This week

- Logistic Regression
- Discriminative vs. Generative Classification
- Linear Discriminant Functions
  - Two Classes
  - Multiple Classes
  - Fisher’s Linear Discriminant
- Perceptron

Recap: Naïve Bayes

- NB Assumption:
  \[ P(X_1 \ldots X_d|Y) = \prod_{i=1}^{d} P(X_i|Y) \]

- NB Classifier:
  \[ f_{NB}(x) = \arg \max_y \prod_{i=1}^{d} P(x_i|y)P(y) \]

- Assume parametric form for \( P(X_i|Y) \) and \( P(Y) \)
  - Estimate parameters using MLE/MAP and plug in
Gaussian Naïve Bayes (GNB)

- There are several distributions that can lead to a linear boundary.
- As an example, consider Gaussian Naïve Bayes:
  \[ Y \sim \text{Bernoulli}(\pi) \]
  \[ P(X_i|Y = y) = \frac{1}{\sqrt{2\pi \sigma_i^2}} e^{-\frac{(X_i - \mu_{i,y})^2}{2\sigma_i^2}} \]

Gaussian class conditional densities

- What if we assume variance is independent of class, i.e. \( \sigma_{i,0}^2 = \sigma_{i,1}^2 \)

GNB with equal variance is a Linear Classifier!

\[ P(X_i|Y = y) = \frac{1}{\sqrt{2\pi \sigma_i^2}} e^{-\frac{(X_i - \mu_{i,y})^2}{2\sigma_i^2}} \]

Decision boundary:

\[ \prod_{i=1}^{d} P(X_i|Y = 0) P(Y = 0) = \prod_{i=1}^{d} P(X_i|Y = 1) P(Y = 1) \]

\[ \log P(Y = 0) \prod_{i=1}^{d} P(X_i|Y = 0) = \log P(Y = 1) \prod_{i=1}^{d} P(X_i|Y = 1) = \log \frac{1 - \pi}{\pi} + \sum_{i=1}^{d} \log \frac{P(X_i|Y = 0)}{P(X_i|Y = 1)} \]

\[ = \log \frac{1 - \pi}{\pi} + \sum_{i=1}^{d} \frac{\mu_{i,0}^2 - \mu_{i,1}^2}{2\sigma_i^2} + \sum_{i=1}^{d} \frac{\mu_{i,0} - \mu_{i,1}}{\sigma_i^2} X_i =: w_0 + \sum_{i} w_i X_i \]

Constant term First-order term

Generative vs. Discriminative Classifiers

- Generative classifiers (e.g. Naïve Bayes)
  - Assume some functional form for \( P(X|Y) \) (or \( P(X|Y) \) and \( P(Y) \))
  - Estimate parameters of \( P(X|Y), P(Y) \) directly from training data
- But \( \text{arg max}_Y P(X|Y) P(Y) = \text{arg max}_Y P(Y|X) \)
- Why not learn \( P(Y|X) \) directly? Or better yet, why not learn the decision boundary directly?

- Discriminative classifiers (e.g. Logistic Regression)
  - Assume some functional form for \( P(Y|X) \) or for the decision boundary
  - Estimate parameters of \( P(Y|X) \) directly from training data
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 a linear function of the data

Logistic function (or Sigmoid):

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

Features can be discrete or continuous!

Logistic Regression is a Linear Classifier!

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

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

Decision boundary:

\[
w_0 + \sum_i w_i X_i = 0
\]

(Logarithmic Decision Boundary)

Logistic Regression is a Linear Classifier!

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

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

for \( k < K \)

\[
P(Y = y_k|X) = \frac{\exp(w_{k0} + \sum_{i=1}^d w_{ki} X_i)}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^d w_{ji} X_i)}
\]

for \( k = K \) (normalization, so no weights for this class)

\[
P(Y = y_K|X) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(w_{j0} + \sum_{i=1}^d w_{ji} X_i)}
\]

Logistic Regression for more than 2 classes

• Logistic regression in more general case, where \( Y \in \{y_1, \ldots, y_K\} \)
Training Logistic Regression

We’ll focus on binary classification:

\[
P(Y = 0|X, w) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}
\]

\[
P(Y = 1|X, w) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}
\]

How to learn the parameters \( w_0, w_1, \ldots w_d \)?

