Linear Classifiers
Outline

- Framework
- “Exact”
  - Minimize Mistakes (Perceptron Training)
  - Matrix inversion
- “Logistic Regression” Model
  - Max Likelihood Estimation (MLE) of \( P( y \mid x ) \)
  - Gradient descent (MSE; MLE)
- “Linear Discriminant Analysis”
  - Max Likelihood Estimation (MLE) of \( P( y, x ) \)
  - Direct Computation
Diagnosing Butterfly-itis

Hmmm... perhaps Butterfly-it is??
Classifier: Decision Boundaries

- **Classifier**: partitions input space $X$ into “decision regions”
- **Linear threshold unit** has a linear decision boundary
- **Defn**: Set of points that can be separated by linear decision boundary is “linearly separable"
Linear Separators

- Draw “separating line”

- If \( \#\text{antennae} \leq 2 \), then butterfly-itis

- So \(?\) is Not butterfly-itis.
Can be “angled”…

\[ 2.3 \times \#w + 7.5 \times \#a + 1.2 > 0 \]

If \[ 2.3 \times \text{Wings} + 7.5 \times \text{antennae} + 1.2 \] > 0 then butterfly-itis
Linear Separators, in General

- Given data (many features)

<table>
<thead>
<tr>
<th></th>
<th>F_1</th>
<th>F_2</th>
<th>...</th>
<th>F_n</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35</td>
<td>95</td>
<td>...</td>
<td>3</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>22</td>
<td>80</td>
<td>...</td>
<td>-2</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>50</td>
<td>...</td>
<td>1.9</td>
<td>No</td>
</tr>
</tbody>
</table>

- find “weights” \{w_1, w_2, \ldots, w_n, w_0\} such that

\[
w_1 \times F_1 + \ldots + w_n \times F_n + w_0 > 0
\]

means \textbf{Class = Yes}
Linear Separator

Just view $F_0 = 0$, so $w_0$ ...
Linear Separator

- Performance
  - Given \( \{w_i\} \), and values for instance, compute response

- Learning
  - Given labeled data, find “correct” \( \{w_i\} \)

- Linear Threshold Unit … “Perceptron”
Linear Separators – Facts

- GOOD NEWS:
  - If data is linearly separated,
  - Then FAST ALGORITHM finds correct \{w_i\} !
- But…
Linear Separators – Facts

GOOD NEWS:

- If data is linearly separated,
- Then FAST ALGORITHM finds correct \( \{w_i\} \)!

But…

Some “data sets” are NOT linearly separable!
Geometric View

- Consider 3 training examples:
  - Want classifier that looks like...
Linear Equation is Hyperplane

Equation $\mathbf{w} \cdot \mathbf{x} = \sum_i w_i \cdot x_i$ is plane

$$y(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases}$$
Linear Threshold Unit: “Perceptron”

- Squashing function:
  \[ \text{sgn}: \mathbb{R} \rightarrow \{-1, +1\} \]

\[
\text{sgn}(r) = \begin{cases} 
1 & \text{if } r > 0 \\ 
0 & \text{otherwise}
\end{cases}
\]

(“heaviside”)

- Actually \( \mathbf{w} \cdot \mathbf{x} > b \) but...

Create extra input \( x_0 \) fixed at 1

Corresponding \( w_0 \) corresponds to \(-b\)
Learning Perceptrons

- Can represent Linearly-Separated surface... any hyper-plane between two half-spaces...

- Remarkable learning algorithm: [Rosenblatt 1960]
  
  If function $f$ can be represented by perceptron, then $\exists$ learning alg guaranteed to quickly converge to $f$!

$\Rightarrow$ enormous popularity, early / mid 60's

- But some simple fns cannot be represented (Boolean XOR) [Minsky/Papert 1969]

- Killed the field temporarily!
Perceptron Learning

- Hypothesis space is...
  - **Fixed Size:** \( \exists O(2^{n^2}) \) distinct perceptrons over \( n \) boolean features
  - **Deterministic**
  - **Continuous Parameters**

- Learning algorithm:
  - Various: **Local** search, **Direct** computation, . . .
  - **Eager**
  - **Online / Batch**
Task

- **Input:** labeled data

  Transformed to

- **Output:** \( w \in \mathbb{R}^{r+1} \)

**Goal:** Want\( w \) s.t.

\[ \forall i \quad \text{sgn}(w \cdot [1, \ x^{(i)}]) = y^{(i)} \]

\[ \ldots \text{minimize mistakes wrt data} \ldots \]
Error Function

Given data \{ [x^{(i)}, y^{(i)}] \}_{i=1..m}, optimize...

1. Classification error
   Perceptron Training; Matrix Inversion

2. Mean-squared error
   Matrix Inversion; Gradient Descent

3. (Log) Conditional Probability
   MSE Gradient Descent; LCL Gradient Descent

4. (Log) Joint Probability
   Direct Computation

\[
\text{err}_{\text{Class}}(w) = \frac{1}{m} \sum_{i=1}^{m} I[y^{(i)} \neq o_w(x^{(i)})]
\]

\[
\text{err}_{\text{MSE}}(w) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} [y^{(i)} - o_w(x^{(i)})]^2
\]

\[
\text{LCL}(w) = \frac{1}{m} \sum_{i=1}^{m} \log P_w(y^{(i)}|x^{(i)})
\]

\[
\text{LL}(w) = \frac{1}{m} \sum_{i=1}^{m} \log P_w(y^{(i)}, x^{(i)})
\]
#1: Optimal Classification Error

- For each labeled instance \([x, y]\)
  \[\text{Err} = y - o_w(x)\]
  
  \(y = f(x)\) is target value
  \(o_w(x) = \text{sgn}(w \cdot x)\) is perceptron output

- **Idea**: Move weights in appropriate direction, to push \(\text{Err} \to 0\)

- If \(\text{Err} > 0\) (error on POSITIVE example)
  - need to increase \(\text{sgn}(w \cdot x)\)
  - need to increase \(w \cdot x\)
  - Input \(j\) contributes \(w_j \cdot x_j\) to \(w \cdot x\)
    - if \(x_j > 0\), increasing \(w_j\) will increase \(w \cdot x\)
    - if \(x_j < 0\), decreasing \(w_j\) will increase \(w \cdot x\)
  
  \[w_j \leftarrow w_j + x_j\]

