MACHINE LEARNING

Vasant Honavar
Artificial Intelligence Research Laboratory
Department of Computer Science
Bioinformatics and Computational Biology Program
Center for Computational Intelligence, Learning, & Discovery
Iowa State University
honavar@cs.iastate.edu
www.cs.iastate.edu/~honavar/
www.cild.iastate.edu/

Neural Networks

- Decision trees are good at modeling nonlinear interactions among a small subset of attributes
- Sometimes we are interested in linear interactions among all attributes
- Simple neural networks are good at modeling such interactions
- The resulting models have close connections with naïve Bayes.
Learning Threshold Functions

- Outline
  - Background
  - Threshold logic functions
  - Connection to logic
  - Connection to geometry
  - Learning threshold functions – perceptron algorithm and its variants
  - Perceptron convergence theorem

Background – Neural computation

- 1900 – Birth of neuroscience – Ramon Cajal et al.
- 1913 – Behaviorist or stimulus response psychology
- 1930-50: Theory of Computation, Church-Turing Thesis
- 1943: McCulloch & Pitts “A logical calculus of neuronal activity”
- 1949: Hebb – Organization of Behavior
- 1960-65: Perceptron model developed by Rosenblatt
Background – Neural computation

- 1969: Minsky and Papert criticize Perceptron
- 1969: Chomsky argues for universal innate grammar
- 1970: Rise of cognitive psychology and knowledge-based AI
- 1975: Learning algorithms for multi-layer neural networks
- 1985: Resurgence of neural networks and machine learning
- 1988: Birth of computational neuroscience
- 1990: Successful applications (stock market, OCR, robotics)
- 1990-2000 New synthesis of behaviorist and cognitive or representational approaches in AI and psychology
- 2000-2006 Synthesis of logical and probabilistic approaches to representation and learning

Background – Brains and Computers

- Brain consists of $10^{11}$ neurons, each of which is connected to $10^4$ neighbors
- Each neuron is slow (1 millisecond to respond to a stimulus) but the brain is astonishingly fast at perceptual tasks (e.g. face recognition)
- Brain processes and learns from multiple sources of sensory information (visual, tactile, auditory...)
- Brain is massively parallel, shallowly serial, modular and roughly hierarchical with recurrent and lateral connectivity within and between modules
- If cognition is -- or at least can be modeled by -- computation, it is natural to ask how and what brains compute
Brain and information processing

- Primary somato-sensory cortex
- Motor association cortex
- Primary motor cortex
- Sensory association area
- Auditory cortex
- Speech comprehension
- Visual association area
- Primary visual cortex
- Auditory association area
- Prefrontal cortex

Neural Networks

Ramon Cajal, 1900

Copyright Vasant Honavar, 2010.
Neurons and Computation

McCulloch-Pitts computational model of a neuron

\[ y = 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \]
\[ y = -1 \text{ otherwise} \]
Threshold neuron – Connection with Geometry

Decision boundary

\[ w_1x_1 + w_2x_2 + w_0 > 0 \]

\[ w_1x_1 + w_2x_2 + w_0 < 0 \]

\[ w_1x_1 + w_2x_2 + w_0 = 0 \]

\[ \sum_{i=1}^{n} w_i x_i + w_0 = 0 \]

Describes a hyperplane which divides the instance space \( \mathbb{R}^n \) into two half-spaces.

\[ \chi_+ = \left\{ x_p \in \mathbb{R}^n \mid w \cdot x_p + w_0 > 0 \right\} \]

\[ \chi_- = \left\{ x_p \in \mathbb{R}^n \mid w \cdot x_p + w_0 < 0 \right\} \]

McCulloch-Pitts Neuron or Threshold Neuron

\[ y = \text{sign} \left( W \cdot X + w_0 \right) \]

\[ = \text{sign} \left( \sum_{i=0}^{n} w_i x_i \right) \]

\[ = \text{sign} \left( W^T X + w_0 \right) \]

\[ \text{sign}(v) = 1 \text{ if } v > 0 \]

\[ = 0 \text{ otherwise} \]

Copyright Vasant Honavar, 2010.
Threshold neuron – Connection with Geometry

The (n-1)-dimensional hyperplane partitions the n-dimensional input space into two half spaces.

