MachineLearning-Lecture07

Instructor (Andrew Ng):Good morning. So just a few quick announcements. One is for the SCPD students – that would be for the students taking the classes remotely – to turn in the process solutions due this Wednesday, please fax the solutions to us at – so the fax number written at the top of the Problem Set 1. And in particular, please do not use the SCPD [inaudible], since that usually takes me longer for your solutions to get to us.

And everyone else, if you're not an SCPD student, then please only turn in hard copies – the physical paper copies of your solutions from Set 1. And please don't email them to us or send them by fax unless you're an SCPD student.

Also, as you know, project proposals are due this Friday. And so in my last few office hours, you should have lively discussions with people about [inaudible] ideas.

This Wednesday, immediately after class, so this Wednesday, starting about, I guess 10:45, right after class, I'll be holding extra office hours in case any of you want to discuss project ideas some more before the proposals are due on Friday, okay?

But is this loud enough? Can people in the back hear me? Is this okay?

Student:[Inaudible] louder, maybe?

Instructor (Andrew Ng):[Inaudible] turn up the volume, [Inaudible]? Is this okay? Testing, testing. It's better? Okay, great.

So that was it for the administrative announcements.

So welcome back. And what I wanna do today is continue our discussion on support vector machines. And in particular, I wanna talk about the optimal margin classifier. Then I wanna take a brief digression and talk about primal and duo optimization problems, and in particular, what's called the KKT conditions. And then we'll derive the duo to the optimization problem that I had posed earlier.

And that will lead us into a discussion of kernels, which I won't really – which we just get to say couple words about, but which I'll do probably only in the next lecture.

And as part of today's lecture, I'll spend some time talking about optimization problems. And in the little time I have today, I won't really be able to do this topic justice. I wanna talk about convex optimization and do that topic justice. And so at this week's discussion session, the TAs will have more time – will teach a discussion session – focus on convex optimization – sort of very beautiful and useful theory. So you want to learn more about that, listen to this Friday's discussion session.

Just to recap what we did in the previous lecture, as we were beginning on developing on support vector machines, I said that a hypothesis represented as H sub [inaudible] wb as g of w transpose [inaudible] x + b, where g will be +1 or -1, depending on whether z is greater than 0. And I said that in our development of support vector machines, we'll use – we'll change the convention of letting y be +1, -1 to note the class labels.

So last time, we also talked about the functional margin, which was this thing, gamma hat i. And so we had the intuition that the if functional margin is a large positive number, then that means that we are classifying a training example correctly and very confidently.

So yi is +1. We would like w transpose xi + b to be very large. And it makes i – if, excuse me, if yi is -1, then we'd w transpose xi + b to be a large negative number. So we'd sort of like functional margins to be large.

We also said – functional margin is a strange property – that you can increase functional margin just by, say, taking your parameters, w and b, and multiplying them by 2.

And then we also defined the geometric margin, which was that we just – essentially, the functional margin divided by the normal w.

And so the geometric margin had the interpretation as being – I'll give you a few examples. The geometric margin, for example, is – has the interpretation as a distance between a training example and a hyperplane.

And it'll actually be a sin distance, so that this distance will be positive if you're classifying the example correctly. And if you misclassify the example, this distance – it'll be the minus of the distance, reaching the point, reaching the training example. And you're separating hyperplane. Where you're separating hyperplane is defined by the equation w transpose x + b = 0.

So – oh, well, and I guess also defined these things as the functional margin, geometric margins, respect to training set I defined as the worst case or the minimum functional geometric margin.

So in our development of the optimal margin classifier, our learning algorithm would choose parameters w and b so as to maximize the geometric margin. So our goal is to find the separating hyperplane that separates the positive and negative examples with as large a distance as possible between hyperplane and the positive and negative examples.

And if you go to choose parameters w and b to maximize this, [inaudible] one copy of the geometric margin is that you can actually scale w and b arbitrarily. So you look at this definition for the geometric margin. I can choose to multiply my parameters w and b by 2 or by 10 or any other constant. And it doesn't change my geometric margin.

And one way of interpreting that is you're looking at just separating hyperplane. You look at this line you're separating by positive and negative training examples. If I scale w and b, that doesn't change the position of this plane, though because the equation wh + b = 0 is the same as equation 2 w transpose x + 2b = 0. So it use the same straight line.

And what that means is that I can actually choose whatever scaling for w and b is convenient for me. And in particular, we use in a minute, I can [inaudible] perfect constraint like that the normal w [inaudible] 1 because this means that you can find a solution to w and b. And then by rescaling the parameters, you can easily meet this condition, this rescaled w [inaudible] 1. And so I can add the condition like this and then essentially not change the problem.

Or I can add other conditions. I can actually add a condition that – excuse me, the absolute value of w1 = 1. I can have only one of these conditions right now [inaudible]. And adding condition to the absolute value – the first component of w must be to 1. And again, you can find the absolute solution and just rescale w and meet this condition.

