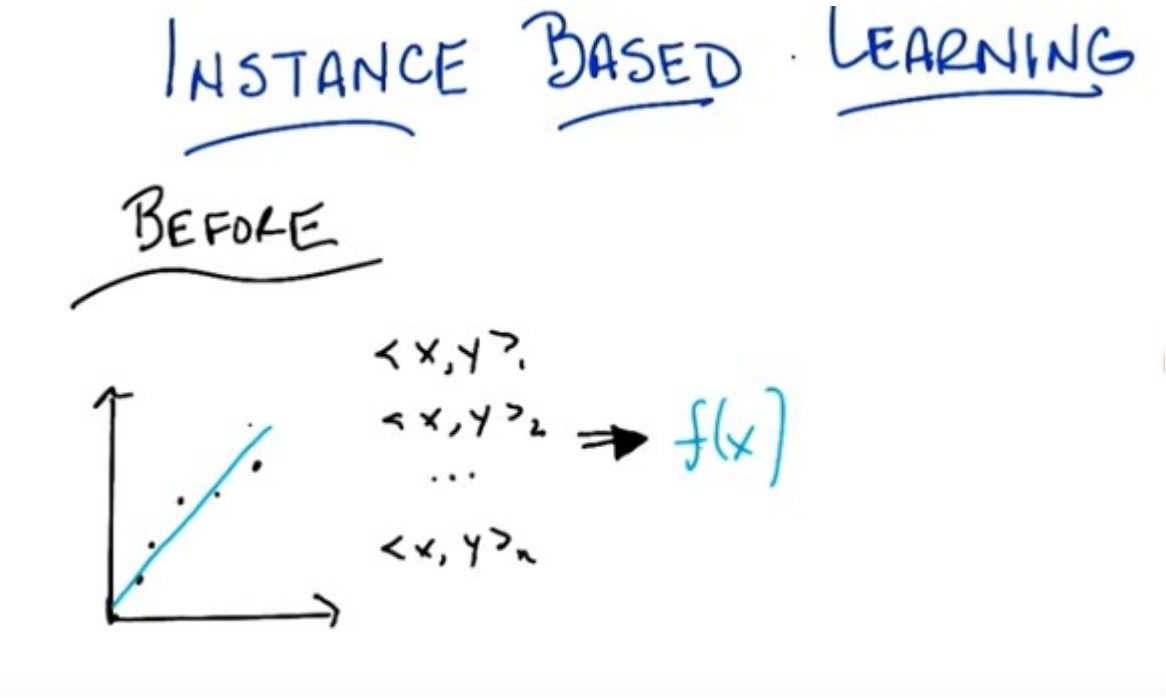
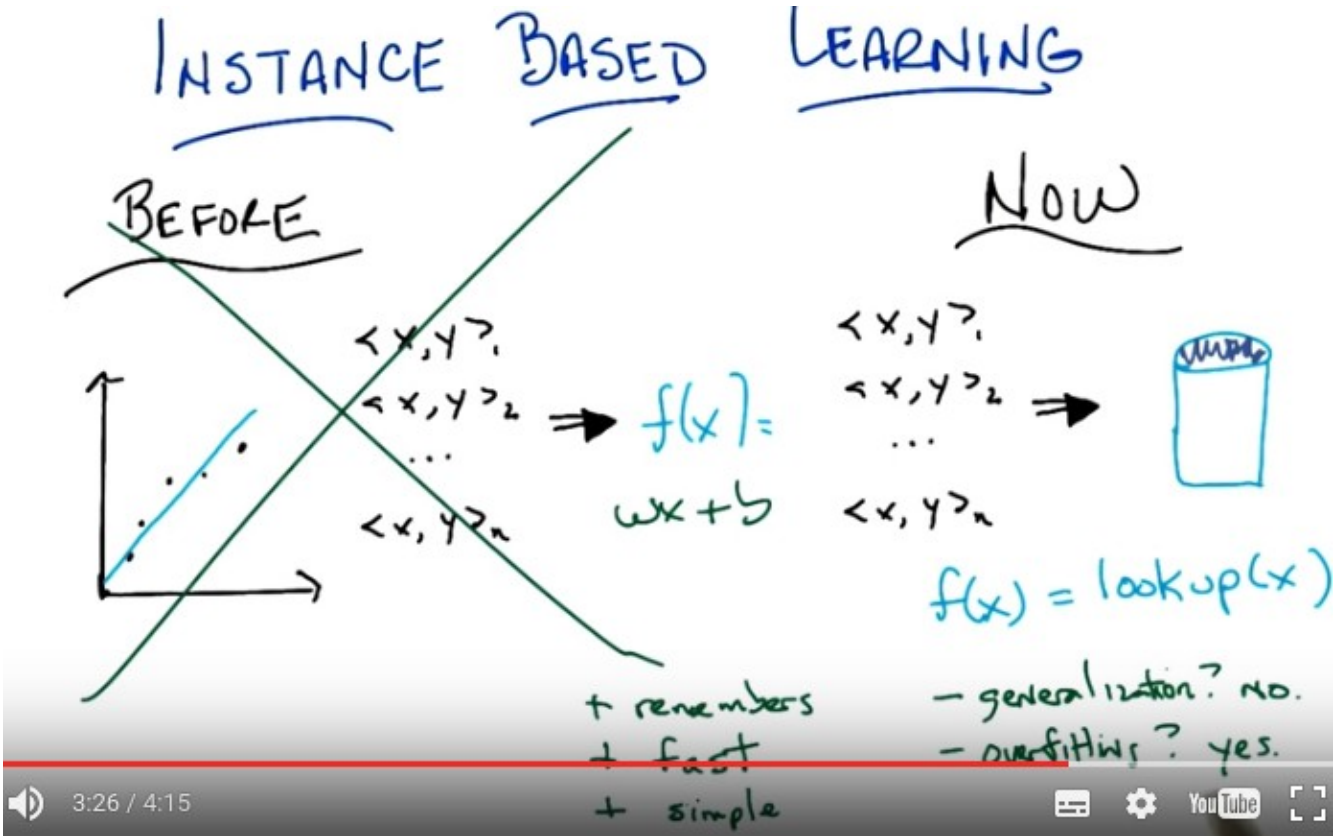
kNN 為何要叫 instance based learning? 可見 ml4t 的 course note 的 03-02 Regression 的第 3 段.



1. Hi Michael! >> Hey Charles! How's it going? >> It's going pretty well. How's it going with you? >> It is a beautiful fall day here in Providence Rhode Island. >> Oh that's right it's fall, when you are. >> [LAUGH] Yeah, I think, that's right. >> So, what we're going to do today, Michael. If you will indulge me. Is ,we're going to talk about a different class of ,uh, learning algorithms and approaches than we've been talking about before. >> So now the other ones were low class, this is going to be high class? >> Exactly. And we call them instance based learning. Which sounds very hoity toity and high voluting. Don't you agree? >> Yeah, sure, why not? >> [LAUGH] >> It sounds like it maybe has good posture. >> It does, in fact, have good posture. >> Well let's, let's learn about it. I'm, I'm, I'm intrigued. >> Yeah, so I think that ,uh, what we're going to end up talking about to day is kind of interesting, I hope. But it's sort of different, and what I'm hoping is through this discussion Is that, we will be able to reveal some of the unspoken assumptions that we've been making so far, okay? >> Unspoken assumptions, it sounds, yeah, okay, that sounds like we should get to the bottom of that. >> Yes, so let's do that. So, just to remind you of what we have been doing in the ,um, past, this is what was going on with all of our little supervised learning tasks, right. We were given a bunch of training data ,labeled here as you know, x, y One, xy two, xy three, dot, dot, dot, xy zen. And ,uh, we would then learn some function. So, for example, if we have a bunch of points in a plane, we might learn a line to represent them, which is what this little blue line is. And what was happening here is we take all this data. We come up with some function that represents the data. And then we would throw the data away effectively, right? >> Okay. Yeah, so like, black is the input here and then the two blue things are what get derived by the learning algorithm. >> Right. And then in the future when we get some data point, let's call it x, we would pass it through this function whatever it is. In this case, probably line. And that would, be how we would determine answers going forward. >> Yeah. That's, that's what we've been talking about. >> Right. And in particular without reference to the original data. So. I want to propose an alternative and the alternative is basically going to not do this. >> [LAUGH] >> So let me give you a, let me, let me tell u exactly what I mean by that.



2. Okay, so here what I mean concretely by not doing this thing over here any more. So here what I'm proposing we do now. We take all the data, all of the training data we had, the xy1, xy2, dot, dot, dot, xyn and we put it in a database >> Ah-ha. >> And then next time we get some new x to look at, we just look it up in the database. And we're done. >> None of this fancy shmancy learning, none of this producing an appropriate function like you know, wx+b, or what ever the equation of a line is. None of that fancy stuff anymore. We just stick in to the data base. People written data base programs before. We'll look it up when we're done. We're done.Period >> I feel like you've changed the paradigm. >> Yes, I am a paradigm changer. So what do you think? >> It's like, it's like disruptive. It's going to throw off the markets. >> Yeah. It's going to change everything. So, what do you think? >> well, I mean, so there's a bunch of really cool things about this idea. Which is why I'm excited. So one is it, you know, it doesn't forget, right? So it actually is very reliable. It's very ,um ,dependable, right, so if you put in an x, y pair you ask for the x you're going to get that y back instead of some kind, you know, crack potty, smooth version of it. >> Right, so we don't, yeah that's a good point. So we look at this little blue line over here, you'll notice that say, for this little x over here, we're not going to get back what we put in, so, it remembers, it's like an elephant. Good. >> So that's kind of cool. Another thing is that there's none of this wasted time, you know, doing learning [LAUGH]. It just takes the data and it, and it's, very rapidly just puts it in the database. So [CROSSTALK] it's fast. >> It's fast. It's like. I like it when you say nice things about my algorithms. Okay. So it's fast. Anything else, you can think of? >> Sh, yeah I cant think of one more thing at least. >> Mm-hm. >> Which is ,that like you, it's simple. >> [LAUGH] That's true. I am simple. I am simple and straightforward. I just need a few things to make me happy. >> Bacon. >> Bacon, and >> And >> Chocolate. >> Oh nice. >> Have you ever had chocolate covered bacon? >> That seems wrong. >> You know, you would think so, but it turns out it's delicious. It's like if you take fat, and sugar, and you put it together somehow you like it [LAUGH]

