Neural Networks

Lecture 11

Learning and Adaptation (2)

Md. Mijanur Rahman, Prof. Dr.

Dept. of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University, Bangladesh.

Web: www.mijanrahman.com | Email: mijanjkkniu@gmail.com; mijan@jkkniu.edu.bd

Chapter Contents

- **This chapter covers the following topics:**
  - Learning in ANN
  - Different Learning Paradigms
  - Offline or Online Learning
  - Training patterns and teaching input
  - **Learning curve and error measurement**
  - **Gradient Descent Algorithm and Optimization Procedures**
  - **Cost Function and The Learning Rate**
  - Different Learning Rules

Learning Curve and Error Measurement…

- The **learning curve** indicates the progress of the error, which can be determined in various ways. The motivation to create a learning curve is that such a curve can indicate whether the network is progressing or not.
- For this, the error should be normalized, i.e. represent **a distance measure between the correct and the current output** of the network.
- For example, we can take the same pattern-specific, **squared error with a prefactor**, which we are also going to use to derive the **backpropagation of error** (let Ω be output neurons and O the set of output neurons):

$$\mathrm{Err}_p = \frac{1}{2} \sum_{\Omega \in O} (t_\Omega - y_\Omega)^2$$

Learning Curve and Error Measurement…

- **Specific error:** The specific error Errp is based on a single training sample, which means it is generated online. Additionally, the **root mean square (RMS)** and the **Euclidean distance** are often used.
- **Euclidean distance:** The Euclidean distance between two vectors *t* and *y* is defined as

$$\mathrm{Err}_p = \sqrt{\sum_{\Omega \in O} (t_\Omega - y_\Omega)^2}.$$

- Generally, the **root mean square** is commonly used since it considers extreme outliers to a greater extent.
- **Root mean square (RMS):** The root mean square of two vectors t and y is defined as

$$\mathrm{Err}_p = \sqrt{\frac{\sum_{\Omega \in O} (t_\Omega - y_\Omega)^2}{|O|}}.$$

Learning Curve and Error Measurement

- As for offline learning, the total error in the course of one training epoch is interesting and useful, too:

- **Total error:** The total error Err is based on all training samples, that means it is generated offline.

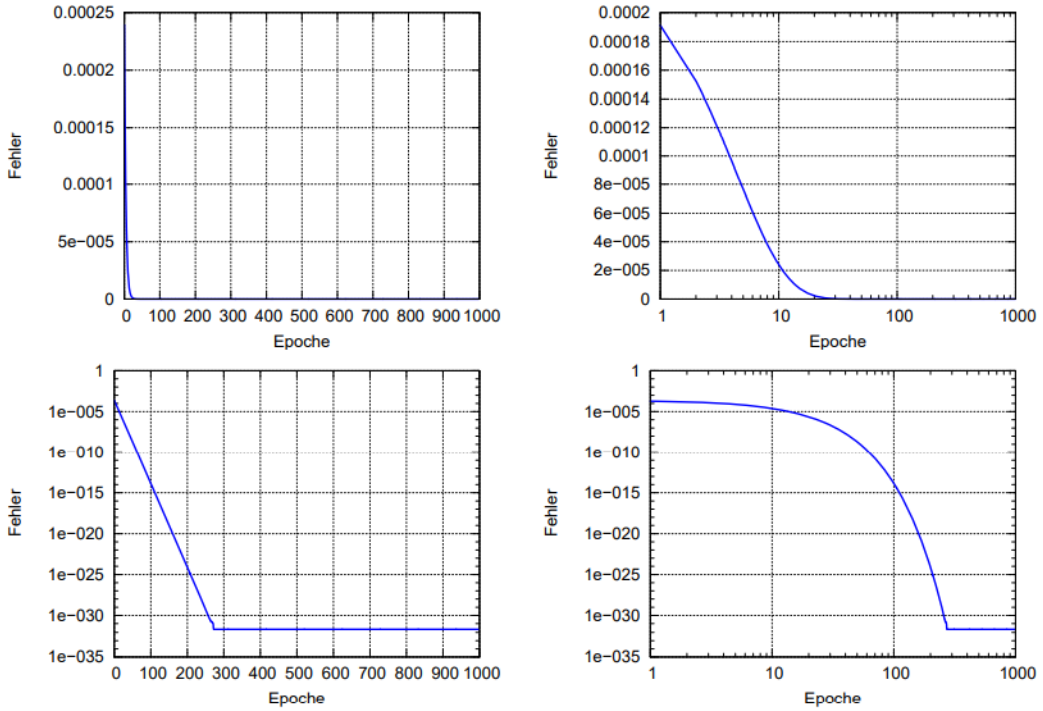
$$\mathrm{Err} = \sum_{p \in P} \mathrm{Err}_p$$

When do we stop learning?

