Lecture 10:
- Linear Discriminant Functions
- Perceptron
Administrative

• **Project proposal** due today!
  - a half page description
  - problem to be investigated, why it is interesting, what data you will use, etc.
  - [http://goo.gl/forms/S5sRXJhKUI](http://goo.gl/forms/S5sRXJhKUI)
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):

$$1 \quad \frac{1}{1 + \exp(-z)}$$

Features can be discrete or continuous!
Naïve Bayes vs. Logistic Regression

- Representation equivalence
  - But only in a special case!!! (GNB with class-independent variances)
- But what’s the difference???
Naïve Bayes vs. Logistic Regression

- Representation equivalence
  - But only in a special case!!! (GNB with class-independent variances)
- But what’s the difference???
- LR makes no assumption about $P(X|Y)$ in learning!!
- Loss function!!!
  - Optimize different functions! Obtain different solutions
Naïve Bayes vs. Logistic Regression

Consider Y Boolean, $X_i$ continuous $X=<X_1 \ldots X_d>$

Number of parameters:
- NB: $4d+1$ $\pi$, $(\mu_{1,y}, \mu_{2,y}, \ldots, \mu_{d,y})$, $(\sigma^2_{1,y}, \sigma^2_{2,y}, \ldots, \sigma^2_{d,y})$ $y=0,1$
- LR: $d+1$ $w_0, w_1, \ldots, w_d$

Estimation method:
- NB parameter estimates are uncoupled
- LR parameter estimates are coupled
Generative vs. Discriminative

Given infinite data (asymptotically),

If conditional independence assumption holds, Discriminative and generative NB perform similar.

\[ \epsilon_{\text{Dis}, \infty} \sim \epsilon_{\text{Gen}, \infty} \]

If conditional independence assumption does NOT hold, Discriminative outperforms generative NB.

\[ \epsilon_{\text{Dis}, \infty} < \epsilon_{\text{Gen}, \infty} \]
Generative vs. Discriminative

[Ng & Jordan, NIPS 2001]

Given finite data \((n \text{ data points, } d \text{ features})\),

\[
\begin{align*}
\epsilon_{\text{Dis},n} & \leq \epsilon_{\text{Dis},\infty} + O \left( \sqrt{\frac{d}{n}} \right) \\
\epsilon_{\text{Gen},n} & \leq \epsilon_{\text{Gen},\infty} + O \left( \sqrt{\frac{\log d}{n}} \right)
\end{align*}
\]

Naïve Bayes (generative) requires \(n = O(\log d)\) to converge to its asymptotic error, whereas Logistic regression (discriminative) requires \(n = O(d)\).

Why? “Independent class conditional densities”

- parameter estimates not coupled – each parameter is learnt independently, not jointly, from training data.
Naïve Bayes vs. Logistic Regression

Verdict

Both learn a linear decision boundary. Naïve Bayes makes more restrictive assumptions and has higher asymptotic error, but converges faster to its less accurate asymptotic error.
**Experimental Comparison** *(Ng-Jordan’01)*

UCI Machine Learning Repository 15 datasets, 8 continuous features, 7 discrete features

- Pima (continuous)
- Adult (continuous)
- Boston (predict if > median price, continuous)
- Optdigits (0’s and 1’s, continuous)
- Optdigits (2’s and 3’s, continuous)
- Ionosphere (continuous)

More in the paper...
What you should know

• 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
Today

- Linear Discriminant Functions
  - Two Classes
  - Multiple Classes
  - Fisher’s Linear Discriminant

- Perceptron
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 $\text{sign}(\cdot)$ 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
  \[ x = x_\perp + r \frac{w}{\|w\|} \]

where \( x_\perp \) is the projection \( x \) on the decision surface. Then

\[
\begin{align*}
  w^T x &= w^T x_\perp + r \frac{w^T w}{\|w\|} \\
  w^T x + w_0 &= w^T x_\perp + w_0 + r\|w\| \\
  y(x) &= r\|w\| \\
  r &= \frac{y(x)}{\|w\|}
\end{align*}
\]

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

\[
y(x) = \tilde{w}^T \tilde{x}
\]
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.

\[
\begin{align*}
\mathcal{R}_1 & \quad \mathcal{R}_2 \\
\not C_1 & \quad \not C_2 \\
C_1 & \quad C_2
\end{align*}
\]

\[
\begin{align*}
\mathcal{R}_1 & \quad \mathcal{R}_2 \\
\mathcal{R}_3 & \quad \mathcal{R}_3 \\
C_1 & \quad C_2 \\
C_3 & \quad C_3
\end{align*}
\]
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

$$C(x) = k, \text{ if } y_k(x) > y_j(x) \quad \forall j \neq k$$

• The decision boundary between class $C_k$ and $C_j$ is given by $y_k(x) = y_j(x)$

$$(w_k - w_j)^T x + (w_{k0} - w_{j0}) = 0$$
Property of the Decision Regions

Theorem

The decision regions of the $K$-class discriminant $y_k(x) = w_k^T x + w_{k0}$ are singly connected and convex.

Proof.

Suppose two points $x_A$ and $x_B$ both lie inside decision region $R_k$. Any point $\hat{x}$ on the line between $x_A$ and $x_B$ can be expressed as $\hat{x} = x_A + (1 - \lambda) x_B$. So $y_k(\hat{x}) = y_k(x_A) + (1 - \lambda) y_k(x_B) > y_j(x_A) + (1 - \lambda) y_j(x_B)$ for $j \neq k$. Therefore, the regions $R_k$ are singly connected and convex.
Theorem

The decision regions of the K-class discriminant \( y_k(x) = w_k^Tx + w_{k0} \) are singly connected and convex.

