CPE504

Artificial Neural Networks (ANNs)

3.0. SINGLE LAYERED NEURAL NETWORKS: FUNDAMENTALS

What you will learn

- Stochastic Gradient Descent (SGD)
- Simple Problem
- SGD Implementation (Online)
- SGD Implementation (Batch)
- Comparison
- Limitations of Perceptrons
- Summary

Stochastic Gradient Descent SGD

Recall, we are still considering ANNs form the **fundamental viewpoint of single-layer architectures** called **perceptrons**.

- **SGD**, the default numerical optimization method used to calculate the weight update, Δw_{ij}, is introduced in this section
- The **Stochastic** S, in the **GD** name, here, means the steepest descent of the first-order gradient is done in a random manner, with respect to the input data, used for training.
- In each iteration, the **GD** algorithm takes the **negative direction of the gradient** as the **descent direction** of the **objective function**. It then takes steps along this descent direction to **find a local minimum**.
- Note that **GD** is a **Line Search method. It is the simplest Newton-type method for nonlinear optimization.** The search direction is the opposite of the gradient. It does not require the **computation of second derivatives**, and it does not require **matrix storage**. The **computational cost per iteration** is **lower** compared to other Newton-type methods.
- Although, with **poor convergence rate** and **getting stuck at local minima point** problems. Surprisingly, this method **has proved satisfactory** for many problems in the **ANN domain**.

Stochastic Gradient Descent SGD

Online SGD

- calculates the error for each training data and adjusts the weights immediately.
- Say we have 100 training data points, the SGD adjusts the weights 100 times.
- The online SGD calculates the weight updates as:

$$\Delta w_{ij} = \alpha \delta_i x_j$$

- This **implies that the delta rule is the GD approach**.

Batch SGD

- each weight update is calculated for all errors of the training data, and the average of the weight updates is used for adjusting the weights.
- This method uses all of the training data and updates the weight only once.
- The batch SGD method calculates the weight update as:

$$\Delta w_{ij} = \frac{1}{N} \sum_{k=1}^{N} \Delta w_{ij}(k)$$

Stochastic Gradient Descent SGD

- where $\Delta \boldsymbol{w_{ij}}(\boldsymbol{k})$ is the **weight update for the k-th training data** and **N** is the **total**

number of the training data.

- The batch scheme, computes the averaged weight update calculation, therefore **consumes a significant amount of time for training**.

Mini Batch SGD

- The mini batch SGD is a **blend** of the **online** and **batch** methods. It **selects a part of the training dataset** and **uses them for training in the batch method**.
- Therefore, it calculates the weight updates of the selected data and trains the neural network with the averaged weight update.
- For example, if **20** arbitrary data points are selected out of **100** training data points, the batch method is applied to the 20 data points. In this case, a total of **five** ($\lfloor \mathbf{100/20} \rfloor$ **= 5**) **weight adjustments** are performed to complete the training process for all the data points.
- The **mini batch SGD**, when **an appropriate number of data points is selected**, obtains the benefits from both methods: **speed from the online SGD** and **stability from the batch SGD**. For this reason, it is **often utilized** in **Deep Learning**.

Stochastic Gradient Descent SGD

On Epoch:

- The **epoch** is the **number of completed training cycles** for **all** of the **training data**.

Online

- The **online** method **instantaneously adjusts the weight for each training data point**. It does not require addition or averaging of the weight updates. Therefore, **the online SGD is the simplest to implement**. The **number of training cycles per epoch** is **N**.

Batch

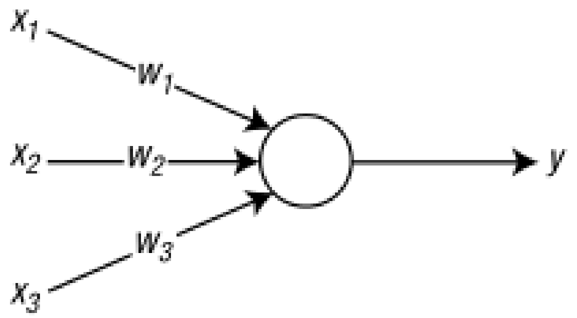
- The **batch** method utilizes **all** of the **N training data points** for **one training cycle.** Therefore **one training cycle** equals **one epoch**. The **number of training cycles per epoch** is **1**.

