$ onto **w** while
  - for class C2 we subtract $\mathbf{x}_n$ from **w**.

Example (+1, -1)



Example (+1, -1)



Arithmetic example

Consider the 2-dimensional training set $C_1 \cup C_2$,
$C_1 = \{(1,1), (1, -1), (0, -1)\}$ with class label 1
$C_2 = \{(-1,-1), (-1,1), (0,1)\}$ with class label -1

Arithmetic example

Consider the augmented training set $C'_1 \cup C'_2$, with first entry fixed to 1 (to deal with the bias as extra weight):
(1, 1, 1), (1, 1, -1), (1, 0, -1) ,(1,-1, -1), (1,-1, 1), (1,0, 1)

Replace **x** with **-x** for all **x** $\in C_2'$ and use the following update rule:

$$w(n+1) = \begin{cases} w(n) + \eta x(n) \text{ if } w^T(n)x(n) \le 0 \\ w(n) \text{ otherwise} \end{cases}$$

Epoch = the application of the update rule to each example of the training set. Then terminate the execution of the learning algorithm if the weights do not change after one epoch.

Arithmetic example

- the execution of the perceptron learning algorithm for each epoch is illustrated below, with w(1)=(1,0,0), η =1, and transformed inputs (1, 1, 1), (1, 1, -1), (1,0, -1), (-1,1, 1), (-1,1, -1), (-1,0, -1)

Adjusted pattern	Weight applied	w(n) x(n)	Update?	New weight
(1, 1, 1)	(1, 0, 0)	1	No	(1, 0, 0)
(1, 1, -1)	(1, 0, 0)	1	No	(1, 0, 0)
(1,0, -1)	(1, 0, 0)	1	No	(1, 0, 0)
(-1,1, 1)	(1, 0, 0)	-1	Yes	(0, 1, 1)
(-1,1, -1)	(0, 1, 1)	0	Yes	(-1, 2, 0)
(-1,0, -1)	(-1, 2, 0)	1	No	(-1, 2, 0)

End epoch 1

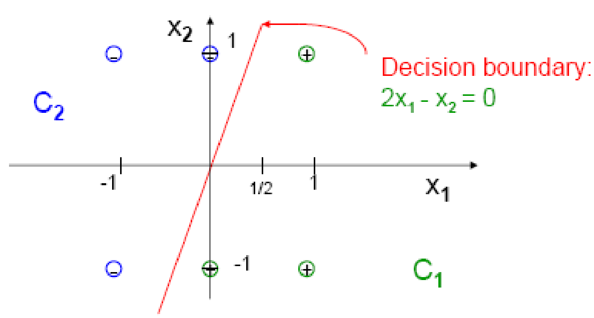
Arithmetic example

Adjusted pattern	Weight applied	w(n) x(n)	Update?	New weight
(1, 1, 1)	(-1, 2, 0)	1	No	(-1, 2, 0)
(1, 1, -1)	(-1, 2, 0)	1	No	(-1, 2, 0)
(1,0, -1)	(-1, 2, 0)	-1	Yes	(0, 2, -1)
(-1, 1, 1)	(0, 2, -1)	1	No	(0, 2, -1)
(-1, 1, -1)	(0, 2, -1)	3	No	(0, 2, -1)
(-1,0, -1)	(0,2, -1)	1	No	(0, 2, -1)

End epoch 2

At epoch 3 no weight changes. (check!) $\Rightarrow$ stop execution of algorithm.
Final weight vect.: (0, 2, -1) $\Rightarrow$ decision hyperplane is $2x_1 - x_2 = 0$.

Arithmetic example: result



Outline

- The perceptron model
- Minimization with gradient descent
- Solution for the perceptron model
- Convergence theorem
- Properties and limitations of perceptron

Convergence theorem

Suppose the classes C_1, C_2 are linearly separable (that is, there exists a hyper-plane that separates them). Then the perceptron algorithm applied to $C_1 \cup C_2$ terminates successfully after a finite number of iterations.

Proof:

Consider the set C containing the inputs of $C_1 \cup C_2$ transformed by replacing x with -x for each x with class label -1.

For simplicity assume $\mathbf{w}(1) = 0$, $\eta = 1$.

