Machine Learning
ITCS 4156

The Perceptron Algorithm
The Kernel Trick

Razvan C. Bunescu
Department of Computer Science @ CCI
rbunescu@uncc.edu
Supervised Learning

Training

Training Examples
\((x_k, t_k)\)

Learning Algorithm

Model \(h\)

Testing

Test Examples
\((x, t)\)

Model \(h\)

Generalization Performance
Supervised Learning

- **Task** = learn an (unknown) function $t : X \rightarrow T$ that maps input instances $x \in X$ to output targets $t(x) \in T$:
  - **Classification**:
    - The output $t(x) \in T$ is one of a finite set of discrete categories.
  - **Regression**:
    - The output $t(x) \in T$ is continuous, or has a continuous component.

- Target function $t(x)$ is known (only) through (noisy) set of training examples:
  $$(x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n)$$
Three Parametric Approaches to Classification

1) **Discriminant Functions**: construct \( f : X \rightarrow T \) that directly assigns a vector \( \mathbf{x} \) to a specific class \( C_k \).
   
   – Inference and decision combined into a single learning problem.
   
   – *Linear Discriminant*: the decision surface is a hyperplane in \( X \):
     
     - Perceptron
     - Support Vector Machines
     - Fisher ‘s Linear Discriminant
Three Parametric Approaches to Classification

2) **Probabilistic Discriminative Models:** directly model the posterior class probabilities $p(C_k \mid x)$.
   - Inference and decision are separate.
   - Less data needed to estimate $p(C_k \mid x)$ than $p(x \mid C_k)$.
   - Can accommodate many overlapping features.
     - Logistic Regression
     - Conditional Random Fields
Three Parametric Approaches to Classification

3) **Probabilistic Generative Models:**
   - Model class-conditional $p(x \mid C_k)$ as well as the priors $p(C_k)$, then use Bayes’s theorem to find $p(C_k \mid x)$.
     - or model $p(x, C_k)$ directly, then marginalize to obtain the posterior probabilities $p(C_k \mid x)$.
   - Inference and decision are separate.
   - Can use $p(x)$ for outlier or novelty detection.
   - Need to model dependencies between features.
     - Naïve Bayes.
     - Hidden Markov Models.
Generative vs. Discriminative

Left-hand mode has no effect on posterior class probabilities.
Three Parametric Approaches to Classification

1) **Discriminant Functions**: construct \( h: X \rightarrow T \) that directly assigns a vector \( x \) to a specific class \( C_k \).
   - Inference and decision combined into a single learning problem.
   - **Linear Discriminant**: the decision surface is a hyperplane in \( X \):
     - Perceptron
     - Support Vector Machines
     - Fisher’s Linear Discriminant
Discriminant Function Approach to Classification

• **Task** = build a function \( h(x) \) such that:
  - \( h \) matches \( t \) well on the training data:
    => \( h \) is able to fit data that it has seen.
  - \( h \) also matches \( t \) well on test data:
    => \( h \) is able to **generalize to unseen data**.

• **Task** = choose \( h \) from a “nice” class of functions that depend on a vector of parameters \( w \):
  - \( h(x) = h_w(x) = h(w,x) \)
  - what classes of functions are “nice”??
Neurons

**Soma** is the central part of the neuron:
- *where the input signals are combined.*

**Dendrites** are cellular extensions:
- *where majority of the input occurs.*

**Axon** is a fine, long projection:
- *carries nerve signals to other neurons.*

**Synapses** are molecular structures between axon terminals and other neurons:
- *where the communication takes place.*
# Neuron Models