Mini Batch

- In **mini batch**, for **one epoch**, the **number of training cycles** varies **depending** on the **number of selected data points** used in each batch. *1 < batch size < N*
- The **number of training cycles per epoch** varies between **1 and N**.
- When the mini-batch is **1**, this corresponds to the **batch method.** *batch size = N*
- When the mini-batch is **N**, corresponds to the **online method**. *batch size = 1*

Simple Problem

- You are now ready to implement the delta rule (SGD) as a function.
- Consider a **neural network** that consists of **3 input nodes** and, as shown in**1 output node**
- The **simplified sigmoid function** is used as the **activation function of the output node**. We have **four (N=4) training data points or patterns** (input-correct output pair), as shown in the following table.



X1	X2	X3=bias	y
0	0	1	0
0	1	1	0
1	0	1	1
1	1	1	1

- The problem is a **combinational logic** problem (**y = x1** logic)
- By observation, we see that such a problem is linear.
- As it **is single-layered** and contains **simple training data**, the **code implementation is not complicated**.

SGD Implementation (Online, Batch, Mini Batch)

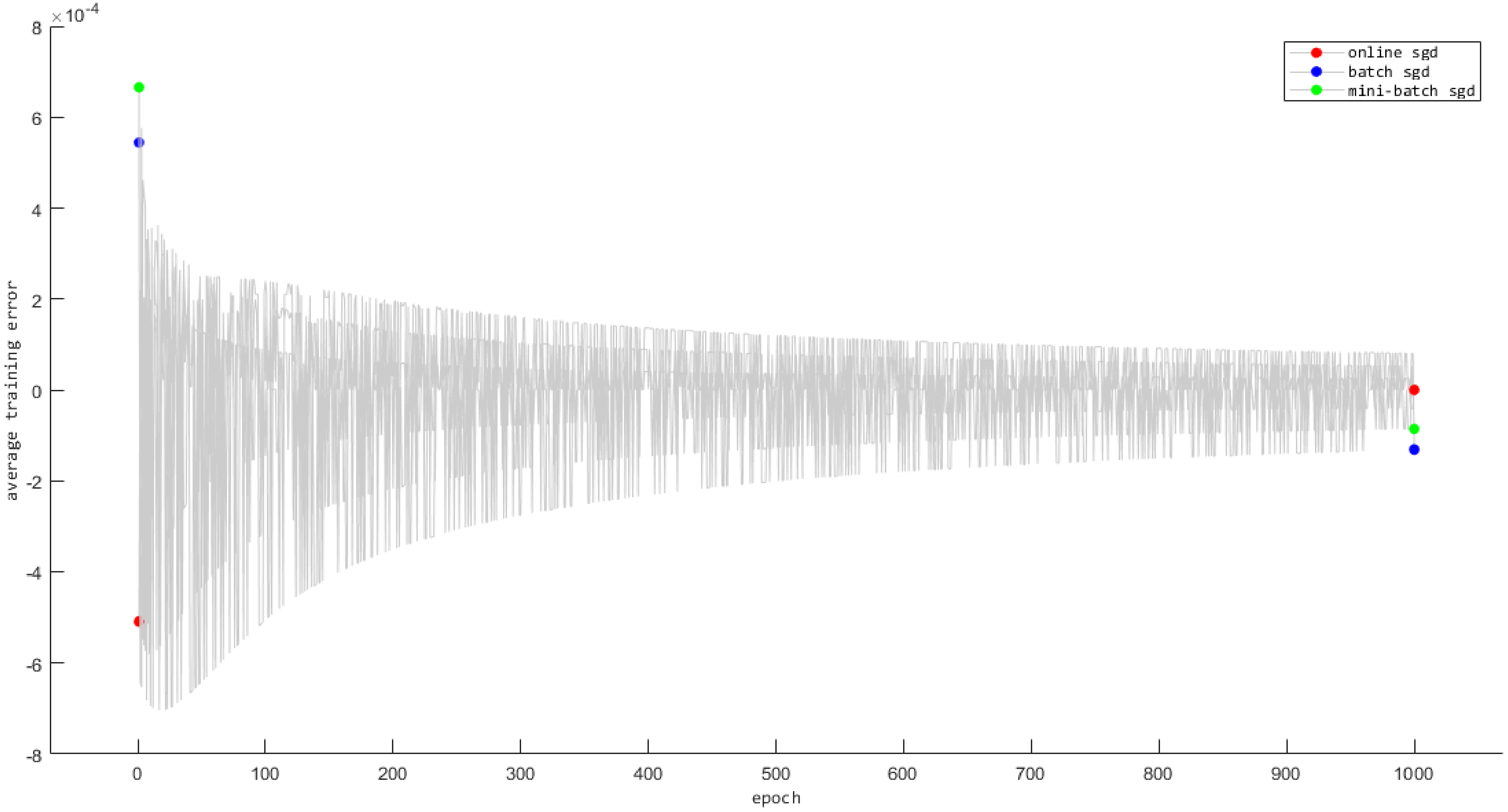
- **Functions** (API)

