

Deep Learning
Prof. Mitesh M. Khapra
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 3.2
Lecture – 03
A typical Supervised Machine Learning Setup

We will start module 2 which brings us to a typical Supervised Machine Learning Setup this is a very important module please pay attention.

(Refer Slide Time: 00:21)

Sigmoid (logistic) Neuron

y

$w_0 = -\theta$ w_1 w_2 \dots w_n

$x_0 = 1$ x_1 x_2 \dots x_n

- What next ?
- Well, just as we had an algorithm for learning the weights of a perceptron, we also need a way of learning the weights of a sigmoid neuron
- Before we see such an algorithm we will revisit the concept of **error**

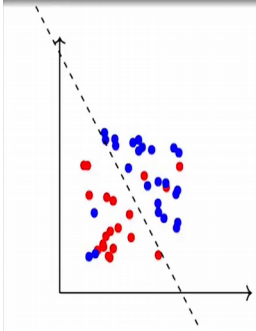
Mitesh M. Khapra Lecture 3

So now we have a sigmoid neuron, we have taken care of the fact that the perceptron was a very harsh function. So, we have a smooth function so, things are fine. Now what next? Where do we go from here? What is my next topic going to be? Yes, a lot of you are giving the right answers; we need to learn these weights. It does not help just to define the function, this function depends on certain weights, and now I need to give you an algorithm which will help you to learn these weights.

Now remember when I talked about perceptrons before giving you an algorithm what did I revisit? What did I talk about the error surfaces, right? And then I had motivated from there that our goal is to find a set of weights which give us close to give us 0 error in that case, right? Or in general's speaking generally they should give us a minimum error,

right they should help us to minimize the error rate. So, I need to set up that similar story here. So, we will again revisit the concept of error.

(Refer Slide Time: 01:13)



- Earlier we mentioned that a single perceptron cannot deal with this data because it is not linearly separable
- What does “cannot deal with” mean?
- What would happen if we use a perceptron model to classify this data ?
- We would probably end up with a line like this ...
- This line doesn't seem to be too bad
- Sure, it misclassifies 3 blue points and 3 red points but we could live with this error in **most** real world applications
- From now on, we will accept that it is hard to drive the error to 0 in most cases and will instead aim to reach the minimum possible error

Mitesh M. Khapra Lecture 3 12/02

So now in the case of perceptron I had shown you this figure which they were this data is not linearly separable which is obvious. And I told you that perceptron cannot handle this data, right. But what do I mean by it cannot handle this data? It cannot give 0 error, right? But what would happen if I run the perceptron algorithm on this, take a guess, what does the perceptron algorithm do?

Fine, and I could convergences my condition, right I could make that condition a bit loose. What is a valid convergence condition that you would lose here use here? Till almost all my points are separated properly, right. So, instead of aiming for 100 percent separation, I could have a threshold which says as long as 90 percent of the points are separated; I am fine. With it that looks like a reasonable thing to do, right.

So now if I run the perceptron algorithm with that condition, what do you think will I get as a decision boundary? Everyone has a picture in mind, ok, let us see, does this match what you had in mind roughly? Of course, there many things possible, but it will basically pass through this. Now what is happening here? What is the problem? There are 3 blue points which are wrongly classified and 3 red points which are wrongly classified.

But in most real world applications we will find that this line is not too bad, you could live with this error, right? This is probably 3 out of 30 on both sides which is roughly 10 percent error, unless you are using it when some mission critical applications or in health care where it is a life and death situation or something, in most cases you could live with this right. So, if you are trying to predict whether people will vote for a particular party, if you make this kind of error it would be largely unless it is a very close election, but largely it would be ok, right.

So, we could live with this kind of errors in most cases. So, from now on we are not going to be too optimistic, and if you are going to say that there would be some error, but my job is to find the weights such that my error is minimized, ok. I want the minimum possible error that I could get is that fine? Ok. So, again whatever weights we want to learn, we are going to be driven by some error function and we would want to minimize that error function, ok.

(Refer Slide Time: 03:14)

This brings us to a typical machine learning setup which has the following components...

