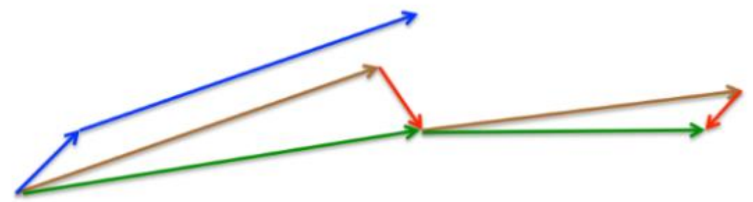
Nesterov Accelerated Gradient Descent

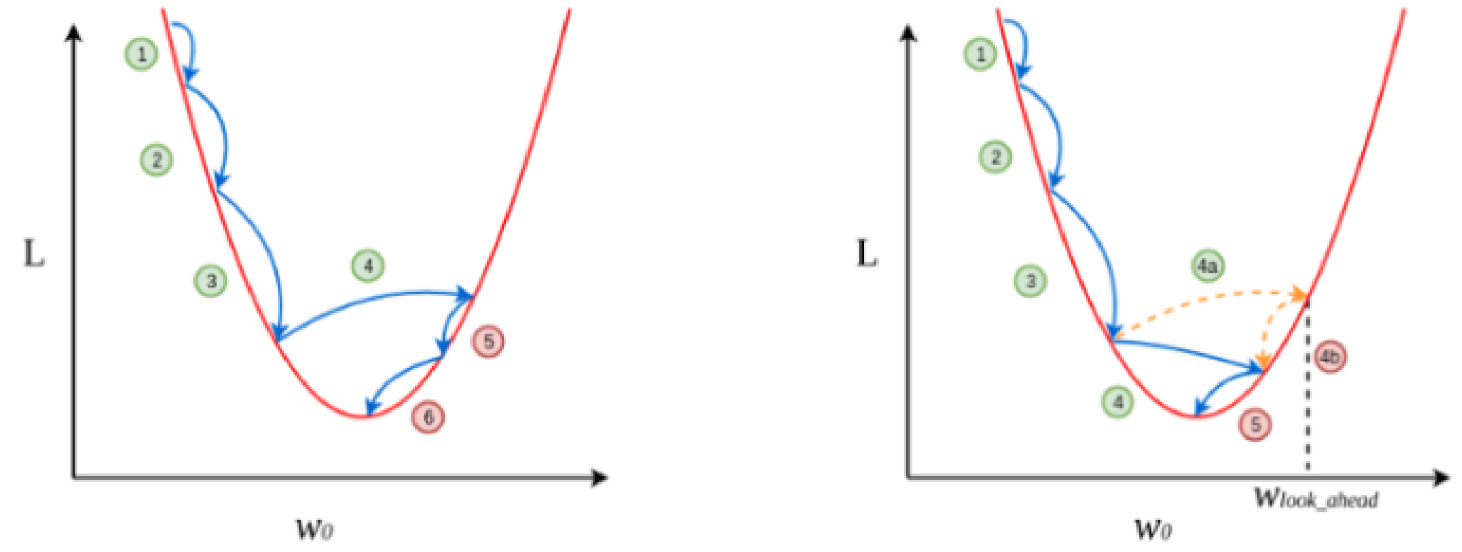
- Momentum based gradient descent works good in gentle regions but it overshoots sometimes as its moving really fast and having many oscillations. And then it has to take a lot of U-turns.
- Any better way to avoid this ? … Yes. The main idea is ‘ Look before you leap ‘.
- Nesterov accelerated gradient descent update has two steps.
- We know that we are moving at least by the history term. So think !!! what can we do next ??
- Yeah !! we are calculating gradient at the look ahead point.
- This prevent us from moving too fast and increased responsiveness.

Update rule for NAG

$$w_{look\_ahead} = w_t - \gamma \cdot update_{t-1}$$
$$update_t = \gamma \cdot update_{t-1} + \eta \nabla w_{look\_ahead}$$
$$w_{t+1} = w_t - update_t$$

We will have similar update rule for b_t





(a) Momentum-Based Gradient Descent

(b) Nesterov Accelerated Gradient Descent

Sources : https://towardsdatascience.com/learning-parameters-part-2-a190bef2d12

Root Mean Squared Prop

- Adagrad decays the learning rate very aggressively(as the denominator grows).
- As a result , after a while the frequent parameters will start receiving smaller updates because of decayed learning rate.
- To overcome this , RMSprop is introduced.
- Beta is close to 0.95. We are doing the same thing but we aren't doing aggressively. At each step ,we are multiplying a fraction of it.

Update rule for RMSProp

$$v_t = \beta * v_{t-1} + (1-\beta)(\nabla w_t)^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} * \nabla w_t$$

... and a similar set of equations for b_t

Additional strategy for optimizing SGD

1. **Shuffling and curriculum learning :**

- Try to avoid providing the training examples in a meaningful order as this may bias the optimization algorithm.
- It is often a good idea to shuffle the training data after every epoch.
- Sometimes supplying the training examples in a meaningful order may actually lead to improved performance. It is called curriculum learning.

1. **Batch Normalization :**

- Normalize the initial values of our parameters by initializing them with zero mean and unit variance.
- It reduces the sensitivity to initializing starting weights.
- Batch normalization reestablishes these normalizations for every mini-batch.Batch normalization additionally acts as a regularizer.

3. Early Stopping :

Always monitor error on a validation set during training and stop if your validation error does not improve enough.

4. Gradient Noise :

Adding this noise makes networks more robust to poor initialization and helps training particularly deep and complex networks.

Resources :

- https://www.youtube.com/watch?v=sV9aiEsXanE
- https://www.youtube.com/watch?v=FKCV76N9Ys0
- https://arxiv.org/pdf/1609.04747.pdf
- https://medium.com/iitg-ai/into-the-depths-of-gradient-descent-52cf9ee92d36
- https://ruder.io/optimizing-gradient-descent/index.html