01:37
Good evening, and welcome to CSC 256.
01:43
This is lecture three, but it's
01:46
the first time we are meeting.
01:49
Sorry about that. For the first two lectures, we
01:52
uploaded slides, and we also put up a couple of videos.
01:58
We understand that there are some issues with the
02:03
second lecture, and we'll be addressing those issues in
02:08
this lecture covering some of the ground that we need
02:12
and some of the other background stuff you can
02:15
read on your own but the key concepts from lecture
02:18
two will be briefly recapped in this lecture.
02:23
All right.
02:24
The focus for today will be mostly on linear models.
02:28
In particular, we'll be focusing on defining a loss
02:33
function for the linear model we have introduced
02:37
in lecture two
02:42
before we dive into the content there's one
02:45
housekeeping matter here which is that we have
02:48
released PA1 it's on canvas if you haven't
02:52
already looked at it and it's due on January 30th
02:57
this here is the late submission policy.
03:03
It's the same policy we have in the syllabus.
03:06
So essentially you are allowed to submit up to
03:09
48 hours late, but with penalty, 15% first
03:14
day late and 30% second day late. After that we
03:18
are not able to accept any more submissions.
03:23
If you have looked at the assignment and you're
03:25
thinking that we haven't covered some of this yet
03:28
that's valid. But we will make sure that we'll
03:31
cover everything that you need before the deadline.
03:36
All right. Any quick administrative questions?
03:40
Any quick logistics -related questions?
03:45
Excellent. We just set up Piazza as well yesterday.
03:50
If you have any questions or quick things that come
03:53
up, you can post on Piazza. The TAs will be monitoring.
03:59
I will be monitoring less often the
04:02
TAs, but I will be monitoring as well.
04:05
And for submitting assignments,
04:08
we'll be using Gradescope.
04:09
That's been set up as well. You
04:12
can access it directly via Canvas.
04:14
Same for Piazza. You can
04:16
access it directly via Canvas.
04:23

So since we haven't met, and perhaps some of
04:27
you might not have looked at the slides yet,
04:29
just a one-slide review of what we have
04:32
done so far. we introduced one task.
04:36
Our very first NLP task is the
04:38
task of text classification.
04:40
This is a very common task in NLP. And often
04:45
when people talk about doing NLP in industry,
04:47
often it ends up that they are actually
04:50
just doing some instance of text
04:51
classification. So it's worth knowing about.
04:55
And so far I have introduced two very simple approaches
04:59
for addressing this task. We talked about K&N,
05:04
which is worth knowing about in the abstract, but these
05:08
days no one really uses it. And then I introduced
05:12
the linear classifier that we'll continue to talk
05:15
about today. And I'll also briefly recap it in more
05:18
detail so you know the context of what follows today.
05:24
All right.
05:25
So text classification.
05:28
This can be defined as we have a document,
05:32
and we simply want to assign that document
05:35
to one of many predefined categories.
05:38
So examples include like spam,
05:41
not spam, classification,
05:43
or if you are classifying news articles,
05:45
categories of news articles,
05:47
doing things like authorship attribution, figuring
05:51
out which author actually wrote an article, and so on.
05:55
And an important thing when we are doing NLP, some
05:59
of you might have already taken a machine learning
06:01
course, but here we are actually looking at problems
06:04
involving text, and whenever we are dealing with
06:06
problems involving text, we need to keep in mind that
06:11
machine learning models are mathematical objects.
06:14
We cannot simply directly feed the text to the model.
06:18
We need a numerical representation of our data.
06:22
and the simplest representation here is
06:25
what is called a bag of words representation
06:27
where we simply create this giant
06:30
whose dimensionality is the size of the vocabulary
06:33
and we put numbers along each dimension by
06:38
simply counting how many times we've seen that
06:40
word in the piece of text we're looking at
06:43

notice that the piece of text is often something
06:46
small like it can be a tweet or it can be
06:48
a news article So a lot of these dimensions
06:50
are going to be zeros, and so this is often a
06:53
very large, sparse, high-dimensional vector.
06:59
This is not the best representation
07:02
we can get, we can use.
07:04
And you will notice in PA1 that the representation
07:07
that you use really matters, because that's
07:09
all we have. Once we actually obtain a numerical
07:13
representation from the text, we throw away the
07:16
text. That's all we have. So that representation
07:18
really has to capture the semantics of our input.
07:22
For PA1, you'll notice that I've already given you the
07:26
solution that uses the bag -of-words representation,
07:29
and what you're going to do is to improve
07:31
on that by using a most sophisticated
07:33
representation that we haven't covered yet.
07:40
All right, so what I'm going to do now is I'm
07:42
going to spend the next few minutes recapping
07:45
some of what we talked about in lecture two.
07:49
If you haven't seen that in the
07:51
audio, there were some audio issues.
07:54
So the linear classifier is a very simple
07:57
model that simply says we're going to
08:01
assume a very specific form of the decision
08:04
boundary that separates our data points,
08:07
and that form is simply a straight line here.
08:13
So that decision boundary here is defined by a
08:18
small number of parameters and we're going to
08:21
figure out what those parameters should be so
08:24
we can separate our classes as best as we can.
08:30
And this is the notation that we're going to be
08:33
using, which is that a linear model here is going
08:37
to be a function f of the input features x
08:42
where x here is going to be our representation
08:45
whatever it might be, bag of words or something
08:47
else and it's also a function of the parameters w
08:52
and so
08:55
once we have a text we get a representation x we
08:59
throw away the text we feed that representation
09:01
through this function f along with the parameters
09:05
w and out comes numbers which are scores
09:09
giving each class the appropriate score, how likely
09:16