- Data: $\{x_i, y_i\}_{i=1}^n$

Handwritten annotations on the slide include:

- x (input)
- y (output)
- Example input: $x = \text{Genre, Actor, Credits}$
- Example output: $y = \text{Salary, Academy Award}$
- Example input: $x = \text{occ, Salary, family}$
- Example output: $y = \text{fraud}$

Equations shown on the slide:

$$y = f(x)$$

$$y = \hat{f}(x; w)$$

$$y = wx + b$$

$$y = mx + c$$

Mitesh M. Khapra | Lecture 3 | 13/02

So, this brings us to a typical machine learning setup which has the following components. So, this perhaps is the most important slide in the course, and I will say this at least for 100 other slides in the course, but at least for now this is the most important, ok.

So, you are given some data x_i, y_i and you are given n such elements right. So, let me just elaborate on this, and give me I will give you some instances of this. Let me give

you some instances of this right; so, one thing we say I already told you was this; so, this is my x and this is my y . So, one example which I gave was about movies, right? So, this was genre, actor and critics rating and so on, right this is one instantiation of this problem. I could also give you another instantiation which was I just told you oil right. So, this is how much oil can I get, and here my factors were salinity, density and so on, right? There were many other factors, ok.

So, this was my x , again x belongs to \mathbb{R}^n , where n is some number integer. And another example could be say fraud detection. So, I have a customer, I am a bank; I have a customer who has bought some credit card, and I want to predict whether he or she would commit a fraud, right? And I would look at factors like what is his occupation, right maybe salary, maybe family size and so on, there could be various factors which I could look at, right? So, here again this becomes an x ok, and you could think of various such examples, right, where you are given an x and you are given a y ok. So, this is the data that you have.

Now, what is machine learning, where does machine learning fit into this, right? So, we know that there is some relation which exists between y and x . In each of these cases all of us are convinced that there is some relation, right. So, whether a person would commit a fraud would depend on these factors. It is reasonable to assume, that it is not a very wild assumption, right? Whether you would find oil at a location would depend on some of these factors and it is related, and similarly for the movie case, right.

So, there exists some true relation between x and y , such that if I plug this value of x into the relation, it would give me the value of y . There exist a true relation, this true relation could be governed by various things right, it could be governed by physical laws example in the oil mining case, it could be even governed by biological laws; again the marine life in that location and so on. It could be governed by economic laws, it could be covered by psychology right we do not know why a person cheats what is his function that he is using when he cheats and so on right. So, these could depend on various factors.

But we all agree that some function exists, hence we get these values for this particular input, right? For every input we get a certain value. So, there is some function which takes us from the input to the output, we do not know what this function is, right? We

never know in practice it is a very, very complex function is all that we know, we do not know this exact function. If you knew this exact function then there is no problem to solve, right, we just use that function and you can predict how much oil and all of us can become billionaires, right.

So, that is not the case, right we do not know what this function is. So, then what do we do in machine learn? We make an assumption, ok, we make an assumption that there is some function which takes x to y , and this function is governed by some parameters, right? And this is our approximation of how the real world works. And now under this assumption, we want to predict the parameters of this model, given the data, ok.

Now, let us take a very simple case where we could assume that y is equal to $w x$ plus b , I am taking this in the scalar single dimensional case, right? Now how would you estimate the values of w and b , oh come on. If I give you 2 data points, you can estimate the value, or should I write it that would jog your memory, right, this is how we all learn right. So, m and c you can estimate if I give you 2 data points, right. So, that is the simplest case, now we will make similar assumptions, but more complex functions and just as we could estimate m and c from the data, we would expect to estimate w 's also from the data, ok. So, that is what the machine learning setup is so, let us see.

(Refer Slide Time: 07:48)

This brings us to a typical machine learning setup which has the following components...

- Data: $\{x_i, y_i\}_{i=1}^n$
- Model: Our approximation of the relation between x and y . For example,

$$\hat{y} = \frac{1}{1 + e^{-(w^T x)}}$$

or $\hat{y} = w^T x$

or $\hat{y} = x^T W x$

x

Mitesh M. Khapra Lecture 3

So, the model when we talk about a machine learning model, it is our approximation of the relation between x and y . And we are free to make any such approximation, right. So,

I could say that this is what I think is the relation between y and x , and which is governed by some parameters w . Do you know what is this function? Have you seen this before? No, not sigmoid, ok.

