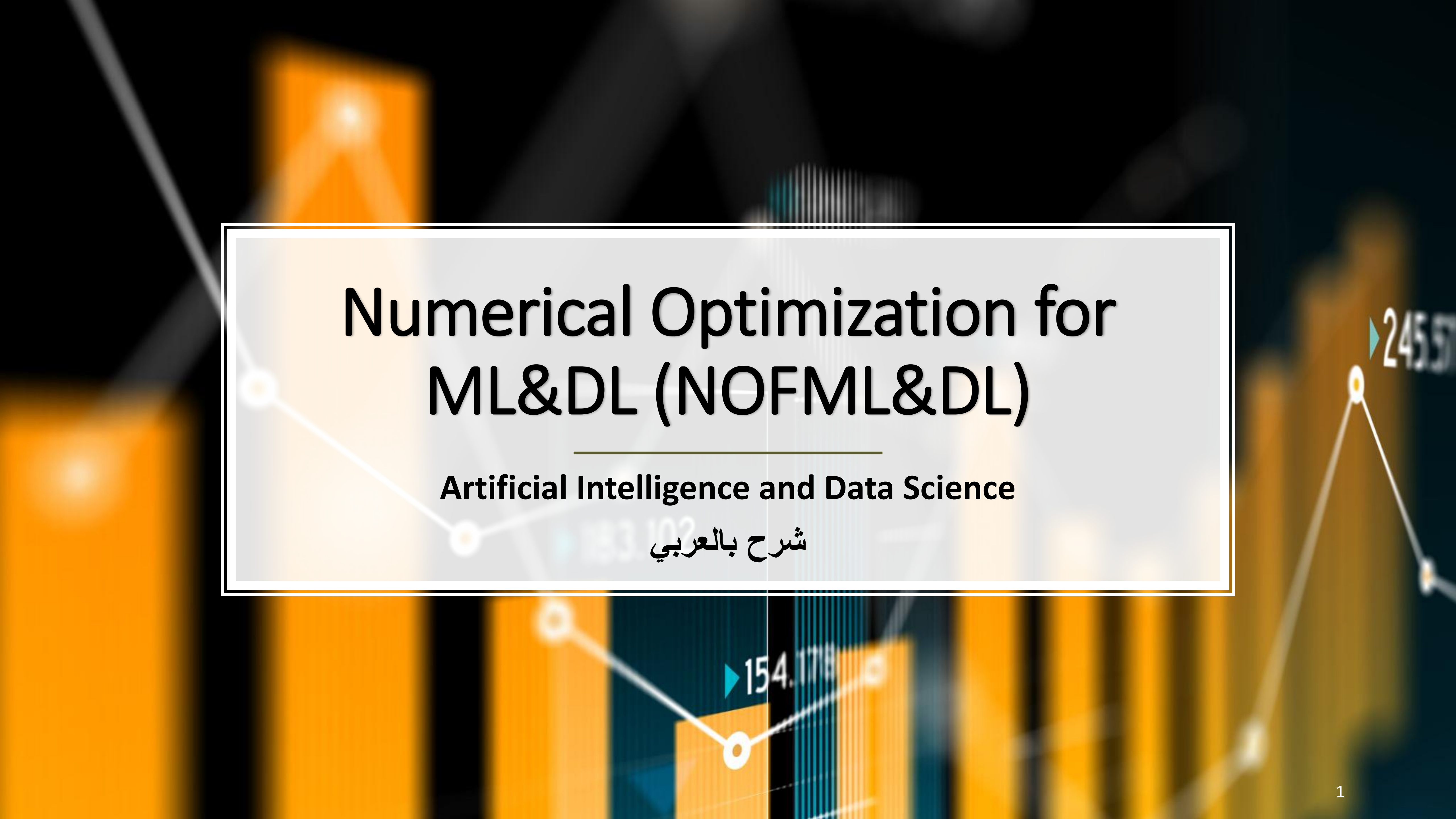


Numerical Optimization for ML&DL (NOFML&DL)

Artificial Intelligence and Data Science

شرح بالعربي

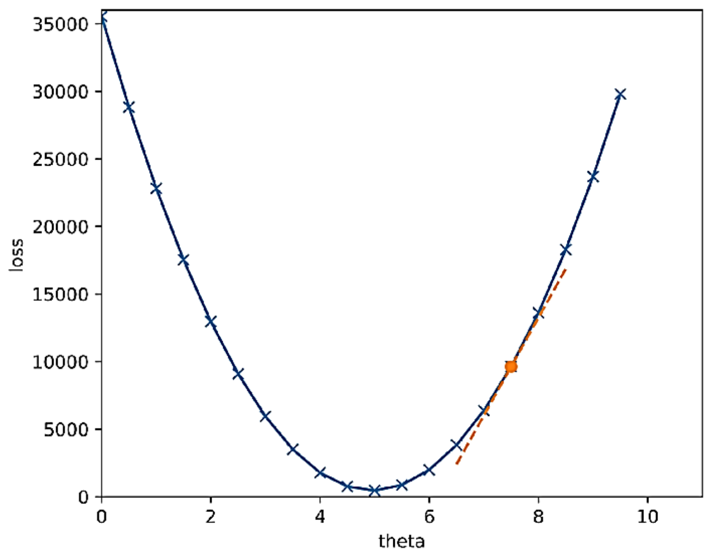
Agenda

Batch GD Problems.

GD Variants.

Stochastic GD.

Mini-Batch GD.



repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

}

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

}

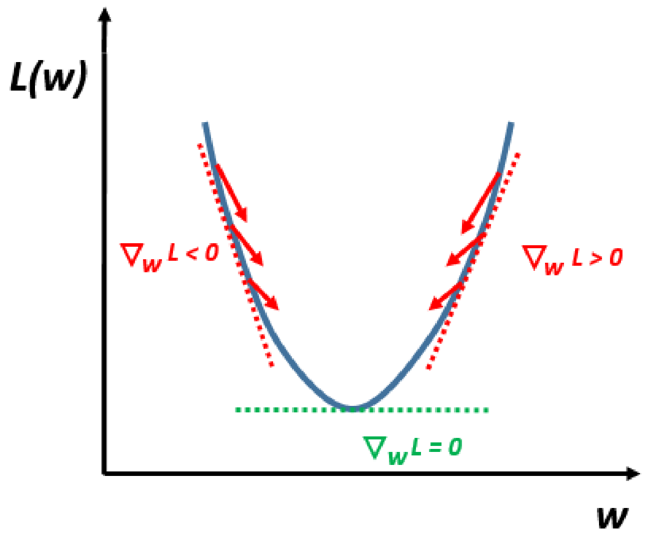
Linear Regression Model

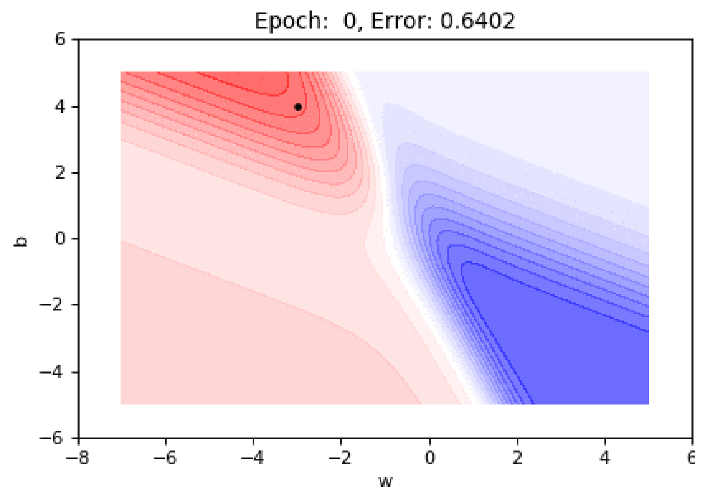
$$h_\theta(x) = \theta_0 + \theta_1 x$$

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

Batch/Vanilla GD Whole Picture

- Batch (vanilla) gradient descent, computes the gradient of the cost function w.r.t. the parameters **θ** for the entire training dataset:
- $\Theta = \Theta - \boldsymbol{\alpha} \nabla_{\Theta} \boldsymbol{J}(\Theta)$





$$w = w - \eta \nabla w \,;\; b = b - \eta \nabla b$$

where $\nabla w = \frac{\partial L(w)}{\partial w}$ and $\nabla b = \frac{\partial L(b)}{\partial b}$

Batch/Vanilla GD Whole Picture

- **GD and Backpropagation** algorithms are used to train artificial neural networks **(ANN)**. i.e., update weights and biases.
- Those are key algorithms in **Deep learning**.

