PROFESSOR: So you'll recall last time we were working on protein-protein interactions. We're going to do a little bit to finish that up, with a topic that will be a good transition to the study of the gene regulatory networks. And the precise things we're going to discuss today, we're going to start off with Bayesian networks of protein-protein interaction prediction. And then we're going to get into gene expression data, at several different levels. We'll talk about some basic questions, of how to compare the two expression vectors for a gene, distance metrics. We'll talk about how to cluster gene expression data. The idea of identifying signatures of sets of genes, that might be predictive of some biological property. For example, a susceptibility to a disease.

And then we'll talk about a number of different ways that people have developed to try to identify gene regulatory networks. That often goes by the name of modules. I don't particularly like that name. But that's what you'll find in the literature. And we're going to focus on a few of these, that have recently been compared head to head, using both synthetic and real data. And we'll see some of the results from that head to head comparison.

So let's just launch into it. Remember last time we had started this unit looking at the structural predictions for proteins. And we started talking about how to predict protein-protein interactions. Last time we talked about both computational methods, and also experimental data, that could give us information about protein-protein interactions. Ostensibly measuring direct interactions, but we saw that there were possibly very, very high error rates. So we needed ways of integrating lots of different kinds of data in a probabilistic framework so we could predict for any pair proteins what's the probability that they interact. Not just the fact that they were detected in one assay or the other.

And we started to talk about Bayesian networks in this context. Both useful as we'll see today, for predicting protein-protein interactions, and also for the gene regulatory network problem. So the Bayesian networks are a tool for reasoning probabilistically. That's the fundamental purpose. And we saw that they consisted of a graph, the network. And then the probabilities that represent the probability for each edge, the conditional probability tables. And that we can learn these from the data, either in a completely objective way, where we learn both the structure and the probability. Or where we impose the structure initially, and then we simply learn the probability tables.

And we had nodes that represented the variables. They could be hidden nodes, where we don't know what the true answer is, and observed nodes, where we do. So in our case, we're trying to predict protein-protein interactions. There's some hidden variable that represents weather protein A and B truly interact. We don't know that answer. But we do know whether that interaction was detected in an experiment one, two, three or 4. Those are the effects, the observed. And so we want to reason backwards from the observations, to the hidden causes.

So last time we talked about the high throughput experiments, that directly we're measuring out protein-protein interactions. We talked about yeast two hybrid and affinity capture mass spec-- here listed as pull-downs. And those could be used to predict protein-protein directions, by themselves. But we want to find out what other kinds of data we can use to amplify these results, to give us independent information about whether two proteins interact.

And one thing you could look at is whether the expression of the two genes that you think might interact are similar. So if you look over many, many different conditions, you might expect the two proteins that interact with each other, would be expressed under similar conditions. Certainly if you saw two proteins that had exactly opposite expression patterns, you would be very unlikely to believe that they interacted.

So the question is, how much is true at the other end of the spectrum? If things are very highly correlated, do they have a high probability of interaction? So this graph

is a histogram for proteins that are known to interact, proteins that were shown in these high throughput experiments to interact, and proteins that are known not to interact, of how similar the expression is. On the far right are things that have extremely different expression patterns, a high distance. And we'll talk specifically about what distance is in just a minute. But these are very dissimilar expression patterns. These are very similar ones.

So what do you see from this plot we looked at the last time? We saw that the interacting proteins are shifted a bit to the left. So the interacting ones have a higher probability of having similar expression patterns than the ones don't interact. But we couldn't draw any cut off, and say everything with this level expression similarity is guaranteed to interact. There's no way to divide these. So this will be useful in a probabilistic setting. But by itself, it would not be highly predictive.

We also talked about evolutionary patterns, and we discussed whether the red or the green patterns here, would be more predictive. And which one was it, anyone remember? How many people thought the red was more predictive? How many the green? Right, the greens win. And we talked about the coevolution in other ways.

So the paper that, I think, was one of the first to do this really nicely, try to predict protein-protein interaction patterns using Bayesian networks, is this one from Mark Gerstein's lab. And they start off as we talked about previously, we need some gold standard interactions, where we know two proteins really do interact or don't. They built their gold standard data set. The positive trending data, they took from a database called MIPS, which is a hand-curated database that digs into the literature quite deeply, to find out whether two proteins interact or not. And then the negative data they took were proteins that were identified as being localized to different parts of the cell. And this was done in yeast, to where there is pretty good data for a lot of proteins, to subcellular localization.

So these are the data that went into their prediction. These were the experiments we've already talked about, the affinity capture mass spec and the yeast two hybrid. And then the other kinds of data they used were expression correlation, one just

talked about. They also looked at annotations, whether proteins had the same annotation for function. And essentiality. So in yeast, it's pretty easy to go through every gene in the genome, knock it out, and determine whether that kills the cell or not. So they can label every gene in yeast, as to whether it's essential for survival or not.

And you can see here, the number of interactions that were involved. And they decided to break this down into two separate prediction problems. So one was an experimental problem, using the four different large scale data sets in yeast from protein-protein interactions, to predict expression. The other one wore these other kinds of data, that were less direct. And they used slightly different kinds of Bayesian networks. So for this one, they used a naive Bayes. And what's the underlying assumption of the naive Bayes? The underlying assumption is that all the data are independent. So we looked at this previously. We discussed how you could, if you're trying to identify the likelihood ratio, and use it to rank things. You primarily need to focus on this term. Because this term will be the same for every pair of proteins that you're examining. Yes?

AUDIENCE: Could you state again whether in a naive Bayes, all data are dependent or independent?

PROFESSOR: Independent.

AUDIENCE: OK.

PROFESSOR: OK. So let's actually look at some of their data. So in this table, they're looking at the likelihood ratio that two proteins interact, based on whether the two proteins are essential. One is essential, and one is a nonessential. Both are nonessential. So that's what these two codes here mean. EE, both essential. NN, both nonessential, and any one and the other. And so they've computed for all those protein pairs, how many in their gold standard, are EE, how many are EN, how many are NN?

So here are the numbers for the EE. There are just over 1,000, out of the 2,000, roughly 2,000 that are EE. So that comes up with a probability of being essential,

given that I know that you're positive. You're in the gold standard of roughly 50%, right? And you can assume something similar for the negatives. So these are the ones that definitely don't interact. So the probability of both being essential, given that it's negative, is about 15%, 14%. And so then the likelihood ratio comes out to just under four. So there's a fourfold increase in probability that something is interacting, given that it's essential, then not.

And this is the table for all of the terms, for all of the different things that they were considering, that were not direct experiments. So this is the sensuality. This is expression correlation, with various values for the threshold, how similar the expression had to be. And these are the terms from the databases for annotation. And then for each of these, then we get a likelihood ratio of how predictive it is. So it's kind of informative to look at some of these numbers. We already saw that essentiality is pretty weak, predicted the fact that two genes are essential. It only gives you a slightly increased chance that they're interacting than not. But if two things, two genes have extremely high expression correlation, then they're more than a hundredfold more likely to interact than not.

And the numbers for the annotations are significantly less than that. So this is a naive Bayes. We're going to multiply all those probabilities together. Now for the experimental data, they said, well, these are probably not all independent. The probably that you pick something up in one two hybrid experiment, is probably highly correlated with the probability that you pick it up in another two hybrid experiment. And one would hope that there's some correlation between things are identifying in two hybrid and affinity caption mass spec. Although we'll see whether or not that's the case.

