NPTEL

NPTEL ONLINE CERTIFICATION COURSE

Introduction to Machine Learning

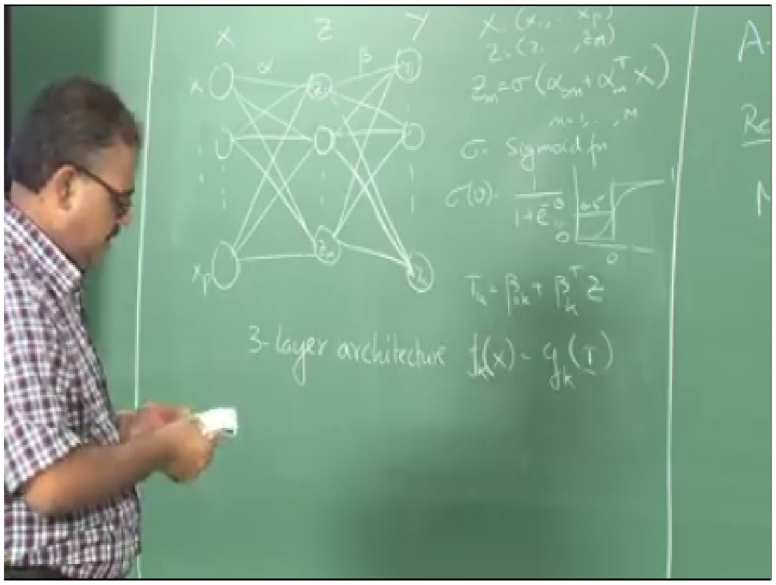
Lecture 33

Prof. Balaraman Ravindran
Computer Science and Engineering
Indian Institute of Technology Madras

Artificial Neural Networks II – Backpropagation

Now we move to artificial neural networks I suppose as a person looking at neurons.

(Refer Slide Time: 00:26)



So we are going to start hooking up neurons in multiple layers like this and the way we are going to train this is using gradient descent right, where we are going to Train this neural network is using gradient descent but what is the problem with using gradient descent for perceptrons the non-linearity right. So we had a threshold function which was not differentiable for the other line we got around it by getting rid of the threshold function altogether as I said we will use a linear output right.

So what is the problem in using linear outputs in multi layers multi-layer perceptrons so if you think about it let us call the first layer weights α and the second layer weights I will call them β right, so if you think about it so what you are producing is some $\alpha^T x$ is the output here if it did not have any non-linearity if I just use a linear neuron the output from here will be $\alpha^T x$ right so that will be $\alpha^T x$ and what will go in here Z so Y will be that which is equal to just like having one layer of weights given by $\alpha \beta$.

There is no point in doing all this layering so if you only have linear neurons then I do not get the power of doing all the layering and I might as well have done a single layer of neurons, so I really need to do something nonlinear in the middle I need to have a threshold function for me to get the power of layering right so the threshold is actually needed but whatever we did that if I did kept it as linear neurons run into trouble great, so we need the threshold so how do we get around the fact that it is.

So I am going to say that okay now you get the drill so why is why I want to wake it yeah so at the σ riser okay so σ is like a soft threshold right, so I can throw in a slope parameter here which I have not done that will give me different rates at which this σ will assign so the actual threshold will be that way σ will give me a soft way of doing the threshold the nice thing about the σ is it is differentiable there are many choices that you can have for the non-linearity a differentiable threshold function.

So the sigmoid is one of them so the thing with the sigmoid is that it will be between 0 and 1 so if you want it to be between -1 + 1multiplied by something that will go from – 1 and + 1 right so you could have different choices, so for the time being I will stick with the regular the sigmoid that we are familiar with okay so each Z here, will be given will be given by that expression right so T is the quantity that you will see that goes as the input to the sigmoid in the output neuron redness the output from here.

