

Data Mining
Prof. Pabitra Mitra
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 29
Artificial Neural Networks II

Let us now explain the mechanism of finding the correct set of weights to solve a prediction problem.

(Refer Slide Time: 00:27)

The slide is titled "Training rules" and contains the following bullet points:

- Finding learning rules to build networks from TEs
- Will examine two major techniques
 - Perceptron training rule
 - Delta (gradient search) training rule (for more perceptrons as well as general ANNs)
- Both focused on learning weights
 - Hypothesis space can be viewed as set of weights

Handwritten notes in red ink on the right side of the slide include: "Given (TE)", x_i, y_i , and W's. The slide also features the IIT Kharagpur and NPTEL Online Certification Courses logos at the bottom, along with a small video inset of the professor.

So, we have seen in the last lecture is that we can realize certain logic gates for example, and then or by proper choice of the weights w . So, this is interesting because what we can do is that the same neuron same neuron we can assign different weight values to realize different-different functions.

Functions means some-some target mapping say if this is say for example, end if this is the input is 1; 1 output should be one if input is 1, 0 output would be 0 or minus 1. So, like this given any input output function input to output mapping the same neuron can realize it or by changing just a value of the weights and the bias. So, what; so, this the same neuron this thing; these are called the architecture of the neural network the structure of the neural network.

And what we are trying to say is that the same architecture same structure can be used to learn different input output mappings for by varying the value of the weights. So, in the next part what we will see is that. So, now, given some input output mapping given some training examples of input output mapping what is the proper value of the weights that will achieve this input output mapping as close as possible.

So, as you know. So, since; so, what will be given is that we will be given some training examples t which will be some $x_i y_i$ a number of $x_i y_i$ like previous problem. So, this y_i may be either class level 0 1 or minus 1; 1 or it may be some real value we are given a number of such weights you have to find a set of w 's which given new x will correctly or as close as possible to correctly predict y .

So, the general technique of this training is as follows to start with initialize w to some random values initialize initial weights $w_1, w_2, w_{naught 2}$ random values and then update the weight.

(Refer Slide Time: 03:25)

Training rules

- Finding learning rules to build networks from TEs
- Will examine two major techniques
 - Perceptron training rule
 - Delta (gradient search) training rule (for more perceptrons as well as general ANNs)
- Both focused on learning weights
 - Hypothesis space can be viewed as set of weights

Handwritten notes in red ink:

- 1 Initialise W's to random values
- 2 Update W's as we see the $x_i y_i$
3. Till W's value do not change significantly

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, the rule is values; sorry, update w 's as we see the training example $x_i y_i$'s 3 till w values do not change significantly. So, this update will go on, we will call these iterations of update there are 2 successive iterations.

If the w values do not change significantly we stop there and use this w for predicting for a future point.

(Refer Slide Time: 04:50)

The slide is titled "Training rules" and contains the following content:

- Finding learning rules to build networks from TEs
- Will examine two major techniques (Update rules)
 - Perceptron training rule ✓ $w_{new} = w_{old} + \Delta w$
 - Delta (gradient search) training rule (for more perceptrons as well as general ANNs) ✓
- Both focused on learning weights
 - Hypothesis space can be viewed as set of weights

Handwritten notes in red ink include the equation $w_{new} = w_{old} + \Delta w$ and a diagram of a 2D coordinate system with several lines and points, some labeled with '+' and '-' signs.

At the bottom of the slide, there is a logo for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a man speaking.

We will consider 2 such update rules will consider 2 update rules the update rule will be of the following form. So, we have we have some initial value of w and this we are updating say suppose at certain time in a certain iteration we have a value of w equal to w let me call it w_{old} current value of w .

We will see an example and rectify this value of w by adding an objective delta w and this new value will be our new value of the weight abdicate value of τ it and this will continue. So, next w_{new} will become w_{old} again we will do $w; w$ an w we compute w go on doing this when the difference between w_{new} and w_{old} comes very small. So, we will study 2 rules as I have set the perceptron root and the delta rule.

