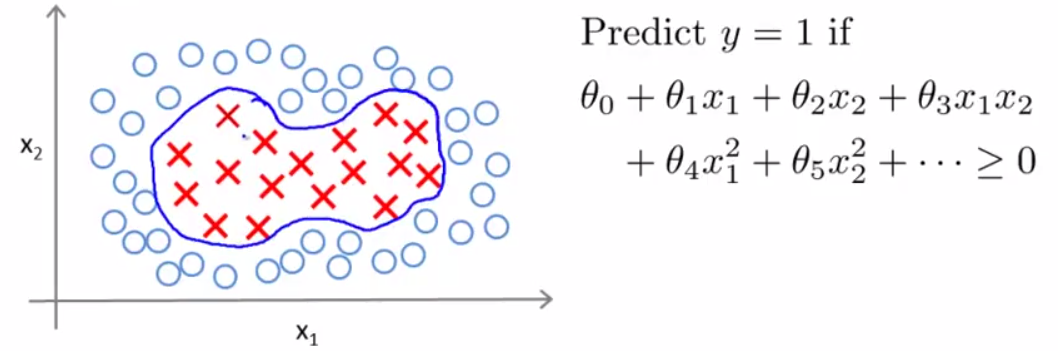
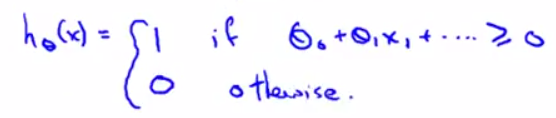
***Kernels***

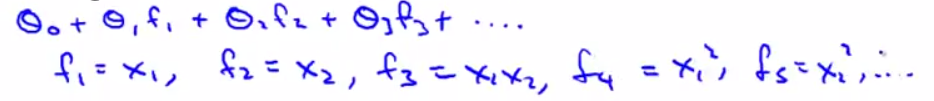
* Now we’d like to start adapting SVMs in order to develop complex nonlinear classifiers via the main technique for doing this, **kernels**.
* If you have a training set that looks like below+ you want to find a nonlinear decision boundary to distinguish positive + negative examples, 1 way to do so is via a set of complex polynomial features:



* You end up w/ an hӨ(x) that predicts 1 if all those polynomial features > 0 + predict 0 otherwise



* Another way of writing this w/ “f” instead of “x” for the new features (f4 = x1^2. f5 = x2^2, etc.)



* The question is, is there a different/better choice of features than high-order polynomials
* Using high order polynomials can become very computationally expensive
* New idea for how to define new features f1, f2, f3 (for real problems we can define a much larger number of features, n):

Ignoring x0 out of this (intercept), 2 manually pick a few points, + then call these points l1, we are going to pick a different point, let's call that l2 + let's pick the third one + call this one l3, + for now let's just say that I'm going to choose these three points manually. I'm going to call these three points line ups, so line up one, two, three. What I'm going to do is define my new features as follows, given an example X, let me define my first feature f1 to be some measure of the similarity between my training example X + my first landmark + this specific formula that I'm going to use to measure similarity is going to be this is E to the minus the length of X minus l1, squared, divided by two sigma squared.