it is that the document belongs to that class.
09:21
So here, let's make this more concrete. So
09:24
we have, consider a 10-way classification
09:27
problem. We have 10 classes. And suppose that,
09:31
for some reason, we have 3,072 features.
09:37
and
09:39
here we're seeing here that a linear classifier
09:44
simply does the following it multiplies the
09:47
feature vector by the weight matrix, by the
09:51
parameter matrix w what we are doing is we're
09:55
essentially doing dot product and we'll see
09:57
precisely how we are doing that with an example
10:02
and so here what we have is output We
10:06
have a 10 by 1 vector, which is obtained
10:10
by doing exactly what I talked about.
10:14
So we have the parameter matrix W, which
10:18
is 10 by, so we have 10 classes by 3072.
10:24
This is the number of features we have.
10:26
And then we have the feature vector X.
10:30
So once we multiply that, what comes out is a vector.
10:36
where each element represents the
10:39
score for each of the classes.
10:42
So take a look at math, this simple math
10:46
here, and look carefully at the dimensions,
10:49
see how that works out.
10:59
Okay, great.
11:02
Usually there's also something else here that we
11:05
haven't mentioned, which is the bias term. And
11:08
the bias term is going to be a number associated
11:11
with each of the classes. So here, for example,
11:15
we have a vector which is 10 by 1 where each
11:19
element tells us the bias for each of the classes.
11:23
What the bias term is telling us is
11:26
that before even looking at the data,
11:31
there's some prior information that we need to
11:36
know about the likelihood of these classes
11:39
happening at all without looking at any data.
11:43
So, for example, if you are doing spam versus
11:46
not spam email classification, a lot of emails
11:48
coming into your inbox are probably not spam,
11:52
right? So the bias stem for the spam class should
11:59
be lower than the bias stem for the non-spam
12:04
class, so that the scores for the non-spam class are
12:08
already high without looking at the actual data.
12:13

So essentially, the classifier, we're
12:14
telling the classifier without knowing
12:16
anything else, you should, by default,
12:18
think about perhaps saying that this is not spam,
12:22
unless the content tells you
12:24
otherwise, in some sense.
12:30
All right.
12:32
So here's a check-in. So I'm going to be using
12:34
these check-ins throughout the lecture to kind of just
12:37
help us think about this together, think about
12:39
whatever it is that we're talking about. so here is
12:42
one simple one and we're just taking the problem
12:46
that we have been looking at but now thinking that
12:48
we have 50 emails that we want to classify it once
12:53
so I'm gonna give you two minutes to think about
12:56
it you can talk to your neighbor or just think
12:58
about it yourself but I'm gonna take two minutes 30
14:37
more seconds think about it
15:04
does anyone want to share their answer what
15:08
are you thinking. What are your thoughts?
15:11
Yeah, sure.
15:15
Excellent. Yeah.
15:17
So here we see that for W,
15:20
the dimensionalities are the same as this one
15:25
here, also the biostim. So in other words,
15:29
once you have trained your model, it doesn't
15:31
matter how many examples you get, that,
15:33
the parameters are now fixed, they're done.
15:37
And now the output changes.
15:41
So essentially what we're getting is, if you can
15:44
think about it this way, what we're getting
15:46
is, for each of the 50 examples, we are getting a
15:50
distribution of scores for all the 10 classes.
15:59
And also think about it every time. So OpenAI trained
16:03
a model, chat divotate. They have these parameters.
16:05
They have frozen them. They are done. so now it doesn't
16:07
matter how many people are now sending queries to them
16:10
the parameters are fixed it's only the output that
16:13
changes output dimensionality so that's the point I was
16:17
trying to make there this linear classifier kind of
16:24
looks super simple but it's actually the workhorse of
16:27
deep learning we'll see it again and again in neural
16:31
networks and so yeah so we can think of kind of neural
16:36
that works is like Lego, and then we can actually kind
16:42

