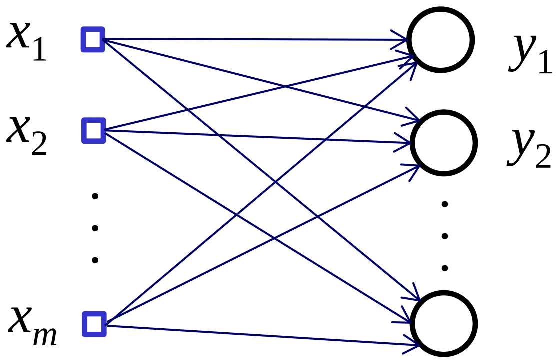
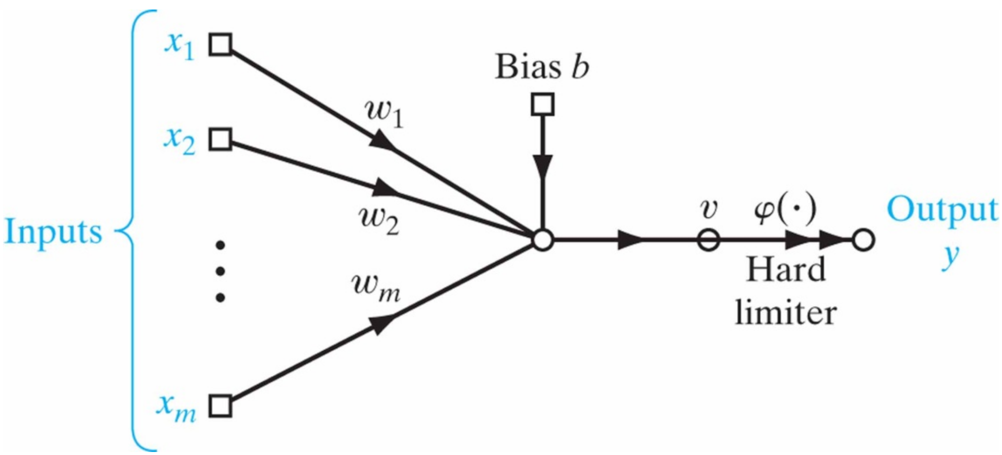
CSE 5526: Introduction to Neural Networks

Perceptrons

Perceptrons

- Architecture: one-layer feedforward net
  - Without loss of generality, consider a single-neuron perceptron





Definition

$$y = \varphi(v)$$

$$v = \sum_{i=1}^{m} w_i x_i + b$$

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Hence a McCulloch-Pitts neuron, but with real-valued inputs

Pattern recognition

- With a bipolar output, the perceptron performs a 2-class classification problem
  - E.g, apples vs. oranges
- How do we learn to perform classification?
- The perceptron is given pairs of input x_p and desired output d_p.
- How can we find w so $y_p = \varphi\left(x_p^T w\right) = d_p \;\; \forall p$?

But first: decision boundary

- Can we visualize the decision the perceptron would make in classifying every potential point?
- Yes, it is called the discriminant function

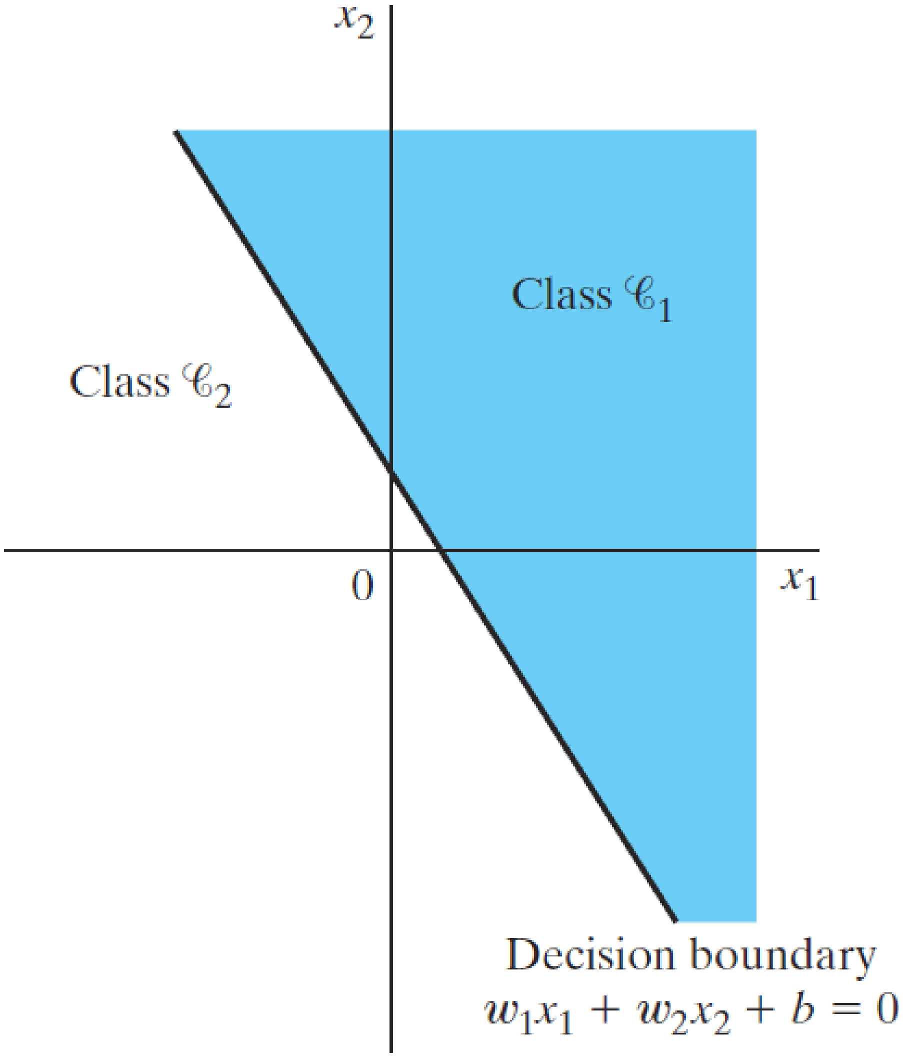
$$g(x) = x^T w = \sum_{i=0}^{m} w_i x_i$$

