Lecture 1

Part 1. Basic terms

If your function has a 1000 input features, 1000 outputs so you will have 1000^2 derivatives, which is bad.

So, you use *backpropagation*. It consists passing the derviative from the output to the input instead of usual.

Part 2. Whatever

Some biological inspiration blabbery yada yada… Look up the first lecture of the DL part of ML course.

Part 4. Optimization

Gradient descend is good, but it is slower than (Quasi-)Newtonian methods, but the latter is kinda sad bc of hessian. So people gravitated towards SGD since its kinda faster and not much worse in terms of convergence time.

Also we can add(?) the momentum term to our SGD to control the optimization process. There is also a technique to adjust LR mid-training that uses(?) momentum. If a weight is updated a lot, it's LR will be decreased.

The exponential moving average is given by

$$\text{ms}_{t+1} = \gamma \text{ms}_t + (1-\gamma) \cdot \left|\left| \frac{\delta L}{\delta x} \right|\right|.$$

Adam(adaptive + momentum) optimizer is the **GOAT**. It is *simple* and it *works*. There are other optimizers, but they are not substantially better.

Initialization matters, since if you initialize all weights to zero, you features may contain identical numbers, so the model won't learn more than one feature, which is antithetical to to the NN purpose: extract a ton of latent features. So, the smart engineer that you are, you initialize your weights with random values. Mostly you choose the distribution such that the output layer does not contain too large values.