CIS399: The Science of Data Ethics	Lecture 5: Jan 31, 2019
Instructor: Michael Kearns & Ani Nenkova	
Scribes: Rohan Shah & Anaya Jhaveri	

Plan for Jan 31 Lecture:

- Generalization and Overfitting - how do these concepts interact with test sets, validation sets and training sets
- How do we evaluate a system or model? There is no standard way to say a classifier is working well

1 Training, Validation and Test sets

We usually divide our data into three sets - training, validation and test sets. We require a a feedback loop for the types of classifiers, etc. that lets us know if we should adjust our feature weights. This is where the training and validation tests are needed. We train our model on our training data, test it on the validation data and then use the results of testing on validation data to tweak the parameters of our model. During development, we have a cycle for adjustment, where theres a continuous loop between training data and validation set. For some parameters, you can use a brute force approach to test a bunch of parameters to see which gives the best result on the validation set.

Once you find best classifier, you can use the test data for verification (i.e. to test how your model would perform in the real-world). If theres a big difference between training loss and test loss, then you know you have overfit your model to the training data. However, what keeps you from overfitting to the test set as well is that you only get 1 run on the test set. On the other hand, theres no set number of iterations that you tweak based on training/validation sets before moving to the test set. We "refresh" as often as seems practical.

Although there is no set division scheme for splitting your data into the three groups, in the Natural Language Processing space, we often find that 20% of data should be test data; another 10-20% should be validation data; another 60-70% should be training data. However, the division scheme is relative to the complexity of the model you are building. 20% test data might be sufficient for some models and not for others. Thus, we base our divisions on the complexity of the model at hand. Of course, the golden rule still stands - the more data you have, the better and this is why we reserve most data for the training set. We want our model to be statistically significant and this is more likely the more data you use. In general, The fewer tweaks you make $\leftarrow$ the less complex you make your model $\leftarrow$ the more reliable your results.

2 n-fold cross validation

Problem: What if by chance our splits between test, training and validation sets results in unrepresentative sets?
Solution: n-fold cross validation

In n-fold cross-validation, we randomly partition our data into n subsets of equal size. Of the

n subsets, we reserve 1 subset as the validation data and use the remaining $n-1$ subsets as training data. The cross-validation process is then repeated n times, with each of the n subsamples used exactly once as the validation data. We can then average the results from the n iterations to determine how successful and stable our model is. The advantage of this method is that every data entry is used for both training and validation, and each data entry is used for validation exactly once (preventing overfitting).

n-fold cross validation usually requires a much larger data set to be possible. However, it serves as a tool to determine how stable/consistent our model/classifier is. If we get consistent results, then we know that its probably a pretty good classifier. On the other hand, if variance in results is large then we know our classifier is not good.

3 Baseline and Ceiling Performance

When evaluating the performance of our model overall, we usually evaluate baseline and ceiling performance. We first must determine what a plausible baseline is. For example, we can set a baseline at 60%, that means if our model can predict on a random sample with 60% accuracy, then it is a good model. Historically, the ceiling can be how accurate are you, relative to a humans performance (or whats the best I can expect given previous model?)

4 Binary Classification

Binary Classification is used in scenarios where there are 2 potential results. For example, these cases fit the bill:

- Will this person commit suicide or not?
- Will this person succeed or not?
- Will this loan be repaid?

In order to evaluate these sorts of cases, you should create a 2×2 confusion matrix with a predicted and a true label. Within the table, there are 4 outcomes: True Positive (TP), False Negative (FN), False Positive (FP), True Negative (TN). After creating this table, you're then able to find percentages which correspond to actual vs. correctly predicted results. This is called a confusion matrix because you can see how much the model is confused per class. An example confusion matrix is included below:

		Prediction outcome: positive	Prediction outcome: negative	
Actual value	positive	TP	FN	$TP + FN$
	negative	FP	TN	$FP + TN$
		$TP + FP$	$FN + TN$	

5 Accuracy

Accuracy is a very important metric in machine learning. Determining accuracy can be found by asking the question, "What fraction of the outputs were predicted correctly?"

Accuracy can be calculated by the following equation: $\frac{(TP+TN)}{(TP+FP+TN+FN)}$
It is not relevant to use accuracy when only a very small fraction of the data is relevant, i.e. imbalanced classes exist. It is very important to track what is known as, "Precision and Recall" (per class) when you have imbalanced classes for a better indicator of performance.

6 Precision and Recall

- Precision = $\frac{TP}{(TP+FP)}$
- Recall = $\frac{TP}{(TP+FN)}$

In general, if you know what sorts of precision metrics you're looking for in the real world, then you should be able to use that as a metric for success. For example, we know that about 7% of the population is gay, so we should use that to determine whether our model is precise enough.

For things like spam detectors, it is incredibly important to have very high precision, or very high confidence that what you determine to be spam is actually spam. Nobody wants a spam detector which may treat valid emails as spam.

It is also important to keep in mind the question, "What is the price or penalty of having less precision vs. less recall?" In general, the more imbalanced the classes, the harder it is to find a good classifier for the smaller class. A way to counteract the issue of an unbalanced problem, one can use weights to create a balanced problem out of an unbalanced problem (weighted classification error). This could be used when theres a 95-5 split, for example. One can also use a model that says something to the effect of, "I dont know more often than I know," in order to increase precision (rather than saying that you have an answer for every single test/data point).