- As the learning in ANN is an iterative process, the big question is: **When do we stop learning?**
  - Generally, the training is stopped when the user in front of **the learning computer thinks the error is small enough**.
  - Indeed, there is no easy answer, and thus I can once again only give you something to think about, which, however, depends on **a more objective view on the comparison of several learning curves**.
  - Confidence in the results, for example, is boosted when the network always reaches nearly the same final error rate **for different random initializations** – so repeated initialization and training will provide a more objective result.

When do we stop learning?

- In Figure, all four illustrations show the same (idealized, because very smooth) learning curve.



- It can be possible that a curve descending fast in the beginning can, after a longer time of learning, be overtaken by another curve.
- **Remember:** Larger error values are worse than the small ones.

Gradient Optimization Procedures…

- To establish the mathematical basis for some of the following learning procedures, it is required to describe **what is meant by gradient descent**:
  - The backpropagation of error learning procedure, for example, involves this mathematical basis and thus inherits the advantages and disadvantages of the gradient descent.
  - Gradient Descent is one of the most used machine learning algorithms in the industry.

Gradient Optimization Procedures

- Gradient descent procedures are generally used where we want to **maximize or minimize n-dimensional functions**.
- Gradient descent is an optimization algorithm used to **find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost)**.

- Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.

What is Gradient Descent?

- *It is a **function** that measures the performance of a model for any given data. **Cost Function** quantifies the error between predicted values and expected values and presents it in the form of a single real number.*
- **After making a hypothesis with initial parameters, we calculate the Cost function.** And with a goal to reduce the cost function, we **modify the parameters by using the Gradient descent algorithm** over the given data.
- Here's the mathematical representation for it:

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

What is Gradient Descent?

