Building Effective Models for Machine Learning

Jessica Young
University of Notre Dame
Center for Social Science Research
jyoung22@nd.edu
Overview

- Cross Validation
- Ensembles
- Model Metrics
The purpose of cross validation is to make sure that we train our algorithm on a subset of the data so that we can then test how well it will do if more data becomes available. This helps us to avoid over-fitting our model.
First cut up our data into a training set and a test set (split on observations, not variables)

Divide our training set into $k$ equally sized folds that do not overlap

Leave out one fold (called the validation set) and train model on rest of training set to optimize hyper parameters

Repeat the step above for each $k$ fold and average the validation errors across the $k$ folds that you’ve done (aka find the cross validation error)

Once you’ve optimized hyper-parameters you can move on to testing how well the model fits the test set
The oldest and most commonly used version of cross validation is leave-one-out cross validation which has a large fold since you leave out one observation and recalculate the model each time. If you don’t average over folds and instead leave the CV errors as they are calculated, then you get the leverage of each point. On the other hand, if you have tiny folds, then you risk instability since the error will jump all over the place.
Ensembles

- Overview
- Bootstrapping
- Bagging
- Random SubSpace Learning
- Boosting
create 1 estimator of $g$ based on training set of data

\[ \hat{g}(\cdot) \]

Problem: What if your estimator is struggling? What if your data is more complex than just 1 model type can handle?
Ensembles

The basic idea behind ensemble learning is to combine all of the available candidate estimators (aka models) that predict better than random guessing to better your overall predictions. Each of these estimators of $g$ are called base learners. The way that these base learners are combined can be changed to reflect different ensemble methods.

For example, instead of using just one regression model, instead you could use 5 models and then average their predictions to theoretically get an overall better predictive performance.
In ensemble learning, we instead create capital M different estimators of $g$

$$\{\hat{g}_1(\cdot), \hat{g}_2(\cdot), \ldots, \hat{g}_m(\cdot), \ldots \hat{g}_M(\cdot)\}$$

Our overarching goal is to find an estimator of $g$ with the smallest prediction error.
Ensembles

- If base learners are strongly correlated, then the overall performance of your ensemble will decrease since you won’t be adding any new knowledge to your machine.
- A base learner should only be added to the ensemble if it’s at least as good as the current best base learner in the ensemble.
- The collection of base learners do not have to be the same model type.
- You could have an ensemble of models built on different random subsets of the data that are all SVM models, or you could aggregate a mix of trees and SVM models built on different random subsets of the data.
So now we’ve created M different models, now what?
Now we need to combine these model’s predictions so that we have only 1 predictions per observation.
Ensembles

This can be done using a weighting scheme.

\[ \alpha_M \text{ for } m = 1, 2, \ldots, M \]

What you choose to set \( \alpha_m \) equal to for each model can be different!
Once you’ve chosen your weighting scheme you aggregate your models by multiplying each model’s outcomes by the scheme and then add them together. Weighting schemes can be basic, like just averaging, or more complex like using model selection scores (AIC, BIC, etc.).
You have 3 models total, $\hat{g}_1(\cdot), \hat{g}_2(\cdot), \hat{g}_3(\cdot)$.

If you want to aggregate these models together by taking the average then what should you set $\alpha_m$ to?
\( \hat{g}^{agg}(\cdot) = \sum_{m=1}^{M} \alpha_m \hat{g}_m(\cdot) \)

\[ = \alpha_1 \hat{g}_1(\cdot) + \alpha_2 \hat{g}_2(\cdot) + \alpha_3 \hat{g}_3(\cdot) \]

\[ = \frac{1}{3} \hat{g}_1(\cdot) + \frac{1}{3} \hat{g}_2(\cdot) + \frac{1}{3} \hat{g}_3(\cdot) \]
Bootstrapping says to break up your data by randomly sampling with replacement n times from the training data. For example, if you initially had 100 observations, you could create a bootstrapped sample of only 70 of those observations (some of which may be repeat observations) to now model with.

Keep in mind that bootstrapping data does not affect the number of variables you work with. Bootstrapping is used as a re-sampling method for splitting up data to optimize models.
Bagging is bootstrap aggregating.

