Holdout and Cross-Validation Methods Overfitting Avoidance

Decision Trees
- Reduce error pruning
- Cost-complexity pruning

Neural Networks
- Early stopping
- Adjusting Regularizers via Cross-Validation

Nearest Neighbor
- Choose number of neighbors

Support Vector Machines
- Choose C
- Choose $\sigma$ for Gaussian Kernels
Reduce Error Pruning

Given a data sets $S$
- Subdivide $S$ into $S_{\text{train}}$ and $S_{\text{dev}}$
- Build tree using $S_{\text{train}}$
- Pass all of the $S_{\text{dev}}$ training examples through the tree and estimate the error rate of each node using $S_{\text{dev}}$
- Convert a node to a leaf if it would have lower estimated error than the sum of the errors of its children
Reduce Error Pruning Example
Cost-Complexity Pruning

The CART system (Breiman et al, 1984), employs cost-complexity pruning:

\[ J(\text{Tree},S) = \text{ErrorRate}(\text{Tree},S) + \alpha |\text{Tree}| \]

where \(|\text{Tree}|\) is the number of nodes in the tree and \(\alpha\) is a parameter that controls the tradeoff between the error rate and the penalty

\(\alpha\) is set by cross-validation
Determining Important Values of $\alpha$

- Goal: Identify a finite set of candidate values for $\alpha$. Then evaluate them via cross-validation.
- Set $\alpha = \alpha_0 = 0; t = 0$
- Train S to produce tree T
- Repeat until T is completely pruned
  - determine next larger value of $\alpha = \alpha_{k+1}$ that would cause a node to be pruned from T
  - prune this node
  - $t := t + 1$
- This can be done efficiently
Choosing an $\alpha$ by Cross-Validation

- Divide $S$ into 10 subsets $S_0, \ldots, S_9$
- In fold $v$
  - Train a tree on $U_{i\neq v} S_i$
  - For each $\alpha_k$, prune the tree to that level and measure the error rate on $S_v$
  - Compute $\varepsilon_k$ to be the average error rate over the 10 folds when $\alpha = \alpha_k$
  - Choose the $\alpha_k$ that minimizes $\varepsilon_k$. Call it $\alpha^*$ and let $\varepsilon^*$ be the corresponding error rate
- Prune the original tree according to $\alpha^*$
The 1-SE Rule for Setting $\alpha$

- Compute a confidence interval on $\varepsilon^*$ and let $U$ be the upper bound of this interval.
- Compute the smallest $\alpha_k$ whose $\varepsilon_k \leq U$. If we use $Z=1$ for the confidence interval computation, this is called the 1-SE rule, because the bound is one “standard error” above $\varepsilon^*$. 

![Graph showing CV error (epollon-bar) against alpha values]
Notes on Decision Tree Pruning

- Cost-complexity pruning usually gives best results in experimental studies.
- Pessimistic pruning is the most efficient (does not require holdout or cross-validation) and it is quite robust.
- Reduce-error pruning is rarely used, because it consumes training data.
- Pruning is more important for regression trees than for classification trees.
- Pruning has relatively little effect for classification trees. There are only a small number of possible prunings of a tree, and usually the serious errors made by the tree-growing process (i.e., splitting on the wrong features) cannot be repaired by pruning.
  - Ensemble methods work much better than pruning.
Holdout Methods for Neural Networks

- Early Stopping using a development set
- Adjusting Regularizers using a development set or via cross-validation
  - amount of weight decay
  - number of hidden units
  - learning rate
  - number of epochs
Early Stopping using an Evaluation Set

- Split $S$ into $S_{\text{train}}$ and $S_{\text{dev}}$
- Train on $S_{\text{train}}$, after every epoch, evaluate on $S_{\text{dev}}$. If error rate is best observed, save the weights
Reconstituted Early Stopping

- Recombine $S_{\text{train}}$ and $S_{\text{dev}}$ to produce $S$
- Train on $S$ and stop at the point (# of epochs or mean squared error) identified using $S_{\text{dev}}$
We can stop either when MSE on the training set matches the predicted optimal MSE or when the number of epochs matches the predicted optimal number of epochs.

Experimental studies show little or no advantage for reconstituted early stopping. Most people just use simple holdout.
Nearest Neighbor: Choosing $k$

- $k=9$ gives best performance on development set and on test set.
- $k=13$ gives best performance based on leave-one-out cross-validation.
SVM Choosing C and \( \sigma \)
(BR Data Set; 100 examples; Valentini 2003)
20% label noise
BR Data Set: Varying $\sigma$ for fixed $C$

- $C=1$
- $C=10$
- $C=50$
- $C=100$
Summary

- Holdout methods are the best way to choose a classifier
  - Reduce error pruning for trees
  - Early stopping for neural networks
- Cross-validation methods are the best way to set a regularization parameter
  - Cost-complexity pruning parameter $\alpha$
  - Neural network weight decay setting
  - Number $k$ of nearest neighbors in $k$-NN
  - $C$ and $\sigma$ for SVMs