So they used what they refer to as a fully connected Bayes. And what do we mean by that? Remember, this was the naive Bayes, where everything is independent. So the probability of some observation is the product of all the individual probabilities.

But in a fully connected Bayes, we don't have that independence assumption. So you need to actually explicitly compute what the probability is for an interaction,

based on all the possible outcomes in those experiments. So that's not that much harder.

We simply have a table now, where these columns represent each of the experimental data types-- the affinity capture mass spec and the two hybrids. Ones indicate that it was detected, Zero is that it's not. And then we simply look again in our gold standard, and see how often a protein that had been detected in whatever the setting is here, in all of them except Ito, how often was it, how many of the gold positives do we get? And how many of the gold negatives? And then we can compute the probabilities.

Now it's important to look at some of the numbers in these tables and dig in. Because you'll see the numbers here are really, really small. So they have to be interpreted with caution. So some of the things that might not hold up with much larger data sets. You might imagine the things that are experimentally detected in all of the high-throughput assays would be the most confident. That doesn't turn out to be the case.

So these are sorted by the law of likelihood ratio, and the best one is not 1, 1, 1. It's up there. But it's not the top of the pack. And that's probably just the statistics of small numbers. If the databases were larger, experiments were larger, it probably would work out that way. So any question about how they formulated this problem, as a Bayesian network, or how they implemented it? OK.

So the results then-- so once we have these likelihood ratios, we can try to choose a threshold for deciding what we're going to consider to be a true interaction and not. So here they've plotted for different likelihood ratio thresholds. On the x-axis, how many of the true positives you get right, versus how many you get wrong. So the true positive over the false positive. And you can arbitrarily decide, OK, well I want to be more-- I want to get more right than wrong. Not a bad way to decide things. So your passing grade here is 50%.

So if I draw a line, a horizontal line, and wanted to get more right than wrong, you'll see that any of the individual signals that they were using, essentiality, database

sanitation, and so on-- all of those fall below that. So individually, they predict more wrongs than rights. But if you combine the data using this Bayesian network, then you can choose a likelihood threshold, where you do get more right than wrong. And you can set your threshold wherever you want. Similarly for the direct experimental data, you do better by combining-- these are light pink lines, than you would with any of the individual data sets.

So this shows the utility of combining the data, and reasoning from the data probabilistically. Any questions? So we'll return to Bayesian networks in a bit in the context of discovering gene regulatory networks.

So we now want to move to gene expression data. And the primary reason to be so interested in gene expression data is simply that there's a huge amount of it out there. So just a short time ago we passed the million mark, with a number of expression data sets that had been collected in the databases. There's much less of any other kind of high throughput data. So if you look at proteomics or high-throughput genetic screens, there are tiny numbers, compared to gene expression data. So obviously techniques for analyzing gene expression data are going to play a very important role for a long time to come.

Some of what I'm going to discuss today is covered in your textbooks. I encourage you to look at text section 16.2. The fundamental thing that we're interested in doing, is seeing how much biological knowledge we can infer from the gene expression data. So we might imagine that genes that are coexpressed under particular sets and conditions, have functional similarity, reflect common regulatory mechanisms, and our goal then, is to discover those mechanisms. So fundamental to this then, any time we have a pair of genes-- and we look at their gene expression data-- we want to decide how similar they are.

So let's imagine that we had these data for four genes. And it's a time series experiment. And we're looking at the different expression levels. And we want some quantitative measure to decide which two genes are most similar. Well, it turns out it's a lot more subtle than we might think. So at first glance, oh, it's pretty obvious

that these two are the most similar. But it really depends on what kind of similarity you're asking about.

So we can describe any expression data set for any gene, is simply a multi-dimensional vector. Where this is the set of expression values we detected for the first gene, across all the different experimental conditions and so on, for the second. And what would be the most intuitive way of describing the distance between two multi-dimensional vectors? It would simply be Euclidean distance, right? So that's perfectly reasonable.

So we can decide that the distance between two gene expression data sets, is simply the square root of the sum of the squares of the distances. So we'll take the sum over all the experimental conditions that we've looked at. Maybe it's a time series. Maybe it's different perturbations. And look at the difference in expression of gene A and gene B in that condition, K. And then evaluating this will tell us how similar two genes are in their expression profiles.

Well, that's a specific example of a distance metric. It turns out that there's a formal definition for a distance metric. Distances have the following properties. They're always greater than zero. We never have negative distances. They are equal to zero under exactly one condition-- the two data points are the same. And they're symmetric. So the distance from A to B is the same as the distance from B to A.

Now, to be a true distance, then you also have to satisfy the triangle inequality, that the distance from x to z is less than or equal to the sum of the distances through a third point. But we will find out that we don't actually need that for similarity measures. So we can have either a true distance metric for comparing gene expression data sets, or similarity measures as well.

So let's go back to the simple example. So we decided that the red and the blue genes were nearly identical, in terms of their distance metrics. But that's not always exactly what we care about. So in biological settings, frequently the absolute level of gene expression is on some arbitrary scale. Certainly with expression arrays, it was completely arbitrary. It had to do with fluorescence properties, and how well probes

hybridize to each other.

But even with mRNA, how do we really know that 1,000 copies is fundamentally different from 1,200 copies of an RNA in the cell? We don't. So we might be more interested in distance metrics that capture not just the similarity of these two. But the fact that these two are also quite similar, in terms of the trajectory of the plot to this one. So can we come up with measures that capture this one as well? A very common one for this is Pearson correlation.

So in Pearson correlation, we're gonna look at not just the expression of a gene across conditions. But we're gonna look at the z-score of that gene. So we'll take all of the data for all of the genes in a particular condition. And we'll compute the z-score by looking at the difference between the expression of a particular gene, in the average expression across the whole data set. And we're going to normalize it by the standard deviation. Yes?

AUDIENCE: [INAUDIBLE] square there?

PROFESSOR: Yes, you're right. There should be a square there. Thank you.

So then to compute the Pearson correlation, we're going between two genes, A and B, we're going to take the sum over all experiments, that the z-score for A and the z-score for B, the product of that, summed over all the experiments. And these values as we'll see in a second, are going to range from plus 1, which would be a perfect correlation, to minus 1, which would be a perfect anti-correlation. And then we're going to find the distance is 1 minus this value. So things that are perfectly correlated then, would have an r of zero. And things that are anti-correlated would have a large one.

So if we take a look at these two obviously by Euclidean distance, they'd be quite different from each other. But the z-scores have converted the expression values into z-scores over here, you can see that the z-scores obviously, this one is the most negative of all of the ones. And this as the lowest one in all of these. This one's the highest. And similarly for the red one, lowest to the highest. So the z-

scores track very well. And when I take the product of this, the signs of the z-score for A and B are always the same. So I summed the product of the z-scores, I get a large number. And then the normalization guarantees that it comes out to one.

And so the red and blue here will have a very high correlation coefficient. In this case, it's going to be an r correlation coefficient of 1. Whereas compared to this one, which is relatively flat, the correlation coefficient will be approximately zero. Any questions on that?

