CS 4245 Lecture 14

Above Learning III

Alexander Gray

agray@cc.gatech.edu

Georgia Institute of Technology

Today

Feature selection approaches:

1. Wrapper approaches
2. Cross-validation with feature selection

Feature Selection

Our goal: Find the subset of the features that yields the best generalization performance.

This performance can be evaluated on a hold-out set, by cross-validation, or by using the training error as a (downward-biased) estimate of the generalization accuracy.

Feature Selection

There are two general types of approaches:

- *Embedded methods:* Through regularization, implicitly perform feature selection as part of the training optimization.
- *Wrapper methods:* An outer loop that sits “above” learning, calling an ML method (such as SVM) as a black box.

Note that though they seem quite different, there are deep connections between the two (for example between forward selection, which we will discuss, and the lasso).

Best-Subset Selection

Best-subset selection: Try each possible subset of the features, and see how well it performs.

Notes:

- The optimal feature subset is not necessarily unique.
- There is a combinatorial number of possible feature subsets.

Forward Stepwise Selection

Forward stepwise selection: Start with 0 features. Then:

For d from 1 to D:

- Add the feature which, when added to the current $(d-1)$-feature model, gives the best-predicting d-feature model.

This returns one best feature subset for each size d from 1 to D; choose the best of these.

Forward Stepwise Selection

Notes:

- This does not consider all possible subsets; it is a greedy procedure.
- The true best feature subset of size 2 does not necessarily include the best feature subset of size 1.
- Nonetheless, in practice, the performance of the best subset of each size found by forward stepwise selection is often close to that of the best subset of that size found by best-subset selection.

Backward Stepwise Selection

Backward stepwise selection: Start with all D features. Then:

For d from $D-1$ to 1:

- Delete the feature which, when deleted from the current d-feature model, gives the best-predicting d-feature model.

As before, choose the best feature subset.

Again, in practice, the performance of the best subset of each size found by backward stepwise selection is often close to that of the best subset of that size found by best-subset selection.

Forward-Stagewise Selection

The *correlation* between two random variables X and Y is

$$\rho_{X,Y} = \text{corr}(X,Y) = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}\left[(X-\mu_X)(Y-\mu_y)\right]}{\sigma_X \sigma_Y} \quad (1)$$

and the *sample correlation* coefficient is

$$r_{x,y} = \frac{\sum_i^N (x_i - \overline{x})(y_i - \overline{y})}{(N-1)s_x s_y} \quad (2)$$

where $s_x = \sqrt{N\sum x_i^2 - (\sum x_i)^2}$ and $s_y = \sqrt{N\sum y_i^2 - (\sum y_i)^2}$.

Forward-Stagewise Selection

Forward-stagewise selection: Start with all the coefficients as 0 and the intercept as $\overline{y}$. Then:

- Choose the feature most correlated with the residual (*error*) of the current model.
- Compute the linear regression coefficient of this feature as a predictor of the residual, and add it to the current coefficient of that feature.

Stop when none of the features have any correlation with the residual.

Forward-Stagewise Selection

This procedure can be slower than the previous ones in moderate-sized dimensionalities, but can be effective in high dimensionalities.

Note the similarity to boosting. Another thing that was similar (and fairly successful) in the context of neural networks was called the *cascade correlation* approach.

Recursive Feature Elimination

This applies only to linear models, such as a linear SVM.

Recursive feature elimination: Start with all D features. Then:

For d from D to 1:

- Learn a model with d features, then delete the k features with the smallest coefficients.

As before, choose the best feature subset.

This is popular in the bioinformatics literature.

Use an Embedded Method

Idea: A decision tree (or lasso) does feature selection as a side-effect of its training; first learn a decision tree (or lasso), then use only the features in your learner that appear in the decision tree (or the non-zero lasso coefficients).

- The features that are optimal for a decision tree might not be optimal for another method.
- A decision tree can only contain about $2N$ variables, where N is the number of data, while the number of dimensions D might be greater than N.

Relative Variable Importance in a Tree

A measure of the relative importance of each dimension X_d in a decision tree T is

$$\text{imp}_d^2(T) = \sum_t \hat{i}_t^2 I(v(t) = X_d) \tag{3}$$

where the sum is over the internal nodes t of the tree, and $v(t)$ is the variable used to split that node's region into two child regions, each containing a constant prediction guess. It was chosen as the variable which gave the largest squared improvement $\hat{i}_t^2$ in error over that for the constant prediction guess for the whole region.

Relative Variable Importance in a Tree

The squared relative importance of variable $\text{imp}_d^2(T)$ is the sum of such squared improvements over all internal nodes for which it was chosen as the splitting variable.

If you have many trees, as in bagging or random forests, you can report for each variable the sum over all trees T, as the extra averaging will yield a more reliable estimate.

Other Feature Selection Ideas

A few other ideas:

- Branch-and-bound search, a generalization of the implicit search methods in forward stepwise, backward stepwise, and best-subset selection
- The leaps-and-bounds procedure, a best-subset procedure for linear regression
- Hypothesis testing on the correlation of each variable with the target, which is popular in the bioinformatics literature
- Racing, a way to efficiently compare different models in parallel

Cross-Validation with Feature Selection

A common strategy is the following:

1. Perform feature selection via some procedure to identify the best features.
2. Create a learner using only those features.
3. Use cross-validation to obtain the best parameters for the learner and to estimate the learner's generalization performance.

What's wrong with this?

Cross-Validation with Feature Selection

The problem is that cross-validation is not mimicking truly unseen data, since the feature selection has already "seen" the entire dataset.

Instead, do a separate feature selection and training within each of the K folds of cross-validation.

In other words, in any multiple-step modeling procedure, cross-validation must be applied to the entire sequence of modeling steps taken as a whole, *if* the steps involve the target variable; unsupervised steps are okay.