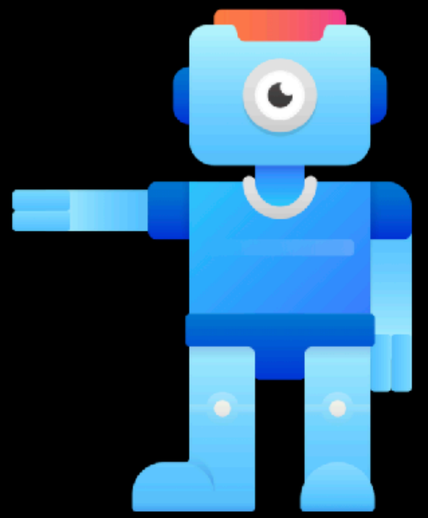


MODERN COMPUTER VISION

BY RAJEEV RATAN

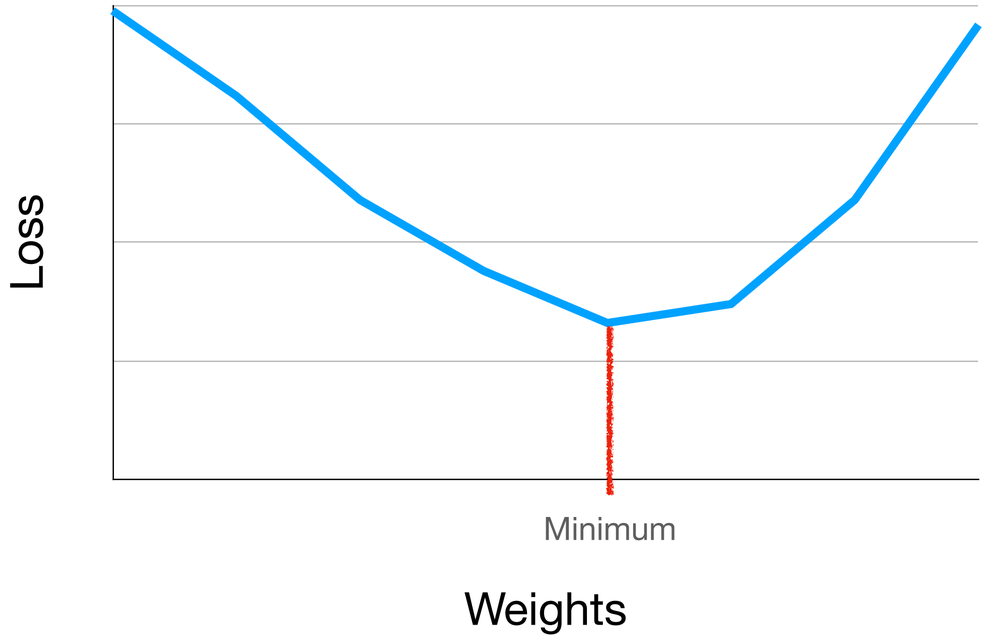
Optimisers & Learning Rate Schedules

Methods or Algorithms used in finding optimal weights

Optimisers

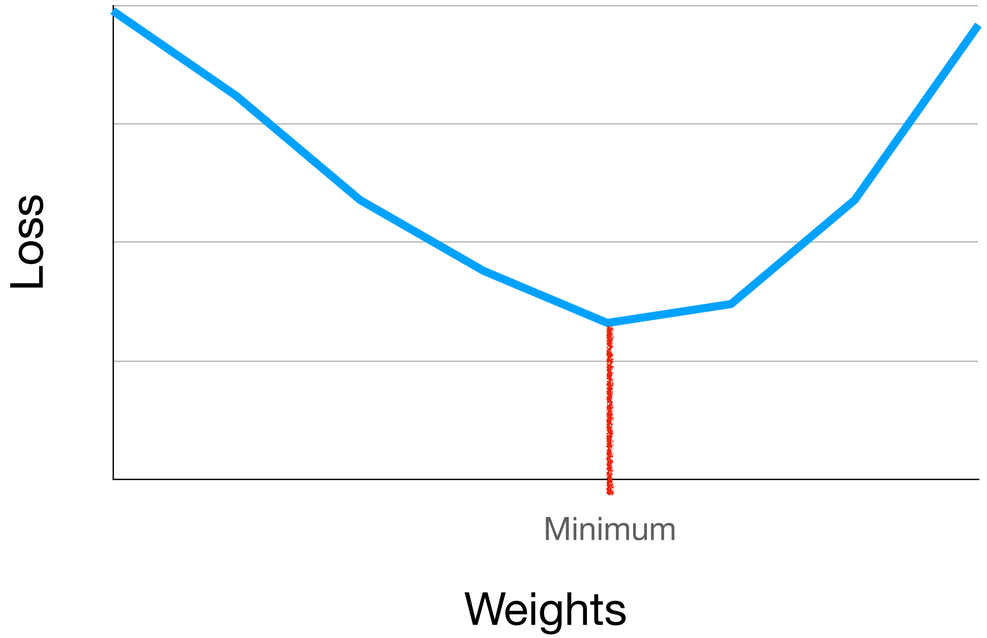
The algorithm we use to update the weights

- You have already been introduced to Gradient Descent which is an example of a first order optimisation algorithm.
- In this section we will explore some alternatives to Stochastic Gradient Descent and take a look at a few other optimisation algorithms.



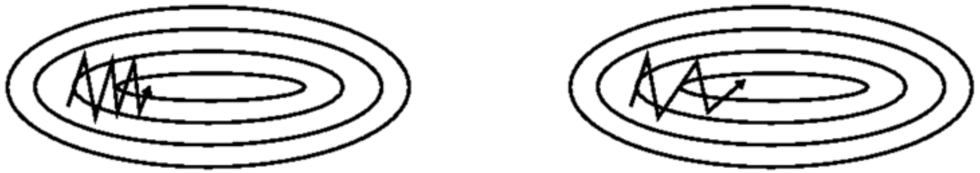
Problems with standard SGD

- Choosing an appropriate Learning Rate (LR), deciding Learning Rate Schedules,
- Using the same learning rate for all parameter updates (as is the case with sparse data), but most importantly SGD is susceptible to getting trapped in Local Minimas or Saddle Points (where one dimensions slopes up and another slopes down).
- To solve some of these issues several other algorithms have been developed including some extensions to SGD which include Momentum and Nestor's Acceleration.



Momentum

- One of the issues with SGD are areas where our hyper-plane is much steeper in one direction.
- This results in SGD oscillating around these slopes making little progress to the minimum point.
- Momentum increases the strength of the updates for dimensions whose gradients switch directions. It also dampens oscillations. Typically we use a Momentum value of 0.9.



(a) SGD without momentum (b) SGD with momentum

Figure 2: Source: Genevieve B. Orr

https://arxiv.org/pdf/1609.04747.pdf

Nesterov's Acceleration

- One problem introduced by Momentum is overshooting the local minimum.
- Nestoerov's Acceleration is effectively a corrective update to the momentum which lets us obtain an approximate idea of where our parameters will be after the update.
- Below we show the corrected Gradient updates (in green)

