6.047/6.878 Fall 2008 Recitation 3 Notes

Pouya Kheradpour

September 19, 2008

1 Practical issues of machine learning

There are many practical issues to deal with when performing machine learning. This is especially true in computational biology because the data sets are so complex that you have to be very careful that your results are meaningful.

1.1 Overfitting

1. In lecture we saw the example of having many clusters leading to a higher probability of the data.
2. This is an extremely, generally unavoidable problem with machine learning – more parameters leads to better *training* performance.
3. Using more parameters/features generally leads to more overfitting.
    (a) Always consider how much data you have and if it is enough to learn all the parameters your model has. If you have more features, it may be more likely that the classifier finds something that separates the data just by chance.
    (b) The more data you have, the more parameters may be appropriate. We will see more examples of this when we are learning about HMMs.
    (c) Consider Naive Bayes; what if certain features are very rarely seen? The counts may not be high enough to properly estimate that parameter. In cases like this, pseudo-counts can be useful.
4. Overfitting also makes it difficult to draw conclusions from the parameters of a learned model. Better to use specific statistical tests.

1.2 Cross-validation

1. In lecture, machine learning is presented as having a training set and a test set... but what if we just have one set of data and we want to estimate performance? Remember, we can't use training performance because of overfitting.
2. We can use cross validation: partition the data set into multiple sets. Train on all but one set and then test on the other; repeat for all sets to get an overall estimate of performance.
3. Some potential caveats:
    (a) The models are still just as overfit (sometimes more so because you have fewer training samples) – you can't necessarily draw any conclusions from the parameters and the training performance will almost always exceed the test performance.
    (b) You have to be careful that you don't have any training examples that are not independent from the test samples. For example, recently duplicated genes may have many features that are similar only due to their recent homology and not due to their belonging to the same class. To avoid misleading performance due to overfitting, these types of samples should be placed in the same cross-validation set (or all but one should be thrown out all together).
    (c) There may be some loss in performance just because you have fewer training samples – using leave-one-out cross-validation can largely solve this problem.

(d) If your model has some parameter you must supply (e.g. the SVM's kernel), you can overfit that if you simply use cross-validation in order to pick the best one. You should break each training set into a training and validation set to optimize that parameter and then apply it to the test set only once in the end. Note: you can't just use training performance because that will often always increase as you increase a specific parameter.

4. You have to be very careful to never "break" the cross-validation. Never let your learning procedure use the labels of the test data at any time!

2 Support vector machines

Support vector machines (SVMs) were discussed briefly in lecture. I will briefly go through the major concepts. See wikipedia and/or the lecture notes for more details. There are *many* other machine learning techniques (take 6.867)!

1. SVMs are used to classify vectors in multi-dimensional space. This is very, very common problem in machine learning. Sometimes even when your features are not continuous or even numbers, you can still make them fit into something an SVM will classify.

2. Their main property is that try attempt to find a separator that maximizes the *margin* between the positive and negative samples (which ends up being $\frac{2}{||\mathbf{w}||}$).

3. Briefly, this is done by choosing $\mathbf{w}$ and b to minimize $||\mathbf{w}||$, subject to $c_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1, \text{ for all } 1 \geq i \geq n$ where the x_i's are the points each with class $c_i \in \{-1, 1\}$. $\frac{b}{||\mathbf{w}||}$ indicates the distance of the hyperplane from the origin.

4. Minimizing $||\mathbf{w}||$ is equivalent to minimizing $\frac{1}{2}||\mathbf{w}||^2$, but the latter is easier because quadratic programming can be applied.

5. This makes them somewhat robust against overfitting.

6. Extensions have been made to allow mislabeled points.

7. In order find this maximal separator, they rely only on dot products between samples. Because of this, we can easily run SVMs in higher dimensional spaces without having to map the data points into the higher dimensional space by computing a simpler formula for the dot products in the higher space.

8. There are a number of high-quality implementations for SVMs (e.g. SVMlight) that you can use off-the-shelf for many problems.