Geometrically a way of doing this is the like this go back to the same picture if we have positive and negative set of weights you start with a random perceptron random w means random line separating line now u delta it. So, that it gets rotated and shifted till it becomes a good discriminator for all the training points actually this can also be looked upon as an optimization problem problem.

(Refer Slide Time: 06:37)

Training rules

- Finding learning rules to build networks from TEs
- Will examine two major techniques
 - Perceptron training rule
 - Delta (gradient search) training rule (for more perceptrons as well as general ANNs)
- Both focused on learning weights
 - Hypothesis space can be viewed as set of weights

Optimization Problem

w₁

w₂

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, if you look at all your w values w_1 w_2 other values of w_2 other values also and we define as a error that given certain value of w_1 w_2 to w naught how much error we are making in predicting y . So, if you plot that value of error along the z axis for defined different values of w_1 w_2 w naught you will get something called a error surface.

What do you want is that find the optimal value of w the value of the weights which gives you the leasted (Refer Time: 07:32)

(Refer Slide Time: 07:36)

Training rules

- Finding learning rules to build networks from TEs
- Will examine two major techniques
 - Perceptron training rule ✓
 - Delta (gradient search) training rule (for more perceptrons as well as general ANNs)
- Both focused on learning weights
 - Hypothesis space can be viewed as set of weights

W = [w₁, w₂, w₃]

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, that way it is a minimization problem also we will see this first algorithm is kind of easily visualized by you have a arbitrary plane sign line and you gradually tilt it by correcting your w ; w , we will call w vector as the vector of the pair w_1, w_2, w naught till it becomes separately. Now I will write down the algorithm.

(Refer Slide Time: 08:01)

Perceptron training rule

- ITERATIVE RULE: $w_i := w_i + \Delta w_i$
 - where $\Delta w_i = \eta(t - o)x_i$ (handwritten: "desired value")
 - t is the target value
 - o is the perceptron output for x
 - η is small positive constant, called the learning rate
- Why rule works:
 - E.g., $t = 1, o = -1, x_i = 0.8, \eta = 0.1$
 - then $\Delta w_i = 0.16$ and $w_i x_i$ gets larger
 - o converges to t

Handwritten notes on the slide include: "desired value", "learning rate", and a diagram of a perceptron with inputs x_1, x_2 , weights w_1, w_2 , and bias w_0 . A graph shows a decision boundary line with a bias $w_0 = 1$.

So, as I have mentioned, this is the object rule take any w_i s for any w_i have apply your threshold function you get your y . So, take any w its new value is the old value to start with in random initialization old value plus some delta of the assignment where what is delta w delta w is your target minus output.

So, what is target say some for some value of input say 1 1; suppose you are learning the n function. So, if input is 1 1, output should be 1, if your input is say this is the and function your input is 1 1 output should be one if your input is 1 1, let us say this point 1 1 outputs will be positive.

So, this desired value is what we call as a target value it is the desired value given x what is the output you want, but maybe the current value of these w 's do not give you the desired value, but it give some value if you take x_1 into w_1 x_2 into w_2 to the plus w naught put the threshold some value of y you will get that I call as o ; the output value; what will you do? You take the difference between target and output (Refer Time: 10:14) multiplied by that by the x the input vector; vector, note these are all vector terms delta w

is a vector because w is a bit that x is a vector $x \ 1 \times 2$ and your η is a learning constant learning rate usually small say 0.1 ok.

So, you multiply that and this has your update rule. So, this is my update rule. So, you note one thing that if your target output and actual output matches then δ is 0, no change in weight only when it does not match then you have a change in weight. So, this is an example.

(Refer Slide Time: 11:34)

Perceptron training rule

- The process will converge if
 - training data is linearly separable, and
 - η is sufficiently small
- But if the training data is not linearly separable, it may not converge (Minsky & Pappert)
 - Basis for Minsky/Pappert attack on NN approach
- Question: how to overcome problem:
 - different model of neuron?
 - different training rule?
 - both?

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 16

There is actually another way of looking at it.

