GILBERT STRANG: OK. Let me just say our next topic, moving really toward machine learning, would be the last section of chapter 6 about stochastic gradient descent. I think Professor Sra from computer science, if he is able, he will take the class Friday and tell us about SGDs, Stochastic Gradient Descent, critical algorithm. I mean that, at a quick look, stochastic gradient descent does smaller batches at a time, so it's computationally faster than pure gradient descent. But also, stochastic gradient descent is able to solve probabilistic problems where you're trying to minimize an expected value instead of a function.

Anyway, it's an important thing. Actually, there is a lecture at 2:00 in neuroscience about the gradient descent algorithms. And then Professor Boyd is speaking here at 4:30 this afternoon. It's overwhelming.

So I'm going to talk today about several topics that are specific parts of optimization. Linear programming is a very famous part of optimization. I don't know if you've met it. I think it's worth knowing what the inputs are and what's the key fact about linear programming. And, of course, also, what are the algorithms.

That's usually what we want to know. What's the problem? What are the results, the mathematical results? And what are the computational tools? Yes, so that's very established. Max-flow min-cut is one specific linear programming that I think maybe would serve as the best example.

And then two-person games, have you met those? So would you know what the rules are in a two-person zero-sum game? In game theory, those are the ones that are clear-- everything's well-established there. And, in fact, they're equivalent to a linear program. So that's why those two are coming in today.

And then the key fact about two-person games and about linear programming is duality. So if there's any word on that board, it's the one in capital letters that has math content. I'm just going to start with linear programming and then move on.

So what's a linear program? It's we're optimizing a linear cost function. So we're minimizing the cost c transpose x. So that vector x is the unknown that we're looking for, and this vector c is the cost vector. So that is c1 x1 plus, plus cn xn.

So you can see why it's called linear programming. The cost is linear. The constraints are also linear. So the constraints on x are a set of linear equations. Of course, I'm not thinking of A as being a square invertible matrix. No way. If it were, that would tell us what x had to be and our problem would be over.

We have n unknowns. This is m by n, of course. x is n by 1. That's our unknown vector. And I'm thinking that m would be smaller than n.

But we do have constraints, and now comes the thing that makes linear programming not actually linear. And that's the constraint x greater or equal to 0. And I've written that as a vector but that means x1 greater or equal to 0, on to xn greater or equal to 0. So it's a vector inequality.

So we have minimizing a very simple function with pretty straightforward constraints but inequality constraints. So the set of-- these two together are the constraints. And in linear algebra-- in linear programming language this is called a feasible set of x's. It's the constraint set, capital K in the notes.

So let me draw a picture to show-- or maybe I'll just ask. How many of you are already familiar with linear programming? Oh, quite a lot. So I won't belabor the point. But let me just draw a picture.

So my picture will be just here. So that's n equals 3. So this is the x1, x2, x3 space. And x greater or equal to 0 means what? It means that I'm in this 1/8 of this space, right? It's the non-negative-- well, I would say quadrant but I really should say octant because it's 1/8 of the full 3D space.

So I'm in here. And then maybe I have m as maybe just one constraint. So let me take the cost to be, say, x1 plus 2 x2 plus 5 x3, say. And the constraint, I'm just going to have one equation-- x1 plus x2. Let me make it easy, make that constraint an easy one to-- so this is what we want to minimize. And that's what we have to satisfy, as well as x greater or equal to 0.

So that's a plane. One equation gives us a plane in R3. The plane would go through-- would hit the axes at 3, 0, 0. And so the min-- the point has to lie on that plane. And it has to lie in the octant, so that plane is chopped off to be a triangle. This is a good visualization of linear programming.

And what's the conclusion? Well, our cost is linear. So the result is that one of these three corners is the winner. It could happen. It could happen that maybe I have equal values on those two corners, and therefore all the way along would be also winners.

But when I have a linear function, it's a maximum at the ends. And these are the ends, those three corners. So 3, 0, 0, 0, 3, 0, or 0, 0, 3 are the three corners, and one of those is the winner and the problem is solved.

And, in fact, for this case, since I'm minimizing, I guess it would be that corner that wins. So let me give it a star, not an x. Yes. 3, 0, 0 because that-- oh. No, that's 3. So the value turned out to be 3 at the point 3, 0, 0. And that's x star. Good. Good, good, good.