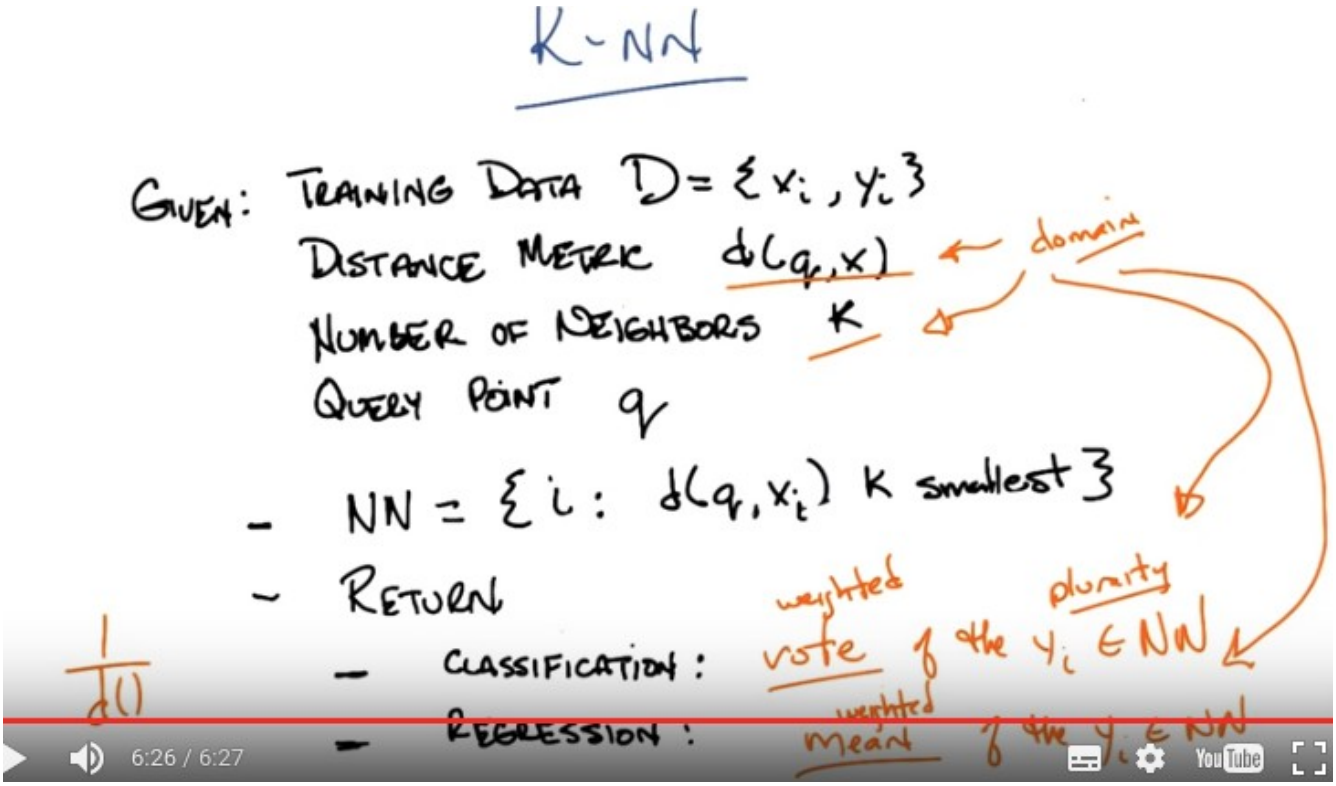
I'm not making this up you can buy chocolate covered bacon, you're unsurprised to here in America. Okay, so we've got three good things in remember stuff. So, you know, none of this little noisy throwing away information. It's very fast, you just stick it in a database. Using your favorite data base. And looking up is going to be equally as fast. And it's very simple. There's really no interesting learning going on here. So it's the perfect algorithm when we're done. >> Okay. I mean, I, it feels like there's more that we need to say though. >> Like what? >> In particular the way that you wrote this, F of X equals look up of X. If I give you one of the other points in between, then it will return no such point found. Which means it's really quite conservative. It's not willing to go out on a limb and say ,well I haven't seen this before but. It ought to be around here, instead it just says, I don't know. >> Mm. So the down side of remembering is, no generalization. >> [LAUGH] And I guess a similar sort of issue is that when you, when you call it memorization, it makes, it reminds me of the issues that we saw with regard to overfitting So, it bottles the noise exactly, it bottles exactly what it was given. So it's going to be very sensitive to noise. So it's kind of a yes and no. >> So that's a little scary and, and it can over fit in a couple of ways, I think it can over fit ,um, by believing the data too much that is literally believing all of it and what do you do if you have couple ,uh, speaking in noise, what if you have you know a couple of examples that are all the same. I have got an x, shows up multiple times but each with a different y. >> Oh,the same x,ah, yeah, and so the look up would return two different things and this algorithm or whatever that you have described so far, wouldn't commit to either of them and it would just say, hey, here is both. >> Yeah, that seems problematic. >> Okay, alright. But I feel like, you know, you are going to to tell me, how to fix those things. So I wasn't too worried. >> Yeah, well, there is gotta be a nice way of fixing it. I think There's sort of a basic problem here, which is that we're taking this remembering and then looking up a little too literally, right? So I stick in the data, and I can get back exactly the data that I got, but I can't get back anything that I don't have, and that seems like something that we might be able to overcome if we're just a little bit clever. >> Mm. >> So let's see if we can be a little bit clever.

3. Okay Michael, so let's see if we can, work together to deal with this minor triffle of a problem, that

,ah, you've observed with my cooling algorithm, okay. So, here's some data, it's a graph and you see here's a y axis and here's an x axis, and each of these points, represents a house, on this map, which I'm, I'm cleverly, using in the background. [CROSSTALK] And, you'll notice, that each of the dots is colored. I'm going to say that red represents, really expensive houses, blue represents, moderately expensive houses, and green represents, really inexpensive houses. Okay? >> Okay, where is this? >> Where is this? Oh, this is Georgia Tech, as you can tell because, it says Georgia Tech. >> Oh, I see it now. >> Okay. So, here's what I want you to do. using machine learning. I want you to look at all of this data, and then I want you to tell me, for these little black dots, whether they are really expensive, moderately expensive or, or inexpensive. But ,I want you to do it, using something like the technique that we talked about before. >> Okay? >> So, let's look at this little dot, over here. Which, by the way, I want to point out. this little black dot here by the US Post Office, underneath the rightmost ,e ,over here, is not a point in our data base. But I think by staring at this, you might be able to come up with a reasonable guess, about whether it is moderately expensive, expensive, or inexpensive. >> Okay, yeah. I think, this is a helpful, example, because, now I see that it does kind of make sense, especially, in this context, to think of the geometric location, as actually being a very useful attribute for deciding how to label the new points. So, that black point that you've pointed out, is in the part of the neighborhood, that has a green dot in it. Like, the nearest dot to it, seems like a pretty good guess as to what, what the value of that house might be, so I'm going to guess green. >> Yes, and I think ,you would be right. And I like the word that you used there. You talked about, its nearest neighbor, so I like that. I'm going to write that down. Neighbor, okay. So, I'm going to look at my nearest neighbor. Well let's see if this works, for another point. Let's look at another point, that's near an, e, let's see, the first e over here. This little black point, over here. What do you think? If I, if I looked at my nearest neighbor, what would I, what would I guess? >> Yeah, this one seems really clear. It's, it's surrounded by red. It's in the red part of town. >> So, you're guessing, the output is then, purple? [LAUGH] >> No, I'm going with red. >> Yes, and I think that that makes perfect sense. So, this is pretty cool. If I have a point that's not in my database, but I still, by looking at my nearest neighbour, can sort of figure out, ah, what the actual value should be. So, there we have solved, the problem. >> Yes, seems like a pretty good role. >> Yeah, just look at your nearest neighbour and you are done. There, so, boom. There is nothing else for you to do. >> Yeah, except that you didn't do all of the houses yet. >> Okay, well, what did I miss? >> The one in the middle and [CROSSTALK] I'm wondering, if maybe you did that on purpose, because, this one has some issues. >> What are its issues? Besides being too, near 10th Street? >> well, yeah, apart from that it doesn't really have any very close neighbors ,on the, on the map. So the closest that you get, is, I don't know, maybe that red one? >> Maybe. >> But I would be really, I'd be very wary of just using that as my guess, because, it's also pretty darn close to a bluepoint. >> Yeah. >> And also not so, far from the green point. >> That's a good point. So, this whole nearest neighbor thing doesn't quite work, in this case when you got a bunch of neighbors that are saying different things. And they are kind of close to you. So, any clever way we might around this? >> I would say, move the black dot, to, >> No, no, no, no, we are not allowed that before. >> No? Okay, right it seems, it seems, like it would be helpful. >> No, no, they are federal laws ,against interesting. >> I was going to say, yeah, so, alright So, short of that, maybe ,we just need to look at a bigger context? >> Ahh, that makes sense. So, you're saying my little nearest neighbor thing, sort of worked ,but the problem was I started out with examples ,that were, you know, very clearly in a neighborhood, and now I'm in a place where I'm not so sure about the neighborhood, so I should let I should look at more of my neighbors, than just the closest one.

4. Okay, so, how man do you want to look at? >> Well in this case it, you know, I feel like I could draw a, a kind of extended city block zone and capture maybe, I don't know, five of the points. >> Okay, let's do it. So let's find our five, our five nearest neighbors. So let's see. This is clearly close, that's one over here. I'd say this is close. I'll say this one is close. This one's close. None of the other blue ones are actually that close. And I'd say that's the next closest one, so here are my little five points. That all seem relative near. So what does that tell you? >> Well, I mean, it's, it feels like it suggests that red is not a bad choice here. >> Mm >> It's in a reddish part of town. >> Yeah, I get that. So, so you think it's a pretty, fair thing to bet that this should be red then? >> Yeah I mean I think that if you were really asking me seriously I would wonder about that blue point to the right of the highway and whether that had any influence. >> That's pretty far away. >> Yeah, it's not that far away. >> Well in Atlanta, once you cross highways you might as well be an infinite distance away. >> Well so, okay, but. That's a good point then. So, I guess I was interpreting your notion of distance as being, you know, like straight line distance on the map. But maybe that doesn't make sense for this kind of neighborhood example. >> Hm, no, that's a good point. So, we've been talking about distance sort of implicitly. But this notion of distance. It's actually quite important. So maybe distance is straight-line distance, maybe it's as the crow flies. Maybe it's driving distance. Maybe it has to take into account the fact that, when you cross highways in Atlanta, you're typically moving into a completely different universe. These sorts of things matter. >> Yeah. So I could imagine I don't know, like Google Maps distance. >> Right. Or how many paths can you get there and which is the shortest one given the traffic? There's all kinds of things like that you could do. So. So that's fair, that's fair. But that just says that this, this distant, we have to be very careful what we mean by distance and that's okay. But let's just say for the sake of this discussion that these are the closest points by some reasonable measure of distance. So, in that world, would you be happy if you had to pick a single example? a single output, a single label of red ,uh, blue or green. Would you be happy picking red? >> Yeah, I mean you know, not ecstatic, but okay. >> That's fair. So, I like this. So, we, we went from just picking our nearest neighbor to picking