* 3:40
* So, depending on whether or not you watched the previous optional video, this notation, you know, this is the length of the vector W. + so, this thing here, this X minus l1, this is actually just the euclidean distance
* 3:58
* squared, is the euclidean distance between the point x + the landmark l1. We will see more about this later.
* 4:06
* But that's my first feature, + my second feature f2 is going to be, you know, similarity function that measures how similar X is to l2 + the game is going to be defined as the following function.
* 4:25
* This is E to the minus of the square of the euclidean distance between X + the second landmark, that is what the enumerator is + then divided by 2 sigma squared + similarly f3 is, you know, similarity between X + l3, which is equal to, again, similar formula.
* 4:46
* + what this similarity function is, the mathematical term for this, is that this is going to be a kernel function. + the specific kernel I'm using here, this is actually called a Gaussian kernel.
* 5:00
* + so this formula, this particular choice of similarity function is called a Gaussian kernel. But the way the terminology goes is that, you know, in the abstract these different similarity functions are called kernels + we can have different similarity functions
* 5:13
* + the specific example I'm giving here is called the Gaussian kernel. We'll see other examples of other kernels. But for now just think of these as similarity functions.
* 5:22
* + so, instead of writing similarity between X + l, sometimes we also write this a kernel denoted you know, lower case k between x + one of my landmarks all right.
* 5:34
* So let's see what a criminals actually do + why these sorts of similarity functions, why these expressions might make sense.
* 5:46
* So let's take my first landmark. My landmark l1, which is one of those points I chose on my figure just now.
* 5:53
* So the similarity of the kernel between x + l1 is given by this expression.
* 5:57
* Just to make sure, you know, we are on the same page about what the numerator term is, the numerator can also be written as a sum from J equals 1 through N on sort of the distance. So this is the component wise distance between the vector X + the vector l. + again for the purpose of these slides I'm ignoring X0. So just ignoring the intercept term X0, which is always equal to 1.
* 6:21
* So, you know, this is how you compute the kernel w/ similarity between X + a landmark.
* 6:27
* So let's see what this function does. Suppose X is close to one of the landmarks.
* 6:33
* Then this euclidean distance formula + the numerator will be close to 0, right. So, that is this term here, the distance was great, the distance using X + 0 will be close to zero, + so
* 6:46
* f1, this is a simple feature, will be approximately E to the minus 0 + then the numerator squared over 2 is equal to squared
* 6:55
* so that E to the 0, E to minus 0, E to 0 is going to be close to one.
* 7:01
* + I'll put the approximation symbol here b/c the distance may not be exactly 0, but if X is closer to landmark this term will be close to 0 + so f1 would be close 1.
* 7:13
* Conversely, if X is far from 01 then this first feature f1 will be E to the minus of some large number squared, divided divided by two sigma squared + E to the minus of a large number is going to be close to 0.
* 7:33
* So what these features do is they measure how similar X is from one of your landmarks + the feature f is going to be close to one when X is close to your landmark + is going to be 0 or close to zero when X is far from your landmark. Each of these landmarks. On the previous line, I drew three landmarks, l1, l2,l3.
* 7:56
* Each of these landmarks, defines a new feature f1, f2 + f3. That is, given the the training example X, we can now compute three new features: f1, f2, + f3, given, you know, the three landmarks that I wrote just now. But first, let's look at this exponentiation function, let's look at this similarity function + plot in some figures + just, you know, understand better what this really looks like.
* 8:23
* For this example, let's say I have two features X1 + X2. + let's say my first landmark, l1 is at a location, 3 5. So
* 8:33
* + let's say I set sigma squared equals one for now. If I plot what this feature looks like, what I get is this figure. So the vertical axis, the height of the surface is the value
* 8:45
* of f1 + down here on the horizontal axis are, if I have some training example, + there
* 8:51
* is x1 + there is x2. Given a certain training example, the training example here which shows the value of x1 + x2 at a height above the surface, shows the corresponding value of f1 + down below this is the same figure I had showed, using a quantifiable plot, w/ x1 on horizontal axis, x2 on horizontal axis + so, this figure on the bottom is just a contour plot of the 3D surface.
* 9:16
* You notice that when X is equal to 3 5 exactly, then we the f1 takes on the value 1, b/c that's at the maximum + X moves away as X goes further away then this feature takes on values
* 9:36