of pick simple components, arrange them in different
16:46
ways, and so a linear layer is usually somewhere near
16:50
the top, but it's usually featured somewhere in there.
16:54
So for example, when we talk about transformers
16:57
a couple weeks from now, you will see that the
17:01
linear layer is here, featured here near the top,
17:04
and usually the linear layer is used in combination
17:07
with the softmax which we'll talk about now to
17:15
make one point here before we talk about the softmax
17:19
which is that one interpretation that we can give
17:24
to this features in a linear classifier specifically
17:28
is that we can think of each row as a kind of template
17:35
representation of each class so suppose so I
17:40
have here a simple example I have a four-dimensional
17:44
email representation so X here is my representation
17:48
of my input so we are simply saying for each email
17:53
we have only four features rapid four numbers representing
17:56
our email so this is a four-dimensional
17:58
representation and so the parameter matrix here is going
18:04
to be three by four so we have three labels span
18:09
personal and work and four four features so we have the
18:16
bias them one term for each class the way we are
18:20
going to compute the score let's say we're computing
18:26
the first score here, we simply take the first row
18:31
and our product with the feature representation.
18:36
So 0.2 here times 56 minus 0.5 times 231 0.1 times 24
18:47
2.0 here
18:50
times 2 and then we get the outputs, to
18:54
which we add the biostim for that class.
19:00
Which we add the biostim for that class.
19:04
So what I'm saying here is that one interpretation
19:07
is that each row of W is the template for
19:12
that class. So in some sense, when you look at,
19:15
when you eyeball just these numbers here, where
19:18
you're saying, okay, these numbers are kind of,
19:24
so for example, this is an email that's actually
19:27
scoring quite highly on the personal label.
19:33
And if we eyeball just the numbers here and
19:36
the numbers in the email representation,
19:38
we are seeing that, okay, we have a positive
19:40
number here, a big positive number here as well,
19:44
a big number here and a tiny number here. So this kind
19:47
of looks like the template for the personal email.
19:51
And that's just one representation.
19:53

So it's one way to think about it.
19:55
OK,
19:55
the rows,
19:57
representation of the class. The
19:59
columns are the features, right?
20:01
Feature one, feature two, feature three,
20:03
feature four. We have four features.
20:05
All right, great.
20:08
Where do these numbers actually come from?
20:11
4x, this is kind of we get a representation
20:15
however we get it, bag of words
20:17
representation or word embeddings.
20:21
For W, this is learned.
20:24
B, the biased terms are also learned.
20:26
And the class scores are calculated.
20:29
So today, we'll make progress towards the learning
20:33
part, how we actually learn W and the biased terms.
20:38
All right.
20:39
So that's a recap of what we need from lecture
20:43
two to talk about what we are actually
20:46
going to talk about today for this lecture,
20:48
the softmax classifier,
20:50
the
20:52
last function, and how we actually
20:56
go about obtaining the best weights.
21:00
All right.
21:03
Before we actually talk about the softmax,
21:05
let me zoom out a little bit here.
21:08
And let's just agree that machine learning models
21:12
are inherently uncertain. We're training these
21:16
models on data that is incomplete, might be
21:20
noisy, might be simply ambiguous. And so the
21:24
models are making predictions that are uncertain.
21:30
And so we need a way to be able to say, this
21:35
is the prediction the model is making, and this
21:38
is how confident we are in that prediction.
21:42
Suppose you are in a high -stakes situation, for
21:46
example. Perhaps you're a physician, and you are
21:48
using a machine learning model to predict if a patient
21:52
should receive a particular operation or not.
21:54
So you would like to have the model tell you, OK,
21:58
yes, maybe we should go ahead with the operation.
22:01
And this is how confident I am.
22:04
So if a model has, like, 57% confidence, it's quite
22:08

different from a model that has 98% confidence.
22:12
Assuming those, we can trust the confidence scores.
22:16
Let's just assume we can trust them now, for now,
22:19
and later on towards the end, we'll dive more into,
22:22
towards the end of the course, we'll dive more into
22:24
that. But we want a way to express this uncertainty.
22:29
And for this, we are going to use probability.
22:33
If you have taken a statistics course, this is
22:37
one very fine definition of probability that
22:41
I like, which is that probability is the
22:43
mathematical language for quantifying uncertainty.
22:47
This is exactly what you want to do. You
22:49
want to be able to quantify the uncertainty
22:51
in the predictions of these models.
22:58
Okay, great. So, so far, we
23:01
defined our linear classifier.
23:04
What we're doing is simply
23:07
weighted some of the inputs.
23:09
And what we're getting back are
23:11
these unnormalized class scores.
23:13
These scores can be negative. They can be
23:16
positive. They can be really large. They can
23:18
be tiny. So it's really hard to interpret
23:21
what these scores are actually telling us.
23:24
And so what we're going to do is we're going
23:27
to apply what is called a softmax function.
23:31
And that self-max function is going to
23:34
transform those scores into probabilities.
23:38
Great. So this is the definition
23:41
of the self-max function.
23:43
So we simply take, to compute the probability
23:48
of this particular input, little xi sub i,
23:53
belonging to the class k, we simply say, we
23:57
compute the score as before. we exponentiate that
24:00
score, we sum up all the scores for all of the
24:04
classes, and then we normalize using that sum.
24:09
So it's a very simple, simple, simple formula to use
24:15
to transform these raw scores into probabilities.
24:19
Alright, so here is a nice example to help us
24:23
think about it more concretely. And so suppose I
24:26
have some email and it's highly likely to be spam
24:30
so here are my raw scores coming from the linear
24:33
classifier these are the un-normalized scores what
24:37
the soft mix is going to do for us is simply we
24:41

