**Week 2 – Notes**

**Optimization Algorithms**

**Mini-batch Gradient Descent**

If there are too many training examples, a cost function update is made extremely slow

The idea is to use instead of batch gradient descent (the gradients being updated only after learning all the examples) a mini-batch gradient descent; thus you update the gradients after learning only a part of the training examples

A picture containing handwriting, font, typography, calligraphy

Description automatically generated - notation for batch t

Mini-batch -> perform one step of gradient descent for each batch (forward prop, cost computation and back prop)

One epoch means a pass through training set, so if you use the mini-batch, you have to pass through each mini-batch

**Understanding Mini-batch Gradient Descent**

When using batch gradient descent, the cost function decreases monotonically, but when the mini-batch gradient descent is used, the cost oscillates because some mini-batches are “harder” to learn

Batch gradient descent: (mini-batch size = m) -> takes a lot of time for each iteration, but it converges to the local minima smoothly; it can be used for small training sets (<= 2000 examples)

Stochastic gradient descent: (mini-batch size = 1) -> you loose the vectorization speed, for many iterations it can go towards wrong directions and it wanders around the local optimum

Mini-batch gradient descent: (the in-between solution) -> faster learning, uses the vectorization, however it may go towards not the best directions, but in the end it will be very close to the local optimum; typical mini-batch sizes are 64, 128, 256, 512 and rarely 1024 (all of them are powers of 2 for optimization)

Pay attention that the mini-batch fits in the CPU/GPU memory

**Exponentially Weighted Averages**

A technique to compute the moving average is to use exponentially weighted moving average that for each step computes:

A picture containing handwriting, font, calligraphy, line

Description automatically generated, where beta is a parameter and theta t is the current point

This method takes into consideration less and less the oldest data points

Vt is approximatively an average over 1 / (1 – beta) data points

For examples if beta = 0.9 -> ~ 10 days; beta 0.98 -> ~ 50 days; beta 0.5 -> ~ 0.5 days