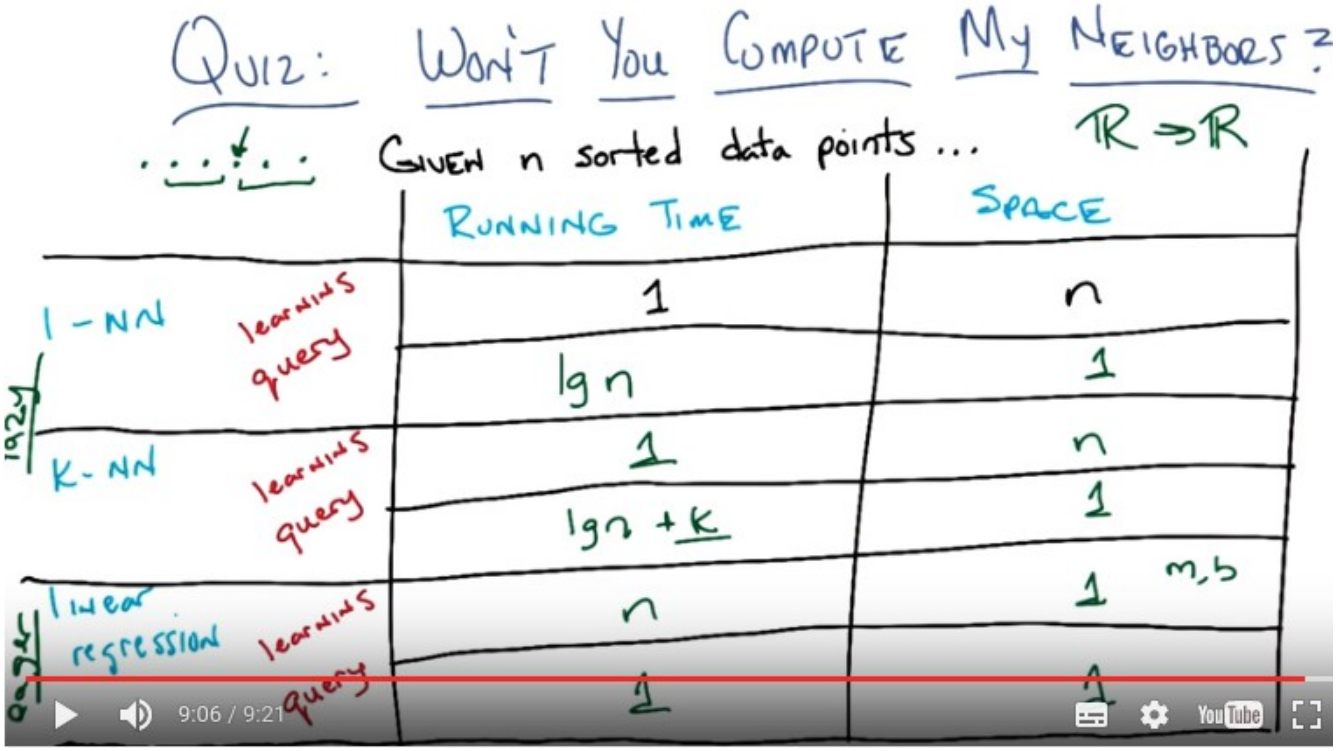
our nearest neighbors. And ,what's a good value you think we should, we should stick to with neighbors? We started with one and that clearly wasn't good. You picked, at least not in all cases and you came up with five. So what do you think? What, what, if I'm going to call this algorithm something, what do you think five nearest neighbors? What do you think? What should I call it? >> Five seems good. I mean I feel like that, that's gotta be universal. >> The number five? >> Yeah. >> Well it is in Atlanta but it might not be univeral in wherever it is you are. >> We'll call it the Georgia Tech nearest neighbors. >> That doesn't seem like an algorithm that's going to to be used very much. >> Fair enough. All right. So what about, we could do as many nearest neighbors as is appropriate. Or maybe we should just make it a free parameter and call it K. >> Ok, I like that. K nearest neighbors, so we'll have K nearest neighbors. And we'll pick our K numbers. Oh, and you said something fancy there, by the way. You said free parameter. I like that. We should, we should come back to that again. So we have an algorithm, k nearest neighbors. Which takes k nearest neighbors as a way of deciding how you're going to label some query point here. And we've identified two parameters to the algorithm so far: k, which is the number of neighbors we're going to use, and some notion of distance. >> Oh, sure. >> Which here we were kind of using in the sort of obvious way, but there might be other ways we might want to use distance here. >> Yeah, like I could imagine if the houses, if, had additional features like how many Square footages they had. >> Right, stuff like that. That would make perfect sense. So, so really distance, we're using distance here in a kind of in an over loaded sense, because this is something on a map. But really distance is a standard for similarity. >> Similarity, good. It's kind of standard for the opposite of similarity. >> [LAUGH] Well distance is just a kind of similarity, right? But in case of, you know, points on the map. Similarity, it sort of makes sense because as you said when we were talking about real estate, location, location, location matters. So, there, similarity really is kind of the inverse of distance. But in other ways, things like the number of veterans you have, whether you're one on side of the highway or the other, the school district you're in, things like that, are other things you might add as features or dimensions when you talk about similarity or distance. Okay, so I like this. I think we have a general algorithm now and I think it does a pretty good job of addressing the points you brought up. We no longer have to worry about overfitting as much, at least it seems that way to me. And we have a way of being a little bit more robust to this, you know, not having an exact data point in the database. So ,maybe we should turn this into an algorithm. >> Yeah, let's go for it. >> Okay, let's do that.



plurality: 選舉中的相對多數票 (即 mode: 眾數)
Classification 後面是 vote of the y_i ...

5. Okay, so what we have here, again, is pseudocode for our K-NN algorithm. And I'm sort of writing it as like, a function. So, you're going to be given some training data D, that's the little x, y points, x y one, x y 2, x y 3, so on and so fourth. You're given some kind of distance metric or similarity function. And this is important because this represents the domain knowledge as I think we, we've already said. You get some number of neighbors that you care about, k, hence the k and n, which also, by the way, represents domain knowledge. Tells you something about how many neighbors you think you should have. And then are given some particular new query point and I want to output some kind of answer, some label, some value. So the k-NN algorithm (k-NN 就是 k Nearest Neighbors 的意思) is remarkably simple given these things you simply find a set of nearest neighbors such that they are the K closest to your query point. >> Okay. I'm sort of processing this. So the, the data the capital D. Are those pairs and there's a set of pairs? >> Yes. >> Ok. And k smallest distances. So this NN this is a set? >> Yes. >> And it's consistent for all the elements in the data that are closest to the query point? >> Yep. And the so the query point is a parameter of that. Okay. Yeah. Alright. I think I. Oh. And then it's, then the so it's just return. >> Yeah, so we haven't figured out what to return. So there's two separate cases we've been talking about so far. One is where, we're doing classification, and one is where we're doing regression. So, a question for you would be, what do you think we should when we're doing classification? Sort of, what we were doing before on the map. What will be a way of returning a proper label? >> So you want to label, not a, like a weight on a label or something like that? >> No. I want a label. You have to produce an answer. You have to commit to something Michael. >> Alright. Can I commit to more than one thing? >> Nope. >> Okay. So I would say that a reasonable thing to do there would be. Did we get Ys associated with the things in NN? >> Yeap. >> So I would go with they should vote. >> I like that. I think that's a good one, so we'll simply vote and what does it mean to

vote? >> It means, let's see, so feel like there would be a way to represent it in terms of NN, the set. Like do you want me to write it formally? >> No. >> Oh, then I would just say The closest point. Whichever yi is most frequent among the closest points wins. >> Yeah. Right. So you want to find a, a vote of basically a vote of the yi's, that are apart of the neighborhood set. And you take the plurality. >> Plurality I see. So it's whichever one occurs the most. >> Right. >> What if there's ties? >> It's the mode. The mode. Right. >> Right. >> Mmmm. Ala mode. >> What if they're ties(平局)? That's a good point. Well, if they are ties among the output, then you're just going to have to pick one. >> OK. >> And there's lots of ways you might do that. You might say, well, I'll take the one. That is say, most commonly represented in the data period. Or I'll just randomly pick each time, or any number of ways you might, you can imagine doing that. >> The one that's first alphabetically. >> The one that's first lexicographically? >> Hm. >> What about in the regression case? >> Okay. So in the regression case our y-is are numbers. >> Uh-huh. And we have the closest Yi's, so we have a bunch of those numbers and it seems like [LAUGH] if you have a pile of numbers and have to return one, a standard thing to do would be to take the average, or the mean. >> Yeah. Now let's just simply take the mean of the Yi's, and at least there, you don't have to worry about a tie. That's right. Though, I guess, you know. We didn't really deal with the question of what happens if there's more than k small. It's, like, what if they're all exactly the same distance? All n of them are exactly the same distance. So which are the k closest? >> Well, there's lots of things you could do there. I guess what I would suggest doing, is, take the, If you have more than k that are close, that are closest because you have a bunch of ties, in terms of the distance. Just take all of them. Get the smallest number greater than or equal to k. Okay. >> That seem reasonable? >> Yeah, I think that's what college rankings do (可能是指大學排名時, 好幾個差不多的大學共用一個名次). >> Actually, that is what college rankings do, and then they, yeah, that's exactly what college rankings do. So, let's do that. We know that college rankings make sense. [LAUGH]. Yeah, those are, they're scientifically proven to be, >> Youths. >> scary, scary to people in colleges. >> That's exactly right. So, here's what we've got, Michael. So, all we do is we take the training data. We have some notion of similarity or distance. We have a notion of the number of neighbors that we care about. We have a query point, we find the K closest to one, you know breaking ties accordingly. And then we basically average in some way, in a way that make sense for classification, in a way they make sense for regression and we are done. It's a very simple algorithm, but some of that's because a lot of decisions are being left up to the designer. The distance metric. The number k, how you're going to break ties. Exactly how you choose to implement voting. Exactly how you choose to implement the mean or the average operation that shows how to do here. And you could put a bunch of different things here and you end up in, completely, you could end up with completely different answer. Mm. >> By the way, one thing that you might do, just to give you an example of just, how much range there is here, is rather than doing a simple vote by counting, you could do a vote that is say, weighted by how far away you are. So we could have a weighted vote. >> Uh-huh. >> That might help us with ties. >> That could help with ties. Yeah. >> You could do a weighted average. Yes, right. So, you're basically saying that the y values that correspond to x values that are closer to the query point have more of an influence on the mean. >> Which makes some sense, right? >> No, I think it makes a lot of sense! >> So, how would you weight that? What would you do? >> I would weight it by the similarity. >> Right, so well in this case, the similarity is we have a distance value similarity, so You would have to, you know, weight it by something like one over the distance. >> Oh I see. Okay. That seems like a hack. >> Sure but it's a hack that sort of makes sense. >> Okay. >> Okay. So anyway. Simple algorithim. Lots and lots of decisions to make here. All of which could in principle have a pretty big effect. And so, in order to see that, I want to do two quizzes that I hope get to heart of this and maybe give us a little bit of insight into how some of these decisions might matter on the one hand, and exactly just how simple or not simple this algorithm turns out to be. Okay? >> Awesome.



下面兩段(6 和 7 段)都在講此圖.

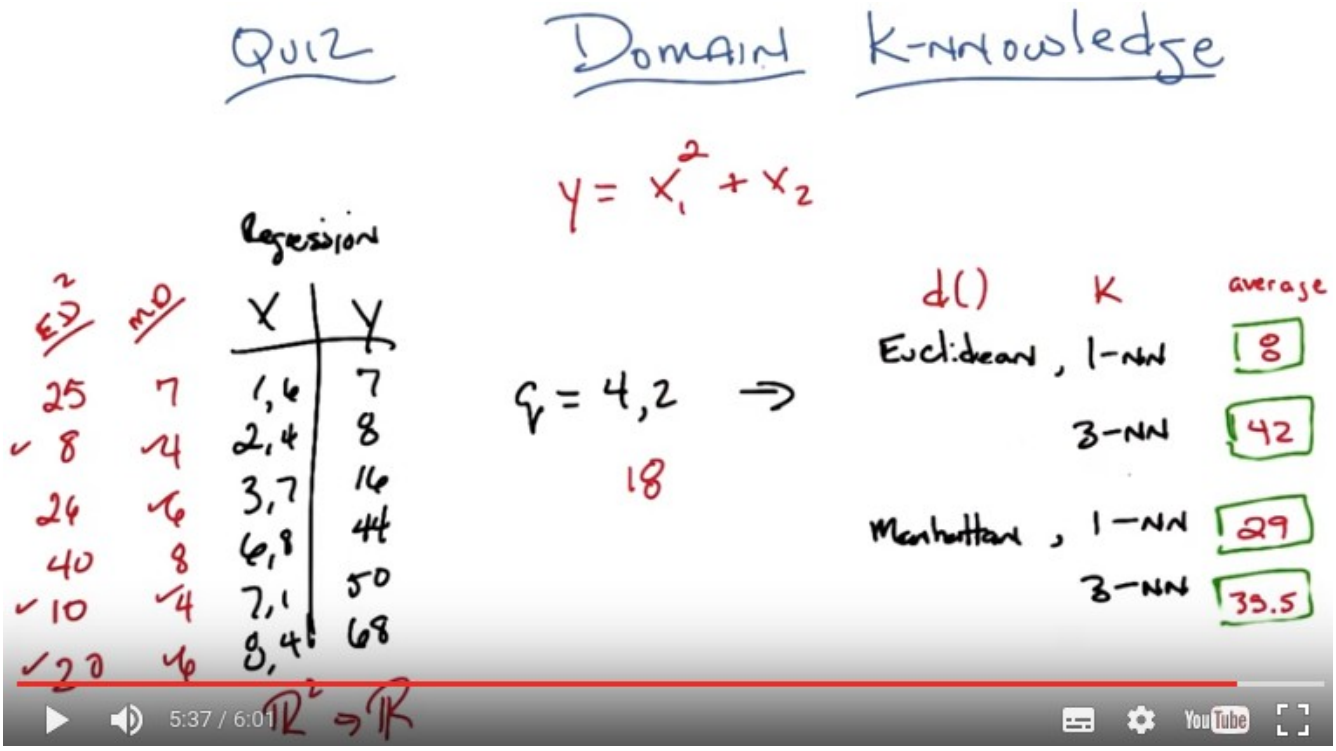
6. Okay Michael, I have two quizzes for you. Okay? >> Yeah, yeah. >> Here's the first quiz, and here's the way it's set up. I want you to fill in the empty boxes of this table. Okay? >> Ooh. >> Got it. >> There's a lot of empty boxes. >> There's a lot of empty boxes. >> Okay, but Okay, let me make sure I understand what's going on here. So we're looking at three different algorithms that are learning algorithms. >> Yep. >> There's one One neural net >> No >> Okay, one nearest neighbor (1-NN). >> Mm-hm >> K nearest neighbor and linear regression. >> Yep >> And for each one you want to know running time and space. >> Mm-hm. >> And this is on n points I assume, yeah, n sort, what does it mean for data points to be sorted? >> So let's assume we're living in a world where all of our data points are you know in r one. Okay. >> Oh okay that well that. That could be sorted. >> That could be. Yeah that could be sorted. And that you know where going to be out putting some real numbers as well. So it points on a. On a number of lines. So to make things simple for you. I'm going to say that the points that your given are already sorted. >> Oh ok alright. And yeah that makes sense. Its just a scalar (即每一個 data point 就是一個數, 而不是有兩個分量的矢量). So then a query point is going to come in. And then its going to be some value. And were going to have to find the nearest neighbor or do the [UNKNOWN] regression or whatever. >> Right. >> Alright now that's for running time. For now space your talking about the space of what. >> How much space you are going to have to do in order to accomplish your task. How much space you going to have to use in order to accomplish your task? >> So this is kind of like the the. The space that's representing the class enviro. Or the regression. After training. >> Yes. So actually that question about after training is important. You'll notice I've divided each of these algorithms into two phases. There's the learning phase. How much time it takes to learn. How much space you need to learn. Then there's the query phase. When I give you some new value and you have to output and answer. What's the running time for that and what are the space requirements for that? Okay? You got that? >> Yeah >> I want that for each one. Of these three algorithms. >>

