Idea 1: Support Vector Machine (SVM)

Classifier boundary: a linear combination of the support vectors:

$$a_1 D(x,V_1) + a_2 D(x,V_2) + a_3 D(x,V_3) = 0$$

$$\sum a_i D(x,V_i) = 0 \quad \text{(weighted sum of distances/similarities to } V_i \text{'s)}$$
Idea 2: Kernel Methods

- SVMs use nonlinear kernels, often mapping to a much higher dimensional space
- Non-separable examples may be separable in the new space
- Kernel function depends only on the dot product between two examples
  - Polynomial
  - RBF (Radial Basis Function) or Gaussian
- Can capture some interactions among features
  - Linear in the high dimensional space
  - Non-linear in the original feature space
  - Moderate cost (related to original low-dimension)
- Some sequence problems are exceptions and use *Linear* Kernels:
  - Natural language processing (NLP)
  - Protein or DNA alignment in bioinformatics
- Many applications of nonlinear kernels beyond SVMs
Mercer’s Condition / Representer Theorem

- The hypothesis space is represented efficiently by using some of the training examples – the support vectors
- Kernel defines a similarity metric (opposite of distance)
- The desired hyperplane can be represented as

\[ \sum_{i=1}^{m} \alpha_i K(s_i, x) \]

Linear weighted sum of similarities to support vectors
- Find the support vectors, \( s' \)s and the weights \( \alpha' \)s
  - Quadratic programming problem (potentially expensive)
  - But solution to a constrained convex optimization problem
- Regularize by preferring a large margin
Kernel Function

• \( K(x,y) \) where \( x \) & \( y \) are examples
• Defined w/ dot product of inputs
• RKHS: reproducing kernel Hilbert space, dot product space, inner product space
• Examples:
  
  \[
  \begin{align*}
  x \cdot y & \quad \text{linear or string kernel (NLP, DNA, proteins...)} \\
  (x \cdot y)^2 & \quad \text{homogeneous quadratic kernel} \\
  (x \cdot y + 1)^2 & \quad \text{nonhomogeneous quadratic kernel} \\
  (x \cdot y)^3 & \quad \text{homogeneous cubic kernel} \\
  \exp(-[(y-x) \cdot (y-x)] / 2\sigma^2) & \quad \text{Gaussian kernel (perhaps most common)}
  \end{align*}
  \]
• Similarity information between the feature vectors
Example

• Homogeneous Quadratic Kernel:  \((x \cdot y)^2\)

• Suppose
  \[ x = <a, b, c> \quad y = <e, f, g> \]
  Need to support a dot product
  Efficient, even if many more attributes

• So \(K(x, y) = (ae + bf + cg)^2\)

• The sum is just a scalar which is then squared...

• But consider it symbolically
Example

• $K(x,y) = (ae + bf + cg)^2$

• $(ae + bf + cg)^2 = (ae + bf + cg) \cdot (ae + bf + cg)$
  
  \[
  = (ae \cdot (ae + bf + cg)) + (bf \cdot (ae + bf + cg)) + (cg \cdot (ae + bf + cg))
  \]

  \[
  = (ae)^2 + (bf)^2 + (cg)^2 + 2aebf + 2aecg + 2bfcg
  \]

• kernel evaluates to a scalar

• But this scalar summarizes all two-way feature interactions (ae w/ bf; ae w/ cg; bf w/ cg)
Example

• $K(x, y) = (x \cdot y)^2$ summarizes all two-way feature interactions
• $K(x, y) = (x \cdot y)^3$ summarizes all three-way feature interactions, ETC.
• $K(x, y) = (x \cdot y + 1)^3$
  – Complete (non-homogeneous) polynomial
  – Summarizes all two- and all three-way interactions
• We pay a low-dimensional cost for a high-dimensional similarity measure
Example

• RBF kernel
  – Based on vector difference
  – Probably most commonly used SVM kernel
• $K(x, y) = \exp\left[ - \frac{[(y-x) \cdot (y-x)]}{2\sigma^2} \right]$
  – Symmetric Gaussian distribution
  – Need to choose $\sigma^2$ (via cross validation)
• Infinite dimensional (due to exponentiation)
• Separator is
  – Linear in the high-dimensional inflated space
  – Linear in the dual kernel space
  – Non-linear in the original feature space
Handwritten Seven’s vs. Two’s and Eight’s