What more do I want to do, I said? Yes.

AUDIENCE: Shouldn't it be at point 0, 0, 3?

GILBERT STRANG: Should it be-- oh, no. 0, 0, 3 would have x3 is 3. That's fine. It gives us a chance, I think. That would give me a cost of 15. This would give me a cost of 6, and that would give me a cost of 3. Yes.

It's obvious that if we could enumerate all the corners, we would have a super fast way to get the answer. But the trouble is, of course, that for large values of m and n, there are exponentially many corners, and we don't want to see them all. So that's what makes linear programming take time and need ideas.

So, for algorithms, there are two types of algorithms, two types of codes that solve these. The older, well-established ones are the-- so one way is the simplex method, which finds a corner. We know the optimum one of the corners. So it will find one corner, and it will go to the next corner.

So if it starts at one of these corners, it will travel along an edge that lowers the cost. And it has to stop at the end because it's linear. The cost will keep going down and going down until it bumps into the end point, and then it can't go further. And then we would, from that next corner, we would go to the next corner.

Each time, it's like steepest descent on the edges. From the first corner, we go the steepest way till we can't go further, we've hit another corner. And we recompute the linear algebra, find which direction is steepest from there.

So that's the idea of the simplex method, which was invented by Dantzig. And the algebra is not going to be in today's lecture but it's straightforward. Well, people have to optimize it because that's a highly frequently used method. Yes.

But then, about 20 years later maybe, or 30, I remember going to a lecture in downtown Boston by Karmarkar. So I have to put his name down-- Karmarkar. And he was in *The New York Times,* all the newspapers, so it was a big deal.

He had an alternative algorithm. And the exact algorithm he proposed hasn't survived until today but the idea has. And the idea was to go to travel inside the feasible set and not around the edges. So his idea-- because in here, would maybe travel down near that edge, start again, travel again. So it's steepest descent in the constraint set, the feasible set. And you can use calculus and this idea.

So this is interior point method. So I'll just use the word interior. That's telling us that we're inside the feasible set. We don't hit, come all the way to the boundary intentionally, because we want room to move, and to find derivatives, and to use Newton's method to minimize.

You choose a search direction, just as all of optimization does. And in that search direction, you track it and you stop before you bump into the edge of the feasible set. And then you compute derivatives again. So you can use calculus.

And it's a method that-- it's an idea that-- the interior idea came before Karmarkar. But he got super publicity, so it really got attention, got new thinking. And new ideas came partly from people at MIT.

And these two are now still locked in competition. One hasn't beaten the other in all problems. So linear programming is here. But then, for nonlinear programming, quadratic programming, where the cost is quadratic, nonlinear programming, semi-definite programming-- that's where you have a matrix unknown and matrix constraints-- those are all-- the more complicated you get, the more it tends to be interior point methods.

That's a summary of linear programming. Now I'd like to give an example. And then I haven't told you the main-- well, let me tell you the main fact about duality. This is really-- and I'll write that maybe here next to it.

So duality is there's a dual program, a dual LP. And that dual is going to do a maximum instead of a minimum And the cost is going to involve the b from the primal problem. So this is now called the primal problem, and this is its dual.

So they're twin problems. They use the same data but quite differently. It's like transposing things. So let's call it the unknown y. So for y1 to ym, I guess, because b, the right-hand side over here, is m by 1. So that's maximize that subject to-- what are the constraints?

Well, the cost function over here, c, the cost vector, goes into the constraints. And I think the greater or equal sign probably becomes a less or equal to sign. The A gets transposed. I think that's probably the constraint in the dual. And it happens it doesn't need y. We don't have y greater or equal to 0 over here.

So that's a dual problem. It has a linear cost. It has linear inequality constraints. It can be solved by the simplex method. You could choose whether you solve that one or this one, because if you solve one, you solve the other. The two are closely connected and that's the key idea of duality.

So maybe I'll put the idea of, first of all, a weak duality. Which says that this quantity that we're trying to maximize-- we're getting it as large as possible-- is always less or equal to any c transpose x. This is for any feasible, allowed-- any x and y that satisfy the constraints. Remember, feasible means satisfies the constraints.

So, in other words, this problem, this maximization problem, you're trying to push this up. The minimization problem, you're trying to pull this down. And it's easier to show that the one, the b transpose-- it's easier to show that inequality. Let's do that.