Batch GD Problems

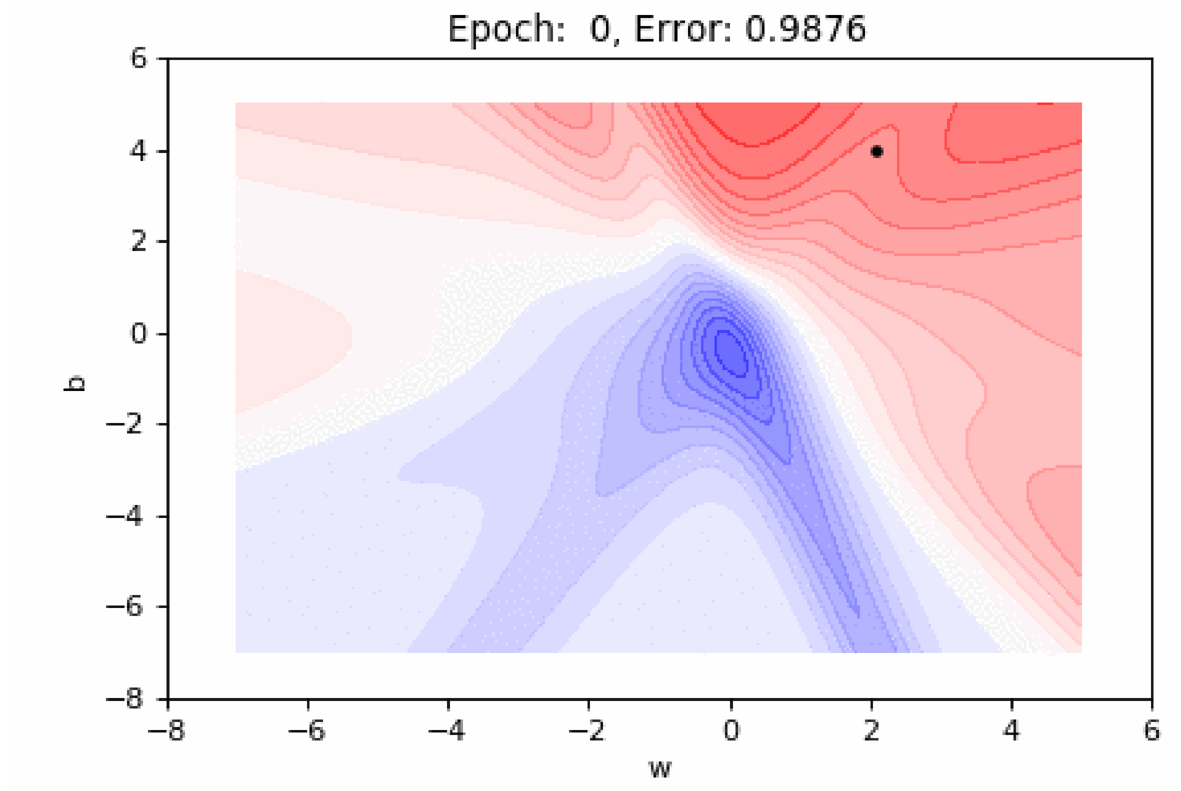
- ***Standard Gradient descent*** updates the *parameters* only after each epoch i.e., after calculating the *derivatives* for all the observations it updates the *parameters*. This phenomenon may lead to the following ***problems:***
  - It can be very slow for very large datasets because only one-time update for each epoch. Large number of **epochs** is required to have a substantial number of updates.
  - For large datasets, the vectorization of data doesn't fit into **memory**.
  - For non-convex surfaces, it may only find the **local minimums.**
- **Different variants of GD can address these challenges**

Stochastic GD (SGD)

- ***Stochastic gradient descent***
- updates the parameters for each observation which leads to a greater number of updates.

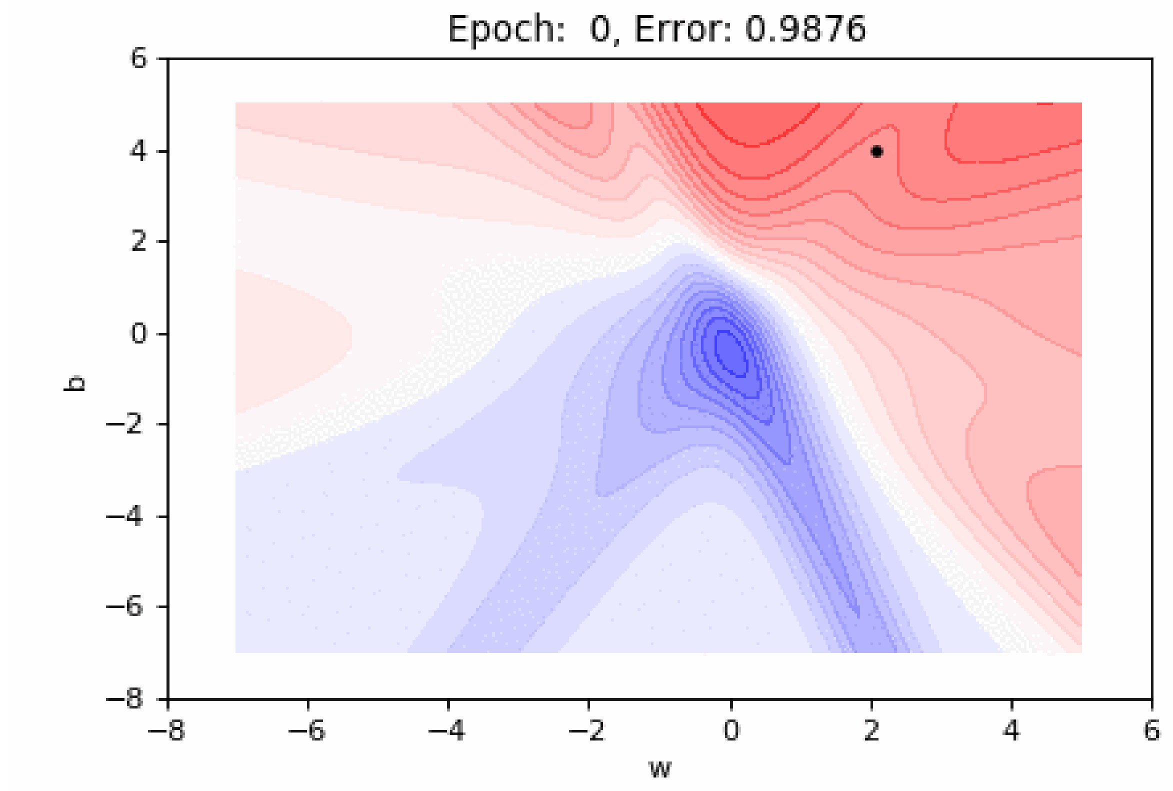
$$\Theta = \Theta - \alpha \nabla_{\Theta} J\left(\Theta; x^{(i)}; y^{(i)}\right)$$

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



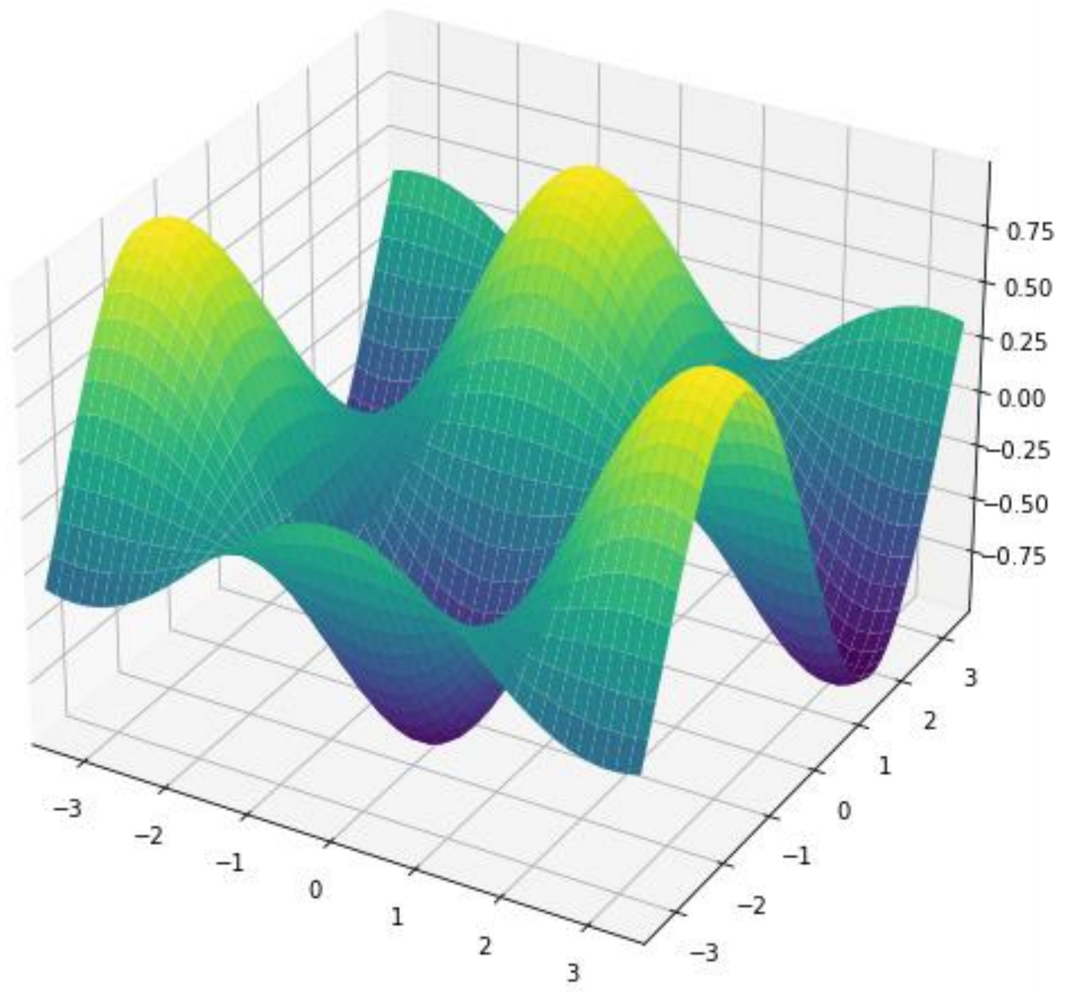
Stochastic GD (SGD)

- ***Disadvantages of SGD:***
  - Due to frequent fluctuations, it will keep overshooting near to the desired exact minima.
  - Add noise to the learning process i.e., the variance becomes large since we only use 1 example for each learning step.
  - We can’t utilize vectorization over 1 example.