So what about, say the blue and the red? Well, their z-scores are going to have almost the opposite sign every single time. And so that's going to add up to a large negative value. So for these, they'll be highly anti-correlated. So A, the blue and the red, have a correlation coefficient of minus 1. OK. So we have these two different ways of computing distance measures. We can compute the Euclidean distance, which would make the red and blue the same, but treat the green one as being completely different. Or we have the correlation, which would group all of these together, as being similar. What you want to do is going to depend on your setting. If you look in your textbook, you'll see a lot of other definitions of distance as well.

Now what if you're missing a particular data point? This used to be a lot more of a problem with arrays than it is with [? RNAC. ?] With arrays, you'd often have dirt on the array, that it actually would literally cover up spots. But you have a bunch of choices. The most extreme would just be to ignore that row or column of your matrix across old data sets. That's usually not what we want to do. You could put in some arbitrary small value. But frequently we will do what's called imputing, where we'll try to identify the genes that have the most similar expression, and replace the value for the missing one with a value from the ones that we do know.

Distance metrics, pretty straightforward. Now we want to use these distance metrics to actually cluster the data. And what's the idea here? That if we look across enough data sets, we might find certain groups of genes that function similarly across all those data sets, that might be revealing as to their biological function.

So this is an example of an unsupervised learning problem. We don't know what the

classes are, before we go in. We don't even know how many there are. We want to learn from the data. This is a very large area of machine learning. We're just gonna scrape the surface. Some of you may be familiar with the fact that these kinds of machine learning algorithms are used widely outside of biology. They're used by Netflix to tell you what would movie to choose next. Or Amazon, to try to sell you new products. And all the advertisers who send pop-up ads on your computer.

But in our biological setting then, we have our gene expression data, collected possibly over very large numbers of conditions. And we want to find groups of genes that have some similarity. This is a figure from one of these very early papers, that sort of establish how people present these datas. So you'll almost always see the same kind of presentation. Typically you'll get a heat map, where genes are rows. And the different experiments here time, but it could be different perturbations, are the columns. And genes that go up in expression are red, and genes ago down in expression are green. And apologies to anyone who's colorblind. But that's just what the convention has become.

OK, so then why cluster? So if we cluster across the rows, then we'll get sets of genes that potentially behave-- that hopefully if we do this properly, behave similarly across different subsets of the experiments. And those might represent similar functions. And if we cluster the columns, then we get different experiments that show similar responses. So that might be in this case, different times that are similar. Hopefully those are ones that are close to each other. But if we have lots of different patients, as we'll see in a second, they might represent patients who have a similar version of a disease.

And in fact, the clustering of genes does work. So even in this very early paper, they were able to identify a bunch of subsets of genes that showed similar expression at different time points, and turned out to be enriched in different categories. These ones were enriched in cholesterol biosynthesis, whereas these were enriched in wound healing, and so on.

So how do you actually do clustering? This kind of clustering is called hierarchical.

That's pretty straightforward. There are two versions of hierarchical clustering. There's what's called agglomerative and divisive. In agglomerative, you start off with each data point in its own cluster. And then you search for the most similar data point to it, and you group those together. And you keep doing that iteratively, building up larger and larger clusters.

So we've discussed how to compare our individual genes. But you should be able to, right now, to find, if I gave you the vector of expression for a single gene, to find the other genes in the data set that's most similar, by either say, Euclidean or Pearson correlation, or what have you. But once you've grouped two genes together, how do you decide whether a third gene is similar to those two? So now we have to make some choices. And so there are number of different choices that are commonly made.

So let's say these are our data. We've got these two clusters, Y and Z. And each circle represents a data point in those clusters. So we've got four genes in each cluster. Now we want to decide on a distance measure to compare cluster Y to cluster Z. So what could we do? So what are some possibilities? What might you do?

AUDIENCE: We could take the average of all points.

PROFESSOR: You could take the average of all points, right. What else could you do? Only a limited number of possibilities.

AUDIENCE: Centroid?

PROFESSOR: Yeah, so centroid, you could take some sort of average, right. Any other possibilities?

AUDIENCE: You can pick a representative from each set [INAUDIBLE].

PROFESSOR: So you could pick a representative, right? How would you decide in advance what that would be though? So maybe you have a way, maybe not. And what other possibilities are there? Yeah?

AUDIENCE: Measure all the distances [INAUDIBLE] to all the nodes in the other.

PROFESSOR: Right. So you could do all to all. What else could you do? You can take the minimum of all those values. You can take the maximum of all those values. And we'll see that all those are things that people do. So this clustering, there are already rather uninformative terms for some of these kinds of decisions. So it's called single linkage, is you decide that the distance between two clusters is based on the minimum distance between any member of cluster Y and any member of cluster Z.

Complete linkage takes the maximum distance. And then the extremely unfortunately named Unweighted Pair Group Method using Centroids-- UPGMC, I won't try to say that very often-- takes the centroid, which was an early suggestion from the class. And then the UPGMA, Unweighted Pair Group Method with Arithmetic Mean, takes the average of all the distances, all suggestions that people have made.

So when would you use one versus the other? Well, a priori, you don't necessarily know. But it's good to know how they'll behave. So what do you imagine is going to happen if you use single linkage, versus complete linkage. Remember, single linkage is the minimum distance. And complete linkage is the maximum distance. So what's going to happen in this case, if I use the minimum distance. Which two groups will I combine?

AUDIENCE: The blue and the red.

PROFESSOR: The blue and the red, right? Whereas if I use the maximum distance, then I'll combine the green and the red. So it's important to recognize, then, that the single linkage has this property of chaining together clusters, based on points that are near each other. Whereas the complete linkage is resistant to grouping things together, if they have outliers. So they'll behave differently.

Now, if your data are compact, and you really do have tight clusters, it's not going to matter too much would you use. But in most biological settings, we're dealing with much noise, there's data. So you actually will get different results based on this. And

as far as I know, there's no really principal way to figure out if you have no prior knowledge, which to use.

Now all of these hierarchical clustering come with what's called a dendogram. And you'll see these at the top of all the clustering. And this represents the process by which the data were clustered. So the things that are most similar are most tightly connected in this dendogram. So these two data points, one and two, you have to go up very little in the y-axis, to get from one to two. Whereas if you want to go from one to 16, you have to traverse the entire dendogram. So the distance between two samples is how far vertically you have to go to connect between them.

Now the good things are that the dendogram is, you can then understand the clustering of the data. So I can cut this dendogram at any particular distance, and get clearly divisions among my data sets. So if I cut here at this distance level, then I have two groups. One small, one consisting of these data. And one large, one consisting of these. Whereas if I cut down here, I have more groups of my data. So it doesn't require me in advance to know how many groups I have. I can look at the dendogram and infer it.

The one risk is that you always get a dendogram that's hierarchical, regardless of if the data were hierarchical or not. So it's more a reflection of how you did your clustering than any fundamental structure of the data. So the fact that you get a hierarchical dendogram means really nothing about your data. It's simply a tool that you can use to try to divide it up into different groups. Any questions on the hierarchical clustering? Yes?