So b transpose y proof. It should be a one-line proof. So b transpose y-- what do I do? So b transpose y well, I look over here. That's where b shows up.

That's x transpose A transpose y, because b is Ax. I'm feasible, so my b is Ax. And for any x-- this is for this is for any x that b is going to be Ax. So am I good so far?

And now what do I do? Now I look here. I see A transpose y sitting right in front of me. So I say, well, OK, less or equal to. And I guess I'm done.

A transpose y is less or equal to c. So I have x transpose c, c transpose x. So this equals this, less or equal that. I've got it. Weak duality is just put together the requirements on x and y and you have it.

But was this important? If the mathematics is right, everything has to have its place, play its role. And so what's the role of x greater or equal to 0 there? Why do we need x-- If we just even think of n equal 1, just a number, where do we use the fact that x is greater or equal to 0? Do we use it?

This looks so smooth, because A transpose y-- I'll write that as x transpose c for the moment, just so your eye says that's the same x transpose, and the A transpose y is less or equal to c. Where is x greater or equal to 0 coming in?

AUDIENCE: [INAUDIBLE].

GILBERT STRANG: Yes.

AUDIENCE: Like above. It's from the that.

GILBERT STRANG: Yes, that's true. But I want to see-- so here's my point. The fact that A transpose y is less or equal to c, that's fine. Highly important. But if x was negative, it would be the other way. You would go the other way and you wouldn't have what you want.

So we really do use the fact that x is greater or equal to 0 to say that I have this less or equal this. And then I multiply by something positive and then I still have less or equal. OK, good. So the math is right. Everything does its part.

And then, of course, the beautiful result, the important result is that there is strong duality, just called duality, which is that at the maximum-- now this is not for any x and y but this is for-- this is going to be for x y star, the winner, and x star, the winner-- equality holds. So that's duality. The maximum in the dual problem is the same as the minimum in the primal problem. The two have met. There is no duality gap.

In some cooked up nonlinear problems, there could be a gap between the maximum and the minimum, but you hope not. And here the big theorem is not. They're equal. You push this up, push that down.

Another way to say that duality is that this is pushing that up at some max-- I could write that as a maximum of a minimum equaling a minimum of a maximum, if I wanted. So the duality in linear programming was the same as von Neumann's minimax theorem. And his theorem applied to two-person games.

So the key math result is duality for linear programming, and it's going to be-- you'll see the same thing happening for two-person games. And it's a minimax theorem, or a saddle point. Or it's just things come out right. Yes.

So just to mention that mathematical programming, of course, includes much more difficult problems. This is linear programming. That problem, as you see, has a beautiful, simple theory. And the paying attention is paying attention to the algorithm, because you've got two important choices and they both get highly developed.

Now, OK. So for game, now I'm going to turn to-- well, I'll do an example of max flow equal min cut, just see what-- and then go to two-person game. So here's an example of a-- so I start with a graph. Let me just imagine.

So this is the source. This is the sink. And I'm sending flow through the network. So it's a network that I'm sending flow through. So my job is to maximize-- I'll set x at the source to be 0. And then the flow, the total flow will come into the sink. So I want to maximize x at the sink.

So it's a linear programming and there are constraints. So what are the constraints? Every edge has a capacity. So let's see. Suppose that edge has capacity 5. Let me put capacities on all these edges. 2, 1, 3, 4. I'm just stabbing around here. 1,000. And let me say 2 and 4.

I have no idea what's happening here. But if we see the problem, we'll probably be able to solve it. So these constraints are that the flow variable, which would be the y that I'm trying to maximize, cannot-- the amount of flow, just in ordinary words, the amount of flow on the edges can't go higher than the capacity.

I could send 11 along this edge, but then I've got nowhere to send it after that. I could send, well, 900 on that edge. but, obviously, it would get stuck there. So the question is, what's the maximum I can send through that network? It's a classical problem.

And, in fact, it's an integer programming problem. These are all integer capacities. I could insist on integer flow. But it's a very remarkable integer problem because I could allow fractions and the answer would not be a-- would not involve fractions.

