Lecture 10:
- Linear Discriminant Functions
- Perceptron
Last time... Logistic Regression

Assumes the following functional form for $P(Y|X)$:

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)}$$

Logistic function applied to linear function of the data

Logistic function (or Sigmoid): $$\frac{1}{1 + \exp(-z)}$$

Features can be discrete or continuous!
Last time.. **Logistic Regression vs. Gaussian Naïve Bayes**

- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by maximizing conditional likelihood
  - no closed-form solution
  - concave! global optimum with gradient ascent
- Gaussian Naïve Bayes with class-independent variances representationally equivalent to LR
  - Solution differs because of objective (loss) function
- In general, NB and LR make different assumptions
  - NB: Features independent given class! assumption on $P(X|Y)$
  - LR: Functional form of $P(Y|X)$, no assumption on $P(X|Y)$
- Convergence rates
  - GNB (usually) needs less data
  - LR (usually) gets to better solutions in the limit
Linear Discriminant Functions
Linear Discriminant Function

• Linear discriminant function for a vector \( x \)

\[
y(x) = w^T x + w_0
\]

where \( w \) is called weight vector, and \( w_0 \) is a bias.

• The classification function is

\[
C(x) = \text{sign}(w^T x + w_0)
\]

where step function \( \text{sign}() \) is defined as

\[
\text{sign}(a) = \begin{cases} 
+1, & a \geq 0 \\
-1, & a < 0
\end{cases}
\]
Properties of Linear Discriminant Functions

- The decision surface, shown in red, is perpendicular to \( \mathbf{w} \), and its displacement from the origin is controlled by the bias parameter \( w_0 \).
- The signed orthogonal distance of a general point \( \mathbf{x} \) from the decision surface is given by \( y(\mathbf{x})/||\mathbf{w}|| \).
- \( y(\mathbf{x}) \) gives a signed measure of the perpendicular distance \( r \) of the point \( \mathbf{x} \) from the decision surface.

- \( y(\mathbf{x}) = 0 \) for \( \mathbf{x} \) on the decision surface. The normal distance from the origin to the decision surface is

\[
\frac{\mathbf{w}^T \mathbf{x}}{||\mathbf{w}||} = -\frac{w_0}{||\mathbf{w}||}
\]

- So \( w_0 \) determines the location of the decision surface.
Properties of Linear Discriminant Functions

- Let
  \[ \mathbf{x} = \mathbf{x}_\perp + r \frac{\mathbf{w}}{\|\mathbf{w}\|} \]
  where \( \mathbf{x}_\perp \) is the projection \( \mathbf{x} \) on the decision surface. Then

  \[ \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mathbf{x}_\perp + r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} \]

  \[ \mathbf{w}^T \mathbf{x} + w_0 = \mathbf{w}^T \mathbf{x}_\perp + w_0 + r \|\mathbf{w}\| \]

  \[ y(\mathbf{x}) = r \|\mathbf{w}\| \]

  \[ r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|} \]

- Simpler notion: define \( \tilde{\mathbf{w}} = (w_0, \mathbf{w}) \) and \( \tilde{\mathbf{x}} = (1, \mathbf{x}) \) so that

  \[ y(\mathbf{x}) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}} \]
Multiple Classes: Simple Extension

- **One-versus-the-rest** classifier: classify $C_k$ and samples not in $C_k$.
- **One-versus-one** classifier: classify every pair of classes.
Multiple Classes: K-Class Discriminant

- A single $K$-class discriminant comprising $K$ linear functions
  \[ y_k(x) = w_k^T x + w_{k0} \]

- Decision function
  \[ C(x) = k, \text{ if } y_k(x) > y_j(x) \forall j \neq k \]

- The decision boundary between class $C_k$ and $C_j$ is given by
  \[ y_k(x) = y_j(x) \]
  \[ (w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0 \]
Fisher’s Linear Discriminant

- Pursue the optimal linear projection on which the two classes can be maximally separated
  \[ y = w^T x \]

- The mean vectors of the two classes
  \[
  m_1 = \frac{1}{N_1} \sum_{n \in C_1} x_n, \quad m_2 = \frac{1}{N_2} \sum_{n \in C_2} x_n
  \]

A way to view a linear classification model is in terms of dimensionality reduction.
What’s a Good Projection?