AUDIENCE: If each data point is its own cluster, then won't that be consistent across, like, single linkage, complete linkage-- like, why would you cluster? Does that question make sense? Like if you cut it down below, then haven't you minimized-- don't you successively minimize the variance, I guess, up to your clusters, by--

PROFESSOR: So if I cut it at the lowest level, everybody is their own cluster. That's true. Right. I'm interested in finding out whether there are genes that behave similarly across the data sets. Or--

AUDIENCE: My question is, how would you go about determining how many clusters your want?

PROFESSOR: Oh, OK. So we'll come to that in a second. So hierarchical clustering, you don't actually have any objective way of doing that. But we'll talk about other means right now, where it's a little bit clearer. But actually fundamentally, there aren't a lot of good ways of knowing a priori what the right number of clusters is. But we'll look at some measures in a second that help.

So hierarchical clustering, as your question implies, doesn't really tell you how many clusters there are. Another approach is to decide in advance how many clusters you expect. And then see whether you can get the data of the group into that number or not. And an example of that is something called k-means clustering. So the nice thing about it, is it does give you the sharp divisions. But again if you chose k incorrectly, we'll see in a second, you will get-- you'll never less still get K-clusters. So K refers the number of clusters that you tell the algorithm you expect to get.

So you specify that in advance. And then you try to find a set of clusters that minimizes the distance. So everybody's assigned to a particular cluster, and the center of that cluster. Is that clear? So that's what these equations represent. So the center of the cluster, the centroid, is just the average coordinates, over all the components of that cluster. And we're trying to find this set of clusters, C, that minimizes the sum of the square of the distances between each member of that cluster and the centroid. Any questions on how we're doing this? OK. All right.

So what's the actual algorithm? That's remarkably simple. I'm choosing that initial set of random positions. And then I have the simple loop, I repeat until convergence. For every point, I assign it to the nearest centroid.

So if my starting centroids would be circles, I look at every data point, and I ask, how close is it to any one of these? That's what the boundaries are, defined by these lines. So everything above this line belongs to the centroid. Everything over here belongs to this centroid. So I divide the data up by which centroid you are closest to. And I assign you to that centroid. That's step one.

And step two, I compute new centroids. And that's what these triangles represent. So after I did that partitioning, it turns out that most of the things that were assigned to the triangular cluster live over here. So the centroid moves from being here to here. And I iterate this process. That's the entire K-means clustering algorithm.

So here's an example where I generated data from three [? calcines. ?] I chose initial data points, which are the circles. I follow that protocol. Here's the first step. It computes new triangles. Second step, and then it converges. The distance stops changing.

Now this question's already come up. So what happens if you choose the wrong K? So I believe there are three clusters. And really that's not the case. So what's going to happen?

So in this data set, there really were. How many, there really were five clusters. Here, they're clustered correctly. What if I told the algorithm to do K-means clustering with a K of three? It would still find a way to come up with three clusters. So now it's grouped these two things, which are clearly generated from different [? calcines ?] scenes together. It's grouped these two, which were generated from different [? calcines ?] together, and so on.

All right. So K-means clustering will do what you tell it to do, regardless of whether that's the right answer or not. And if you tell it there are more clusters than you expect-- than really are there, then it'll start chopping up well-defined clusters into sub-clusters. So here it split this elongated one into two sub-clusters. It split this one arbitrarily into two. Just so it gets the final number that we asked for.

Then how do you know what to do? Well, as I said, you don't-- there's no guarantee to know. But one thing you can do is make this kind of plot, which shows for different values of K on the x-axis, the sum of the distances within the cluster. So the distance to the centroid within each cluster on the y-axis.

And as I increase the number of K's, when I'm correctly [? purchasing ?] my data, when there really are more subgroups than I've already defined, then I'll see big

drops. So I go from saying there are two to three in that case. I get a big drop in the distance between members of the cluster. Because I'm no longer including a data point over here. And in this cluster, with a data point in that cluster.

But once I go beyond the correct number, which was five, you see that the benefits really start to trail off. So there's an inflection point here. There's an elbow-- sometimes it's called an elbow plot. After I go past the right number, I get less and less benefit from each additional clustering. So this gives us an empirical way of choosing approximately a correct value for K. Any questions on K-means? Yes?

AUDIENCE: Does K-means recapitulate the clusters that you would get if you cut off your dendogram from hierarchical clustering at a certain level?

PROFESSOR: Not necessarily.

AUDIENCE: OK. But maybe. I don't know. It sort of seems to me as if you picked a level where you have a certain number of clusters, that that's similar, at least by centroid, by using the center?

PROFESSOR: Yeah, I think because of the way that you do it, you're not even guaranteed to have a level, where you have exactly the right-- other questions? Yes?

AUDIENCE: Could you just very quickly go over how you initialized where the starting points are, and the break ups?

PROFESSOR: All right, so the question is how do you initialize the starting points? In fact, you have to make some arbitrary decisions about how the initialize the starting points. So they're usually chose in a random. And you will get different results, depending on how you do that. So that's another-- so when you do it, it's non-deterministic in that sense. And you often want to initialize multiple times. And make sure you get similar results. Very good question. And in fact, that was not a set up. But what happens if you choose pathologically bad initial conditions?

So you have the potential to converge to the right answer. But you're not guaranteed to converge to the right answer. So here's an example where I had-- I

guess there really are three clusters in the data. I chose [INAUDIBLE] three, but I stuck all my initial coordinates down in the lower right-hand corner. And then when I do the clustering, if things go well, I get the right answer. But we're not guaranteed.

But one thing we are guaranteed, is we always get convergence. So the algorithm will converge. Because at each step, it's either reducing the objective function, or it's leaving it the same. So we're guaranteed convergence. But it may be as we've seen previously in other settings, we may end up with local minimum, rather than the global optimum. And the way to fix that then would be to initialize again, with new starting positions. Other questions?

What about a setting like this? Where we've got two well-defined clusters, and somebody who lives straight in the middle. So what's the algorithm going to do? Well, sometimes it'll put it in one side, of one cluster. And sometimes it'll end up in the other side. So an alternative to K-means clustering, which has to make one or the other arbitrary decision, is something that's called fuzzy K-means, which can put something actually literally, membership into both clusters. And it's very similar in structure to the K-means, with one important difference, which is a membership variable, that tells you for every data point, how much it belongs to the cluster one, cluster two, cluster three, and so on.

So in both algorithms, we start off by choosing initial points as a cluster means, and looping through each of them. Now previously, we would make a hard assignment of each data point x sub i to a single cluster. And here we're going to calculate the probability that each data point belongs to a cluster. And that's where you get the fuzziness, because you could have a non unit, or a nonzero probability, belonging to any of the clusters. And now we're going, K-means, we recalculated the mean value, by just looking at the average of everybody in that cluster.

Now in fuzzy K-means, we don't have everybody in the cluster. Because everybody belongs partially to the cluster. So we're going to take a weighted average. So here are the details of how you do that. In K-means, we are minimizing this function. We were trying to decide the class structure, the class memberships, that would

minimize the distance of every member of that cluster, to the defined centroid of that cluster. Here it looks almost the same. Except we now have this new variable, mu, which is the membership. It's the membership of point j, in cluster i. So I'm trying to minimize a very similar function. But now if mu is one-- if all my mus are one, then what do I get? K-means, right? But as soon as the mus are allowed to vary from one, they can be between zero and one, then points can contribute more or less. So that point there was stuck in the middle of the two clusters, if it had a mu of 0.5 for each, it would contribute half to each. And then both the centroids would move a little bit towards the middle.