In other words, if I keep it as an integer problem, then the mathematics is definitely harder for an integer problem. What's different? So the x's could be integers or they could be real numbers. Over here, they were real numbers. I happened to get an integer for this, but if I had 10 crossing planes in 15 dimensions, the integers would be totally lost.

But the point is, here, that if I allow real numbers, it doesn't get me any more flow, that the winning flow is an integer anyway. So it's an integer problem which can be safely relaxed, and you can safely use simplex method, or Karmarkar's method, or any interior point method with non-integers, because in the interior here you're not starting or ending at integers. You can do it because the integer answer will be, in the end, better.

And what is that answer? I think I've made those capacities too easy to-- oh, I didn't do this one. So what shall I-- shall I make it large, like 9? I don't know.

What's the best we can do through that network? Can you-- I can't see it from here yet? What should I do? Obviously, in this simple problem I can get anything I want pretty much that far. But then what can I do?

Let me even make that 19. So I'm not imposing much of a limit on that edge either. But these edges are quite tight limits, and these are sort of intermediate limits.

What do you think is the most I can send through? And how would you show me that I couldn't send more? That's the key question. I want to get a bound on the-- an upper bound on this maximum. This maximum getting into the sink is less or equal to-- and what number would you propose?

Could I get 1,000 through? I could start, but of course it would pile up there. I couldn't get further. What do you think is the best I can do?

AUDIENCE: 10?

GILBERT STRANG: 10? I can't do more than 10? How could I do 10?

AUDIENCE: So you go 12--

GILBERT 12 this way?

STRANG:

AUDIENCE: Yes. 19.

GILBERT STRANG: Oh, OK.

AUDIENCE: And then split the 8 and 2. Can you split?

GILBERT STRANG: I see. 10 that way, 10 this-- yes, absolutely you can split. And then this could go here, and that would be able to go there. So I can get 10 through. Correct.

AUDIENCE: [INAUDIBLE].

GILBERT STRANG: Can I do anything better? Oh, I could be sending some up this way at the same time. So I could get 3 along the top. So this is like an auction.

We can get up to 13. Can we get-- so that was 3 going this way and 10 going this way. Is 13 an-- can I not exceed 13?

AUDIENCE: 14?

AUDIENCE: 14.

GILBERT STRANG: Do I hear 14? Oh, I've got a lot of room for one more. So, OK. So 14 in any case. All right. Do I hear 15?

AUDIENCE: Can you do two more on the bottom, like 12 instead of 10?

GILBERT STRANG: If I did 12, then what would I do?

AUDIENCE: Split the other two up.

GILBERT STRANG: I'd send two up here. What am I going to do with them from here?

AUDIENCE: [INAUDIBLE].

GILBERT STRANG: Send them?

AUDIENCE: Up again.

GILBERT STRANG: Up again, and along. But then the 3 that I had right now would be cut back to 1.

It's a lot of fun, this max flow problem. And I'm looking for a bound to know when to quit, to know when I've optimized. That's the whole idea of duality, is to find some upper thing that I'm trying to push down but I can't go beyond it.

So I don't think I could get more than-- you see, everything has to cross this middle. So 3 and 4. And I don't think I could beat 23. If somebody said more than 23, I would be very doubtful, because I couldn't get it across the middle.

But then, can I get 23? I doubt it. Maybe 14 is the best possible. So how would we show that 14 is the best possible?

I think if I could find-- so this is called a cut, a cut in this network. Oh, yes, I see a cut. A cut like there. You see that?

Every bit of flow has got to cross that cut. And the total capacity crossing is the 3 and the 1 and the 2 and the 8, which is 14. So I can't get more than 14 through. And somehow that cut is loaded to capacity. Probably those edges all have to be fully up to capacity that cross the cut.

So the cut is a separation of edges that go with the source and-- sorry-- nodes that go with the source, nodes that go with the sink. Yes. And then it's the edges across the cut.

Is that OK for an example? So that's the duality. The maximum flow turned out to be 14, and the minimum cut turned out to be 14. And when those match, 14 equal 14, I know I'm through. I know I'm through because I'm able to get 14 through and I could never get more. Yes. 3 and 4 and 6 and 8. Yes.

And, of course, in a big network, the maximum cut is not going to be visible. Well, you couldn't-- it would have thousands of edges. You couldn't see what you were doing, But you could solve this problem fast, actually fast.