Except for one nearest neighbor(1-NN) which the, it appears as though you filled in for me to get me started. >> Right so just to get you started and make it easier for you know to know what I'm talking about. I'm talking about big O times here. Right. I'm not going to make you write out big o. Big O is implicit. So if we look at one nearest neighbor (**1-NN, learning**), and we ask well what's the running time of learning? Well, it's constant. Right? Because there's no learning. >> I see. You just take that sorted set of data points and you just pass it along through the query here. >> Right. Now, you could say that" Well, I'm going to take the data points or I'm going to copy them into this database," and so it's linear. But let's assume they already come in a data base, or some data structure that you can use, okay? >> Gotcha. >> Okay, so now that actually brings us to a nice little question about how much space, learning takes here. And, well because you have to store those points, and keep them around. The space requirements are big O of N. >> Yeah, that makes sense. >> Okay, good. So given that as an example. Do you think your one example in your data base. Mm, do you think you can use that to build up labels for all the rest of the phases of the different algorithms? >> Yeah, I think so. >> Okay, cool. Go for it.

7. Okay Michael, are you ready? >> I am afraid so. >> All right, which one do you want to fill out first? >> Let's just do them in order, so, one nearest neighbor (**1-NN, query**). You, you explained, how the training works. We just take the assorted list, and leave it there. >> Mm-hm. >> And we have the classifier, or the aggressor itself ,has linear space, and now at query time, we need to find the nearest neighbor, Um-huh. >> Which we could do by taking the query point, and running through the whole list ,and seeing which one it's closest to, but, because, it's sorted I think we, we outta be able to use binary search and, and in log time, find the closest, point to the query. >> That's exactly right, you should be able to do that in log base, two time. What if it weren't sorted? >> Yeah then, like I said, I think you could just scan through the whole list and that would be linear time and that's not a big deal. >> Right, yeah, we could do linear time, but, because I gave you a sorted list, because, I'm so helpful, you can do it in [INAUDIBLE] time, okay, but. >> That was, that was very, very thoughtful of you. >> It was I thought, I thought of her, so what about on the, space side? >> Alright, so the amount of space that you need to process its query is linear. We don't need to take any special, set aside, space beyond, a couple simple variables. And the data that we're given, which we've already accounted for. >> Right, so then why would it be linear, if we accounted for it? >> Did I say linear? I meant constant. >> Yes, yes, that's right, constant. That's what you meant. That's what you said. That's what happened. >> [LAUGH] >> Okay. >> It's a good thing, this wasn't being recorded, so we could verify one way or the other. >> [LAUGH] It is a good thing, maybe we'll look it up on Wikipedia, and it'll say confusingly [LAUGH] >> Linear sometimes use to mean constant. >> Constant. [LAUGH]. Yeah, that is pretty confusing. >> Okay, what about **K-NN**? >> Alright, K and n. So k and n, so the training process, the **learning** process, is exactly the same, as it is for one year stamper, which is to say you do nothing and you pass all the data forward, to the, the query processor. So it's going to be 1 (time) n (space). >> That is correct, nice. >> Now **querying**, seems like its a little more subtle. So we can find the single nearest neighbor in log n time. >> Mm-mm. >> Where we going to get the other K minus one? So, I'm pretty sure, that ,once we find the nearest neighbor we can kind of start doing a little, Spread out search, from there, until we found the k nearest neighbors. >> Sure. So, you're saying, you know, you've got these points. They're already in a line, you find the nearest neighbor. You know ,the, the next nearest neighbors have to be within k of the points surrounding it, and so you can just move in kind of, either direction and pick them up as you go. Yeah, something like that. >> Okay. >> I mean the way that, the way that I was thinking about it is that, I, I think you can use the same algorithm that you used for merging lists, in merge sort, but here the lists actually corresponds, to being, to the left of the query point, and being to the right of the query point and they are both sorted ,in terms of their distance from the query point. Sure, yeah, I buy that. >> So, so that ought to give us log n + k (time). >> Okay, so, do we need to write the k? >> I'm going to say yes, because, if k is on the order of like, n over 2, then it's

going to dominate. If k is on the order of log n, then it's not going to dominate. Mm, that's a good point. So, yeah, we'll do k. I will point out that if k is on the order of n over 2, you're right, it will dominate, and then really this is big o of n. >> That's right. >> But if it's on the order of log n, then it's just log n plus n, and so it's a big old long n, log n. But ,you're right, so we should probably keep the k around because, we don't know its relationship to n. Okay, fair enough. Okay, what about the space requirements? >> We know one bit of relationship (應該是指 k 和 n), it's smaller than or equal to it. >> That's true. >> Because that would be really weird, if I gave you ten data points and asked you for the 20 nearest neighbors. >> That's the sort of thing you would do. >> It's the sort of thing you would do, but then ,it would be really confusing. >> No, no, no, it's the sort of thing you would do, again, let's go to Wikipedia. >> Confusingly, Twinny is sometimes [CROSSTALK]. [LAUGH] >> Okay. So what about space? >> Space, so, I don't understand why, it would ever need more than constant space. So, so we're going to, zip around in that. I mean, I guess, if do it really badly ,we can use k space. To kind of copy over what those, possible nearest neighbors are. But ,we don't need to keep track of them. We can just point to them in place, so it's constant. >> Okay, yea that's true in fact, because, it's sorted all you really need to know is the beginning and the ending. So, that's two things that's constant. Okay, cool, alright, good, so what about **linear regression**? Your favorite little algorithm thing that you did? When we talked about this before. >> I do like, linear regression. The **learning** in this case, is what we are mapping, real number input to a real number output. The way we are doing that is we are taking, its probably m * x + b sort of form. We need to find, the multiplier and the additive constant. which, It seems, like, in general doing a regression involves, inverting a matrix. But, in this case,I think the matrix that we're talking about is of constant size. So inverting, is, constant time. I think, it's as easy, as basically just scanning through the list (例如求係數時的公式中的 Σ x_i 之類的), to populate that, that, constant size matrix. So, I'm going to say order n (time). >> Yep. >> To process the data. >> That is correct. >> There's probably a really nice algorithm for that. >> Yeah, probably some kind of linear regression algorithm. >> Yeah. [LAUGH] No, I mean like the general linear regression algorithm is, is involves inverting a matrix. >> Right. >> Or something like it, something equivalent to it. But here because, we're, it's all in scaler land, I think it's simpler. >> Yeah, I think that's right. Okay ,so what about space? >> All right, so space interpreted the data that you passed forward from the learning algorithm, to the regressor, is just, well m * x + b. It's just m and b, which is constant. There's the two numbers. >> Right, that's 2, 2 is like 1, [UNKNOWN] large values [LAUGH] of 1, so it's constant. >> Yeah, All right, now at **query** time, you give me an x. I multiply, it by m and add b, so that's constant time [CROSSTALK]. So, before the query, cost was expensive and the learning cost was cheap. And now we've kind of swapped that around. >> Yeah, we have, so space would be. >> Space, oh, that you're asking me? >> Yeah. >> Space for the query would be constant as well. >> Right, exactly, so yeah, so you made a good point here. So earlier on, we had the situation where learning was fast, was constant, and querying was, you know, not as fast. It was, you know, probably logarithmic. But, in the regression case, learning was expensive and the querying was easy, so we swapped around, that's exactly right. So, why would you care about that, well, let's see, I'll point out something, which is though, even though we swapped out what was expensive in terms of time and what wasn't, you'll notice that. It's only logarithmic at query time for these first two (1-NN & K-NN) but it's linear for the learning time, in, linear regression, so doesn't that mean that linear regression is always slower and worse? >> No really, because we only have to learn once, but we can query many, many times. >> Right, right. So I guess if we query more than, you know, n times for example ,it'll certainly be, worse overall. In terms of running time. >> That's right. >> Okay. >> Though, it's, though it's interesting, because like when I see numbers like this, my, my algorithm, hat tells me that I should try to balance them a little bit more. Like, here it's, it's you can make, learning ,essentially free and the other one you can make querying essentially free. Really you want to split those, somewhat evenly. Though it's, not obvious to me how you would do that. Like, square root of n, learning time and then square root of n query time or something like that. >> Yeah, but you did say something else that was important right?

Which is that you only have to learn once. >> Yeah, It's true. >> So, the balance you know, really depends on how often you're going to do querying and exactly, what power it gives you. I mean, the trade off is really there. Don't you think? >> Yeah, I guess so. I mean so, so specifically, in the, in the version where you just query ones. >> Right. >> Then the balance thing could be more interesting. >> Sure, okay, cool. All right anything else you want to observe about this. Let's see, we got the trade off between learning versus querying. So, either you do all your work upfront (即 linear regression, 因為 learning 花較多時間, query 不花時間), or you put it off and do your work only when you're forced to at query time (即 K-NN, 因為 learning 不花時間, query 花較多時間). >> Yeah, I guess, well one thing, is, I want to point out that there's a nice Mr. Rodgers, reference in the title. That was, that was very cool. >> Thank you very much. And the second thing, is that it does strike me, in a sense, that what's going on here for the nearest neighbor algorithms, is that you just put off doing any work until you absolutely have to. >> Mm-hm. >> Which strikes me as kind of a procrastinatory(拖延) approach. >> So that's a good word, that you used there, procrastinate. The words that people use, in the literature are lazy. >> Mm. >> They say that these (1-NN, K-NN) are lazy learners, versus something like linear regression, which is an eager learner. >> Eager. >> Yes. So, linear regression, is eager, it wants to learn right away, and it does, but nearest neighbor algorithms are lazy. They want to put off, learning until they absolutely ,have to, and so we refer to this class ,as lazy and this class as eager. >> I See, so I guess its the case, that, if we never query it, then the lazy learner definitely comes out ahead. >> Right, that makes sense. >> Yeah. >> It's just in time learning, or JIL. [LAUGH] Or JITL, I guess, which doesn't roll off the tongue anywhere near as well. Okay, cool. So we've gotten through this quiz. Would you like to do another one? >> Yeah, I just have to get this JITL off my tongue. >> [LAUGH]



下面兩段(8 和 9 段)都是在講此圖.
X 是 input, Y 是 output.
最左邊的 ED = Euclidean distance, MD = Manhattan distance.
ED^2 即距離的平方