And it can have other, most esoteric conditions like that because again, this is a condition that you can solve for the optimal margin, and then just by scaling, you have w up and down. You can – you can then ensure you meet this condition as well.

So again, [inaudible] one of these conditions right now, not all of them. And so our ability to choose any scaling condition on w that's convenient to us will be useful again in a second.

All right. So let's go ahead and break down the optimization problem. And again, my goal is to choose parameters w and b so as to maximize the geometric margin.

Here's my first attempt at writing down the optimization problem. Actually wrote this one down right at the end of the previous lecture. Begin to solve the parameters gamma w and b such that – that [inaudible] i for in training examples.

Let's say I choose to add this normalization condition. So the norm condition that w – the normal w is equal to 1 just makes the geometric and the functional margin the same. And so I'm saying I want to find a value – I want to find a value for gamma as big as possible so that all of my training examples have functional margin greater than or equals gamma, and with the constraint that normal w equals 1, functional margin and geometric margin are the same. So it's the same. Find the value for gamma so that all the values – all the geometric margins are greater or equal to gamma.

So you solve this optimization problem, then you have derived the optimal margin classifier – that there's not a very nice optimization problem because this is a nasty, nonconvex constraints. And [inaudible] is asking that you solve for parameters w that lie on the surface of a unisphere, lie on his [inaudible]. It lies on a unicircle – a unisphere.

And so if we can come up with a convex optimization problem, then we'd be guaranteed that our [inaudible] descend to other local [inaudible] will not have local optimal. And it turns out this is an example of a nonconvex constraint. This is a nasty constraint that I would like to get rid of.

So let's change the optimization problem one more time. Now, let me pose a slightly different optimization problem. Let me maximize the functional margin divided by the normal w subject to yi w transpose xi.

So in other words, once you find a number, gamma hat, so that every one of my training examples has functional margin greater than the gamma hat, and my optimization objective is I want to maximize gamma hat divided by the normal w. And so I wanna maximize the function margin divided by the normal w.

And we saw previously the function margin divided by the normal w is just a geometric margin, and so this is a different way of posing the same optimization problem.

[Inaudible] confused, though. Are there questions about this?

Student:[Inaudible] the second statement has to be made of the functional margin y divided by – why don't you just have it the geometric margin? Why do you [inaudible]?

Instructor (Andrew Ng):[Inaudible] say it again?

Student: For the second statement, where we're saying the data of the functional margin is divided [inaudible].

Instructor (Andrew Ng):Oh, I see, yes.

Student:[Inaudible] is that [inaudible]?

Instructor (Andrew Ng):So let's see, this is the function margin, right? This is not the geometric margin.

Student: Yeah.

Instructor (Andrew Ng):So – oh, I want to divide by the normal w of my optimization objective.

Student:I'm just wondering how come you end up dividing also under the second stage [inaudible] the functional margin. Why are you dividing there by the normal w?

Instructor (Andrew Ng):Let's see. I'm not sure I get the question. Let me try saying this again. So here's my goal. My – I want [inaudible]. So let's see, the parameters of this optimization problem where gamma hat w and b – so the convex optimization software solves this problem for some set of parameters gamma w and b. And I'm imposing the constraint that whatever values it comes up with, yi x [inaudible] x5 + b must be greater than gamma hat. And so this means that the functional margin of every example had better be greater than equal to gamma hat. So there's a constraint to the function margin and a constraint to the gamma hat.

But what I care about is not really maximizing the functional margin. What I really care about – in other words, in optimization objective, is maximizing gamma hat divided by the normal w, which is the geometric margin.

So in other words, my optimization [inaudible] is I want to maximize the function margin divided by the normal w. Subject to that, every example must have function margin and at least gamma hat. Does that make sense now?

Student:[Inaudible] when you said that to maximize gamma or gamma hat, respect to gamma w and with respect to gamma hat so that [inaudible] gamma hat are no longer [inaudible]?

Instructor (Andrew Ng):So this is the – so it turns out – so this is how I write down the – this is how I write down an optimization problem in order to solve for the geometric margin. What is it – so it turns out that the question of this – is the gamma hat the function of w and b?

And it turns out that in my previous mathematical definition, it was, but the way I'm going to pose this as an optimization problem is I'm going to ask the convex optimization solvers – and this [inaudible] software – unless you have software for solving convex optimization problems – hen I'm going to pretend that these are independent variables and ask my convex optimization software to find me values for gamma, w, and b, to make this value as big as possible and subject to this constraint.

And it'll turn out that when it does that, it will choose – or obviously, it will choose for gamma to be as big as possible because optimization objective is this: You're trying to maximize gamma hat.

So for x value of w and b, my software, which choose to make gamma hat as big as possible – well, but how big can we make gamma hat? Well, it's limited by use constraints. It says that every training example must have function margin greater than equal to gamma hat. And so my – the bigger you can make gamma hat will be the value of the smallest functional margin. And so when you solve this optimization problem, the value of gamma hat you get out will be, indeed, the minimum of the functional margins of your training set.

