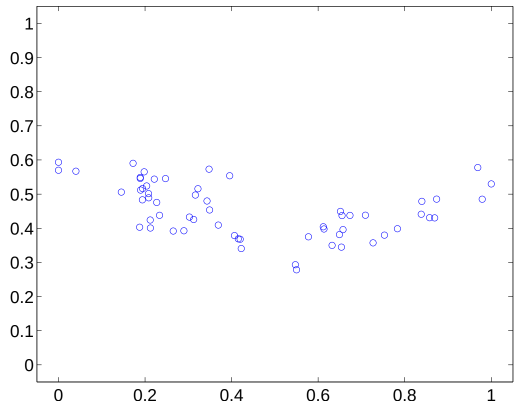
Statistical Machine Learning Notes 1

Overfitting, Model Selection, Cross Validation, Bias-Variance

Instructor: Justin Domke

1 Motivation

Suppose we have some data that we want to fit a curve to:



Here, we want to fit a polynomial, of the form

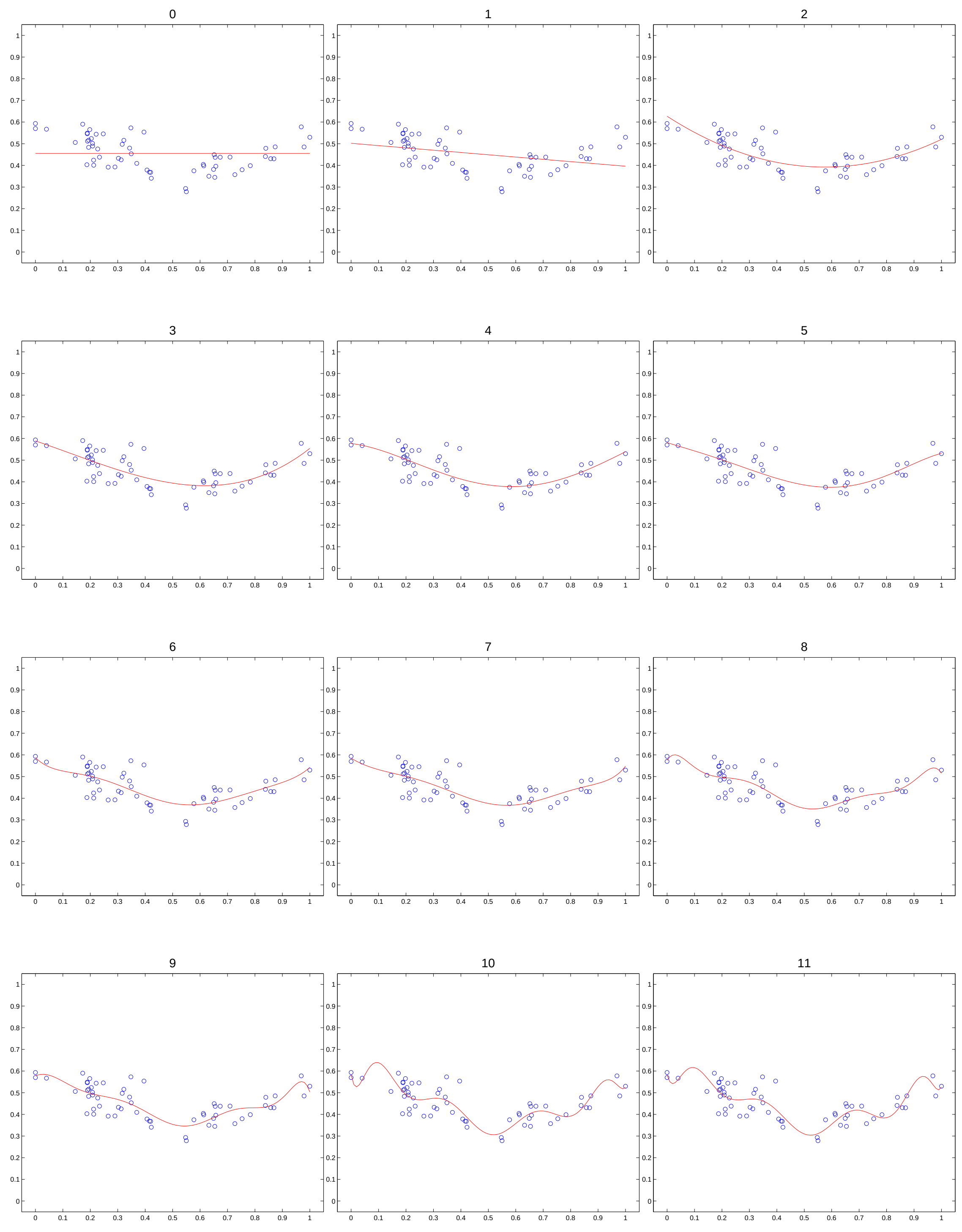
$$y = w_0 + w_1 x + w_2 x^2 + ... + w_p x^p.$$