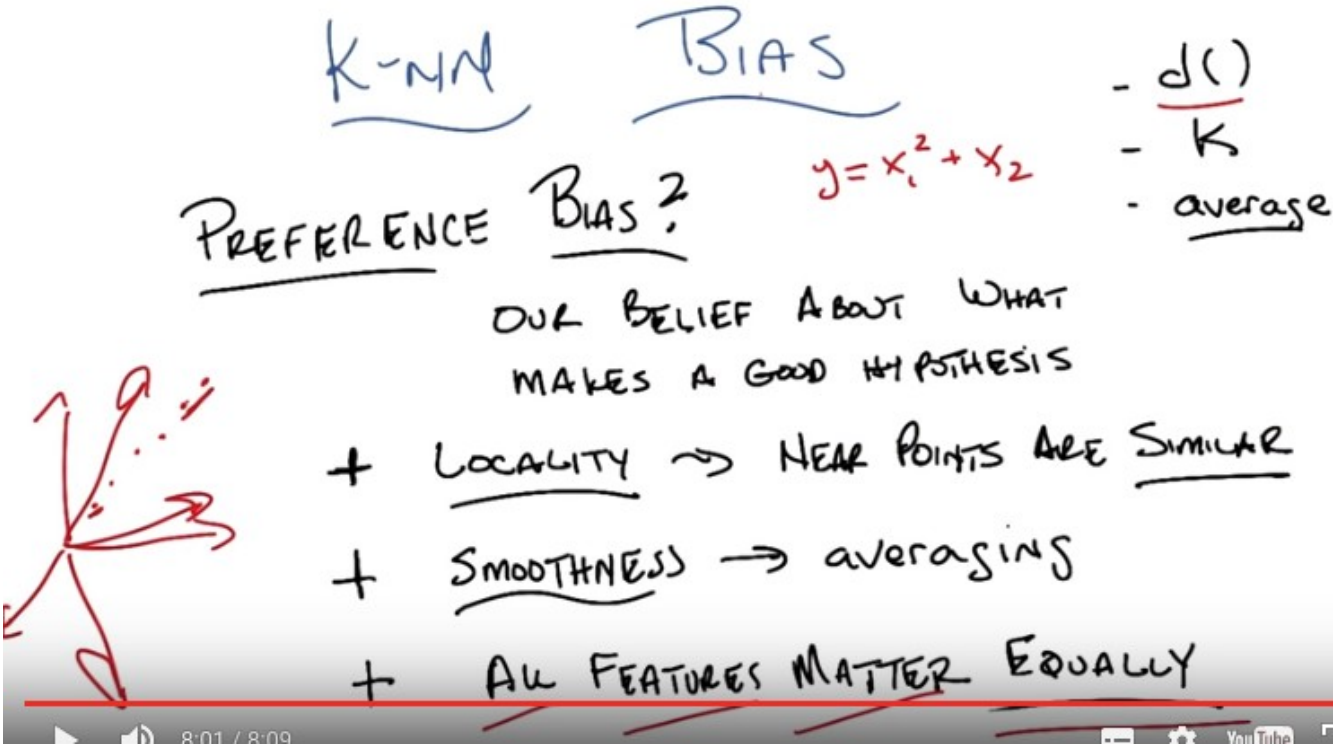
From online: Manhattan distance: the distance between two points is the sum of the absolute differences of their Cartesian coordinates,

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

8. Okay Michael, so here's our second ,quiz, is a row. In the last quiz, we talked about running time and space time, but now we're going to talk about ,how the k-nn algorithm, actually works. And in particular how different choices, between distance metrics, values of k, and how you're going to put them together, can give you different answers, okay? So, what I have over here on the left is training data. This is a regression problem and you're training data is made up of xy pairs. X is two dimensional. Okay? So this is a function from R squared to some value in R1. Okay? >> Mm-hm. >> So, the first dimension represents, something and the second dimension, represents something. And then there's some particular, output over here. And what I want you to do, is given a query point, 4, 2 produce what the proper y or output ought to be, given all of this training did. You're with me? >> Yeah. >> Okay, so I want you to do it in four different cases, I want you to do it in the case where, your distance matrix is Euclidean, Okay. >> The distance metric, in R2? >> Yes. >> Oh I see because, we're going to measure the distance between the query and the different data points. >> Right. >> Yeah. Okay. Uh-huh. >> Mm-hm. So it's euclidean, for a case of one nearest neighbor and three nearest neighbor and I want you to take, for example, in the three nearest neighbor (3-NN) case. I want you take their output and average them. Okay? >> Okay. >> Now, in the I also want you to do the same thing. But in the case where instead of using Euclidean distance, we use Manhattan distance. But again, for both 1 nearest neighbor and 3 nearest neighbor And in any case where we have ties, like in three nearest neighbor where we absolutely have to have at least three of these things show up, just let 'em average. Okay? >> Got you. >> Now we're doing averaging instead of straight voting, because, this is a regression problem. >> Got it. >> Okay. Any questions? >> Maybe. Let's see. Three nearest neighbor. And so if there's ties, we, we use the college ranking trick of including everybody who's at least as good as the k, largest or k closest. >> Yes, exactly. >> Okay, yeah, no I think, I think I can take a stab at this. >> Okay, cool then go.

9. Alright Michael, you ready? >> Yeah. >> Okay. What's the answer? Walk me through. >> I will walk you through. Alright. So let's, let's do this. Let's write the Euclidean distance. Well let's write the Manhattan distance, because I don't, I don't want to take square root to my head. >> Okay. >> Let's write the Manhattan distances next to the Xs. >> OKay. >> Or the Ys. >> Alright. >> Either way. >> Let's do it next to the Xs. So this is the Manhattan distance or MD as the cool kids call it. >> [LAUGH] >> Is that true? >> Yea. >> The cool kids called it L one. >> No, no, no have you ever heard a cool kid ever say something like L one? >> Well, to me the cool kids are the people at neps who know more math than I do. Yea, do you think any of them are going to watch this video? Actually I'm afraid all of them are going to watch this video. >> Now I'm really afraid. >> Mm-Hm so you better get it right, every ones watching. >> All right, well let me do, let me complete the Manhattan distances. So the first one what you do is you take the 1 minus 4. >> Mm-Hm. >> And that's three. And you take the 6 minus 2 and that's 4. And you add the two together and you get 7. Which interestingly is the same as y, but I think that's a coincidence. >> Okay. [CROSSTALK] >> And now I'll do all the rest of them 'cause I pre-computed them of Four, six, eight, four, six. Alright, so now we've got ti set up so we can do one and three nearest neighbor relatively quickly. So, the one nearest neighbor, the closest distance, is 4. >> Mm--hm. >> But unfortauntely there are two points that have that (2, 4) in set number one. We have outputs of 8 and 50 because they. Almost agree with each other. >> Uh-huh. >> Not. And if we take the average of those two things we get 29 (這其實用了前面說的college ranking trick). >> Yep. That is correct Michael. >> Great now in terms of the three nearest

neighbors we have the fours and the sixes (即距離為 4 和 6 的點, 總共有 4 個這樣的點, 多於 3-NN 中的 3, 故又要用 college ranking trick). >> So the four, three nearest neighbors. >> Yep. >> Somewhat awkwardly. And the we have the average of those things which is what. 8, 50 and 16 and 68 which gets us 35.5. >> Right. Obviously. [LAUGH] Okay. Alright, so that was pretty straightforward. And those answers aren't too far off from one another. So what about the Euclidean case? >> Alright, so one thing to point out. I, I was worried about computing square roots but it occurs to me that I actually don't have to compute square roots because that's the monotonic transformation, and we only care about the orders. >> Hm, okay. >> So for Euclidean distance, or as I like to call him casually, ED, >> Mmhm. >> We can just take the square differences summed up. >> Okay, so this would be ED squared. >> Yes, it would be ED squared. >> Okay, ED. >> Good. So the first one, it'll be the one minus four is three, squared is nine. And the 6 minus 2 is 4, squared is 16. And 9 plus 16 is 25. So the first one will be 25. >> And notice the square of 25 is pretty easy to compute. >> Yeah, but the other ones aren't going to be. It just so happens that we've got a pythagorean triple on our hands. >> Mm. I love those. >> Al right so the remaining ones, the x squared are eight, 26, 40, ten, and 20. >> Hm, none of those are easily square rootable. >> Exactly, though 40 feels like it really was trying and failed. >> Yeah. An eight over shot and now it's a perfect cube. So, eight is the smallest distance. >> Yep. >> And again, seemingly, coincidentally, they Y value associated with that, is eight. >> Hm. >> So an eight, eight is our answer. >> Good and that's correct. >> And the three closest are eight, ten and 20. >> Mm-hmm. >> And if we average the Y values for those that's eight, 50 and 68, which gives us an average of 42. The meaning of life, the universe. And pretty much everything? >> Yes! And that is absolutely correct. So that's kind of That's kind of cool that you get completely different answers depending upon what you do. >> Yeah, it does seem very different, doesn't it? I mean there's like several orders of magnitude spread here. >> Well, that's. >> Maybe not orders of magnitude but orders of doubling. >> Yes, there are orders of doubling spread. Well, you know what Michael, I actually had a specific function in mind when I did this (即他列左邊的數据時, output 是通過一個函數得到的). >> Okay! Let's find out which one is the right one! >> Well, the function I had was (Y=X_1^2+X_2) Y was equal to the first dimension squared plus the second dimension. So, let's call that X1 and X2, and this was actually the function that I had in place. So, you square the first term and you add the second. >> Okay, and so like looking at the second last one, for example, seven squared is 49 plus one is 50 Oh, [UNKNOWN]. It's very consistent. >> Thank you. So what would be the actual answer for q = (4, 2)? >> Okay so four squared is 16 plus two is 18, which is close to none of them. >> Right. So there's a lesson here, there's several lessons here. And one lesson I don't want you to take away. So here's the lesson. So I actually had a real function here. There was no noise. It was fairly well represented. The proper answer was 18 and basically none of these are right. But the first thing I want you to notice is you get completely different answers, depending upon exactly whether you do one versus three, whether you do Euclidean versus Manhattan. And that's because these things make assumptions about your domain that might not be particularly relevant. And this sort of suggests, that maybe this thing doesn't do very well. K-NN doesn't do very well because none of these are close to 18. That seems a little sad. But I've good new for you Michael. >> Okay. >> The good news is that, actually K-NN tends to work really, really well. Especially given it's simplicity, it just doesn't in this particular case. And there's really a reason for that, and it has to do with this sort of fundamental assumptions in bias of K and N, I happen to pick an example that sort of violates some of that bias. So I think it's worth to take a moment to think about what the preference bias is for K-NN and to see if that can lead us to understanding why we didn't get anything close to 18 here. >> Okay that sounds useful. >> Okay, so let's do that.

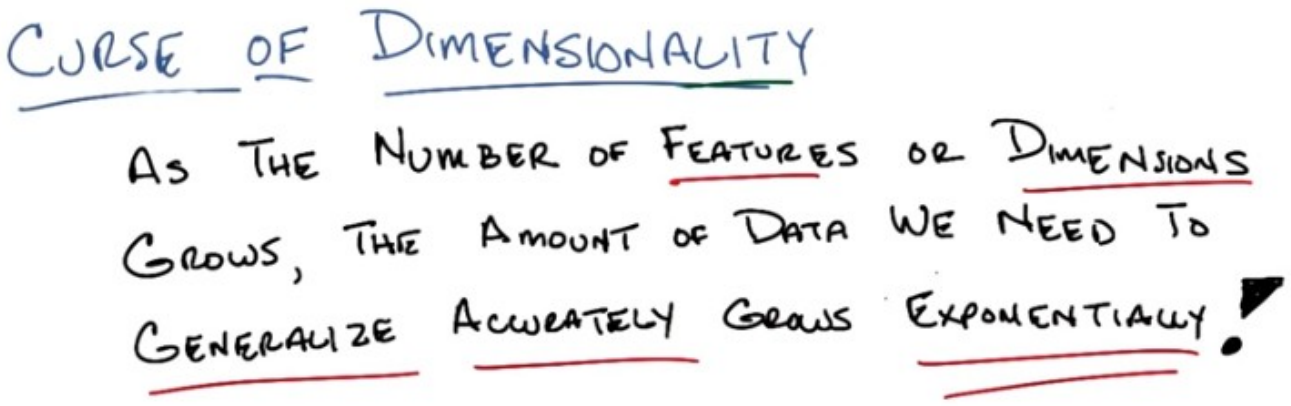


最後一點的 features 是指 前面的 X1, X2, …
後來我覺得 features(和 dimension)應該是指數据的 attributes, 例如 UCI Wine Quality Data Set 中的那 11 個 attribute(如 fixed acidity, residual sugar 等), 由這些 attribute 可以推知 quality.