(Refer Slide Time: 11:47)

Perceptron learning rule:

Weight vector $w = [w_1 \ w_2 \ w_0]$
 input vector $X = [x_1 \ x_2 \ 1]$ Training Samples:
 $x_1, y_1 \ + \ -1$
 x_2, y_2
 x_3, y_3

Rule:

1. Initialise w to random values
2. Look at a training instance x_i, y_i
 if Output O_i for x_i is same as y_i
 - no action
 otherwise update w
 $w_{new} = w_{old} + \eta (t - o) x_i$

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES 16

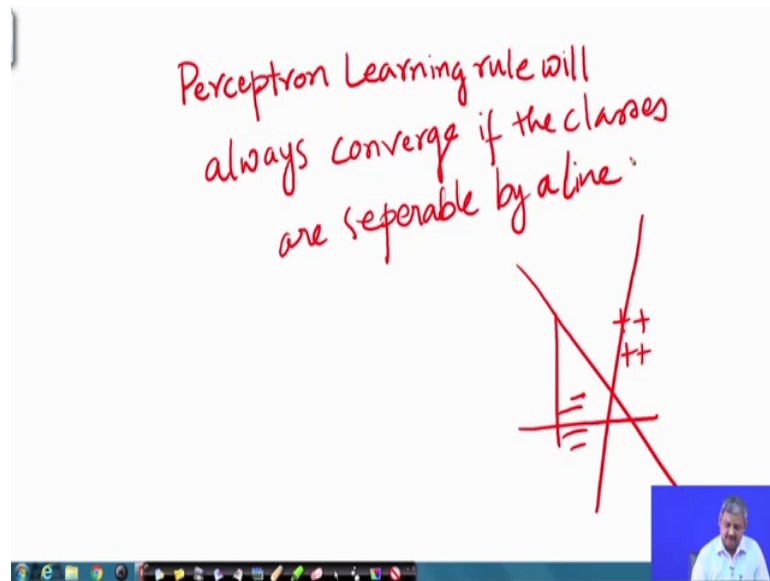
Suppose, I have 2 inputs it will be w_1 w_2 and the w_0 bias that will be the weight vector input vector it will be x_1 x_2 and third input x_0 is always 1. So, I write it as on bias input and you have a set of training samples this is a vector this is either plus one or minus one or something plus 1 0 also you can write.

So, the rule is as follows look at the training instance x_i y_i if the output value for x_i is same as y_i , I do not do anything. In other words, if a particular x_i is correctly classified by your w do not do anything for that x_i ; otherwise, if it is misclassified, if it is in the misclassified by this that is class level and predicted o does not match otherwise update w .

How you update plus η times note if it does not match $t - o$ is one otherwise it is 0. So, I am not writing $t - o$ that will be one and x_i . So, what do you do your new vector w vector is the old w vector plus your η times x_i times x_i often in perceptron learning rule this η particularly in the rule it is taken to be one and you get this new w .

So, you can see what is meaning that this is your w vector this is your x if it is misclassified you tilt it more towards that. So, that you include it in the positive side tilt it make it new w new. So, this is just another form of writing the perceptron group Minsky and Papert; 2 scientist famous artificial intelligence people; Marvin Minsky of MIT, it is what long back that this update rule is always going to converge if the classes are separable; that means, if you have classes like these that are separable whatever random w start with, if you follow this rule after sometime it will get a proper value.

(Refer Slide Time: 16:17)



So, of course, I am not talking about sigma. I am talking about only threshold function activation function by a line this as a nice result, all right. So, coming back to our discussion now let us look. So, this is clear I guess. So, let us look at the second rule second beta update rule which is the delta rule.

(Refer Slide Time: 17:23)

Gradient descent

- Solution: use alternate rule
 - More general
 - Basis for networks of units
 - Works in non-linearly separable cases
- Let $o(x) = w_0 + w_1x_1 + \dots + w_nx_n$
 - Simple example of linear unit (will generalize)
 - Omit the thresholding initially
- D is the set of training examples $\{d = \langle x, t_d \rangle\}$
- We will learn w_i 's that minimize the squared error

$$E[\bar{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$


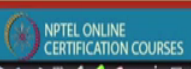

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

This rule uses the following file of a file of you known as the gradient descent.