- If \(\text{Err} < 0\) (error on NEGATIVE example)
  \[w_j \leftarrow w_j - x_j\]
#1a: Mistake Bound Perceptron Alg

Initialize \( \mathbf{w} = 0 \)
Do until bored
    Predict “+” iff \( \mathbf{w} \cdot \mathbf{x} > 0 \)  
    else “−”
    Mistake on positive: \( \mathbf{w} \leftarrow \mathbf{w} + \mathbf{x} \)
    Mistake on negative: \( \mathbf{w} \leftarrow \mathbf{w} - \mathbf{x} \)

<table>
<thead>
<tr>
<th>Action Instance</th>
<th>Weights</th>
<th>Orig Data</th>
<th>Data + “( x_0 = 1 )”</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>[0 0 0]</td>
<td>( \langle x_1 \ x_2 \rangle )</td>
<td>( c(x) )</td>
</tr>
<tr>
<td>( i_1 )</td>
<td></td>
<td>0 0</td>
<td>+</td>
</tr>
<tr>
<td>( i_2 )</td>
<td></td>
<td>1 0</td>
<td>−</td>
</tr>
<tr>
<td>( i_3 )</td>
<td></td>
<td>1 1</td>
<td>+</td>
</tr>
</tbody>
</table>
Mistake Bound Theorem

Theorem: [Rosenblatt 1960]
If data is consistent w/some linear threshold \( w \), then number of mistakes is \( \leq (1/\Delta)^2 \),
where \( \Delta = \min_{x} \frac{|w \cdot x|}{|w| \times |x|} \)

- \( \Delta \) measures “wiggle room” available:
  - If \( |x| = 1 \), then \( \Delta \) is max, over all consistent planes, of minimum distance of example to that plane
  - \( w \) is \( \perp \) to separator, as \( w \cdot x = 0 \) at boundary
  - So \( |w \cdot x| \) is projection of \( x \) onto plane, PERPENDICULAR to boundary line
    ... ie, is distance from \( x \) to that line (once normalized)
Proof of Convergence

For simplicity:
0. Use \( x_0 = 1 \), so target plane goes thru 0
1. Assume target plane doesn’t hit any examples
2. Replace negative point \( \langle x_0, x_1, \ldots, x_n \rangle 0 \) by positive point \( \langle -x_0, -x_1, \ldots, -x_n \rangle 1 \)
3. Normalize all examples to have length 1

- Let \( w^* \) be unit vector rep'ning target plane
  \( \Delta = \min_x \{ w^* \cdot x \} \)
  Let \( w \) be hypothesis plane

- Consider:
  \( \frac{(w \cdot w^*)}{|w|} \)

- On each mistake, add \( x \) to \( w \)

\[
\begin{align*}
\text{w} &= \sum_{\{x \mid x \cdot w < 0\}} x
\end{align*}
\]

\( x \) wrong wrt \( w \) iff \( w \cdot x < 0 \)
Proof (con't)

If $w$ is mistake...

Numerator increases by $x \cdot w^* \geq \Delta$
(denominator)$^2$ becomes

$$(w + x)^2 = w^2 + x^2 + 2(w \cdot x) < w^2 + 1$$
as $w \cdot x < 0$

As initially $w = \langle 0, \ldots, 0 \rangle$.

After $m$ mistakes,

numerator is $\geq m \times \Delta$
(denominator)$^2$ is $\leq 0 + \underbrace{1 + \ldots + 1}_m = m$

so denominator $\leq \sqrt{m}$

- As $(w \cdot w^*)/|w| = \cos(\text{angle between } w \text{ and } w^*)$
it must be $\leq 1$, so
numerator $\leq$ denominator

\[ \Rightarrow \quad \Delta \times m \leq \sqrt{m} \quad \Rightarrow \quad m \leq \frac{1}{\Delta^2} \]
For each labeled instance \( [x, y] \)
\[
\text{Err}( [x, y] ) = y - o_w(x) \in \{ -1, 0, +1 \}
\]

- If \( \text{Err}( [x, y] ) = 0 \)  \text{Correct!}  \ldots \text{Do nothing!}  \\
  \[ \Delta w = 0 \equiv \text{Err}( [x, y] ) \cdot x \]

- If \( \text{Err}( [x, y] ) = +1 \)  \text{Mistake on positive!}  \text{Increment by}  \ +x  \\
  \[ \Delta w = +x \equiv \text{Err}( [x, y] ) \cdot x \]

- If \( \text{Err}( [x, y] ) = -1 \)  \text{Mistake on negative!}  \text{Increment by}  \ -x  \\
  \[ \Delta w = -x \equiv \text{Err}( [x, y] ) \cdot x \]

In all cases...
\[
\Delta w^{(i)} = \text{Err}( [x^{(i)}, y^{(i)}] ) \cdot x^{(i)} = [y^{(i)} - o_w(x^{(i)})] \cdot x^{(i)}
\]

**Batch**
\[
\Delta w_j = \sum_i \Delta w_j^{(i)}
= \sum_i x_j^{(i)} (y^{(i)} - o_w(x^{(i)}))
\]
\[
w_j += \eta \Delta w_j
\]
\( \eta \) \text{ is learning rate (small pos “constant”} \ldots \approx 0.05? \)
0. New $\mathbf{w}$
1. For each row $i$, compute
   a. $\Delta \mathbf{w} = 0$
   b. $E^{(i)} = y^{(i)} - o_w(x^{(i)})$
   c. $\Delta \mathbf{w} +\! = E^{(i)} x^{(i)}$
      
      [ ... $\Delta w_j +\! = E^{(i)} x^{(i)}_j$ ... ]

2. Increment $\mathbf{w} +\! = \eta \Delta \mathbf{w}$
Correctness

- Rule is intuitive: **Climbs in correct direction. . .**

- Thrm: Converges to correct answer, if . . .
  - training data is linearly separable
  - sufficiently small $\eta$

- Proof: Weight space has **EXACTLY 1 minimum!** (no non-global minima)
  $\Rightarrow$ with enough examples, finds correct function!