10. Ok, Michael, so I'm going to talk a little bit about bias. In particular, the preference bias for K-NN. So, let me remind you what preference bias is. Preference bias is kind of our notion of why we would prefer one hypothesis over another. And they say all things, other things being equal. And what that really means is, it's the thing that encompasses our belief. About what makes a good hypothesis. So in some of the previous examples that we used it was things like shorter trees, smoother functions, simpler functions, [UNKNOWN] those sorts of things were the ways that we expressed our preferences over various hypothesis. And K-NN is no exception. It also has preference by its built in as does every algorithm of any note. So I just wanted to go through three that I thought of is as being indicitive of this bias. And they're, kind of all related to one another. So the first one is a notion of locality. Right? There's this idea that near points are similiar to one another. Does that make sense to you? >> Yeah. Yeah. That was really important. It came out nicely in the the real estate example. Right. So the whole idea. The whole thing we are using to generalize from one thing to another is this notion of nearness. >> Right. And exactly how this notion of nearness works out is embedded in whatever distance function we happen to be given. And so, there's further bias that might come out, based upon exactly the way we implement distances. So, in the example we just did, euclidian distance is making a different assumption about what nearness or similarity is, compared to Manhattan distance, for example. >> So is there like, a perfect distance function for a given problem? >> There's certainly a perfect distance function for any particular problem. >> Yeah, that's what I mean. Not one that works for the universe, but one, like, you know, if I give you a problem and you can work on it all day long. Can you find, is there a notion that there"s a distance function that would capture things perfectly? >> Well, it has to be the case for any given fixed problem. That there is some distance function that minimizes, say, some of squared errors or something like that. First is some other distance function.

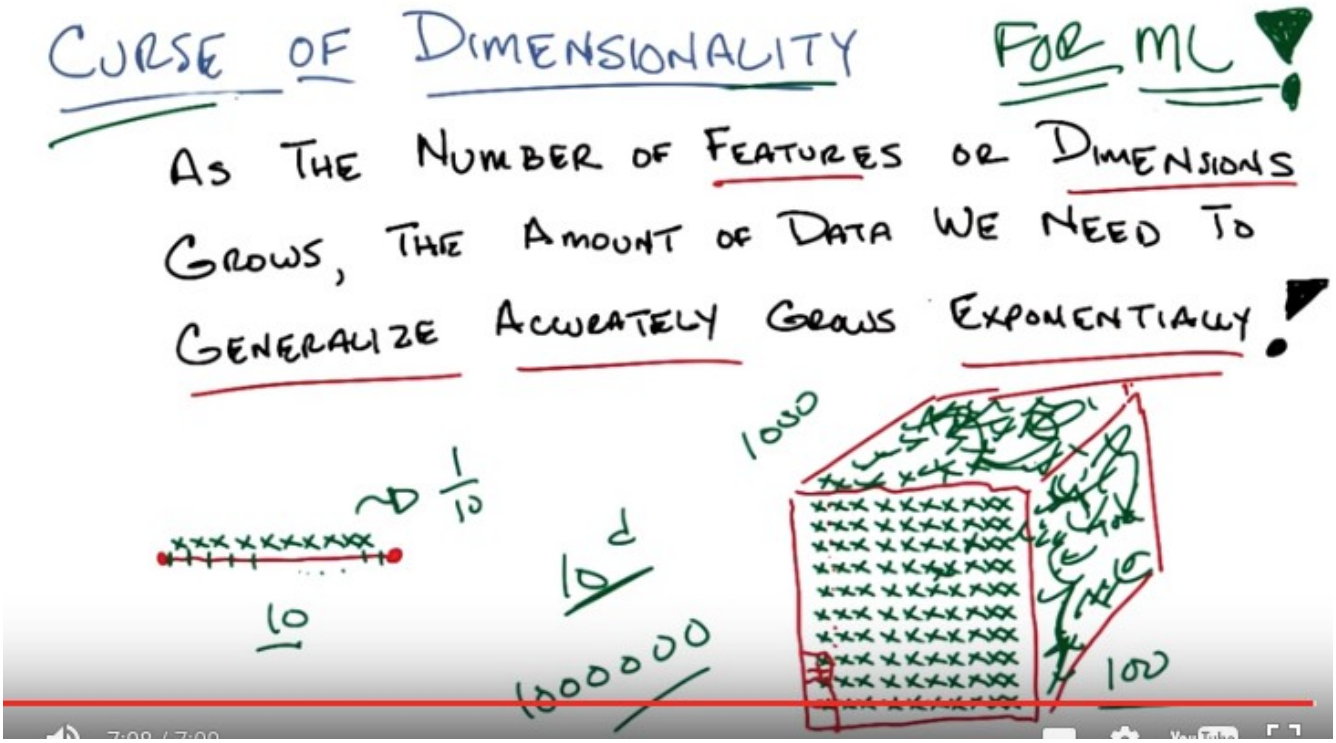
Right? >> Okay. >> That has to be the case. So there, there always to be at least one best distance function given everything else is fixed. >> That makes sense. >> Right. Now, what that is, who knows. Maybe you finding it might be. Arbitrarily difficult. Because there's at least an infinite, there's at least an infinite number of distance [UNKNOWN] >> Well yeah, I was thinking that, that for latter to find distance functions to be anything we want. What about a distance function that said the distance between all the things that have the same answer, is zero. >> Mm-hm. >> And the distance between them and the ones that have different answers is you know, infinity or something big. >> Yeah. >> And then, then the distance function, like, somehow already has built in the solution to the problem because it's already put the things that have the same answers together. >> Right, you could do that, and of course, doing that would require again solving the original problem. But yeah, so. So, such a function has to exist, or, well, you know, there's always noise. What if there's noise in your data, you know? But some such function like that has to exist, the question is finding it. But I think the real point to take there is, there are some good distance functions for our problem and there are some bad distance functions for our problem. And how you pick those is really fundamental assumption your making about the domain. That's why it's domain knowledge. >> Yeah, that sounds right. >> Mm 'Kay. So, locality however it's expressed to the distance function, that is similarity. Is built in to K-NN that we believe that near points are similar. Kind of by definition. That leads actually to the second preference bias which is this notion of smoothness. Alright. That we are by choosing to average. And by choosing to look at points that are similar to one another. We are expecting functions to behave, smoothly. Alright, so, you know, in the two D case. It's, it's kind of easy to see, right? You, you, you, you have these, these sort of points and you're basically saying, look, these two points should somehow be related to one another more than this point and this point. And that sort of assumes kind of smoothly changing behavior as you move from one neighborhood to another. [INAUDIBLE] that make sense? >> I mean, it seems like we're defining to be pretty similar to locality. >> In this case. So I'm, I'm drawing an example, such that, you know, whatever we meant by locality has already been kind of expressed in the graph. >> Okay. >> And you know, by picking, you know, this is really for pedagogical reasons. You know can imagine, this you know, these are points that live in 77,000 dimensions, and it's impossible to actually visualize them much less draw them. And I could try. [LAUGH] But here's, here's three dimensions and here's the fourth dimension. I think I'm going to get tired before I hit seven and seven thousand but, you know, you kind of, you kind of get the idea, right? That, if. In, you know, if you can imagine in your head points that are really near one another in some space, you kind of hope that they behave similarly. Right. >> Right. Okay, so locality and smoothness. And I think these make sense. I mean, these, this is hardly the only algorithm that makes these kind of assumptions. But there is another assumption which is a bit more subtle I think. Which is worth spending a second talking about, which is, for at least the distance functions we've looked at before. The Euclidian distance and the Manhattan distance. They all kind of looked at each of the dimensions, sort of, and subtracted them, and squared them, or didn't, or took their absolute value and added them all together. What that means is, we were treating, at least in those cases, that all the features (即 X1, X2, ...) mattered. And not only did they matter, they mattered equally. Right. So think about the the last quiz I gave you. Right. It said Y = X1^2 + X2. And you noticed we got answers that were wildly off from what the actual answer was. Well if I know that the first dimension. The first feature is going to be squared and the second one is not going to be squared. Do you think either one of these features is more important or more important to get right? >> Okay. Right. Trying to think about what that might mean. So, if, yea its definitely the case that when you look for similar examples in the database you want to care more about X1 because a little bit of a difference in X1 gets squared out. Right? It can lead to a very large difference in the corresponding Y value. Whereas in the x2's, it's not quite as crucial. Th, th, the, if you're off a little bit more, then you're off a little bit more, it's just a linear relationship. So yeah, it does seems like that first dimension needs to be a lot more important, I guess, when you're doing the matching. Then the second one. >> Right so, we probably would have gotten different, I'm

not going to go through this but, (本 blue block 說得不清楚, 看後面的就清楚了)we probably would have gotten different answers if, in the Euclidian or Manhattan case we had instead of just taking the difference between the first two The first dimensions, we had taken that difference and squared it. And then in the case, including this and squaring it again, and then some of those things that were closer in the first dimension instead of the second dimension would've looked more similar and we might've gotten better answer. That's probably a good exercise to go back and do for someone else. >> [LAUGH] Yeah, I was thinking of doing it right now, but yeah, probably should leave it for other people. >> Well you can do it if you want to. So did you do it Michael? >> I did. >> And? >> So it's a kind of now a mix between the Manhattan distance and the Euclidian distance. So, I'm taking the first component, take the difference, square it. >> Mm-hm. >> Take the second component, take the difference, absolute value it. And add those two things together. >> Sure. >> All right. So if I do that, with one nearest neighbor, I still get that tie, but the output answer ends up being 12. >> Hm. Which is better than 24.7. >> And that's better than eight, which is what it was before. So the eight has gone up to 12, which is better than the other one, which I think was 35.5, comes down to 29.5 Close here again to the correct answer which is eighteen. So in both cases it kind of pushed in the right direction, it was using more, of the, the answers that were relevant and fewer of the answers that were not relevant. >> Right. There you go. So the notion of relevance by the way, turns out to be very, very important. And highlights a weakness of K-NN. So this brings me to a kind of theorem or fundamental results of a machine learning that is particularly relevant to K-NN but its actually relevant everywhere. Do you think its worth while to mention it? >> Sure it sounds [INAUDIBLE] relevant. >> Alright let's do it.



11. Okay, Michael. So this notion of having different features or different dimensions throw us off has a name and it's called the Curse of Dimensionality. >> Oh, nice audio effect. >> I did like that effect in post-production. And it refers to well, a very particular thing. So let me just read out what it refers to. As the number of features or equivalently ,uh, dimensions grows that is as we add more and more features we go x of 1, x of two then we add x of three, add more and more of these features. As those features grows or as the number of dimensions grow ,the amount of data ,that we need to generalize accurately also grows exponentially. Now this is a problem of course because Exponentially means, bad in computer science land because when things are exponential they're effectively untenable. You just, you just, you sort of can't win. >> I think everybody knows that in the sense that if you look, I've done this experiment actually, if you look in the popular press like, you know, Time Magazine Or New York Times, USA Today. People will use the word exponentially sometimes to mean exponentially, and sometimes to mean, a lot. >> Yeah that's actually a pet peeve of mine. The whole notion of, >> Me too. >> Oh, it's exponentially bigger. No, that's, that's not meaningful. If you're saying I have one point. And then I have another point, and I want to say this one point is exponentially bigger than this one. That's meaningless! It's also liberally bigger than that one. Exponentially refers to a trend. >>