So what's the result of K-means-- I'm sorry, fuzzy K-means clustering? We still get K clusters. But now every gene or every object that we're clustering has a partial membership. So here's an example of that, where they did K-means clustering, with these six different clusters. But now every profile, every gene, has a color associated with it, that represents this mu value. Whether it goes from zero to one, with these rainbow colors, to the things that are reddish, or pink-- those are the high confidence things that are very strongly, only in that cluster. Whereas the things that are more towards the yellow end of the spectrum are partially in this cluster and partially in other clusters. Questions? Any questions?

So K-means, we've defined in terms of Euclidean distance. And that has clear advantages, in terms of computing things very easily. But it has some disadvantages as well. So one of the disadvantages is because we're using the squared distance, then outliers have a very big effect. Because I'm squaring the difference between vectors. That may not be the worst thing. But they also restrict us to things for which we can compute a centroid. We have to have data that are-- four or more, you can actually compute the mean value of all members of the cluster.

Sometimes you want to cluster things that we only have qualitative data. Where instead of having a distance measure, we have similarity. This doesn't come up quite as often in-- well, it certainly doesn't come up in gene expression data or [? RNAC. ?] But you can imagine more qualitative data, where you ask people about

similarity between different things or behavioral features, where you know the similarity between two objects. But you have no way of calculating the average object.

One setting that you might [INAUDIBLE] have looked at-- if you're trying to cluster say, sequence motifs that you've computed with the EM algorithm. So what's the average sequence motif? That doesn't necessarily represent any true object, right? You might be better off-- you can calculate it. But it doesn't mean anything. You might be better off calculating using rather than the average motif, the most central of the motifs that you actually observed. So that would be called a medoid, or an exemplar. It's a member of your cluster that's closest to the middle, even if it it's not smack dab in the middle.

So instead of K-means, we can just think, well, K-medoids. So in K-means, we actually computed a centroid. And in medoids, we'll choose the existing data point that's most central. So what does that mean?

If these are my data, the true mean is somewhere over here. But this one is the medoid. It's an exemplar that's close to the central point. But if there actually isn't anything here, then there isn't. So we're going to use the thing that's closest. So if these were all sequence motifs, rather than using some sequence motif that doesn't exist as the center of your cluster, you would use a sequence motif that actually does exist, and it's close to the center.

So it's a simple variation on the K-means. Instead choosing K points in arbitrary space as our starting positions, we're going to choose K examples from the data as our starting medoids. And then we're going to place each point in the cluster that has the closest medoid, rather than median. And then when we do the update step, instead of choosing the average position to represent the cluster, we'll choose the medoid. The exemplar that's closest to the middle. Any questions on this? Yes?

AUDIENCE: So if you use the medoid, do you lose the guaranteed convergence? Because I can picture a situation where you're sort of oscillating because now you have a discrete stack.

PROFESSOR: That's a good question. That's probably right. Actually, I should think about that. I"m not sure. Yeah, that's probably right. Other questions? OK.

There are a lot of other techniques for clustering. Your textbook talks about self organizing maps, which were popular at one point quite a lot. And there's also a nice technique called affinity propagation, which is a little bit outside the scope of this course, but has proved quite useful for clustering.

OK. So why bother to do all this clustering? Our goal is to try to find some biological information, not just to find groups of genes. So what can you do with these things? Well, one thing that was identified early on, is if I could find sets of genes that behave similarly, maybe those could be used in a predictive way, to predict outcomes for patients, or some biological function.

So we're going to look at that first. So one of the early papers in this field did clustering of microarrays for patients who had B-cell lymphoma. The patients had different kinds of B-cell lymphomas. And so they took their data, they clustered it. Again, each row represents a gene. And each column represents a patient here.

And with this projector, it's a little bit hard to see. But when you look at the notes separately, you'll be able see that in the dendogram, there's a nice, sharp division between two large groups of patients. And it turns out that when you look at the pathologist's annotations for these patients, which was completely independent of the gene expression data, all of patients in the left hand group-- almost all the patients in the left hand group, had one kind of lymphoma. And all the patients in the right hand group had a different kind of lymphoma.

And this got people very excited. Because it suggested that the pure molecular features might be at least as good as pathological studies. So maybe you could completely automate the identification of different tumor types.

Now the next thing that got people even more excited, was the idea that maybe you could actually use these patterns not just to recapitulate what a pathologist would find, but go beyond it, and actually make predictions from the patients. So in these

plots-- I don't know if we've seen these before yet in the class. But on the x-axis is survival. In the y-axis are the fraction of patients in a particular group, who survived that long. So as the patient's die, obviously the curve is dropping down. Each one of these drops represents the death of a patient, or the loss of the patient to the study for other reasons.

And so in the middle, let's start with this one. This is what the clinicians would have decided. There are here, patients that they defined by clinical standards as being likely to do well, versus patients whom they defined by clinical standards, as likely to do poorly. And you could see there is a big difference in the plots for the low clinical risk patients at the top, and the high clinical risk patients at the bottom. On the left hand side, or what you get when you use purely gene expression data to cluster the patients into groups that you turn out to be high risk or low risk. And you can see that it's a little bit more statistically significant for the clinical risk. But it's pretty good over here, too.

Now the really impressive thing is, what if you take the patients that the clinicians define as low clinical risk? And then you look at their gene expression data. Could you separate out the patients in that allegedly low clinical risk who are actually at high risk? And maybe then they would be diverted to have more aggressive therapy than patients who really and truly are low risk patients. And what they will show with just barely statistical significance, is that even among the clinically defined low risk patients, there is-- based on these gene signatures-- the ability to distinguish patients who are going to do better, and patients who are going to do worse.

So this was over a decade ago. And it really set off a frenzy of people looking for gene signatures for all sorts of things, that might be highly predictive. Now the fact that something is correlated, doesn't of course prove any causality. So one of the questions is, if I find a gene signature that is predictive of an outcome in one of the studies, can I use it then to go backwards, and actually define a therapy? In the ideal setting, I would have these gene signatures. I'd discover that they are clinically associated with outcome. I could dig in and discover what makes the patients to do worse, worse. And go and treat that. So is that the case or not? So let me show you

some data from a breast cancer data set.

Here's a breast cancer data set. Again the same kind of plot, where we've got the survival statistic on the y-axis, the number of years on the x-axis. And based on a gene signature, this group has defined a group that does better, and a group that does worse, the p value is significant. And it has a ratio, the death rate versus control is approximately two. OK. So does this lead us to any mechanistic insight into breast cancer. Well, it turns out in this case, the gene signature was defined based on postprandial laughter. So after dinner humor.

Here's a gene set that defined something that has absolutely nothing to do with breast cancer, and it's predicting the outcome of breast cancer patients. Which leads to somewhat more of a joke that the testing whether laughter really is the best medicine. OK. So they went on-- they tried other genes sets. Here's the data set-- gene set that's not even defined in humans. It's the homologs of genes that are associated with social defeat in mice. And once again, you get a statistically significant p-value, and good hazard ratios.

So what's going on? Well, these are not from a study that's actually trying to predict an outcome in breast cancer. It's a study that shows that most gene expression-- most randomly selected sets of genes in the genome will give an outcome that's correlated-- a result that's correlated with a patient outcome in breast cancer. Yes?

