02:29
Good evening, and welcome to lecture four.
02:33
We are actually almost done with the math for machine
02:37
learning that we need for us to be able to do
02:41
NLP. For example, for today, we are completely going
02:44
to be done with linear models, so that's exciting.
02:48
And then there's one more lecture that's going to
02:53
focus on sort of foundations of ML. we will have a
02:58
lecture on feedforward neural nets. That will be
03:01
next week. And once that's over we will dive right
03:05
into NLP after that starting with word embeddings
03:10
second half of next week.
03:13
But for now we are going to do a bit
03:18
more math for machine learning and
03:22
so we can actually feel better
03:24
once we start doing NLP A
03:31
brief recap of what we talked about in the
03:35
last lecture. We talked about the softmax
03:39
function, which allows us to convert raw scores
03:42
from a linear classifier into probabilities.
03:46
So simply exponentiate the scores and normalize,
03:49
and that's the softmax. We talked about the
03:52
last function for the softmax. In general,
03:56
the last function here takes the form of, So for
04:00
each example, we compute the loss, L sub i,
04:03
which is trying to simply tell us how well the model
04:08
is doing in terms of predicting the observed labels.
04:13
And then we have a second term, which we said
04:16
that we need that term to make sure that while we
04:20
want to fit the data, the observed labels, we
04:23
do want to make sure that we do not overfit the
04:27
training data. So this is the second term here.
04:31
And in terms of the soft max loss, the exact form of
04:36
L sub i is the following, which is simply the negative
04:39
log probability of the observed label, given the
04:43
input and the current setting of the parameters w.
04:49
All right, so now that we have the loss function,
04:52
we are now going to think about how to find
04:56
the best setting of W that minimizes the loss.
05:04
And so that's the plan for today's lecture.
05:08
We will begin with not the math itself,
05:12
but we'll begin with an intuition,
05:14
where we will think of the loss as
05:18
defining a landscape of other parameters.
05:22

And so within that analogy, we'll think about
05:27
what it means to move downhill in the landscape.
05:31
All right, so once we have the intuition
05:33
where we think about something
05:35
that we're familiar with and we know,
05:37
we'll move on to thinking about, okay,
05:41
when we are moving downhill,
05:43
how do we best move to the lowest point of the landscape
05:48
of the last? And for that, we will need gradients.
05:52
We'll begin by first computing gradients numerically.
05:58
This is kind of an approximation of what
06:03
we actually want of the derivatives,
06:05
the way we are going to compute them.
06:06
But you'll see that it's useful to conceptually
06:09
think about these numerical gradients.
06:12
But we are also going to see that when we have
06:16
large models, numerical gradients don't scale.
06:18
So what we actually want are analytic
06:21
gradients. So we are going to work out the
06:24
analytic gradient for the softmax loss.
06:26
And we'll look at a concrete example in
06:29
text classification to see how the analytic
06:33
gradient for the softmax loss actually
06:35
works in the setting of a concrete example.
06:41
And once we go through all that, we'll have everything
06:45
we need to define the workhorse of deep learning,
06:48
gradient descent. and gradient design at our
06:51
point will basically be putting everything together
06:54
and kind of understanding how everything fits in
06:58
so gradient design as we can see, depends
07:01
on gradients computing gradients and
07:05
you'll notice that in PA1 for the rest of this
07:08
course, we are not going to be literally sitting
07:09
there doing calculus, computing the gradient,
07:12
we are actually going to have frameworks like
07:16
PyTorch compute that for us And the way to do it
07:19
is by using an algorithm called backpropagation.
07:22
So it'll do the calculus for us, but it's
07:26
useful to know sort of what exactly it is doing.
07:30
All right.
07:33
Let's begin with the intuition.
07:37
Here we want to think of the loss as
07:40
defining a landscape over the parameters.
07:44
So this is some landscape where there
07:47

are hills and valleys in this landscape.
07:51
And we can think of, so when it's a hill, it
07:54
means that the last valley is really high. So the
07:57
parameter setting at that point is not great.
08:01
Whereas when it's a valley,
08:03
then it means that the last function is low. So
08:06
the parameter setting at that point is great.
08:08
And so by moving through this landscape,
08:10
we are looking for the valley.
08:12
Ideally, the global minimum.
08:16
So we are looking for the value. So we are just
08:18
going to be thinking about how to find a solution.
08:22
How do we find a solution in this landscape?
08:24
Try to find the lowest point in such a landscape.
08:32
We are not going to be working in the landscape.
08:37
We are actually going to be working
08:39
with mathematical functions.
08:40
And here, let's think about it in terms
08:44
of exactly that. Here, I am giving you a
08:46
simple function with just two parameters,
08:48
W1 and W2. And suppose that our loss function is simply
08:53
W1 squared and W2 squared. So what we have here
08:57
is we are seeing for a particular setting of W1 and
09:02
W2, we have some value. For example, here, the value of
09:05
the loss is really high. and what we are looking for
09:09
is the setting of w1 and w2 where the loss is low.
09:14
Here's an even simpler function, just a single
09:18
parameter. So here it's even easier to see.
09:21
So where the loss is simply w squared and
09:25
we are looking for the setting where the loss
09:28
is the lowest. In this case, we can just
09:30
eyeball it and see that it's when w is zero.
09:34
Okay, great.
09:37
Now, let's think about a solution, how we would
09:40
actually go about finding a valley in this space here.
09:45
And here is one solution.
09:47
This is just a thought experiment.
09:49
We're not actually going to do this.
09:51
Okay.
09:52
So this thought experiment says, how about we
09:56
just randomly jump around? We jump around this
09:59
landscape. And then all we have, though, is an
10:03
altitude sensor. So when we are at a particular
10:06
point, we know the value, how high we are.
10:10