- After projection, the two classes are separated as much as possible. Measured by the distance between projected center.
  \[
  \left( w^T (m_1 - m_2) \right)^2 = w^T (m_1 - m_2)(m_1 - m_2)^T w \\
  = w^T S_B w
  \]
  where \( S_B = (m_1 - m_2)(m_1 - m_2)^T \) is called between-class covariance matrix.

- After projection, the variances of the two classes are as small as possible. Measured by the within-class covariance.
  \[
  w^T S_W w
  \]
  where
  \[
  S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T
  \]
Fisher’s Linear Discriminant

- Fisher criterion: maximize the ratio w.r.t. \( \mathbf{w} \)
  \[
  J(\mathbf{w}) = \frac{\text{Between-class variance}}{\text{Within-class variance}} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}
  \]

- Recall the quotient rule: for \( f(x) = \frac{g(x)}{h(x)} \)
  \[
  f'(x) = \frac{g'(x)h(x) - g(x)h'(x)}{h^2(x)}
  \]

- Setting \( \nabla J(\mathbf{w}) = 0 \), we obtain
  \[
  (\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w}) \mathbf{S}_B \mathbf{w}
  \]
  \[
  (\mathbf{w}^T \mathbf{S}_B \mathbf{w}) \mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w})(\mathbf{m}_2 - \mathbf{m}_1)\left((\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w}\right)
  \]

- Terms \( \mathbf{w}^T \mathbf{S}_B \mathbf{w}, \mathbf{w}^T \mathbf{S}_W \mathbf{w} \) and \( (\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w} \) are scalars, and we only care about directions. So the scalars are dropped. Therefore
  \[
  \mathbf{w} \propto \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)
  \]
From Fisher’s Linear Discriminant to Classifiers

• Fisher’s Linear Discriminant is not a classifier; it only decides on an optimal projection to convert high-dimensional classification problem to 1D.

• A bias (threshold) is needed to form a linear classifier (multiple thresholds lead to nonlinear classifiers). The final classifier has the form

\[ y(x) = \text{sign}(w^T x + w_0) \]

where the nonlinear activation function \( \text{sign}(\cdot) \) is a step function

\[ \text{sign}(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases} \]

• How to decide the bias \( w_0 \)?
Perceptron
early theories of the brain
Biology and Learning

• Basic Idea
  - Good behavior should be rewarded, bad behavior punished (or not rewarded). This improves system fitness.
  - Killing a sabertooth tiger should be rewarded ...
  - Correlated events should be combined.
  - Pavlov’s salivating dog.

• Training mechanisms
  - Behavioral modification of individuals (learning) Successful behavior is rewarded (e.g. food).
  - Hard-coded behavior in the genes (instinct) The wrongly coded animal does not reproduce.
Neurons

- Soma (CPU)
  Cell body - combines signals

- Dendrite (input bus)
  Combines the inputs from several other nerve cells

- Synapse (interface)
  Interface and **parameter store** between neurons

- Axon (cable)
  May be up to 1m long and will transport the activation signal to neurons at different locations
Neurons

\[ f(x) = \sum_i w_i x_i = \langle w, x \rangle \]
Perceptron

- Weighted linear combination
- Nonlinear decision function
- Linear offset (bias)

- Linear separating hyperplanes (spam/ham, novel/typical, click/no click)

- Learning
  Estimating the parameters $w$ and $b$

\[ f(x) = \sigma \left( \langle w, x \rangle + b \right) \]
Perceptron
Perceptron

Rosenblatt

Widom
The Perceptron

initialize $w = 0$ and $b = 0$
repeat
  if $y_i [\langle w, x_i \rangle + b] \leq 0$ then
    $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$
  end if
until all classified correctly

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} y_i x_i$
- Classifier is linear combination of inner products $f(x) = \sum_{i \in I} y_i \langle x_i, x \rangle + b$
Convergence Theorem

