19ECS 773: Deep Learning

Topic: Gradient Descent and Backpropagation

Unit I

April 19 2021, 2.00-2.50PM

Dr. Sireesha Rodda

Professor,

Dept of CSE,

GITAM Institute of Technology

GITAM Deemed to be University

Outline of the Presentation

- Introduction
- Historical Background
- Perceptron
- Back propagation algorithm
- Limitations and improvements
- Questions and Answers

Introduction

- Artificial Neural Networks are crude attempts to model the highly massive parallel and distributed processing we believe takes place in the brain.
- Back propagation, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks used in conjunction with an optimization method such as gradient descent.
- The method calculates the gradient of a loss function with respect to all the weights in the network.
- The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

Why do we need multi layer neural networks??

- Some input and output patterns can be easily learned by single-layer neural networks (i.e. perceptron). However, these single-layer perceptron cannot learn some relatively simple patterns, such as those that are not linearly separable.
- Single layer neural network can’t learn any abstract features of the input since it is limited to having only one layer. A multi-layered network overcomes this limitation as it can create internal representations and learn different features in each layer.
- Each higher layer learns more and more abstract features that can be used to classify the image. Each layer finds patterns in the layer below it and it is this ability to create internal representations that are independent of outside input that gives multi-layered networks their power.

Historical Background

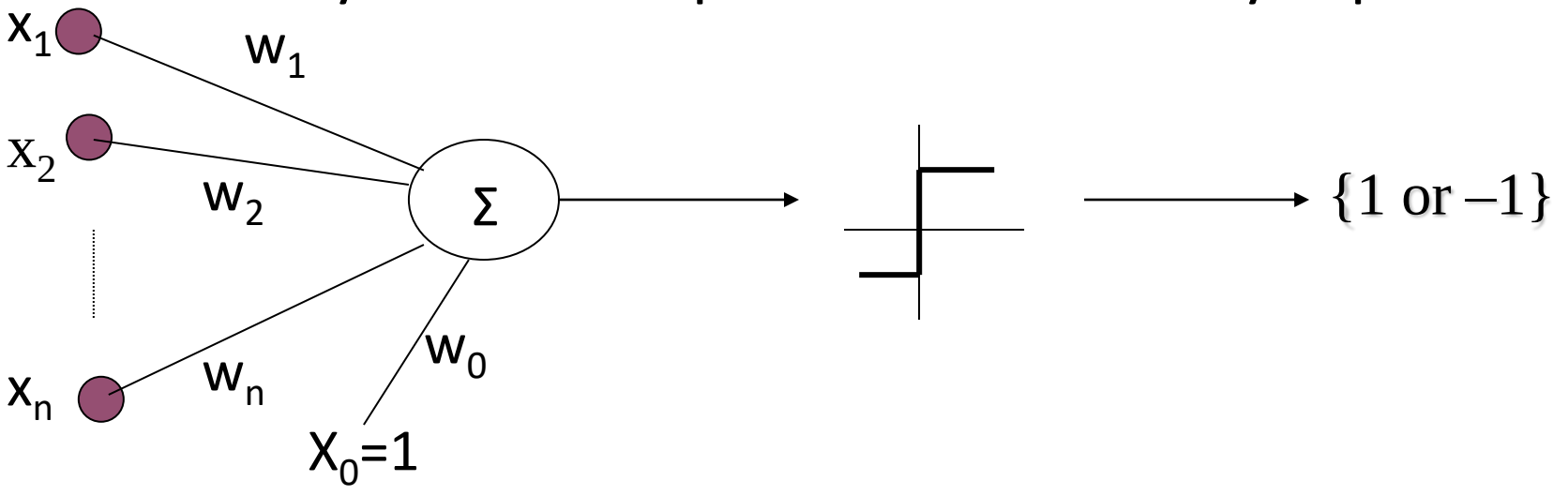
- Early attempts to implement artificial neural networks: McCulloch (Neuroscientist) and Pitts (Logician) (1943) [1]
  - Based on simple neurons (MCP neurons)
  - Based on logical functions
- Donald Hebb (1949) gave the hypothesis in his thesis “The Organization of Behavior”:
  - “Neural pathways are strengthened every time they are used.”
- Frank Rosenblatt (1958) created the perceptron, an algorithm for pattern recognition based on a two-layer computer learning network using simple addition and subtraction . [1]

Historical Background

- However, neural network research stagnated when Minsky and Papert (1969) criticized the idea of the perceptron discovering two key issues in neural networks: [1]
  - Could not solve the XOR problem.
  - Training time grows exponentially with the size of the input.
- Neural network research slowed until computers achieved greater processing power. Another key advance that came later was the back propagation algorithm which effectively solved the exclusive-OR problem (Werbos 1975) . [1]

Perceptron

- It is a step function based on a linear combination of real-valued inputs. If the combination is above a threshold it outputs a 1, otherwise it outputs a –1
- A perceptron can only learn examples that are linearly separable.