exponentiate and once we exponentiate these scores
24:45
we've made progress towards probabilities
24:48
because now everything is greater or equal to zero.
24:52
Next, once we have these positive numbers, we
24:56
normalize by simply summing up these numbers
25:00
and then dividing these entries by the sum.
25:04
And so now we have gone from raw schools, where
25:07
it's really hard to say exactly what's going
25:10
on, how confident the model is, to these
25:12
probabilities where we are seeing that the model is
25:15
pretty confident that this is a personal email.
25:20
despite this being a spam email
25:23
so that's that's the idea here so
25:31
once you have probabilities this is really great because
25:35
now we can talk about the likelihood of your document
25:40
belonging to a particular class right and when we
25:47
can talk about that we can define our last function and
25:51
the last function is doing something simple it's telling
25:55
us how good or bad the predictions of the model are
26:02
so it's simply telling us so if the there's a high
26:09
probability assigns to the correct class then this loss is
26:12
going to be low so we are saying okay we're happy the
26:17
correct class is actually getting high probability but
26:21
if If the classifier is assigning low probability to
26:25
the current class, then the loss is going to be high.
26:29
In other words, the loss is telling us how
26:31
unhappy we are with the current model. And
26:34
the current model is defined by the parameters
26:36
we have at the moment. So it's telling us
26:39
how good our current type of parameters are.
26:43
That's what the loss is going to be doing.
26:48
Here's a precise definition.
26:50
So we have a data set with a lot of examples.
26:54
And we are going to compute the loss
26:57
of that data set one example at a time.
27:02
So this summation here, we are summing up over all
27:05
examples. And one by one, we are computing the loss.
27:08
And that loss is simply computed by we take
27:12
the prediction from the classifier. So
27:14
this F component is doing the prediction.
27:17
and we are comparing it to the true label.
27:22
So one by one, we compute the loss of each
27:25
of the examples and then we normalize.
27:29
Normalizing here is actually important.
27:32

We'll see why later, but one thing it allows us to
27:37
be able to say is to say we're talking about the
27:40
average loss per example because we have normalized it.
27:45
and more importantly it's going to actually
27:47
help make training stable that will become
27:51
clear once we talk about optimization in
27:54
the next lecture but we do want to normalize
27:57
and that gives us the loss over the entire data set
28:02
great
28:04
so the key question we're really trying
28:08
to ask is how much probability does
28:10
the model assign to the correct class
28:14
Now, let's go back to our example. Assume that the
28:18
correct class is spam, as we've been talking about.
28:20
And so here, we have computed probabilities using
28:25
the softmax, so from Roscoe's to probabilities.
28:29
And now, we are going to compare
28:32
to the correct probabilities.
28:35
So we're saying the correct probabilities are that
28:39
there's... All of the probability maths should
28:41
be on SPAM and zero on the rest of the classes.
28:46
So now we saw that the loss function takes the
28:51
predictions on the true labels and it's going to compute
28:56
how happy or not happy we are with these predictions.
29:02
Right.
29:04
And so this is the process of actually arriving at
29:08
the loss function, what is called the soft max
29:11
loss function the first step we say what we're really
29:16
interested in is we want to fit the data we
29:19
want the model to assign high probability to the
29:22
correct class that makes sense right we want the model
29:27
to do well in on the training data the second
29:30
step here is we apply the log to that probability.
29:37
So this is, again, very critical for training.
29:40
And the reason why is that these
29:43
probabilities can become quite tiny
29:46
and can lead to numerical instability
29:49
once we start multiplying them and so on.
29:52
This can lead to underflow. So we want to use logs
29:58
instead of directly using their probabilities.
30:03
So finally, we arrive at the last step, which
30:08
is we want to actually be able to minimize the
30:12
loss. So we want to, usually in optimization,
30:15
people prefer to minimize things instead of
30:17

maximizing. So we negate the log probability.
30:21
So the loss, for a particular example,
30:24
I, is the negative log probability
30:27
of that class under the current model.
30:32
Right.
30:33
And this is called the self-max loss, also
30:36
called the cross entropy loss, and it's
30:41
what we are using in PA1.
30:44
Yeah, sure.
30:52
Yeah.
30:53
Oh.
31:06
Yeah, so here we simply, so here we simply converted
31:12
these to probabilities right so we end up with
31:17
probabilities and then we are saying that these
31:21
numbers can be quite tiny so for example if you
31:26
have a language model its classifier is predicting
31:29
the next word and the classification task is
31:33
basically across the entire vocabulary so a lot of those
31:36
are going to have tiny numbers right so now we
31:40
are going to say we are going to take the log of
31:45
these numbers when we are computing the loss yeah
31:57
we are normalizing right we call
31:58
that we are normalizing right here
32:18
right okay I have a quick all right sure okay
32:22
so that's what we have that's all you're
32:23
using in PA1 and okay so I just want to show
32:29
you the loss for this example that we've been
32:32
looking at. So for this particular example,
32:35
the loss here, remember we are interested in the correct
32:39
class, which is SPAM, and we look at the probability
32:43
for that class assigned. And then we simply take the
32:46
negative log probability of the correct class under
32:50
the current model, which is 0.13. Not great. So our loss
32:55
is 2.04. And so that's how we do it. So for every
33:00
example in our data set, that's what we are going to do.
33:05
Notice that the other terms, these things here,
33:10
although we have probability mass allocated by the
33:13
model to these non-true labels, these are not really
33:19
contributing to the loss because their terms
33:21
actually cancel out. Excuse me, I'm going to go back.
33:24
Their terms cancel out due to
33:26
these zeros to the ground truth.
33:32
so those are not contributing
33:33
to the loss, just the true class
33:37