Instance space $\mathcal{X}^n$

Hypothesis space is the set of (n-1)-dimensional hyperplanes defined in the n-dimensional instance space

A hypothesis is defined by

$$\sum_{i=0}^{n} w_i x_i = 0$$

- Orientation of the hyperplane is governed by $w (w_1 \ldots w_n)^T$
- and the perpendicular distance of the hyperplane from the origin is given by

$$\frac{|w_0|}{\sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}}$$
Threshold neuron as a pattern classifier

- The threshold neuron can be used to classify a set of instances into one of two classes $C_1, C_2$
- If the output of the neuron for input pattern $X_p$ is +1 then $X_p$ is assigned to class $C_1$
- If the output is -1 then the pattern $X_p$ is assigned to $C_2$

Example

$$[w_0 \ w_1 \ w_2]^T = [-1 \ -1 \ 1]^T$$

$$X_p^T = [1 \ 0]^T \ W \cdot X_p + w_0 = -1 + (-1) = -2$$

$X_p$ is assigned to class $C_2$

---

Threshold neuron – Connection with Logic

- Suppose the input space is $\{0,1\}^n$
- Then threshold neuron computes a Boolean function $f : \{0,1\}^n \rightarrow \{-1,1\}$

**Example**

Let $w_0 = -1.5; \ w_1 = w_2 = 1$

In this case, the threshold neuron implements the logical AND function

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$g(X)$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1.5</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-0.5</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-0.5</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>
Threshold neuron – Connection with Logic

- A threshold neuron with the appropriate choice of weights can implement Boolean AND, OR, and NOT function
- **Theorem:** For any arbitrary Boolean function \( f \), there exists a network of threshold neurons that can implement \( f \).
- **Theorem:** Any arbitrary finite state automaton can be realized using threshold neurons and *delay* units
- Networks of threshold neurons, given access to unbounded memory, can compute any Turing-computable function
- **Corollary:** Brains if given access to enough working memory, can compute any computable function

---

**Threshold neuron: Connection with Logic**

**Theorem:** There exist functions that cannot be implemented by a single threshold neuron.

**Example** exclusive OR

![Diagram](diagram.png)

Why?
Threshold neuron – Connection with Logic

- **Definition**: A function that can be computed by a single threshold neuron is called a threshold function
- Of the 16 2-input Boolean functions, 14 are Boolean threshold functions
- As $n$ increases, the number of Boolean threshold functions becomes an increasingly small fraction of the total number of $n$-input Boolean functions

$$N_{\text{Threshold}}(n) \leq 2^{n^2} \quad N_{\text{Boolean}}(n) = 2^{2^n}$$

Terminology and Notation

- **Synonym**: Threshold function, Linearly separable function, linear discriminant function
- **Synonym**: Threshold neuron, McCulloch-Pitts neuron, Perceptron, Threshold Logic Unit (TLU)
- We often include $w_0$ as one of the components of $W$ and incorporate $x_0$ as the corresponding component of $X$ with the understanding that $x_0 = 1$. Then $y = 1$ if $W.X > 0$ and $y = -1$ otherwise.
Learning Threshold functions

A training example $E_k$ is an ordered pair $(x_k, d_k)$ where

$$X_k = [x_{0k}, x_{1k}, \ldots, x_{nk}]$$

is an $(n+1)$ dimensional input pattern, $d_k = f(X_k) \in \{-1, 1\}$ is the desired output of the classifier and $f$ is an unknown target function to be learned.

A training set $E$ is simply a multi-set of examples.

\[
S^+ = \left\{ X_k \middle| (X_k, d_k) \in E \text{ and } d_k = 1 \right\},
\]
\[
S^- = \left\{ X_k \middle| (X_k, d_k) \in E \text{ and } d_k = -1 \right\}
\]

We say that a training set $E$ is linearly separable if and only if

$$\exists W^* \text{ such that } \forall X_p \in S^+, W^* \cdot X_p > 0$$

and

$$\forall X_p \in S^-, W^* \cdot X_p < 0$$

**Learning Task:** Given a linearly separable training set $E$, find a solution

$$W^* \text{ such that } \forall X_p \in S^+, W^* \cdot X_p > 0 \text{ and } \forall X_p \in S^-, W^* \cdot X_p < 0$$
Rosenblatt’s Perceptron Learning Algorithm