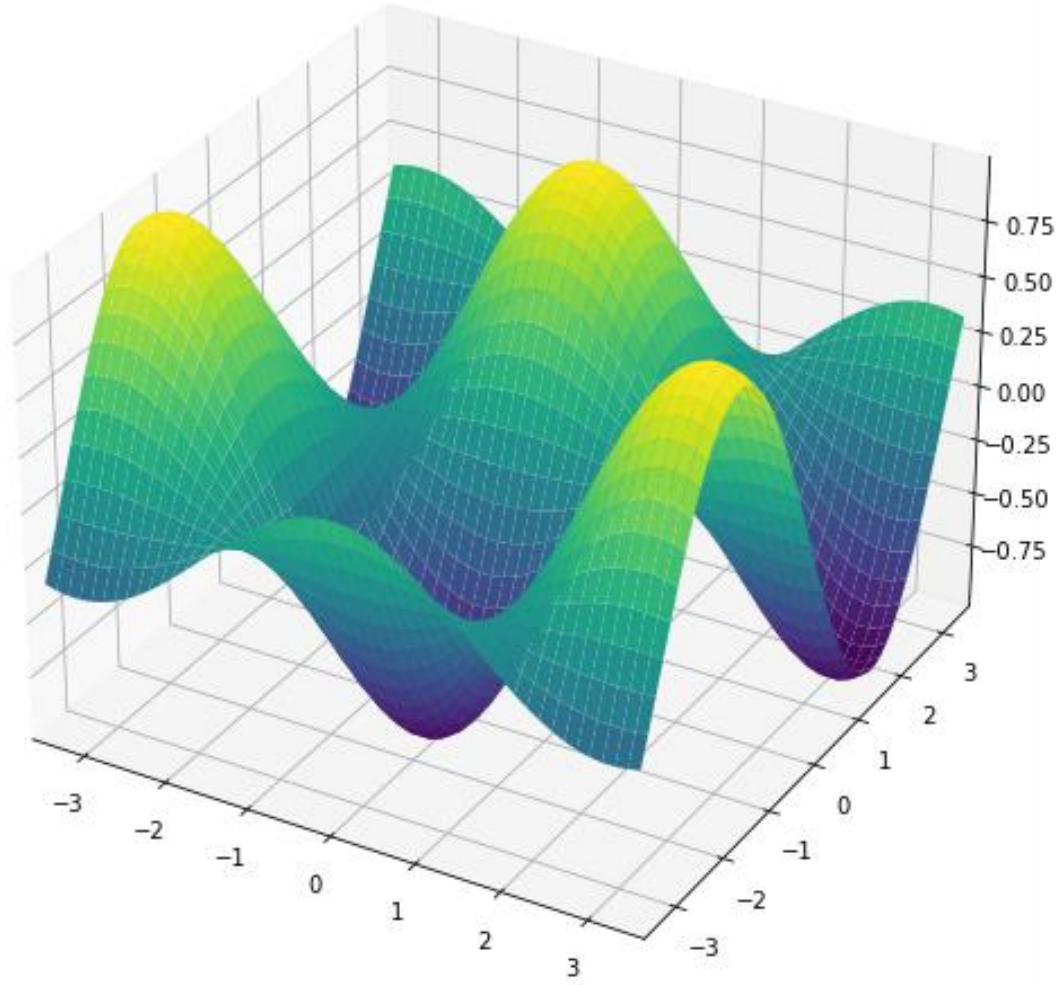
Stochastic GD (SGD)

- **SGD** works better than batch gradient descent for error manifolds that have lots of local maxima/minima. In this case, the somewhat noisier gradient calculated using the reduced number of samples tends to jerk the model out of local minima into a region that hopefully is more optimal.



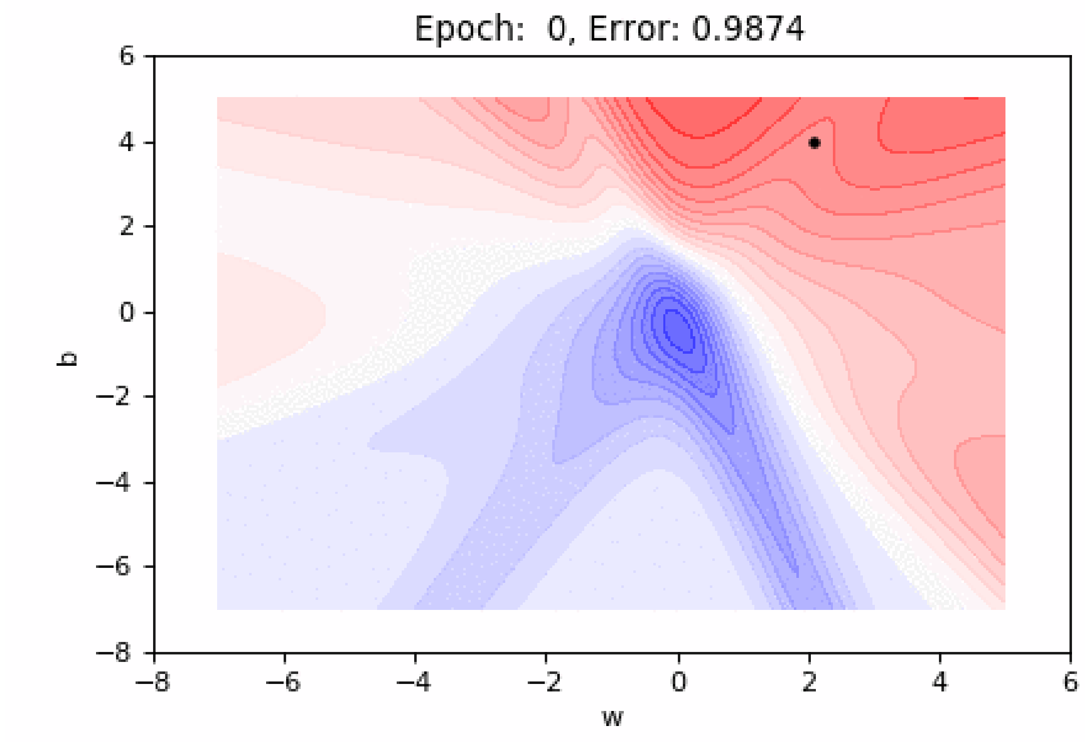
Stochastic GD (SGD)

- However, **minibatch** is the optimum between **Batch** and **SGD**.



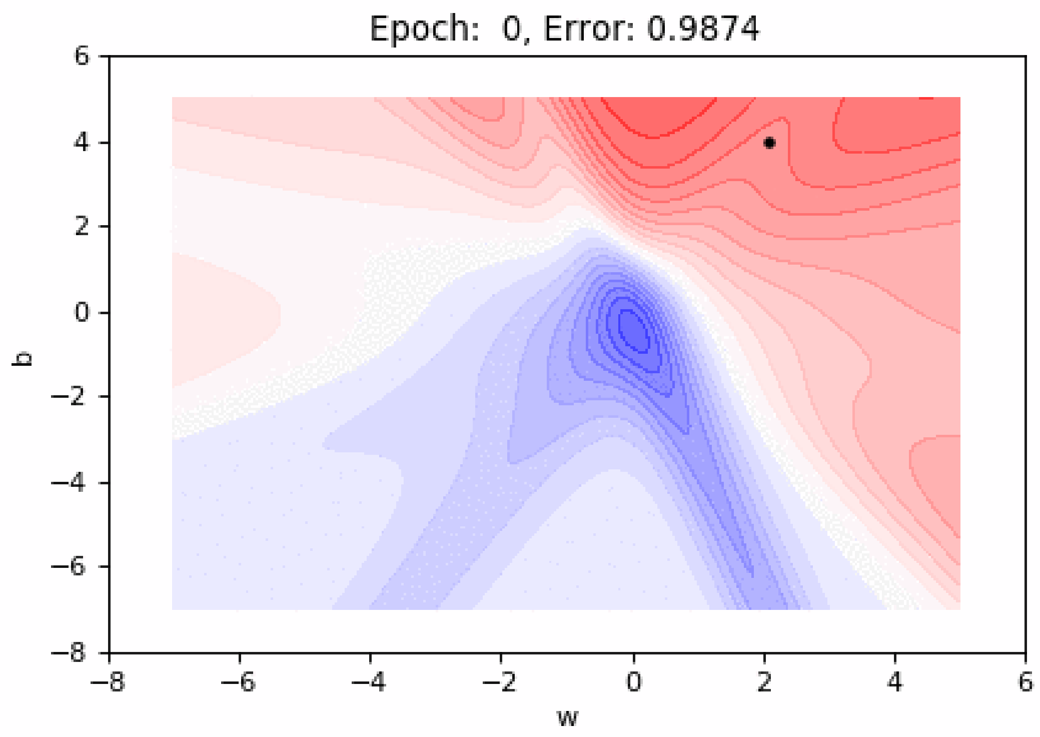
Mini Batch GD

- Instead of going over all examples, Mini-batch Gradient Descent sums up over lower number of examples based on the batch size. Therefore, learning happens on each mini-batch of **b** examples:
- $\Theta = \Theta - \boldsymbol{\alpha} \boldsymbol{\nabla}_{\Theta} \boldsymbol{J}\left(\Theta; \boldsymbol{x}^{(i:i+b)}; \boldsymbol{y}^{(i:i+b)}\right)$
- $\boldsymbol{J}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1) = \frac{1}{2b} \sum_{i=1}^{b} \left(\boldsymbol{h}_{\theta}\left(\boldsymbol{x}^{(i)}\right) - \boldsymbol{y}^{(i)}\right)^2$



