18CSE751 – Introduction to Machine Learning

Lecture 11: Variants in Gradient Descent & MSE1

Dr.Vani Vasudevan

Professor –CSE, NMIT

OUTLINE

- LOCAL MINIMA
- PICKING UP MOMENTUM
- VARIANTS IN GRADIENT DESCENT…
- OTHER IMPROVEMENTS
- MSE1 REVISION :
  - UNIT I & UNIT II

LOCAL MINIMA…

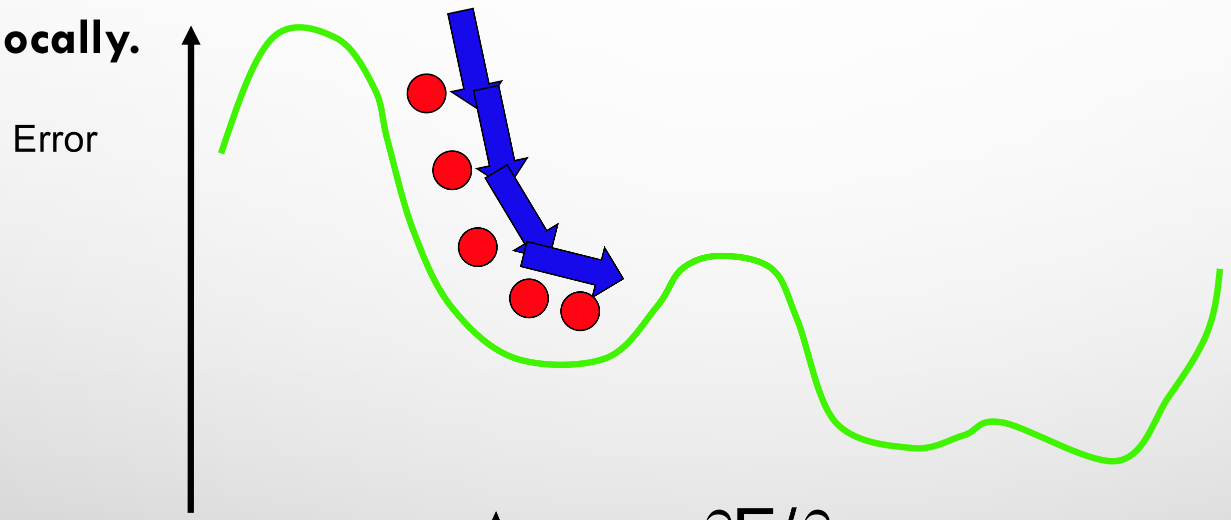
- The driving force behind the learning rule is the **minimisation of the network error by gradient descent (using the derivative of the error function to make the error smaller)**

- performing an optimisation: we are adapting the values of the weights in order to minimise the error function.

- By approximating the gradient of the error and following downhill ends up at the bottom of the slope.
- Following the **slope downhill only guarantees to end up at a local minimum,** a point that is lower than those close to it.

LOCAL MINIMA

- If we imagine a ball rolling down a hill, it will settle at the bottom of a dip. However, there is no guarantee that it will have stopped at the lowest point—**only the lowest point locally.**



$$\Delta w_i = -\eta \, \partial E / \partial w_i$$

- There may be a much lower point over the next hill, but the ball can't see that, **and it doesn't have enough energy to climb over the hill and find the global minimum**

PICKING UP MOMENTUM…

- Commonly done finding the global minimum
    - By trying out several different starting points
    - By training several different networks
- Adding momentum
    - can help to avoid local minima,
    - makes the dynamics of the optimisation more stable, improving convergence

$$w^{t}_{\zeta\kappa} \leftarrow w^{t-1}_{\zeta\kappa} + \eta\delta_o(\kappa)a^{\text{hidden}}_{\zeta} + \alpha\Delta w^{t-1}_{\zeta\kappa},$$

PICKING UP MOMENTUM…

$$w^{t}_{\zeta\kappa} \leftarrow w^{t-1}_{\zeta\kappa} + \eta\delta_o(\kappa)a^{\text{hidden}}_{\zeta} + \alpha\Delta w^{t-1}_{\zeta\kappa},$$

