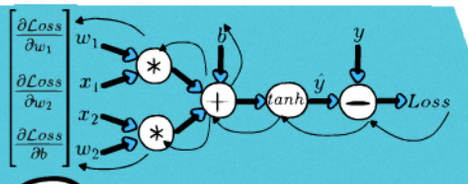
TRAINING LOOP

1 Initialize parameters to random values

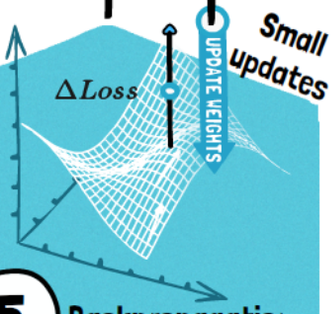
2 Forward pass

Predicted labels $\hat{y}$ → y True labels

3 Calculate Loss



4 Backward pass

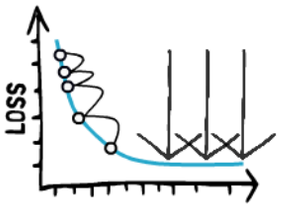


5 Backpropagation

Small updates

UNTIL CONVERGENCE

CONTINUE...



- More precise
- Slower

Small

- Faster
- Might overshoot the minimum

Large

LEARNING RATE

$$w_1 = w_1 - \eta\left(\frac{\partial Loss}{\partial w_1}\right)$$

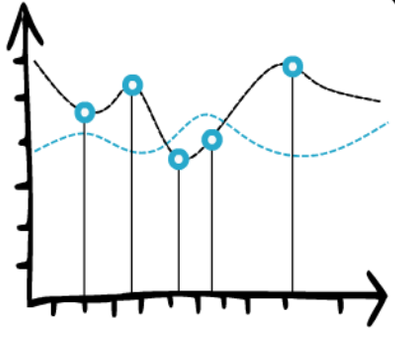
0.1, 0.001, ...

FULL-BATCH

Use average gradient over entire dataset to update parameters in a single step

Less noisy

Challenges with large datasets



training sample

function to learn

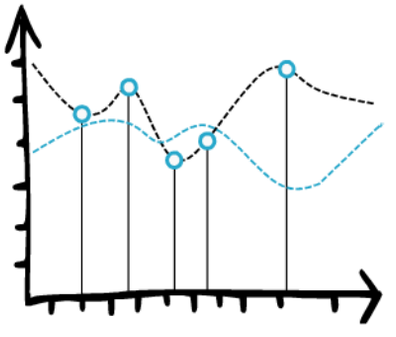
current network approximation

INCREMENTAL

Update parameters after each training example is processed

Smaller updates

Repetitive patterns



MINI-BATCH

Batch Size (32, 64, 128, ...)

Efficient

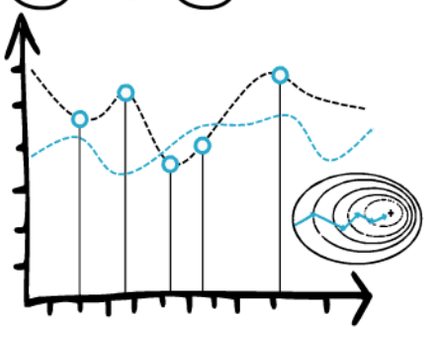
Less noisy

STOCHASTIC GRADIENT DESCENT (SGD)

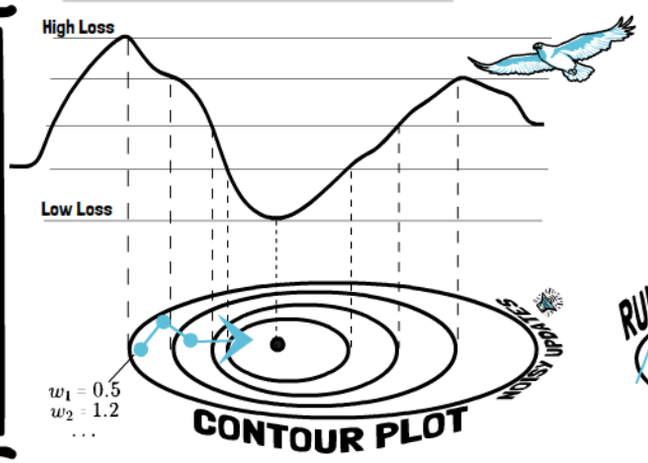
Random samples

Noisy

More robust



GRADIENT DESCENT OPTIMIZATIONS



MOMENTUM

Keep moving in consistent direction

$\Delta Loss +$

RUNNING AVERAGE = MEMORY

ROOT MEAN SQUARE PROPAGATION (RMSProp)

Adjust learning rate for each parameter

Boost Smaller updates

Adaptive learning rate

ADAM

Adaptive Moment Estimation