Training Data \( \{(X^{(j)}, Y^{(j)})\}_{j=1}^{n} \)

\( X^{(j)} = (X^{(j)}_1, \ldots, X^{(j)}_d) \)

Maximum Likelihood Estimates

\[
\hat{w}_{MLE} = \arg \max_w \prod_{j=1}^{n} P(X^{(j)}, Y^{(j)} | w)
\]

But there is a problem …

Don’t have a model for \( P(X) \) or \( P(X|Y) \) — only for \( P(Y|X) \)

Expressing Conditional log Likelihood

\[
l(W) = \sum_{l} Y_l \ln P(Y^l = 1|X^l, W) + (1 - Y^l) \ln P(Y^l = 0|X^l, W)
\]

\[
P(Y = 0|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}
\]

\[
P(Y = 1|X) = \frac{\exp(w_0 + \sum_{i=1}^{n} w_i X_i)}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i X_i)}
\]

\( Y \) can take only values 0 or 1, so only one of the two terms in the expression will be non-zero for any given \( Y^l \)

Training Logistic Regression

How to learn the parameters \( w_0, w_1, \ldots w_d \)?

Training Data \( \{(X^{(j)}, Y^{(j)})\}_{j=1}^{n} \)

\( X^{(j)} = (X^{(j)}_1, \ldots, X^{(j)}_d) \)

Maximum (Conditional) Likelihood Estimates

\[
\hat{w}_{MLE} = \arg \max_w \prod_{j=1}^{n} P(Y^{(j)} | X^{(j)}, w)
\]

Discriminative philosophy – Don’t waste effort learning \( P(X) \), focus on \( P(Y|X) \) – that’s all that matters for classification!

Expressing Conditional log Likelihood

\[
l(W) = \sum_{l} Y^l \ln P(Y^l = 1|X^l, W) + (1 - Y^l) \ln P(Y^l = 0|X^l, W)
\]

\[
= \sum_{l} Y^l \ln \frac{P(Y^l = 1|X^l, W)}{P(Y^l = 0|X^l, W)} + \ln P(Y^l = 0|X^l, W)
\]

\[
= \sum_{l} Y^l (w_0 + \sum_{i} w_i X^l_i) - \ln(1 + \exp(w_0 + \sum_{i} w_i X^l_i))
\]
Maximizing Conditional log Likelihood

\[ \max_w l(w) \equiv \ln \prod_j P(y^j|x^j, w) \]
\[ = \sum_j y^j(w_0 + \sum_i w_ix_i^j) - \ln(1 + \exp(w_0 + \sum_i w_ix_i^j)) \]

**Bad news:** no closed-form solution to maximize \(l(w)\)

**Good news:** \(l(w)\) is concave function of \(w\) \(\Rightarrow\) concave functions easy to optimize (unique maximum)

---

**Optimizing concave/convex functions**

- Conditional likelihood for Logistic Regression is concave
- Maximum of a concave function = minimum of a convex function

**Gradient Ascent (concave)/ Gradient Descent (convex)**

Gradient:
\[ \nabla_w l(w) = \left[ \frac{\partial l(w)}{\partial w_0}, \ldots, \frac{\partial l(w)}{\partial w_n} \right]^t \]

Update rule:
\[ \Delta w = \eta \nabla_w l(w) \]
\[ w_i^{(t+1)} = w_i^{(t)} + \eta \frac{\partial l(w)}{\partial w_i} \]

**Effect of step-size \(\eta\)**

- Large \(\eta\) \(\Rightarrow\) Fast convergence but larger residual error
  - Also possible oscillations
- Small \(\eta\) \(\Rightarrow\) Slow convergence but small residual error

**Gradient Ascent for Logistic Regression**

Gradient ascent algorithm: iterate until change < \(\epsilon\)
\[ w_0^{(t+1)} = w_0^{(t)} + \eta \sum_j [y^j - \hat{P}(Y^j = 1 | x^j, w^{(t)})] \]

For \(i=1,\ldots,d\),
\[ w_i^{(t+1)} = w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 | x^j, w^{(t)})] \]

repeat

- Gradient ascent is simplest of optimization approaches
  - e.g., Newton method, Conjugate gradient ascent, IRLS (see Bishop 4.3.3)
Gaussian 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

Consider Y boolean, X_i continuous, X=<X_1 ... X_d>

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

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 holds, Discriminative outperforms generative NB.

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

Given finite data (n data points, d features), [Ng & Jordan, NIPS 2001]

\[ \epsilon_{\text{Dis}, n} \leq \epsilon_{\text{Dis}, \infty} + O \left( \sqrt{\frac{d}{n}} \right) \]

\[ \epsilon_{\text{Gen}, n} \leq \epsilon_{\text{Gen}, \infty} + O \left( \sqrt{\frac{\log d}{n}} \right) \]

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.

