Applied Data Mining

Rockhurst University

Model evaluation and tuning

Section 1

Evaluation

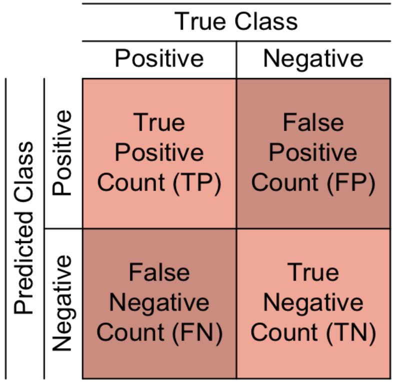
Model performance

Earlier we talked about accuracy as a way to evaluate the model. But how do you know that's the "right" thing to care about?

We now know how to build several types of classification models: KNN, NB, decision trees, and logistic regression. How do we compare them? How do we know which one is "better"?

Let's check the **confusion matrix**:

Confusion matrix



		True Class	
		Positive	Negative
Predicted Class	Positive	True Positive Count (TP)	False Positive Count (FP)
	Negative	False Negative Count (FN)	True Negative Count (TN)

This should look a lot like the cross tabs that we did early on.

Parts of the confusion matrix

- **True positive (TP)**: number/percent that are true and we predicted to be true
- **True negative (TN)**: number/precent that are false and we predicted to be false
- **False positive (FP)**: number/percent that are false and we predicted to be true
- **False negative (FN)**: number/precent that are true and we predicted to be false

The **accuracy** of a model is the number of cases it predicts correctly divided by the total number of cases:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Kappa statistic

One more metric to consider - the *kappa* statistic:

$$Kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}$$

where:
$Pr(a)$: proportion of actual agreement between the classifier and the true values
$Pr(e)$: proportion of expected agreement between the classifier and the true values

Kappa "adjusts accuracy by accounting for the possibility of a correct prediction by *chance alone* Kappa values range to a maximum number of 1, which indicates perfect agreement between the model's predictions and the true values–a rare occurrence. Values less than one indicate imperfect agreement" (Lantz 2013, p. 303)

Kappa example

Simple confusion matrix:

		Actual		
		Cats	**Dogs**	**MF**
	Cats	10	7	17
Predicted	**Dogs**	5	8	13
	MF	15	15	30

Observed accuracy: (10 + 8) / 30 = 0.6
Expected accuracy: based on marginal frequencies (MF): multiply the marginal frequency of Cats for one dimension by the marginal frequency of Cats for the other dimension, and divide by the total number of instances. Repeat for Dogs.

- For cats: 15 * 17 / 30 = 8.5
- For dogs: 15 * 13 / 30 = 6.5
- Add all these values together, and finally divide again by the total number of instances: (8.5 + 6.5) / 30 = 0.5

So Kappa = (0.60 - 0.50)/(1 - 0.50) = **0.20**

Kappa example 2

Simple confusion matrix:

		Actual		
		Cats	**Dogs**	**MF**
	Cats	22	7	29
Predicted	**Dogs**	9	13	22
	MF	31	20	51

Observed accuracy: (22 + 13) / 51 = 0.69

- For cats: 29 * 31 / 51 = 17.63
- For dogs: 20 * 22 / 51 = 8.63
- Add all these values together, and finally divide again by the total number of instances: (17.63 + 8.63) / 51 = 0.51

So Kappa = (0.69 - 0.51)/(1 - 0.51) = **0.37**
Higher Kappa means model matched more of the true data controlling for accuracy of random classifier (expected accuracy)

What is a “good” kappa?

It depends. $\ddot\smile$

Landis and Koch (1977):

0-0.20 ⇒ slight,

0.21-0.40 ⇒ fair,

0.41-0.6 ⇒ moderate,

0.61-0.8 ⇒ substantial,

0.81 -1 ⇒ almost perfect

Fleiss (1981):

0-0.4 ⇒ poor

0.41-0.75 ⇒ fair to good

0.76-1 ⇒ excellent

Kappa‘s aren‘t perfect though. If accuracy isn‘t the best measure for you, then kappa isn‘t the best metric.