Okay, so Justin?

Student: Yeah, I was just wondering, I guess I'm a little confused because it's like, okay, you have two class of data. And you can say, "Okay, please draw me a line such that you maximize the distance between – the smallest distance that [inaudible] between the line and the data points." And it seems like that's kind of what we're doing, but it's – it seems like this is more complicated than that. And I guess I'm wondering what is the difference.

Instructor (Andrew Ng):I see. So I mean, this is – the question is [inaudible]. Two class of data – trying to find separate hyperplane. And this seems more complicated than trying to find a line [inaudible]. So I'm just repeating the questions in case – since I'm not sure how all the audio catches it.

So the answer is this is actually exactly that problem. This is exactly that problem of given the two class of data, positive and negative examples, this is exactly the formalization of the problem where I go is to find a line that separates the two – the positive and negative examples, maximizing the worst-case distance between the [inaudible] point and this line.

Okay? Yeah, [Inaudible]?

Student: So why do you care about the worst-case distance [inaudible]?

Instructor (Andrew Ng):Yeah, let me – for now, why do we care about the worst-case distance? For now, let's just say – let's just care about the worst-case distance for now. We'll come back, and we'll fix that later. We'll – that's a – caring about the worst case is is just – is just a nice way to formulate this optimization problem. I'll come back, and I'll change that later.

Okay, raise your hand if this makes sense – if this formulation makes sense? Okay, yeah, cool.

Great. So let's see – so this is just a different way of posing the same optimization problem. And on the one hand, I've got to get rid of this nasty, nonconvex constraint, while on the other hand, I've now added a nasty, nonconvex objective. In particular, this is not a convex function in parameters w.

And so you can't – you don't have the usual guarantees like if you [inaudible] global minimum. At least that does not follow immediately from this because this is nonconvex.

So what I'm going to do is, earlier, I said that can pose any of a number of even fairly bizarre scaling constraints on w. So you can choose any scaling constraint like this, and things are still fine. And so here's the scaling I'm going to choose to add. Again, I'm gonna assume for the purposes of today's lecture, I'm gonna assume that these examples are linearly separable, that you can actually separate the positive and negative classes, and that we'll come back and fix this later as well.

But here's the scaling constraint I want to impose on w. I want to impose a constraint that the functional margin is equal to 1. And another way of writing that is that I want to impose a constraint that min over i, yi – that in the worst case, function y is over 1.

And clearly, this is a scaling constraint because if you solve for w and b, and you find that your worst-case function margin is actually 10 or whatever, then by dividing through w and b by a factor of 10, I can get my functional margin to be over 1. So this is a scaling constraint [inaudible] would imply. And this is just more compactly written as follows. This is imposing a constraint that the functional margin be equal to 1.

And so if we just take what I wrote down as No. 2 of our previous optimization problem and add the scaling constraint, we then get the following optimization problem: min over wb.

I guess previously, we had a maximization over gamma hats divided by the normal w. So those maximize 1 over the normal w, but so that's the same as minimizing the normal w squared. It was great. Maximum normal w is min w – normal w squared. And then these are our constraints. Since I've added the constraint, the functional margin is over 1.

And this is actually my final – well, final formulation of the optimal margin classifier problem, at least for now.

So the picture to keep in mind for this, I guess, is that our optimization objective is once you minimize the normal w. And so our optimization objective is just the [inaudible] quadratic function. And [inaudible] those pictures [inaudible] can draw it. So it – if [inaudible] is w1 and w2, and you want to minimize the quadratic function like this – so quadratic function just has [inaudible] that look like this.

And moreover, you have a number of linear constraints in your parameters, so you may have linear constraints that eliminates that half space or linear constraint eliminates that half space [inaudible]. So there's that half space and so on.

And so the picture is you have a quadratic function, and you're ruling out various half spaces where each of these linear constraints. And I hope – if you can picture this in 3D, I guess [inaudible] kinda draw our own 3D, hope you can convince yourself that this is a convex problem that has no local optimum. But they be run great [inaudible] within this set of points that hasn't ruled out, then you convert to the global optimum.

And so that's the convex optimization problem. The – does this [inaudible] nice and [inaudible]. Questions about this?

Actually, just raise your hand if this makes sense. Okay, cool.

So this gives you the optimal margin classifier algorithm. And it turns out that this is the convex optimization problem, so you can actually take this formulation of the problem and throw it at off-the-shelf software – what's called a QP or quadratic program software.

This [inaudible] optimization is called a quadratic program, where the quadratic convex objective function and [inaudible] constraints – so you can actually download software to solve these optimization problems for you. Usually, as you wanna use the – use [inaudible] because you have constraints like these, although you could actually modify [inaudible] work with this, too.

