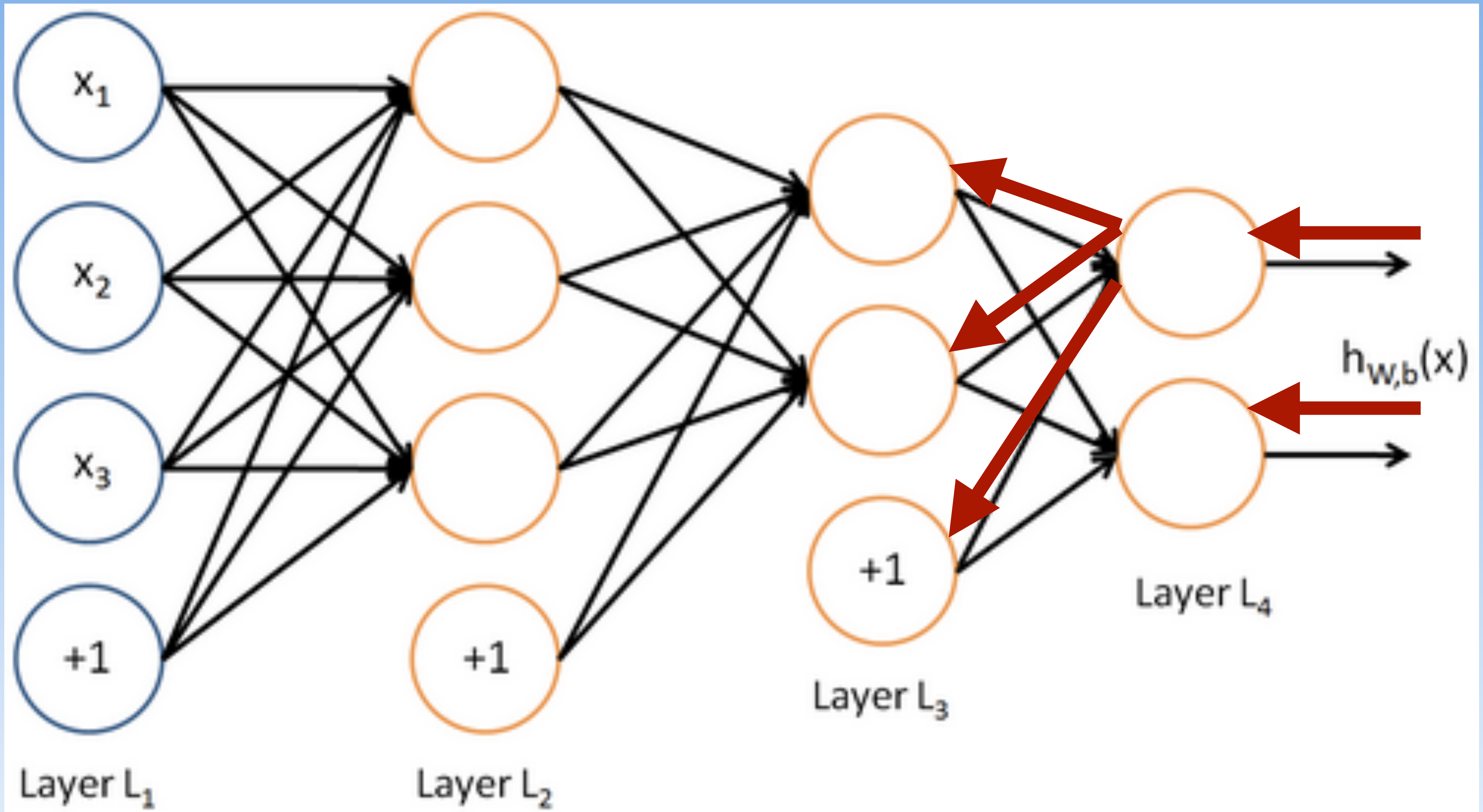
GRADIENT DESCENT

Catalit LLC

AGENDA

- * learning rate
- * loss function and its nonlinearity
- * local minima
- * backpropagation
- * effect of activation function on learning (gradients of the activation function)