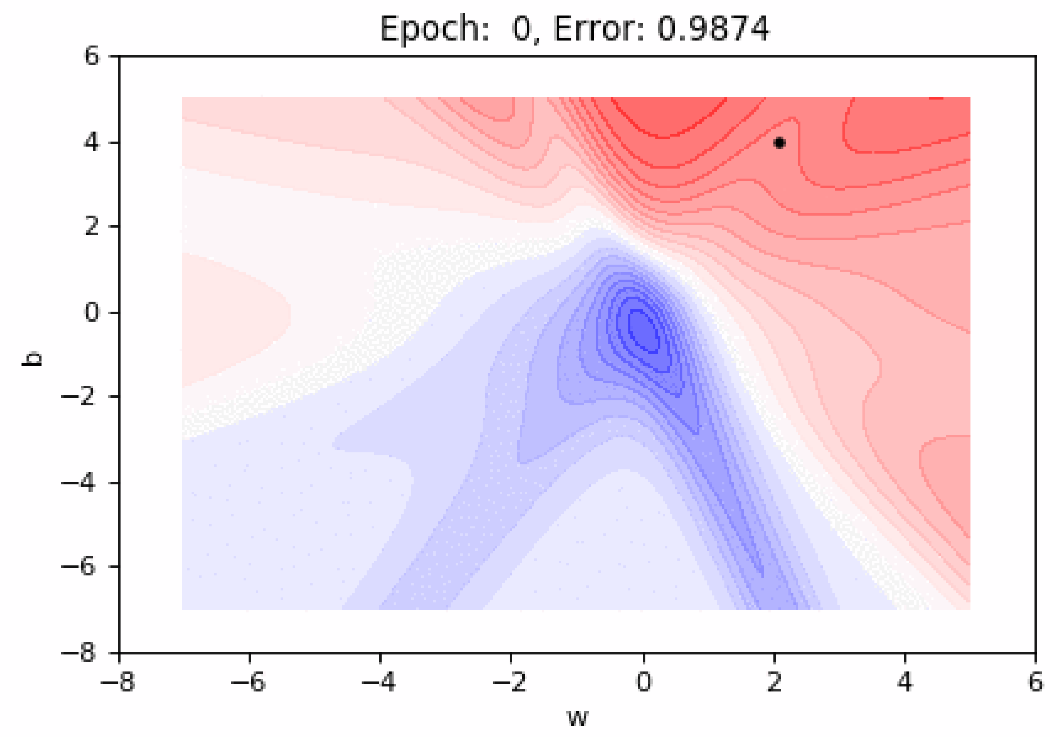
Mini Batch GD

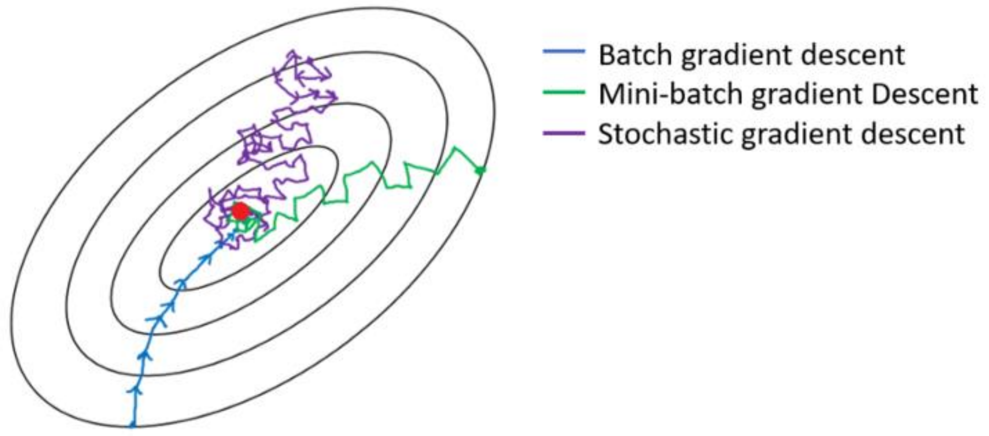
- **Advantages of Mini-batch GD:**
  - Updates are less noisy compared to SGD which leads to better convergence.
  - A high number of updates in a single epoch compared to GD so a smaller number of epochs are required for large datasets.
  - Fits very well to the processor memory which makes computing faster.

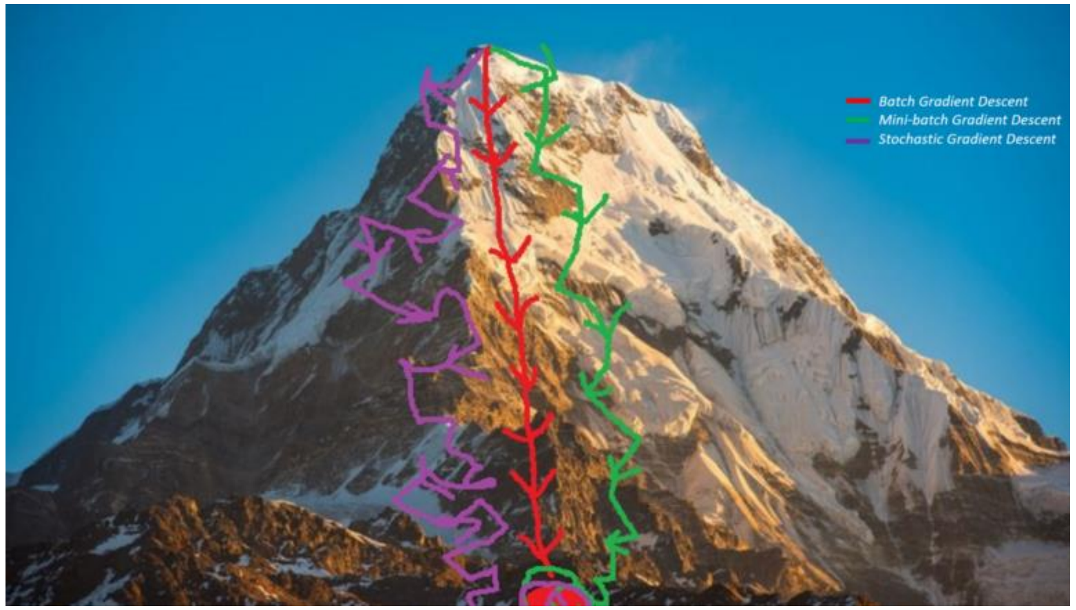


Mini Batch GD

- ***Note:*** The batch size is something we can tune. It is usually chosen as power of 2 such as 32, 64, 128, 256, 512, etc.
- ***Implementation Note: for both SGD and minibatch it is preferable to shuffle data before using it.***




Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent


Batch Gradient Descent
Mini-batch Gradient Descent
Stochastic Gradient Descent

GD Variants

GD Challenges

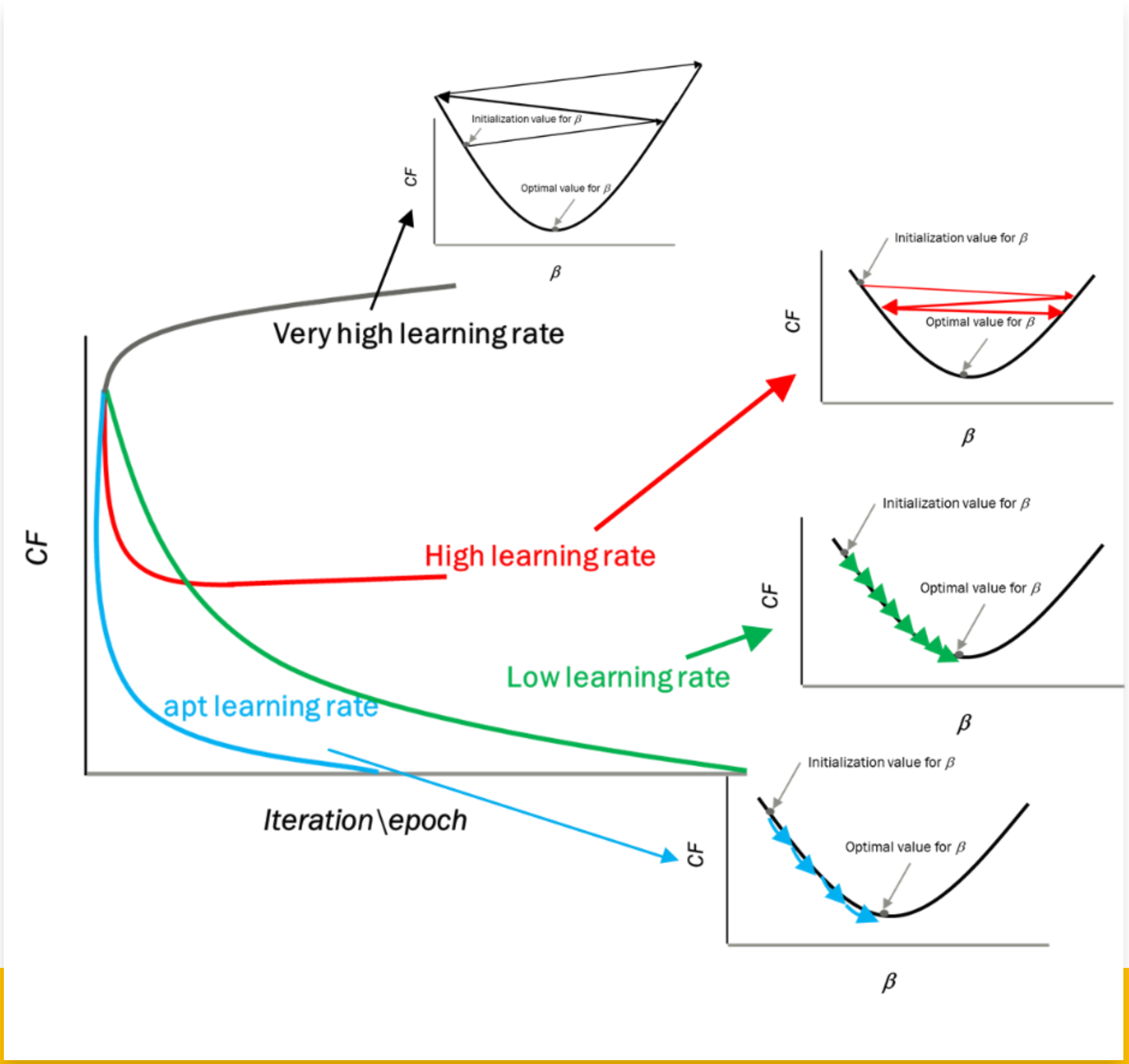
How to find a good value for the learning rate?