[3]

Delta Rule

- The delta rule is used for calculating the gradient that is used for updating the weights.
- We will try to minimize the following error:
  - $E = \frac{1}{2} \Sigma_i (t_i - o_i)^2$
- For a new training example $X = (x_1, x_2, \ldots, x_n)$, update each weight according to this rule:
  - $w_i = w_i + \Delta w_i$

    where, $\Delta w_i = -n * E'(w)/w_i$

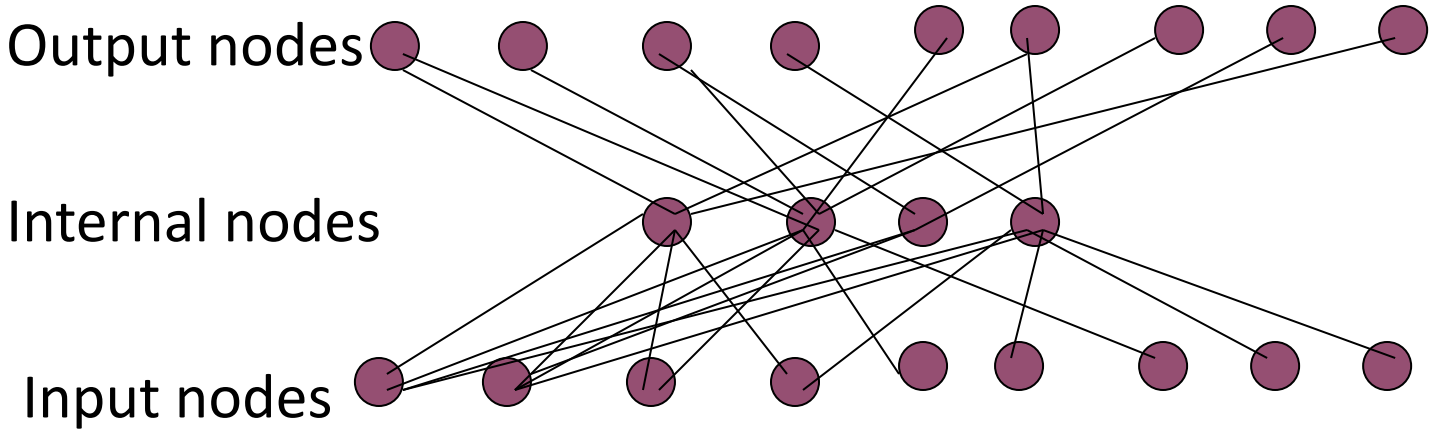
Delta Rule

- The derivative gives,
  - $E'(w)/w_i = \Sigma_i (t_i - o_i)*(- x_i)$

- So that gives us the following equation:
  - $\Delta w_i = \eta \Sigma_i (t_i - o_i) x_i$

Multi-layer neural networks

- In contrast to perceptrons, multilayer neural networks can not only learn multiple decision boundaries, but the boundaries may be nonlinear.



[3]

Learning non-linear boundaries

- To make nonlinear partitions on the space we need to define each unit as a nonlinear function (unlike the perceptron). One solution is to use the sigmoid (logistic) function. So,
- $$O(x1,x2,\ldots,xn) = \sigma (WX)$$

  where: $\sigma (WX) = 1 / 1 + e^{-WX}$ [3]

- We use the sigmoid function because of the following property,
  - $d \sigma(y) / dy = \sigma(y) (1 – \sigma(y))$

Back propagation Algorithm

- The back propagation learning algorithm can be divided into two phases:
- Phase 1: Propagation
  - Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
  - Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the input and output values) of all output and hidden neurons.
- Phase 2: Weight update
  - Multiply its output delta and input activation to get the gradient of the weight.
  - Subtract a ratio (percentage) of the gradient from the weight.