Which model is this logistic regression, right ok, but I could also have made a different assumption, I could have made this assumption. What do I get? Linear regression, ok, please note that this error on the slide, ok, and I could make some other assumption; I could assume that y is actually a quadratic function of x , right. I am free to make any assumptions, the only thing I need to ensure is there is some parameter involved. What is wrong with making this assumption? If this is valid, is this also valid? If not, why? There are no parameters.

So, no not for any x we will get the we will it will depend still depend on the value of x , right if I plug in different values of x , I will still get a different output, there is nothing to learn. What do I do with all the data that I have, right? There is absolutely nothing I can use it to learn, I have just said that y is equal to 1 over 1 plus e raised to minus x , I can ignore all the data that you had given me, whenever you give me a new x , I will just plug it into this formula and tell you the answer, right? And that is bound to be wrong because I have not adjusted this formula.

Now, once I put in the w 's, it gives me this degree of freedom where I can now adjust the formula, I can learn the w 's in such a way that my predicted y 's are very close to the actual y 's, right. So, that is why we need always need a parametric form, right of course, there is nonparametric learning also, but I am just saying in this supervised setup, we are thinking of models whether you have parameters ok, ok. So, you have the data, you have the model, the model always has some parameters.

(Refer Slide Time: 09:50)

This brings us to a typical machine learning setup which has the following components...

- **Data:** $\{x_i, y_i\}_{i=1}^n$
- **Model:** Our approximation of the relation between x and y . For example,

$$\hat{y} = \frac{1}{1 + e^{-(w^T x)}}$$

or $\hat{y} = w^T x$

or $\hat{y} = x^T W x$

or just about any function

- **Parameters:** In all the above cases, w is a parameter which needs to be learned from the data

Mitsh M. Khapra Lecture 3 13/62

In all of the above cases, w is a parameter right, either the small w which is a vector, or the capital W which is a matrix right. So, notice that this is a matrix, this is one cross n , n cross n and n cross 1 , right, fine.

Now, how do we learn these parameters? That is the question that we need to answer, right how do we learn these parameters? We are convinced about 2 things that we never know the true function. So, we come up with an approximate function, and we have to insert some parameters in that function, so far good. Now I have to be able to learn these parameters, right?

Now for learning these parameters, we have something known as an learning algorithm. So, did you see any learning algorithm so far? Perceptron learning algorithm right. So, you already saw the perceptron learning algorithm, and it was able to learn the weights for a perceptron.

(Refer Slide Time: 10:43)

This brings us to a typical machine learning setup which has the following components...

- **Data:** $\{x_i, y_i\}_{i=1}^n$
- **Model:** Our approximation of the relation between \mathbf{x} and y . For example,

$$\hat{y} = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x})}}$$

or $\hat{y} = \mathbf{w}^T \mathbf{x}$

or $\hat{y} = \mathbf{x}^T \mathbf{W} \mathbf{x}$

or just about any function

- **Parameters:** In all the above cases, w is a parameter which needs to be learned from the data
- **Learning algorithm:** An algorithm for learning the parameters (w) of the model (for example, perceptron learning algorithm, gradient descent, etc.)
- **Objective/Loss/Error function:** To guide the learning algorithm - the learning algorithm should aim to minimize the loss function

Mitsh M. Khapra Lecture 3 13/62

There are various such algorithms, today we are going to learn one such algorithm which is gradient descent, ok.

Now, any kind of learning what is it driven by? Learning is driven by errors, objective function. And so, the analogy which I like to give is, suppose you are trying to learn trigonometry, right you have a chapter that is your training data, the chapter has a lot of formulae, that is your training data. Now what is your objective? There are 2 objectives, actually I will tell you the easy objective, the training time objective is that once you read to the chapter a few times, at least whatever formulae are given in the chapter, you should be able to produce the correct output for that, right? So, if I ask you what is sin square theta plus cos square theta, you should be able to answer them. And you should be able to give me the correct answer, right?