Initialize $W = [0 \ 0 \ldots 0]$ Set learning rate $\eta > 0$

Repeat until a complete pass through $E$ results in no weight updates

For each training example $E_k \in E$

$$y_k \leftarrow sign(W \cdot X_k)$$
$$W \leftarrow W + \eta (d_k - y_k)X_k$$

$$W^* \leftarrow W; \quad Return (W^*)$$

Perceptron learning algorithm – Example

Let

$S^+ = \{(1, 1, 1), (1, 1, -1), (1, 0, -1)\}$

$S^- = \{(1,-1, -1), (1,-1, 1), (1,0, 1)\}$

$W= (0 \ 0 \ 0)$;

<table>
<thead>
<tr>
<th>$X_k$</th>
<th>$d_k$</th>
<th>$W$</th>
<th>$W \cdot X_k$</th>
<th>$y_k$</th>
<th>Update?</th>
<th>Updated $W$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1, 1)</td>
<td>1</td>
<td>(0, 0, 0)</td>
<td>0</td>
<td>-1</td>
<td>Yes</td>
<td>(1, 1, 1)</td>
</tr>
<tr>
<td>(1, 1, -1)</td>
<td>1</td>
<td>(1, 1, 1)</td>
<td>1</td>
<td>1</td>
<td>No</td>
<td>(1, 1, 1)</td>
</tr>
<tr>
<td>(1, 0, -1)</td>
<td>1</td>
<td>(1, 1, 1)</td>
<td>0</td>
<td>-1</td>
<td>Yes</td>
<td>(2, 1, 0)</td>
</tr>
<tr>
<td>(1, -1, -1)</td>
<td>-1</td>
<td>(2, 1, 0)</td>
<td>1</td>
<td>1</td>
<td>Yes</td>
<td>(1, 2, 1)</td>
</tr>
<tr>
<td>(1, -1, 1)</td>
<td>-1</td>
<td>(1, 2, 1)</td>
<td>0</td>
<td>-1</td>
<td>No</td>
<td>(1, 2, 1)</td>
</tr>
<tr>
<td>(1, 0, 1)</td>
<td>-1</td>
<td>(1, 2, 1)</td>
<td>2</td>
<td>1</td>
<td>Yes</td>
<td>(0, 2, 0)</td>
</tr>
<tr>
<td>(1, 1, 1)</td>
<td>1</td>
<td>(0, 2, 0)</td>
<td>2</td>
<td>1</td>
<td>No</td>
<td>(0, 2, 0)</td>
</tr>
</tbody>
</table>
Perceptron (1957)

---

Perceptron
Perceptron Convergence Theorem (Novikoff)

**Theorem** Let \( E = \{ (x_k, d_k) \} \) be a training set where \( x_k \in \{1\} \times \mathbb{R}^n \) and \( d_k \in \{-1, 1\} \)

Let \( S^+ = \{ (x_k, d_k) \in E \& d_k = 1 \} \) and \( S^- = \{ (x_k, d_k) \in E \& d_k = -1 \} \)

The perceptron algorithm is guaranteed to terminate after a bounded number \( t \) of weight updates with a weight vector \( W^* \) such that \( \forall x_k \in S^+, W^* \cdot x_k \geq \delta \) and \( \forall x_k \in S^-, W^* \cdot x_k \leq -\delta \) for some \( \delta > 0 \), whenever such \( W^* \in \mathbb{R}^{n+1} \) and \( \delta > 0 \) exist -- that is, \( E \) is linearly separable. The bound on the number \( t \) of weight updates is given by

\[
    t \leq \left( \frac{\|W^*\|L}{\delta} \right)^2
\]

where \( L = \max_{x_k \in S} \|x_k\| \) and \( S = S^+ \cup S^- \)

Proof of Perceptron Convergence Theorem

Let \( W_t \) be the weight vector after \( t \) weight updates.

\[ W^* \]

\[ W_t \]

**Invariant:** \( \forall \theta \quad |\cos \theta| \leq 1 \)
Proof of Perceptron Convergence Theorem

Let $W^*$ be such that

$$\forall X_k \in S^+, W^* \cdot X_k \geq \delta \text{ and } \forall X_k \in S^-, W^* \cdot X_k \leq -\delta$$

WLOG assume that $W^* \cdot X = 0$ passes through the origin.

