MIT OpenCourseWare
http://ocw.mit.edu

6.00 Introduction to Computer Science and Programming, Fall 2008
Transcript – Lecture 21

PROFESSOR: So let's start. I have written a number on the board here. Anyone want to speculate what that number represents? Well, you may recall at the end of the last lecture, we were simulating pi, and I started up running it with a billion darts. And when it finally terminated, this was the estimate of pi it gave me with a billion. Not bad, not quite perfect, but still pretty good. In fact when I later ran it with 10 billion darts, which took a rather long time to run, didn't do much better. So it's converging very slowly now near the end.

When we use an algorithm like that one to perform a Monte Carlo simulation, we're trusting, as I said, that fate will give us an unbiased sample, a sample that would be representative of true random throws. And, indeed in this case, that's a pretty good assumption. The random number generator is not truly random, it's what's called pseudo-random, in that if you start it with the same initial conditions, it will give you the same results. But it's close enough for, at least for government work, and other useful projects. We do have to think about the question, how many samples should we run? Was a billion darts enough? Now since we sort of all started knowing what pi was, we could look at it and say, yeah, pretty good. But suppose we had no clue about the actual value of pi. We still have to think about the question of how many samples? And also, how accurate do we believe our result is, given the number of samples? As you might guess, these two questions are closely related. That, if we know in advance how much accuracy we want, we can sometimes use that to calculate how many samples we need.

But there's still always the issue. It's never possible to achieve perfect accuracy through sampling. Unless you sample the entire population. No matter how many samples you take, you can never be sure that the sample set is typical until you've checked every last element. So if I went around MIT and sampled 100 students to try and, for example, guess the fraction of students at MIT who are of Chinese descent. Maybe 100 students would be enough, but maybe I would get unlucky and draw the wrong 100. In the sense of, by accident, 100 Chinese descent, or 100 non-Chinese descent, which would give me the wrong answer. And there would be no way I could be sure that I had not drawn a biased sample, unless I really did have the whole population to look at.

So we can never know that our estimate is correct. Now maybe I took a billion darts, and for some reason got really unlucky and they all ended up inside or outside the circle. But what we can know, is how likely it is that our answer is correct, given the assumptions. And that's the topic we'll spend the next few lectures on, at least one of the topics. It's saying, how can we know how likely it is that our answer is good. But it's always given some set of assumptions, and we have to worry a lot about

those assumptions. Now in the case of our pi example, our assumption was that the random number generator was indeed giving us random numbers in the interval to 1. So that was our underlying assumption. Then using that, we looked at a plot, and we saw that after time the answer wasn't changing very much. And we use that to say, OK, it looks like we're actually converging on an answer. And then I ran it again, with another trial, and it converged again at the same place. And the fact that that happened several times led me to at least have some reason to believe that I was actually finding a good approximation of pi.

That's a good thing to do. It's a necessary thing to do. But it is not sufficient. Because errors can creep into many places. So that kind of technique, and in fact, almost all statistical techniques, are good at establishing, in some sense, the reproduce-ability of the result, and that it is statistically valid, and that there's no error, for example, in the way I'm generating the numbers. Or I didn't get very unlucky. However, they're other places other than bad luck where errors can creep in.

So let's look at an example here. I've taken the algorithm we looked at last time for finding pi, and I've made a change. You'll remember that we were before using 4 as our multiplier, and here what I've done is, just gone in and replaced 4 by 2. Assuming that I made a programming error. Now let's see what happens when we run it. Well, a bad thing has happened. Sure enough, we ran it and it converged, started to converge, and if I ran 100 trials each one would converge at roughly the same place. Any statistical test I would do, would say that my statistics are sound, I've chosen enough samples, and for some accuracy, it's converting. Everything is perfect, except for what? It's the wrong answer. The moral here, is that just because an answer is statistically valid, does not mean it's the right answer.

And that's really important to understand, because you see this, and we'll see more examples later, not today, but after Thanksgiving, comes up all the time in the newspapers, in scientific articles, where people do a million tests, do all the statistics right, say here's the answer, and it turns out to be completely wrong. And that's because it was some underlying assumption that went into the decision, that was not true. So here, the assumption is, that I've done my algebra right for computing pi based upon where the darts land. And it turns out, if I put 2 here, my algebra is wrong. Now how could I discover this? Since I've already told you no statistical test is going to help me. What's the obvious thing I should be doing when I get this answer? Somebody? Yeah?