AUDIENCE: I'm a little confused. In the previous graph, could you just explain what is the black and what is the red? Is that individuals or groups?

PROFESSOR: So the black are people that have the genes set signature, who have high levels of the genes that are defined in this gene set. And the red are ones have low, or the other way around. But it's defining all patients into two groups, based on whether they have a particular level of expression in this gene set, and then following those patients over time. Do they do better or worse? And similarly for all these plots.

And he had another one which is a little less amusing, location of skin fibroblasts. The real critical point is this. Here, they compared the probability based on

expectation an that all genes are independent of each other, the probability that that gene signatures correlated with outcome, for genes there were chosen at random or genes that were chosen from a database of gene signatures, that people have identified as being associated with pathways. And you get a very, very large fraction. So this is the p-value. So negative log of p-value, so negative values are more significant. A huge fraction of all genes sets that you pull at random from the genome, or that you pull from a compendium of known pathways, are going to be associated with outcome, in this breast cancer data set.

So it's not just well annotated cancer pathways, that are associated. Its gene sets associated as we've seen, with laughter or social defeat in mice, and so on-- all sorts of crazy things, that have no mechanistic link to breast cancer. Let's take a second for that to sink in. I pull genes at random from the genome. I define patients based on whether they have high levels of expression of a random set of genes, or low levels of expression of that random set of genes. And I'm extremely likely to be able to predict the outcome in breast cancer. So that should be rather disturbing, right?

So it turns out-- before we get to the answer then-- so this is not unique to breast cancer. They went through a whole bunch of data sets in the literature. Each row is a different previously published study, where someone had claimed to identify a signature for a particular kind of disease or outcome. And they took their random gene sets and asked how well the random genes sets did in predicting the outcome in these patients? And so these yellow plots represent the probability distribution for the random gene sets-- again on this projector, it's hard to see-- but there's a highlight in the left hand side at where the 5%, the best 5% of the random gene sets are. This blue line is the near measure of statistical significance. It turns out that a few of these studies didn't even reach a normal level of statistical significance, let alone comparing to random gene sets. But for most of these, you don't do better than a good fraction of the randomly selected gene sets.

So how could this be? So it turns out there is an answer to why this happens. And it's really quite fascinating. So here, we're using the hazard ratio, which is the death

rate for the patients who have the signature, over the control group. So high hazard ratio means it's a very, very dissociative outcome. And they've plotted that against the correlation of the genes in the gene signature, with the expression of a gene called PCNA, Proliferating Cell Nuclear Antigen

And it turns out a very, very large fraction of the genome is coexpressed. So genes are not expressed like random, completely independent random variables. There are lots of genes that show very similar expression levels, across all the data sets. Now PCNA is a gene that's been known by pathologists for a long time, as having higher levels than most digressive tumors. So a very, very large fraction of the genome is coexpressed with PCNA. Then high levels of randomly selected genes are going to be a very good predictor of tumor outcome. Because high levels of randomly expressed genes also means a very high probability of having a high level PCNA, which is a tumor marker.

So we have to proceed with a lot of caution. We can find things that are highly correlated with outcome, that could have good value in terms of prognostic indicators. But there are going to be a lot of possibilities for sets of genes that have that property, they're good predictors of outcome. And many of them will have absolutely nothing to causally, with the process of the disease. So at the very least, it means don't start a drug company over every set of genes, if you identify this as associated with outcome. But the worst case scenario, it also means that those predictions will break down under settings that we haven't yet examined. And so that's the real fear, that you have a gene set signature that you think has a highly predictive outcome. It's only because you looked at a particular set of patients. But you look at a different set of patients, and that correlation will break down.

So this is an area of research that's still quite in flux, in terms of how much utility there will be in identifying genes set signatures, in this completely objective way. And what we'll see in the course of this lecture and the next one, is it's probably going to be much more useful to incorporate other kinds of information that will constrain us to be more mechanistic. Any questions?

All right. So now we're going to really get into the meat of the identification of gene modules. And we're going to try to see how much we can learn about regulatory structure from the gene expression data. So we're going to move up from just the pure expression data-- say these genes at the bottom, to try to figure out what set of transcription factors we're driving, and maybe what signaling pathways lived upstream in those transcription factors, and turn them on. And the fundamental difference then between clustering-- which is what we've been looking in until now, and these modules, as people like to call them-- is that you can have a whole bunch of genes, and we've just seen that, that are correlated with each other, without being causally linked to each other. So we like to figure out which ones are actually functionally related, and not just statistically related.

And the paper that's going to serve as our organizing principle in the rest of this lecture, maybe bleeding into the next lecture, is this paper, recently published that's called The DREAM5 Challenge. And this, like some of these other challenges that we've seen before, is the case where the organizers have data sets, where are they know the answer to what the regulatory structure is. They send out the data. People try to make the best predictions they can. And then they unseal the data, to let people know how well they did. And so you can get a relatively objective view of how well different kinds of approaches work.

So this is the overall structure of this challenge. They had four different kinds of data. Three are real data sets from different organisms, E. coli, yeast, and Staphylococcus aureus. And then the fourth one, the one at the top here, is completely synthetic data that they generated it. And you get a sense of the scale of the data sets. So how many genes are involved, how many potential regulators. In some cases, they've given you specific information on knockouts, antibiotics, toxins, that are perturbing. And again here, the number of conditions that are being looked at, the number of arrays.

So then they provide this data in a way that's very hard for the groups that are analyzing to trace it back to particular genes. Because you don't want people to use external data necessarily, to make their predictions. So every makes their

predictions. They also, as part of this challenge, they actually they made their own metapredictions, based on the individual predictions by different groups. And we'll take a look at that in a second. And then they score how well they did.

Now we'll get into the details of the scoring a little bit later. But what they found at the highest levels, that different kinds of methods behaved similarly. So the main groups that they found were these regression-based techniques. We'll talk about those in a second. Bayesian networks, which we've already discussed in a different context. A hodgepodge of different kinds of things. And then mutual information and correlation. So we're going to look in each of these main categories of prediction methods.

So we're going to start with the Bayesian networks, which we just finished talking about in a completely different context. Here, instead of trying to predict whether interaction is true, based on the experimental data, we're going to try to predict whether a particular protein is involved in regulating a set of genes, based on the expression data. So in this context-- let's say I have cancer data sets, and I wanted to decide whether p53 is activated in those tumors, So this is a known pathway for p53. So if I told you the pathway, how might you figure out if p53 is active from gene expression data?

I tell you this pathway, give you this expression data-- what's kind of a simple thing that you could do right away, to decide whether you think p53 is active or not? p53 is a transcriptional activator, but it should be turning on the genes of its targets when its on. So what's an obvious thing to do?

AUDIENCE: Check the expression levels from the targets.

PROFESSOR: Thank you. Right, so we could check the expression levels. The targets compute some simple statistics, right? OK. Well, that could work. But of course there could be other transcriptional regulators that regulate a similar set of genes. So that's not a guarantee that p53 is on. It might be some other transcriptional regulator.

We could look for the pathways that activate p53. We could ask whether those

genes are on. So we've got in this pathway, a bunch of kinases, an ATM, CHK1, and so on, that activate p53. Now if we had proteomic data, we could actually look whether those proteins are phosphorylated.