What to do in case of local minima?

How to solve the vanishing gradient problem?

Learning Rate

- $\theta_{t+1} = \theta_t - \alpha * gradient$



Initialization value for β
Optimal value for β
CF
β
Very high learning rate
Initialization value for β
Optimal value for β
CF
β
High learning rate
Initialization value for β
Optimal value for β
CF
β
Low learning rate
apt learning rate
CF
Iteration\epoch
Initialization value for β
Optimal value for β
CF
β

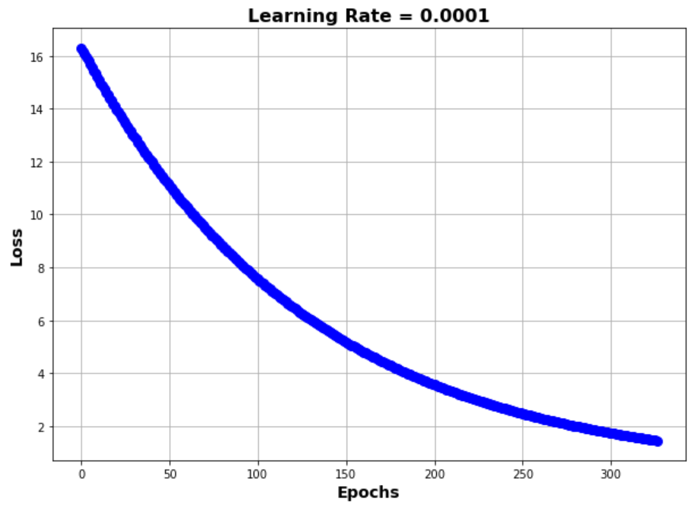
How to find a good value for the learning rate?

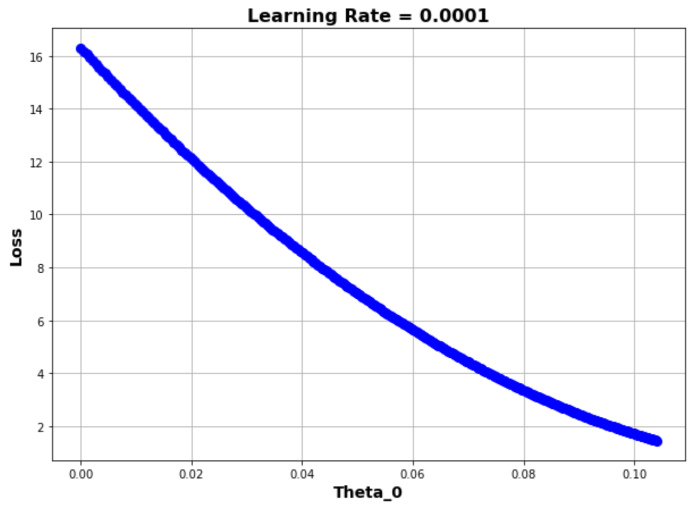
Unfortunately, there is no magic bullet to find the perfect learning rate.

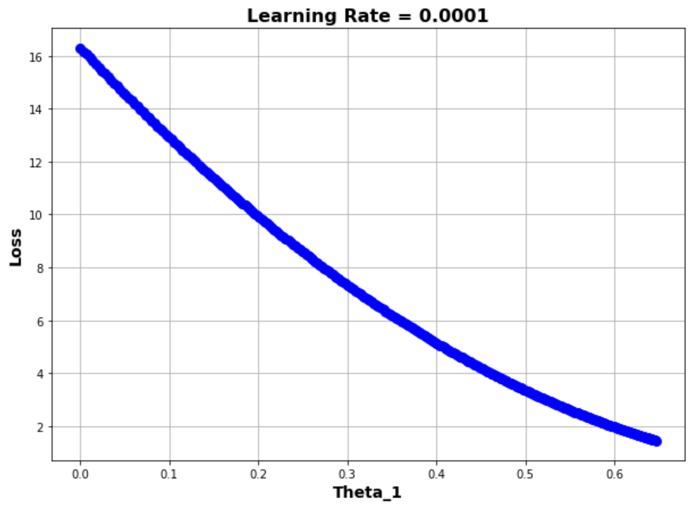
To find a good value, you have to test several values and pick the best.

How to find a good value for the learning rate?

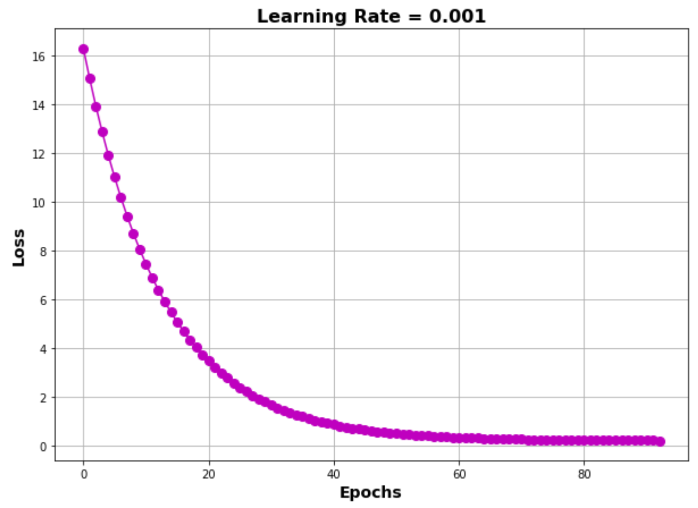
- **Advices to choose the learning rate:**
  - Plot cost function with epochs (iterations) and check if it is decreasing.
  - Convergence check $\boldsymbol{cost(i-1)}$ – $\boldsymbol{cost(i)} < \mathbf{0.001}$.
  - Try range of $\boldsymbol{\alpha}$ e.g., 0.001, 0.01,0.1,1 then plot cost vs. epochs and check for rapid and smooth conversion. Then you can select another $\boldsymbol{\alpha}$ close the value in that range. **e.g.,** if **0.001** is fine and **0.01** is bad you can try values in between such as **0.005**
  - Plot parameters vs. cost. Sometimes large learning rate implies many number of iterations due to parameter oscillation (cost function overshooting). Although, the decrease of cost function, overshooting increases number of iterations.

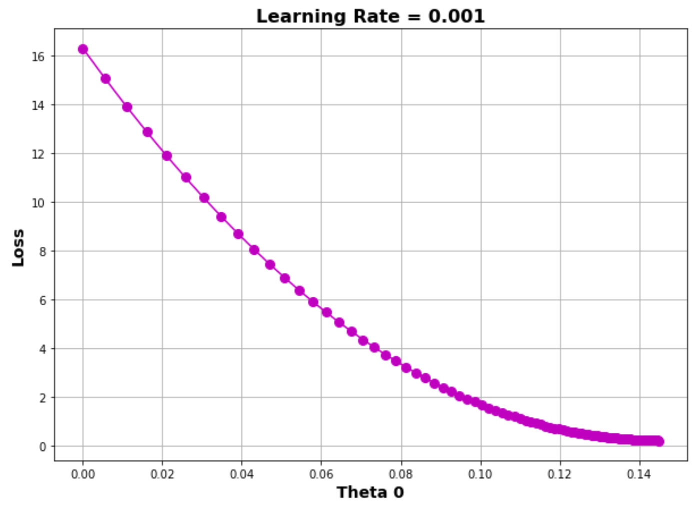


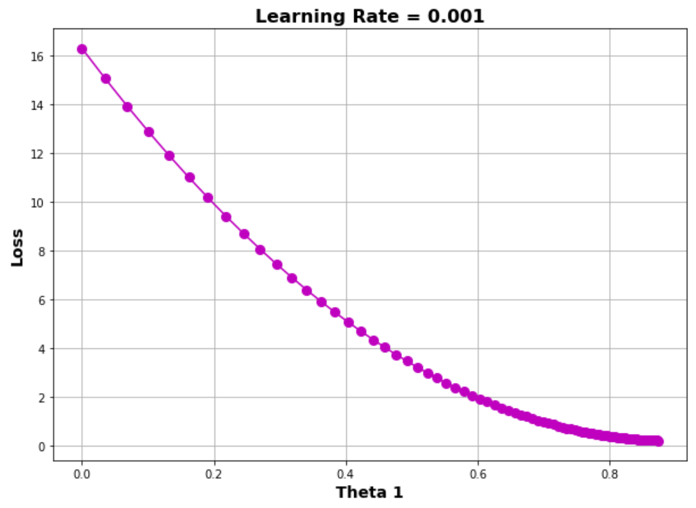




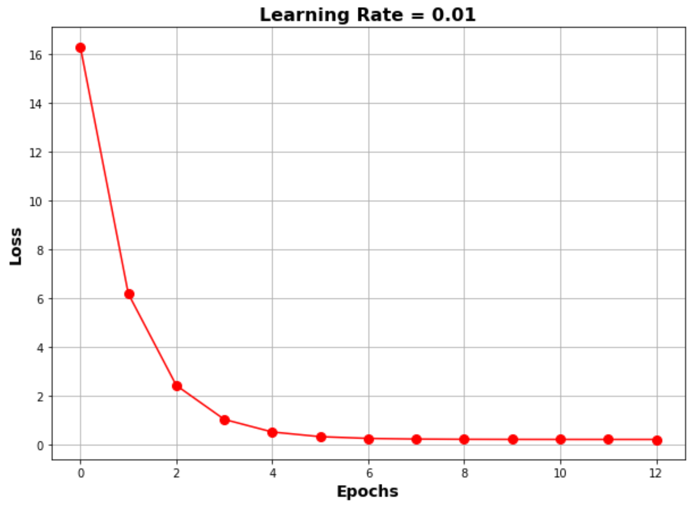
How to find a good value for the learning rate?