STUDENT: [INAUDIBLE]

PROFESSOR: Exactly. Checking against reality. I started with the notion that pi had some relation to the area of a circle. So I could use this value of pi, draw a circle with a radius. Do my best to measure the area. I wouldn't need to get a very good, accurate measurement, and I would say, whoa, this isn't even close. And that would tell me I have a problem. So the moral here is, to check results against physical reality. So for example, the current problem set, you're doing a simulation about what happens to viruses when drugs are applied. If you were doing this for a pharmaceutical company, in addition to the simulation, you'd want to run some real experiments. And make sure that things matched. OK, what this suggests, is that we often use simulation, and other computational techniques, to try and model the real world, or the physical world, in which we all live. And we can use data to do that. I now want to go through another set of examples, and we're going to look at the

interplay of three things: what happens when you have data, say from measurements, and models that at least claim to explain the data. And then, consequences that follow from the models. This is often the way science works, its the way engineering works, we have some measurements, we have a theory that explains the measurements, and then we write software to explore the consequences of that theory. Including, is it plausible that it's really true?

So I want to start, as an example, with a classic chosen from 8.01. So I presume, everyone here has taken 8.01? Or in 8.01? Anyone here who's not had an experience with 801? All right, well. I hope you know about springs, because we're going to talk about springs. So if you think about it, I'm now just talking not about springs that have water in them, but springs that you compress, you know, and expand, and things like that. And there's typically something called the spring constant that tells us how stiff the spring is, how much energy it takes to compress this spring. Or equivalently, how much pop the spring has when you're no longer holding it down. Some springs are easy to stretch, they have a small spring constant. Some strings, for example, the ones that hold up an automobile, suspension, are much harder to stretch and compress.

There's a theory about them called Hooke's Law. And it's quite simple. Force, the amount of force exerted by a spring, is equal to minus some constant times the distance you have compressed the spring. It's minus, because the force is exerted in an opposite direction, trying to spring up. So for example, we could look at it this way. We've got a spring, excuse my art here. And we put some weight on the spring, which has therefore compressed it a little bit. And the spring is exerting some upward force. And the amount of force it's exerting is proportional to the distance x. So, if we believe Hooke's Law, and I give you a spring, how can we find out what this constant is? Well, we can do it by putting a weight on top of the spring. It will compress the spring a certain amount, and then the spring will stop moving. Now gravity would normally have had this weight go all the way down to the bottom, if there was no spring. So clearly the spring is exerting some force in the upward direction, to keep that mass from going down to the table, right? So we know what that force is there. If we compress the spring to a bunch of different distances, by putting, say, different size weights on it, we can then solve for the spring constant, just the way, before, we solved for pi.

So it just so happens, not quite by accident, that I've got some data from a spring. So let's look at it. So here's some data taken from measuring a spring. This is distance and force, force computed from the mass, basically, right? Because we know that these have to be in balance. And I'm not going to ask you to in your head estimate the constant from these, but what you'll see is, the format is, there's a distance, and then a colon, and then the force. Yeah?

STUDENT: [INAUDIBLE]

PROFESSOR: Ok, right, yes, thank you. All right, want to repeat that more loudly for everyone?

STUDENT: [INAUDIBLE]

PROFESSOR: Right, right, because the x in the equation -- right, here we're getting an equilibrium. OK, so let's look at what happens when we try and examine this. We'll look at spring dot pi. So it's pretty simple. First thing is, I've got a function that

reads in the data and parses it. You've all done more complicated parsing of data files than this. So I won't belabor the details. I called it get data rather than get spring data, because I'm going to use the same thing for a lot of other kinds of data. And the only thing I want you to notice, is that it's returning a pair of arrays. OK, not lists. The usual thing is, I'm building them up using lists, because lists have append and arrays don't, and then I'm converting them to arrays so I can do matrix kinds of operations on them.

So I'll get the distances and the forces. And then I'm just going to plot them, and we'll see what they look like. So let's do that. There they are. Now, if you believe Hooke's Law, you could look at this data, and maybe you wouldn't like it. Because Hooke's Law implies that, in fact, these points should lie in a straight line, right? If I just plug in values here, what am I going to get? A straight line, right? I'm just multiplying k times x. But I don't have a straight line, I have a little scatter of points, kind of it looks like a straight line, but it's not. And why do you think that's true? What's going on here? What could cause this line not to be straight? Have any you ever done a physics experiment? And when you did it, did your results actually match the theory that your high school teacher, say, explained to you. No, and why not. Yeah, you have various kinds of experimental or measurement error, right? Because, when you're doing these experiments, at least I'm not perfect, and I suspect at least most of you are not perfect, you get mistakes. A little bit of error creeps in inevitably. And so, when we acquired this data, sure enough there was measurement error. And so the points are scattered around.