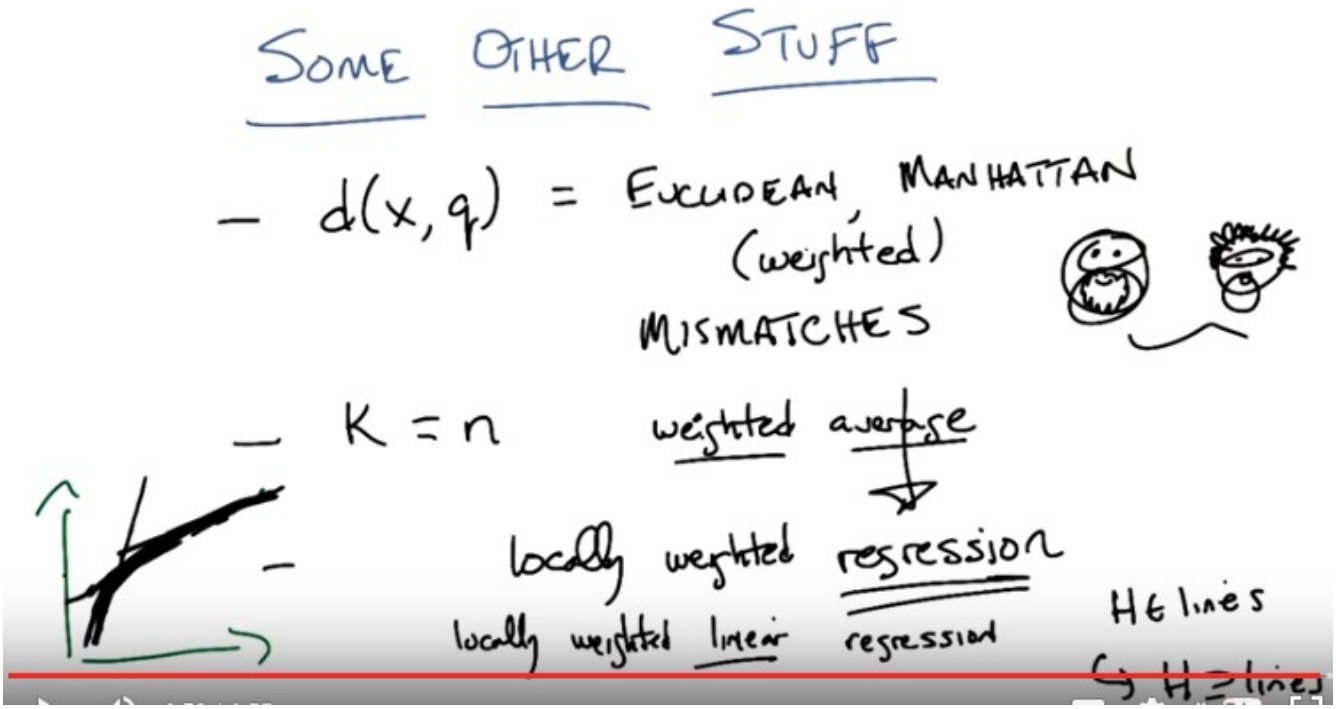
Again, their,their,their not talking about the mathematical relationship. They just mean a lot. >> Okay, so they're wrong. And it bothers me deeply but I'm willing to accept it for the purposes of this discussion. Okay. Exponentially means bad. It means that we need more, and more, and more, and more data as we add features and dimensions. Now as a machine learning person this is a real problem right, because What you want to do, or like what your instinct tells you to do is, oh ,we've got this problem, we've got a bunch of data, we're not sure what's important. So why don't we just keep adding more and more and more features. You know, we've got all these sensors and we'll just add this little bit and this little bit, and we'll keep track of GPS location and we'll see the time of the day and we'll just keep adding stuff and then we'll figure out which ones are important. But the curse of dimensionality says that every time you add another one of these features. You add another dimension, to your input space, you're going to need exponentially more data, as you add those features, in order to be able to generalize accurately. >> Mm. >> This is a very serious problem, and it sort of captures, a little bit of what the difficulties are in k and n. If you have a di, if you have distance function or a similarity function, that assumes that everything is Relevant, or equally relevant, or important, and some of them (indeed) aren't. You're going to have to see a lot of data before you can figure that out, sort of before it washes itself away. >> [CROSSTALK] >> Yeah, that makes a lot of sense. >> Yeah, it seems a little scary. So, you know, I think you can say these words, and the words sort of make sense, but I think it helps to kind of draw a picture, and so I'm going to draw a little picture. Okay? >> Yeah. >> All right.



12. Okay, Michael, so let's, let's look at this little line segment, okay? And then say I've got ten little points that I could put down on this line segment, and I want them all to represent some part of this line, alright? That's kind of [UNKNOWN] -ish. So, I'm going to put a little X here, I'll put one here, I'll put one here, put one here, here, here, here, here, here. Is that ten? Three... six. Nine, ten. Ten. Okay. And let's pretend I did the right thing here and I have them kind of uniformly distributed across the line segment. So that means each one of these points is sort of owning, an equal size sub segment of this segment. Does that make sense? >> Yeah, so, so it's representing it in the sense that, that point. Uh,when you're trying to estimate values of other places on the line it's going to default as

the nearest neighbor to being that point so there's a very small little neighborhood of the red line segment that is covered by each of the green X's. >> That is exactly right and in fact each one of these green X's represents. How much of this segment? >> Each of the green X's covers one tenth? >> That's exactly right. You cover one tenth. Alright Michael, so let's say I move from a line segment now to a two dimensional space. So a little square segment. If that's the right technical term. And I've taken my little ten x's, and I put them down here before. Well, here's something you'll notice; you'll notice that each one of these x's is going to still end up representing one-tenth of all of this space, but you'll also notice that, that, that it's representing now you know. Really really really really big. >> I see. >> So one way of putting it is, you know if you think about the farthest point, as opposed to the furthest point which would be incorrect. >> [LAUGH] >> The farthest point that this particular first x over here is representing, its got some distance here. Over here, the farthest point from this x, the distance is very, very far away. So, a question would be, how can I make it so that each of the x's I put down represents the same amount of. I don't know diameter or distance as the xs in this line segment over here. So what do you think I have to do? >> I feel like you need to fill up the square with xs. >> Yeah, that's exactly right so let's do that. So filling em up Michael as you suggested. You'll notice that at least if we pretend that I drew this right. Each of these X's is now going to end up being the nearest neighbor for a little square like this, and the diameter of these little squares are going to be the same as the diameter of these little line segments. >> Yeah, I agree for some definition of the word diameter. >> Yes, and for some definition of our demonstration. Okay, so how many of these X's are there, Michael? Can you tell? You want to count? >> [LAUGH]. I'm going to multiple [CROSSTALK] cause it looks like you did ten by ten so that'll be 100. >> That'll be 100. So each one now holds a hundredth of the space, and I went from needing ten points to, of course, 100 points in order to cover the same amount of space. >> Alright, so that definitely seems like the mild rebuke of dimensionality. >> Yes. >> But doesn't seem that bad. >> Okay well, what happens if I now move into three dimensions? So now, if I want to cover the same amount of, you know, diameter space for, you know, sufficient definition of diameter. I'm going to have to do a bunch of copying and pasting that I'm not willing to do so, you know, there would be more x's here and you know, there will be x's there and an x here and it'll just kind of go and fill up some space and you know, I'm not going to do this whatever but [SOUND] and you'll get x's everywhere. And you notice, I need a lot more X's than I had before. And by the way, I'm just showing you the outside of this little cube, there are actually X's on the inside as well, that you can't see. How many X's do you think I have? >> I don't think you drew any X's. You're just like scribbling on the side of the cube. >> These are X's. >> You, you were doing so well for awhile, and then just lost it entirely. >> Well, wouldn't you lose it if you had to write 1000 x's. >> Hm. No because I would use computers to help me but yes, yes it is very frustrating to have to have that many x's. And so but in particular in this case we're talking about data points in a nearest neighbor method and that, boy that does seem like a big growth from ten to a 100 to 1000. >> In fact the growth is, exponential. >> Exponential >> >Right. So if we went into four dimensions, which I'm not going to draw, then we would need not 1,000, but 10,000 points. And if five dimensions we would need 100,000 points. And in six dimensions, we would need 1,000,000 points. And so on and so forth. So something like. Ten to the D, where D is the number of dimensions. >> Wow. >> Right. So this is really problematic right. In my little nearest neighbor method, wanted to be able to say, well, I want to make sure the neighborhood remains small, as I add dimensions, I'm going to need to grow the number of points that I have in my training set exponentially. And that's really bad. And by the way, this isn't just an issue. Of k-nn. This is true in general, don't think about this now as nearest neighbors in the sense of [INAUDIBLE], but think of it as points that are representing or covering the space. And if you want to represent the same sort of hyper-volume of space as you add dimensions, you're going to have to get exponentially more points in order to do that. And coverage is necessary to do learning. So the curse of dimensionality does not just to K-NN, it is a curse of dimensionality for ML period [SOUND]. >> You mean for me? >> Yes. Because of [INAUDIBLE] >> [LAUGH] Okay. And that seems really problematic because it's

very natural to just keep throwing dimensions into a machine learning problem. Like it's, it's having trouble learning. Let me give it a few more dimensions so, to give it hints. But really what you're doing is just giving it a larger and larger volume to fill. >> Yeah. And it can't fill it unless you give it more and more data. So you're better off giving more data than you are giving more dimensions. >> Zoinks. >> Mm-hm. There's an entire series of lessons that we will get eventually that, that deals with this issue. >> The issue of? >> Dimensionality. >> Finding the right dimensionality? >> Yeah. >> That would be a useful thing. >> It would. But it's far off in the future. It's like infinitely far in the future. So we'll worry about that in a few weeks. >> Okay. >> Okay. All right. So there you go, Michael. Curse of Dimensionality is real and it's a real problem. >> Where did that term come from, it's a cool term. I think it came from, oh what's his name. Bellman. >> Oh, Bellman, like the dynamic programming guy. >> Yeah, the dynamic programming guy, uh,the Bellman of Bellman equation guy. >> Which we haven't gotten to yet in the course. >> Which we haven't gotten to in the course but we will get to in the course. Because it's central. So it looks like actually, the element's central to a lot of things. >> Wow. >> Sometimes it gives us equations that helps us but sometimes it gives us curses. >> [LAUGH] Curses Betterman. Foiled again. >> [LAUGH]

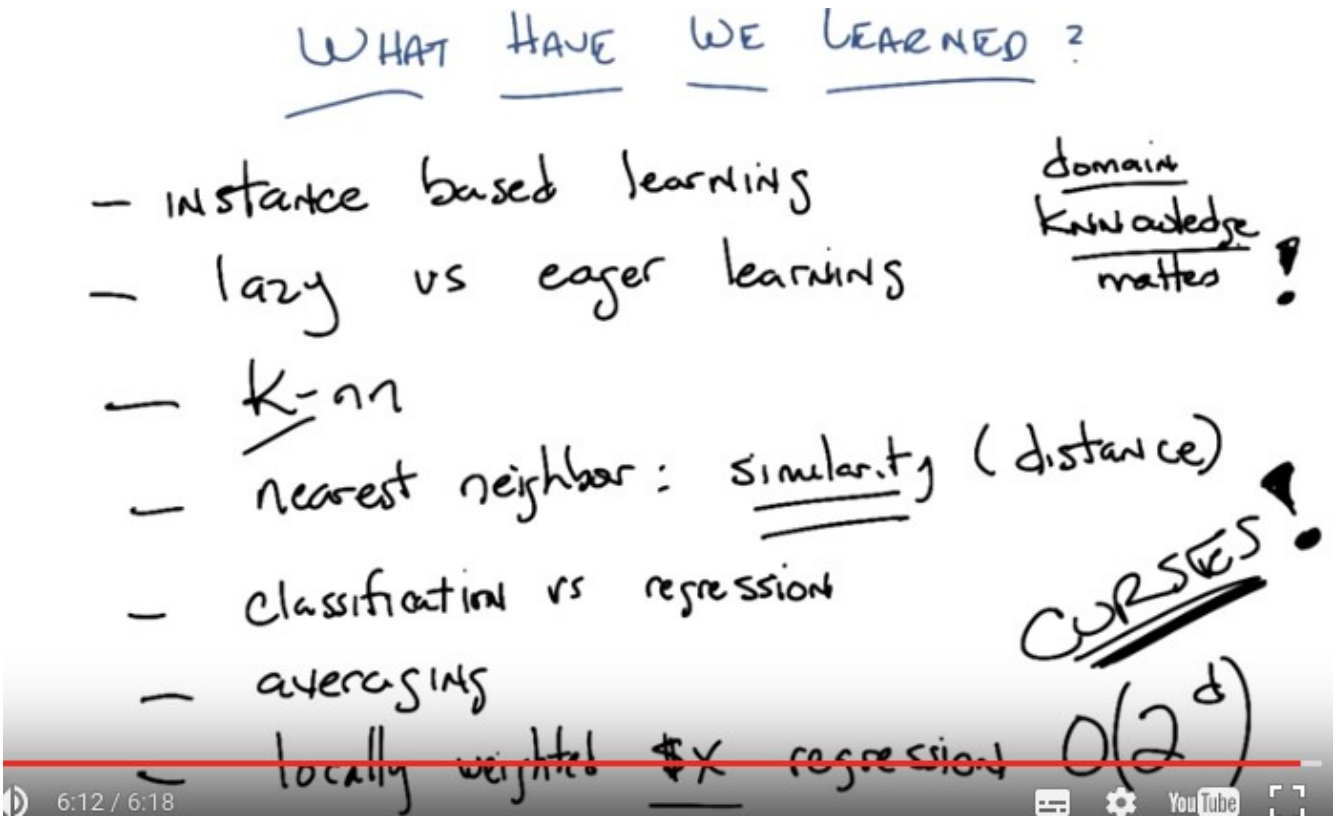


From Yutube video Data Mining with Weka (3.6: Nearest neighbor):
If training set size n → ∞ and k → ∞ and k/n → 0, error approaches minimum.