- What is the boundary between the two classes like?

$$g(x) = x^T w = 0$$

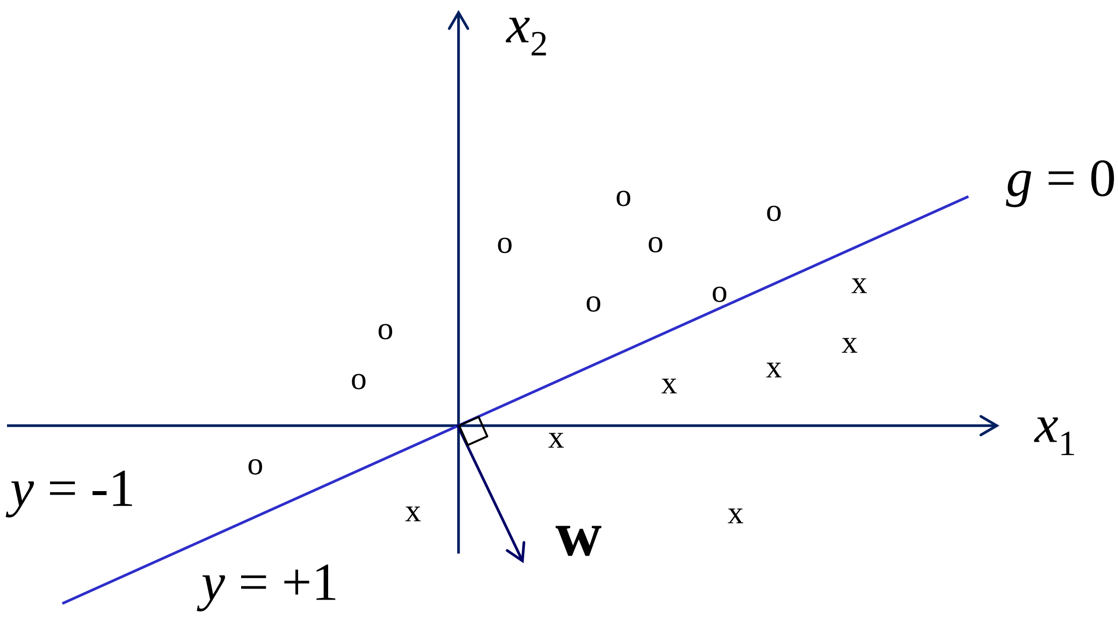
- This is a linear function of *x*

Decision boundary example



Decision boundary (cont.)