<table>
<thead>
<tr>
<th>Year</th>
<th>Model Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1907</td>
<td>Integrate and fire</td>
<td>[13]</td>
</tr>
<tr>
<td>1943</td>
<td>McCulloch and Pitts</td>
<td>[11]</td>
</tr>
<tr>
<td>1952</td>
<td>Hodgkin-Huxley</td>
<td>[12]</td>
</tr>
<tr>
<td>1958</td>
<td>Perceptron</td>
<td>[14]</td>
</tr>
<tr>
<td>1961</td>
<td>Fitzhugh-Nagumo</td>
<td>[15]</td>
</tr>
<tr>
<td>1965</td>
<td>Leaky integrate-and-fire</td>
<td>[16]</td>
</tr>
<tr>
<td>1981</td>
<td>Morris-Lecar</td>
<td>[17]</td>
</tr>
<tr>
<td>1986</td>
<td>Quadratic integrate-and-fire</td>
<td>[18]</td>
</tr>
<tr>
<td>1989</td>
<td>Hindmarsh-Rose</td>
<td>[19]</td>
</tr>
<tr>
<td>1999</td>
<td>Wilson Polynomial</td>
<td>[21]</td>
</tr>
<tr>
<td>2000</td>
<td>Integrate-and-fire or burst</td>
<td>[22]</td>
</tr>
<tr>
<td>2001</td>
<td>Resonate-and-fire</td>
<td>[23]</td>
</tr>
<tr>
<td>2003</td>
<td>Izhikevich</td>
<td>[24]</td>
</tr>
<tr>
<td>2003</td>
<td>Exponential integrate-and-fire</td>
<td>[25]</td>
</tr>
<tr>
<td>2004</td>
<td>Generalized integrate-and-fire</td>
<td>[26]</td>
</tr>
<tr>
<td>2005</td>
<td>Adaptive exponential integrate-and-fire</td>
<td>[27]</td>
</tr>
<tr>
<td>2009</td>
<td>Mihalas-Neibur</td>
<td>[28]</td>
</tr>
</tbody>
</table>
Spiking/LIF Neuron Function


Fig. 2. (a) Illustration and (b) functional description of a leaky integrate-and-fire neuron. Weighted and delayed input signals are summed into the input current $I_{\text{app}}(t)$, which travel to the soma and perturb the internal state variable, the voltage $V$. Since $V$ is hysteric, the soma performs integration and then applies a threshold to make a spike or no-spike decision. After a spike is released, the voltage $V$ is reset to a value $V_{\text{reset}}$. The resulting spike is sent to other neurons in the network.
# Neuron Models


<table>
<thead>
<tr>
<th>Year</th>
<th>Model Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1907</td>
<td>Integrate and fire</td>
<td>[13]</td>
</tr>
<tr>
<td>1943</td>
<td>McCulloch and Pitts</td>
<td>[11]</td>
</tr>
<tr>
<td>1952</td>
<td>Hodgkin-Huxley</td>
<td>[12]</td>
</tr>
<tr>
<td>1958</td>
<td>Perceptron</td>
<td>[14]</td>
</tr>
<tr>
<td>1961</td>
<td>Fitzhugh-Nagumo</td>
<td>[15]</td>
</tr>
<tr>
<td>1965</td>
<td>Leaky integrate-and-fire</td>
<td>[16]</td>
</tr>
<tr>
<td>1981</td>
<td>Morris-Lecar</td>
<td>[17]</td>
</tr>
<tr>
<td>1986</td>
<td>Quadratic integrate-and-fire</td>
<td>[18]</td>
</tr>
<tr>
<td>1989</td>
<td>Hindmarsh-Rose</td>
<td>[19]</td>
</tr>
<tr>
<td>1999</td>
<td>Wilson Polynomial</td>
<td>[21]</td>
</tr>
<tr>
<td>2000</td>
<td>Integrate-and-fire or burst</td>
<td>[22]</td>
</tr>
<tr>
<td>2001</td>
<td>Resonate-and-fire</td>
<td>[23]</td>
</tr>
<tr>
<td>2003</td>
<td>Izhikevich</td>
<td>[24]</td>
</tr>
<tr>
<td>2003</td>
<td>Exponential integrate-and-fire</td>
<td>[25]</td>
</tr>
<tr>
<td>2004</td>
<td>Generalized integrate-and-fire</td>
<td>[26]</td>
</tr>
<tr>
<td>2005</td>
<td>Adaptive exponential integrate-and-fire</td>
<td>[27]</td>
</tr>
<tr>
<td>2009</td>
<td>Mihalas-Neibur</td>
<td>[28]</td>
</tr>
</tbody>
</table>
McCulloch-Pitts Neuron Function

• Algebraic interpretation:
  – The output of the neuron is a linear combination of inputs from other neurons, rescaled by the synaptic weights.
  - weights $w_i$ correspond to the synaptic weights (activating or inhibiting).
  - summation corresponds to combination of signals in the soma.
  – It is often transformed through a monotonic activation function.
Activation/Output Functions

**unit step** \( f(z) = \begin{cases} 
0 & \text{if } z < 0 \\
1 & \text{if } z \geq 0
\end{cases} \)

Perceptron

**logistic** \( f(z) = \frac{1}{1 + e^{-z}} \)

Logistic Regression

**identity** \( f(z) = z \)

Linear Regression
• Assume classes $T = \{c_1, c_2\} = \{1, -1\}$.
• Training set is $(x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n)$.
  