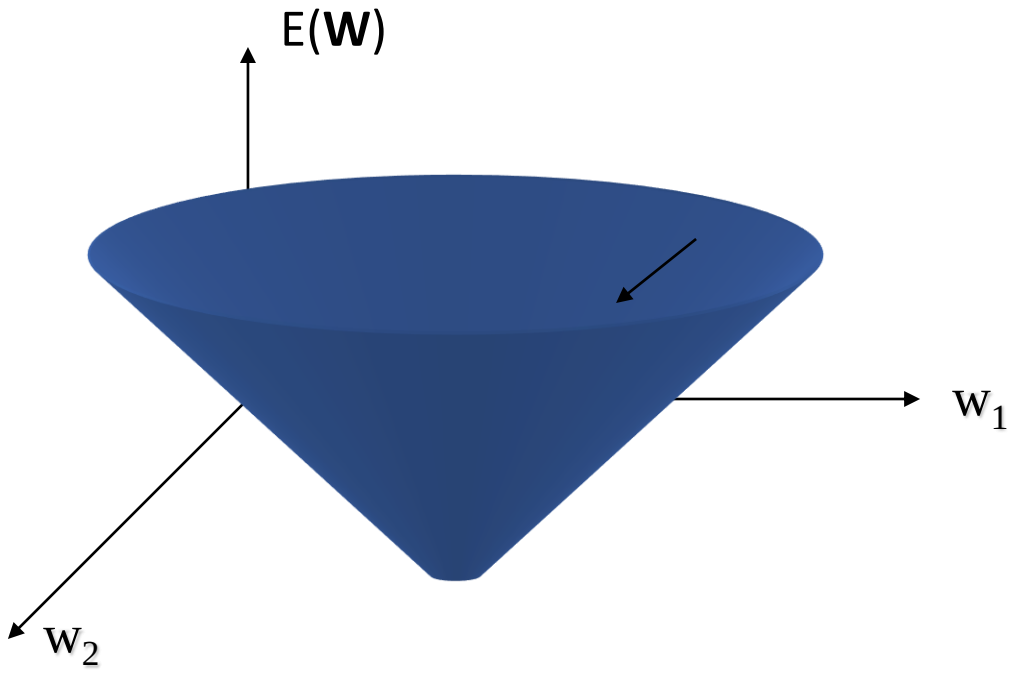
Back propagation Algorithm (contd.)

```
initialize network weights (often small random values)
do
   forEach training example ex
      prediction = neural-net-output(network, ex)  // forward pass
      actual = teacher-output(ex)
      compute error (prediction - actual) at the output units
      compute Δw_h for all weights from hidden layer to output layer  // backward pass
      compute Δw_i for all weights from input layer to hidden layer   // backward pass continued
      update network weights // input layer not modified by error estimate
until all examples classified correctly or another stopping criterion satisfied
return the network
```

[2]

What is gradient descent algorithm??

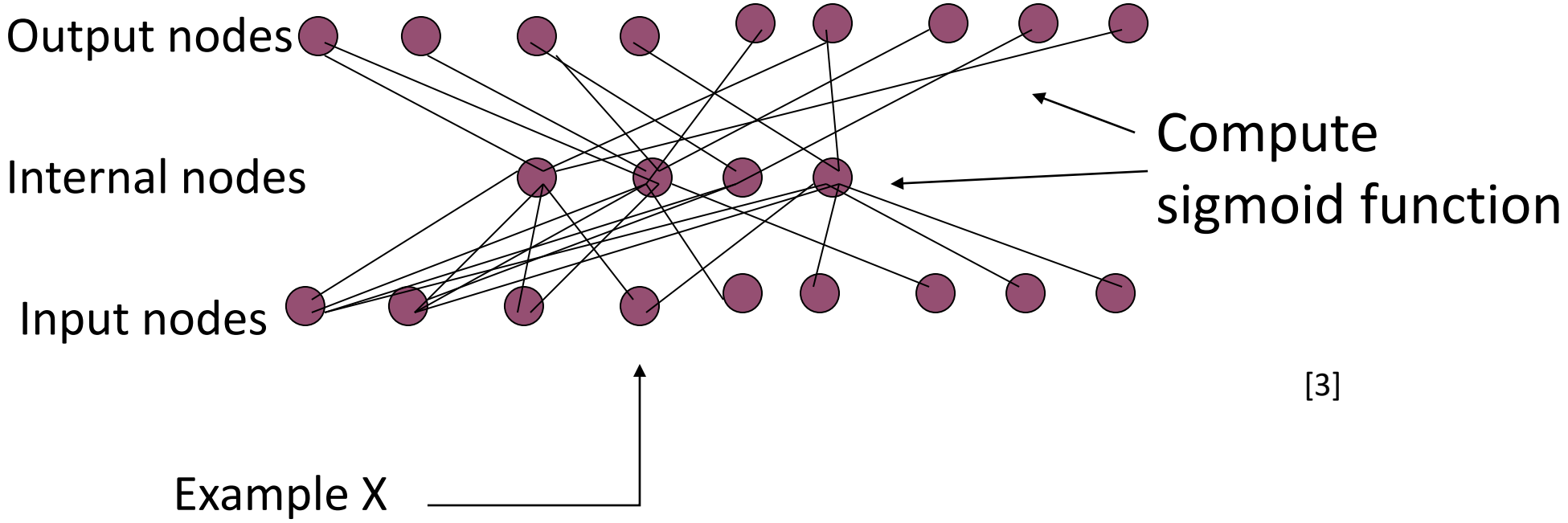
- Back propagation calculates the gradient of the error of the network regarding the network's modifiable weights.
- This gradient is almost always used in a simple stochastic gradient descent algorithm to find weights that minimize the error.