- Where t is used to indicate the current update and t − 1 is the previous one.
- $\Delta w^{t-1}_{\zeta\kappa}$ is the previous update that we made to the weights
- So $\Delta w^{t}_{\zeta\kappa} = \eta\delta_o(\kappa)a^{\text{hidden}}_{\zeta} + \alpha\Delta w^{t-1}_{\zeta\kappa}$
- $0 < \alpha < 1$ is the momentum constant. Typically, a value of $\alpha = 0.9$ is used. This is a very easy addition to the code and can **improve the speed of learning a lot.**

PICKING UP MOMENTUM

- Another parameter that can be added is known as **weight decay**.
  - This reduces the size of the weights as the number of iterations increases.
  - small weights are better since they lead to a network that is closer to linear (since they are close to zero, they

    are in the region where the sigmoid is increasing linearly),
- Only those weights that are essential to the non-linear learning should be large.
- After each learning iteration through all of the input patterns, every weight is multiplied by some constant $\mathbf{0 < \epsilon < 1}$. This makes the network simpler and can often produce improved results,
- But occasionally it can make the learning significantly worse, so it should be used with care.
- Setting the value of ϵ is typically done experimentally.

MINIBATCHES AND STOCHASTIC GRADIENT DESCENT…

- To reiterate,
- **Batch algorithm converges to a local minimum faster than the sequential algorithm**
- **Sequential algorithm** computes the error for each input individually and then does a weight update, but it is sometimes **less likely to get stuck in local minima.**
- Reason,
- **Batch algorithm** makes a better estimate of the steepest descent direction, so that the direction it chooses to go is a good one, but this just **leads to a local minimum**.

MINIBATCHES

- A **minibatch method** is to find some happy middle ground between the two.
- Split the training set into random batches
  - Estimating the gradient based on one of the subsets of the training set
  - Performing a weight update,
  - And then using the next subset to estimate a new gradient and using that for the weight update,
- Until all the training set have been used.
- The training set are then randomly shuffled into new batches and the next iteration takes place.
- **If the batches are small, then there is often a reasonable degree of error in the gradient estimate, and so the optimisation has the chance to escape from local minima, though at the cost of heading in the wrong direction.**

STOCHASTIC GRADIENT DESCENT

- A more extreme version of the minibatch idea is
  - **to use just one piece of data** to estimate the the gradient at each iteration of the algorithm,
  - to pick that piece of data uniformly at random from the training set.
  - So, a single input vector is chosen from the training set, and the output
  - Hence the error for that one vector computed, and this is used to estimate the gradient and so update the weights.
- A new random input vector (which could be the same as the previous one) is then chosen and the process repeated.

MINIBATCHES AND STOCHASTIC GRADIENT DESCENT

- This is known as **stochastic gradient descent(SGD)**, and can be used for any gradient descent problem, not just the MLP.
- It is often used if the training set is very large, since it would be very expensive to use the whole dataset to estimate the gradient in that case.

OTHER IMPROVEMENTS

- There are a few other things that can be done **to improve the convergence and behaviour of the back-propagation algorithm.**

1. **To reduce the learning rate** as the algorithm progresses.
    - Reason : Network should only be making large-scale changes to the weights at the beginning when the weights are random; if it is still making large weight changes later, then something is wrong.

2. Something that results in much larger performance gains: to **include information about the second derivatives** of the error with respect to the weights.

MSE 1

Unit –I (T1 – Chapters 1&2)

Introduction : Machine Learning, **Types of Machine Learning**, Machine Learning Process, Supervised Learning, **Examples of Machine Learning Applications,**

Machine Learning Preliminaries: Weight Space, Curse of Dimensionality, Testing Machine Learning Algorithms: **Overfitting, Training, Testing, and Validation Sets, Confusion Matrix, Accuracy Metrics,** ROC Curve, **Unbalanced Datasets, Measurement Precision,**

Basic Statistics : Averages, Variance, Covariance, Gaussian, **Bias, Variance Tradeoff**

Unit –II (T1 – Chapters 3 & 4)

Neurons, Neural Networks: The Brain and the Neuron, Neural Networks, The Perceptron, **Training a Perceptron, Learning Boolean Functions, Linear Separability, Multilayer Perceptron** : The Multi-layer Perceptron Algorithm, Initialising the Weights , Different Output Activation Functions, **Backpropagation Algorithm, Sequential and Batch Training, local minima, picking up momentum, minibatches and stochastic gradient descent,** other improvements

REFERENCE

1. STEPHAN MARSLAND, **MACHINE LEARNING, AN ALGORITHMIC PERSPECTIVE,** CRC PRESS SECOND EDITION, 2015.