- Explains early popularity

- If $\eta$ too large, may overshoot
  If $\eta$ too small, takes too long

- So often $\eta = \eta(k)$ … which decays with # of iterations, $k$
#1c: Matrix Version?

Task: Given \( \{ (x^i, y^i) \} \)
\[ y^i \in \{-1, +1\} \] is label

Find \( w \) s.t.
\[
\begin{align*}
y^1 & = w_0 + w_1 x^1_1 + \cdots + w_n x_n^1 \\
y^2 & = w_0 + w_1 x^2_1 + \cdots + w_n x_n^2 \\
\vdots \\
y^m & = w_0 + w_1 x^m_1 + \cdots + w_n x_n^m \\
\end{align*}
\]

- Linear Equalities: \( y = Xw \)

\[
X = \begin{pmatrix}
1 & x^1_1 & \cdots & x^1_n \\
1 & x^2_1 & \cdots & x^2_n \\
\vdots & \vdots & \ddots & \vdots \\
1 & x^m_1 & \cdots & x^m_n
\end{pmatrix}
\]

\[
w = [w_0, w_1, \ldots, w_n]^\top
\]

- Solution: \( w = X^{-1}y \)
Issues

1. Why restrict to only \( y^i \in \{ -1, +1 \} \) ?
   - If from discrete set \( y^i \in \{ 0, 1, \ldots, m \} \): General (non-binary) classification
   - If ARBITRARY \( y^i \in \mathbb{R} \): Regression

2. What if NO \( \mathbf{w} \) works?
   ...\( \mathbf{X} \) is singular; overconstrained ...
   Could try to minimize residual

\[
\sum_i I[ y^{(i)} \neq \mathbf{w} \cdot \mathbf{x}^{(i)} ]
\]
\[
\| \mathbf{y} - \mathbf{X} \mathbf{w} \|_1 = \sum_i | y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} |
\]
\[
\| \mathbf{y} - \mathbf{X} \mathbf{w} \|_2 = \sum_i ( y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} )^2
\]

NP-Hard! \quad \Rightarrow \quad \text{Easy!}
L₂ error vs 0/1-Loss

- “0/1 Loss function” not smooth, differentiable
- MSE error is smooth, differentiable… and is overbound…
Gradient Descent for Perceptron?

- Why not Gradient Descent for THRESHOLDed perceptron?
- Needs gradient (derivative), not

Gradient Descent is General approach. Requires
  + continuously parameterized hypothesis
  + error must be differentiable wrt parameters
But . .
  – can be slow (many iterations)
  – may only find LOCAL opt
#1. LMS version of Classifier

- **View as Regression**
  - Find “best” linear mapping $\mathbf{w}$ from $\mathbf{X}$ to $\mathbf{Y}$
    - $\mathbf{w}^* = \text{argmin} \ Err_{\text{LMS}}^{(X,Y)}(\mathbf{w})$
    - $Err_{\text{LMS}}^{(X,Y)}(\mathbf{w}) = \sum_i ( y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} )^2$
  - Threshold: if $\mathbf{w} \cdot \mathbf{x} > 0.5$, return 1; else 0

- See Chapter 3
Use Linear Regression for Classification?

1. Use regression to find weights $w$
2. Classify new instance $x$ as $\text{sgn}(w \cdot x)$

- But ... regression minimizes sum of squared errors on target function
- ... which gives strong influence to outliers
#3: Logistic Regression

- Want to compute $P_w(y=1|\ x)$
  ... based on parameters $w$
- But …
  - $w \cdot x$ has range $[-\infty, \infty]$
  - probability must be in range $\in [0; 1]$
- Need “squashing” function $[-\infty, \infty] \rightarrow [0, 1]$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
Alternative Derivation…

\[ P(+y|x) = \frac{P(x|+y)P(+y)}{P(x|+y)P(+y) + P(x|-y)P(-y)} \]

\[ = \frac{1}{1 + \exp(-a)} \]

\[ a = \ln \frac{P(x|+y)P(+y)}{P(x|-y)P(-y)} \]
Logistic Regression (con’t)

Assume 2 classes:

\[ P_w(y=1|x) = \sigma(w \cdot x) = \frac{1}{1 + e^{-(x \cdot w)}} \]
\[ P_w(y=-1|x) = 1 - \frac{1}{1 + e^{-(x \cdot w)}} = \frac{e^{-(x \cdot w)}}{1 + e^{-(x \cdot w)}} \]

Log Odds:

\[ \log \frac{P_w(y=1|x)}{P_w(y=-1|x)} = x \cdot w \]
How to learn parameters $w$?

… depends on goal?

- A: Minimize MSE?
  \[ \sum_i (y^{(i)} - o_w(x^{(i)}))^2 \]

- B: Maximize likelihood?
  \[ \sum_i \log P_w(y^{(i)} | x^{(i)}) \]
MSError Gradient for Sigmoid Unit

Error: \[ \sum_j ( y^{(j)} - o_w(x^{(j)}) )^2 = \sum_j E^{(j)} \]

For single training instance

Input: \( x^{(j)} = [x^{(j)}_1, \ldots, x^{(j)}_k] \)

Computed Output: \( o^{(j)} = \sigma( \sum_i x^{(j)}_i \cdot w_i ) = \sigma( Z^{(j)} ) \)

\[ \text{where } Z^{(j)} = \sum_i x^{(j)}_i \cdot w_i \text{ using current } \{w_i\} \]

Correct output: \( y^{(j)} \)

Stochastic Error Gradient:

\[
\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \left[ \frac{1}{2} (o - y)^2 \right] = \frac{1}{2} \left[ 2(o - y) \frac{\partial}{\partial w_i} (o - y) \right] \]

\[
= (o - y) \left( \frac{\partial o}{\partial w_i} \right) = (o - y) \frac{\partial \sigma(z)}{\partial z} \frac{\partial z}{\partial w_i} \]

\[
\sigma(z) = \frac{1}{1 + e^{-z}} \]
Derivative of Sigmoid

\[
\frac{d}{da} \sigma(a) = \frac{d}{da} \frac{1}{1+e^{-a}}
\]

\[
= \frac{-1}{(1+e^{-a})^2} \frac{d}{da} (1+e^{-a}) = \frac{-1}{(1+e^{-a})^2}(-e^{-a})
\]

\[
= \frac{e^{-a}}{(1+e^{-a})^2} = \frac{1}{(1+e^{-a})(1+e^{-a})} = \sigma(a) [1-\sigma(a)]
\]
Updating LR Weights (MSE)