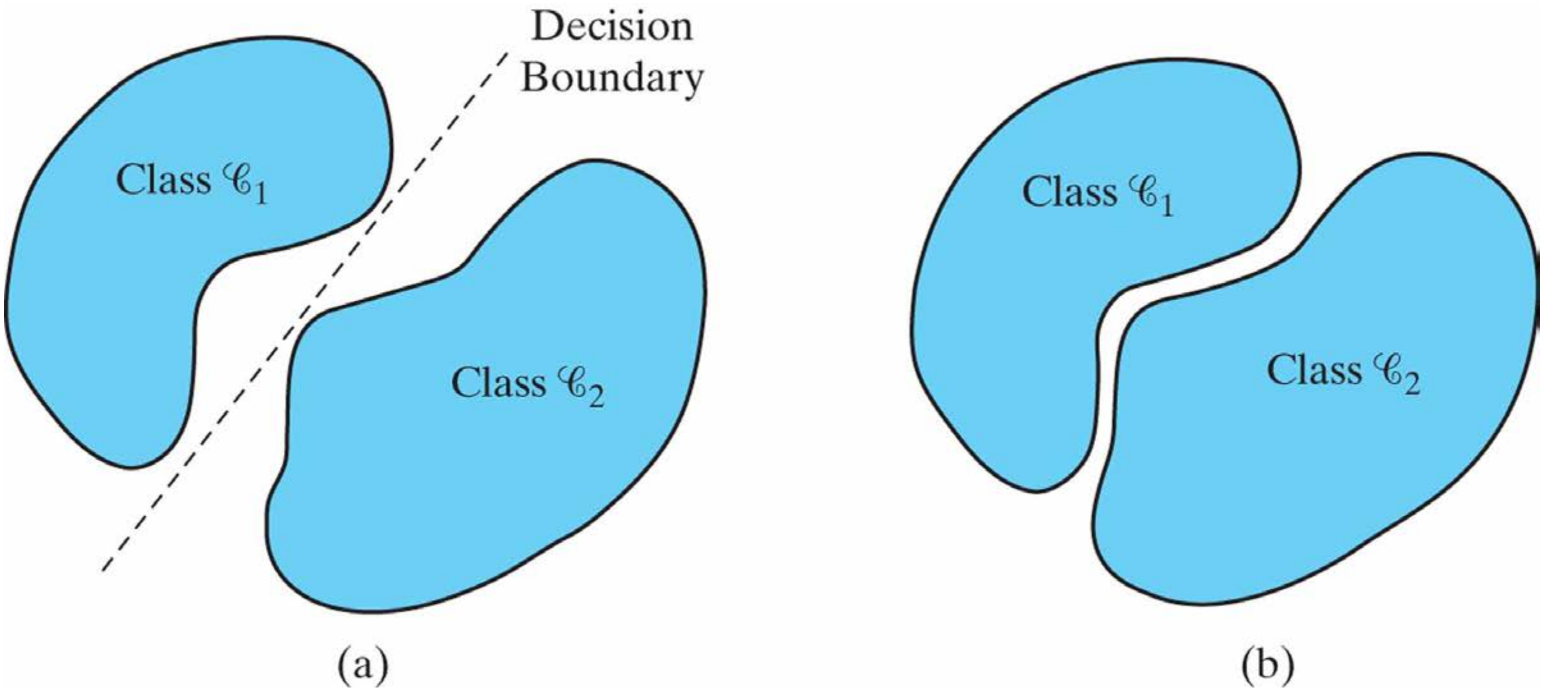
- For an *m*-dimensional input space, the decision boundary is an (*m* – 1)-dimensional hyperplane perpendicular to w. The hyperplane separates the input space into two halves, with one half having *y* = 1, and the other half having *y* = -1
  - When *b* = 0, the hyperplane goes through the origin



Linear separability

- For a set of input patterns x_p, if there exists at least one w that separates $d = 1$ patterns from $d = -1$ patterns, then the classification problem is linearly separable
  - In other words, there exists a linear discriminant function that produces no classification error
  - Examples: AND, OR, XOR (see blackboard)
- A very important concept

Linear separability: a more general illustration



Perceptron definition again

$$y = \varphi(v)$$

$$v = \sum_{i=1}^{m} w_i x_i + b$$

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Hence a McCulloch-Pitts neuron, but with real-valued inputs

Perceptron learning rule

- Learn parameters *w* from examples $(x_{p,}\ d_p)$
- In an online fashion, i.e., one point at a time
- Adjust weights as necessary, i.e. when incorrect
- Adjust weights to be more like d=1 points and more like negative d=-1 points

Biological analogy

- Strengthen an active synapse if the postsynaptic neuron fails to fire when it should have fired; weaken an active synapse if the neuron fires when it should not have fired
  - Formulated by Rosenblatt based on biological intuition