So we could just declare success and say that we're done with this formulation of the problem. But what I'm going to do now is take a digression to talk about primal and duo optimization problems. And in particular, I'm going to – later, I'm going to come back and derive yet another very different form of this optimization problem. And the reason we'll do that is because it turns out this optimization problem has certain properties that make it amenable to very efficient algorithms.

And moreover, I'll be deriving what's called the duo formulation of this that allows us to apply the optimal margin classifier even in very high-dimensional feature spaces – even in sometimes infinite dimensional feature spaces.

So we can come back to that later. But let me know, since I'm talking about convex optimization. So how many here is – how many of you, from, I don't know, calculus, remember the method of Lagrange multipliers for solving an optimization problem like minimum – minimization, maximization problem subject to some constraint? How many of you remember the method of Lagrange multipliers for that?

Oh, okay, cool. Some of you, yeah. So if you don't remember, don't worry. I – I'll describe that briefly here as well, but what I'm really gonna do is talk about the generalization of this method of Lagrange multipliers that you may or may not have seen in some calculus classes. But if you haven't seen it before, don't worry about it.

So the method of Lagrange multipliers is – was – well, suppose there's some function you want to minimize, or minimize f of w. We're subject to some set of constraints that each i of w must equal 0 – for i = 1 [inaudible] l.

And given this constraint, I'll actually usually write it in vectorial form in which I write h of w as this vector value function. So that is equal to 0, where 0 is the arrow on top. I used that to denote the vector of all 0s.

So you want to solve this optimization problem. Some of you have seen method of Lagrange multipliers where you construct this [inaudible] Lagrange, which is the original optimization objective plus some [inaudible] Lagrange multipliers the highest constraints.

And these parameters – they derive – we call the Lagrange multipliers. And so the way you actually solve the optimization problem is you take the partial derivative of this with respect to the original parameters and set that to 0. So the partial derivative with respect to your Lagrange multipliers [inaudible], and set that to 0. And then the same as theorem through [inaudible], I guess [inaudible] Lagrange was that for w – for some value w star to get a solution, it is necessary that – can this be the star?

Student: Right.

Instructor (Andrew Ng):The backwards e – there exists. So there exists beta star such that those partial derivatives are equal to 0. So the method of Lagrange multipliers is to solve this problem, you construct a Lagrange, take the derivative with respect to the original parameters b, the original parameters w, and with respect to the Lagrange multipliers beta. Set the partial derivatives equal to 0, and solve for our solutions. And then you check each of the solutions to see if it is indeed a minimum.

Great. So great – so what I'm going to do is actually write down the generalization of this. And if you haven't seen that before, don't worry about it. This is [inaudible].

So what I'm going to do is actually write down the generalization of this to solve a slightly more difficult type of constraint optimization problem, which is suppose you want to minimize f of w subject to the constraint that gi of w, excuse me, is less than equal to 0, and that hi of w is equal to 0.

And again, using my vector notation, I'll write this as g of w is equal to 0. And h of w is equal to 0. So in [inaudible]'s case, we now have inequality for constraint as well as equality constraint.

I then have a Lagrange, or it's actually still – called – say generalized Lagrange, which is now a function of my original optimization for parameters w, as well as two sets of Lagrange multipliers, alpha and beta. And so this will be f of w.

Now, here's a cool part. I'm going to define theta subscript p of w to be equal to max of alpha beta subject to the constraints that the alphas are, beta equal to 0 of the Lagrange.

And so I want you to consider the optimization problem min over w of max over alpha beta, such that the alpha is a greater than 0 of the Lagrange. And that's just equal to min over w, theta p of w.

And just to give us a name, the [inaudible] – the subscript p here is a sense of primal problem. And that refers to this entire thing. This optimization problem that written down is called a primal problem. This means there's the original optimization problem in which [inaudible] solving. And later on, I'll derive in another version of this, but that's what p stands for. It's a – this is a primal problem.

Now, I want you to look at – consider theta over p again. And in particular, I wanna consider the problem of what happens if you minimize w – minimize as a function of w this quantity theta over p. So let's look at what theta p of w is. Notice that if gi of w is greater than 0, so let's pick the value of w. And let's ask what is the state of p of w? So if w violates one of your primal problems constraints, then state of p of w would be infinity. Why is that?

[Inaudible] p [inaudible] second. Suppose I pick a value of w that violates one of these constraints. So gi of w is positive. Then – well, theta p is this – maximize this function of alpha and beta – the Lagrange. So one of these gi of w's is this positive, then by setting the other responding alpha i to plus infinity, I can make this arbitrarily large. And so if w violates one of my primal problem's constraints in one of the gis, then max over alpha of this Lagrange will be plus infinity.

There's some of – and in the same way – I guess in a similar way, if hi of w is not equal to 0, then theta p of w also be infinity for a very similar reason because if hi of w is not equal to 0 for some value of i, then in my Lagrange, I had a beta i x hi theorem. And so by setting beta i to be plus infinity or minus infinity depending on the sign of hi, I can make this plus infinity as well.