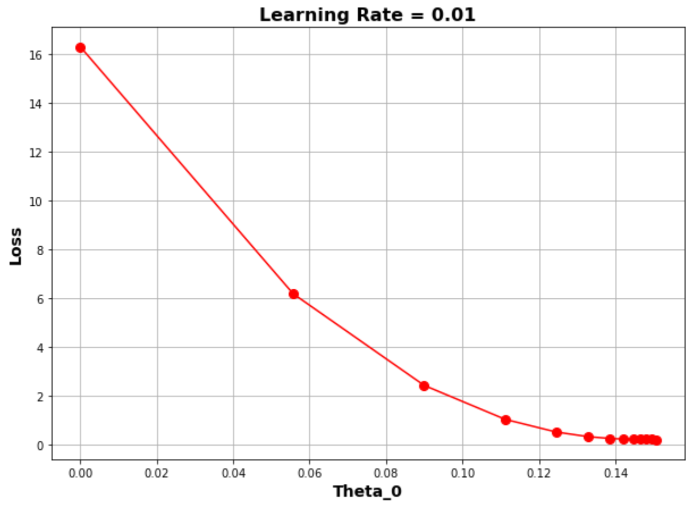


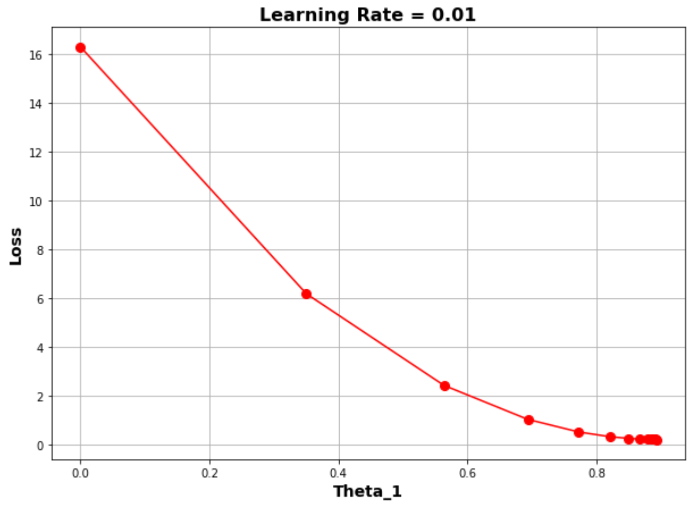




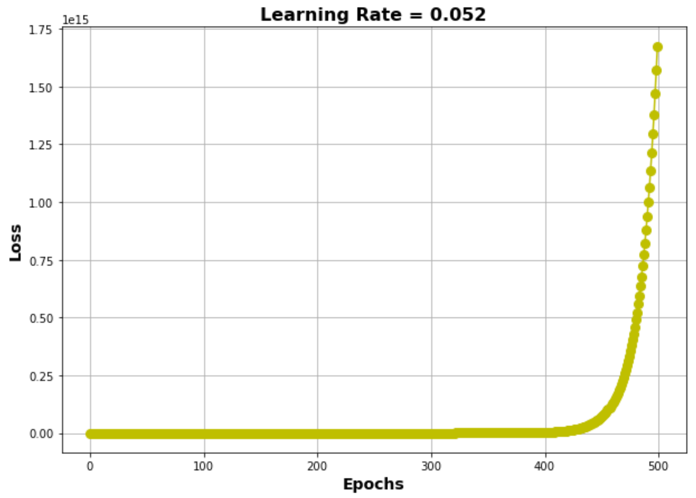
How to find a good value for the learning rate?

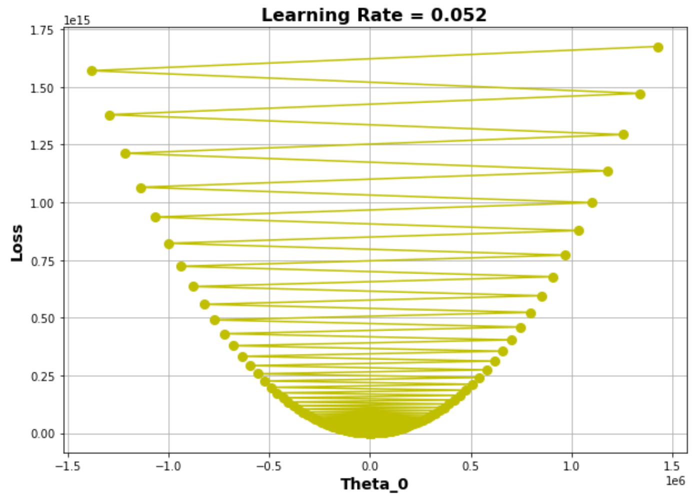


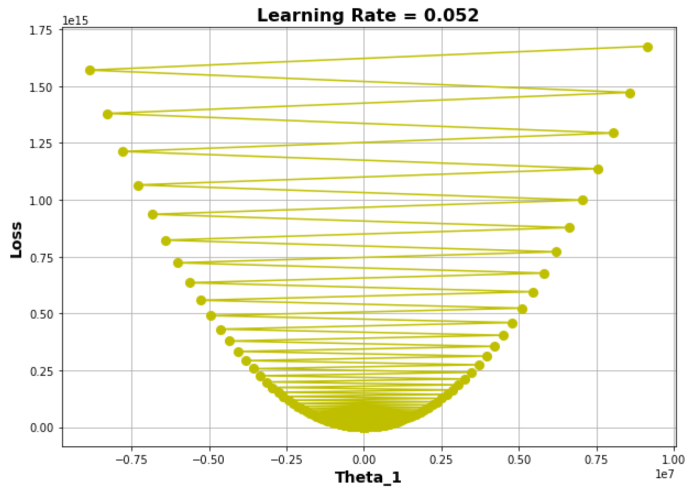




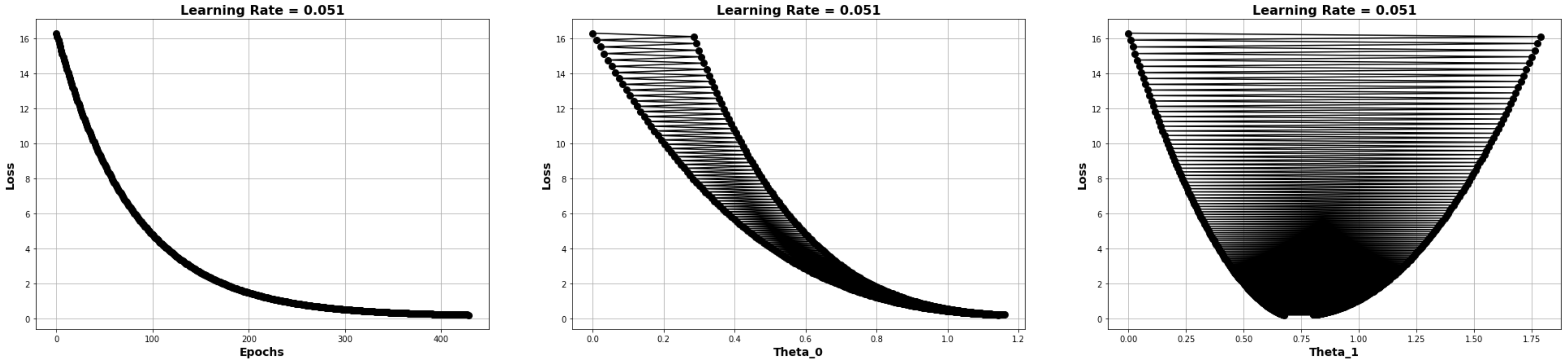
How to find a good value for the learning rate?







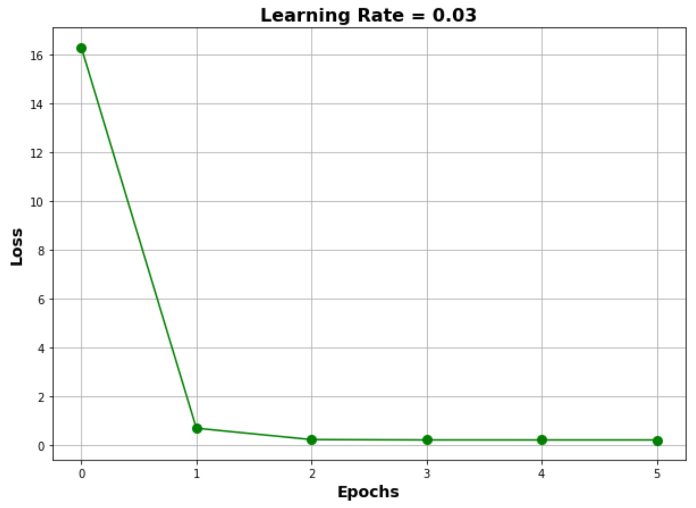
How to find a good value for the learning rate?