so let's think about this
33:39
together, so I want to give you
33:43
maybe three minutes for this one because
33:45
it requires some thinking to think
33:47
about how this is actually behaving we
36:52
can think about it together
36:57
does anyone want to share? What do
37:00
you have? What are you thinking?
37:03
Anyone want to share? Yeah, sure.
37:08
Right. Yeah, this is correct.
37:11
So the lowest is zero because
37:15
when the model is assigning
37:19
probability one to the correct label, that is what
37:26
we want. Sort of this is the best case scenario.
37:28
And that's when we take log of 1 and
37:31
log of 1 is 0. So that's the lowest.
37:36
And for the highest, this happens when the
37:40
model is assigning very tiny probabilities
37:46
to the correct labels.
37:48
And log of these tiny numbers is going to be
37:52
kind of something like negative infinity. and
37:57
then we take negative of that so we end up
37:59
with unbounded so this is the highest possible
38:03
loss is unbounded it can become really large
38:07
and often we don't want that so yeah
38:12
alright
38:14
any questions about this one
38:20
okay great
38:23
alright let's think about this one. This one is helpful
38:28
for debugging purposes when it's helpful to think
38:31
about it as well when you're working on your code.
38:34
So I'm going to give you maybe
38:36
say two minutes for this one.
39:14
Notice that here we're saying the loss
39:17
for a single example, a single example.
41:36
Think about it together.
41:40
Anyone want to share what they have?
42:02
Okay.
42:10
Any other thoughts? Yeah.
42:27
Right. Yeah. So B is definitely the right
42:30
answer. So let's just work through it a
42:33
little bit so we know how to arrive here.
42:37
Okay.
42:39
Great. So we're talking about when we have, we've
42:43

just initialized our parameters here. So at that
42:47
point, usually the values that we have in the matrix
42:53
W are kind of small numbers kind of close to zero
42:56
because at that point we don't know anything so we
42:59
don't want to introduce biases in the weights and
43:03
having them large so at that point the model has no
43:05
feelings really like it's kind of just giving scores
43:09
kind of close to zero for all the classes so suppose
43:13
these are the scores we have computed them using
43:16
the linear model so they're kind of close to zero
43:20
and now we exponentiate those according to
43:23
the softmax what is the exponential of zero?
43:30
if I exponentiate zero
43:36
one right exactly so all of these are going to be very
43:40
close to one so these are kind of small numbers close
43:44
to zero exponentiate those are going to be kind of
43:46
close to one and so we sum them up and we get a number
43:51
which is close to the total number of classes because
43:55
each one is essentially kind of just about one and now
44:00
when you normalize we have essentially one over the
44:04
total number roughly the total number of classes we have
44:08
so in this example for the data point that we just
44:12
looked at for example where we have three classes
44:17
for spam the label spam might get something
44:21
like 0.35 as the probability so you can kind of see
44:25
that there's the uniform probabilities across all
44:28
the classes because at that point it's a set the
44:30
model doesn't know anything doesn't get too much
44:33
to go one direction or the other unless we have
44:38
initialized the bias biases in the right
44:41
way but usually people just let the model
44:45
learn these biases and learn everything.
44:47
So that's why at the beginning,
44:49
then, we have no preferences.
44:52
And we have sort of roughly uniform probabilities
44:55
across the classes.
45:01
And so when you're debugging, you can look at your
45:03
model and check if you have really initialized
45:05
everything correctly by kind of doing that kind of
45:07
check if you're doing classification, all right?
45:11
Any questions about this? Oh,
45:19
yeah. right
45:27
will this be considered a high loss or low loss the
45:37
lowest we saw is zero right so that's the lowest the
45:43

highest is unbounded really gonna be huge so when
45:51
you're getting to zero that's good I mean this is not
45:54
bad but often you can do better because as we can see
45:58
that this is just sort of the model is giving uniform
46:02
probabilities to this and so when you're optimizing
46:07
okay yeah alright let me just answer
46:09
a question right here which is that
46:11
the loss can be super huge any
46:19
other comments or questions
46:24
okay
46:26
alright so what we are doing here with kind
46:31
of saying we really want the model to put
46:34
a lot of probability mass on the correct
46:37
class has a name, it's called maximum likelihood
46:39
estimation, where we're saying we want to
46:45
model here, and by model we are really saying we
46:49
want parameters here that let's call them W sub ML,
46:55
so this is the maximum likelihood estimation of these
46:58
parameters we want this to be this matrix out of
47:04
all matrices of the right dimensionality that we're
47:06
interested in we want to grab the one that makes
47:11
the data that we have to observe labels as likely as
47:16
possible so when we are changing w this probability
47:22
will also change right so if we have very bad
47:26
weights. This log probability here is going to be low.
47:30
And maximum likelihood is basically saying
47:33
that we're going to actually grab the W that
47:36
makes this expression as high as possible,
47:40
makes the observed labels as likely as possible.
47:44
Great.
47:45
This is maximum likelihood.
47:51
And this is great, But remember we said
47:54
that the data that we have often is incomplete,
47:58
could be noisy, could be ambiguous.
48:01
So this is an OK goal, but what we are really
48:05
interested in is not the labels that we have already
48:08
observed because we already know those labels.
48:11
We don't care too much about those. What we
48:14
care about is how the model performs on unseen
48:17
data. We're interested in generalization,
48:19
wanting the model to generalize, right?
48:25
And for maximum likelihood estimators,
48:28
they can overfit to the data if we let them.
48:33
Especially once we end up with really complex models
48:36