  \[ x = [1, x_1, x_2, \ldots, x_k]^T \]
  
  \[ h(x) = \text{sgn}(w^T x) = \text{sgn}(w_0 + w_1 x_1 + \ldots + w_k x_k) \]

A linear discriminant function
Linear Discriminant Functions

• Use a linear function of the input vector:
  \[ h(x) = w^T \varphi(x) + w_0 \]

• Decision:
  \[ x \in C_1 \text{ if } h(x) \geq 0, \text{ otherwise } x \in C_2. \]
  \[ \Rightarrow \text{decision boundary is hyperplane } h(x) = 0. \]

• Properties:
  – \( w \) is orthogonal to vectors lying within the decision surface.
  – \( w_0 \) controls the location of the decision hyperplane.
Geometric Interpretation

\[ h > 0 \]
\[ h = 0 \]
\[ h < 0 \]
Linear Discriminant Functions:
Two Classes ($K = 2$)

• What algorithms can be used to learn $y(x) = w^T \varphi(x) + w_0$?
  Assume a training dataset of $N = N_1 + N_2$ examples in $C_1$ and $C_2$.

  – Perceptron:
    • Voted/Averaged Perceptron
    • Kernel Perceptron

  – Support Vector Machines:
    • Linear
    • Kernel

  – Fisher’s Linear Discriminant
• Assume classes $T = \{c_1, c_2\} = \{1, -1\}$.
• Training set is $(x_1, t_1), (x_2, t_2), \ldots, (x_n, t_n)$.
  $x = [1, x_1, x_2, \ldots, x_k]^T$
  $h(x) = sgn(w^T x) = sgn(w_0 + w_1 x_1 + \ldots + w_k x_k)$

An example of a linear discriminant function.
Perceptron Learning

• Learning = finding the “right” parameters $w^T = [w_0, w_1, \ldots, w_k]$
  - Find $w$ that minimizes an error function $E(w)$ which measures the misfit between $h(x_n, w)$ and $t_n$.
  - Expect that $h(x, w)$ performing well on training examples $x_n \Rightarrow h(x, w)$ will perform well on arbitrary test examples $x \in X$.

• Least Squares error function?

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \left\{ h(x_n, w) - t_n \right\}^2$$

2 x # of mistakes
Least Squares vs. Perceptron Criterion

- **Least Squares** => cost is # of misclassified patterns:
  - Piecewise constant function of $\mathbf{w}$ with discontinuities.
  - Cannot find closed form solution for $\mathbf{w}$ that minimizes cost.
  - Cannot use gradient methods (gradient zero almost everywhere).

- **Perceptron Criterion**:
  - Set labels to be $+1$ and $-1$. Want $\mathbf{w}^T \mathbf{x}_n > 0$ for $t_n = 1$, and $\mathbf{w}^T \mathbf{x}_n < 0$ for $t_n = -1$.
    - would like to have $\mathbf{w}^T \mathbf{x}_n t_n > 0$ for all patterns.
    - want to minimize $-\mathbf{w}^T \mathbf{x}_n t_n$ for all misclassified patterns $M$.

  $\Rightarrow$ minimize $E_p(\mathbf{w}) = - \sum_{n \in M} \mathbf{w}^T \mathbf{x}_n t_n$
Stochastic Gradient Descent

- **Perceptron Criterion:**

  \[
  \text{minimize } E_p(w) = -\sum_{n \in M} w^T x_n t_n
  \]

- Update parameters \( w \) sequentially after each mistake:

  \[
  w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_p(w^{(\tau)}, x_n)
  \]

  \[
  = w^{(\tau)} + \eta x_n t_n
  \]

- The magnitude of \( w \) is inconsequential \( \Rightarrow \) set \( \eta = 1 \).

  \[
  w^{(\tau+1)} = w^{(\tau)} + x_n t_n
  \]
The Perceptron Algorithm: Two Classes

1. **initialize** parameters $\mathbf{w} = 0$
2. **for** $n = 1 \ldots N$
3. $h_n = sgn(\mathbf{w}^T \mathbf{x}_n)$
4. **if** $h_n \neq t_n$ **then**
5. $\mathbf{w} = \mathbf{w} + t_n \mathbf{x}_n$

Repeat:

a) until convergence.

b) for a number of epochs $E$.

Theorem [Rosenblatt, 1962]:

If the training dataset is linearly separable, the perceptron learning algorithm is guaranteed to find a solution in a finite number of steps.

- see Theorem 1 (Block, Novikoff) in [Freund & Schapire, 1999].