And it's an important-- in practice, it's an important example. A lot of other things fit into the max flow min cut example. And therefore solving it in faster than-- oh, I didn't even say about speed.

So the simplex method, almost always it's average case. Dan Spielman, who was on the faculty here, who's just terrific in this area, was maybe among the first to study the average case.

So for an average choice of A and b and c, instead of making the worst choice-- for the worst choice, you can create a feasible set that it has to go corner, corner, corner, corner to get to the end, so the simplex method would take exponentially long. But that's extremely rare. It doesn't happen in practice, I think. And the average one is a polynomial, so it's a fast method on average.

And we can show-- actually, that was a famous result that came from Russia, that linear programming is in the big P versus LP world, linear programming goes with P. Linear programming is a problem that can be solved in polynomial time. The simplex method won't always do it but it can be done. Yes. There came into the world, the ellipsoid methods, and it was an exciting time to decide that linear programming was actually P and not LP.

So this is a case of duality where you can understand duality by the statement that the flow cannot exceed the capacity of the cut. So that's what duality is, that any flow has to be less or equal the capacity of any cut, the capacity being the sum of the ones.

So, of course, there are other cuts. That cut has to be crossed, but that's gone 3, 7, 14, 22 capacity. So the 14 capacity was the minimum, and that gave the maximum. Nice. Isn't it nice? Yes.

So there is another world here of two-person games that also has duality and could be expressed. So let me just talk finally today about two-person games. So the two persons are x and y of course. What other names could they have?

And there is a matrix involved. This is the payoff matrix. So let me take a very simple game first.

So x is going to choose one of these rows, and y is going to choose one of these columns. And let's say payoff from x to y. I like it that way because, then, x, who's paying, is going to be minimizing, as x was up here. And y, who's collecting, is going to be maximizing, as in the dual.

So the payoff-- well, let's see. I think maybe 1, 4, 2, 8 would be probably a fairly easy game to play. So it's a two-person game, a zero-sum game. Zero sum means-- that means all pay-- what x pays goes to y. y gets all that x pays. There's no third party here. No lawyers involved.

And y is going to choose a column, and x is going to choose a row. And x wants to make it small, and y wants to make it big. So what happens in this game?

What does y choose to make it big? The second column. What does x choose to make it small? The first row. So it's going to be focused in on that 2, because if y keeps choosing that column, 2 is the best-- the least that-- x is going to have to pay 2, and he achieves that by choosing the first row.

So it's a simple game. That's a saddle point. It's a minimum for y in its column, and it's a maximum for x in its row. But, of course, a matrix, another matrix might not have such a saddle point.

So let me-- do you see-- OK with this game? That would be a sort of straightforward game where simple strategy-- column 2 every time, row 1 every time is the optimal. But now let me just exchange that.

So, again, x1 and x2. y1 and y2 are the two columns. What happens now? Well, y kind of likes the big number 8 there. So it goes for the second. The second column still has the bigger numbers. So y aims for column 2.

But then what does x do? What row does x choose if y is in the column 2?

AUDIENCE: Second.

GILBERT STRANG: The second. OK. So have I found a saddle point? Where y chooses this column, x chooses this row, is that a saddle point? No. Because what will y do now?

He sees x choosing the second row all the time, and y sees a 4, a very tempting 4, in that row. So y is going to choose y1, the first column, pretty often. Not all the time, because what happens if he just-- if y chooses this column all the time, then x will choose this row and the payoff would only be 1, so that things went downhill for y there.

So this is not a saddle point. And what do we do? Mixed strategy, which is y will choose the two columns with some probabilities that add to 1.

So there will be a third possibility, that p x1 and 1-- I'm sorry-- p y1. p times this column and 1 minus p times the second column. So that will be p plus 1 minus p times 8, and 4p plus 1 minus p times 2.

That's this mixed column. By mixing his strategy, we have a strategy like a third person, y3. And, of course, the x is going to notice after a while what the strategy is. This is an open competition. You're not hiding-- you're not able to hide anything.

You might think, well, maybe y will jump around, but that's foolishness. y is going to end up finding the best choice of p between 0 and 1. And it will actually be between 0 and 1 because the extreme strategies of column 1 and column 2 were not winners.

But now x is going to do the same thing. p x1-- he's going to-- he or she is going to combine those rows. So this would be p1 for the first row, and 1-- sorry-- p-- oh, I better choose another number. What's another letter than p?

