Bagging, Boosting and Costs

Three *meta-level* techniques are often useful in Data Mining applications.

They are termed meta-level because they apply to a learning algorithm rather than an instance set, and are aimed at improving the performance of a learner on an instance set.

They are generally applicable to any learning algorithm.

- Bagging
- Boosting
- Cost-based Learning
Bagging

Bagging is a simple technique generally useful to:

• reduce the impact of the order of instances on learning algorithms whose output models are order-dependent, and/or

• reduce the probability of misclassification based on any single induced model

Let $L$ be the chosen learning algorithm, $N$ be a user-defined parameter specifying the number of samples/bags, and $d$ the size of each bag.

Algorithm Bagging($Instance\_set$, $L$, $N$, $d$)

For $k \leftarrow 1$ to $N$

$S_k \leftarrow$ random sample of size $d$ drawn from $Instance\_set$

$M_k \leftarrow$ the model induced by $L$ from $S_k$

For each new query instance $q$

$Class(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, M_i(q))$

where $V$ is the finite set of target class values, and $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise.

Note the similarity between bagging and $N$-fold cross-validation.
Boosting

Boosting is based on the observation that finding many rough rules of thumb (i.e., weak learning) can be a lot easier than finding a single, highly accurate prediction rule (i.e., strong learning).

Boosting assumes that weak learners can be made strong by repeatedly running a given weak learner on various distributions over the training data (i.e., varying the focus of the learner), and then combining the weak classifiers into a single composite classifier.

As with bagging, boosting generates a hypothesis whose error on the training set is small by combining many hypotheses whose error may be large (but still better than random guessing - see the test on $\epsilon_t$ in the AdaBoost.M1 algorithm).

However, unlike bagging, boosting tries actively to force the weak learning algorithm to change its hypothesis by changing the distribution over the training instances as a function of the errors made by previously generated hypotheses.
AdaBoost.M1

Let $L$ be the chosen “weak” learning algorithm and $T$ be the number of iterations to perform.

Algorithm AdaBoost.M1($Instance\_set$, $L$)

For $i \leftarrow 1$ to $|Instance\_set|$ 

\[ D_1(i) \leftarrow \frac{1}{|Instance\_set|} \]

For $t = 1$ to $T$

$h_t \leftarrow$ the model induced by $L$ from $Instance\_set$ with distribution $D_t$

$\epsilon_t \leftarrow \sum_{i:h_t(x_i)\neq y_i} D_t(i)$

If $\epsilon_t > .5$

\[ T \leftarrow t - 1 \]

Abort loop

$\beta_t \leftarrow \frac{\epsilon_t}{1-\epsilon_t}$

For $i \leftarrow 1$ to $|Instance\_set|$

\[ D_{t+1}(i) \leftarrow \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases} \]

where $Z_t$ is a normalisation constant, chosen so that $D_{t+1}$ will be a distribution

\[ h_{final}(x) \leftarrow \arg\max_{y \in Y} \sum_{t:h_t(x) = y} \log \frac{1}{\beta_t} \]