* that are close to 0.
* 9:38
* + so, this is really a feature, f1 measures, you know, how close X is to the first landmark + if varies between 0 + one depending on how close X is to the first landmark l1.
* 9:52
* Now the other was due on this slide is show the effects of varying this parameter sigma squared. So, sigma squared is the parameter of the Gaussian kernel + as you vary it, you get slightly different effects.
* 10:05
* Let's set sigma squared to be equal to 0.5 + see what we get. We set sigma square to 0.5, what you find is that the kernel looks similar, except for the width of the bump becomes narrower. The contours shrink a bit too. So if sigma squared equals to 0.5 then as you start from X equals 3 5 + as you move away,
* 10:24
* then the feature f1 falls to zero much more rapidly + conversely,
* 10:32
* if you has increase since where three in that case + as I move away from, you know l. So this point here is really l, right, that's l1 is at location 3 5, right. So it's shown up here.
* 10:48
* + if sigma squared is large, then as you move away from l1, the value of the feature falls away much more slowly.
* 11:03
* So, given this definition of the features, let's see what source of hypothesis we can learn.
* 11:09
* Given the training example X, we are going to compute these features
* 11:14
* f1, f2, f3 + a
* 11:17
* hypothesis is going to predict one when Ө 0 plus Ө 1 f1 plus Ө 2 f2, + so on is greater than or equal to 0. For this particular example, let's say that I've already found a learning algorithm + let's say that, you know, somehow I ended up w/ these values of the parameter. So if Ө 0 equals minus 0.5, Ө 1 equals 1, Ө 2 equals 1, + Ө 3 equals 0 + what I want to do is consider what happens if we have a training example that takes
* 11:49
* has location at this magenta dot, right where I just drew this dot over here. So let's say I have a training example X, what would my hypothesis predict? Well, If I look at this formula.
* 12:04
* B/c my training example X is close to l1, we have that f1 is going to be close to 1 the b/c my training example X is far from l2 + l3 I have that, you know, f2 would be close to 0 + f3 will be close to 0.
* 12:21
* So, if I look at that formula, I have Ө 0 plus Ө 1 times 1 plus Ө 2 times some value. Not exactly 0, but let's say close to 0. Then plus Ө 3 times something close to 0.
* 12:37
* + this is going to be equal to plugging in these values now.
* 12:41
* So, that gives minus 0.5 plus 1 times 1 which is 1, + so on. Which is equal to 0.5 which is greater than or equal to 0. So, at this point, we're going to predict Y equals 1, b/c that's greater than or equal to zero.
* 12:58
* Now let's take a different point. Now lets' say I take a different point, I'm going to draw this one in a different color, in cyan say, for a point out there, if that were my training example X, then if you make a similar computation, you find that f1, f2,
* 13:15
* Ff3 are all going to be close to 0.
* 13:18
* + so, we have Ө 0 plus Ө 1, f1, plus so on + this will be about equal to minus 0.5, b/c Ө 0 is minus 0.5 + f1, f2, f3 are all zero. So this will be minus 0.5, this is less than zero. + so, at this point out there, we're going to predict Y equals zero.
* 13:44
* + if you do this yourself for a range of different points, be sure to convince yourself that if you have a training example that's close to L2, say, then at this point we'll also predict Y equals one.
* 13:56
* + in fact, what you end up doing is, you know, if you look around this boundary, this space, what we'll find is that for points near l1 + l2 we end up predicting positive. + for points far away from l1 + l2, that's for points far away from these two landmarks, we end up predicting that the class is equal to 0. As so, what we end up doing,is that the decision boundary of this hypothesis would end up looking something like this where inside this red decision boundary would predict Y equals 1 + outside we predict
* 14:32
* Y equals 0. + so this is how w/ this definition of the landmarks + of the kernel function. We can learn pretty complex non-linear decision boundary, like what I just drew where we predict positive when we're close to either one of the two landmarks. + we predict negative when we're very far away from any of the landmarks. + so this is part of the idea of kernels of + how we use them w/ the SVM, which is that we define these extra features using landmarks + similarity functions to learn more complex nonlinear classifiers.
* 15:08
* So hopefully that gives you a sense of the idea of kernels + how we could use it to define new features for the SVM.
* 15:15
* But there are a couple of questions that we haven't answered yet. One is, how do we get these landmarks? How do we choose these landmarks? + another is, what other similarity functions, if any, can we use other than the one we talked about, which is called the Gaussian kernel. In the next video we give answers to these questions + put everything together to show how SVMs w/ kernels can be a powerful way to learn complex nonlinear functions.