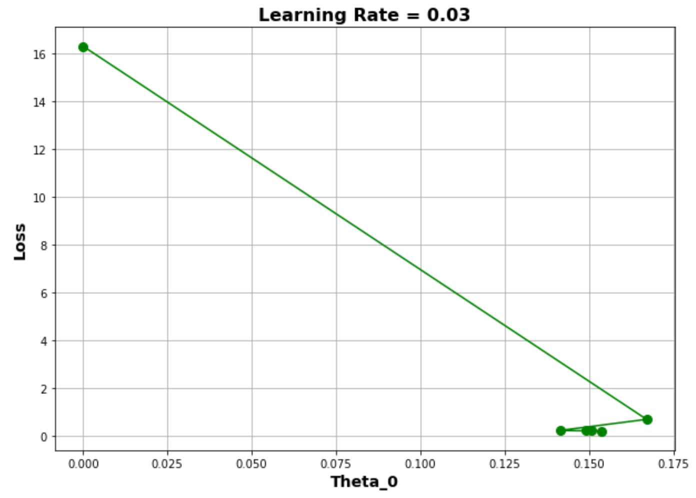


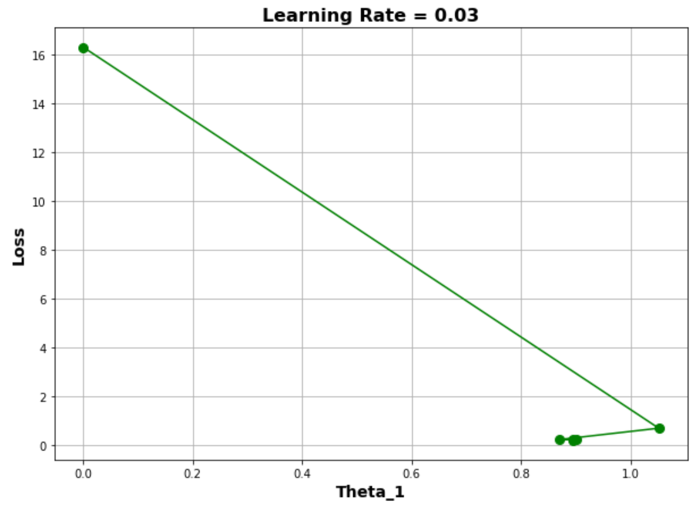
How to find a good value for the learning rate?

How to find a good value for the learning rate?

Can you guess now the suitable α ???

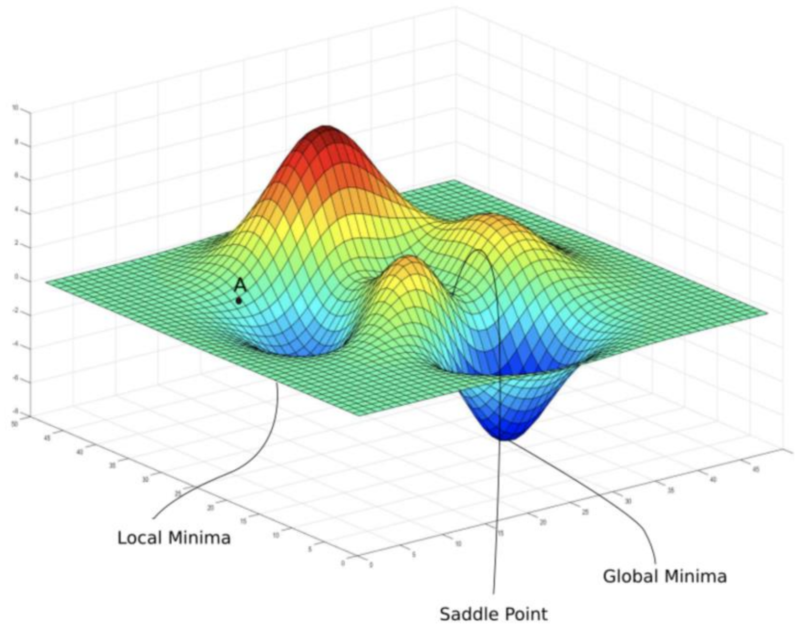


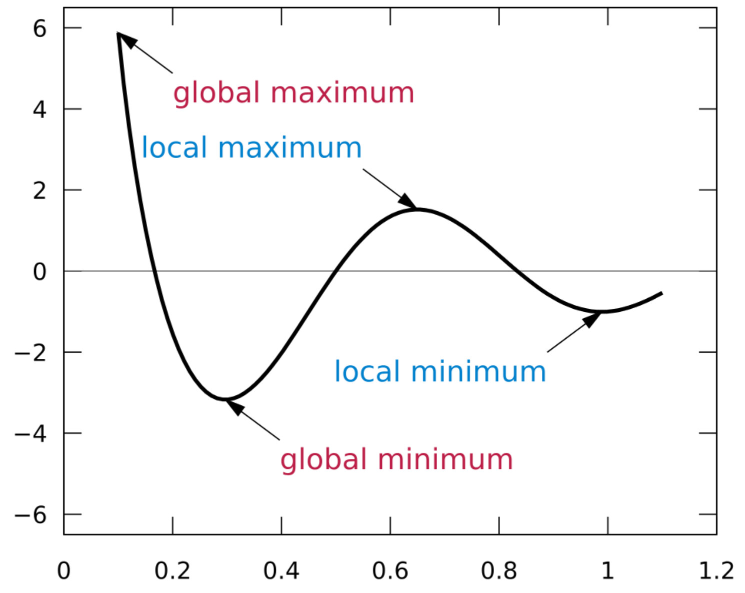




How to find a good value for the learning rate?

Of course, model accuracy need to be checked for these parameters (θ_0, θ_1)

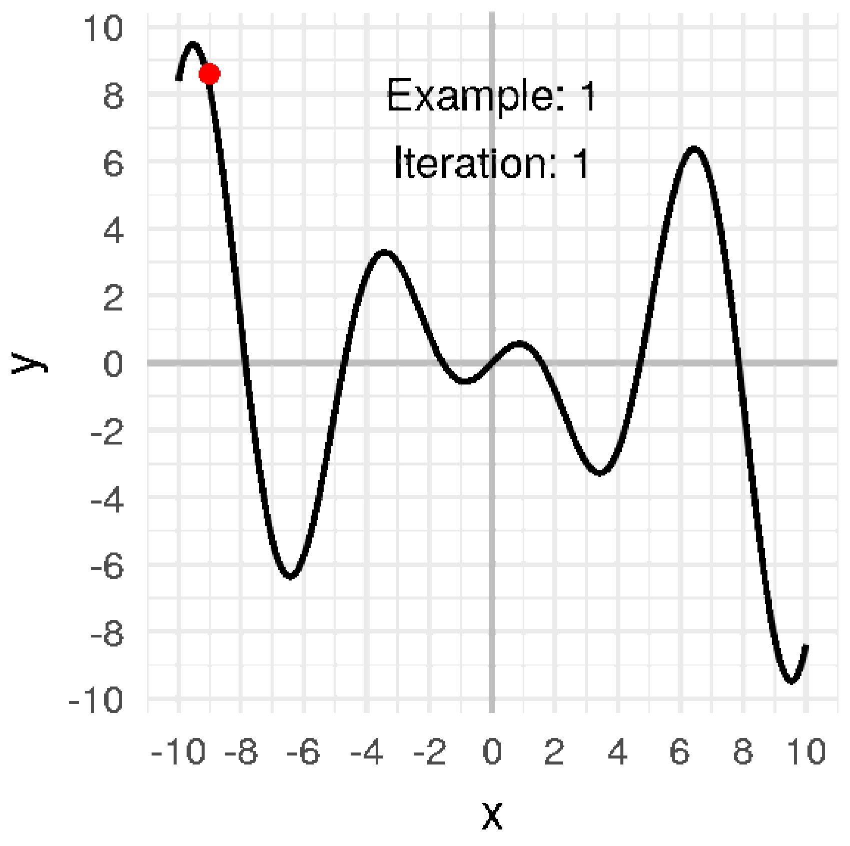




Local Minimum

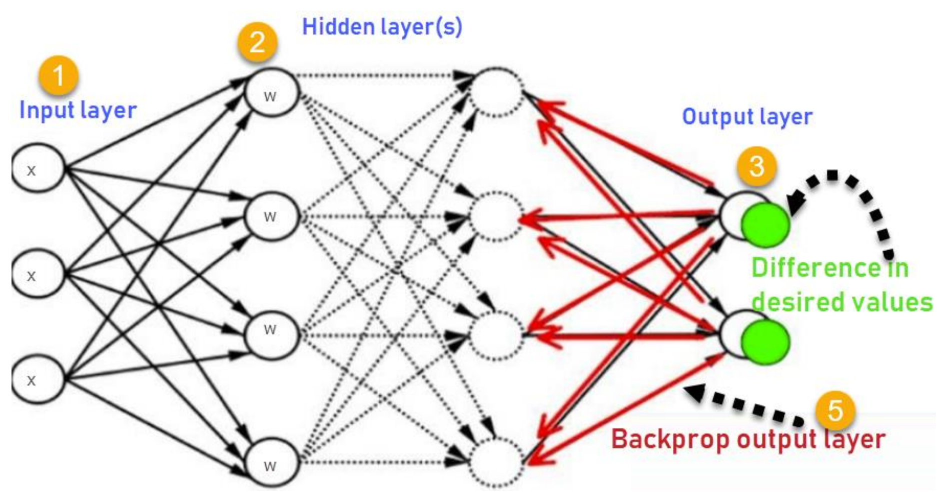
Local Minimum

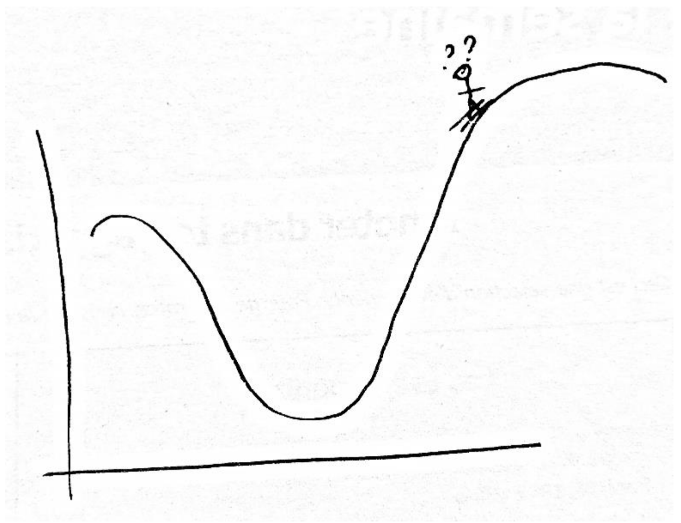
- Notice the final convergence point depends a lot on the initial point.
- Sometimes it'll find the global minimum. Other times.. not.
- To avoid this problem, the best way is to run the algorithms multiple times and keep the best minimum of all times. But, of course, it takes a long time to run.