[3]

Propagating forward

- Given example X, compute the output of every node until we reach the output nodes:



[3]

Propagating Error Backward

- For each output node k compute the error:

$$\delta_k = O_k (1-O_k)(t_k - O_k)$$

- For each hidden unit h, calculate the error:

$$\delta_h = O_h (1-O_h) \Sigma_k W_{kh} \delta_k$$

- Update each network weight:

$$W_{ji} = W_{ji} + \Delta w_{ji}$$

where $\Delta w_{ji} = \eta \delta_j X_{ji}$ (W_{ji} and X_{ji} are the input and weight of node i to node j)

Number of Hidden Units

- The number of hidden units is related to the complexity of the decision boundary.
- If examples are easy to discriminate few nodes would be enough. Conversely complex problems require many internal nodes.
- A rule of thumb is to choose roughly m / 10 weights, where m is the number of training examples.

Learning Rates

- Different learning rates affect the performance of a neural network significantly.

- Optimal Learning Rate:
  - Leads to the error minimum in one learning step.

Learning Rates

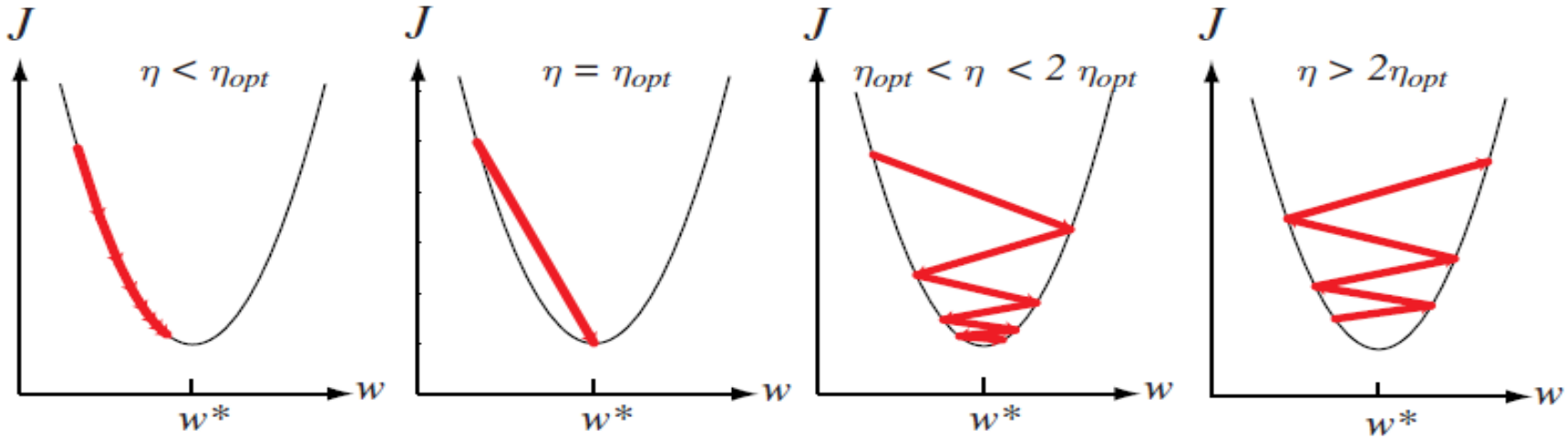