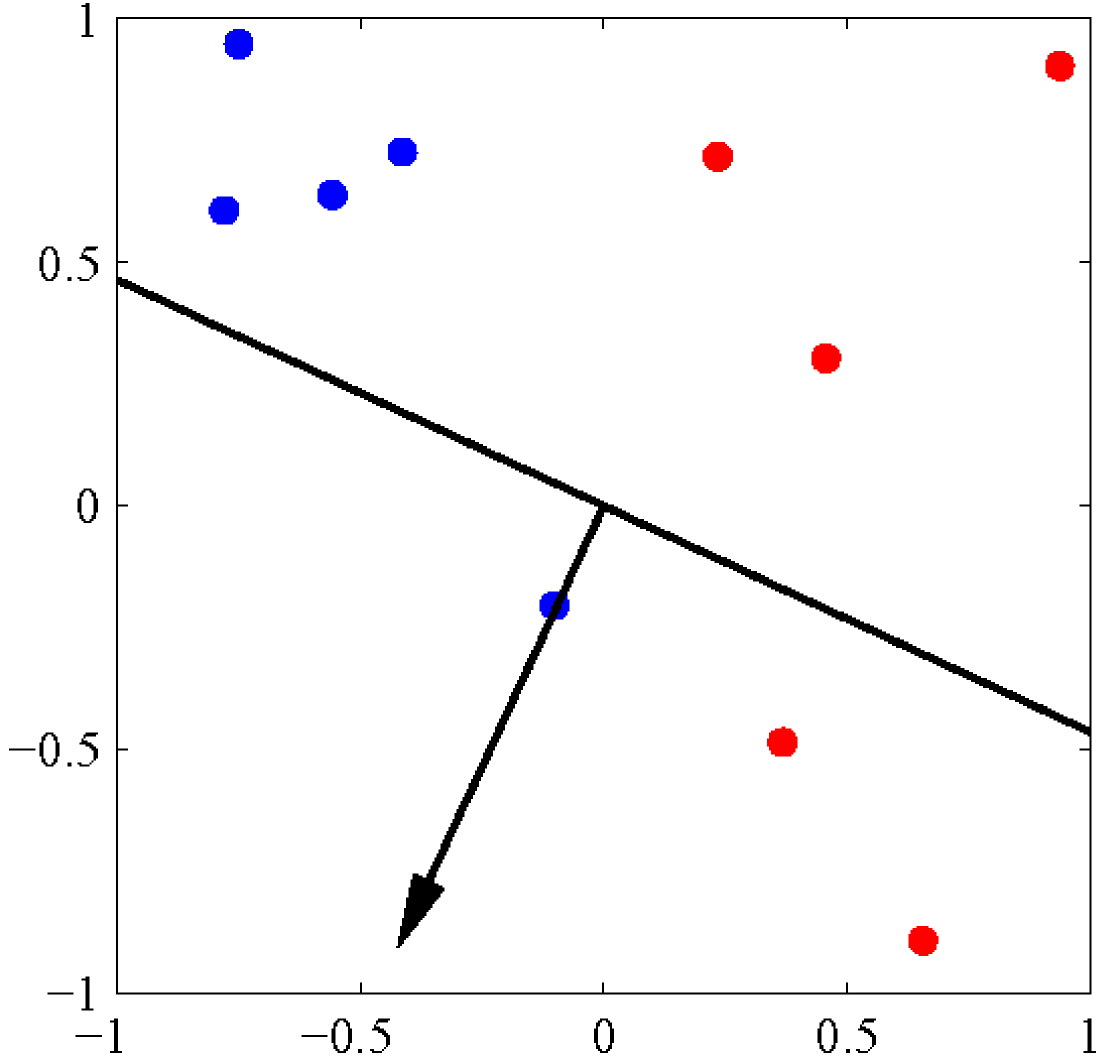
Quantitatively

$$w(n+1) = w(n) + \Delta w(n)$$

$$= w(n) + \eta[d(n) - y(n)]x(n)$$

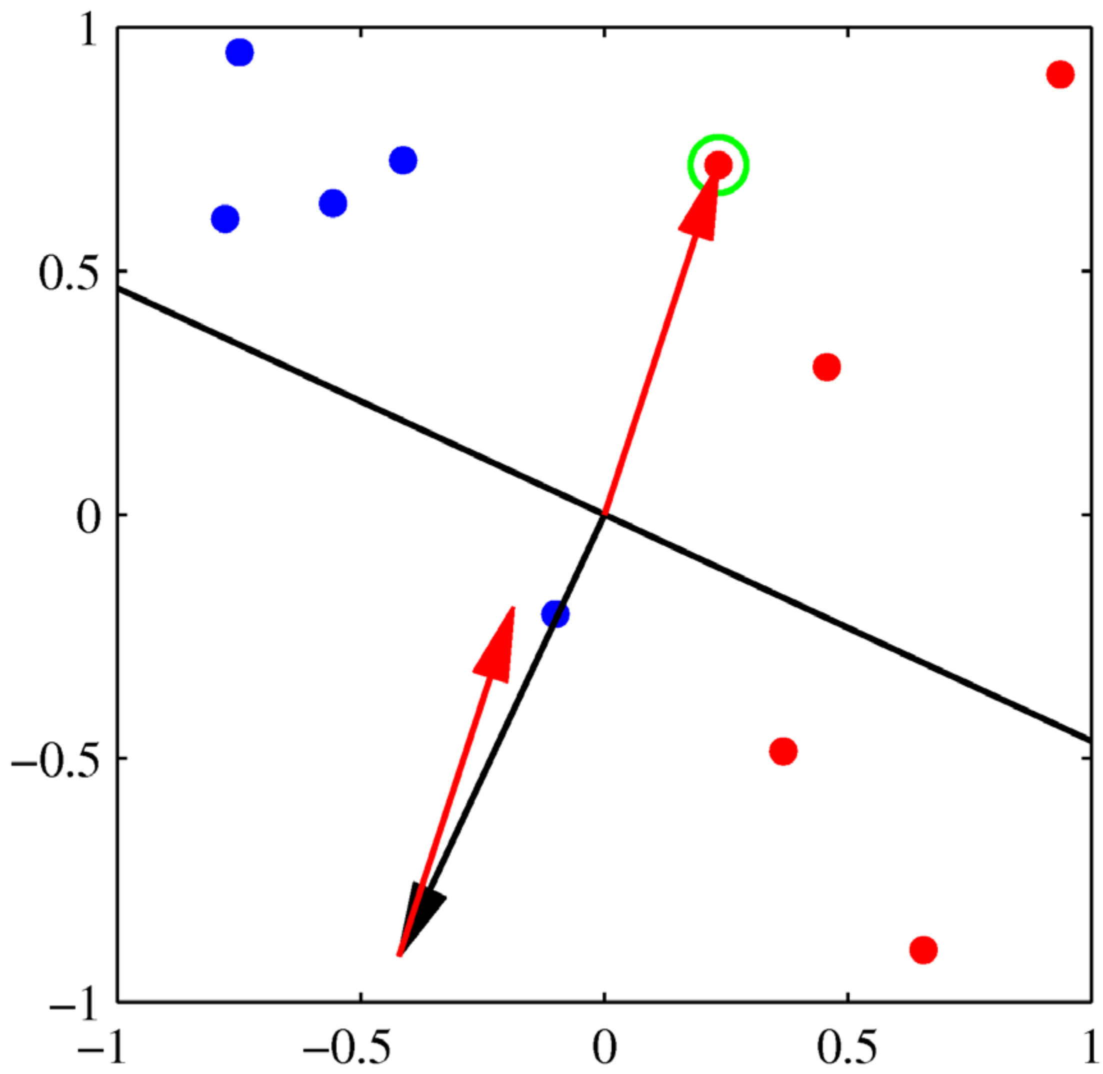
- *n*: iteration number, iterating over points in turn
- η: step size or learning rate, = 1 WLOG
- Only updates *w* when y(n) is incorrect