But we have much, much less proteomic data. And most of these settings only have gene expression data. But you look at, is that gene expressed? Has the expression of one of these activating proteins gone up? And you can try to make an inference then. From whether there's more of these activating proteins, then maybe p53 is active. And therefore it's turning on it's targets. That's one step removed. So just the fact that there's a lot of ATM mRNA around doesn't mean that there's a lot of the ATM protein, which certainly doesn't mean that the ATM is phosphorylated and turning on its target. So again, we don't have a guarantee there.

We could look more specifically whether the genes are differentially expressed. So the fact that they're on may not be as informative as if they were uniquely on in this tumor, and not on in control cells from the same patient. So that can be informative. But again changes in gene expression are not uniquely related to changes in protein level. So we're going to have to behave with a bit of caution.

So the first step we're going to take in this direction, is try to build a Bayesian network. That's going to give us a way to reason probabilistically over all of these kinds of data, which by themselves are not great guarantees that we're getting the right answer. Just like in the protein prediction interaction problem, where individually coexpression wasn't all that great, essentiality wasn't all that great. But taken together, they could be quite helpful. So we want to compute the probability that the p53 pathway is active, given the data. And the only data we're going to have in the setting is gene expression data. So we're going to assume that for the targets of a transcription factor to be active, the transcription factor itself has to be expressed at a higher level. That's a restriction of analyzing these kinds of data that's very commonly used.

So we're going to try to compute the probability that p53 is activated, given the data. So how would I compute the probability, that given that some transcription factors

on, that I see expression from target genes? How would I do this? I would just go into the data, and just count in the same way that we did in our previous setting. We could just look over all the experiments and tabulate whether one of the targets is up in expression, how often is the transcription factor that's potentially activating it up? And how often are all the possible combinations the case? And then we can use Bayesian statistics to try to compute the probability that a transcription factor is up, activated, given that I've seen the gene expression data. Is that clear? Good.

So we want to try to not include just the down stream factors. Because that leads possibly, maybe there are multiple transcription factors that are equally likely to be driving expressions instead of genes. We want to include the upstream regulators as well.

And so here, we're going to take advantage of one of the properties of Bayesian nets at where we looked at, explaining a way. And you'll remember this example, where we decided that if see that the grass is wet, and I know that it's raining, then I can consider less likely that the sprinklers were on. Even though there's no causal relationship between them. So if I see that a set of targets of transcription factor A are on, and I have evidence that the pathway upstream of A is on, that reduces my inferred probability that the transcription factor B is responsible. So that's of the nice things about Bayesian networks that gives us a way of reasoning automatically, over all the data, and not just the down stream targets.

And the Bayesian networks can have multiple layers. So we can have one transcription factor turning another one, turns on other one, turns on another one. Again, we can have as many layers as necessary. But one thing we can't have are cycles. So we can't have a transcription factor that's at the bottom of this, going back and activating things that are at the top. And that's a fundamental limitation of Bayesian networks. We've already talked about the fact that in Bayesian networks, with these two problems that we to have to solve, we have to be able to define the structure. If we don't know any a priori. Here, we don't know what a priori. So we're going to have to learn the structure of the network. And then with the structure of the network, we're going to have to learn all the probabilities. So the conditional

probability tables that relate to each variable to every other one.

And then just two more small points about it. So if I just give you expression data, without any interventions-- just the observations, then I can't decide what is a cause and what is an effect. So here this was done in the context of proteomics, but the same is true for gene expression data.

If I have two variables, x and y, that are highly correlated, it could be that x activates y. It could be that y activates x. But if I perturb the system, and I block the activity of one of these two genes or proteins, then I can start to tell the difference. In this case, if you inhibit x, you don't see any activation of y. That's the yellow, all down here. But if you inhibit y, you see the full range of activity of x.

So that implies that x is the activator of y. And so in these settings, if you want to learn a Bayesian network from data, you need more than just a compendium of gene expression data. If you want to get the directions correct, you need perturbations where someone has actually inhibited particular genes or proteins.

Now, in a lot of these Bayesian networks, we're not going to try to include every possible gene and every possible protein. Either because we don't have measurements of it, or because we need a compact network. So there will often be cases where the true regulator in some causal chain, is missing from our data.

So imagine this is the true causal chain-- x activates y, which then activates z and w. But either because we don't have the data on y, or because we left it out to make our models more compact, it's not in the model. We can still pick up the relationships between x and z, and x and w. But the data will be much noisier. Because we're missing that information. In the conditional probability tables, relating x to y, and then y because it's too targets.

So Bayesian networks, we've already seen quite a lot. We now have some idea of how to transfer them from one domain to the domain of gene expression data. The next approach we want to look at is a regression-based approach.

So the regression-based approaches are founded on a simple idea, which is that

the expression gene is going to be some function of the expression levels of the regulator. We're going to actually try to come up with a formula that relates the activity levels of the transcription factors, and the activity level of the target. In this cartoon, I've got a gene that's on under one condition, that's off under some other conditions. What transforms it from being off to on, is the introduction of more of these transcription factors, that are binding to the promoter.

So in general, I have some predicted level of expression for the gene. It's called the predicted level y. And it's some function, unspecified at this point, f of g, of all the expression levels of the transcription factors that regulate that gene. So just again, nomenclature is straight, x sub g is going to be the expression of gene x-- I'm sorry, expression of gene g. This capital X, sub t of g is the set of transcription factors, that I believe are regulating that gene. And then f is an arbitrary function. We're going to have a noise term as well. Because this is the observed gene expression, not some sort of platonic view of the true gene expression.

Now frequently, we'll have a specific function. So the simplest one you can imagine, which is a linear function. So the expression of any particular gene is going to be a linear function, a sum, of the expression of all of it's regulators, where each one has associated with it a coefficient beta. And that beta coefficient tells us how much particular a regulator influences that gene.

So say, p53 might have a very large value. Some other transcriptional regulator might have a small value, representing their relative influence. Now, I don't know the beta values in advance. So that's one of the things that I need to learn. So I want to be able to find a setting that tells me what the beta values are for every possible transcription factor. If the algorithm sets the beta value to zero, what does that tell me? If a beta value is zero here, what does that tell me about that transcriptional regulator? No influence, right. And the higher the value, then the greater the influence.

OK. So how do we discover these? So the tip of the approach then is to come up with some objective function that we're going to try to optimize. And an obvious

objective function is the difference between the observed expression value for each gene, and the expected one, based on that linear function. And we're going to choose a set of data parameters that minimize the difference between the observed and the expected, minimize the sum of the squares. So the residual sum of the squares error, between the predicted and the observed.

So this is a relatively standard regression problem, just in different setting. Now one of the problems with a standard regression problem, is that we'll typically get a lot of very small values of beta. So we won't get all zeros or all ones, meaning the algorithm is 100% certain that these are the drivers and these are not. We'll get a lot of small values for many, many transcription factors. And OK, that could represent the reality. But the bad thing is that those data values are going to be unstable. So small changes in the training data will give you big changes, in which transcription factors have which values. So that not a desirable setting. There's a whole field built up around trying to come up with better solutions. I've given you some references here. One of them is to a paper that did well in the DREAM challenge. The other one is to a very good textbook, Elements of Statistical Learning. And there are various techniques that allow you to try to limit the number of betas that are non-zero. And by doing that, you get more robust predictions. At a cost, right, because there could be a lot of transcription factors that really do have small influences. But we'll trade that off, by getting more accurate predictions from the ones that have the big influences. Are there any questions on regression?

