Notes on Applying SVM

• Many SVM implementations are available, and can be found at [www.kernel-machine.org/software.html](http://www.kernel-machine.org/software.html)

• Handling multiple class problem with SVM requires transforming a multiclass problem into multiple binary class problems
  – One against rest
  – Pairwise
  – etc
Model selection for SVM

- There are a number of model selection questions when applying SVM
  - Which kernel functions to use?
  - What c parameter to use (soft margin)?
- You can choose to use default options provided by the software, but a more reliable approach is to use cross-validations
Strength vs weakness

• **Strengths**
  – The solution is globally optimal
  – It scales well with high dimensional data
  – It can handle non-traditional data like strings, trees, instead of the traditional fixed length feature vectors
    • Why? Because as long as we can define a kernel function for such input, we can apply svm

• **Weakness**
  – Need to specify a good kernel
  – Training time can be long if you use the wrong software package

<table>
<thead>
<tr>
<th></th>
<th>Pegasos</th>
<th>SVM-Perf</th>
<th>SVM-Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAT</td>
<td>2</td>
<td>77</td>
<td>20,075</td>
</tr>
<tr>
<td>Covertype</td>
<td>6</td>
<td>85</td>
<td>25,514</td>
</tr>
<tr>
<td>astro-ph</td>
<td>2</td>
<td>5</td>
<td>80</td>
</tr>
</tbody>
</table>

2007  2006  1998
Ensemble Learning
Ensemble Learning

- So far we have seen learning algorithms that take a training set and output a classifier
- What if we want more accuracy than current algorithms afford?
  - Develop new learning algorithm
  - Improve existing algorithms
- Another approach is to leverage the algorithms we have via ensemble methods
  - Instead of calling an algorithm just once and using its classifier
  - Call algorithm multiple times and combine the multiple classifiers
What is Ensemble Learning

Traditional:

- Training set: $S$
- Learner: $L_1$
- Classifier: $h_1$

$y^* = h_1(x)$

Ensemble method:

- Training set: $S$
- Learner: $L_1, L_2, \ldots, L_S$
- Classifier: $h_1, h_2, \ldots, h_S$

$y^* = h^*(x) = F(h_1, h_2, \ldots, h_S)$

Different training sets and/or learning algorithms
Ensemble Learning

• **INTUITION:** Combining Predictions of multiple classifiers (an ensemble) is more accurate than a single classifier.

• Justification:
  
  – easy to find quite good “rules of thumb” however hard to find single highly accurate prediction rule.
  
  – If the training set is small and the hypothesis space is large then there may be many equally accurate classifiers.
  
  – Hypothesis space does not contain the true function, but it has several good approximations.
  
  – Exhaustive global search in the hypothesis space is expensive so we can combine the predictions of several locally accurate classifiers.
How to generate ensemble?

• There are a variety of methods developed
• We will look at two of them:
  – Bagging
  – Boosting (Adaboost: adaptive boosting)
• Both of these methods takes a single learning algorithm (we will call this the base learner) and use it multiple times to generate multiple classifiers
Bagging: **Bootstrap Aggregation**
(Breiman, 1996)

- Generate a **random sample** from training set $S$ by a random re-sampling technique called **bootstrapping**
- Repeat this sampling procedure, getting a sequence of $T$ **training sets**: $S_1, S_2, ..., S_T$
- Learn a sequence of classifiers $h_1, h_2, ..., h_T$ for each of these training sets, using the same base learner
- To classify an unknown point $X$, let each classifier predict
  - $h_1(X) = 1, h_2(X) = 1, h_3(X) = 0, ..., h_T(X) = 1$,
- Take simple **majority vote** to make the final prediction
  - Predict the class that gets the most vote from all the learned classifiers
Bootstrapping

\[ S' = \{\} \]
For \( i=1, \ldots, N \) (\( N \) is the total number of points in \( S \))
\[ \text{draw a random point from } S \text{ and add it to } S' \]
End
Return \( S' \)

• This is a sampling procedure that samples with replacement
  – Each time a point is drawn, it will not be removed
  – This means that we can have multiple copies of the same data point in my sample
  – New training set contains the same number of points (may contain repeats) as the original training set
  – On average, 66.7% of the original points will appear in a random sample
Bagging Example

The true decision boundary
Decision Boundary by the CART Decision Tree Algorithm

Note that the decision tree has trouble representing this decision boundary
By averaging 100 trees, we achieve better approximation of the boundary, together with information regarding how confidence we are about our prediction.
Empirical Results for Bagging Decision Trees
(Freund & Schapire)

Why can bagging improve the classification accuracy?
The Concept of Bias and Variance

Target

Bias

Variance
Bias/Variance for classifiers

• Bias arises when the classifier cannot represent the true function – that is, the classifier underfits the data

• Variance arises when the classifier overfits the data – minor variations in training set cause the classifier to overfit differently

• Clearly you would like to have a low bias and low variance classifier!
  – Typically, low bias classifiers (overfitting) have high variance
  – high bias classifiers (underfitting) have low variance
  – We have a trade-off
Effect of Algorithm Parameters on Bias and Variance

• k-nearest neighbor: increasing k typically increases bias and reduces variance
• decision trees of depth D: increasing D typically increases variance and reduces bias
Why does bagging work?

• Bagging takes the average of multiple models -- reduces the variance
• This suggests that bagging works the best with low bias and high variance classifiers
Boosting
Boosting

• Also an ensemble method: the final prediction is a combination of the prediction of multiple classifiers.