Geometric interpretation



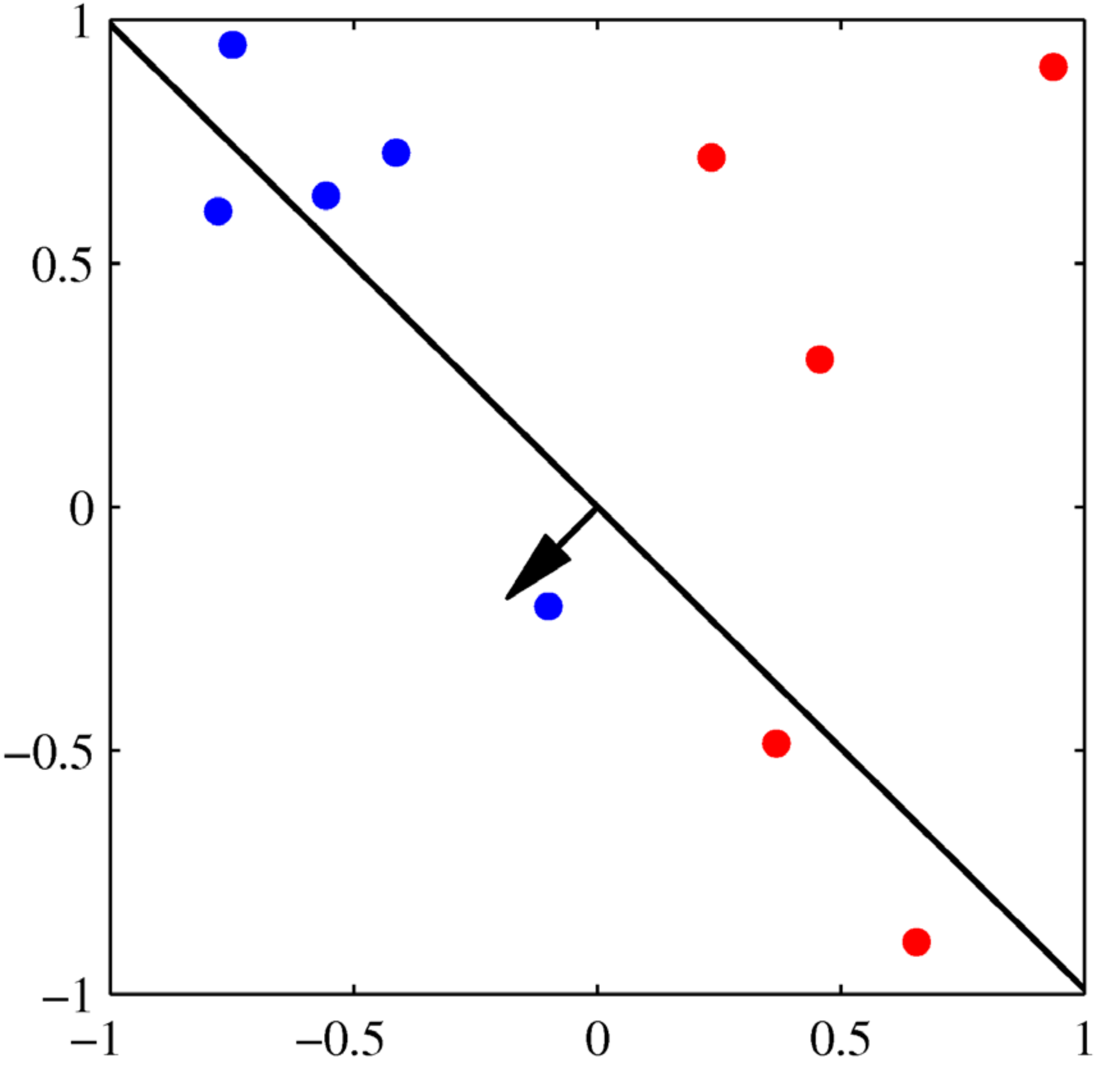
From Bishop (2006)

Geometric interpretation



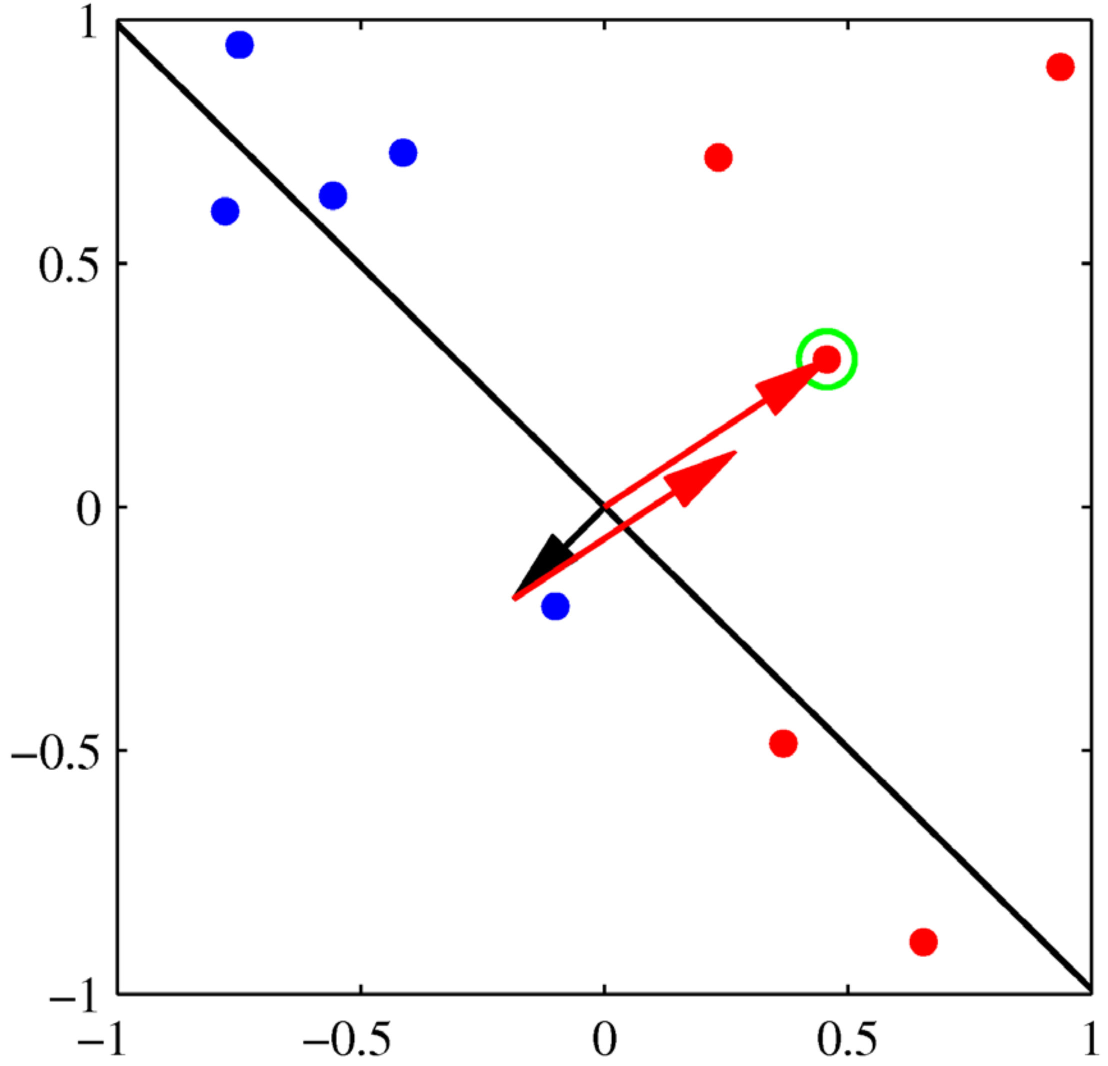
From Bishop (2006)