Let $\mathbf{x}(1) \ldots \mathbf{x}(k) \in \mathbf{C}$ be the sequence of inputs that have been used after k iterations. Then

$$\left.\begin{aligned} \mathbf{w}(2) &= \mathbf{w}(1) + \mathbf{x}(1) \\ \mathbf{w}(3) &= \mathbf{w}(2) + \mathbf{x}(2) \\ &\vdots \\ \mathbf{w}(k+1) &= \mathbf{w}(k) + \mathbf{x}(k) \end{aligned}\right\} \Rightarrow \mathbf{w}(k+1) = \mathbf{x}(1) + \ldots + \mathbf{x}(k)$$

Convergence theorem

Since C_1 and C_2 are linearly separable then there exists $\mathbf{w}_*$ such that $\mathbf{w}_*^T \mathbf{x} > 0, \quad \forall\, \mathbf{x} \in C$.

Let $\alpha = \min \mathbf{w}_*^T \mathbf{x}$

Then $\mathbf{w}_*^T \mathbf{w}(k+1) = \mathbf{w}_*^T \mathbf{x}(1) + \ldots + \mathbf{w}_*^T \mathbf{x}(k) \geq k\alpha$

By the Cauchy-Schwarz inequality we get:

$$||\mathbf{w}_*||^2 \; ||\mathbf{w}(k+1)||^2 \geq [\mathbf{w}_*^T \mathbf{w}(k+1)]^2$$

$$||\mathbf{w}(k+1)||^2 \geq \frac{k^2 \alpha^2}{||\mathbf{w}_*||^2} \qquad \textbf{(A)}$$

39

Convergence theorem

- Now we consider another route:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{x}(k)$$

$$||\,\mathbf{w}(k+1)||^2 = ||\,\mathbf{w}(k)||^2 + ||\mathbf{x}(k)||^2 + \underbrace{2\,\mathbf{w}^T(k)\mathbf{x}(k)}_{\leq 0 \text{ because } \mathbf{x}(k) \text{ is misclassified}}$$

euclidean norm

$$\Rightarrow ||\mathbf{w}(k+1)||^2 \leq ||\mathbf{w}(k)||^2 + ||\mathbf{x}(k)||^2$$

=0

$$||\mathbf{w}(2)||^2 \leq ||\mathbf{w}(1)||^2 + ||\mathbf{x}(1)||^2$$

$$||\mathbf{w}(3)||^2 \leq ||\mathbf{w}(2)||^2 + ||\mathbf{x}(2)||^2$$

$$\vdots$$

$$\Rightarrow ||\mathbf{w}(k+1)||^2 \leq \sum_{i=1}^{k} ||\mathrm{x}(i)||^2$$

40

Convergence theorem

- Let $\beta = \max ||\mathbf{x}(n)||^2 \quad \mathbf{x}(n) \in C$
- $||\mathbf{w}(k+1)||^2 \leq k\,\beta$ **(B)**
- For sufficiently large values of k:
  (B) becomes in conflict with **(A).**
  Then k cannot be greater than k_{max} such that **(A)** and **(B)** are both satisfied with the equality sign.

$$\frac{k_{max}^2 \alpha^2}{||w_*||^2} = k_{max}\beta \Rightarrow k_{max} = \frac{||w_*||^2}{\alpha^2}\beta$$

- The algorithm terminates successfully in at most $\frac{\beta\, ||\mathbf{w}_*||^2}{\alpha^2}$ iterations, i.e.

$$\lim_{k\to\infty} w(k) = w(k_{\max}) \quad \text{and} \quad \lim_{k\to\infty} w(k+1) - w(k) = 0$$

41

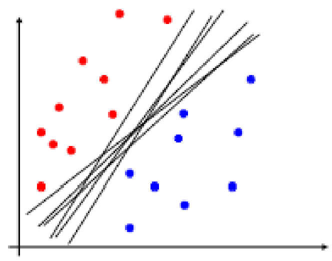
Outline

- The perceptron model
- Minimization with gradient decent
- Solution for the perceptron model
- Convergence theorem
- Properties and limitations of perceptron

42

Many solutions

- Which of these linear separators is optimal?



Nonlinear cases

- The perceptron can only model linearly separable classes, like (those described by) the following Boolean functions:
- AND
- OR
- It cannot model the XOR!