From the middle are a hidden layer this one not one of this one 0 m α_0 m + α m^T x can you zoom into that a little bit what is opening that can you see it better, enough people at the back who did not complain the phones are small can you see it better now so what are you doing on the laptoper actually seen the video feed of this or something right so that is the output of the first layer okay and the output of the second layer is given by some other function G acting on this input T okay right.

So like to see if anyone notices something funny actually all can be different each one of these is a unit like this of that like that right, each one of those it is a one block this evil there are a good point okay yes there are three layers and this is sometimes called the standard three layer architecture okay but the first layer is really a fake layer this layer is really a fake layer it just takes x1 and gives x1 out on all the outputs okay, this one takes x2 in and gives x2 out on all the outputs okay this is this is really not a neuron okay so sometimes I like to call this as a two layer network because for me there are two layers of weights okay so it is a two layer Network right but in the literature for some reason this is called a three layer architecture.

So this is called the input layer this is called the output layer okay one in the middle is called the hidden layer, so why is it called the hidden layer because I do not see the outputs of that layer directly okay so they are called the hidden layer so this is called the standard three layer architecture but there are other ways of doing it where actually you take outputs from the middle layer as well right and we can do and we can have inputs feeding into somewhere in the middle so you can have all kinds of craziness okay.

So the standard architecture is EC and we will stop with that okay I am not going to going to all the crazy neural network architectures are out there so you might want to take another course on a enhance specifically, if you want to I know more about thee all the crazy architectures outside so there is time permitting I will come back and do something very quickly at the end of the course not today but this is the standard architecture will stick with okay right so still people have not told me is there is something odd about this yeah.

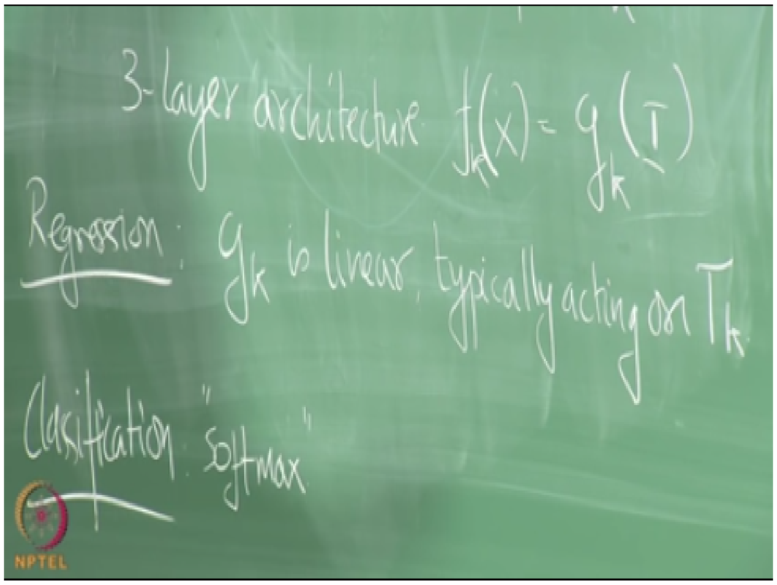
Lastly it always has to be linear not necessarily but it did not also be sigmoid that is why I written it as decay, so the last layer could be sigmoid it could be linear right when we do you want the last layer to be linear when we want it to be linear is when you want to do regression for sure right when you want it to be a sigmoid is when you want to do classification and still people have not told me what is odd about this I am assuming people are thinking but I am waiting for the answer.

So that I can go that is why I am stalling yeah, so why did I write this directly as α m + αm^T x α or not m + αm^T x but here I split it up into T_K and F_K sorry good point, but why is it TY is it not T_K so if I am doing regression I might as well do it T_K right, so usually I because my regression my regression variables right if I am doing multiple output regression okay I am not talking about multiple input regressions multiple output regression my output variables are typically taken to be independent right.

So min so what is the what value I predict for one I will usually does not affect the prediction I make for the other right so I do not know if you read the book I did not I did not talk about multiple output regression before this but if you read the book they would have actually told you that you can do that regression independently right but if it is classification really they are not independent.