Geometric interpretation



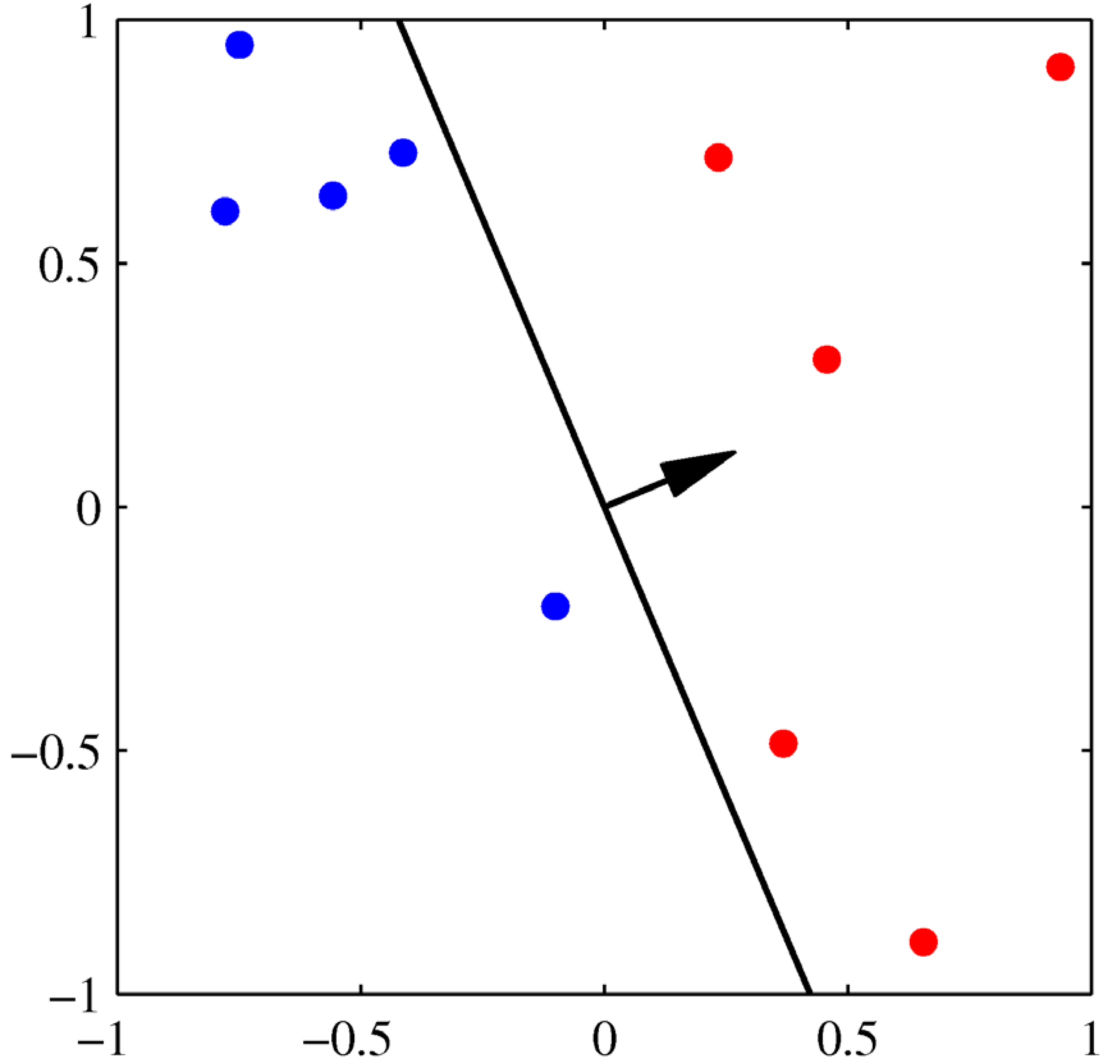
From Bishop (2006)