Vanishing Gradient

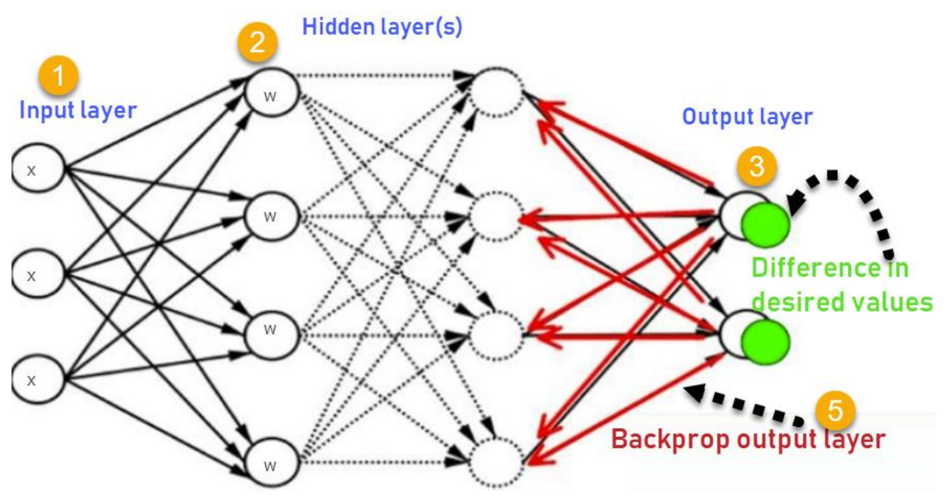
- There are two frequent problems in **<u>Deep Learning</u>:** ***exploding gradient*** **and** ***vanishing gradient***.
- In the first case, it's similar to having a too big learning rate. The algorithm is unstable and never converges.
- With Deep Learning, it can happen when your network is too deep. Since the gradients from each layer get multiplied with each other, you quickly obtain a gradient that explodes exponentially.

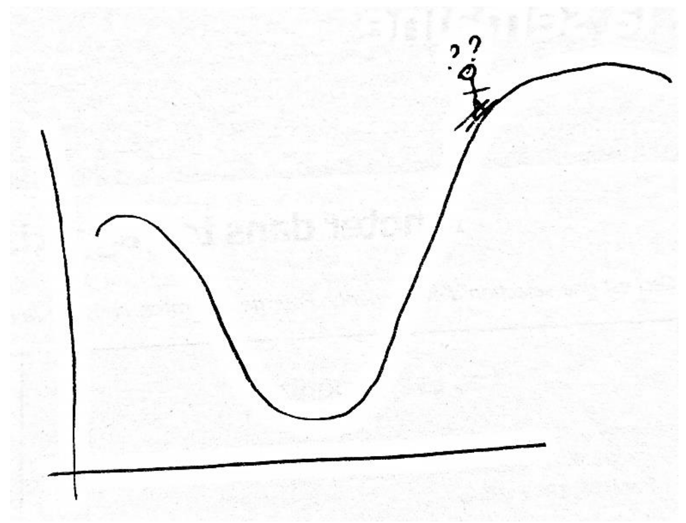




Vanishing Gradient

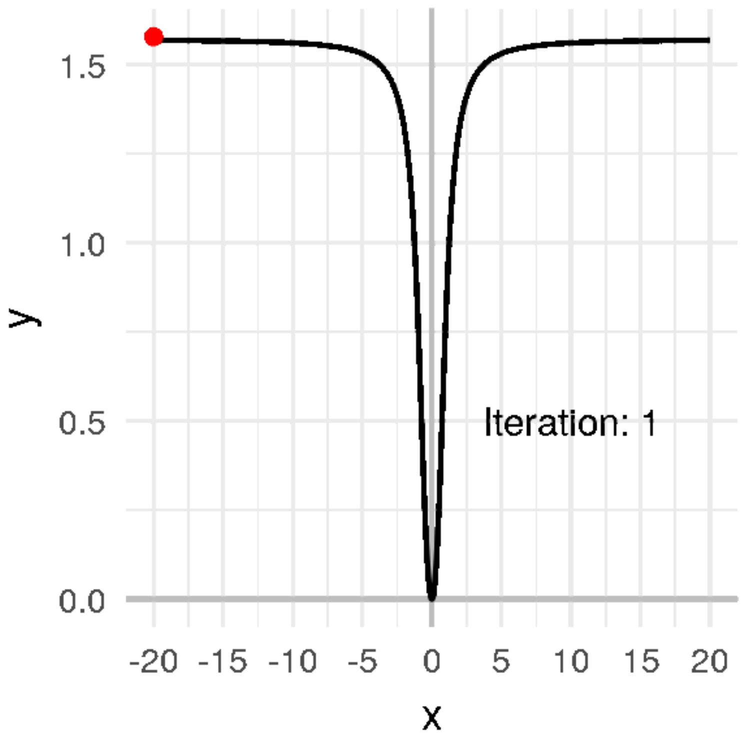
- For the vanishing gradient, it’s the opposite.
- The gradient becomes so small that the skier barely moves anymore.
- It can happen if the learning rate is too small.
- But it can also happen if the skier (the algorithm) is stuck on a flat line.





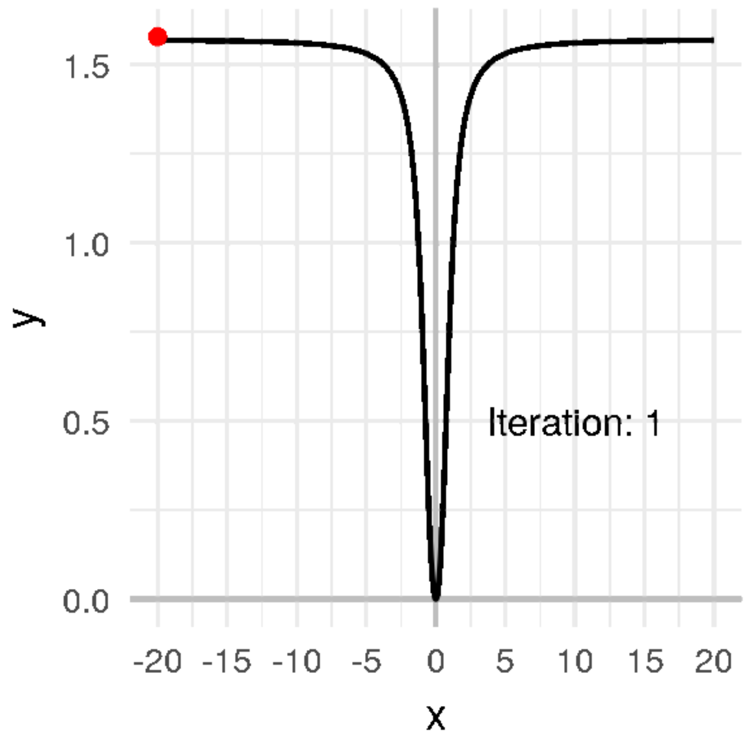
Vanishing Gradient

- It’s clear that the minimum is at x =0.
- But suppose the initial random point we pick is very far from 0, like −20.
- Even if I pick a big learning rate, the algorithm will take very long to converge.



Vanishing Gradient

- In this example, we can still find the minimum. But with a more complex function that has flat lines in some places and is very curvy in other places… it becomes a mess.
- In this case, whatever value you take for the learning rate, you'll be in trouble.
- Either a very slow convergence or an unstable algorithm.



GD Isn’t Enough

GD
Challenges
Conclusion

Resources

- Andrew Ng, Machine Learning, Stanford University, Coursera
- https://ruder.io/optimizing-gradient-descent/index.html#batchgradientdescent
- https://towardsdatascience.com/calculus-behind-linear-regression-1396cfd0b4a9
- https://towardsdatascience.com/why-gradient-descent-isnt-enough-a-comprehensive-introduction-to-optimization-algorithms-in-59670fd5c096
- https://towardsdatascience.com/why-gradient-descent-isnt-enough-a-comprehensive-introduction-to-optimization-algorithms-in-59670fd5c096
- https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3
- https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3
- https://www.charlesbordet.com/en/gradient-descent/#how-does-it-work
- https://www.charlesbordet.com/en/gradient-descent/#how-to-find-a-good-value-for-the-learning-rate
- https://www.charlesbordet.com/en/gradient-descent/#what-to-do-in-case-of-local-minima
- https://towardsdatascience.com/why-gradient-descent-isnt-enough-a-comprehensive-introduction-to-optimization-algorithms-in-59670fd5c096
- https://medium.com/hackernoon/implementing-different-variants-of-gradient-descent-optimization-algorithm-in-python-using-numpy-809e7ab3bab4
- https://ranasinghiitkgp.medium.com/optimization-algorithm-d62ac8f597b1
- http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Resources

- https://towardsdatascience.com/why-gradient-descent-isnt-enough-a-comprehensive-introduction-to-optimization-algorithms-in-59670fd5c096
- https://ranasinghiitkgp.medium.com/optimization-algorithm-d62ac8f597b1
- https://www.youtube.com/watch?v=lAq96T8FkTw&ab_channel=DeepLearningAI
- https://www.youtube.com/watch?v=NxTFlzBjS-4&ab_channel=DeepLearningAI
- https://www.youtube.com/watch?v=lWzo8CajF5s&ab_channel=DeepLearningAIDeepLearningAI
- https://medium.datadriveninvestor.com/exponentially-weighted-average-for-deep-neural-networks-39873b8230e9
- https://towardsdatascience.com/understanding-gradient-descent-and-adam-optimization-472ae8a78c10
- DOI: 10.1177/0735633118757015
- https://www.asimovinstitute.org/author/fjodorvanveen/
- guru99.com
- https://towardsdatascience.com/an-introduction-to-reinforcement-learning-1e7825c60bbe