This is something to be expected. Real data almost never matches the theory precisely. Because there usually is some sort of experimental error that creeps into things. So what should we do about that? Well, what usually people do, when they think about this, is they would look at this data and say, well, let me fit a line to this. Somehow, say, what would be the line that best approximates these points? And then the slope of that line would give me the spring constant.

So that raises the next question, what do I mean by finding a line that best fits these points? How do we, fit, in this case, a line, to the data? First of all, I should ask the question, why did I say let's fit a line? Maybe I should have said, let's fit a parabola, or let's fit a circle? Why should I had said let's fit a line. Yeah?

STUDENT: [INAUDIBLE]

PROFESSOR: Well, how do I know that the plot is a linear function? Pardon? Well, so, two things. One is, I had a theory. You know, I had up there a model, and my model suggested that I expected it to be linear. And so if I'm testing my model, I should and fit a line, my theory, if you will. But also when I look at it, it looks kind of like a line. So you know, if I looked at it, and it didn't look like a line, I might have said, well, my model must be badly broken. So let's try and see if we can fit it. Whenever we try and fit something, we need some sort of an objective function that captures the goodness of a fit. I'm trying to find, this is an optimization problem of the sort that we've looked at before. I'm trying to find a line that optimizes some objective function. So a very simple objective function here, is called the least squares fit. I want to find the line that minimizes the sum of observation sub i, the i'th data point I have, minus what the line, the model, predicts that point should have been, and then I'll square it. So I want to minimize this value. I want to find the line that gives me the smallest value for this.

Why do you think I'm squaring the difference? What would happen if I didn't square the difference? Yeah? Positive and negative errors might cancel each other out. And in judging the quality of the fit, I don't really care deeply -- you're going to get very fat the way you're collecting candy here -- I don't care deeply whether the error is, which side, it is, just that it's wrong. And so by squaring it, it's kind of like taking the absolute value of the error, among other things. All right, so if we look at our example here, what would this be? I want to minimize, want to find a line that minimizes it. So how do I do that? I could easily do it using successive approximation, right? I could choose a line, basically what I am, is I'm choosing a slope, here, right? And, I could, just like Newton Raphson, do successive approximation for awhile, and get the best fit. That's one way to do the optimization. It turns out that for this particular optimization there's something more efficient. You can actually, there is a closed form way of attacking this, and I could explain that, but in fact, I'll explain something even better. It's built into Pylab.

So Pylab has a function built-in called polyfit. Which, given a set of points, finds the polynomial that gives you the best least squares approximation to those points. It's called polynomial because it isn't necessarily going to be first order, that is to say, a line. It can find polynomials of arbitrary degree. So let's look at the example here, we'll see how it works. So let me uncomment it. So I'm going to get k and b equals Pylab dot polyfit here. What it's going to do is, think about a polynomial. I give you a polynomial of degree one, you have all learned that it's a x plus b, b is the constant, and x is the single variable. And so I multiply a by x and I add b to it, and as I vary x I get new values. And so polyfit, in this case, will take the set of points defined by these two arrays and return me a value for a and a value for b. Now here I've assigned a to k, but don't worry about that. And then, I'm gonna now generate the predictions that I would get from this k and b, and plot those.

So let's look at it. So here it said the k is 31.475, etc., and it's plotted the line that it's found. Or I've plotted the line. You'll note, a lot of the points don't lie on the line, in fact, most of the points don't lie on the line. But it's asserting that this is the best it can do with the line. And there's some points, for example, up here, that are kind of outliers, that are pretty far from the line. But it has minimized the error, if you will, for all of the points it has. That's quite different from, say, finding the line that touches the most points, right? It's minimizing the sum of the errors. Now, given that I was just looking for a constant to start with, why did I bother even plotting the data? I happen to have known before I did this that polyfit existed, and what I was really looking for was this line. So maybe I should have just done the polyfit and said here's k and I'm done. Would that have been a good idea? Yeah?

STUDENT: You can't know without seeing the actual data how well it's actually fitting it.

PROFESSOR: Right. Exactly right. That says, well how would I know that it was fitting it badly or well, and in fact, how would I know that my notion of the model is sound, or that my experiment isn't completely broken? So always, I think, always look at the real data. Don't just, I've seen too many papers where people show me the curve that fits the data, and don't show me the data, and it always makes me very nervous. So always look at the data, as well as however you're choosing to fit it.