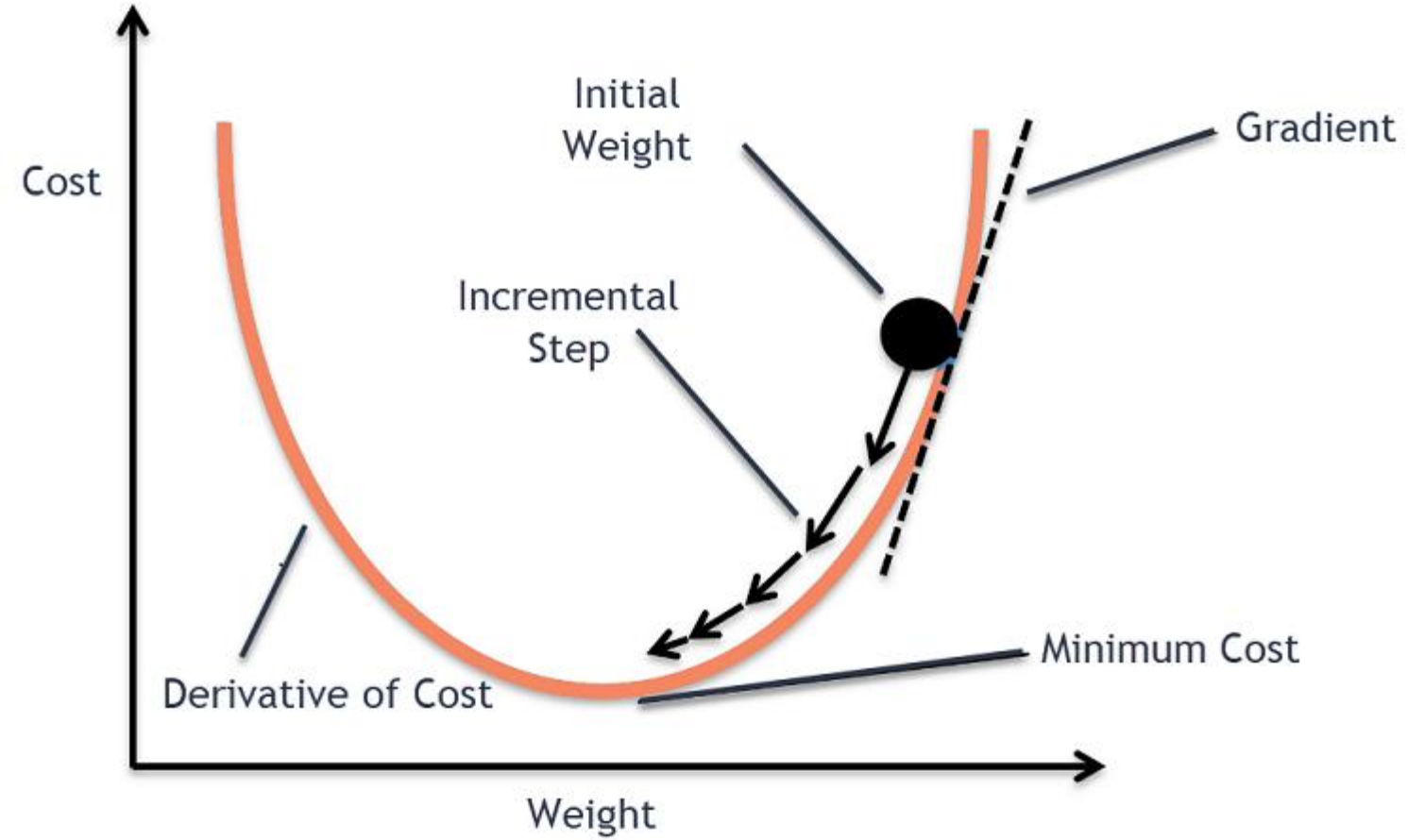
- Let's say you are playing a game where the players are at the top of a mountain, and they are asked to reach the lowest point of the mountain. Additionally, they are blindfolded. So, what approach do you think would make you reach the lake?
- The best way is to observe the ground and find where **the land descends**. From that position, take a step in the **descending direction and iterate this process** until we reach the **lowest point**.

- ***Gradient descent is an iterative optimization algorithm for finding the local minimum of a function.***

What is Gradient Descent?

- To find the **local minimum of a function using gradient descent**, we must take steps proportional to the **negative of the gradient (move away from the gradient)** of the function at the current point.
- If we take steps proportional to the **positive of the gradient (moving towards the gradient)**, we will approach a **local maximum** of the function, and the procedure is called **Gradient Ascent.**
- Gradient descent was originally proposed by **CAUCHY** in 1847. It is also known as **steepest descent**.

What is Gradient Descent?



Gradient Descent Algorithm

- The goal of the gradient descent algorithm is to minimize the given function (say cost function). To achieve this goal, it performs two steps iteratively:
  1. **Compute the gradient (slope)**, the first order derivative of the function at that point.
  2. **Make a step (move) in the direction opposite to the gradient**, opposite direction of slope increase from the current point by alpha times the gradient at that point.
- **Gradient descent algorithm:**

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$
$$(\text{for } j = 1 \text{ and } j = 0)$$
$$\}$$

- Alpha is called **Learning rate** – a tuning parameter in the optimization process. It decides the length of the steps.

Gradient Descent Algorithm

Plotting the Gradient descent algorithm:

- When we have a single parameter (theta), we can plot the dependent variable cost on the y-axis and theta on the x-axis. If there are two parameters, we can go with a 3-D plot, with cost on one axis and the two parameters (thetas) along the other two axes.