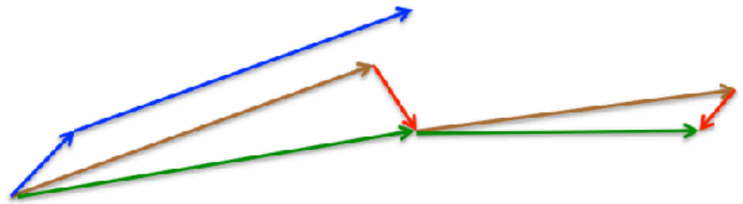
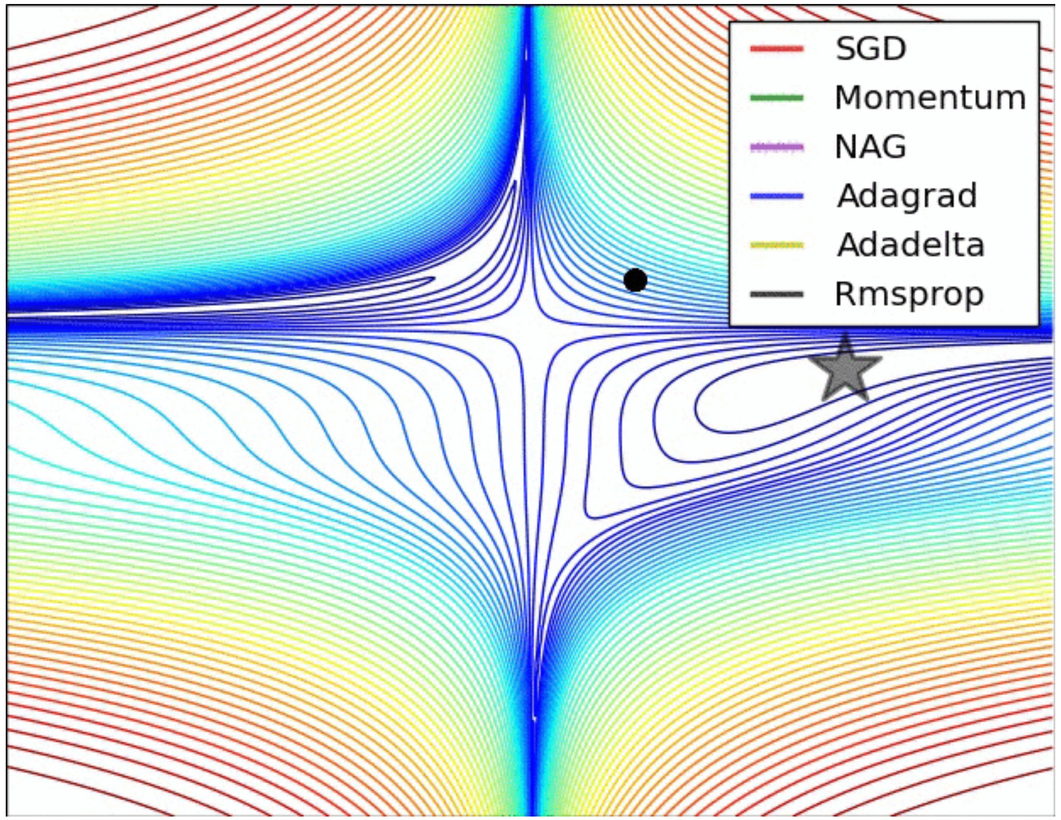