like neural networks that have a lot of parameters,
48:39
they can easily memorize the data set. They
48:42
can easily overfit to the data set. And so that's not
48:45
really what we want. What we want, what we're
48:47
interested in in machine learning is generalization.
48:50
and so in addition to the
48:54
maximum likelihood estimation
48:56
we are interested in regularization
48:59
where we do want the model to fit the
49:02
data but we want it to fit it responsibly
49:08
we want to fit the data but responsibly and so
49:12
we want to avoid fitting the data too closely
49:15
and as I said this becomes important when you have
49:18
these flexible models huge models with lots of
49:21
parameters that can fit the data if you will really
49:24
let them all right so pure maximum likelihood
49:31
estimation focuses on training performance rest
49:35
regularization helps us with generalization all right
49:41
so in practice we don't really do pure MLE we
49:46
don't do pure maximum likelihood estimation we actually
49:51
do something like let the following we add a
49:57
second term to the loss and this term here is the
50:05
regularization term and this is the term that we just
50:09
talked about which is the term that is going to say
50:11
okay we do want to fit the data which is this portion
50:15
of the loss which we just we've been talking about
50:18
maximum likelihood part but we want to do it without
50:23
overfitting to the data without fitting the data too well
50:29
to
50:36
get an intuition for this let's look at this two
50:41
example we have this data points in two dimensions and
50:46
suppose that we want a function that fits this data
50:50
but we also really want to be able to make predictions
50:54
on future data points that may come up. So that's really
50:59
what we have, but our goal is really to generalize.
51:03
Here are two possible functions.
51:08
We have F1 here in blue.
51:14
This function here is very complicated.
51:17
It's kind of wiggly everywhere,
51:21
and it's fitting the data super well.
51:26
On the other hand, we have F2, which is
51:29
very simple. It's just a straight line.
51:31
It's not fitting the data as well.
51:35
But notice what happens with the unseen examples.
51:39

So remember the blue points are the ones we've
51:42
already seen. These are the training examples. but now
51:45
we have these unseen examples these are the light
51:48
colored points so the blue line for example this
51:54
complicated function that we have estimated here
51:57
fits the data super well perfectly it's going to
52:01
make this that this point here is supposed to be here
52:03
it's going to predict right here whereas the
52:07
straight line is making a better prediction and we see
52:13
this one here for example the blue line is predicting
52:16
that that point should have been here but it's
52:20
actually really here so this blue line is super
52:26
it's doing super well at fitting the data but not
52:30
so well at generalizing and so this is why we are
52:33
saying we can get these super wiggly complex
52:36
functions but that's not really what we are after we
52:39
prefer simpler model that might generalize better
52:48
Let's look at exactly sort of what's
52:52
going on here with the weight.
52:54
So W here contains all the learnable parameters
53:00
of the classifier, linear classifier.
53:02
Forget about the biostems for now.
53:06
And as we said, sort of we can think of these
53:08
as the rows are sort of representation templates
53:13
of the classes, and the columns are the features.
53:20
So we have these many features, for example.
53:24
Now, here's one way we can penalize these weights.
53:32
Here's one way we can regularize. We can say we
53:36
want these individual weights to be not super large,
53:39
right? so we are going to have what is called
53:43
L2 regularization where we square each of these
53:47
elements and then we sum them all up so if the
53:52
weights are large this term is going to be really high
53:55
and when we add that term to the last term it
53:59
means that the loss is going to be high and we
54:02
would want to do better and by doing better means
54:05
that we want to get these weights to be smaller so
54:13
right now you can see that sort of the this our term
54:17
here is kind of summarizing all these weights in one
54:21
term and you're thinking oh how are we going to know
54:24
which term actually contributes quite a lot to the
54:29
loss and so on so that will be exactly the subject of
54:35
the next lecture where we actually update individual
54:40
weights so we're going to get gradients we're going to
54:44