Geometric interpretation



From Bishop (2006)

Geometric interpretation



From Bishop (2006)

Geometric interpretation

- Each weight update moves w closer to $d = 1$ patterns, or away from $d = -1$ patterns.
- Final weight vector in example solves the classification problem
- Is that true in all cases?

Summary of perceptron learning algorithm

- Definition
  - $w(n)$: (m+1)-by-1 weight vector (including bias) at step n
- Inputs
  - $x(n)$: n^{th} (m+1)-by-1 input vector with first element = 1
  - $d(n)$: n^{th} desired response
- Initialization: set w(0) = 0
- Repeat until no points are mis-classified
  - Compute response: $y(n) = sgn\{w(n)^T x(n)\}$
  - Update: $w(n+1) = w(n) + [d(n) - y(n)]x(n)$

Perceptron convergence theorem

- Theorem:
  - Assume that there exists some unit vector w_0 and some α such that $d(n) w_0^T x(n) \geq \alpha$
    - i.e. the data are linearly separable
  - Assume also that there exists some R such that $\|x(n)\| = \sqrt{x(n)^T x(n)} \leq R \quad \forall n$
    - i.e. the data lie within a sphere of radius R
  - Then the perceptron algorithm makes at most $\frac{R^2}{\alpha^2}$ errors
- Exposition based on Collins (2012)

Perceptron convergence proof outline

- Define w_k as the parameter vector when the algorithm makes its k^{th} error (note $w_1 = 0$)
- First show $k\alpha \leq \|w_{k+1}\|$ by induction
- Second show $\|w_{k+1}\|^2 \leq kR^2$ by induction
- Then it follows that $k \leq \frac{R^2}{\alpha^2}$
  - I.e., the perceptron makes a finite number of errors

First show $k\alpha \leq \|w_{k+1}\|$ by induction

- Assume that the k^{th} error is made on example n
- Because of the perceptron update rule,
  $$\begin{aligned} w_{k+1}^T w_0 &= (w_k + d(n)x(n))^T w_0 \\ &= w_k^T w_0 + d(n)x(n)^T w_0 \\ &\geq w_k^T w_0 + \alpha \end{aligned}$$
- Because, by assumption, $d(n)x(n)^T w_0 \geq \alpha$
- Then, by induction on k, $w_{k+1}^T w_0 \geq k\alpha$
- In addition, $\|w_{k+1}\| \cdot \|w_0\| \geq w_{k+1}^T w_0$ by Cauchy-Schwartz, with $\|w_0\| = 1$
- Thus, $\|w_{k+1}\| \geq w_{k+1}^T w_0 \geq k\alpha$

Second show $\|w_{k+1}\|^2 \le kR^2$ by induction

- Because of the perceptron update rule
$$\|w_{k+1}\|^2 = \|w_k + d(n)x(n)\|^2$$
$$\|w_{k+1}\|^2 = \|w_k\|^2 + d^2(n)\|x(n)\|^2$$
$$+2d(n)x(n)^T w_k$$
- By definition, $d^2(n) = 1$
- By assumption, $\|x(n)\|^2 \le R^2$
- Because the nth point was misclassified
$2d(n)x(n)^T w_k \le 0$
- Thus, $\|w_{k+1}\|^2 \le \|w_k\|^2 + R^2$
- And, by induction on k, $\|w_{k+1}\|^2 \le kR^2$

Then it follows that $k \leq R^2/\alpha^2$

- We have shown
  $k\alpha \leq \|w_{k+1}\|$ and $\|w_{k+1}\|^2 \leq kR^2$
- So, $k^2\alpha^2 \leq \|w_{k+1}\|^2 \leq kR^2$
- Then it follows that $k \leq \frac{R^2}{\alpha^2}$
- Thus the perceptron learning algorithm makes a bounded number of mistakes, i.e., converges

Perceptron learning remarks

- If the data are not linearly separable
  - Algorithm will iterate forever
- Scaling w does not affect the perceptron’s decision
  - So the learning rate, η, does not affect the perceptron’s decision either, and can be set to 1
- The solution weight vector, *w*, is not unique

Generalization

- Performance of a learning machine on test patterns not used during training
- Perceptrons generalize by deriving a decision boundary in the input space. Selection of training patterns is thus important for generalization