- \[
\frac{\partial E}{\partial w_i} = (o - y) \frac{\partial \sigma(z)}{\partial z} \frac{\partial z}{\partial w_i}
\]

- Using:
  \[
  \frac{\partial \sigma(z)}{\partial z} = \sigma(z) (1 - \sigma(z)) = o(1-o)
  \]
  \[
  \frac{\partial z}{\partial w_i} = \frac{\partial (\sum_i w_i \cdot x_i)}{\partial w_i} = x_i
  \]

⇒
\[
\frac{\partial E(j)}{\partial w_i} = (o^{(j)} - y^{(j)}) o^{(j)} (1 - o^{(j)}) x_i^{(j)}
\]

Note: As already computed \( o^{(j)} = \sigma(z^{(j)}) \) to get answer, trivial to compute \( \sigma'(z^{(j)}) = \sigma(z^{(j)})(1 - \sigma(z^{(j)})) \)!

- Update \( w_i \) += \( \Delta w_i \) where

\[
\Delta w_i = \eta \cdot \frac{\partial E(j)}{\partial w_i}
\]
B: Or... Learn Conditional Probability

As fitting probability distribution, better to return probability distribution (≈ \( w \)) that is most likely, given training data, \( S \)

Goal: \( w^* = \arg \max_w \frac{P(S|w)P(w)}{P(S)} \)

= \( \arg \max_w P(S|w)P(w) \)

= \( \arg \max_w \log P(S|w) \)

Bayes Rules

As \( P(S) \) does not depend on \( w \)

As \( P(w) \) is uniform

As log is monotonic
ML Estimation

- $P(S | w) \equiv$ likelihood function

$$L(w) = \log P(S | w)$$

- $w^* = \text{argmax}_w L(w)$

is “maximum likelihood estimator” (MLE)
Computing the Likelihood

- As training examples \([x^{(i)}, y^{(i)}]\) are iid
  - drawn independently from same (unknown) prob \(P_w(x, y)\)

\[
\log P(S \mid w) = \log \prod_i P_w(x^{(i)}, y^{(i)})
\]

\[
= \sum_i \log P_w(x^{(i)}, y^{(i)})
\]

\[
= \sum_i \log P_w(y^{(i)} \mid x^{(i)}) + \sum_i \log P_w(x^{(i)})
\]

- Here \(P_w(x^{(i)}) = 1/n \ldots \)
  - not dependent on \(w\), over empirical sample \(S\)

\[w^* = \arg\max_w \sum_i \log P_w(y^{(i)} \mid x^{(i)})\]
Fit Logistic Regression…
by Gradient Ascent

- Want $w^* = \arg \max_w J(w)$

- $J(w) = \sum_i r(y^{(i)}, x^{(i)}, w)$

- For $y \in \{0, 1\}$

$$r(y, x, w) = \log P_w(\ y | \ x) = y \log(P_w(\ y=1 | \ x)) + (1 - y) \log(1 - P_w(\ y=1 | \ x))$$

- So climb along…

$$\frac{\partial J(w)}{\partial w_j} = \sum_i \frac{\partial r(y^{(i)}, x^{(i)}, w)}{\partial w_j}$$
Gradient Descent ...

\[ \frac{\partial r(y, x, w)}{\partial w_j} = \frac{\partial}{\partial w_j} \left[ y \log(p_1) + (1-y)\log(1-p_1) \right] = \frac{y}{p_1} \frac{\partial p_1}{\partial w_j} + (-1) \times \frac{1-y}{1-p_1} \frac{\partial p_1}{\partial w_j} = \frac{y-p_1}{p_1(1-p_1)} \frac{\partial p_1}{\partial w_j} \]

\[ \frac{\partial p_1}{\partial w_j} = \frac{\partial P_w(y=1|x)}{\partial w_j} = \frac{\partial}{\partial w_j} (\sigma(x \cdot w)) = \sigma(x \cdot w)[1-\sigma(x \cdot w)] \frac{\partial}{\partial w_j} (x \cdot w) = p_1(1-p_1) \cdot x^{(i)}_j \]

\[ \frac{\partial J(w)}{\partial w_j} = \sum_i \frac{\partial r(y^{(i)}, x^{(i)}, w)}{\partial w_j} = \sum_i \frac{y^{(i)}-p_1}{p_1(1-p_1)} p_1(1-p_1) \cdot x^{(i)}_j \]

\[ = \sum_i (y^{(i)}-P_w(y=1|x)) \cdot x^{(i)}_j \]
Gradient Ascent for Logistic Regression (MLE)

Given: training examples \( \langle x^{(i)}, y^{(i)} \rangle, \ i = 1..N \)
Set initial weight vector \( w = \langle 0, 0, 0, 0, \ldots, 0 \rangle \)
Repeat until convergence

Let gradient vector \( \Delta w = \langle 0, 0, 0, 0, \ldots, 0 \rangle \)
For \( i = 1 \) to \( N \) do
\[ p_1^{(i)} = \frac{1}{1 + \exp[w \cdot x^{(i)}]} \]
error\(_i\) = \( y^{(i)} - p_1^{(i)} \)
For \( j = 1 \) to \( n \) do
\[ \Delta w_j \ += \ \text{error}_i \cdot x_{ij} \]
\( w \ += \ \eta \ \Delta w \) % step in direction of increasing gradient
Comments on MLE Algorithm

- This is BATCH;
  - obvious online alg (stochastic gradient ascent)
- Can use second-order (Newton-Raphson) alg for faster convergence
  - weighted least squares computation; aka “Iteratively-Reweighted Least Squares” (IRLS)
Use Logistic Regression for Classification

Return YES iff

\[
\frac{P(y=1|x)}{P(y=0|x)} > 0
\]

\[
\ln \frac{P(y=1|x)}{P(y=0|x)} = \ln \frac{1}{\exp(-w \cdot x)/(1 + \exp(-w \cdot x))} = w \cdot x > 0
\]

\[P(y=0|x) = \frac{1}{1 + \exp(-w \cdot x)} = \frac{1}{\exp(w \cdot x)} = w \cdot x > 0
\]

Logistic Regression learns a LTU!
Logistic Regression for $K > 2$ Classes

- To handle $K > 2$ classes
  - Let class $K$ be "reference"
  - Represent each other class $k \neq K$ as logistic function of odds of class $k$ versus class $K$:

  - Apply gradient ascent to learn all $w_k$ weight vectors, in parallel.