We know the value of the loss.
10:12
And so what we can do is randomly jump around
10:16
and then compute the loss. And then when we
10:20
have jumped around enough, we are saying
10:22
this particular point gave us the lowest loss.
10:26
As you can imagine, this is not going to be great in
10:29
terms of number one, it's going to take too long,
10:32
especially in high dimensions there are just too many
10:34
numbers to tweak and also it's not guaranteed to find
10:39
you the lowest point in the landscape all right so
10:44
we're not going to do that here is a much better strategy
10:50
which is iterative improvement and what we do here
10:54
is to say okay finding the best way globally is a
11:01
really, really hard problem. So what we are going to do
11:04
is we are going to do iterative search where we start
11:09
at some position. We start with some setting of W.
11:12
And then from that setting, we are going to
11:16
refine our parameters just a tiny bit, tiny bit,
11:20
multiple times iteratively.
11:22
So we initialize that view randomly.
11:25
And we are going to update W so that
11:28
every time we update it, we actually
11:30
improve our setting. We reduce the loss.
11:37
Okay, so we want to, every time we move,
11:39
we want to make an improvement. We actually
11:41
don't want to make the situation worse.
11:43
So we have now a key question, which is that,
11:46
given the current setting of W, how do we know where
11:49
to go? How do we know to change the parameters
11:51
so that we are actually improving our situation?
11:57
okay we want to move such that
11:59
we want to reduce the loss
12:03
and here's one way to think about
12:05
what we are actually going to do
12:08
back to the landscape analogy what we're going
12:12
to do is we're going to start at some position
12:15
and then we're going to look around we're
12:18
going to look around and say which way is uphill
12:22
So if we are standing somewhere, we are just going to
12:26
kind of fill the ground under our feet and see, oh,
12:29
this is kind of going down. This is kind of going up.
12:33
Mathematically, this is equivalent
12:35
to computing the gradient.
12:37
The gradient is actually going
12:38

to tell us which way is uphill.
12:43
And notice that when you actually just do that,
12:46
kind of try to fill your feet under at the ground,
12:48
the ground at our feet, you are just really making
12:51
a local decision. It's very local. You are not
12:54
going to make a huge decision because all you know
12:56
about by doing that is just really here locally.
12:59
So we look around, and then once we look
13:02
around and figure out which way is uphill, we
13:04
take a small step in the opposite direction.
13:07
So the gradient tells us the direction of
13:11
where the function is increasing. So we're going
13:12
to move in the opposite direction of that.
13:15
So we're going to move downhill
13:16
because we are minimizing the loss.
13:19
And the step side, how far we move, is
13:23
defined by what is called a learning rate.
13:25
We'll return to that later.
13:28
And so we are going to do that multiple times.
13:31
Every time we move, we reevaluate. We again feel
13:34
the ground and our feet and then decide which
13:37
way is uphill, move in the opposite direction.
13:39
And so at each time for the last function, we are
13:44
recomputing the gradient at each point. and figure
13:46
out which way we are going to move at a time step.
13:53
If we do this enough times, we'll
13:56
end up with a good solution.
13:58
We'll return to what it means by enough times later.
14:03
All right, if we return to this simple function here,
14:09
the 1D function here,
14:12
what we are seeing here is that, OK, so the slope
14:15
here, in one dimension, the gradient is the slope.
14:19
So the slope indicates the direction and rate of
14:22
change of the function. So if we look at this point
14:24
here, for example, we draw the tangent line, and we
14:28
are seeing that this line here has a positive slope.
14:32
So if we are here, we are going to move
14:35
in the opposite direction of that slope,
14:38
meaning that we are going to
14:40
actually decrease the value.
14:43
So the slope is positive. We're going to move in the
14:45
opposite direction. So we're going to decrease the value
14:47
of W. So that's great because the minimum is here.
14:51
And the slope tells us to move this way,
14:55