Bagging helps reduce the variability of models that may be unstable; if the model is stable, then bagging has been shown to slightly degrade its performance. (Breiman 1996)

1. Draw a bootstrap sample from the data
2. Train a model based on the bootstrap sample and store model
3. After models are all created, average predicted values of all bootstrap models
Random subpace learning is like bagging, but instead only uses some of the predictors

1. Randomly select $d \approx \sqrt{p}$ variables to use
2. Draw a bootstrap sample of the $d$ variables from the data
3. Train a model based on the bootstrap sample and store model
4. After models are all created, average predicted values of all bootstrap models (Skurichina and Duin 2002)
Boosting

- Goal is combine numerous weak learners that predict certain areas of data well together to form a strong aggregate model.
- Boosting searches and obtains different learners to explain different portions of the data; each new learner leads to an aggregate performing better than the previous.
- Ada-boost is a version of boosting which is fairly robust to over-fitting. (Buhlmann and Hothorn 2005)
From here, we’ll look at different ways to assess goodness of fit for models that deal with 2 different tasks:

- Classification
- Regression
### Classification

<table>
<thead>
<tr>
<th>Actual</th>
<th>Prediction</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative</td>
<td>Negative</td>
<td>TN</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>FP</td>
</tr>
<tr>
<td>Positive</td>
<td>FN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>TP</td>
</tr>
</tbody>
</table>

Recall/Sensitivity = \[ \frac{TP}{TP + FN} \]

Specificity = \[ \frac{TN}{TN + FP} \]

Precision = \[ \frac{TP}{TP + FP} \]

Beware the dangers imbalanced data will have on your results! If you only report accuracy (that is if you only report how many times overall your machine’s predictions matched the test set) then your results will be artificially inflated!
The F-measure is a good way to reconcile precision and recall since trying to maximize one will minimize the other (much like the bias-variance trade-off, there is a precision-recall trade-off). Therefore our F is the geometric average of precision and recall.
AUC refers to the Area Under the ROC Curve. The ROC curve is formed by plotting the true positive rate vs the false positive rate (as the y and x values respectively).

AUC can be found by calculating the integral of the ROC curve.
The most common metric for determining goodness of fit in a model is MSE, mean squared error. MSE simply find the residuals from the model and the actual data, squares them, and then calculates the mean. Another approach is to then take the square root of the MSE which produces the RMSE, the root mean squared error. Another way of thinking of MSE and RMSE is:

\[
\text{MSE} = \frac{RSS}{n} \quad \text{RMSE} = \sqrt{\frac{RSS}{n}}
\]
While most non-statisticians are familiar with the concept of $R^2$, they are most likely not familiar with its more unbiased cousin $R^2_{adj}$. While still an imperfect measure of goodness of fit, $R^2_{adj}$ is a much better measure for comparing models than $R^2$. Compare them below:

$$R^2 = 1 - \frac{RSS}{SST} \quad R^2_{adj} = 1 - \frac{RSS/(n - p - 1)}{SST/(n - 1)} = 1 - \frac{MSE}{MST}$$

Notice that $R^2_{adj}$ takes into account the size of the data ($n$ being the number of observations and $p$ being the attributes in the data set). Thus if the model uses more attributes than observations, then it is penalized heavily. Following Occam’s razor dictates, $R^2_{adj}$ shows that a simpler model is preferred over an overly complex model.
AIC, Akaike’s information criterion, tries to choose the best model by penalizing overly complex models while rewarding models’ for how well they fit the data; the smaller the AIC the better.

\[ AIC = 2p + n \log \left( \frac{RSS}{n} \right) \]

Similar to AIC is BIC, Bayesian information criterion which penalizes complex models that rely on few observations more heavily than AIC. Again, the smaller the BIC the better.

\[ BIC = p \log(n) + n \log \left( \frac{SSE}{n} \right) \]

So while AIC tends to choose overly complex models, BIC tends to choose too simple of a model. Outside of comparing models though, AIC and BIC do not say how well a model fits the data.