High-dimensional linear boundary mapped back to the original feature space

Two’s

Eight’s

Seven’s

Handwritten 32 x 32 gray scale pixels

Linear in input feature space performs poorly

But linearly separable in the inflated space
Mercer Kernels

Support vector $s$, unknown test example $x$, scalar $c$

$(s \cdot x)^d \quad \text{Homogeneous polynomials}$

$(s \cdot x + 1)^d \quad \text{Complete polynomials}$

$\exp\left(-\frac{(s - x) \cdot (s - x)}{2 \, \sigma^2}\right) \quad \text{Gaussian / RBF}$

If $K$ and $k$ are kernels, so are

$K + k$

$c \cdot K$

$K + c$

$K \cdot k$
SVM Summary

• Support Vectors
  – Most constraining training examples

• Kernel trick
  – Benefit of a high dimension
  – Cost of a low dimension
  – Constrained high-dimensional mapping (dot-product)

• Maximum Margin
  – Use least-expressive separating hyperplane

• Soft margin
  – If still not separable
  – Penalty, hinge loss
Problems
SVMs & statistical learning generally

• Little useful information from each training example
  – Signal must show through the noise
  – Need many training examples
  – Thousands are needed for handwritten digits
  – Millions for other applications

• Weak bias vocabulary

• Much information must be discovered / invented

• Compare similar human behavior
  – Novel handwritten shapes of similar complexity
  – Master with several tens (perhaps a hundred) training examples
  – Exceedingly small non-fatigue error rate
## Two Related Classification Problems

<table>
<thead>
<tr>
<th></th>
<th>No. examples</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humans</td>
<td>&lt; 100 ?</td>
<td>negligible</td>
</tr>
<tr>
<td>SVMs</td>
<td>60000</td>
<td>1.2%</td>
</tr>
</tbody>
</table>
Two Related Classification Problems

<table>
<thead>
<tr>
<th></th>
<th>No. examples</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Humans</td>
<td>&lt; 100 ?</td>
<td>negligible</td>
</tr>
<tr>
<td>SVMs</td>
<td>60000</td>
<td>1.2%</td>
</tr>
</tbody>
</table>

a fixed permutation over pixels
Two Related Classification Problems

To an SVM these are the same problem

Apparently the SVM ignores information crucial to people
Ensemble Methods

• Blend multiple hypotheses
• Bagging
  – Learn multiple h’s
  – Combine labels assigned by the h’s
• Bayes Optimal
  – Learn w/ all hypotheses (or...)
  – Weigh by Pr(h | Z)
• Boosting
  – Re-weight examples
  – Keep training
  – Adding new weighted classifiers
• Generally “learn” some h ∉ H
  (label test examples according to some h ∉ H)
Bagging

• Discussed earlier w/ decision tree learning
• Average over a set of quasi-independent concepts
• Alter the learning protocol
  – Changes the h selected by the algorithm
  – Does not change the performance of h on Z
  – Order of presentation for perceptrons
  – Different random subsets of Z
  – …
• Learn a set of h’s
• Classify new examples by vote
  – Majority, Average Pr, Weighted Votes…
• “Decision Forest”
Bayes Optimal

• Evaluate all hypotheses on training set Z
  – Intractable
  – Perhaps avoid the poor ones
• Calculate $Pr(h \mid Z)$ for each
  – Difficult to get a true distribution
  – Can use accuracy as a stand-in
  – Some difficulties
• Each $h$ may select a $y$ or assign a distribution over $y$’s

$$\arg\max_y \sum_{h \in H} Pr(y \mid h(x)) Pr(h \mid Z)$$
Boosting

• Weak vs. Strong Classifiers
  – Weak classifier performs slightly better than chance
  – Strong classifier performs as accurately as you would like

• Can a strong classifier be constructed by assembling weak ones?

In theory: YES
In practice: Kind of
Boosting

• Suppose we can learn weak classifiers
  – Maybe a hypothesis space H with many simple classifiers
  – Decision stumps are popular
    Decision tree w/ just one test

• Weak learner must be trainable to perform slightly better than chance
  (but on *any* distribution of examples)
Boosting

• Given a training set Z and a hypothesis space H
• Learn a sequence of classifiers
• At each iteration, add a weak classifier \( h_i \)
• Weigh \( h_i \) by performance on (weighted) Z
• Each new h is trained on same Z but \textit{re-weighted} so that hard \( z_j \) count more
• Classify using the weighted majority ensemble
What two things are strange about this???
Boosting

• Boosting can yield excellent ensembles
• Even with many components boosting seems not to overfit
• Boosting decision stumps is popular
• Why does it work?
  – Difficult examples keep growing in weight
  – Empirical identification of support vectors(?)
• Why is the weak learner assumption difficult to satisfy in reality?
• When does the weak learner assumption hold?
• Large H of diverse weak learners can make up for a lack of complex hypotheses