Let $\forall X_k \in S^+, Z_k = X_k$,

$\forall X_k \in S^-, Z_k = -X_k$,

$Z = \{Z_k\}$

$$(\forall X_k \in S^+, W^* \cdot X_k \geq \delta \text{ and } \forall X_k \in S^-, W^* \cdot X_k \leq -\delta) \iff (\forall Z_k \in Z, W^* \cdot Z_k \geq \delta)$$

Let $E^* = \{Z_k, 1\}$

---

Proof of Perceptron Convergence Theorem

$$W_{t+1} = W_t + \eta (d_k - y_k)Z_k$$

where $W_0 = [0 \ 0 \ ... \ 0]^T$ and $\eta > 0$

[Weight update based on example ($Z_k, 1$)]

$$\iff [(d_k = 1) \wedge (y_k = -1)]$$

$$\therefore W^* \cdot W_{t+1} = W^* \cdot (W_t + 2\eta Z_k) = (W^* \cdot W_t) + 2\eta (W^* \cdot Z_k)$$

Since $\forall Z_k \in Z, (W^* \cdot Z_k \geq \delta), W^* \cdot W_{t+1} \geq W^* \cdot W_t + 2\eta \delta$

$$\therefore \forall t \ W^* \cdot W_t \geq 2t\eta \delta \hdots (a)$$
Proof of Perceptron Convergence Theorem

\[ \|W_{t+1}\|^2 = W_{t+1} \cdot W_{t+1} \]
\[ = (W_t + 2\eta Z_k) \cdot (W_t + 2\eta Z_k) \]
\[ = (W_t \cdot W_t) + 4\eta (W_t \cdot Z_k) + 4\eta^2 (Z_k \cdot Z_k) \]

Note weight update based on \(Z_k \iff (W_t \cdot Z_k \leq 0)\)

\[ \therefore \|W_{t+1}\|^2 \leq \|W_t\|^2 + 4\eta^2 \|Z_k\|^2 \leq \|W_t\|^2 + 4\eta^2 L^2 \]

Hence \(\|W_t\|^2 \leq 4\eta^2 L^2\)

\[ \therefore \forall t \|W_t\| \leq 2\eta L\sqrt{t} \]

..............................(b)

Proof of Perceptron Convergence Theorem

From (a) we have : \(\forall t \ (W^* \cdot W_t) \geq 2t\eta \delta\)

\(\Rightarrow \{ \forall t \ 2t\eta \delta \leq (W^* \cdot W_t) \} \Rightarrow \{ \forall t \ 2t\eta \delta \leq \|W^*\|\|W_t\|\cos \theta \} \)

\(\Rightarrow \{ \forall t \ 2t\eta \delta \leq \|W^*\|\|W_t\| \} \Rightarrow \forall \theta \ \cos \theta \leq 1,\)

Substituting for an upper bound on \(\|W_t\|\) from (b),

\(\forall t \ \{2t\eta \delta \leq \|W^*\|2\eta L\sqrt{t}\} \Rightarrow \{ \forall t \ \left( \delta \sqrt{t} \leq \|W^*\|L \right) \}

\(\Rightarrow t \leq \left( \frac{\|W^*\|L}{\delta} \right)^2 \)
Notes on the Perceptron Convergence Theorem

- The bound on the number of weight updates does not depend on the learning rate
- The bound is not useful in determining when to stop the algorithm because it depends on the norm of the unknown weight vector and delta
- The convergence theorem offers no guarantees when the training data set is not linearly separable

**Exercise:** Prove that the perceptron algorithm is robust with respect to fluctuations in the learning rate

\[ 0 < \eta_{\min} \leq \eta_t \leq \eta_{\max} < \infty \]

---

### Multiple classes

- **$K - 1$ binary classifiers**
- **$\frac{K(K - 1)}{2}$ binary classifiers**

**One-versus-rest**

**One-versus-one**

**Problem:** The green region has ambiguous class membership
Multi-category classifiers

Define $K$ linear functions of the form:

$$y_k(X) = W_k^T X + w_{k0}$$

$$h(X) = \arg \max_k y_k(X)$$

$$= \arg \max_k (W_k^T X + w_{k0})$$

Decision surface between class $C_k$ and $C_j$

$$(W_k - W_j)^T X + (w_{k0} - w_{j0}) = 0$$

Linear separator for $K$ classes

- Decision regions defined by

$$\left(W_k - W_j\right)^T X + (w_{k0} - w_{j0}) = 0$$

are singly connected and convex

For any points $X_A, X_B \in R_k$,

any $\hat{X}$ that lies on the line connecting $X_A$ and $X_B$

$$\hat{X} = \lambda X_A + (1 - \lambda)X_B$$ where $0 \leq \lambda \leq 1$

also lies in $R_k$
Winner-Take-All Networks

\[ y_p = 1 \text{ iff } \mathbf{W}_i \cdot \mathbf{X}_p > \mathbf{W}_j \cdot \mathbf{X}_p \quad \forall j \neq i \]

\[ y_p = 0 \text{ otherwise} \]  