right?
14:58
We're moving in the opposite direction of the slope.
15:02
Let's look at that when the slope is negative.
15:07
So here we see that the slope is negative.
15:11
that means we are going to move in the opposite
15:13
direction of that so we are going to increase the value
15:17
of w so if we increase the value of w when we're
15:20
here it means we are moving in this direction so that's
15:24
great because again the minimum is this direction
15:32
so that's the sort of very simple function when
15:38
we are dealing with high dimensional functions
15:40
but it's exactly the same thing happening.
15:44
Okay, so I kind of talked through this, but let's take
15:49
a minute just to think about it together. So I'm
15:52
going to give you two minutes just to think about it.
17:35
15 more seconds to think about it.
18:09
Okay, great.
18:11
So since I talked about this already,
18:13
I won't kind of call out people
18:17
because I just wanted you to think about it
18:20
yourself and notice that we are going to be
18:25
moving left. So opposite direction of the slope.
18:30
So slope is positive, we decrease the parameter.
18:34
Slope is negative, we increase the parameter. Move
18:38
in the opposite direction of the slope or in high
18:41
dimensions in the opposite direction of the gradient.
18:44
All right, great.
18:47
so we need the gradient to figure out
18:50
where we need to go now let's look at how
18:55
to compute the gradients numerically so
19:03
in one dimension so we have the slope right and the
19:08
derivative measures the slope of a function and here
19:13
we can use this definition of the derivative which is
19:18
the derivative of f of x with respect to
19:21
x is the limit as h tends to 0 as of f
19:27
of x plus h minus f of x divided by h.
19:31
So if we're at a particular setting of our parameter,
19:35
we are going to evaluate the loss at that point
19:40
and then evaluate the loss again at that point
19:43
plus a tiny bit and then divide by that tiny bit.
19:50
So in practice, the way we are going to compute this
19:53
is by using a small but finite h. And so this is why
19:58
this is called the finite differences approach because
20:01
we are just going to set h to a small number and
20:04

then we can actually compute the derivative that way.
20:10
In multiple dimensions, it's a similar thing.
20:14
Except that we're now going to have
20:16
not a single value, but we're going to
20:20
have a vector of partial derivatives.
20:24
And that gradient vector is going to give us
20:28
the direction where the function is increasing.
20:31
And we're going to want to move in the opposite
20:33
direction of that. And the gradient's
20:36
magnitude tells us the steepness of the function,
20:41
how fast the function is actually increasing.
20:45
Let's work through an example of computing the
20:51
numerical gradient so you can better appreciate how
20:55
this is all working in high dimensions and why
20:59
maybe we don't want to actually use a numerical
21:02
gradient when we are working with very big models.
21:09
Suppose we have a current setting of W This is it
21:14
And we compute We evaluate the loss for this
21:18
setting We're saying, okay, here we are We are here
21:21
right now This is our setting And this is the loss
21:26
Now we are asking What is the gradient?
21:29
This is going to be this vector, right? And we need
21:34
the gradient for each of these dimensions. We need to
21:38
compute the slope for each of these dimensions, right?
21:43
And this is how we're going to
21:45
do it, one dimension at a time.
21:48
So to use finite differences, so first we add a tiny
21:52
bit to the first dimension because that's what we are
21:54
looking at right now. And then we reevaluate the last.
21:59
Okay?
22:01
And now if we look at the last,
22:03
look at the last three the last two digits this one
22:07
is 47 this one is 22 so the last actually went down
22:13
when we increased the value in the first dimension
22:18
so the last went down and now
22:22
what we are interested in is computing the value
22:27
for the gradient in the first dimension there
22:30
so what do we do? We use this definition that we
22:33
looked at. So it's the value we get after we
22:38
increased the first dimension here by a tiny bit
22:42
minus the value before increasing divided by H.
22:49
And here it works out to be minus 2.5.
22:57
So we have a negative value in this dimension,
23:04
which means that we want to move in the opposite
23:08

direction of that. And by that, it means we
23:11
want to increase the value. And if we increase the
23:14
value, we indeed saw that the loss went down.
23:25
Gradient is negative. We're going to
23:27
move in the opposite direction of that.
23:31
And that means increasing that, that value.
23:38
And that's all we are doing. We are actually
23:42
improving our situation by doing that.
23:49
That's just one dimension, though.
23:51
We have to keep going, second dimension.
23:56
We add h to that value.
24:00
we compute the loss
24:02
compute the gradient
24:05
so here we are noticing that for that value the
24:10
last three digits are 53 as opposed to 47 so the loss
24:15
went up by doing that so here the gradient is
24:20
positive when you compute it it's positive so we want
24:24
to move in the opposite direction of that and And
24:27
so that makes sense because we saw that when we
24:30
actually increased that value, it went up, the loss
24:33
went up. So the gradient is telling us the right
24:36
information that we've got to decrease that value.
24:47
All right, let's look at one last dimension.
24:49
This is the third dimension. We increase the
24:54
value for that dimension. We compute the loss.
24:57
And here we are seeing that the last two
24:59
digits are exactly the same. Nothing changed.
25:02
And if we compute here this dimension in the
25:06
gradient, we see that the gradient is zero.
25:11
The gradient is zero means that
25:14
we are not going to do anything.
25:16
We're not actually changing that value. Changing
25:19
it doesn't really improve the situation or make
25:22
it worse, so we are not going to change that.
25:27
Right, so this is the numerical gradient. and
25:29
we actually have to repeat that for all the
25:32
dimensions one by one doing this operation.
25:39
So you can see now why numerical gradients don't scale,
25:44
because when we have billions of parameters in
25:48
models, this is going to be painfully expensive, and
25:53
it's going to take too long. Also, this is simply an
25:56
approximation because we are using this finite H, So,
26:00
typically, we do not want to use numerical gradients.
26:06
All right. So, instead, we're going to use calculus
26:09

