

Neural Network Learning \rightarrow finding the right weights & biases

Neural Network function

input = 784 numbers (pixels)

output = 10 numbers

Parameters = 73002 weights/biases

Cost function

input = 73002 weights/biases

output = 1 number (cost)

Parameters = many training examples

Cost \rightarrow Squared difference from output to correct

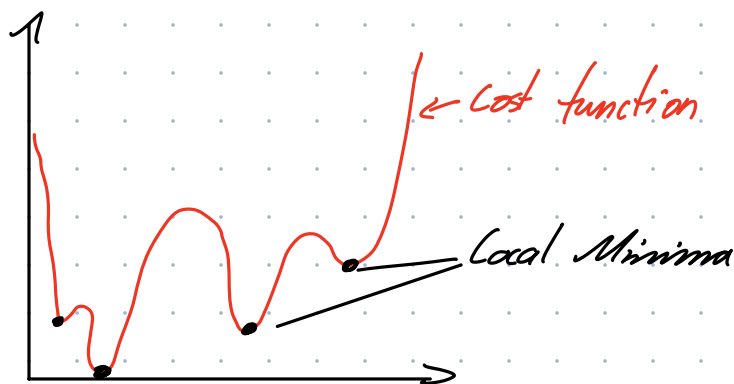
Cost of 3	
0.0006	$\leftarrow (0.02 - 0.00)^2 +$
0.0007	$\leftarrow (0.03 - 0.00)^2 +$
0.0039	$\leftarrow (0.06 - 0.00)^2 +$
0.0009	$\leftarrow (0.97 - 1.00)^2 +$
0.0055	$\leftarrow (0.07 - 0.00)^2 +$
0.0004	$\leftarrow (0.02 - 0.00)^2 +$
0.0022	$\leftarrow (0.05 - 0.00)^2 +$
0.0033	$\leftarrow (0.06 - 0.00)^2 +$
0.0072	$\leftarrow (0.08 - 0.00)^2 +$
0.0018	$\leftarrow (0.04 - 0.00)^2 +$
0.03	

Low cost
correct output

Cost of 3	
0.1863	$\leftarrow (0.43 - 0.00)^2 +$
0.0809	$\leftarrow (0.28 - 0.00)^2 +$
0.0357	$\leftarrow (0.19 - 0.00)^2 +$
0.0138	$\leftarrow (0.88 - 1.00)^2 +$
0.5242	$\leftarrow (0.72 - 0.00)^2 +$
0.0001	$\leftarrow (0.01 - 0.00)^2 +$
0.4079	$\leftarrow (0.64 - 0.00)^2 +$
0.7388	$\leftarrow (0.86 - 0.00)^2 +$
0.9817	$\leftarrow (0.99 - 0.00)^2 +$
0.3998	$\leftarrow (0.63 - 0.00)^2 +$
3.37	

High cost
incorrect output

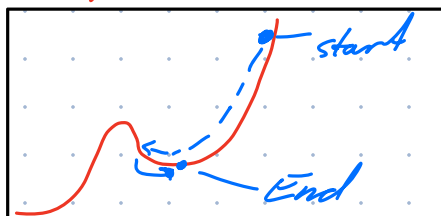
• to find the optimal weights & biases \rightarrow minimize Cost function



Minimization Options

Hill climb \rightarrow step where function decreases most
 \rightarrow repeat

Endpoint = startpoint \pm step size



Things to consider?
 \rightarrow step size

```
# Hill climbing algorithm
for i in range(num_iterations):
    current_val = f(x)
    candidate_x1 = x + alpha
    candidate_x2 = x - alpha
    candidate_val1 = f(candidate_x1)
    candidate_val2 = f(candidate_x2)

    if candidate_val1 > current_val:
        x = candidate_x1
    elif candidate_val2 > current_val:
        x = candidate_x2
```

- ↳ starting point / parameters
- ↳ number of steps / iterations

Gradient descent

↳ (think multi-dimensional hill climb)

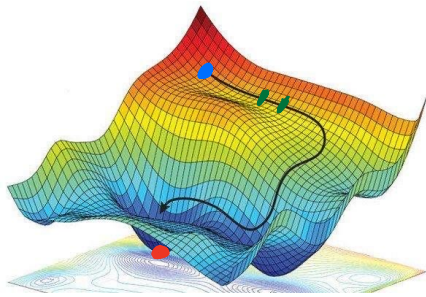
$$\text{Minimum / Endpoint} = \text{Startpoint} - \text{Learning rate} \cdot \text{Gradient}$$

Startpoint = random guess / informed point

Learningrate = think step size \rightarrow it just dotproducts the Gradient

Gradient = direction of steepest increase

↳ calculated by partial derivatives for each variable



\rightarrow repeat the formula until

1. max number of iterations
2. tolerance is achieved (Difference of outputs is small)

things to consider:

1. learningrate size
2. tolerance
3. number of iterations

Gradient Vector of Costfunction encodes:

$$\vec{W} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{13,000} \\ w_{13,001} \\ w_{13,002} \end{bmatrix}$$

$$-\nabla C(\vec{W}) = \begin{bmatrix} 0.31 \\ 0.03 \\ -1.25 \\ \vdots \\ 0.78 \\ -0.37 \\ 0.16 \end{bmatrix}$$

1. if weight/biases should increase/decrease

2. the importance of each weight/bias

w_0 should increase somewhat
 w_1 should increase a little
 w_2 should decrease a lot
 \vdots
 $w_{13,000}$ should increase a lot
 $w_{13,001}$ should decrease somewhat
 $w_{13,002}$ should increase a little