Note: \( \mathbf{W}_j \) are augmented weight vectors

\[ \mathbf{W}_1 = [1 \quad -1 \quad -1]^T, \mathbf{W}_2 = [1 \quad 1 \quad 1]^T, \mathbf{W}_3 = [2 \quad 0 \quad 0]^T \]

<table>
<thead>
<tr>
<th></th>
<th>( \mathbf{W}_1 \cdot \mathbf{X}_p )</th>
<th>( \mathbf{W}_2 \cdot \mathbf{X}_p )</th>
<th>( \mathbf{W}_3 \cdot \mathbf{X}_p )</th>
<th>( y_1 )</th>
<th>( y_2 )</th>
<th>( y_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>3</td>
<td>-1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>+1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>+1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>+1</td>
<td>+1</td>
<td>-1</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

What does neuron 3 compute?

Linear separability of multiple classes

Let \( S_1, S_2, S_3, \ldots, S_M \) be multisets of instances

Let \( C_1, C_2, C_3, \ldots, C_M \) be disjoint classes

\[ \forall i \quad S_i \subseteq C_i \]

\[ \forall i \neq j \quad C_i \cap C_j = \emptyset \]

We say that the sets \( S_1, S_2, S_3, \ldots, S_M \) are linearly separable iff there exist weight vectors \( \mathbf{W}_1^*, \mathbf{W}_2^*, \ldots, \mathbf{W}_M^* \) such that

\[ \forall i \quad \left\{ \forall \mathbf{X}_p \in S_i, \left( \mathbf{W}_i^* \cdot \mathbf{X}_p > \mathbf{W}_j^* \cdot \mathbf{X}_p \right) \forall j \neq i \right\} \]
Training WTA Classifiers

\[ d_{kp} = 1 \text{ iff } X_p \in C_k ; d_{kp} = 0 \text{ otherwise} \]
\[ y_{kp} = 1 \text{ iff } W_k \cdot X_p > W_j \cdot X_p \; \forall k \neq j \]

Suppose \( d_{kp} = 1, y_{jp} = 1 \) and \( y_{kp} = 0 \)

\[ W_k \leftarrow W_k + \eta X_p, \; W_j \leftarrow W_j - \eta X_p ; \]

All other weights are left unchanged.

Suppose \( d_{kp} = 1, y_{jp} = 0 \) and \( y_{kp} = 1 \).

The weights are unchanged.

Suppose \( d_{kp} = 1, \forall j \; y_{jp} = 0 \) (there was a tie)

\[ W_k \leftarrow W_k + \eta X_p \]

All other weights are left unchanged.

---

WTA Convergence Theorem

Given a linearly separable training set, the WTA learning algorithm is guaranteed to converge to a solution within a finite number of weight updates.

Proof Sketch: Transform the WTA training problem to the problem of training a single perceptron using a suitably transformed training set. Then the proof of WTA learning algorithm reduces to the proof of perceptron learning algorithm.
Let \( W = [W_1, W_2, \ldots, W_M]^T \) be a concatenation of the weight vectors associated with the \( M \) neurons in the WTA group. Consider a multi-category training set \( E = \left\{ (X_p, f(X_p)) \right\} \) where \( \forall X_p \in \{C_1, \ldots, C_M\} \).

Let \( X_p \in C_i \). Generate \((M-1)\) training examples using \( X_p \) for an \( M_n + 1 \) input perceptron:

\[
\begin{align*}
X_{p12} &= [X_p - X_p \phi \phi \ldots \phi] \\
X_{p13} &= [X_p - X_p \phi \phi \ldots \phi] \\
&\vdots \\
X_{p1M} &= [X_p \phi \phi \ldots \phi - X_p] 
\end{align*}
\]

where \( \phi \) is an all zero vector with the same dimension as \( X_p \) and set the desired output of the corresponding perceptron to be 1 in each case.