1. [y,dy] = **lsig_std**(x)
2. y = **perceptron**(x)
3. **sgd_obm**()
4. **train**(X,opts)
5. **infer**(X,opts)
6. main : opts

- **Visualize Training Performance**

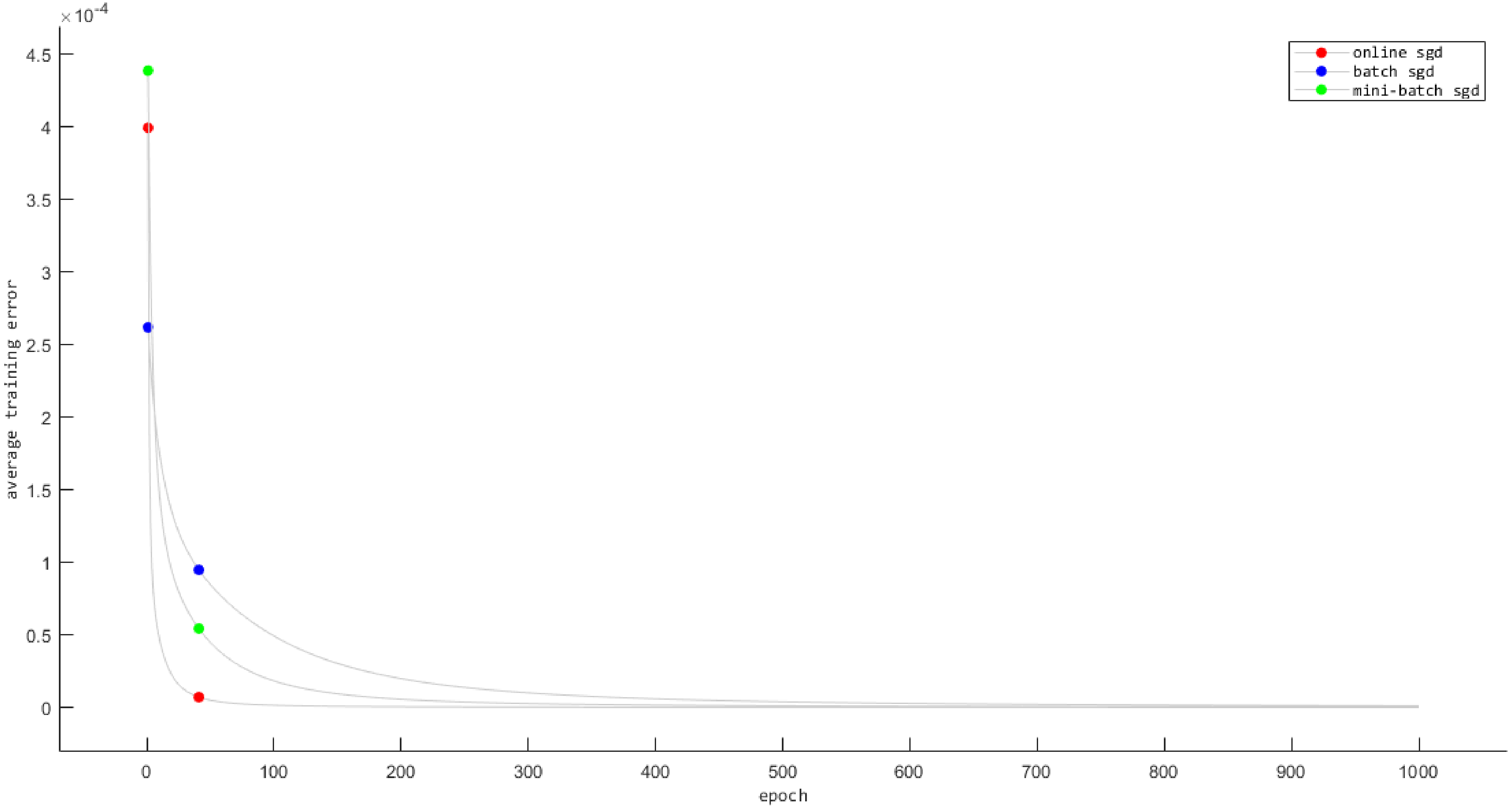
SGD Implementation (Online, Batch, Mini Batch)

- **Visualizer**



SGD Implementation (Online, Batch, Mini Batch)

- **Visualizer**



Comparison

- Typically, batch SGD method **requires more time to train** the neural network **to yield a similar level of accurac**y to that of the online SGD method.
- In other words, the **batch SGD method learns slowly**, while the online SGD results in faster reduction of the learning error than the batch; hence **online SGD learns faster**.
- This **averaging feature** of the **batch SGD** method is **the fundamental difference** that makes distinct, the batch from the online SGD method.
- The **averaging feature** of the **batch SGD method** allows the **training to be less sensitive to the training data**.
- The **mini-batch SGD** allows to generalize to the online and batch methods.

