Lecture 9: Evaluating Predictive Models

36-350: Data Mining

October 9, 2006

So far in the course, we have largely been concerned with descriptive models, which try to summarize our data in compact and comprehensible ways, with information retrieval, etc. Now, in the second half, we are going to focus on **predictive** models, which attempt to foresee what will happen with new data.

We have already seen an example of this in classification problems: algorithms like the nearest-neighbor method try to guess which classes new data belong to, based on old data. You are also familiar with linear regression, which can be used both to describe trends in existing data, and to predict the value of the dependent variable on new data. In both cases, we can gauge how well the predictive model does by looking at its **accuracy**, or equivalently at its **errors**. For classification, the usual measure of error is the fraction of cases mis-classified, called the **mis-classification rate** or just the **error rate**. For linear regression, the usual measure of error is the sum of squared errors, or equivalently $1 - R^2$, and the corresponding measure of accuracy is R^2. In the method of maximum likelihood, the accuracy is just the likelihood value, and the error is conventionally the negative log-likelihood.

What we would like, ideally, is a predictive model which has zero error on future data. We basically never achieve this, partly because our models are imperfect and partly because the world is just noisy and stochastic, and even the ideal model will not have zero error all the time. Instead, we would like to minimize the **expected error**, or **risk**, on future data.

If we didn't care about *future* data, this would be easy. We have various possible models, each with different parameter settings, conventionally written θ. We also have a collection of data $x_1, x_2, \ldots x_n \equiv \mathbf{x}$. For each possible model, then, we can compute the error on the data, $L(\mathbf{x}, \theta)$, called the **in-sample loss** or the **empirical risk**. The simplest strategy is then to pick the model, the value of θ, which minimizes the in-sample loss. This means picking the classifier with the lowest in-sample error rate, or the regression which minimizes the sum of squared errors, or the likelihood-maximizing parameter value — what you've usually done in statistics courses so far.

There is however a potential problem here, because $L(\mathbf{x}, \theta)$ is not what we *really* want to minimize. That is $\mathbb{E}[L(\mathbf{X}, \theta)]$, the *expected* loss on new data drawn from the same distribution. This is also called the risk, as I said, or the **out-of-sample loss**, or the **generalization error** (because it involves generalizing from the old data to new data). The in-sample loss equals the risk plus sampling

noise:

$$L(\mathbf{x}, \theta) = \mathbb{E}\left[L(\mathbf{X}, \theta)\right] + \epsilon(\theta)$$

Here $\epsilon(\theta)$ is a random term which has mean zero, and represents the effects of having only a finite data sample, rather than the complete probability distribution. (I write it $\epsilon(\theta)$ as a reminder that different models are going to be effected differently by the same sampling fluctuations.) The problem, then, is that the model which minimizes the in-sample loss could be one with good generalization performance ($\mathbb{E}\left[L(\mathbf{X}, \theta)\right]$ is small), or it could be one which got very lucky ($\epsilon(\theta)$ was large and negative).

To see how this can matter, consider choosing among different models with different degrees of complexity — also called **model selection**. In particular, consider Figure 1. Here, I have fitted ten different models — ten different polynomials — to the same data set. The higher-order polynomials give, visually at least, a significantly better fit. This is confirmed by looking at the residual sum of squares.

degree	RSS
1	17.19
2	16.62
3	16.02
4	14.53
5	13.54
6	13.01
7	12.25
8	10.80
9	9.77
10	8.65

Since there are only twenty data points, if I continued this out to polynomials of degree twenty, I could get the residual sum of squares down to zero, apparently perfect prediction.

The problem comes when we take these models and try to generalize to new data, such as an extra 200 data points drawn from exactly the same distribution (Figure 2).

degree	Sum of Squared Errors
1	453.68
2	455.28
3	475.53
4	476.36
5	484.18
6	486.97
7	488.77
8	518.39
9	530.25
10	545.37

