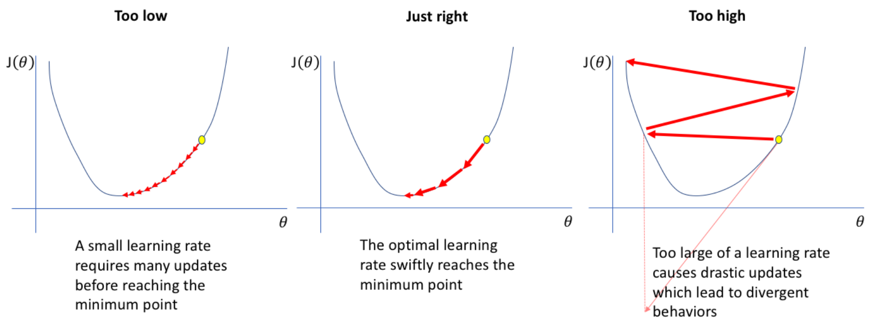
3 Neural Network

Neural network is build of multiple layers of single units, each layer can have multiple units too. The backpropagation algorithm is the same:
1. forward pass - calculating loss 2. using chain rule we calculate the gradient of the loss of the top most weights and update them, then continue backward and update the weights in each layer until we get to the input layer... -¿ hence Backpropagation!

3.1 Learning Rate

learning rate has a huge effect on how the network will converge to a solution. If the learning rate is to big we might ”miss”the minima. However if it is too small it might take a long time to reach the minima.



3.2 Learning Rate optimizations

some optimization methods for NN manipulate the learning rate. One way is by reducing the learning rate when loss reaches a plateau. Adagrad, another optimizer, uses different learning rates for different features (feature is a sample). This way one can use small learning rates for features which occur often and larger to features which are rare, helping with learning of sparse samples.

3.3 Momentum

The loss manifold can be very non-smooth causing the convergence to be very chaotic. The Momentum is aimed to help with ßmoothing”this manifold. Momentum in physics describes that an object will continue moving in the same direction with the same mass if no external forces are exerted on it. In our case the idea is to take the previous history of your trajectory going down the loss manifold slope in determining your next step. This will result in a more smooth trajectory ignoring small bumps occurring on the way. The Adam optimizer uses momentum on top of learning rate optimizations.
Vanila Gradient Descent:

$$W = W - \alpha * W gradient \tag{11}$$

With Momentum:

$$V_t =_t -1 + \alpha \nabla W f(W) \tag{12}$$

$$W = W - V_t \tag{13}$$

More about optimization methods for NN: Optimization and NN

3.4 Regularizations

There are infinite sets of weights that can yield a reasonable outcome. However, not all of these will result in equally good results on a test data-set, that the model has never seen. One reason is that the model can learn to fit to the noise of the training set, which will yield good loss results, but will result in bad fit for a test set. To avoid this we want to not over fit during training. There are methods to avoid over fitting:
1. Adding regularization to the loss function. Typical terms include L1/L2.

$$L1 = \sum_i \sum_j |W_{i,j}| \tag{14}$$

$$Loss = \frac{1}{N}\sum_{n=1}^{N} L_i + \gamma L1 \tag{15}$$

$$L2 = \sum_i \sum_j W_{i\ ,j}^2 \tag{16}$$

2. Dropout. During training some weights (a certain fraction) will be omitted. This forces the network to learn using different connections and not relying to heavily on one particular path.
3. Early stopping of training: stopping training after a fixed number of epochs or when loss is small enough
4. Data augmentation: adding synthetically generated data (from actual examples). For example pictures which are stretched/ rotated/ contrast changed/ added noise etc.

3.5 Further resources

Backprop visualized
Backprop deeper

NEXT WE WILL DO PRACTICAL 4