So the last of the methods that we're examining-- this is a mutual information. We've already seen mutual information in the course. So information content is related to the probability of observing some variable in an alphabet. So in most languages, the probability of observing letters is quite variable. So Es are very common in the English language. Other letters are less common. As anyone who plays Hangman or watches Wheel of fortune knows. And we defined the entropy as the sum over all possible outcomes. The probability of observing some variable, and the information on to that variable, we can define the discrete case, or in the continuous case. And the critical thing is to have mutual information between two variables. So that's the difference between the entropy of those variables independently, and then the joint

entropy.

So things with a high mutual information, means that one variable gives the significant knowledge of what the other variable is doing. It reduces my uncertainty. That's the critical idea. OK. So we looked at correlation before. There could be settings where you have very low correlation between two variables, but have high mutual information. So consider these two genes, protein A and protein b, and the blue dots are the relationship between them. You can see that there's a lot of information content in these two variables. Knowing the value of A gives me a high confidence in the value of B. But there's no linear relationship that describes these. So if I use mutual information, I can capture situations like this, that I can't capture with correlation. And these kinds of situations actually occur.

So for example in a feed-forward loop-- say we've got a regulator A, and it directly activates B. It also directly activates C. But C inhibits B. So you've got the path on the left-hand sides that are pressing the accelerator. And the path on the right hand side pressing the stop pedal. That's called an incoherent feed-forward loop. And you can get under different settings, different kinds of results, where this is one of those examples.

You can get much more complicated behavior. [INAUDIBLE] are papers that have really mapped out these behaviors across many parameters settings. You can get switches in the behavior. But in a lot of these settings, you will have high mutual information between two variables, even if you don't have any correlation, linear correlation between them.

A well-publicized algorithm that uses mutual information to infer gene regulatory networks is called ARACNe. They go through and they compute the mutual information between all pairs of genes in their data set. And now one question you have with mutual information is, what defines a significant level of mutual information?

So an obvious way to do this, to try to figure out what's significant, is to do randomizations. And so that's what they did. They shuffled the expression data, to

compute mutual information among pairs of genes, where there isn't actually a need-- there shouldn't be any relationships. Because the data had been shuffled. And then you can decide whether the observed mutual information is significantly greater than when you get from the randomized data.

Now, the other thing that happens with mutual information is that indirect effects still apply to degrees of mutual information. So let's consider the set of genes that are shown on this. So you've got G2, which is actually a regulator of G1 and G3. So G2 is going to have high mutual information with G1, and with G3.

Now, what's it going to be about G1 and G3? They're going to behave very similarly, as well. So it'll be a high degree of mutual information between G1 and G3. So if I just rely on mutual information, I can't tell what's a regulator and what's a fellow at the same level of regulation. They're both being affected by something above them. I can't tell the difference between those two.

So they use what's called the data processing inequality, where they say, well, these regulatory interactions should have higher mutual information, than this, which is just between two common targets in the same parent. And so they drop from their network, those things which are the lower of the three in a triangle.

So that was the original ARACNe algorithm, and then they modified it a little bit, to try to be more specific in terms of the regulators that were being picked up. And so they called this approach MINDy. And the core idea here, is that in addition to the transcription factors, you might have another protein that turns a transcription factor on or off. So if I look over different concentrations of the transcription factor, different levels of expression between transcription factors, I might find that there are some cases where this other protein turns it on, and other cases where it turns it off.

So here, consider these two data sets. Looking at different concentrations of particular transcription factor and different expression levels, and in one case-- the blue ones, the modulator isn't present at all, or present at it's lowest possible level. And in the red case, it's present as a high level. And you can see that when the

modulator is present only in low levels, there's no relationship between a target and it's transcription factor. Or when the modulator is present at a high level, then there's this linear response of the target to it's transcription factor. So this modulator seems to be a necessary component. So they went through and defined a whole bunch of settings like this. And then systematically search the data for these modulators.

So they started off with the expression data set, genes in rows, experiments in columns. They do a set of filtering to remove things that are going to be problematic for the analysis. They look, for example, for settings where you have-- they had to start with a list of modulators and transcription factors, and they moved the ones where there isn't enough variation, and so on. And then they examine, for every modulator and transcription factor pair, cases where the modulator is present at its highest level, and where it's present at it's lowest level. So when the modulator is present at a high level-- let's say, when the modulator is present at a high level, there's a high mutual information between the transcription factor and the target. When the modulator is absent, there's no mutual information. That's a setting we looked at before. That would suggest that the modulator is an activator. It's a positive modulator.

You can have the opposite situation, where when the modulator is present at low levels, there's mutual information between a transcription factor and it's target. When the modulator is present at a high level, you don't see anything. That would suggest that the modulator is a negative regulator. And then there are scenarios where there's either uniformly high information content between transcription factor target, or uniformly low. So the modulator doesn't seem to be doing anything.

So we break it down into these categories. And you can look at all the different categories, in their supplemental tables. One thing that's kind of interesting is they assume that regardless of how high the transcription factor goes, you'll always see an increase in the expression of the target. So there is no saturation, which is an unnatural assumption in these data sets. OK. So I think I'll close with this example, from their experiment. And then in the next lecture, we'll look at how these different

methods fare against each other in the DREAM challenge.

So they specifically wanted to find regulators of MYC. So here's data for a particular regulator, SDK 38. Here's the set of expression of tumors where SDK 38 expression is lowest. And a set of tumors where SDK 38 expression is highest. And they're sorted by the expression level of MYC. So on the left hand side, you'll see there's no particular relationship between the expression level of MYC and the targets. In the right hand side, there is a relationship between the expression level of MYC and targets. So having, apparently-- at least at this level of mutual information, having higher levels of SDK 38, cause a relationship to occur. That would be example of an activator.

OK. So this technique has a lot of advantages, and allows you to search rapidly over very large data sets, to find potential target transcription factor relationships, and also potential modulators. It has some limitations. Where the key limitations is that the signal has to be present in the expression data set.

So in the case of a protein like p53, where we know it's activated by all sort of other processes, phosphorylation or NF-kappaB, where it's regulated by phosphorylation, you might not get any signal. So there has to be a case where the transcription factor itself, is changing expression. It also won't work if the modulator is always highly correlated with its target, for some other biological reason. So the modulator has to be on, for other reasons, when the target is, then you'll never be able to divide the data in this way.

One of the other things I think that is problematic with these networks is that you get such large networks, and they're very hard to interpret. So in this case, this is the nearest neighbors of just one node in ARACNe. This is the mutual information network of microRNA modulators that has a quarter of a million interactions. And in these data sets, often you end up selecting a very, very large fraction of all the potential modulators. So of all the candidate transcription factors in modulators, it comes up with an answer that's roughly 10% to 20% of them are regulating any particular gene, which seems awfully high.

OK. So any questions on the methods we've seen so far? OK. So when we come back on Thursday, we'll take a look at head to head of how these different methods perform on both the synthetic and the real data sets.