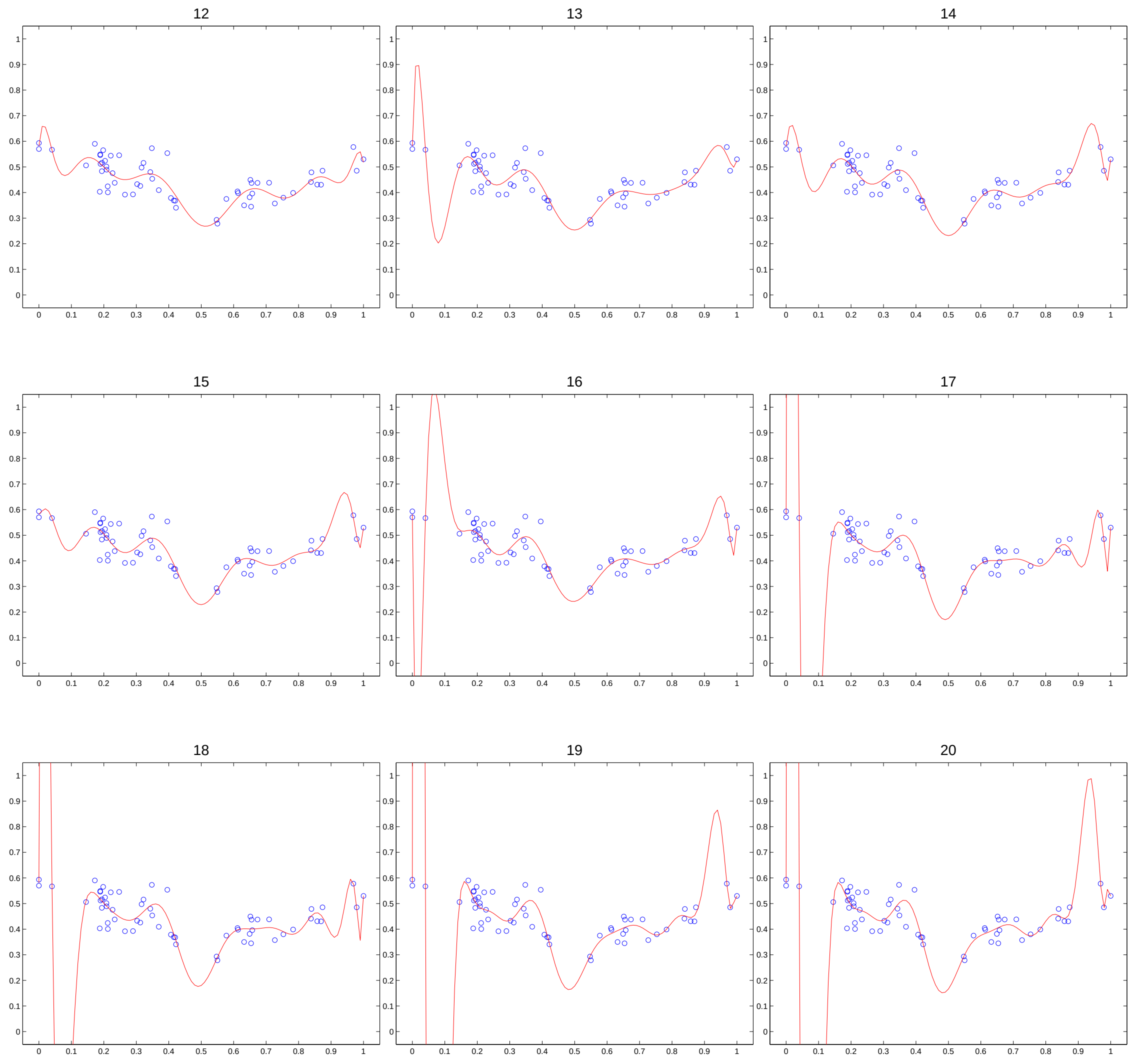
In these notes, we are interested in the question of how to choose p. For various p , we can find and plot the best polynomial, in terms of minimizing the squared distance to all the points $(\hat{y}, \hat{x})$

$$\min_{\mathbf{w}} \sum_{(\hat{y},\hat{x})\in D} \bigl(w_0 + w_1 \hat{x} + w_2 \hat{x}^2 + ... + w_p \hat{x}^p - \hat{y}\bigr)^2. \tag{1.1}$$