If I am going to be outputting class probabilities right they cannot be independent right if I am outputting class one probability is higher than class two probability necessarily has to be low okay so I had to say do some way of normalizing the outputs to produce probabilities right, that is why I am saying that this will operate on the entire T, I can produce a output probability vector so in case of classification you need to operate on the entirety so we will come back to that in a second right for classification. We will do a soft max like we did well I will just take a regression do you remember that so E power.

(Refer Slide Time: 15:16)



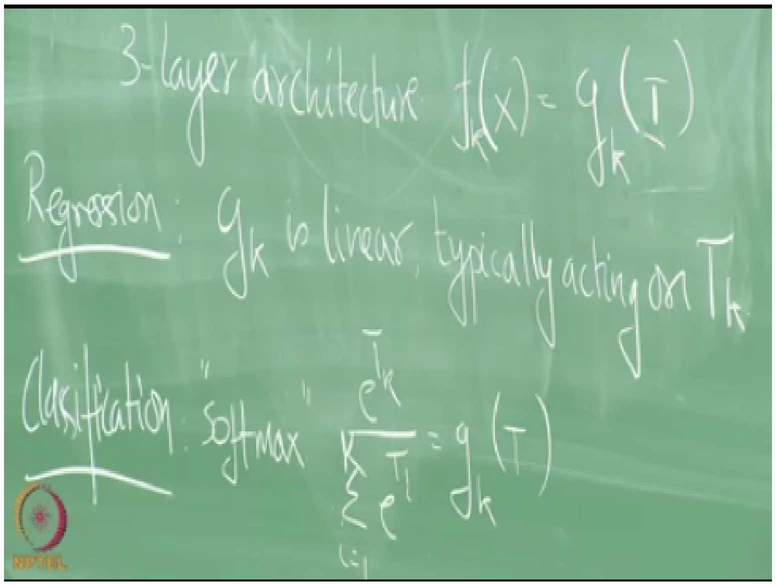
3-layer architecture $f_k(x) = g_k(T)$

Regression: g_k is linear, typically acting on T_k.

Classification: "softmax"

Yeah so I need the ∑ over all the outputs right.

(Refer Slide Time: 15:23)



3-layer architecture $f_k(x) = g_k(T)$

Regression: g_k is linear, typically acting on T_k.

Classification: "softmax" $\frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}} = g_k(T)$

I need a ∑ over all the classes in the denominator and that is why my gk operates on T so this will be the soft max thing, so E power T_K divided by $\sum^e$ at this will be the gk of T so what will happen if it is a single class I mean 2 class problem, it will reduce to a sigmoid it registers sigmoid I can pick one class and have that output as the sigmoid and I can just say that okay, if this is greater than 0.5 then it is that class if it is lesser than 0.5 this other class correct so this is a 2 class problem that gk will reduce to a sigmoid.

So if I am doing classification with only two classes I can straightaway keep a sigmoid as my output, output neuron and then I can solve it one how many neurons I need one right suppose in solving a three class problem how many neurons I need as output 3 and you can always have the third one as dummy and then you can say I am going to do 1 - the sum of the rest right but if you are going to do this right, so I can always say that okay I am going to have three outputs which will give me the probability of each of those three classes okay.