And otherwise, theta p of w is just equal to f of w. Turns out if I had a value of w that satisfies all of the gi and the hi constraints, then we maximize in terms of alpha and beta – all the Lagrange multiply theorems will actually be obtained by setting all the Lagrange multiply theorems to be 0, and so theta p just left with f of w.

Thus, theta p of w is equal to f of w if constraints are satisfied [inaudible] the gi in hi constraints, and is equal to plus infinity otherwise.

So the problem I wrote down that minimizes the function of w – theta p of w – this is [inaudible] problem. That's just exactly the same problem as my original primal problem because if you choose a value of w that violates the constraints, you get infinity. And if you satisfy the constraints, you get f of w. So this is really just the same as – well, we'll say, "Satisfy the constraints, and minimize f of w." That's really what minimizing the state of p of w is.

Raise your hand if this makes sense. Yeah, okay, cool. So all right. I hope no one's getting mad at me because I'm doing so much work, and when we come back, it'll be exactly the same thing we started with.

So here's the cool part. Let me know if you find it in your problem. To find theta d and d [inaudible] duo, and this is how the function of alpha and beta is. It's not the function of the Lagrange multipliers. It's not of w. To find this, we minimize over w of my generalized Lagrange.

And my duo problem is this. So in other words, this is max over that. And so this is my duo optimization problem. To maximize over alpha and beta, theta d over alpha and beta. So this optimization problem, I guess, is my duo problem.

I want you to compare this to our previous prime optimization problem. The only difference is that I took the max and min, and I switched the order around with the max and min. That's the difference in the primal and the duo optimization [inaudible].

And it turns out that it's a – it's sort of – it's a fact – it's true, generally, that d star is less than [inaudible] p star. In other words, I think I defined p star previously. P star was a value of the prime optimization problem. And in other words, that it's just generally true that the max of the min of something is less than equal to the min of the max of something. And this is a general fact.

And just as a concrete example, the max over y in the set 01 x – oh, excuse me, of the min of the set in 01 of indicator x = y – this is [inaudible] equal to min.

So this equality – this inequality actually holds true for any function you might find in here. And this is one specific example where the min over xy – excuse me, min over x of [inaudible] equals y – this is always equal to 0 because whatever y is, you can choose x to be something different. So this is always 0, whereas if I exchange the order to min and max, then thing here is always equal to 1. So 0 [inaudible] to 1.

And more generally, this min/max – excuse me, this max/min, thus with the min/max holds true for any function you might put in there.

But it turns out that sometimes under certain conditions, these two optimization problems have the same value. Sometimes under certain conditions, the primal and the duo problems have the same value. And so you might be able to solve the duo problem rather than the primal problem.

And the reason to do that is that sometimes, which we'll see in the optimal margin classifier problem, the support vector machine problem, the duo problem turns out to be much easier than it – often has many useful properties that will make user compared to the primal.

So for the sake of – so what I'm going to do now is write down formally the certain conditions under which that's true – where the primal and the duo problems are equivalent. And so our strategy for working out the [inaudible] of support vector machine algorithm will be that we'll write down the primal optimization problem, which we did previously, and maximizing classifier.

And then we'll derive the duo optimization problem for that. And then we'll solve the duo problem. And by modifying that a little bit, that's how we'll derive this support vector machine.

But let me ask you – for now, let me just first, for the sake of completeness, I just write down the conditions under which the primal and the duo optimization problems give you the same solutions. So let f be convex. If you're not sure what convex means, for the purposes of this class, you can take it to mean that the Hessian, h is positive. [Inaudible], so it just means it's a [inaudible] function like that.

And once you learn more about optimization – again, please come to this week's discussion session taught by the TAs.

Then suppose hi – the hi constraints [inaudible], and what that means is that hi of w equals alpha i transpose w plus vi. This actually means the same thing as linear. Without the term b here, we say that hi is linear where we have a constant interceptor as well. This is technically called [inaudible] other than linear.

And let's suppose that gi's are strictly feasible. And what that means is that there is just a value of the w such that from i, gi of w is less than 0. Don't worry too much [inaudible]. I'm writing these things down for the sake of completeness, but don't worry too much about all the technical details. Strictly feasible, which just means that there's a value of w such that all of these constraints are satisfy were stricter than the equality rather than what less than equal to.

Under these conditions, there were exists w star, alpha star, beta star such that w star solves the primal problem. And alpha star and beta star, the Lagrange multipliers, solve the duo problem. And the value of the primal problem will be equal to the value of the duo problem will be equal to the value of your Lagrange multiplier – excuse me, will be equal to the value of your generalized Lagrange, the value of that w star, alpha star, beta star.

In other words, you can solve either the primal or the duo problem. You get the same solution.

Further, your parameters will satisfy these conditions. Partial derivative perspective parameters would be 0. And actually, to keep this equation in mind, we'll actually use this in a second when we take the Lagrange, and we – and our support vector machine problem, and take a derivative with respect to w to solve a – to solve our – to derive our duo problem. We'll actually perform this step ourselves in a second.