FIGURE 6.16. Gradient descent in a one-dimensional quadratic criterion with different learning rates. If $\eta < \eta_{opt}$, convergence is assured, but training can be needlessly slow. If $\eta = \eta_{opt}$, a single learning step suffices to find the error minimum. If $\eta_{opt} < \eta < 2\eta_{opt}$, the system will oscillate but nevertheless converge, but training is needlessly slow. If $\eta > 2\eta_{opt}$, the system diverges. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

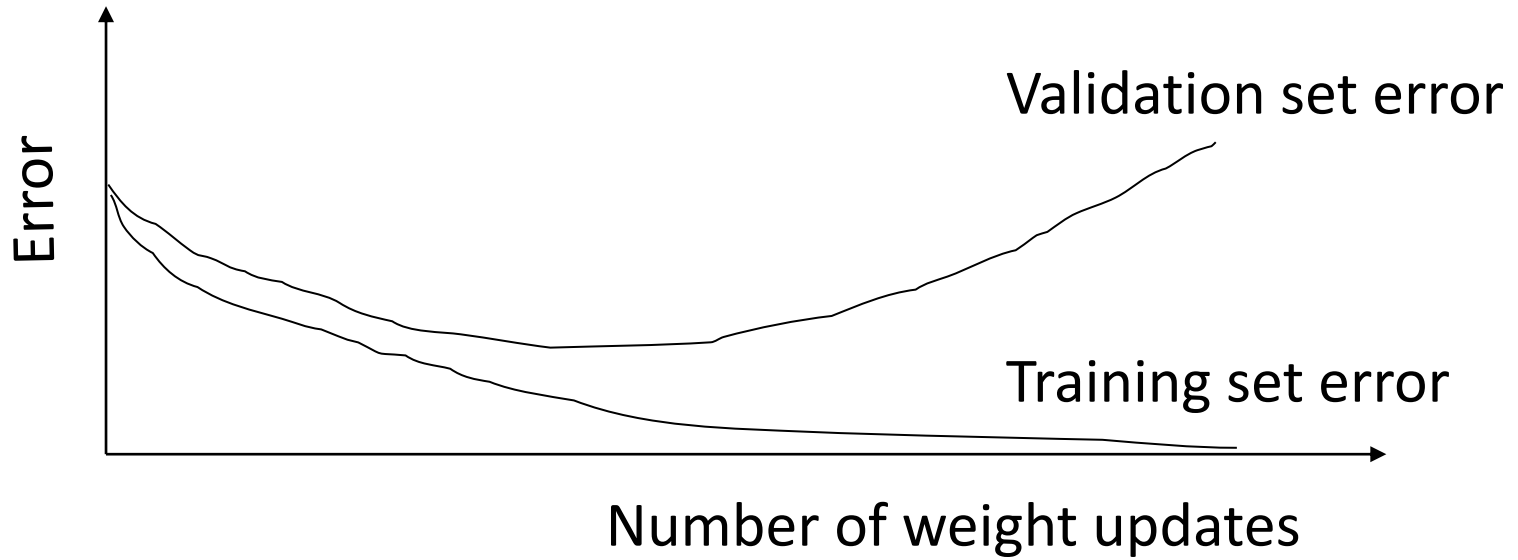
[3]

Limitations of the back propagation algorithm

- It is not guaranteed to find global minimum of the error function. It may get trapped in a local minima,
  - Improvements,
    - Add momentum.
    - Use stochastic gradient descent.
    - Use different networks with different initial values for the weights.

- Back propagation learning does not require normalization of input vectors; however, normalization could improve performance.
  - Standardize all features previous to training.

Generalization and Overfitting



[3]

- Solutions,
  - Use a validation set and stop until the error is small in this set.
  - Use 10 fold cross validation

References

- Artificial Neural Networks, https://en.wikipedia.org/wiki/Artificial_neural_network
- Back propagation Algorithm, https://en.wikipedia.org/wiki/Backpropagation
- Lecture Slides from Dr. Vilalta's machine learning class

Questions??

Thank you!!