  Conditional probabilities:
  \[
P(y = k \mid x) = \frac{\exp(w_k \cdot x)}{1 + \sum_{\ell=1}^{K-1} \exp(w_\ell \cdot x)}
  \]
  and
  \[
P(y = K \mid x) = \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(w_\ell \cdot x)}
  \]

  \[
  \begin{align*}
  \log \frac{P(y = 1 \mid x)}{P(y = K \mid x)} &= w_1 \cdot x \\
  \log \frac{P(y = 2 \mid x)}{P(y = K \mid x)} &= w_2 \cdot x \\
  \vdots & \quad \vdots \\
  \log \frac{P(y = K - 1 \mid x)}{P(y = K \mid x)} &= w_{K-1} \cdot x
  \end{align*}
  \]

  Note: $k-1$ different $w_i$ weights, each of dimension $|x|$
Learning LR Weights

Task: Given data \( \{ (x^{(i)}, y^{(i)}) \} \)

find \( w \) in \( p_w(y|x) = \begin{cases} \frac{1}{1 + \exp(-w \cdot x)} & \text{if } y=1 \\ \frac{\exp(-w \cdot x)}{1 + \exp(-w \cdot x)} & \text{if } y=0 \end{cases} \)

s.t. \( p_w(y^{(i)}|x^{(i)}) > \frac{1}{2} \) \( \iff y^{(i)} = 1 \)

Approach 1: MSE — “Neural nets”

Minimize \( \sum_i (o^{(i)} - y^{(i)})^2 \)

Gradient:

\[ \Delta w^{(i)}_j = (o^{(i)} - y^{(i)}) o^{(i)} (1 - o^{(i)}) \]

Approach 2: MLE — “Logistic Regression”

Maximize \( \sum_i p_w(y|x) \)

Gradient:

\[ \Delta w^{(i)}_j = (y^{(i)} - p(1|x^{(i)})) x^{(i)}_j \]
(MaxProb)

0. New $\mathbf{w}$

1. For each row $i$, compute
   a. $\Delta \mathbf{w} = \mathbf{0}$
   b. $E^{(i)} = y^{(i)} (y^{(i)} - p(1|x^{(i)}))$
   c. $\Delta \mathbf{w} += E^{(i)} \mathbf{x}^{(i)}$
      [ ... $\Delta w_j += E^{(i)} x^{(i)}_j$ ... ]

2. Increment $\mathbf{w} += \eta \Delta \mathbf{w}$
Logistic Regression Algs for LTUs

- Learns Conditional Probability Distribution $P( y \mid x )$

- **Local Search:**
  Begin with initial weight vector; iteratively modify to maximize objective function log likelihood of the data (ie, seek $w$ s.t. probability distribution $P_w( y \mid x )$ is most likely given data.)

- **Eager:** Classifier constructed from training examples, which can then be discarded.

- **Online or batch**
LDA learns joint distribution $P(y, x)$

- As $P(y, x) \neq P(y | x)$; optimizing $P(y, x)$ does not equal optimizing $P(y | x)$

“generative model”
- $P(y, x)$ model of how data is generated
- Eg, factor
  
  $P(y, x) = P(y) P(x | y)$

- $P(y)$ generates value for $y$; then
- $P(x | y)$ generates value for $x$ given this $y$

Belief net:
Linear Discriminant Analysis, con't

- \( P( y, x ) = P( y ) P( x \mid y ) \)
- \( P( y ) \) is a simple discrete distribution
  - Eg: \( P( y = 0 ) = 0.31; P( y = 1 ) = 0.69 \)
    (31% negative examples; 69% positive examples)
- Assume \( P( x \mid y ) \) is multivariate normal, with mean \( \mu_k \) and covariance \( \Sigma \)

\[
P( x \mid y = k ) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp \left[ -\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k) \right]
\]
Estimating LDA Model

- Linear discriminant analysis assumes form

\[
P(x,y) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu_y)^\top \Sigma^{-1} (x - \mu_y) \right]
\]

- \( \mu_y \) is mean for examples belonging to class \( y \);
- covariance matrix \( \Sigma \) is shared by all classes!

- Can estimate LDA directly:

\[
m_k = \# \text{training examples in class } y = k
\]

- Estimate of \( P(y = k) \): \( p_k = m_k / m \)

\[
\hat{\mu}_k = \frac{1}{m} \sum_{i: y_i = k} x_i
\]

\[
\hat{\Sigma} = \frac{1}{m} \sum_i \frac{x_i - \hat{\mu}_y}{i} (x_i - \hat{\mu}_y)^\top
\]

(Subtract each \( x_i \) from corresponding \( x \) before taking outer product)
Example of Estimation

- $m = 7$ examples; 
  $m_+ = 3$ positive; $m_- = 4$ negative
  $\Rightarrow \quad p_+ = \frac{3}{7} \quad p_- = \frac{4}{7}$

- Compute $\hat{\mu}_i$ over each class

  $\hat{\mu}_+ = \frac{1}{3} \sum_{i: \langle y^{(i)} \rangle = +} x^{(i)}$

  $= \frac{1}{3} \begin{pmatrix} 13.1, 20.2, 0.4 \end{pmatrix} + \begin{pmatrix} 6.0, 17.7, -4.2 \end{pmatrix} + \begin{pmatrix} 8.2, 18.2, -2.5 \end{pmatrix}$

  $= [9.1, 18.7, -2.1]$

  $\hat{\mu}_- = \frac{1}{4} \sum_{i: \langle y^{(i)} \rangle = -} x^{(i)} = [-1.8, 12.0, 16.3]$
Estimation...