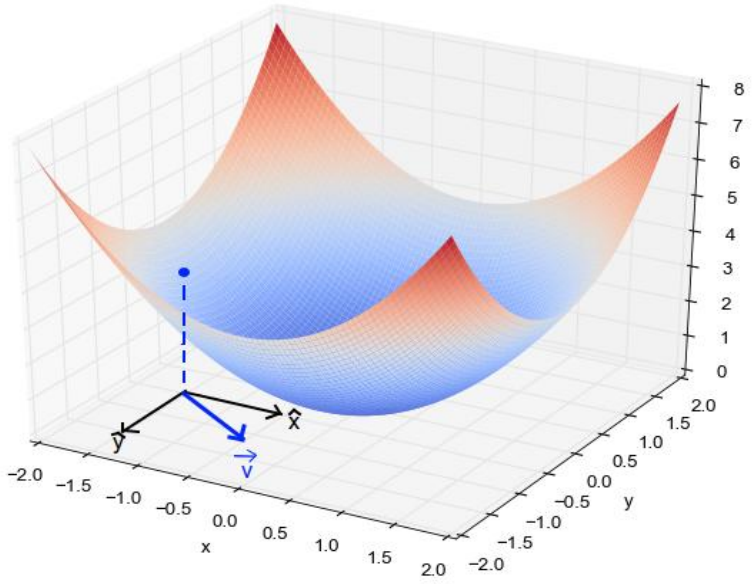


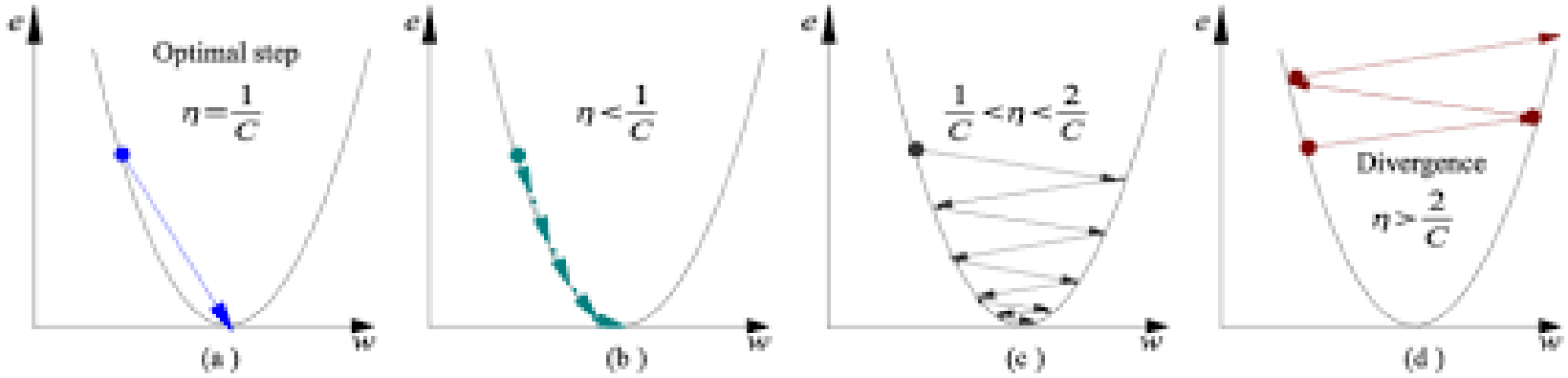
Fig: Cost along z-axis and parameters(thetas) along x-axis and y-axis

Alpha – The Learning Rate

- Alpha is called **Learning rate** – a tuning parameter in the optimization process. It decides the length of the steps.
- ***It must be chosen carefully to end up with local minima.***
  1. If the learning rate is too high, we might OVERSHOOT the minima and keep bouncing, without reaching the minima
  2. If the learning rate is too small, the training might turn out to be too long