So, in other words you are trying to minimize the error on the training data, right? Whatever training data you have which is the chapter content, you want to make 0 errors in anything which is given in the chapter. That is your training error. Of course, there is also something as known as a test error; because after you have learned the chapter. I will give you an independent set of exercises, which might contain questions which are not seen in the textbook earlier, right. So, you would have seen sin square theta plus cos square theta.

But now I could ask you some other formula, which you should be able to give me answers, if you have learned properly right. So now, right now we are just talking about the training error; that means, getting all the formula in the chapter correctly and our chapter is actually the training data which is given to you, this is what we are reading. So, this always going to be driven by an objective function and our goal is just as we wanted to minimize the errors that we make on the formula given in the chapter.

We want to minimize the errors on the training data; is this set up absolutely clear to everyone? Anyone who does not understand this has any doubts? So, this is something this is the same framework that we will use again in lecture 18, 19, 20 and so on to explain some more complex models, right. So, you have to absolutely make sure that you understand this, right it is not very difficult, but just make sure you understand this fine. So, let us concrete at this a bit more.

(Refer Slide Time: 12:44)

As an illustration, consider our movie example



- **Data:** $\{x_i = \text{movie}, y_i = \text{like/dislike}\}_{i=1}^n$
- **Model:** Our approximation of the relation between \mathbf{x} and y (the probability of liking a movie).

$$\hat{y} = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x})}}$$

- **Parameter:** w
- **Learning algorithm:** Gradient Descent [we will see soon]
- **Objective/Loss/Error function:** One possibility is

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

The learning algorithm should aim to find a w which function (squared error between y and \hat{y})

NPTEL Mitesh M. Khapra Lecture 3

And we will consider our movie example and try to fit that into this framework. So, what is our training data there? They are given movie comma like dislike. And when I say movie, I am just using a shortcut it is actually all the details of the movie, right? Genre, actor, director, critics rating and so on, that is our input, and our output is the like dislike value.

What is a model that I chose? What is a model that I chose? I do not know what is my true relation between when I like a movie or not, but I made this approximation, that this

is how y depends on x , and I made sure I introduced some parameters there. I could have chosen some other functions also, but I chose this, right? Now the parameter is w this should be bold w , the learning algorithm that we are going to use is gradient descent which we will be seeing soon.

And what is an objective function here? Can you tell me a formula? So, we have been talking about it in terms of English, that I should be able to get predictions which are as close to the true prediction. Can you put it into a formula, y_i minus

Student: (Refer Time: 13:51).

\hat{y}_i where \hat{y}_i is the prediction, right this is the prediction. So, that is y_i minus \hat{y}_i that should be minimized, is that fine? Whole square of that. So, why do you square it? So, that is correct.

So, for all the training points, n training points, I want to minimize the square difference between y_i the true prediction, and the prediction, sorry the true value and the predicted value, is that fine? And why do I use squares, differentiable is 1, the other thing?

Student: (Refer Time: 14:24).

The positive errors and the negative errors should not cancel, right? So, it would be happen that on some movies, I make a positive error of 0.5 right; that means, the actual label should have been 0.5 and I gave 1. On some movies I make a negative error of 0.5, and these would cancel each other, and I will get the false impression that I am making 0 errors. But once I square the values the negative values also become positive: so, this cannot happen right. So, that is why we always use the squared error function, and also this is differentiable which is more important right, ok.

Now, the learning algorithm should try to minimize this particular quantity, ok? So, this is a typical machine learning setup, almost any supervised learning problem that you see you could cast it in this framework, change the \hat{y} function appropriately, change the parameters appropriately, maybe use a different learning algorithm depending on the problem that you are trying to tackle. And you should be able to fit it into the same thing, is that fine? Ok, at least for this course everything that we do we will largely be able to fit it into this framework, ok, fine.

(Refer Slide Time: 15:28)

Module 3.3: Learning Parameters: (Infeasible) guess work

NPTEL Mitesh M. Khapra Lecture 3

So, that is where we end this module.