to actually work out the analytic gradient.
26:13
We have our last function. It's just a function, just
26:17
a mathematical function. We are going to compute the
26:20
derivative of that function. And then, once we have
26:24
that, we can actually compute the gradient at once.
26:28
the entire vector once we have worked out the math.
26:38
But maybe we can think a little bit about numerical
26:41
gradients here before we move on to analytic gradients.
26:44
So I'm going to give you maybe two
26:47
minutes for this one to think about it.
28:13
Okay, 30 more seconds to think about it.
29:15
Anyone want to share what they're thinking?
29:18
What do you have? Yeah,
29:32
right.
29:33
Yeah, I thought, yeah. Yeah, so he
29:35
said the answer is P, which is correct.
29:37
So if you decide to be brave and implement
29:44
gradient descent by hand, which I completely
29:47
recommend, especially for this linear model,
29:52
you're going to sit down, use
29:54
calculus to compute the derivative.
29:56
But you might have made a mistake somewhere. So to
30:00
check, to verify that your solution is correct, you can
30:03
compare against the numerical gradient and then once
30:08
you actually verify that your solution works throw
30:11
away the numerical gradient because it's too slow and
30:14
then proceed with your solution in PA1 and the rest
30:18
of the course we're not going to be writing down the
30:23
gradients manually we're going to
30:26
have a backprop do it for us in PyTorch
30:32
all right all right moving on to analytic gradients
30:39
so we're going to look at the analytic gradient
30:42
for the soft max loss and we want to compute the
30:47
gradient of the loss with respect to these parameters
30:50
w and recall that this is the soft max loss,
30:57
which is the negative log probability of the target
31:01
label given the impact and the current setting.
31:08
Okay, we are going to massage this loss here to
31:14
get to a form here of that loss that's easy to work
31:20
with when we are actually computing derivatives.
31:23
and so we're not changing the laws at all we can
31:27
walk through it briefly but we're not going to spend
31:31
too much time on it if you want to kind of get
31:34
every step you can look at it at home but here's
31:37

the process okay so we have the log probability of
31:42
the target label right and that we know is simply
31:48
we take that our project
31:52
between the vector corresponding to the row
32:00
for the target class. So this is a vector of
32:03
weights corresponding to the target class.
32:06
And then we take the dot product with the input
32:09
representation. So this is simply the score for the target
32:12
class. That's what we want, the probability of the
32:15
target class. So this is the score of the target class.
32:17
we apply the softmax by exponentiating and then
32:20
normalize over all the classes so we just replaced that
32:25
probability with the softmax wrote out the softmax and
32:30
now we are redistributing the log here so log of this
32:38
numerator here is so the log and the exponential
32:42
cancel out so we end up with this term
32:45
here we just grab that Let me bring it here.
32:49
And then log of a over b is log of a minus b. So
33:00
that's why we now have this expression here. So the
33:04
log of that part minus the log of this denominator.
33:11
And so, yeah, that's what we end up with.
33:14
We simply took the softmax, redistributed
33:18
the log, and we end up with this expression.
33:21
So we are going to compute the
33:24
derivative of this expression here.
33:27
All
33:35
right, so this expression is nice. It's made out of
33:38
simple pieces, and we know how to compute the derivative
33:41
of those simple pieces, like dot products, exponentials,
33:44
log, and so on. So what we are going to do is we
33:47
are going to apply the chain rule of calculus to these
33:51
pieces to compute the derivative of the entire thing.
33:59
And for that, we are going to look at
34:01
just the loss for a single example.
34:05
So this is the loss of a single
34:07
example. So we remove the sum over here.
34:11
and
34:13
so this is the negative so negative log
34:18
probability and we just did the massaging right
34:22
and now we can differentiate the first term
34:27
in the first term here we are
34:32
we are going to begin by first differentiating with
34:37
respect to the parameters of the target class w sub yi.
34:43
And for that, from calculus,
34:47