get gradients and those gradients are going to be
54:47
specific to the weights and they're going to tell us
54:50
whether this weight is too high or too low it's contributing
54:52
too much the loss or not and so on and so we're
54:56
going to get precisely details and information for
54:59
each of those elements and we're going to update them
55:03
according to how much they contribute to the loss so
55:12
here's one thing to keep in mind which is that
55:15
regularization is not either on or off. We have
55:20
what is called the strength and this strength
55:24
here if we set it to zero we simply go back to
55:28
maximum likelihood estimation and so the loss
55:31
is only affected by this first component here.
55:37
but if we set it to a really large value then we're
55:43
kind of saying the data is not contributing too
55:46
much we're just kind of focusing on regularizing
55:50
and often people use a value like 0.5 which is kind
55:55
of trying to balance between fitting the data and
55:58
generalizing trying to generalize and regularize
56:03
any questions yeah
56:34
right so
56:39
we can think of it this way so for example here
56:44
this curve here so yeah
56:55
I do want to answer your question I'm
56:58
just trying to see if I should answer
57:00
it now or keep going a little bit.
57:07
Okay, maybe let me get back to
57:10
your question in like two slides.
57:11
Yeah, okay, good.
57:14
All
57:21
right, we were here, and we'll get back to the
57:23
question. We'll think a little bit about it
57:24
together as well, but if I haven't answered
57:26
it by then, then we can go back to it as well.
57:30
So the question was kind of about sort of
57:32
model complexity and how the weights
57:35
kind of relate to model complexity, right?
57:37
Yeah, we'll try to get to that together.
57:40
And if we're still not there,
57:43
then we can go back to that question.
57:45
All right. So, yeah, let's think about
57:47
this together with a check-in first.
57:52
So I'll give you maybe three minutes for
57:56
this because it requires some thinking.
1:00:26

Okay, 30 more seconds.
1:01:14
Does anyone want to share what they think
1:01:28
to randomly pick someone because it
1:01:30
looks like people know the answer?
1:01:31
Oh, yeah, we have someone already. Yeah, go ahead.
1:01:34
B,
1:01:40
okay, cool, yeah.
1:01:56
Right,
1:01:57
yeah, yeah, that's a very fine explanation.
1:02:00
So the answer is indeed B. so here you have a
1:02:05
model that really is going to be overreacting so for
1:02:09
example if you have a really high weight on one
1:02:13
feature so you get this whole email it has other
1:02:17
words and so on but then you have this huge
1:02:20
influence coming just from one feature and then the
1:02:24
model simply overreacts from seeing that feature
1:02:29
now Now, if you think about the complexity
1:02:40
of the model, as you were asking about,
1:02:45
one extreme way to think about this is,
1:02:47
so if we have, we can think about if some
1:02:52
features have zero weights, like if the
1:02:54
weights are so small that they are zeros,
1:02:56
we're essentially throwing out the
1:03:02
influence of those weights, of those features.
1:03:04
And so the model becomes simpler.
1:03:07
There's a method for regularizing that actually
1:03:12
prefers those kind of solutions, where
1:03:15
the model actually throws out certain features
1:03:18
and it just gives zero weights to those.
1:03:20
For L2, as we have seen it, it won't throw out
1:03:24
features, but it's going to actually have, it will have
1:03:28
small weights on those features. So the model
1:03:30
doesn't overreact too much to any particular feature.
1:03:34
So even if you add a single word, so the model
1:03:39
is going to be behaving in a more reasonable
1:03:41
way than being super wild, which is sort of maybe
1:03:45
equivalent to a function that's kind of like
1:03:48
that. I mean these are super high dimensional
1:03:55
feature spaces so the figure might not always
1:03:59
make sense as we have seen the simple toy 2D
1:04:01
example but kind of this is the intuition yeah
1:04:08
all right
1:04:11
okay so we talked about L2 so this is one of
1:04:15
the simplest ways we could use to regularize
1:04:18

There's also L1, which I kind of described in words,
1:04:22
which is penalizing the absolute value, sort of
1:04:25
summing up the absolute values of the weights.
1:04:28
So these are super simple,
1:04:31
but there are others that we'll
1:04:32
see throughout the course.
1:04:34
In neural networks, what is more common these days
1:04:39
are methods like dropout, where you literally
1:04:42
drop out some of the connections in the network.
1:04:47
at training time at inference time you have the
1:04:51
entire network but at training time you throw out some
1:04:54
of the connections so that the network does not
1:05:00
depend does not depend too much on certain
1:05:03
connections so you are kind of making the thing
1:05:05
to be robust to changes and relying on this
1:05:09
and batch norm is another common one which
1:05:15
also helps with training dynamics and so on, and
1:05:18
we'll talk about it once we get to Transformers.
1:05:23
But the whole idea across all these methods is that
1:05:27
we're going to have a preference for simpler methods.
1:05:36
Alright, so this is a kind of a summary of what we
1:05:41
talked about with regard to regularizing and sort of
1:05:46
adding that term to the last. So we start off with
1:05:49
inputs, the x sub i's, these little x's feature absentations
1:05:55
of our documents. And then we have the weights.
1:05:59
Together, they go into the linear model, which
1:06:03
competes the score for each of the classes.
1:06:07
Now the true labels don't actually go through the
1:06:11
linear model that's hidden from the model. But the
1:06:14
way the model interacts with them is through the
1:06:16
loss, where we are comparing the true label to the
1:06:20
prediction of the model. So we have the data loss,
1:06:25
the first term of the loss. And then independently,
1:06:29
we have a regularization term which tries to say
1:06:35
we don't want to fit the data too well because
1:06:39
ultimately we do want to generalize to unseen examples.
1:06:45
All right.
1:06:47
Any questions about this one?
1:06:51
Yeah.
1:06:52
I have a question about regularization.
1:07:01
...of all the dimensions, which means that term scale
1:07:05
is network size, but the loss function term is like
1:07:10
scared over the sample size, which means it should
1:07:13
be order one for all model size. So with the increase
1:07:18