- If there exists some $(w^*, b^*)$ with unit length and
  \[ y_i \left[ \langle x_i, w^* \rangle + b^* \right] \geq \rho \text{ for all } i \]
  then the perceptron converges to a linear separator after a number of steps bounded by
  \[ \left( b^{*2} + 1 \right) \left( r^2 + 1 \right) \rho^{-2} \text{ where } \|x_i\| \leq r \]

- Dimensionality independent
- Order independent (i.e. also worst case)
- Scales with ‘difficulty’ of problem
Consequences

• Only need to store errors. This gives a compression bound for perceptron.
• Stochastic gradient descent on hinge loss
  \[ l(x_i, y_i, w, b) = \max(0, 1 - y_i \langle w, x_i \rangle + b) \]
• Fails with noisy data

do NOT train your avatar with perceptrons
Hardness: margin vs. size

hard

easy
Concepts & version space

• Realizable concepts
  - Some function exists that can separate data and is included in the concept space
  - For perceptron - data is linearly separable

• Unrealizable concept
  - Data not separable
  - We don’t have a suitable function class (often hard to distinguish)
Minimum error separation

- XOR - not linearly separable
- Nonlinear separation is trivial
- Caveat (Minsky & Papert)

Finding the minimum error linear separator is NP hard (this killed Neural Networks in the 70s).
Nonlinear Features

• Regression
  We got nonlinear functions by preprocessing

• Perceptron
  - Map data into feature space $x \rightarrow \phi(x)$
  - Solve problem in this space
  - Query replace $\langle x, x' \rangle$ by $\langle \phi(x), \phi(x') \rangle$ for code

• Feature Perceptron
  - Solution in span of $\phi(x_i)$
Quadratic Features

- Separating surfaces are Circles, hyperbolae, parabolae
### Constructing Features
(very naive OCR system)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Loops</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>3 Joints</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>4 Joints</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>Angles</strong></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Ink</strong></td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Feature Engineering for Spam Filtering

- bag of words
- pairs of words
- date & time
- recipient path
- IP number
- sender
- encoding
- links
- ... secret sauce ...
More feature engineering

- Two Interlocking Spirals
  Transform the data into a radial and angular part
  \[ (x_1, x_2) = (r \sin \phi, r \cos \phi) \]
- Handwritten Japanese Character Recognition
  - Break down the images into strokes and recognize it
  - Lookup based on stroke order
- Medical Diagnosis
  - Physician’s comments
  - Blood status / ECG / height / weight / temperature ...
  - Medical knowledge
- Preprocessing
  - Zero mean, unit variance to fix scale issue (e.g. weight vs. income)
  - Probability integral transform (inverse CDF) as alternative
The Perceptron on features

- Nothing happens if classified correctly
- Weight vector is linear combination
- Classifier is linear combination of inner products

\[
\begin{align*}
\text{initialize } w, b &= 0 \\
\text{repeat} \\
\text{Pick } (x_i, y_i) \text{ from data} \\
\text{if } y_i(w \cdot \Phi(x_i) + b) \leq 0 \text{ then} \\
& \quad w' = w + y_i\Phi(x_i) \\
& \quad b' = b + y_i \\
\text{until } y_i(w \cdot \Phi(x_i) + b) > 0 \text{ for all } i
\end{align*}
\]

\[
\begin{align*}
\text{if } y_i(w \cdot \Phi(x_i) + b) \leq 0 \text{ then} \\
& \quad w' = w + y_i\Phi(x_i) \\
& \quad b' = b + y_i \\
\text{until } y_i(w \cdot \Phi(x_i) + b) > 0 \text{ for all } i
\end{align*}
\]
Problems

• Problems
  - Need domain expert (e.g. Chinese OCR)
  - Often expensive to compute
  - Difficult to transfer engineering knowledge

• Shotgun Solution
  - Compute many features
  - Hope that this contains good ones
  - Do this efficiently
Solving XOR

- XOR not linearly separable
- Mapping into 3 dimensions makes it easily solvable