Partial derivative with respect to the Lagrange multiplier beta is equal to 0. Turns out this will hold true, too. This is called the – well – this is called the KKT complementary condition. KKT stands for Karush-Kuhn-Tucker, which were the authors of this theorem. Well, and by tradition, usually this [inaudible] KKT conditions. But the other two are – just so the [inaudible] is greater than 0, which we had previously and that your constraints are actually satisfied.

So let's see. [Inaudible] All right. So let's take those and apply this to our optimal margin optimization problem that we had previously. I was gonna say one word about this, which is – was gonna say one word about this KTT complementary condition is that a condition that is a – at your solution, you must have that alpha star i times gi of w is equal to 0.

So let's see. So the product of two numbers is equal to 0. That means that at least one of these things must be equal to 0. For the product of two things to be equal to 0, well, that's just saying either alpha or i or gi is equal to 0.

So what that implies is that the – just Karush-Kuhn-Tucker – most people just say KKT, but we wanna show you the right spelling of their names. So KKT complementary condition implies that if alpha i is not 0, that necessarily implies that gi of w star is equal to 0. And usually, it turns out – so all the KKT condition guarantees is that at least one of them is 0. It may actually be the case that both alpha and gi are both equal to 0. But in practice, when you solve this optimization problem, you find that to a large part, alpha i star is not equal to 0 if and only gi of w star 0, 0.

This is not strictly true because it's possible that both of these may be 0. But in practice, when we – because when we solve problems like these, you're, for the most part, usually exactly one of these will be non-0.

And also, when this holds true, when gi of w star is equal to 0, we say that gi – gi of w, I guess, is an active constraint because we call a constraint – our constraint was a gi of w must be less than or equal to 0. And so it is equal to 0, then we say that that's a constraint that this is an active constraint.

Once we talk about [inaudible], we come back and [inaudible] and just extend this idea a little bit more. [Inaudible] board. [Inaudible] turn to this board in a second, but – so let's go back and work out one of the primal and the duo optimization problems for our optimal margin classifier for the optimization problem that we worked on just now.

As a point of notation, in whatever I've been writing down so far in deriving the KKT conditions, when Lagrange multipliers were alpha i and beta i, it turns out that when applied as [inaudible] dm, turns out we only have one set of Lagrange multipliers alpha i.

And also, as I was working out the KKT conditions, I used w to denote the parameters of my primal optimization problem. [Inaudible] I wanted to minimize f of w. In my very first optimization problem, I had that optimization problem [inaudible] finding the parameters w.

In my svn problem, I'm actually gonna have two sets of parameters, w and b. So this is just a – keep that sort of slight notation change in mind.

So problem we worked out previously was we want to minimize the normal w squared and just add a half there by convention because it makes other work – math work a little nicer. And subject to the constraint that yi x w [inaudible] xi + v must be = greater than 1.

And so let me just take this constraint, and I'll rewrite it as a constraint. It's gi of w, b. Again, previously, I had gi of w, but now I have parameters w and b. So gi of w, b defined as 1.

So let's look at the implications of this in terms of the KKT duo complementary condition again. So we have that alpha i is basically equal to 0. That necessarily implies that gi of w, b is equal to 0. In other words, this is an active constraint.

And what does this mean? It means that it actually turns out gi of wb equal to 0 that is – that means exactly that the training example xi, yi has functional margin equal to 1. Because this constraint was that the functional margin of every example has to be greater equal to 1. And so if this is an active constraint, it just – inequality holds that equality. That means that my training example i must have functional margin equal to exactly 1.

And so – actually, yeah, right now, I'll do this on a different board, I guess. So in pictures, what that means is that, you have some training sets, and you'll have some separating hyperplane. And so the examples with functional margin equal to 1 will be exactly those which are – so they're closest to my separating hyperplane.

So that's my equation. [Inaudible] equal to 0. And so in this – in this cartoon example that I've done, it'll be exactly – these three examples that have functional margin equal to 1, and all of the other examples as being further away than these three will have functional margin that is strictly greater than 1.

And the examples with functional margin equal to 1 will usually correspond to the ones where the corresponding Lagrange multipliers also not equal to 0. And again, it may not hold true. It may be the case that gi and alpha i equal to 0. But usually, when gi's not – is 0, alpha i will be non-0.

And so the examples of functional margin equal to 1 will be the ones where alpha i is not equal to 0.

One useful property of this is that as suggested by this picture and so true in general as well, it turns out that we find a solution to this – to the optimization problem, you find that relatively few training examples have functional margin equal to 1.

In this picture I've drawn, there are three examples with functional margin equal to 1. There are just few examples of this minimum possible distance to your separating hyperplane. And these are three – these examples of functional margin equal to 1 – they are what we're going to call the support vectors. And this needs the name support vector machine. There'll be these three points with functional margin 1 that we're calling support vectors.