it follows that the derivative of that is simply xi.
34:54
Right? You see that?
35:10
So it's the same way that the derivative
35:12
of 2x is simply 2. So here we are seeing
35:18
the derivative of this expression with respect
35:21
to that is simply this part here, xi.
35:27
Next, we are going to differentiate the second term.
35:32
And the second term is this here,
35:36
the log of this expression here.
35:39
And for that, we are going to apply the chain rule.
35:42
And first of all, we need log of something,
35:48
where something is this whole expression.
35:50
So the derivative of log of z is 1 over z.
35:56
So 1 over z.
35:59
So we're going to end up with
36:01
something like 1 over this whole thing.
36:06
So that's what we have,
36:08
1 over this whole expression here.
36:13
And now we apply the chain rule to compute
36:18
the derivative of this bottom part here,
36:23
which is here. And so we're going to work that out.
36:30
One key observation here that's going to
36:33
help us is that this part here only depends
36:38
on the variable that we're looking at here,
36:41
which is this w sub yi, when y prime is yi.
36:48
So we can throw out all the other terms because they
36:51
become constants. They do not depend on w sub yi.
36:59
And so that becomes, we're simply going to
37:02
end up with the exponential of this expression
37:09
when W is W sub YI, the dot product.
37:15
And then, so this is the derivative of this
37:22
thing here. And then we again apply the chain
37:26
rule. So we want the derivative of this thing
37:28
here, which is XI as it worked out before.
37:34
Any questions about this one?
37:44
Great.
37:45
Now let's combine the results. here. The
37:49
first part, remember we had a minus, so we're
37:51
going to bring that back. So it was the
37:53
negative log, right? So we're going to bring that
37:55
back. So the first part becomes minus xi.
38:03
And then the second part here had a negative.
38:06
So we are going to actually, now it becomes a
38:10
positive. and this is what we have and you will
38:15

realize that this is just the softmax up here
38:19
and so if we replace that with the softmax expressing
38:25
the probability of the target label right here so
38:29
we have something very nice and simple which is minus
38:32
xi plus the probability of the target label times xi
38:40
and so this has a very nice interpretation
38:43
which is that for the core class this
38:47
is the class with derivative we just
38:49
computed
38:50
for the core class what we want to do with the
38:53
weights if we look at what the gradient is doing
38:55
we want to pull the weights towards the input
38:59
we want the weights to look very similar to the input
39:04
and then we are going to decrease them we are going
39:07
to kind of make them look less like the input,
39:10
proportional to how confident the model already is.
39:16
And we haven't worked out the gradient
39:19
for the non-target classes, but it works
39:22
out that it's simply this part here,
39:27
without the first term. It's similar without the
39:29
first step. You can work it out at home. And so the
39:33
interpretation for that is that for the non-target
39:35
classes, we are going to make the weights of
39:38
those non-target classes look less like the example
39:44
that we are looking at because it's not from that.
39:49
So this is actually kind of nice because remember
39:53
when we were looking at the weights and I said the
39:56
weights of these classes, sort of the rows correspond
39:59
to the classes, and we can kind of see that those rows
40:03
looked similar to examples from the target class.
40:06
So they were kind of prototypical representations
40:10
or examples of the data from the target class.
40:17
But this is not surprising, because when we are
40:21
updating, according to this update rule, actually the
40:24
ways are pulled to us the examples. We are kind of
40:28
creating this clustered representation. a mishmash
40:32
of the examples becomes the weight for that class
40:41
so we are going to actually think about this
40:44
this is a nice intuition so let's think about it
40:48
together for a sec so this kind of requires a bit of
40:51
thinking so I'm going to give maybe three minutes 30
43:46
more seconds
44:39
anyone want to share what they're thinking yeah
44:53
He's
45:04

agreeing to go in the other direction of x of
45:08
the influence of y, so probably he pushes away.
45:14
He pushes away.
45:17
I think your reasoning is right.
45:21
Your reasoning is kind of going in the right
45:24
direction. I'm just not sure where it ended up with.
45:28
But, yeah, you're thinking in the right way.
45:34
Let's see what's actually happening. Okay, so the
45:37
model assigns nearly probability zero to the
45:41
correct label, right? This is very low probability.
45:44
So essentially this term is kind of gone.
45:50
So this term is gone, and we are left with minus xi.
45:56
So we are going to move in the opposite direction
46:00
of that. so we are actually going to make the
46:05
weights to look more like xi because this is
46:10
negative and so when we move the opposite it's
46:13
positive so we're going to make it positive we're
46:15
going to add xi to the weights essentially so
46:25
the answer here is b it pulls w i this towards xi so
46:31
it's just building a representation in this weight
46:36
that Carlos just says, look more like the
46:39
examples, look more like examples from that class.
46:43
All right, so here there's one contrast
46:49
here, just to sort of complete the thought.
46:54
So it's similar, but also still requires
46:58
some thoughts. So I'm going to give you maybe,
47:01
let's say, two minutes or so for that.
49:10
30 more seconds.
49:46
Okay, great.
49:48
Does anyone want to share what they're thinking?
50:19
Yeah.
50:20
Does anyone want to share their thinking?
50:41
Like with the negative XI,
50:45
it's interesting because the trigon is
50:47
large, so there has to be a failing date.
50:49
In the second case,
50:51
as the conditional probability is close to 1,
50:54
the trigon is near 0,
50:56
which means it's at a minimum.
50:59
It doesn't require enough data.
51:02
Yeah, excellent.
51:03
Yeah, that's a great answer. great explanation so
51:08
indeed for case one we're going we just saw that in the
51:13
previous slide we're going to actually make an update
51:15