We will refer to this sum of squares distances as the **training error**.

$$\sum_{(\hat{y},\hat{x})\in D} \bigl(w_0 + w_1 \hat{x} + w_2 \hat{x}^2 + ... + w_p \hat{x}^p - \hat{y}\bigr)^2$$


0
1
2
3
4
5
6
7
8
9
10
11



The question we address in these notes is: what p do we expect to generalize best? Specifically, we will imagine that we will get some new data TEST from the same distribution. We want to pick p to minimize the same sum-of-squares difference we used for fitting to the training data. This is called the **test data**.

$$\sum_{(\hat{y},\hat{x})\in \text{TEST}} \left(w_0 + w_1\hat{x} + w_2\hat{x}^2 + ... + w_p\hat{x}^p - \hat{y}\right)^2.$$

Our assumption here is that the elements of D and TEST are both "drawn from the same distribution". What this means is that there is some "true" distribution $p_0(x, y)$. We don't know p_0, but we assume all the elements of D and TEST are both drawn from it independently.

An example should make this clear. Suppose that x is a person's height, and y is a person's age. Then $(\hat{x}, \hat{y}) \sim p_0(x, y)$ means that we got $\hat{x}$ and $\hat{y}$ by showing up at a random house on a random day, grabbing a random person, and measuring their height and age. If we got D by doing this in Rochester, and TEST by doing this in Buffalo, then we would be violating the assumption that D and TEST came from the same distribution!

Now, return to our example of choosing p. What will happen to the training error as p gets bigger? Now, clearly, a larger p means more "power", and so the training error will strictly decrease. However, the high degree polynomials appear to display severe artifacts, that don't appear likely to capture real properties of the data. What is going wrong here?

2 The Bias-Variance xTradeoff

We now formally define the "riskiness[1]" K as the expected squared distance between the curve that we fit, and a new point from the true distribution

$$K = E\big[\big(y - f(x; D)\big)^2\big]. \tag{2.1}$$

Here, f is a function that denotes the following process:

- Take the training data D, fit it in a given manner. (For example by Eq. 1.1 with a fixed p)
- Take that function, then output the predicted value for the single point x.

Notice that there are two random events here:

- The drawing of the training data D
- The drawing of a new datum (x, y) from the true distribution.

The expectation in Eq. 2.1 is with respect to *both* of these.

Now, define three terms, the "bias"

$$B = E\big[f(x; D) - E[y|x]\big],$$

the "variance"

$$V = E\Big[\big(f(x; D) - E\big[f(x; D)\big]\big)^2\Big],$$

[1]This would usually by called the "risk", and written with an R. However, we will define a different (but related) "risk" in the next notes, which we will use going forward. We will not see the "riskiness" again after these notes.

and the "intrinsic error"

$$I = E[(y - E[y|x])^2].$$

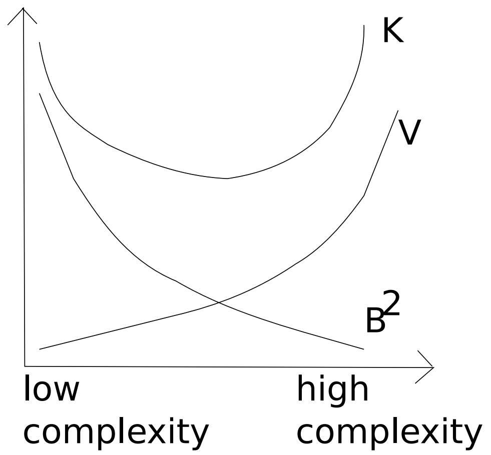
Roughly speaking, the bias measures how different the *average predicted value* is from the *true average value*, the variance measures how different the *predicted value* is from the *average predicted value*. The intrinsic error measures fundamental uncertainties (noise) of the problem. Since nothing can be done about the intrinsic error, we usually don't worry about it too much. Notice that we could also have defined each of these quantities for a fixed x.