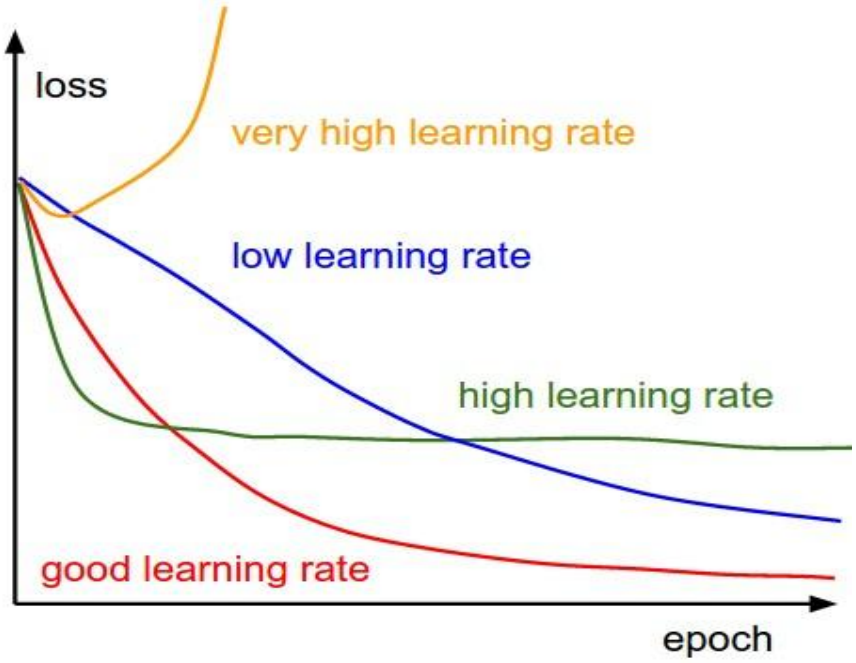
Alpha – The Learning Rate

- **From the figure:**
  a) Learning rate is optimal, model converges to the minimum
  b) Learning rate is too small, it takes more time but converges to the minimum
  c) Learning rate is higher than the optimal value, it overshoots but converges ($1/C < \eta < 2/C$)
  d) Learning rate is very large, it overshoots and diverges, moves away from the minima, performance decreases on learning



Alpha – The Learning Rate

- **Net learning according to the learning rate:**



- ***Note:*** *As the gradient decreases while moving towards the local minima, the size of the step decreases. So, the learning rate (alpha) can be constant over the optimization and need not be varied iteratively.*

Local Minima and Convergence of Cost Function

- The cost function may consist of **many minimum points**. **The gradient may settle on any one of the minima**, which depends on the initial point, i.e., initial parameters (theta), and the learning rate.
- Therefore, the optimization may **converge to different points with different starting points and learning rate**.

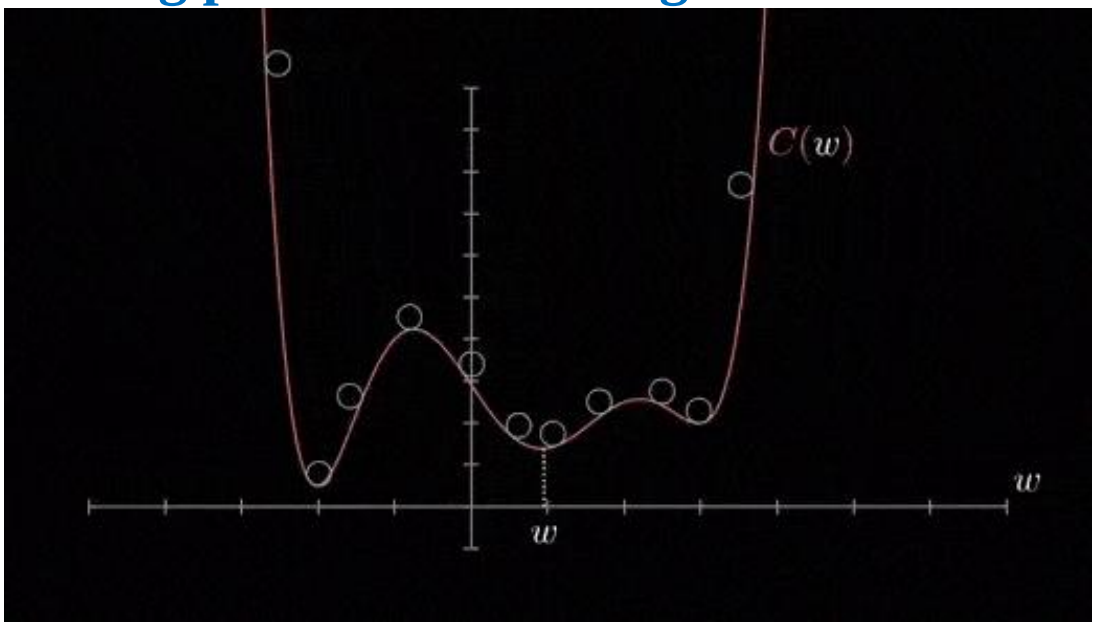


Fig: Convergence of cost function with different starting points

LEARNING AND ADAPTATION

To be continued…