Similarly, from each training example for an \((n + 1)\) - input WTA, we can generate \((M - 1)\) examples for an \( M(n + 1) \) input single neuron. Let the union of the resulting \( |E|/M - 1 \) examples be \( E' \).

By construction, there is a one-to-one correspondence between the weight vector \( W = [W_1, W_2, \ldots, W_M]^T \) that results from training an \( M \) - neuron WTA on the multi-category set of examples \( E \) and the result of training an \( M(n + 1) \) input perceptron on the transformed training set \( E' \). Hence the convergence proof of WTA learning algorithm follows from the perceptron convergence theorem.
Weight space representation

**Pattern space representation**
Coordinates of space correspond to attributes (features)
A point in the space represents an instance
Weight vector $W_v$ defines a hyperplane $W_v \cdot X = 0$

**Weight space (dual) representation**
Coordinates define a weight space
A point in the space represents a choice of weights $W_v$
An instance $X_p$ defines a hyperplane $W \cdot X_p = 0$
Weight space representation

\[ W_{i+1} \leftarrow W_i + \eta X_p \]

Fractional correction rule

\[ W_{i+1} \leftarrow W_i + \lambda \left( \frac{|W_i \cdot X_p| + \varepsilon}{X_p \cdot X_p + \varepsilon} \right) (d_p - y_p) X_p \]

0 < \lambda < 1; \lambda \neq 0.5 when \( d_p, y_p \in \{-1, 1\} \)

\varepsilon > 0 is a constant (to handle the case when the dot product \( W_i \cdot X_p \) or \( X_p \cdot X_p \) (or both) approach zero.

---

The Perceptron Algorithm Revisited

The perceptron works by adding misclassified positive or subtracting misclassified negative examples to an arbitrary weight vector, which (without loss of generality) we assumed to be the zero vector. So the final weight vector is a linear combination of training points

\[ W = \sum_{i=1}^{l} \alpha_i y_i x_i, \]

where, since the sign of the coefficient of \( x_i \) is given by label \( y_i \), the \( \alpha_i \) are positive values, proportional to the number of times, misclassification of \( x_i \) has used the weight to be updated. It is called the embedding strength of the pattern \( x_i \).
Dual Representation

The decision function can be rewritten as:

\[ h(x) = \text{sgn}\left(\langle w, x \rangle + b\right) = \text{sgn}\left(\left(\sum_{j=1}^{l} \alpha_j y_j \langle x_j, x \rangle\right) + b\right) \]

on training example \((x_i, y_i)\)

The update rule is:

\[ \text{if } y_i \left(\sum_{j=1}^{l} \alpha_j y_j \langle x_j, x_i \rangle + b\right) \leq 0, \]

\[ \text{then } \alpha_i \leftarrow \alpha_i + \eta \]

WLOG, we can take \(\eta = 1\).

Limitations of perceptrons

- Perceptrons can only represent threshold functions
- Perceptrons can only learn linear decision boundaries

What if the data are not linearly separable?
- More complex networks?
- Non-linear transformations into a feature space where the data become separable?
Extending Linear Classifiers - Learning in feature spaces

Map data into a feature space where they are linearly separable

In the feature (hidden) space:

\[
\begin{align*}
\varphi_1(x_1, x_2) &= e^{-||x - w_1||^2} = z_1 \\
\varphi_2(x_1, x_2) &= e^{-||x - w_2||^2} = z_2
\end{align*}
\]

\[
W_1 = [1, 1]^T \\
W_2 = [0, 0]^T
\]

When mapped into the feature space \(<z_1, z_2>\), C1 and C2 become linearly separable. So a linear classifier with \(\varphi_1(x)\) and \(\varphi_2(x)\) as inputs can be used to solve the XOR problem.
“Perceptrons” (1969)

“The perceptron […] has many features that attract attention: its linearity, its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues 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.”

[pp. 231 – 232]

Learning in the Feature Spaces

High dimensional Feature spaces

\[ \mathbf{x} = (x_1, x_2, \ldots, x_n) \rightarrow \varphi(\mathbf{x}) = (\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_d(\mathbf{x})) \]

where typically \( d \gg n \) solve the problem of expressing complex functions

But this introduces a

- computational problem (working with very large vectors)
  - Solved using the kernel trick – implicit feature spaces
- generalization problem (curse of dimensionality)
  - Solved by maximizing the margin of separation – first implemented in SVM (Vapnik)

We will return to SVM later