(Refer Slide Time: 17:38)

Error minimization

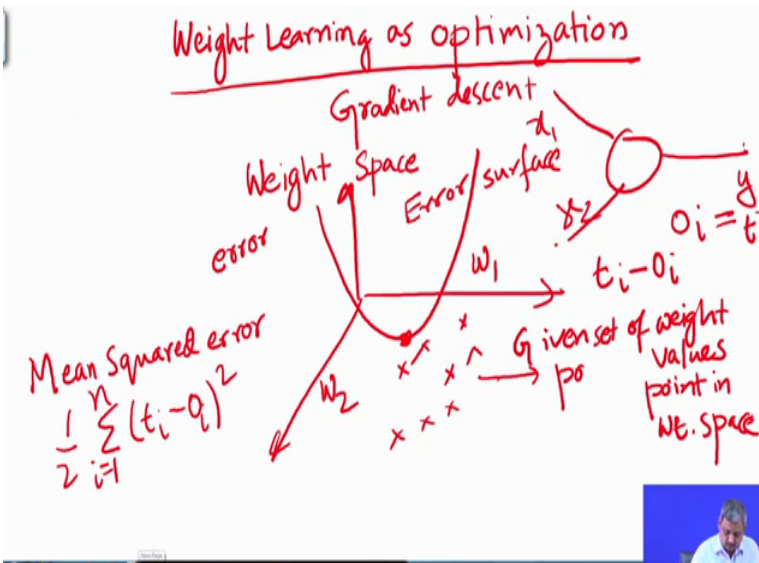
- Look at error E as a function of weights $\{w_i\}$
- Slide down gradient of E in weight space
- Reach values of $\{w_i\}$ that correspond to minimum error
 - Look for global minimum
- Example of 2-dimensional case:
 - $E = w_1^2 + w_2^2$
 - Minimum at $w_1 = w_2 = 0$
- Look at general case of n -dimensional space of weights



(Refer Slide Time: 17:51)

Weight Learning as optimization

Gradient descent



Weight Space

Error surface


Mean Squared error

$$\frac{1}{2} \sum_{i=1}^n (t_i - o_i)^2$$

Given set of weight values point in Wt. space

$t_i - o_i$

$o_i = y_i$

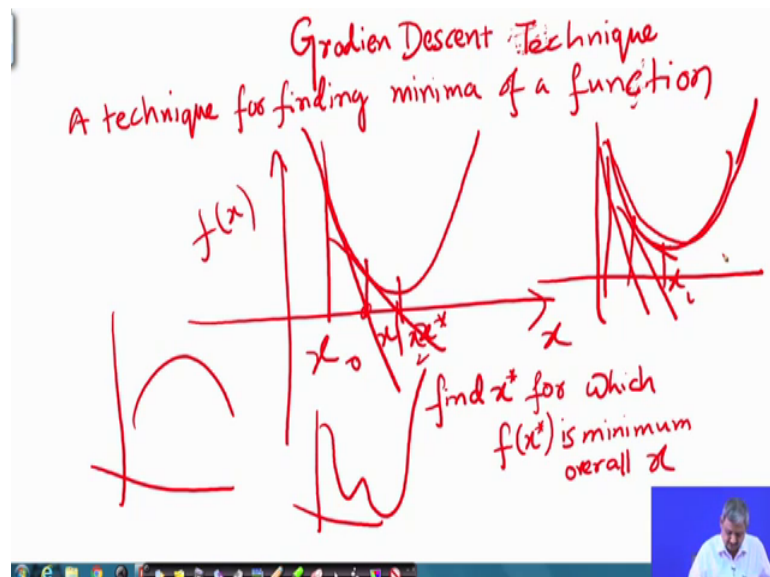


What is the file of; so, as I had said it considered something called a weight space what it weight space it take all the weights as axes, I can draw more. So, a given set of weight values is a point in this weight set weight space weight space this is one. So, you can have different type of weight values that may be more weights, I am not drawing them because you cannot visualize and in that z axis you plot the error what is error see if you take certain value of weight for some input we will get some output.