• What is different?
  – Its iterative.

  **Boosting:** Successive classifiers depends upon its predecessors - look at **errors from previous classifiers** to decide what to **focus** on for the next iteration over data

  **Bagging:** Individual classifiers were independent.

  – All training examples are used in each iteration, but with different weights – more weights on difficult examples. (the ones on which we committed mistakes in the previous iterations)
**Adaboost: Illustration**

**Final Classifier**

\[ H(X) = \text{sign} \left[ \sum_{m=1}^{M} \alpha_m h_m(x) \right] \]

- **Original data:** uniformly weighted
  - **Training Sample**
  - **Update weights**
    - **Weighted Sample**
      - **Update weights**
        - **Weighted Sample**
          - **Update weights**
            - **Weighted Sample**
              - **Final Classifier**

Update weights
The AdaBoost Algorithm

**Input:** a set $S$, of $m$ labeled examples: $S = \{(x_i, y_i), i = 1, 2, \ldots, m\}$,
labels $y_i \in Y = \{1, \ldots, K\}$
Learn (a learning algorithm)
a constant $L$. 
The AdaBoost Algorithm

Input: a set $S$, of $m$ labeled examples: $S = \{(x_i, y_i), i = 1, 2, \ldots, m\}$, labels $y_i \in Y = \{1, \ldots, K\}$
Learn (a learning algorithm)
a constant $L$.

1. initialize for all $i$: $w_1(i) := 1/m$ initialize the weights
2. for $\ell = 1$ to $L$ do
3. for all $i$: $p_\ell(i) := w_\ell(i)/(\Sigma_i w_\ell(i))$ compute normalized weights
4. $h_\ell := \text{Learn}(p_\ell)$ call Learn with normalized weights.
5. $\epsilon_\ell := \Sigma_i p_\ell(i)[h_\ell(x_i) \neq y_i]$ calculate the error of $h_\ell$
6. if $\epsilon_\ell \geq 1/2$ then
7. $L := \ell - 1$
8. exit
9. $\beta_\ell := \epsilon_\ell/(1 - \epsilon_\ell)$
10. for all $i$: $w_{\ell+1}(i) := w_\ell(i)\beta_\ell^{1-[h_\ell(x_i) \neq y_i]}$ compute new weights

Output: $h_f(x) = \arg\max_{y \in Y} \sum_{\ell=1}^L \left(\log \frac{1}{\beta_\ell}\right) \left[h_\ell(x) = y\right]$
AdaBoost(Example)

Original Training set: Equal
Weights to all training samples

$D_1$

$\varepsilon_1 = 0.30$
$\alpha_1 = 0.42$

Taken from “A Tutorial on Boosting” by Yoav Freund and Rob Schapire
AdaBoost (Example)

ROUND 1

\[ h_1 \]

\[ \varepsilon_1 = 0.30 \]
\[ \alpha_1 = 0.42 \]

\[ D_2 \]
AdaBoost(Example)

ROUND 2

\[ h_2 \]

\[ \varepsilon_2 = 0.21 \]
\[ \alpha_2 = 0.65 \]

\[ D_3 \]
AdaBoost(Example)

ROUND 3

\[ h_3 \]

\[ \epsilon_3 = 0.14 \]
\[ \alpha_3 = 0.92 \]
AdaBoost(Example)

\[ H_{\text{final}} = \text{sign} \left( 0.42 + 0.65 + 0.92 \right) \]
Weighted Error

• Adaboost calls $L$ with a set of prespecified weights
• It is often straightforward to convert a base learner $L$ to take into account an input distribution $D$.

Decision trees?

K Nearest Neighbor?

Naïve Bayes?

• When it is not straightforward we can resample the training data $S$ according to $D$ and then feed the new data set into the learner.
Boosting Decision Stumps

Decision stumps: very simple rules of thumb that test condition on a single attribute.

Among the most commonly used base classifiers – truly weak!

Boosting with decision stumps has been shown to achieve better performance compared to unbounded decision trees.

Steep decrease in error
Boosting Performance

- Comparing C4.5, boosting decision stumps, boosting C4.5 using 27 UCI data set
  - C4.5 is a popular decision tree learner
Boosting vs Bagging of Decision Trees
Overfitting?

• Boosting drives training error to zero, will it overfit?
• Curious phenomenon

- Boosting is often robust to overfitting (not always)
- Test error continues to decrease even after training error goes to zero
Explanation with Margins

\[ f(x) = \sum_{l=1}^{L} w_l \cdot h_l(x) \]

Histogram of functional margin for ensemble just after achieving zero training error
Effect of Boosting: Maximizing Margin

Even after zero training error the margin of examples increases. This is one reason that the generalization error may continue decreasing.
Bias/variance analysis of Boosting

- In the early iterations, boosting is primarily a bias-reducing method.
- In later iterations, it appears to be primarily a variance-reducing method.
What you need to know about ensemble methods?

- Bagging: a randomized algorithm based on bootstrapping
  - What is bootstrapping
  - Variance reduction
  - What learning algorithms will be good for bagging?
- Boosting:
  - Combine weak classifiers (i.e., slightly better than random)
  - Training using the same data set but different weights
  - How to update weights?
  - How to incorporate weights in learning (DT, KNN, Naïve Bayes)
  - One explanation for not overfitting: maximizing the margin