And the fact that they're relatively few support vectors also means that usually, most of the alpha i's are equal to 0. So with alpha i equal to 0, for examples, though, not support vectors.

Let's go ahead and work out the actual optimization problem. So we have a [inaudible] margin optimization problem. So there we go and write down the margin, and because we only have inequality constraints where we really have gi star constraints, no hi star constraint. We have inequality constraints and no equality constraints, I'll only have Lagrange multipliers of type alpha – no betas in my generalized Lagrange. But my Lagrange will be one-half w squared minus. That's my Lagrange.

And so let's work out what the duo problem is. And to do that, I need to figure out what theta d of alpha – and I know again, beta's there – so what theta d of alpha is min with respect to wb of lb alpha. So the duo problem is the maximize theta d as the function of alpha. So as to work out what theta d is, and then that'll give us our duo problem.

So then to work out what this is, what do you need to do? We need to take a look at Lagrange and minimize it as a function of lv and b so – and what is this?

How do you minimize Lagrange? So in order to minimize the Lagrange as a function of w and b, we do the usual thing. We take the derivatives of w – Lagrange with respect to w and b. And we set that to 0. That's how we minimize the Lagrange with respect to w and b.

So take the derivative with respect to w of the Lagrange. And I want – I just write down the answer. You know how to do calculus like this. So I wanna minimize this function of w, so I take the derivative and set it to 0. And I get that. And then so this implies that w must be that.

And so w, therefore, is actually a linear combination of your input feature vectors xi. This is sum of your various weights given by the alpha i's and times the xi's, which are your examples in your training set. And this will be useful later.

The other equation we have is – here, partial derivative of Lagrange with respect to p is equal to minus sum of i plus 1 to m [inaudible] for i. And so I'll just set that to equal to 0. And so these are my two constraints. And so [inaudible].

So what I'm going to do is I'm actually going to take these two constraints, and well, I'm going to take whatever I thought to be the value for w. And I'm gonna take what I've worked out to be the value for w, and I'll plug it back in there to figure out what the Lagrange really is when I minimize with respect to w. [Inaudible] and I'll deal with b in a second.

So let's see. So my Lagrange is 1/2 w transpose w minus. So this first term, w transpose w – this becomes sum y equals one to m, alpha i, yi, xi transpose. This is just putting in the value for w that I worked out previously.

But since this is w transpose w – and so when they expand out of this quadratic function, and when I plug in w over there as well, I find that I have that. Oh, where I'm using these angle brackets to denote end product, so this thing here, it just means the end product, xi transpose xj. And the first and second terms are actually the same except for the minus one half. So to simplify to be equal to that.

So let me go ahead and call this w of alpha. My duo problem is, therefore, the following. I want to maximize w of alpha, which is that [inaudible]. And I want to the – I realize the notation is somewhat unfortunate. I'm using capital W of alpha to denote that formula I wrote down earlier.

And then we also had our lowercase w. The original [inaudible] is the primal problem. Lowercase w transpose xi. So uppercase and lowercase w are totally different things, so unfortunately, the notation is standard as well, as far as I know, so. So the duo problem is that subject to the alpha [inaudible] related to 0, and we also have that the sum of i, yi, alpha i is related to 0.

That last constraint was the constraint I got from this – the sum of i – sum of i, yi alpha i equals to 0. But that's where that [inaudible] came from.

Let me just – I think in previous years that I taught this, where this constraint comes from is just – is slightly confusing. So let me just take two minutes to say what the real interpretation of that is. And if you don't understand it, it's not a big deal, I guess.

So when we took the partial derivative of the Lagrange with respect to b, we end up with this constraint that sum of i, yi, alpha i must be equal to 0. The interpretation of that, it turns out, is that if sum of i, yi, alpha i is not equal to 0, then theta d of wb is – actually, excuse me. Then theta d of alpha is equal to minus infinity for minimizing.

So in other words, it turns out my Lagrange is actually a linear function of my parameters b. And so the interpretation of that constraint we worked out previously was that if sum of i or yi, alpha i is not equal to 0, then theta d of alpha is equal to minus infinity. And so if your goal is to maximize as a function of alpha, theta d of alpha, then you've gotta choose values of alpha for which sum of yi alpha is equal to 0.

And then when sum of yi alpha is equal to 0, then theta d of alpha is equal to w of alpha. And so that's why we ended up deciding to maximize w of alpha subject to that sum of yi alpha is equal to 0.

Yeah, the – unfortunately, the fact of that d would be [inaudible] adds just a little bit of extra notation in our derivation of the duo. But by the way, and [inaudible] all the action of the optimization problem is with w because b is just one parameter.

So let's check. Are there any questions about this?

Okay, cool. So what derived a duo optimization problem – and really, don't worry about this if you're not quite sure where this was. Just think of this as we worked out this constraint, and we worked out, and we took partial derivative with respect to b, that this constraint has the [inaudible] and so I just copied that over here.