- Compute common $\hat{\Sigma}$

- "Normalize" each $z := x - \mu_y(x)$

$z^{(1)} := [13.1, 20.2, 0.4] - [9.1, 18.7, -2.1]$
$\qquad = [4.0, 1.5, -1.7]$

$\ldots$

$z^{(4)} := [0.4, 10.1, 19.2] - [-1.8, 12.0, 16.3]$
$\quad = [2.2, -1.9, 2.9]$

$\ldots z^{(7)} := \ldots$

- Compute covariance matrix, for each $i$:

For $x^{(1)}$, via $z^{(1)}$:

$z^{(1)} \times z^{(1)^\top} = \begin{bmatrix} 4.0 \\ 0.5 \\ -1.7 \end{bmatrix} \cdot \begin{bmatrix} 4.0 \\ 0.5 \\ -1.7 \end{bmatrix}$

$\quad = \begin{bmatrix} 16.0 & 2.0 & -6.8 \\ 2.0 & 0.25 & -0.85 \\ -6.8 & -0.85 & 2.89 \end{bmatrix}$

$\quad = \frac{1}{m} \sum_i z^{(i)}z^{(i)^\top}$
Classifying, Using LDA

- How to classify new instance, given estimates

Eg, \( \hat{p}_+ = 3/7 \quad \hat{p}_- = 4/7 \)

\[
\begin{align*}
\hat{\mu}_+ &= [9.1, 18.7, -2.1] \\
\hat{\mu}_- &= [-1.8, 12.0, 16.3] \\
\hat{\Sigma} &= \begin{bmatrix}
7.22 & -1.31 & 6.35 \\
-1.31 & 2.91 & 0.32 \\
6.35 & 0.32 & 26.03
\end{bmatrix}
\end{align*}
\]

- Class for instance \( \mathbf{x} = [5, 14, 6] \)?

\[
P(y = +, \mathbf{x} = [5, 14, 6]) = \frac{3}{7} \times P(\mathbf{x} = [5, 14, 6] | y = +) \\
= \frac{3}{7} \times \frac{1}{(2\pi)^{3/2} |\hat{\Sigma}|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{x} - \hat{\mu}_+)^\top \hat{\Sigma}^{-1} (\mathbf{x} - \hat{\mu}_+) \right] \\
= 16.63 \times 10^{-11}
\]

\[
P(y = -, \mathbf{x} = [5, 14, 6]) = \frac{4}{7} \times P(\mathbf{x} = [5, 14, 6] | y = -) \\
= \frac{4}{7} \times \frac{1}{(2\pi)^{3/2} |\hat{\Sigma}|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{x} - \hat{\mu}_-)^\top \hat{\Sigma}^{-1} (\mathbf{x} - \hat{\mu}_-) \right] \\
= 43.33 \times 10^{-11}
\]

\[
P(y = + | [5, 14, 6]) = \frac{P(y = +, [5, 14, 6])}{P(y = +, [5, 14, 6]) + P(y = -, [5, 14, 6])} = 0.2774
\]

\[
P(y = - | [5, 14, 6]) = 0.7226
\]
LDA learns an LTU

- Consider 2-class case with a 0/1 loss function
- Classify $\hat{y} = 1$ if

\[
\log \frac{P(y = 1| x)}{P(y = 0| x)} > 0 \quad \text{iff} \quad \log \frac{P(y = 1, x)}{P(y = 0, x)} > 0
\]

\[
\frac{P(x, y = 1)}{P(x, y = 0)} = \frac{P(y = 1)}{P(y = 0)} \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp \left[ -\frac{1}{2} (x - \mu_1)^T \Sigma^{-1} (x - \mu_1) \right] \frac{P(y = 1)}{P(y = 0)} \exp \left[ -\frac{1}{2} (x - \mu_0)^T \Sigma^{-1} (x - \mu_0) \right]
\]

\[
\ln \frac{P(x, y = 1)}{P(x, y = 0)} = \ln \frac{P(y = 1)}{P(y = 0)} - \frac{1}{2} [(x - \mu_1)^T \Sigma^{-1} (x - \mu_1) - (x - \mu_0)^T \Sigma^{-1} (x - \mu_0)]
\]
LDA Learns an LTU (2)

- \((x-\mu_1)^T \Sigma^{-1} (x-\mu_1) - (x-\mu_0)^T \Sigma^{-1} (x-\mu_0)\)
  
  \[
  = x^T \Sigma^{-1} (\mu_0 - \mu_1) + (\mu_0 - \mu_1)^T \Sigma^{-1} x + \mu_1^T \Sigma^{-1} \mu_1 - \mu_0^T \Sigma^{-1} \mu_0
  \]

- As \(\Sigma^{-1}\) is symmetric,

\[
\ln \frac{P(x, y = 1)}{P(x, y = 0)} = \ln \frac{P(y = 1)}{P(y = 0)} - \frac{1}{2} [(x - \mu_1)^T \Sigma^{-1} (x - \mu_1) - (x - \mu_0)^T \Sigma^{-1} (x - \mu_0)]
\]

\[
= x^T \Sigma^{-1} (\mu_1 - \mu_0) + \ln \frac{P(y = 1)}{P(y = 0)} + \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1
\]
LDA Learns an LTU (3)

\[
\ln \frac{P(x, y = 1)}{P(x, y = 0)} = x^\top \Sigma^{-1}(\mu_1 - \mu_0) + \ln \frac{P(y=1)}{P(y=0)} + \frac{1}{2} \mu_0^\top \Sigma^{-1} \mu_0 - \frac{1}{2} \mu_1^\top \Sigma^{-1} \mu_1
\]

- So let...

\[
w = \Sigma^{-1}(\mu_1 - \mu_0)
\]

\[
c = \ln \frac{P(y=1)}{P(y=0)} + \frac{1}{2} \mu_0^\top \Sigma^{-1} \mu_0 - \frac{1}{2} \mu_1^\top \Sigma^{-1} \mu_1
\]

- Classify \( \hat{y} = 1 \) iff \( w \cdot x + c > 0 \)

LTU!!
## Variants of LDA

- Covariance matrix $\Sigma$
- $n$ features; $k$ classes

<table>
<thead>
<tr>
<th>Same for all classes?</th>
<th>Diagonal</th>
<th>#param’s</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>$k$</td>
<td>LDA</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>$n^2$</td>
<td>Naïve Gaussian Classifier</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>$k , n^2$</td>
<td>General Gaussian Classifier</td>
</tr>
</tbody>
</table>
Generalizations of LDA