A lot of algebra can show what is called (with apologies to the intrinsic error) the **bias-variance decomposition**,

$$\boxed{K = I + B^2 + V.}$$

$$\begin{aligned}
I + B^2 + V &= E[(y - E[y])^2] + (E[f] - E[y])^2 + E[(f - E[f])^2] \\
&= E[y^2] - 2E[y]^2 + E[y]^2 + E[f]^2 - 2E[f]E[y] + E[y]^2 + E[f^2] - 2E[f]^2 + E[f]^2 \\
&= E[y^2] - 2E[f]E[y] + E[f^2] \\
&= E[y^2 - 2fy + f^2]
\end{aligned}$$

This decomposition takes advantage of a sort of coincidence in the squared error. However, it helps to motivative a more informal idea, the **bias-variance tradeoff**. This is commonly thought of by drawing a picture like this:



This shows the bias, variance, and riskiness of a range of different learning methods, for a given amount of training data. It shows the common situation in practice that (1) for simple models, the bias increases very quickly, while (2) for complex models, the variance increases very quickly. Since the riskiness is additive in the two, the optimal complexity is somewhere in the middle. Note, however, that these properties do *not* follow from the bias-variance decomposition, and need not even be true.

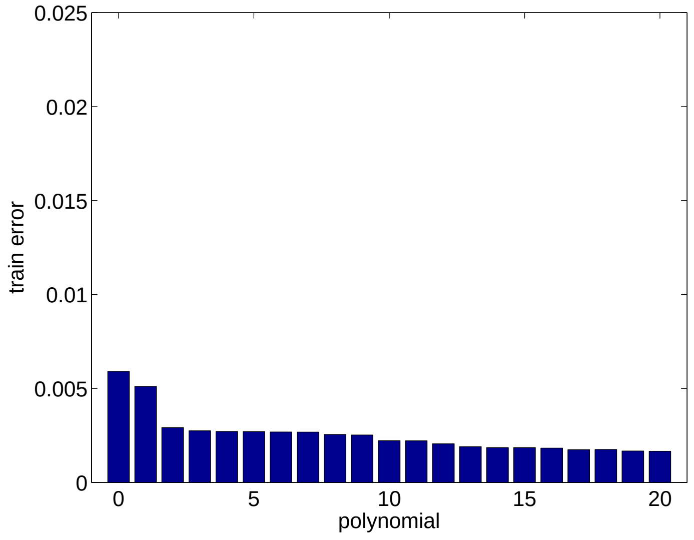
There has been some effort to create more general bias-variance decompositions that apply to, e.g., classification, but there is not consensus so far. (In your lecturer's opinion, there is no particular reason to think that risk should always be a *linear* function of bias and variance.) However, bias-variance *tradeoffs* are seen very frequently, in all sorts of problems. We can very often understand the differences in performance between different algorithms as trading off between bias and variance. Usually, if we take some algorithm, and change it to reduce bias, we will also increase variance. This doesn't **have** to happen, though. If you work hard, you can change an algorithm in such a dopey way as to increase both bias and variance.

3 Cross Validation

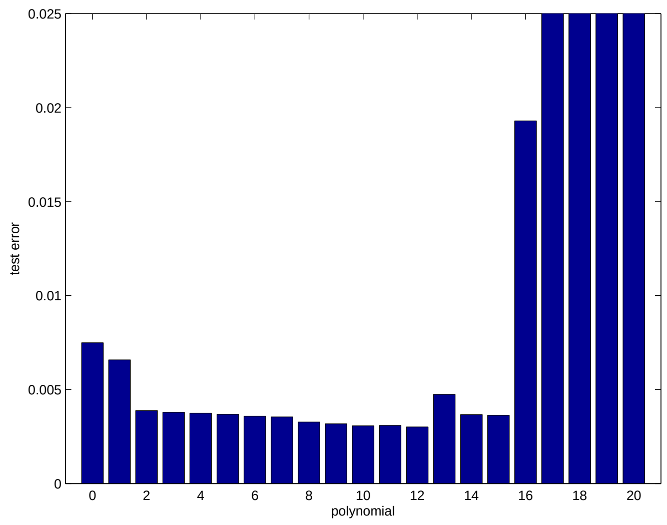
Let us return to our initial problem of trying to pick the right degree p for our polynomial. The first idea that springs to mind is to pick that p that fits the data best. If we plot

$$\underset{(\hat{y},\hat{x})\in D}{\text{mean}} \big(w_0 + w_1\hat{x} + w_2\hat{x}^2 + ... + w_p\hat{x}^p - \hat{y}\big)^2.$$

for each p, we see something disturbing:



Clearly this is not what we want. Instead, suppose we had a giant pile of 100,000 extra point drawn from the same distribution as the training error. We will call this **test data**, and the average error on it the **test error**.



Things blow up after $p = 13$. (This is intuitively plausible if you look at the plots of the polynomials above.)

Now, what would you do if you don't happen to have a spare 100,000 extra data sitting around? The first idea that comes to mind is to "hold out" some of our original training data.