But so – worked out the duo of the optimization problem, so our approach to finding – to deriving the optimal margin classifier or support vector machine will be that we'll solve along this duo optimization problem for the parameters alpha star. And then if you want, you can then – this is the equation that we worked out on the previous board. We said that w – this [inaudible] alpha – w must be equal to that. And so once you solve for alpha, you can then go back and quickly derive w in parameters to your primal problem. And we worked this out earlier.

And moreover, once you solve alpha and w, you can then focus back into your – once you solve for alpha and w, it's really easy to solve for v, so that b gives us the interpretation of [inaudible] training set, and you found the direction for w. So you know where your separating hyperplane's direction is. You know it's got to be one of these things. And you know the orientation and separating hyperplane. You just have to decide where to place this hyperplane. And that's what solving b is.

So once you solve for alpha and w, it's really easy to solve b. You can plug alpha and w back into the primal optimization problem and solve for b.

And I just wrote it down for the sake of completeness, but – and the intuition behind this formula is just that find the worst positive [inaudible] and the worst negative example. Let's say this one and this one – say [inaudible] and [inaudible] the difference between them. And that tells you where you should set the threshold for where to place the separating hyperplane.

And then that's the – this is the optimal margin classifier. This is also called a support vector machine. If you do not use one y [inaudible], it's called kernels. And I'll say a few words about that.

But I hope the process is clear. It's a duo problem. We're going to solve the duo problem for the alpha i's. That gives us w, and that gives us b.

So there's just one more thing I wanna point out as I lead into the next lecture, which is that – I'll just write this out again, I guess – which is that it turns out we can take the entire algorithm, and we can express the entire algorithm in terms of inner products. And here's what I mean by that.

So say that the parameters w is the sum of your input examples. And so we need to make a prediction. Someone gives you a new value of x. You want a value of the hypothesis on the value of x. That's given by g of w transpose x plus b, or where g was this threshold function that outputs minus 1 or plus 1. And so you need to compute w transpose x plus b. And that is equal to alpha i, yi.

And that can be expressed as a sum of these inner products between your training examples and this new value of x [inaudible] value [inaudible].

And this will lead into our next lecture, which is the idea of kernels. And it turns out that in the source of feature spaces where used to support vector machines – it turns out that sometimes your training examples may be very high-dimensional. It may even be the case that the features that you want to use are inner-dimensional feature vectors.

But despite this, it'll turn out that there'll be an interesting representation that you can use that will allow you to compute inner products like these efficiently. And this holds true only for certain feature spaces. It doesn't hold true for arbitrary sets of features.

But we talk about the idea of kernels. In the next lecture, we'll see examples where even though you have extremely high-dimensional feature vectors, you can compute – you may never want to represent xi, x plus [inaudible] inner-dimensional feature vector. You can even store in computer memory. But you will nonetheless be able to compute inner products between different [inaudible] feature vectors very efficiently. And so you can – for example, you can make predictions by making use of these inner products.

This is just xi transpose. You will compute these inner products very efficiently and, therefore, make predictions. And this pointed also – the other reason we derive the duo was because on this board, when we worked out what w of alpha is, w of alpha – actually are the same property – w of alpha is again written in terms of these inner products.

And so if you actually look at the duo optimization problem and step – for all the steps of the algorithm, you'll find that you actually do everything you want – learn the parameters of alpha. So suppose you do an optimization problem, go into parameters alpha, and you do everything you want without ever needing to represent xi directly. And all you need to do is represent this compute inner products with your feature vectors like these.

Well, one last property of this algorithm that's kinda nice is that I said previously that the alpha i's are 0 only for the – are non-0 only for the support vectors, only for the vectors that function y [inaudible] 1.

And in practice, there are usually fairly few of them. And so what this means is that if you're representing w this way, then w when represented as a fairly small fraction of training examples because mostly alpha i's is 0 – and so when you're summing up the sum, you need to compute inner products only if the support vectors, which is usually a small fraction of your training set. So that's another nice [inaudible] because [inaudible] alpha is 0.

And well, much of this will make much more sense when we talk about kernels.

[Inaudible] quick questions before I close? Yeah.

Student: It seems that for anything we've done the work, the point file has to be really well behaved, and if any of the points are kinda on the wrong side –

Instructor (Andrew Ng):No, oh, yeah, so again, for today's lecture asks you that the data is linearly separable – that you can actually get perfect [inaudible]. I'll fix this in the next lecture as well. But excellent assumption.

Yes?

Student: So can't we assume that [inaudible] point [inaudible], so [inaudible] have [inaudible]?

Instructor (Andrew Ng):Yes, so unless I – says that there are ways to generalize this in multiple classes that I probably won't [inaudible] – but yeah, that's generalization [inaudible].

Okay. Let's close for today, then. We'll talk about kernels in our next lecture.

[End of Audio]

Duration: 77 minutes