of the model size, the regularization term will
1:07:21
finally dominate. I feel it's like very contradictory,
1:07:25
like the purpose of the regularization, because
1:07:31
Oh yeah I see what you're saying there
1:07:34
that once you end up with these models
1:07:41
that are so huge you have all these layers and have
1:07:45
these deep models so when you don't have this term
1:07:49
which is a function of these parameters then it might
1:07:56
end up dominating yeah so there's a point to that
1:08:01
there's a good point to that but with these giant
1:08:05
models actually there's more happening with regularization
1:08:10
we don't simply do this kind of thing there's
1:08:15
for example things like layer norm happening where you
1:08:20
have to actually indeed normalize things within
1:08:23
layers and so on. And then you do batch norm where
1:08:27
you're normalizing things within batches. So you do want
1:08:30
to keep things in scale exactly because often the
1:08:33
scale is not really telling you useful information.
1:08:36
So that you want to get rid of that effect.
1:08:40
And so indeed, so modern regularization methods
1:08:46
are doing precisely that, getting rid
1:08:49
of fluctuations in scale and things
1:08:51
like that that are not informative.
1:08:55
Yeah, good question. Any other?
1:08:59
Yeah, we will talk about those methods
1:09:04
precisely when we talk about the transformer
1:09:08
because that's when it becomes relevant.
1:09:11
For the linear model, the number of parameters are
1:09:15
not that many, so these effects are not as pronounced.
1:09:19
so usually we just do this kind of simple thing yeah
1:09:24
any other comments or questions
1:09:29
right okay alright we have a check-in here so
1:09:33
I'm going to keep this one short because I
1:09:36
think we sort of touched on it so I'm just going
1:09:38
to give you two minutes to think about it 30
1:11:09
more seconds
1:11:49
great
1:11:52
does anyone want to share their thoughts
1:11:55
we kind of talked about this along the way
1:11:58
so probably people know the answer already
1:12:02
right
1:12:07
alright any thoughts
1:12:10
you want to share what are you
1:12:12
thinking I think you know the answer no
1:12:16

yeah
1:12:19
right exactly oh yay thank you
1:12:23
so we kind of talked about the fact that all right
1:12:27
you know you make a tiny change and
1:12:30
because it has high weights on it then
1:12:33
basically that causes the model to overreact
1:12:37
so in function space we said okay these
1:12:40
are functions that are super wiggly and we
1:12:43
don't want that and so that's what we are
1:12:46
trying to go against with regularization
1:12:50
all right okay so here is the summary kind of the
1:12:55
basic supervised machine learning recipe what we are
1:13:01
trying to do is okay we get some finite hopefully
1:13:04
many input output examples so data is really the key
1:13:09
thing in machine learning even in this day and age
1:13:12
of these giant models especially in this day and age
1:13:15
of gen models data is the commodity in machine
1:13:19
learning you need that and so we start with data a lot
1:13:24
of data and we are really finding a function that
1:13:28
maps our input to output and we want that mapping to
1:13:35
be such that hopefully this prediction that comes
1:13:38
from this map is very close to the true label and
1:13:44
then we define a last function which is a function of
1:13:50
the prediction from the model and the gold label.
1:13:57
And then we said, okay, we actually don't
1:13:59
just want to fit the data. We want to have a
1:14:04
simple model that allows us to generalize.
1:14:08
All right, so now we have done really a lot up to
1:14:12
this point, which is that we have now precisely defined
1:14:16
this loss function for source max, how it looks like,
1:14:21
why we need it, how it behaves, so this is great.
1:14:25
But now we really want to figure
1:14:28
out, okay, given that function here,
1:14:32
how do we actually then minimize
1:14:35
it? How do we optimize it?
1:14:38
We are going to do optimization.
1:14:45
for linear models this is kind of still a nicely
1:14:50
behaved function but once you go to complex models
1:14:55
such as neural network the functions are actually
1:14:58
not convex at all they are very complicated
1:15:00
and scary but it turns out that we the method that
1:15:06
we're going to introduce still actually gives
1:15:10
us good solutions for reasons we will discuss in
1:15:16
the next lecture or the next couple of lectures.
1:15:20

But the key thing, though, is that the only
1:15:23
thing that we are imposing is that this
1:15:25
last function be smooth, something differentiable,
1:15:29
something differentiable. That's all.
1:15:36
so I'm not going to start optimization today we
1:15:41
need fresh minds for that so we are going to start
1:15:45
with that in the next lecture for now if you have
1:15:48
comments or questions we have five minutes just
1:16:19
now so the the battery is low so
1:16:22
for this lecture I don't know.