As an example of that, let's look at another set of inputs. This is not a spring. It's the same get data function as before, ignore that thing at the top. I'm going to analyze it

and we'll look at it. So here I'm plotting the speed of something over time. So I plotted it, and I've done a least squares fit using polyfit just as before to get a line, and I put the line vs. the data, and here I'm a little suspicious. Right, I fit a line, but when I look at it, I don't think it's a real good fit for the data. Somehow modeling this data as a line is probably not right. A linear model is not good for this data. This data is derived from something, a more complex process. So take a look at it, and tell me what order were calling of polynomial do you think might fit this data? What shape does this look like to you? Pardon?

STUDENT: Quadratic.

PROFESSOR: Quadratic, because the shape is a what? It's a parabola. Well, I don't know if I dare try this one all the way to the back. Ooh, at least I didn't hurt anybody. All right, fortunately it's just as easy to fit a ravel parabola as a line. So let's look down here. I've done the same thing, but instead of passing it one, as I did up here as the argument, I'm passing it two. Saying, instead of fitting a polynomial of degree one, fit a polynomial of degree two. And now let's see what it looks like. Well, my eyes tell me this is a much better fit than the line. So again, that's why I wanted to see the scatter plot, so that I could at least look at it with my eyes, and say, yeah, this looks like a better fit. All right, any question about what's going on here?

What we've been looking at is something called linear regression. It's called linear because the relationship of the dependent variable y to the independent variables is assumed to be a linear function of the parameters. It's not because it has to be a linear function of the value of x, OK? Because as you can see, we're not getting a line, we're getting a parabola. Don't worry about the details, the point I want to make is, people sometimes see the word linear regression and think it can only be used to find lines. It's not so. So when, for example, we did the quadratic, what we had is y equals a x squared plus b x plus c. The graph vs. x will not be a straight line, right, because I'm squaring x. But it is, just about, in this case, the single variable x. Now, when I looked at this, I said, all right, it's clear that the yellow curve is a better fit than the red. It's a red line. But that was a pretty informal statement.

I can actually look at this much more formally. And we're going to look at something that's the statisticians call r squared. Which in the case of a linear regression is the coefficient of determination. Now, this is a big fancy word for something that's actually pretty simple. So what r squared its going to be, and this is on your handout, is 1 minus e e over d v. So e e is going to be the errors in the estimation. So I've got some estimated values, some predicted values, if you will, given to me by the model, either the line or the parabola in this case. And I've got some real values, corresponding to each of those points, and I can look at the difference between the 2 And that will tell me how much difference there is between the estimated data and the, well, between the predicted data and the measured data, in this case. And then I want to divide that by the variance in the measured data. The data variance. How broadly scattered the measured points are. And I'll do that by comparing the mean of the measured data, to the measured data. So I get the average value of the measured data, and I look at how different the points I measure are.

So I just want to give to you, informally, because I really don't care if you understand all the math. What I do want you to understand, when someone tells you, here's the r squared value, is, informally what it really is saying. It's attempting

to capture the proportion of the response variation explained by the variables in the model. In this case, x. So you'll have some amount of variation that is explained by changing the values of the variables. So if, actually, I'm going to give an example and then come back to it more informally. So if, for example, r squared were to equal 0.9, that would mean that approximately 90 percent of the variation in the variables can be explained by the model. OK, so we have some amount of variation in the measured data, and if r squared is 0.9, it says that 90 percent can be explained by the models, and the other 10 percent cannot. Now, that other 10 percent could be experimental error, or it could be that, in fact, you need more variables in the model. That there are what are called lurking variables. I love this term. A lurking variable is something that actually effects the result, but is not reflected in the model.

As we'll see a little bit later, this is a very important thing to worry about, when you're looking at experimental data and you're building models. So we see this, for example, in the medical literature, that they will do some experiment, and they'll say that this drug explains x, or has this affect. And the variables they are looking at are, say, the disease the patient has, and the age of the patient. Well, maybe the gender of the patient is also important, but it doesn't happen to be in the model. Now, if when they did a fit, it came out with 0.9, that says at worst case, the variables we didn't consider could cause a 10 percent error. But, that could be big, that could matter a lot. And so as you get farther from 1, you ought to get very worried about whether you actually have all the right variables. Now you might have the right variables, and just experiment was not conducted well, But it's usually the case that the problem is not that, but that there are lurking variables. And we'll see examples of that.