- **General Gaussian Classifier**
  Allow each class $k$ to have its own $\Sigma_k$
  $\Rightarrow$ Classifier $\equiv$ quadratic threshold unit (not LTU)

- **Naïve Gaussian Classifier**
  Allow each class $k$ to have its own $\Sigma_k$
  but require each $\Sigma_k$ be diagonal.
  $\Rightarrow$ within each class,
  any pair of features $x_i$ and $x_j$ are independent
  $\Box$ Classifier is still quadratic threshold unit
  but with a restricted form
Summary of Linear Discriminant Analysis

- Learns Joint Probability Distr'n $P(y, x)$
- **Direct Computation.**
  MLEstimate of $P(y, x)$ computed directly from data without search.
  But need to invert matrix, which is $O(n^3)$
- **Eager:**
  Classifier constructed from training examples, which can then be discarded.
- **Batch:** Only a batch algorithm.
  An online LDA alg requires online alg for incrementally updating $\Sigma^{-1}$
  [Easy if $\Sigma^{-1}$ is diagonal. . . ]
Two Geometric Views of LDA
View 1: Mahalanobis Distance

- Squared Mahalanobis distance between $\mathbf{x}$ and $\mu$
  \[
  D_M^2(\mathbf{x}, \mu) = (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)
  \]

  $\Sigma^{-1} \approx$ linear distortion
  
  … converts standard Euclidean distance into Mahalanobis distance.

- LDA classifies $\mathbf{x}$ as 0 if
  \[D_M^2(\mathbf{x}, \mu_0) < D_M^2(\mathbf{x}, \mu_1)\]

- \[
  \log P(\mathbf{x} \mid y = k) \approx \log \pi_k - \frac{1}{2} D_M^2(\mathbf{x}, \mu_k)
  \]
View 2: Most Informative Low Dimensional Projection

- LDA
  - Finds \( K-1 \) dim hyperplane (\( K = \) number of classes)
  - Project \( \mathbf{x} \) and \( \{ \mu_k \} \) to that hyperplane
  - Classify \( \mathbf{x} \) as nearest \( \mu_k \) within hyperplane

- Goal: Hyperplane that maximally separates projection of \( \mathbf{x} \)'s wrt \( \sum^{-1} \)

\[ \mathbf{w} \approx \text{Fisher's Linear Discriminant} \]
Fisher Linear Discriminant

- Recall any vector $\mathbf{w}$ projects $\mathbb{R}^n \rightarrow \mathbb{R}$
- Goal: Want $\mathbf{w}$ that “separates” classes
  - Each $\mathbf{w} \cdot \mathbf{x}^+$ far from each $\mathbf{w} \cdot \mathbf{x}^-$

- Using $\mathbf{m}_+ = \frac{\sum_i y^{(i)} \cdot x^{(i)}}{\sum_i y^{(i)}}$ $\mathbf{m}_- = \frac{\sum_i (1-y^{(i)}) \cdot x^{(i)}}{\sum_i (1-y^{(i)})}$

- Mean of $\mathbf{x}$'s projections:
  - $m_+ = \frac{\sum_i y^{(i)} \mathbf{w}^\top \cdot x^{(i)}}{\sum_i y^{(i)}} = \mathbf{w}^\top \cdot \mathbf{m}_+$
  - $m_- = \frac{\sum_i (1-y^{(i)}) \mathbf{w}^\top \cdot x^{(i)}}{\sum_i (1-y^{(i)})} = \mathbf{w}^\top \cdot \mathbf{m}_-$

- Perhaps project on $\mathbf{m}_+ - \mathbf{m}_-$?
- Still overlap… why?
Fisher Linear Discriminant

- Using $\mathbf{m}_+ = \frac{\sum_i y^{(i)} \cdot x^{(i)}}{\sum_i y^{(i)}}$ and $\mathbf{m}_- = \frac{\sum_i (1-y^{(i)}) \cdot x^{(i)}}{\sum_i (1-y^{(i)})}$

- Mean of $x$'s projections:
  - $\mathbf{m}_+ = \frac{\sum_i y^{(i)} \mathbf{w}^\top \cdot x^{(i)}}{\sum_i y^{(i)}} = \mathbf{w}^\top \cdot \mathbf{m}_+$
  - $\mathbf{m}_- = \frac{\sum_i (1-y^{(i)}) \mathbf{w}^\top \cdot x^{(i)}}{\sum_i (1-y^{(i)})} = \mathbf{w}^\top \cdot \mathbf{m}_-$

- Problem with $\mathbf{m}_+ - \mathbf{m}_-$:
  - Doesn’t consider “scatter” within class
  - Goal: Want $\mathbf{w}$ that “separates” classes

  - Each $\mathbf{w} \cdot x^+$ far from each $\mathbf{w} \cdot x^-$
  - Positive $x^+$'s: $\mathbf{w} \cdot x^+$ close to each other
  - Negative $x^-$'s: $\mathbf{w} \cdot x^-$ close to each other

- “scatter” of +instance; –instance
  - $\mathbf{s}_+^2 = \sum_i y^{(i)} (\mathbf{w} \cdot x^{(i)} - \mathbf{m}_+)^2$
  - $\mathbf{s}_-^2 = \sum_i (1 - y^{(i)}) (\mathbf{w} \cdot x^{(i)} - \mathbf{m}_+)^2$
Fisher Linear Discriminant

- Recall any vector \( \mathbf{w} \) projects \( \mathbb{R}^n \rightarrow \mathbb{R} \)

- Goal: Want \( \mathbf{w} \) that “separates” classes
  - Positive \( \mathbf{x}^+ \)'s: \( \mathbf{w} \cdot \mathbf{x}^+ \) close to each other
  - Negative \( \mathbf{x}^- \)'s: \( \mathbf{w} \cdot \mathbf{x}^- \) close to each other
  - Each \( \mathbf{w} \cdot \mathbf{x}^+ \) far from each \( \mathbf{w} \cdot \mathbf{x}^- \)