Some set of inputs we will get that output let me call it as o_i may be same as target value t_i , but the desired output or may not be same. So, the error is this difference. So, one possible way of computing error is what is called the mean squared error is half summation all the training points square. So, if you fix your value of w you get some error now if you change your value of w you get a different-different values of error if you plot them you get the error surface.

And weight learning is nothing, but finding a weight which gives a minimum error. Now let me tell you a technique of doing that.

(Refer Slide Time: 20:47)



Suppose, I have a function $f(x)$ it looks like this let's say all x this is what you have to find how do I do guess take certain value of x ; x_0 let me call it draw a tangent to $f(x)$ see where it intersects that we will call as x_1 look at $f(x)$ again draw a tangent call that as x_2 you see if you go on doing this soon, we will your x_i will come to the minima for functions which are well kept look parabolic.

If the function is like this it will not work, but if the function is like this again it will not work it may stop here fortunately that the perceptron error looks like this. So, you can calculate the gradient and do like this. So, this is the fundamental principle of something called gradient descent. So, we had a rule the delta rule which is based on this technique I will explain it.

(Refer Slide Time: 23:45)

Gradient descent

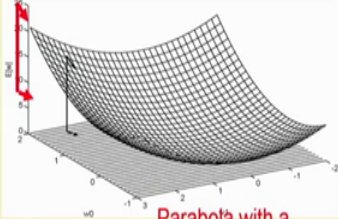
- Gradient “points” to the steepest increase:

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$




- Training rule: $\Delta \vec{w} = -\eta \nabla E[\vec{w}]$
 where η is a positive constant (learning rate)

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

How might one interpret this update rule?



Parabola with a single minima


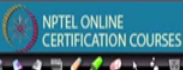





So, this is what I have said this it is. So, another way of looking at it is like suppose you are in a mountain you have to come down what you do you look around find out which direction the slope is steepest take a step again look around take a step if you do this you will eventually come to the lowest point it will eventually come to the lowest point this is the fundamental principle.

(Refer Slide Time: 24:12)

Gradient descent

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\ &= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w}_d \cdot \vec{x}_d) \\ &= \sum_{d \in D} (t_d - o_d) (-x_{i,d}) \\ \Delta w_i &= -\eta \frac{\partial E}{\partial w_i} = -\eta \sum_{d \in D} (t_d - o_d) (-x_{i,d}) = \eta \sum_{d \in D} (t_d - o_d) x_{i,d} \\ \Delta w_i &= \sum_{d \in D} (\eta (t_d - o_d) x_{i,d}) \end{aligned}$$

So, this method is used for deriving the delta rule not going through all the derivation, but you can finally, write down your training algorithm as this.

(Refer Slide Time: 24:32)

Gradient descent algorithm

Gradient-Descent (*training examples, η*)

Each training example is a pair $\langle x, t \rangle$: x is the vector of input values, and t is the target output value. η is the learning rate (e.g., .05).

- Initialize each w_i to some small random value
- Repeat until the termination condition is met
 1. Initialize each Δw_i to zero
 2. For each training example $\langle x, t \rangle$
 - Input x to the unit and compute the output o
 - For each linear unit weight w_i
 $\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$
 3. For each linear unit weight w_i
 $w_i \leftarrow w_i + \Delta w_i$
- At each iteration, consider reducing η

Also called
• LMS (Least Mean Square) rule
• Delta rule

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

I am sorry, after computing all this delta and everything you can comment like this initialize to 0, let us say random value and this is your update rule same as before an update weight this is known as the delta rule or the LMS rule.

This is guaranteed to converge to the correct value i am not going to the derivation that this is indeed. So, as I have said the rule is take the steepest descent look around which direction which change in delta value a w value delta w will give you least maximum reduction in error change it in that direction go on doing it till you can touch.

So, this is the gradient almost same functional form as the perceptron rule except for that you add up over your error function is sum up over all the training set otherwise earlier in the perceptron rule, it is the not sum, but single input error. So, this is the gradient rule in the next lecture, we will move on to more complex functions than a linear hyperplane.

Thank you.