Proof.

Suppose two points \( x_A \) and \( x_B \) both lie inside decision region \( R_k \). Any point \( \hat{x} \) on the line between \( x_A \) and \( x_B \) can be expressed as

\[
\hat{x} = \lambda x_A + (1 - \lambda) x_B
\]

So

\[
y_k(\hat{x}) = \lambda y_k(x_A) + (1 - \lambda) y_k(x_B) \\
> \lambda y_j(x_A) + (1 - \lambda) y_j(x_B) \quad (\forall j \neq k) \\
= y_j(\hat{x}) \quad (\forall j \neq k)
\]

Therefore, the regions \( R_k \) is single connected and convex.
Property of the Decision Regions

**Theorem**

The decision regions of the $K$-class discriminant $y_k(x) = w_k^T x + w_{k0}$ are singly connected and convex.

If two points $x_A$ and $x_B$ both lie inside the same decision region $R_k$, then any point $x$ that lies on the line connecting these two points must also lie in $R_k$, and hence the decision region must be singly connected and convex.
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

where

\[
w^T S_W w
\]

\[
S_W = \sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T
\]
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

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w})\mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w})\mathbf{S}_B \mathbf{w}$$

$$(\mathbf{w}^T \mathbf{S}_B \mathbf{w})\mathbf{S}_W \mathbf{w} = (\mathbf{w}^T \mathbf{S}_W \mathbf{w})(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w}$$

- 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 \( w_0 \)?
Given a set of labeled samples \( \{x_n, t_n\}_{n=1}^{N} \) where \( x_n \in \mathbb{R} \) and \( t_n \in \{-1, 1\} \). Without loss of generality, assume that \( x_n \)'s are sorted, namely \( x_n \leq x_{n+1}, \forall n < N \). Define an accumulative function \( F(\cdot) \):

\[
F(x) = \sum_{n=1}^{N} t_n \delta(x_n \leq x)
\]

The optimal classifier (suppose the sign is correct)

\[
y(x) = \text{sign}(x - w_0)
\]

the minimizes the training error

\[
- \sum_{n=1}^{N} y(x_n) t_n
\]

is decided by

\[
w_0 = \arg \max F(x)
\]
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

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

Ham

Spam

slide by Alex Smola
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

\[
y_i \left[ \langle x_i, w^* \rangle + b^* \right] \geq \rho \quad \text{for all } i
\]

then the perceptron converges to a linear separator after a number of steps bounded by

\[
\left( b^{*2} + 1 \right) \left( r^2 + 1 \right) \rho^{-2} \quad \text{where } \|x_i\| \leq r
\]

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

Starting Point
We start from $w_1 = 0$ and $b_1 = 0$.

Step 1: Bound on the increase of alignment
Denote by $w_i$ the value of $w$ at step $i$ (analogously $b_i$).

Alignment: $\langle (w_i, b_i), (w^*, b^*) \rangle$

For error in observation $(x_i, y_i)$ we get

$$\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle$$

$$= \langle [(w_j, b_j) + y_i(x_i, 1)], (w^*, b^*) \rangle$$

$$= \langle (w_j, b_j), (w^*, b^*) \rangle + y_i \langle (x_i, 1) \cdot (w^*, b^*) \rangle$$

$$\geq \langle (w_j, b_j), (w^*, b^*) \rangle + \rho$$

$$\geq j \rho.$$

Alignment increases with number of errors.
Proof

Step 2: Cauchy-Schwartz for the Dot Product

\[
\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle \leq \| (w_{j+1}, b_{j+1}) \| \| (w^*, b^*) \| = \sqrt{1 + (b^*)^2} \| (w_{j+1}, b_{j+1}) \|
\]

Step 3: Upper Bound on \( \|(w_j, b_j)\| \)

If we make a mistake we have

\[
\| (w_{j+1}, b_{j+1}) \|^2 = \| (w_j, b_j) + y_i(x_i, 1) \|^2 = \| (w_j, b_j) \|^2 + 2y_i \langle (x_i, 1), (w_j, b_j) \rangle + \| (x_i, 1) \|^2 \\
\leq \| (w_j, b_j) \|^2 + \| (x_i, 1) \|^2 \\
\leq j(R^2 + 1).
\]

Step 4: Combination of first three steps

\[
 j \rho \leq \sqrt{1 + (b^*)^2} \| (w_{j+1}, b_{j+1}) \| \leq \sqrt{j(R^2 + 1)((b^*)^2 + 1)}
\]

Solving for \( j \) proves the theorem.
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

\[ \text{do NOT train your avatar with perceptrons} \]
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>Loops</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>3 Joints</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>4 Joints</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>Angles</td>
<td>0</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>
<td>0</td>
</tr>
<tr>
<td>Ink</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

initialize $w, b = 0$
repeat
Pick $(x_i, y_i)$ from data
if $y_i(w \cdot \Phi(x_i) + b) \leq 0$ then
    $w' = w + y_i \Phi(x_i)$
    $b' = b + y_i$
until $y_i(w \cdot \Phi(x_i) + b) > 0$ for all $i$

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