Other measures of model performance

Sometimes overall accuracy isn't the best measure by which to judge performance:

- **True Positive rate** is how many positives we predicted correctly out of all the cases that were really positive: $\frac{TP}{TP+FN}$ (Also called **sensitivity** and **recall**)
- **True Negative rate** is how many negatives we predicted correctly out of all the cases that were really negative: $\frac{TN}{TN+FP}$ (Also called **specificity**)
- **Precision** is how many positives we predicted correctly out of all the positive predictions we made: $\frac{TP}{TP+FP}$

When might these be more appropriate than overall accuracy to judge a model's performance?

Note: Wikepedia entry on this is really good

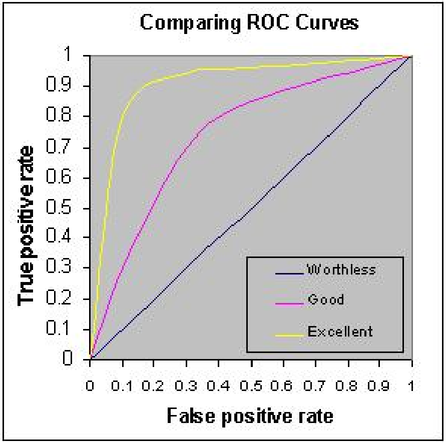
ROC curve

If accuracy isn't the key thing, you might want to consider the evaluating the tradeoff between sensitivity and specificity.

We can do that by looking at the **ROC curve**.

The ROC (receiver operating characteristics) curve displays the true positive rate (sensitivity) against the false positive rate (1-specificity). The closer the curve follows the left hand border and then the top left border of the ROC space, the more accurate the model.

ROC picture


Comparing ROC Curves
True positive rate
False positive rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Worthless
Good
Excellent

Section 2

Cross validation

Making more data

Even with “big data”, we rarely have all the data. How do we make the most of the data we have and avoid overfitting?

We can use our dataset multiple times as if we have multiple data sets.

This is known as **cross validation**

K-fold cross-validation

One method create multiple data sets from the one you have by randomly dividing the dataset into k groups, or folds, of equal size and using $k-1$ folds as the training dataset and the other dataset as the test dataset.

A common k to use is 10.

What does this look like?

- Keep one fold as the validation (hold out) set. Fit the model on the remaining k-1 folds.
- Test fitted model on the held out fold. Calculate the mean squared error (MSE) of the held out fold.
- Repeat Steps 1 and 2 over and over again so that a different fold is used as a validation set. For a 10-fold cross-validation, the procedure is done 10 times.
- The true error rate is estimated as the average error rate.

Choosing K

With a large number of folds (big K):

- The variance of the true error rate estimator will be large
- The computational time will be very large as well (many model runs)

With a small number of folds (little K):

- The number of model runs and, therefore, computation time are reduced
- The variance of the estimator will be small

In practice, the choice of k will depend on the size of the dataset:

- For large datasets, even 3-fold cross-validation will be quite accurate
- For very sparse datasets, we may have to use leave-one-out cross validation in order to train on as many examples as possible

Leave one out cross-validation

If you really want to maximize the amount of data on which you train your model, consider **leave one out** cross-validation. It works just like it sounds:

- For a dataset with N observations, run the model N times
- For each model run use N-1 examples for training and the remaining example for testing
- As before, the true error is estimated as the average error rate on test examples
- Can take a looooong time

Leave one out cross-validation is the extreme case of K-fold where K is number of observations.

Bootstrapping

Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set with replacement.

Each of these “bootstrap data sets” is created by sampling with replacement, and is the same size as our original dataset. As a result some observations may appear more than once in a given bootstrap data set and some not at all.