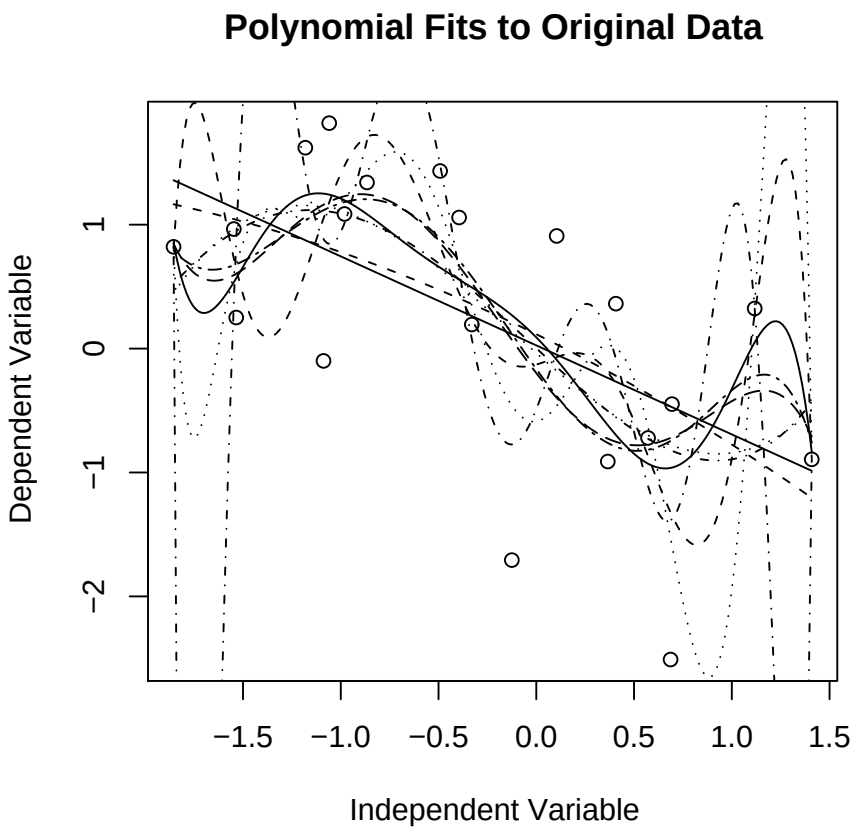

Figure 1: Fits of polynomials from degree 1 to 10 to twenty data points (X is standard Gaussian, $Y = -0.5X +$ another standard Gaussian.

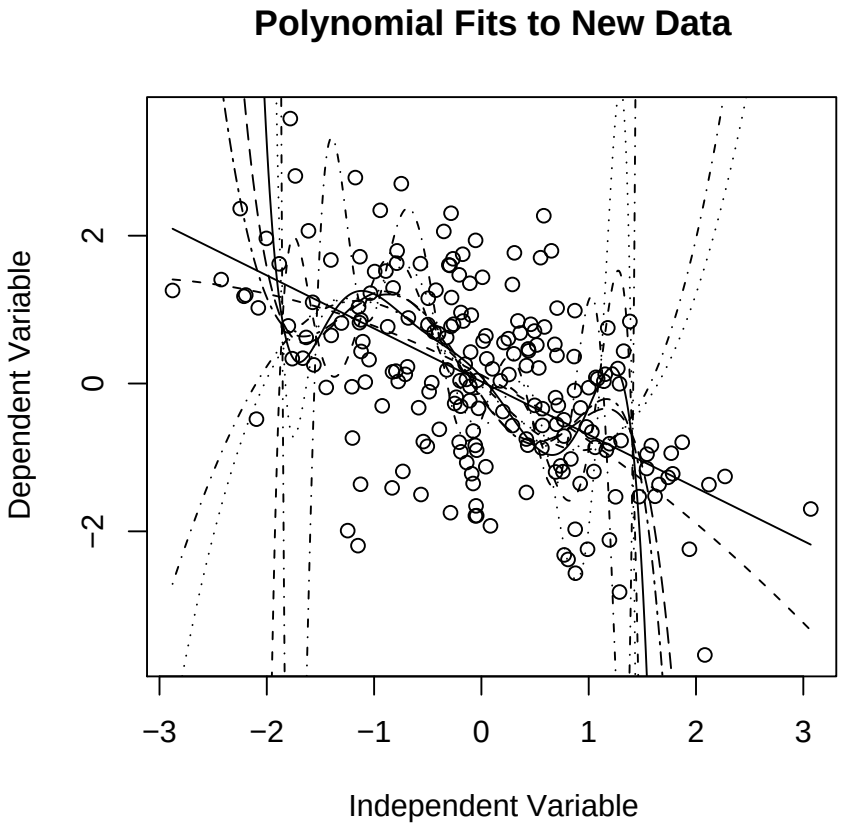

Figure 2: Predictions of the polynomial models on 200 new data points.

What's going on here is that the more complicated models — the higher-order polynomials, with more terms and parameters — were not actually fitting the *generalizable* features of the data. Instead, they were fitting the sampling noise, the accidents which don't repeat. That is, the more complicated models **over-fit** the data. In terms of our earlier notation, ϵ is bigger for the more flexible models. The model which does the best on *new* data is the linear model, because, as it happens the true model here *is* linear. If the true model had been, say, quadratic, then we should have seen the second-degree polynomials do best, and the linear models would have **under-fit** the data, because the (real, generalizable) curvature would have been missed.

After bad data, over-fitting may well be the single biggest problem confronting data mining. People have accordingly evolved many ways to combat it.

Big Data The simplest approach to dealing with over-fitting is to hope that it will go away. As we get more and more data, the law of large numbers (and other limit theorems) tell us that it should become more and more representative of the true data-generating distribution, *assuming there is one*. Thus, ϵ, the difference between the empirical risk and the generalization risk, should grow smaller and smaller. If you are dealing with a *lot* of data, you can hope that ϵ is very small, and that minimizing the in-sample loss will give you a model which generalizes *almost* as well as possible. Unfortunately, if your model is very flexible, "lots of data" can be exponentially large.

Penalization The next bright idea is to say that if the problem is over-flexible models, we should penalize flexibility. That is, instead of minimizing $L(x, \theta)$, minimize $L(x, \theta) + \lambda g(\theta)$, where $g(\theta)$ is some kind of indication of the complexity or flexibility of θ, say the number of parameters, and λ is our trade-off factor. Standard linear regression implements a scheme like this in the form of "adjusted R^2." It can work very well, but the trick is in choosing the penalty term; the number of parameters is often used but often *not* appropriate.

Cross-Validation The most reliable trick, in many ways, is related to what I did above. Divide your data at random into two parts. Call the first part the **training** set, and use it to fit your models. Then evaluate their performance on the other part, the **testing** set. Because you divided the data up randomly, the performance on the test data should be an *unbiased* estimate of the generalization performance. (But, unbiased doesn't necessarily mean "close".) There are many wrinkles here, some of which we'll see, like multi-fold cross-validation — repeat the division q times, and chose the model which comes out best on average over the q different test sets.