So, easier to read than the math, at least by me, easier to read than the math, is the implementation of r square. So it's measured and estimated values, I get the diffs, the differences, between the estimated and the measured. These are both arrays, so I subtract 1 array from the other, and then I square it. Remember, this'll do an element-wise subtraction, and then square each element. Then I can get the mean, by dividing the sum of the array measured by the length of it. I can get the variance, which is the measured mean minus the measured value, again squared. And then I'll return 1 minus this. All right? So, just to make sure we sort of understand the code, and the theory here as well, what would we get if we had absolutely perfect prediction? So if every measured point actually fit on the curb predicted by our model, what would r square return? So in this case, measured and estimated would be identical. What gets return by this? Yeah, 1. Exactly right. Because when I compute it, it will turn out that these two numbers will be the, I'll get 0, 1 minus is 0, right? Because the differences will be zero. OK?

So I can use this, now, to actually get a notion of how good my fit is. So let's look at speed dot pi again here, and now I'm going to uncomment these two things, where I'm going to, after I compute the fit, I'm going to then measure it. And you'll see here that the r squared error for the linear fit is 0.896, and for the quadratic fit is 0.973. So indeed, we get a much better fit here. So not only does our eye tell us we have a better fit, our more formal statistical measure tells us we have a better fit, and it tells us how good it is. It's not a perfect fit, but it's a pretty good fit, for sure.

Now, interestingly enough, it isn't surprising that the quadratic fit is better than the linear fit. In fact, the mathematics of this should tell us it can never be worse. How do I know it can never be worse? That's just, never is a really strong word. How do I

know that? Because, when I do the quadratic fit, if I had perfectly linear data, then this coefficient, whoops, not that coefficient, wrong, this coefficient, could be 0. So if I ask it to do a quadratic fit to linear data, and the a is truly perfectly linear, this coefficient will be 0, and my model will turn out to be the same as the linear model. So I will always get at least as good a fit.

Now, does this mean that it's always better to use a higher order polynomial? The answer is no, and let's look at why. So here what I've done is, I've taken seven points, and I've generated, if you look at this line here, the y-values, for x in x vals, points dot append x plus some random number. So basically I've got something linear in x, but I'm perturbing, if you will, my data by some random value. Something between and 1 is getting added to things. And I'm doing this so my points won't lie on a perfectly straight line. And then we'll try and fit a line to it. And also, just for fun, we'll try and fit a fifth order polynomial to it. And let's see what we get. Well, there's my line, and there's my fifth order polynomial. Neither is quite perfect, but which do you think looks like a closer fit? With your eye. Well, I would say the red line, the red curve, if you will, is a better fit, and sure enough if we look at the statistics, we'll see it's 0.99, as opposed to 0.978. So it's clearly a closer fit.

But that raises the very important question: does closer equal better, or tighter, which is another word for closer? And the answer is no. It's a tighter fit, but it's not necessarily better, in the sense of more useful. Because one of the things I want to do when I build a model like this, is have something with predictive power. I don't really necessarily need a model to tell me where the points I've measured lie, because I have them. The whole purpose of the model is to give me some way to predict where unmeasured points would lie, where future points would lie. OK, I understand how the spring works, and I can guess where it would be if things I haven't had the time to measure, or the ability to measure. So let's look at that. Let's see, where'd that figure go. It's lurking somewhere. All right, we'll just kill this for now.

So let's generate some more points, and I'm going to use exactly the same algorithm. But I'm going to generate twice as many points. But I'm only fitting it to the first half. So if I run this one, figure one is what we looked at before. The red line is fitting them a little better. But here's figure two. What happens when I extrapolate the curve to the new points? Well, you can see, it's a terrible fit. And you would expect that, because my data was basically linear, and I fit in non-linear curve to it. And if you look at it you can see that, OK, look at this, to get from here to here, it thought I had to take off pretty sharply. And so sure enough, as I get new points, the prediction will postulate that it's still going up, much more steeply than it really does. So you can see it's a terrible prediction.

And that's because what I've done is, I over-fit the data. I've taken a very high degree polynomial, which has given me a good close fit, and I can always get a fit, by the way. If I choose a high enough degree polynomial, I can fit lots and lots of data sets. But I have reason to be very suspicious. The fact that I took a fifth order polynomial to get six points should make me very nervous. And it's a very important moral. Beware of over-fitting. If you have a very complex model, there's a good chance that it's over-fit. The larger moral is, beware of statistics without any theory. You're just cranking away, you get a great r squared, you say it's a beautiful fit. But there was no real theory there. You can always find a fit. As Disraeli is alleged to have said, there are three kinds of lies: lies, damned lies, and statistics. And we'll

spend some more time when we get back from Thanksgiving looking at how to lie with statistics. Have a great holiday, everybody.