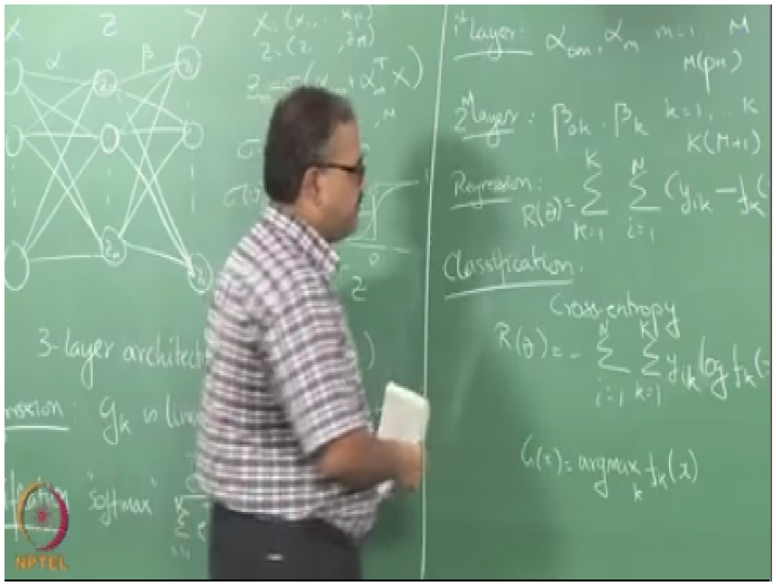
That is typically how it is done for two you just have one right but then for more than two classes you typically tend to have as many at the output is there any problem for doing that with doing that think about it I am not going to give you the answer right okay, so how do we fit the neural network parameters now and I have two layers the first layer I am into P + 1 parameters am for right, so m for each of the hidden neurons right P for the α hems and + 1 for the α_0 and in the second layer okay likewise, so I have that many parameters that they have to fit so I have to find all the α and all the meters so why do not you do more layers than this where did I stop with only three layers yes so empirically people observed that it is harder and harder to train why does it become harder and harder to train.

I will tell you in a minute okay but there is another reason for stopping with two layers right, so if you think of these as some kind of Boolean gates right if you think of the neural networks are some kind of Boolean gates it turns out that I can implement any Boolean function just using two layers of neurons except that the branching will become very large right but I can still implement any so as long as they do not give you any kind of gate with right I can have as many inputs coming in to a neuron.

As I want and I can implement any Boolean function in just two layers of neurons so why is that all of you know that right you can write midterm expansions right, so all of those things you know that and so you can essentially implement it in two layers of neurons and people thought oh, two layers is sufficient is a universal function approximator I can represent any function I want so let me not even think of what higher layers so that is one school of thought so people stop there but then there are others who are interested in going into more complex neural networks.

Because they did observe that when they got it to work okay adding more layers worked well so people kept at it and they made it work more robustly and so there like I was telling you that third wave of mutant light works is all about having deep networks where you have more than two layers.

(Refer Slide Time: 21:08)


1st Layer: α_{0m}, α_m m=1..M
M(p+1)
2nd Layer: β_{0k}, β_k k=1..K
K(M+1)
Regression: $R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k($
Classification.
Cross-entropy
$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k($
$G(x) = \arg\max_k f_k(x)$
3-layer architecture
softmax

So regression what will be a loss function so my I am going to say define my regression loss for the parameters θ what our θ here all the α and β okay, so I wanted a single notation for the parameters instead of saying α β everyday so I will say θ is given by essentially this quiet loss that should look really familiar to you guys by now because they have been writing squared loss almost once every class if not often so what about classification what can we use for classification.

But 01 is incredibly hard last function to optimize array so you could use squared error itself right so what is the rationale for using squared error same rationale that we use to linear regression right, so why I K is an indicator variable that this one that gives you the probability of this particular data point being class K right and you are trying to fit the probability anyway that is what you are trying to fit and therefore for every data point you can take its probability of being class case 1 or 0.

And then you can try to do some squared error and try to make a prediction okay that is one way of thinking about it the other way is to use what is called the cross-entropy error or the deviance which is related to whatever we did in logistic regression, so here I will define my so we will all put the problem the actual class table as the one that has the highest F_K of x like we did in the discriminant based classifiers except that F_K of x is no longer a simple discriminant function right.

And so this is essentially a error term okay that stands in for likelihood that we maximized earlier, so and to optimize this error term we will essentially train the neural network using maximum likelihood right so I am not going to go there so we will look at a more popular more mechanism for training neural networks which essentially looking at the gradient the squared error and do very indifferent okay.

IIT Madras Production

Funded by
Department of Higher Education
Ministry of Human Resource Development
Government of India

www.nptel.ac.in