- Using \( \mathbf{m}_+ = \frac{\sum_i y^{(i)} \cdot \mathbf{x}^{(i)}}{\sum_i y^{(i)}} \quad \mathbf{m}_- = \frac{\sum_i (1-y^{(i)}) \cdot \mathbf{x}^{(i)}}{\sum_i (1-y^{(i)})} \)

  \[ m_+ = \frac{\sum_i y^{(i)} \cdot \mathbf{w}^T \cdot \mathbf{x}^{(i)}}{\sum_i y^{(i)}} = \mathbf{w}^T \cdot \mathbf{m}_+ \]

  \[ m_- = \frac{\sum_i (1-y^{(i)}) \cdot \mathbf{w}^T \cdot \mathbf{x}^{(i)}}{\sum_i (1-y^{(i)})} = \mathbf{w}^T \cdot \mathbf{m}_- \]

- “scatter” of +instance; –instance
  - \( \mathbf{s}_+^2 = \sum_i y^{(i)} \left( \mathbf{w} \cdot \mathbf{x}^{(i)} - \mathbf{m}_+ \right)^2 \)
  - \( \mathbf{s}_-^2 = \sum_i (1 - y^{(i)}) \left( \mathbf{w} \cdot \mathbf{x}^{(i)} - \mathbf{m}_+ \right)^2 \)
FLD, con't

- Separate means \( m_- \) and \( m_+ \)
  \( \Rightarrow \) maximize \( (m_- - m_+)^2 \)
- Minimize each spread \( s_+^2, s_-^2 \)
  \( \Rightarrow \) maximize \( (s_+^2 + s_-^2) \)
- Objective function: maximize

\[
\begin{align*}
#1: (m_- - m_+)^2 &= (\mathbf{w}^\top \mathbf{m}_+ - \mathbf{w}^\top \mathbf{m}_-)^2 \\
&= \mathbf{w}^\top (\mathbf{m}_+ - \mathbf{m}_-)(\mathbf{m}_+ - \mathbf{m}_-)^\top \mathbf{w} = \mathbf{w}^\top \mathbf{S}_B \mathbf{w} \\
&= \text{“between-class scatter”} \quad \mathbf{S}_B = (\mathbf{m}_+ - \mathbf{m}_-)(\mathbf{m}_+ - \mathbf{m}_-)^\top
\end{align*}
\]

\[
J_S(\mathbf{w}) = \frac{(m_+ - m_-)^2}{(s_+^2 + s_-^2)}
\]
FLD, III

- $s_+^2 = \sum_i y^{(i)} (\mathbf{w} \cdot \mathbf{x}^{(i)} - m_+)^2$
  
  $= \sum_i \mathbf{w}^\mathsf{T} y^{(i)} (\mathbf{x}^{(i)} - m_+) (\mathbf{x}^{(i)} - m_+) \mathbf{w}$

  $= \mathbf{w}^\mathsf{T} \mathbf{S}_+ \mathbf{w}$

$\mathbf{S}_+ = \sum_i y^{(i)} (\mathbf{x}^{(i)} - m_+) (\mathbf{x}^{(i)} - m_+) \mathbf{w}$

... “within-class scatter matrix” for +

- $\mathbf{S}_- = \sum_i (1 - y^{(i)}) (\mathbf{x}^{(i)} - m_-) (\mathbf{x}^{(i)} - m_-) \mathbf{w}$

... “within-class scatter matrix” for −

- $\mathbf{S}_w = \mathbf{S}_+ + \mathbf{S}_-$

so $s_+^2 + s_-^2 = \mathbf{w}^\mathsf{T} \mathbf{S}_w \mathbf{w}$

$$J_S(\mathbf{w}) = \frac{(m_+ - m_-)^2}{(s^2_+ + s^2_-)}$$
FLD, IV

$$J_S(w) = \frac{(m_+ - m_-)^2}{(s_+^2 + s_-^2)} = \frac{w^T S_B w}{w^T S_w w} = \frac{(w^T (m_1 - m_2))^2}{w^T S_w w}$$

Solving \( \frac{\partial J_S(w)}{\partial w_j} = 0 \) \( \Rightarrow \)

\( w = \alpha S_B^{-1}(m_+ - m_-) \)
FLD, V

\[ J_S(w) = \frac{(m_+ - m_-)^2}{s_+^2 + s_-^2} = \frac{w^T S_B w}{w^T S_w w} = \frac{(w^T (m_1 - m_2))^2}{w^T S_w w} \]

Solving \[ \frac{\partial J_S(w)}{\partial w_j} = 0 \Rightarrow w = \alpha S_B^{-1}(m_+ - m_-) \]

When \[ P(x \mid y_i) \sim N(\mu_i; \Sigma) \]
\[ \exists \text{LINEAR DISCRIMINANT: } w = \sum^{-1}(\mu_+ - \mu_-) \]

⇒ FLD is optimal classifier, if classes normally distributed

Can use even if not normal:
After projecting d-dim to 1, just use any classification method

Analogous derivation for \( K > 2 \) classes
Comparing LMS, Logistic Regression, LDA

- Which is best: LMS, LR, LDA?
- Ongoing debate within machine learning community about relative merits of
  - direct classifiers \([ LMS ]\)
  - conditional models \(P(y|x)\) \([ LR ]\)
  - generative models \(P(y, x)\) \([ LDA ]\)
- Stay tuned...
Issues in Debate

- **Statistical efficiency**
  If generative model \( P( y, x ) \) is correct, then ... usually gives better accuracy, particularly if training sample is small.

- **Computational efficiency**
  Generative models typically easiest to compute (LDA computed directly, without iteration)

- **Robustness to changing loss functions**
  LMS must re-train the classifier when the loss function changes. ... no retraining for generative and conditional models

- **Robustness to model assumptions.**
  Generative model usually performs poorly when the assumptions are violated.
  Eg, LDA works poorly if \( P( x | y ) \) is non-Gaussian.
  Logistic Regression is more robust, ... LMS is even more robust

- **Robustness to missing values and noise.**
  In many applications, some of the features \( x_{ij} \) may be missing or corrupted for some of the training examples.
  Generative models typically provide better ways of handling this than non-generative models.
Other Algorithms for learning LTUs

- **Naive Bayes** [Discuss later]
  For $K = 2$ classes, produces LTU

- **Winnow** [Discuss later?]
  Can handle large numbers of “irrelevant” features
  - (features whose weights should be zero)