Experimental Comparison (Ng-Jordan’01)

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

![Graphs showing error rates for Naïve Bayes and Logistic Regression with various datasets](image)

More in the paper...

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.

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
This week

- Logistic Regression
- Discriminative vs. Generative Classification
- Linear Discriminant Functions
  - Two Classes
  - Multiple Classes
  - Fisher’s Linear Discriminant
- Perceptron

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 \( \mathbf{w}_0 \).
- 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

\[ \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} = -\frac{\mathbf{w}_0}{\|\mathbf{w}\|} \]

- So \( \mathbf{w}_0 \) determines the location of the decision surface.

Linear Discriminant Function

- Linear discriminant function for a vector \( \mathbf{x} \)
  \[ y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \mathbf{w}_0 \]
  where \( \mathbf{w} \) is called weight vector, and \( \mathbf{w}_0 \) is a bias.

- The classification function is
  \[ C(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + \mathbf{w}_0) \]
  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

- 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
  \[ \mathbf{w}^T \mathbf{x} = \mathbf{w}^T \mathbf{x}_\perp + r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} \]
  \[ \mathbf{w}^T \mathbf{x} + \mathbf{w}_0 = \mathbf{w}^T \mathbf{x}_\perp + \mathbf{w}_0 + r \frac{\mathbf{w}^T \mathbf{w}}{\|\mathbf{w}\|} \]
  \[ y(\mathbf{x}) = r \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\|} \]
  \[ r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|} \]

- Simpler notion: define \( \tilde{\mathbf{w}} = (\mathbf{w}_0, \mathbf{w}) \) 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.

**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

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

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.

**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.

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

**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**

- **Pursue the optimal linear projection on which the two classes can be maximally separated**
  \[y = w^T x\]
  A way to view a linear classification model is in terms of dimensionality reduction.

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

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

  - **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 \( \nabla J(w) = 0 \), we obtain**
    \[w^T S_B w w = (w^T S_W w) S_B w\]
    \[w^T S_B w w = (w^T S_W w)(m_2 - m_1)(m_2 - m_1)^T w\]

  - **Terms** \( w^T S_B w \), \( w^T S_W w \) and \((m_2 - m_1)^T w\) are scalars, and we only care about directions. So the scalars are dropped. Therefore
    \[w \propto S_W^{-1}(m_2 - m_1)\]
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 \( w_0 \)?

This week

- Logistic Regression
- Discriminative vs. Generative Classification
- Linear Discriminant Functions
  - Two Classes
  - Multiple Classes
- Perceptron

1D Linear Classifier

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

Ham

Spam

The Perceptron

initialize \( w = 0 \) and \( b = 0 \)
repeat
  if \( y_i [(w, x_i) + b] \leq 0 \) 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
  \( y_i [(x_i, w^*) + b^*] \geq \rho \) for all \( i \)
  then the perceptron converges to a linear separator after a number of steps bounded by
  \[ (b^*^2 + 1) (\rho^2 + 1) \rho^{-2} \] 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: $\langle w_i, b_i \rangle = \langle w^{*}, b^{*}\rangle$
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
$\langle (w_{j+1}, b_{j+1}) \cdot (w^{*}, b^{*}) \rangle \leq \|(w_{j+1}, b_{j+1})\| \| (w^{*}, b^{*}) \|$
$= \sqrt{1 + (b^{*})^2} \|(w_{j+1}, b_{j+1})\|$\n
Step 3: Upper Bound on $\|(w_{i}, b_{i})\|$.
If we make a mistake we have
$\|(w_{j+1}, b_{j+1})\|^2 \leq \|(w_j, b_j) + y_i(x_i, 1)\|^2$
$\leq \|(w_j, b_j)\|^2 + 2y_i \langle (x_i, 1), (w_j, b_j) \rangle + \|(x_i, 1)\|^2$
$\leq \|(w_j, b_j)\|^2 + \|(x_i, 1)\|^2$
$\leq j(R^2 + 1)$.

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
  $l(x_i, y_i, w, b) = \max(0, 1 - y_i [(w, x_i) + b])$
- Fails with noisy data

Hardness: margin vs. size

do NOT train your avatar with perceptrons
**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>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>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</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>
</tr>
<tr>
<td>4 Joints</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>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>
</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>
</tr>
</tbody>
</table>

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

Feature Engineering for Spam Filtering

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

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
- 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

Next week

• Support Vector Machines
• Multi-class classification
• Kernels