and it's going to be a large update in this case
51:18
because the model is super wrong so this term although
51:21
this term is kind of zero this one is large but then
51:26
for the case of when the model is correct this is
51:31
one so we have minus x plus x so Basically, the gradient
51:36
is flat, so we are not really making an update
51:40
there. The model is already getting the example right.
51:47
Yeah, sure.
51:54
This one?
51:56
Last slide? Okay.
52:20
W is 2, XI is 1, then W plus XI is 3, then W is 2.
52:37
So, remember that here, this is a vector,
52:42
right?
52:44
And so, this vector is like saying that, okay, so
52:47
we take this vector, and then we make it look
52:52
more like the vector, the input vector, the feature
52:55
vector. So, in other words, we are saying, okay,
52:58
suppose the feature vector is quite positive in
53:01
the first dimension. we're going to make the first
53:04
dimension of the weights also quite positive.
53:07
And then if the feature is negative in the second
53:10
dimension, we're also going to make it a little
53:12
bit negative. So basically, we are making the
53:15
weight vectors behave in a similar way to the
53:20
feature vector of the example we are looking at.
53:35
Yeah, so the gradient, so maybe let's think about it.
53:42
So the gradient is really just telling us the direction
53:46
in which we should move so that the loss decreases.
53:52
And so this is what the gradient is, and we're just
53:56
trying to interpret exactly what it's doing,
53:59
right? So what is actually happening. And we're saying
54:02
intuitively, this is sort of what's happening.
54:06
And we saw that when we were eyeballing the
54:09
weights of the classifier we saw earlier with spam.
54:17
And when you're doing something like
54:20
computer vision, it's even more visible.
54:24
because when you actually visualize the weights,
54:27
like for the cat class, for example,
54:29
the weights kind of look like a blurry cat, the
54:32
weights that correspond to the cat class and so on.
54:34
So it is actually a thing that's happening in there
54:38
where the weights are kind
54:41
of for a linear classifier,
54:46
you're actually getting sort of
54:48

a prototypical representation
54:49
of your input for each of the classes
54:53
in the weights
54:56
once you visualize them
55:05
you can kind of try this at home by implementing
55:09
a simple classifier and eyeballing the features or
55:13
even maybe you could try it a little bit for PA1
55:19
yeah it might work it might sort of help
55:24
you answer that that's a good question
55:29
Yeah, call me some questions.
55:32
Right, okay, great.
55:35
All right, so yeah, so essentially the
55:38
linear classifier here is learning,
55:41
in some sense, the intuition
55:43
is sort of learning templates.
55:45
And gradient descent, as we
55:48
saw from the derivation there,
55:51
is sort of making each class weight look more like
55:54
the training examples of that class. um is this statement
55:59
uh universally true when we are training models
56:02
um so here in the linear model this is kind of clear
56:08
because the um the the the linear model the scores
56:14
are actually working directly with the the feature
56:17
absentation so we compute the score by saying it's the
56:20
product between the rows of W and the actual input.
56:24
But in a neural network, for example,
56:28
the linear classifier is at the top of the network.
56:31
So the linear classifier is
56:34
really doing a product between
56:37
the rows of W and some function of
56:42
the input. And that function is basically a bunch of
56:46
layers of the network. So the input goes through the
56:49
layers of the network, and once it comes out, we
56:52
get a richer representation of the input, and the linear
56:56
classifier works with that. And unfortunately, we
56:59
don't really know exactly what the network did with
57:02
the inputs to generate that richer representation,
57:05
so there's not that nice interpretability that
57:09
you get from the linear classifier, but here,
57:12
the weights are directly operating on the
57:15
feature representations of the input directly.
57:20
And so we can kind of have
57:22
that layer of interpretability.
57:28
All right.
57:32

Okay, let's make this more concrete by going
57:35
back to our domain of text. So here we
57:38
have a text document in text classification,
57:42
and the document says, too many
57:43
drug trials, too few patients.
57:46
And we want to categorize this text into one of
57:50
these categories, health, sports, and science.
57:55
And suppose that our feature representation
57:58
is something like a bag -of-words representation,
58:00
but we only really have three features.
58:05
Whether or not the text mentions the word drug,
58:09
and here it does once. so we have one here and
58:14
without not it mentions patients, it does, one here
58:17
and baseball, that feature is absent, so we have a
58:21
zero this is a feature vector we throw away the
58:24
text, we now have a numerical representation of
58:27
that text that's what we always do when we are dealing
58:30
with machine learning in text we need a numerical
58:33
representation alright, so here we have that
58:37
and now
58:39
if you go back to the gradient. So the
58:41
gradient for the correct class, we
58:43
worked that out. This is this expression.
58:45
If you work out the gradient for the non
58:49
-target class, it is simply this expression.
58:52
So that's the second half of this.
58:55
Now,
58:58
suppose that we computed the probability
59:02
of the various labels, health, science,
59:08
and sports, and this is what came out.
59:11
Just for the sake of our example. So
59:16
we computed the scores, we normalized,
59:18
and this is what came out.
59:23
Now, according to this gradient here, the way
59:29
we are going to update the weights is as follows.
59:32
So the gradient here, with respect to the
59:38
health parameters is going to be minus Xi. So this
59:44
is Xi. This is our representation for the
59:46
document plus 0.2 times Xi. So this is this one.
59:52
And then for sports, we grab this expression
59:57
here. So it's 0.5 times the feature vector Xi
1:00:05
for science, 0.3 times the feature vector Xi.
1:00:11
And so now we are going to grab these gradients,
1:00:15
and we are going to update the parameters corresponding
1:00:19