Limitations of Perceptrons

- This section presents the **critical reason** that the single-layer neural network had to evolve into a multi-layer neural network.
- Assume that we have four training data points, representing **XOR logic**. It **shouldn't cause any trouble, right**?
- We discover that even after training for long times, there is no improvement on the learnt output accuracy with respect to the. We can compare them with the correct outputs.
- **What happened**?
- One thing to notice is that that the **input-output relation is no more linear**. This **requires a nonlinear curve for classification**. This type of problem is said to be **linearly inseparable**.
- To put it simply, the **single-layer neural network as historically designed could only solve linearly separable problems**.
- Such single-layer neural networks are models that linearly divides the input data space.
- In order **to overcome this limitation** of the single-layer neural network**, we need more layers in the network**, to achieve what the single-layer neural network cannot.

Summary

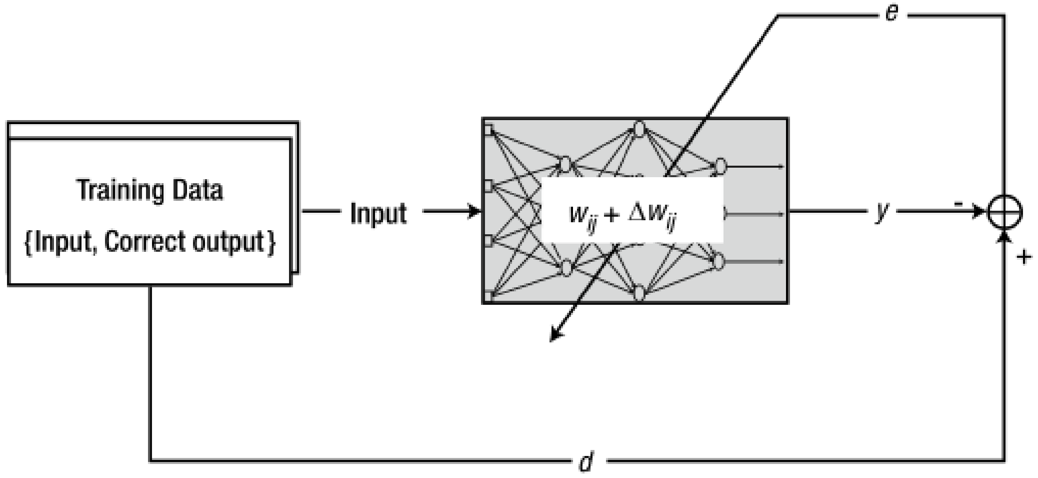
Recap:

- Notes on **SGD and its schemes**.
- **Implementation** of the SGD schemes
- **Comparisons** of the SGD schemes
- **Limitations** of the Perceptron
- The single-layer neural network is applicable only to **linearly separable** types of problems. Therefore, the single-layer neural network (perceptron) as designed from the literature **had very limited usages**.
- The **multi-layer neural network** (multiple hidden layers) was then developed to overcome the essential limitations of the single-layer neural network.

Summary

Recap:

- **Setting the learning rate correctly is important.**
- **Note that:** we only considered the learning rule from the viewpoint of the **single-layer ANN.**
- **This supervised learning or training process for the ANN is illustrated as below**



Tools

Recommended Languages

- MATLAB (Fast Matrix Prototyping)
- JavaScript (Language of the Web)

Instructions to Student

- **All ANN functions will be written from scratch in form of custom libraries**
- **Learn to transfer maths to software.**
- **Copying of another person's code (or work) will be heavily penalized.**

Recommended Texts

Main Texts

- **MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence** by Phil Kim
- **Pattern Recognition and Machine Learning** by Christopher M. Bishop
- **Understanding Machine Learning: From Theory to Algorithms** by Shai Shalev-Shwartz and Shai Ben-David
- **PATTERNS, PREDICTIONS, AND ACTIONS: A story about machine learning** by Moritz Hardt and Benjamin Recht
- **Mathematics for Machine Learning** by Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong

Good luck!