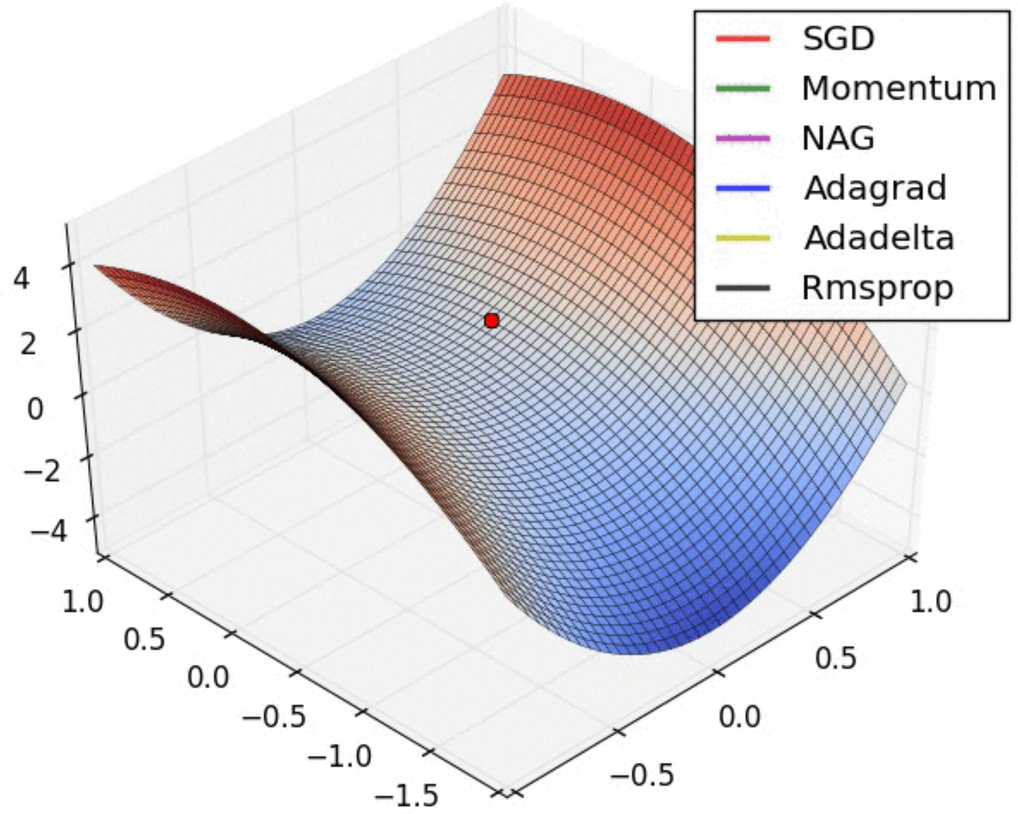
Figure 3: Nesterov update (Source: G. Hinton's lecture 6c)

Other Optimisers

- **Adagrad** - Good for Sparse data. It adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features, and larger updates (i.e. high learning rates) for parameters associated with infrequent features.
- **Adadelta** - Extension of Adagrad that reduces its aggressive, monotonically decreasing learning rate.
- **Adam** - Adaptive Moment Estimation is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients, similar to momentum.
- **RMSProp** - Similar to adadelta,
- **AdaMax** - It is a variant of Adam based on the infinity norm.
- **Nadam** - Nesterov accelerated gradient (NAG) is superior to vanilla momentum. Nadam (Nesterov-accelerated Adaptive Moment Estimation) thus combines Adam and NAG. In order to incorporate NAG into Adam, we need to modify its momentum term
- **AMSGrad** - AMSGrad is an extension to the Adam version of gradient descent that attempts to improve the convergence properties of the algorithm, avoiding large abrupt changes in the learning rate for each input variable

Visual Comparison of some Optimisers

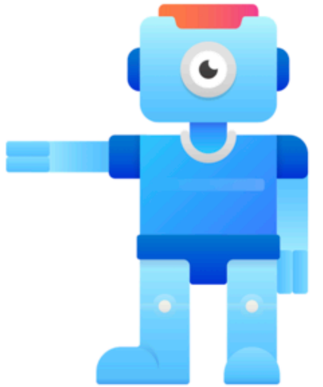




http://louistiao.me/notes/visualizing-and-animating-optimization-algorithms-with-matplotlib/

Learning Rate Schedules

- A preset list of learning rates used for each epoch.
- Progressively reduce over time.
- We use LR Schedules because if our LR is too high, it can overshoot the minimum points (areas of lowest loss).
- Applying a progressively decreasing learning rate allows the network to take smaller steps (when gradients are updated) allowing our network to find the point of lowest loss instead of jumping over it.
- Early in the training process, we can afford to take big steps, however as the decrease in loss slows, it is often better to use smaller learning rates to avoid oscillations.
- Learning rate schedules are simply to implement with our Deep learning libraries (PyTorch or Keras/Tensorflow) as they incorporate a decay parameter in optimisers that support LR schedules, such as SGD.



MODERN COMPUTER VISION

BY RAJEEV RATAN

Next...

A Review of all CNN Theory