to health by moving in the opposite direction of
1:00:23
the gradient, so plus the example. So we are
1:00:26
basically adding this example to the weights and then
1:00:30
adding 0.2 and subtracting 0.2 times the thing because
1:00:36
we are moving in the opposite direction. And then
1:00:38
for sports, we are saying we are going to move in
1:00:42
the opposite direction of that. So it's going to be
1:00:44
the weights for sports minus 0.5 times this example.
1:00:49
So we are making the parameters of these other
1:00:52
classes to be less like this example and in proportion
1:00:57
to how much probability mass the current weights
1:01:01
are actually putting on those non-target classes.
1:01:17
So that's what's happening there.
1:01:25
Yeah, so here I'm just sort of repeating that
1:01:29
interpretation, but I talked about it already.
1:01:32
Any questions or comments about this part here?
1:01:38
Okay, great.
1:01:40
Moving right along.
1:01:43
All right, so let's put everything
1:01:45
together that we have looked at so far.
1:01:47
So gradient descent is doing the following.
1:01:53
It's simply saying, okay, we do want to find
1:01:56
the minimum of the last function. We want to
1:02:00
find a good setting of our parameters that
1:02:03
minimizes the loss, which means that the
1:02:05
parameters that do well on the observed labels.
1:02:09
And the key idea for gradient descent is that, OK, it's
1:02:13
really hard to find the best setting, but we are
1:02:16
going to try our best by starting from an initial guess.
1:02:19
And then we are going to compute the gradient of the
1:02:24
loss with respect to the parameters. And we are going
1:02:26
to make tiny steps in the direction of the steepest
1:02:31
descent, so in the opposite direction of the gradient.
1:02:33
So if I'm starting here, I have randomly
1:02:38
initialized my parameters to some setting.
1:02:42
I then get the gradient. I move in the opposite
1:02:48
direction. At first, I'm making these giant steps
1:02:52
probably, But then as I get better and better, I'm
1:02:55
making smaller and smaller steps towards the minimum.
1:03:01
And so I repeat this multiple times until I get
1:03:05
to a good solution. Typically, in practical terms,
1:03:09
that means one or two things. I can decide to stop
1:03:13
one's performance on my validation set plateaus.
1:03:17
I no longer get improvements in my performance.
1:03:21

or what we are doing in this course typically we
1:03:26
are going to look at epochs sort of how many epochs
1:03:33
are we training our model so an epoch usually means
1:03:37
how basically going through our entire data set so
1:03:42
one epoch is making stats over the entire data set so
1:03:48
typically something like six epochs then you're done.
1:03:51
Sometimes if you do too much you might end
1:03:55
up overfitting to the training data so
1:03:57
perhaps six epochs or something like that.
1:04:02
Let me talk about so for language models the current
1:04:07
models are trained on such huge data sets such that
1:04:10
really they're all trained on just one epoch of the data.
1:04:14
Just one epoch. So we go through the data just once.
1:04:20
all right so gradient descent we said we are going
1:04:26
to essentially go over the entire data set compute the
1:04:31
gradient with respect of the loss but notice that
1:04:35
we are really so for each point we have to make a
1:04:39
forward pass through the model which means that we have
1:04:42
to compute the prediction from the model. So this is
1:04:45
the function here. And if your model is large, this
1:04:50
is going to cost you. This is going to be expensive.
1:04:53
So typically, we don't want to compute the gradients
1:05:01
by summing up over all the data set. Typically, we
1:05:05
want to just make an approximation by taking a small
1:05:08
part of our data set, a batch. We work in batches.
1:05:11
And so
1:05:12
common sizes include things like 256, 128, and so on.
1:05:19
Yeah, so this is going to be expensive
1:05:23
because not only do you have to compute the forward
1:05:28
path, but because we're computing the gradient,
1:05:31
we're actually going to be doing the backward path. So
1:05:35
this is going to be too expensive if you're making
1:05:41
if you have a huge data set. So you
1:05:43
want to use batches. And this is what
1:05:46
is called stochastic gradient descent,
1:05:49
where you have some randomness coming from the
1:05:53
fact that you are approximating this gradient over
1:05:57
a small part of your data set at every time step.
1:06:03
So this is what you might see in practice, where so
1:06:08
you are going to be sampling batches from your data
1:06:12
set and then you update the weights by simply
1:06:18
evaluating the gradient on just that small batch as
1:06:21
opposed to the whole data set and then you update the
1:06:24
weights in the opposite direction of the gradient.
1:06:29