AUDIENCE: q.

GILBERT STRANG: q? OK, q. So this row is a combination of those rows, so it's q plus 1 minus q times 4 in the first position, and p times 8 and 1 minus p-- q times 8. q times 8 and 1 minus q times 2. I'm sorry that I've written this too small.

But what's going to happen? What's going to determine p and q and solve the game? Well, if these are equal, then what happens? Then x is OK with either one. x has nothing to choose if those are equal and y is staying with his mixed strategy, which gives him this third combination column. Then x is good.

Now, you might say, well, then x could do what it wanted. But x couldn't do what it wanted. If x doesn't stick with the optimal strategy q for x, then y will take advantage. So, really, the game settles down to once-- so these should be equal.

So when those are equal, what do I get? I have p-- it looks like I have 8 minus 7p there for the first one. And here, I have 4p minus 2p. That's 2p plus 1. Did I get that possibly right?

AUDIENCE: Plus 2.

GILBERT STRANG: 2, 4p minus 2p is the 2p. Oh, it's 2. Yes. OK. So those are equal. And that tells me that 9p is 6, and p is 2/3.

So it turned out that the best strategy for y is 2/3 on the first column which didn't look so promising, and 1/3 on the second column. And by creating that strategy, what do these numbers come out to be? And they're supposed to come out equal.

So 4p plus-- so 4 times 2/3 is 8/3, plus 1 minus p is 1/3 times-- plus 2/3 is 10/3. So I think that this is 10/3. And if that one isn't 10/3, I'm very surprised.

So that's 2/3-- help. Is it? 4p, 2p, 4/3. Yes, it's telling me they're the same. Did I set them equal? Yes. I set them equal to find p. So they have to be the same. Except for instructor mistakes, which never happen, I suppose. OK.

So those should be equal. Then x has no way to do better, choose these columns. But he can't choose those columns freely-- those rows freely because y could take advantage, unless-- so this would give the best strategy for q-- sorry-- the best q for x. Yes.

Do you see that picture now? There could be other-- it could be a much bigger matrix, of course. There could be other columns. Suppose there was a 0, 0 column. What difference-- what would be the effect on the optimal strategy of having that additional column, that additional option for y3? Well, he wouldn't take it.

So let's make it more tempting. 10, 10, 10. Oh, he would take that, right? Yes. That was not-- so I'm not sure what I am trying to say here. Yes.

There certainly could be-- there could be rows that-- or columns, or rows for x, that don't enter the mixed, the optimal mixed strategy. A mixed strategy is some combination of strategies, some combination of pure strategies, like choose this column, choose this row. But some columns may not-- or rows-- may not show up in the best mixture. So I won't complete that partial thought there.

So do you see that we have a duality theorem? First of all, you could write that as a linear program. The unknown is p for x-- for y. The unknown is q for x. So, actually, you just have one unknown in this simple problem, for this small problem.

But you have a minimization, a maximization. They meet at the optimum. And the duality theorem becomes the theorem for two-person games. Linear programming matches two-person games.

The point about two persons is very important. Three-person games are incredibly complicated. No theory at this simple level would solve three-person games. So that's where

John Nash's Nobel Prize comes into the picture. So his Nobel Prize was in economics because of the wide applications, but he was able to analyze the problem of-- and for functions more than for matrices-- of a three-person or n-person game.

So you know the story of John Nash? The book *A Beautiful Mind* and the movie *A Beautiful Mind.* So it's one of those movies that involves MIT because he was here. But you can't recognize anything about MIT in the movie. It's like *Good Will Hunting.* You're maybe in some basement of some remote building. But anyway.

It was of course a tragic-- tragic, and then cheer, then wonderful, and then tragic again for John Nash. He had-- so when I met him, he was going to teach linear algebra, one section, but that never developed. That was when he was-- his mental state was going downhill. And he moved to Princeton and just stayed. And then the wonderful thing was that he improved, and then the very sad thing was his death in a car accident on the way home from the Nobel Prize. So quite a story, an amazing story. Yes.

So that's specific optimization problems, linear programming and two-person games. And I hope that Friday it will be Professor Sra. Anyway, the lecture will certainly be about stochastic gradient descent. Good. Thanks.