13. Okay, Michael so we talked a little bit about the curse of dimensionality, but I think it's worthwhile to talk about some other stuff that comes up. We've been sort of skirting around this and you know bring it up in various ways throughout our discussion so far. But I think it's worthwhile kind of writing them all down on a slide and trying to think through them for a little bit. So ,uh, the other stuff that comes up in K-NN mainly comes up in these sort of assumptions we make about parameters to the algorithm. So the one we talked about ,uh, probably the most is our distance measure, you know our distance between some X and some query point Q and we've explored a couple. We looked at Eucudean and we looked at Manhattan. And we even looked at weighted versions of those. And this

really matters, I've said this before but I really think it bears repeating that your choice of distance function Really matters. If you pick the wrong kind of distance function, you're just going to get very poor behavior. >> So I, so I have a question about these these distance functions. So you mentioned Euclidean and Manhattan, are there other distance functions that the students should know? Like, things that they, that might come up, or things that they should think of first if they have a particular kind of data? >> yeah, there's a, there's a ton of them. I think well, first off, it, it's probably worth pointing out that this, this notion of weighted distance is one way to deal with the curse of dimensionality. You can weight different dimensions differently. And that would be one, and you might come up with sort of automatic ways of doing that. That, that's sort of worth mentioning. But you will notice that both Euclidean and Manhattan distance at least as we have talked about them, are really useful for things like regression (相對於 classification). Their kind of assuming that you have numbers in that subtraction kind of makes sense. But there are other functions, distance functions that you might do if you are dealing with cases like, I don't know discrete data, right? Where instead of it all being numbers, it's colors, or something like that. Alright so, your distance might be mismatches. For example, or it might be a mixture of those. In fact, one of the nice things about KNN, is that we've been talking about it with points, because it's sort of easy to think about it that way. But this distance function is just a black box. You can take Arbitrary things and try to decide how similar they are based on whatever you know about the domain and that could be very useful. So ,you could talk about images right, where you take pictures of people and you know rather than doing something like a pixel by pixel comparison, you try to line up their eyes. And look at their mouths, and try to see if they're the same shape you know things like that, that might be more complicated and and perhaps even arbitrarily computational to determine notions of similarity so really this idea of distance in similarity tells you a lot about your domain and what you believe about it. Another thing that's worth what what's pushing on a little bit is how you pick k. Well there's no good way to pick k. You just, you just have to know something about it, but I want to think about a particular case. Well, what if we end up in a world where K=N? >> Well, that would be silly. >> Why would it be silly? >> Well, so if K=N, then what you're doing is you're taking, so in the case of regression for example, you're taking all of the data points and averaging the Y values together. Basically ignoring the query. So, you end up with a constant function. >> But that's only if you do a simple average. What if you do a weighted average? >> A weighted average. So the near, the points that are the query are going to get more weight in the average, so that acually will be diffrent. Even though k equals n, it will be different depending on where you actually put your query down. >> Exactly. That's exactly right so, for example, if I have a little bunch of points like this say. Where you notice it kind of looks like I have two different lines here and I can pick a query point way over here, all of these points are going to influence me as oppose to these points and so I'm going to end up. Estimating with something that looks more like this because these points over here won't have much to say. But if I have a query point that's way over here somewhere these points are going to matter and I'm going to end up looking something looks a little bit more like this than like that. Now I'm drawing these as lines. They won't exactly look like lines because these points will have some influence. They'll be more curvy than that. But the point is that near, near the place we want to do the query it will look To be more strongly influenced by these points over here or these points over here depending on where you are. >> Well that gives me an idea. >> Oh, what kind of idea does it give you? >> Well, what about instead of just taking a weighted average, what about using a distance matrix to pick up some of the points? And then do a different regression on that substantive point. >> Right, I like that. So we can replace this whole notion of average with a more kind of, regression-y thing. >> So it actually, instead of using the same value for the whole patch. Actually, it still continues to use the input values. >> Yeah. So, in fact, average is just a special case of a kind of regression, right? >> Mm hm, mm hm. >> Right? So this actually has a name, believe it or not. It's actually called locally weighted regression. Yeah, so this actually works pretty well and in place of sort of averaging function, you can do just about anything you want to. You could throw in a

decision tree, you could throw in a neural network, you could throw in lines do linear regression. You can do, almost anything that you can imagine doing. >> Neat, >> Yeah. Add that works out very well. And again, it gives you a little bit of power. So here's something I don't think is very obvious until it's pointed out to you. Which is this notion of replacing the average with a more general regression or even classification function. It actually allows you to do something more powerful than it seems. So let's imagine that we were going to do locally weighted regression and we were going to do, in fact, linear regression. So, what would locally- weighted linear regression look like? Well, if we go back to this example over here on the left basically, you take all the points that are nearby and you try to fit a line to it. And, so you would end up with stuff that looked, pretty much, like this. While you're over here, you would get the line like this, but while you were over here you'd get a line like this. Then, somewhere in the middle you would get lines that started to look like this and And you would end up with something that kind of ended up looking a lot like a curve. So that's kind of cool because you notice that we start with a hypothesis state of lines and this locally weighted linear regression. But then we end up actually being able to represent a hypothesis space that is strictly bigger. Then the set of lines. Hm. So we can use a very simple kind of hypothesis space but by using this locally weighted regression we end up with a more complicated space that is complicated, that's made more complicated depending upon the complications that are represented by your data points. So this results, this sort of reveals another bit of power with K-NN. Which is, it allows you to take local information and build functions or build concepts around the local things that are similar to you. And that allows you to make arbitrarily complicated functions. >> Neat. >> At least in principle. Okay, cool. Alright so you got all that? >> I, think so yeah. >> Okay, cool. So then, I think we should wrap up. >> Nice. >> Nice.



14. Okay Michael so I think with that little bit of discussion, I feel like we're done. >> Cool! Alright. Well it's been nice talking to you. I hope the course went well and, oh you mean just for this lesson? >> Yeah just for this lesson. >> Alright. >> So let's. >> Well let's wrap up this lesson then. >> Yeah let's see what have we learned? So remind me Michael what have we learned? >> Well we were

talking about instance based learning. >> That's true. That's the first thing we learned. You will notice by the way, I never actually told you why it was called instance based learning. >> Charles, why is it called instance based learning? >> I don't know but I am willing to guess that it has to do with a fact that we look at the exact instances that we have and we base our learning on that. >> Alright and we brought it in by starting off thinking eager and lazy learning. >> Right. What is the difference, Michael? >> I will tell you when I need to tell you. >> [Laugh] That's exactly right. >> So lazy learning is about putting off the work until it's actually needed. Eager is about, as soon as the problem is posed, solve it. And then, you know, if you're lucky, the answer will eventually come in handy. >> Exactly right. Okay what else? >> So as a concret example of a lazy learner we talked about K's nearest neighbor or K-nn. >> K-nn. And this whole notion of nearest neighbor. Is in fact one way of talking about similarity functions. >> Right and the similarity functions play a really central role in all this. >> Right. Similarity. We talk about it as if they were distance functions. But distance is just another way of talking about, a similarity. So, this is actually keeping. K-nn was a specific, algorithm we used, and we talked about various versions of it, and, the nearest neighbor part really got us to think a little bit about similarity and distance and, and what all that means. A really important thing here is, I think, that similarity is just another way of capturing the main knowledge. And K, in K-nn is another way of capturing the main knowledge. And that if we saw through the quizes and some of our discussions, that this is actually very, very important. That the main knowledge matters. Now, we also talked about KNN in the context of both regression and classification. I see what you did there, 'Knnowledge', it's got KNN in it. >> >Okay, so yeah, classification and regression are different things. But, KNN can handle both of them. And, at the end of the day, that's all stuck in our notion of similarity, and our notion of averaging. Which we kind of took as an overall term, for a bunch of different things you might do, which some people find confusing. >> [LAUGH]. >> But I think that other people would really kind of understand what we mean. >> I'm very tempted to take that out of Wikipedia, but that would, that would just be rude. >> It would be rude. Anything else? We learned one big thing. >> We looked at how to compose different learning algorithms together. For example in the context of locally weighted linear regression. >> Mm hm. >> We used this instance based idea along with linear regression to get something that was both locally smooth but globally bumpy. >> Right. So, I'm going to just say locally weighted regression. Where we can do any kind of regression we might want to. I was going to call that $X. So, stick in your favorite value. >> Let's see, what else? Oh, oh, a really big thing was Bellman's curse of dimensionality. >> Yes. >> And the idea there was that the more features that you include The more data you need to fill up that space. >> Yep, It's exponential. >> And in fact I even just I decided to go and play with this a little bit so that example that you were doing before where the y equals x1 squared plus x2 >> Mm-hm. >> When I gave it well as we saw in the example we gave it like ten or 12 examples and it did really badly. So it continued to do somewhat badly until I got to about 100,000 and then it was actually doing [LAUGH] really well. >> Hmm. >> But that seems like an awful lot of examples for what is otherwise a very simple problem. >> Right. Well if you think about it, the amount of data you have to see to determine the relative. Relevance of the two different dimensions is quite a bit in that particular kind of function. >> Hm. >> Yeah. That's a lot of space to cover. All possible real values [LAUGH] across a potentially infinite space. >> Yeah, I guess that's true. >> Yeah, so the cursor dimensionality is real and we just sort of can't get around it. Although As I mentioned earlier we will see in the second part of the course ways that people try to get around recursive dimensionality. >> Ahah! >> Mm-hm. Okay. >> Or at least blunt it. Right? >> Yeah or at least blunt it because you can't actually get around recursive dimensionality, you can only deal with it. >> There is no free lunch. >> There is no free lunch. In fact, that's a theorem. >> [LAUGH] >> [INAUDIBLE] a theorem? >> Yeah, I think so. >> What's the theorem? >> No free lunch. That any learning algorithm that you create is going to have the property that if you average over all possible instances, it's not doing any different than random. >> Right. And, and another way of thinking about that, a practical way of thinking about that is; if I don't know

anything about the data that I'm going to have to learn over, then it doesn't really matter what I do because there's all possible kind of data sets. However, if I have domain knowledge, I can use that to choose the best learning algorithm for the problems that I'm going to encounter. >> So does that mean that, that all of machine learning really comes down to, you have to already know what you need to solve the problem to apply these techniques to solve the problem? >> No, but you have to know a little bit about your problem in order to decide what to do. And in fact, you could make the argument that this entire class is about exposing the students to a wide range of techniques and giving them enough practice so that they can do a pretty good job of telling, given a problem. Would it be better to use this kind of technique or this kind of technique? Is K-nn a sort of better way of approaching it? Decision tree a better way of approaching it? Um,It's a lot of what this class is about is helping them to get enough domain knowledge or enough knowledge anyways so that they can apply it to particular domains. >> Cool, so alright, that seems like a plenty useful lesson. >> Yes, it's a very hopeful note to end on, so let's end on that. >> Alright, see you next time. >> Alright, bye Michael.