So this check-in, I kind of already talked about it so
1:06:32
I'm just going to maybe read it out and then tell you
1:06:36
the answer, which is that computing the full gradient
1:06:39
is expensive because it's not because of A, which is
1:06:44
applying calculus. The math itself is not the problem.
1:06:47
The problem is that for every data point, we have
1:06:52
to actually perform inference for the whole model.
1:06:56
So we are essentially, if you have a large model,
1:06:59
we have to put the entire example through
1:07:01
all these layers. This is going to be doing
1:07:04
all that math. all the dot products,
1:07:06
everything you need to make the forward pass.
1:07:10
And that is going to be expensive, doing all
1:07:13
that compute for all these examples just to make
1:07:16
one step. And you're going to be making many of
1:07:18
these steps, so you want to avoid doing that.
1:07:23
All right.
1:07:26
All right, so let me talk about
1:07:28
backpropagation here just briefly.
1:07:32
So gradient descent wants gradients. So, we
1:07:37
need gradients to figure out where to go,
1:07:40
how to update the laws at every time step.
1:07:44
We can write down, to get those gradients,
1:07:49
we can use calculus and write it down.
1:07:52
But as I mentioned already,
1:07:57
backpropagation is going to do it for us.
1:08:00
is an algorithm for computing gradients
1:08:08
for complex functions,
1:08:10
any function you can think of that's
1:08:12
differentiable, based on the chain rule of calculus.
1:08:16
So it's actually really nice and beautiful,
1:08:21
but for this course, maybe we don't need to do that
1:08:25
to look at it, but if you are doing a machine
1:08:28
learning course, They will probably go through it.
1:08:30
And it simply reuses sort of intermediate
1:08:33
computations and does little local
1:08:37
computations and then adds them together.
1:08:41
And so it's very beautiful, but
1:08:44
we're not going to look at it.
1:08:47
PyDoch implements it, and so we're going to be
1:08:50
using what is called autograd, which is sort of we
1:08:54
are getting automatic differentiation from these from
1:08:58
these frameworks all right so let me so this is
1:09:06
great we have pretty much finished text classification
1:09:09

with Lydia models and so here's the big picture
1:09:13
of what we talked about when we get our input
1:09:17
we want to represent it as numbers and here we have
1:09:21
looked at very simple representations, and later
1:09:27
we'll look at more sophisticated representations.
1:09:30
And for a linear model, we're going to score each class
1:09:35
by simply saying we do this linear operation, the
1:09:41
product of the correct rows with the product of these
1:09:47
rows with the feature vector. And then we convert those
1:09:51
scores to probabilities using the softmax function.
1:09:55
So that's great. And we pick basically
1:09:58
the class with the highest probability.
1:10:00
Good.
1:10:02
Feature vector, the simple linear model, we
1:10:05
just put it through the softmax, we pick the
1:10:07
highest probability class, and then we're done.
1:10:10
And in addition to the parameters which we
1:10:13
pick using gradient descent. We also have
1:10:18
hyperparameters like the learning rates,
1:10:22
the batch size,
1:10:24
number of iterations,
1:10:26
how many times we actually want to do gradient
1:10:29
updates, and so on. And for these ones, we talked about
1:10:34
the fact that in all of machine learning, there's
1:10:36
a rule that we pick these on the validation set,
1:10:40
not on the test and not on the training data set.
1:10:46
All right, so we're done with linear classifiers.
1:10:50
What are their strengths and limitations?
1:10:52
The linear classifier is good because it's simple.
1:10:59
It is very fast. So, for example, for PA1,
1:11:09
you'll find that if you do a simple linear classifier,
1:11:13
You can run it very fast on your laptop.
1:11:18
And one thing that people like a lot about linear
1:11:22
classifiers that you don't get in complex models
1:11:24
is interpretability. We kind of looked a
1:11:27
little bit at how this actually ends up being
1:11:29
interpretable. You have this direct interaction between
1:11:32
the weights and the feature representations,
1:11:35
whereas you don't have that in the complex
1:11:38
models. there's a giant path between the
1:11:41
input and the linear classifier in a feed
1:11:47
-forward or even a transfer of network.
1:11:51
The key limitation is that the decision boundary
1:11:54
in a linear model has to be this straight line.
1:11:58

Suppose, for example, you have this study here where
1:12:01
you have this green point and this red point. The
1:12:05
decision boundary in a linear classifier doesn't
1:12:07
actually allow us to get all of this correct. So the
1:12:10
green points here are now mislabeled as being red, but
1:12:15
it would be really nice if we can have a function
1:12:18
that is going to allow us to also get these points,
1:12:22
to get them to be correctly labeled.
1:12:25
And so what we are going to be doing next, we are
1:12:29
going to be introducing one class of models that are
1:12:33
more expressive and can actually model decision
1:12:38
boundaries that are more complex than a linear model.
1:12:42
So that's kind of what we're
1:12:46
going to be doing next week,
1:12:47
the feed-forward neural nets.
1:12:53
Yeah, so this is what we have for
1:12:55
today. Any questions or comments?
1:13:03
Awesome.
1:13:07
Thank you.