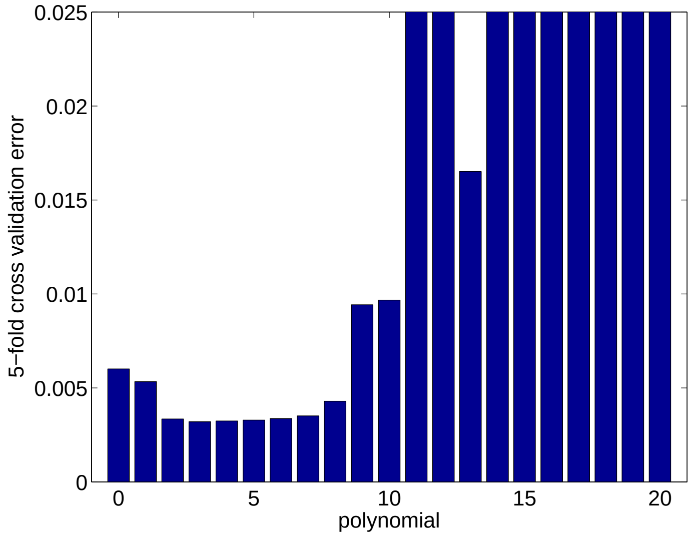
1. Split D into D$_{\text{train}}$ and D_{test}. (e.g. half in each)
2. Fit each model to D_{train}, and evaluate how well it does on D_{test}
3. Output the model that has the best score on D_{test}.

This can work reasonably well, but it "wastes" the data by only training on half, and only testing on half. We can make better use of the data by making several different splits of the data. Each datum is used once for testing, and the other times for training. This algorithm is called **K-fold cross validation**.

1. Split D into K chunks
2. For $k = 1, 2, ..., K$:
   (a) Set D_{test} to be the kth chunk of data, and D_{train} to be the other $K - 1$ chunks.
   (b) Fit each model to D_{train} and evaluate how well it does on D_{test}.

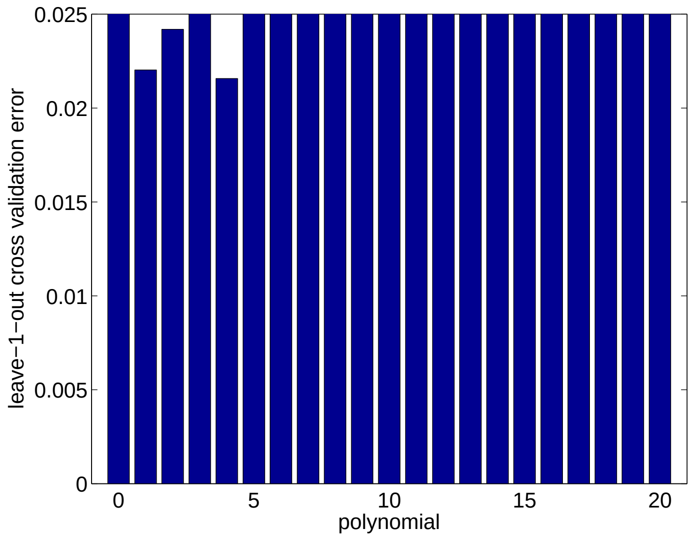
3. Pick the model that has the best average test score.
4. Retrain that model on all of D, and output that.

If we do 5-fold cross validation on our original set of 60 points, we get:



We can see that this picks $p = 3$, rather than the optimal $p = 12$. Clearly, cross validation is no substitute for a large test set. However, if we only have a limited training set, it is often the best option available.

In this case, cross-validation selected a simpler model than optimal. What do we expect to happen on average? Notice that with 60 points, 5-fold cross validation effectively tries to pick the polynomial that makes the best bias-variance tradeoff for 48 $(60 \cdot \frac{4}{5})$ points. If we had done 10-fold cross validation, it would instead try to pick the best polynomial for 54 $(60 \cdot \frac{9}{10})$ points. Thus, cross validation *biases towards simpler models.* We could reduce this to the minimum possible by doing 60-fold cross validation. This has a special name: **leave-one-out cross validation**. In complex models, this can be expensive, though there has been research on clever methods for doing leave-one-out cross validation with out needing to recompute the full model. In practice, we usually don't see too much benefit for doing more then 5 or 10 fold cross validation. In this particular example, 60-fold cross validation still selects $p = 3$.



Cross validation is a good technique, but it doesn't work miracles: there is only so much information in a small dataset.

Though cross validation is extremely widely used, there has been difficulty proving that it actually works better than just using a single hold-out set of $\frac{1}{K}$th of the data! There is even a \$500 bounty. See: http://hunch.net/?p=29 http://hunch.net/?p=320