BACKPROPAGATION

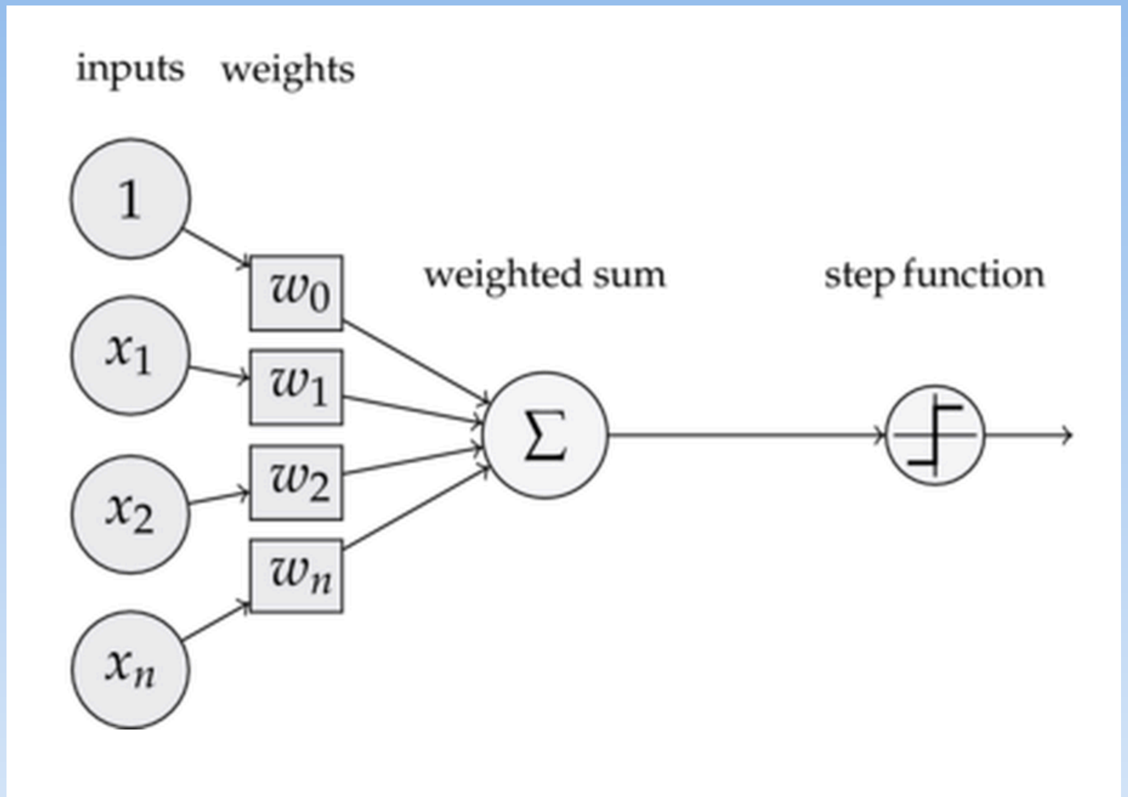


- Adjust weights using the error on value predicted by a node

BACKPROPAGATION

Activation at node j:

$net_j = \sum_i (o_i \times w_{ij})$



Output at node j:

$o_j = f(net_j)$ where $f(net_j) = 1 / (1 + e^{-netj})$

BACKPROPAGATION

Calculate the first derivative of the transfer function:

$$f'(net_j) = f(net_j) \times (1.0 - f(net_j))$$

BACKPROPAGATION

Calculate the first derivative of the transfer function:

$f'(net_j) = f(net_j) \times (1.0 - f(net_j))$

and from that we can calculate the deltas (δ):

$\delta = f'(net) \times (\text{received error})$

BACKPROPAGATION

deltas for output node is:

$$\delta_{output} = f'(net) \times (t - o)$$

BACKPROPAGATION

deltas for output node is:

$\delta_{output} = f'(net) \times (t - o)$

delta for hidden node is:

$\delta_j = f'(net_j) \times \sum_k (\delta_k \times w_{jk})$

BACKPROPAGATION

deltas for output node is:

$\delta_{output} = f'(net) \times (t - o)$

delta for hidden node is:

$\delta_j = f'(net_j) \times \sum_k (\delta_k \times w_{jk})$

And the correction to apply to the weight is:
$dw_{ij} = L \times o_i \times \delta_j$

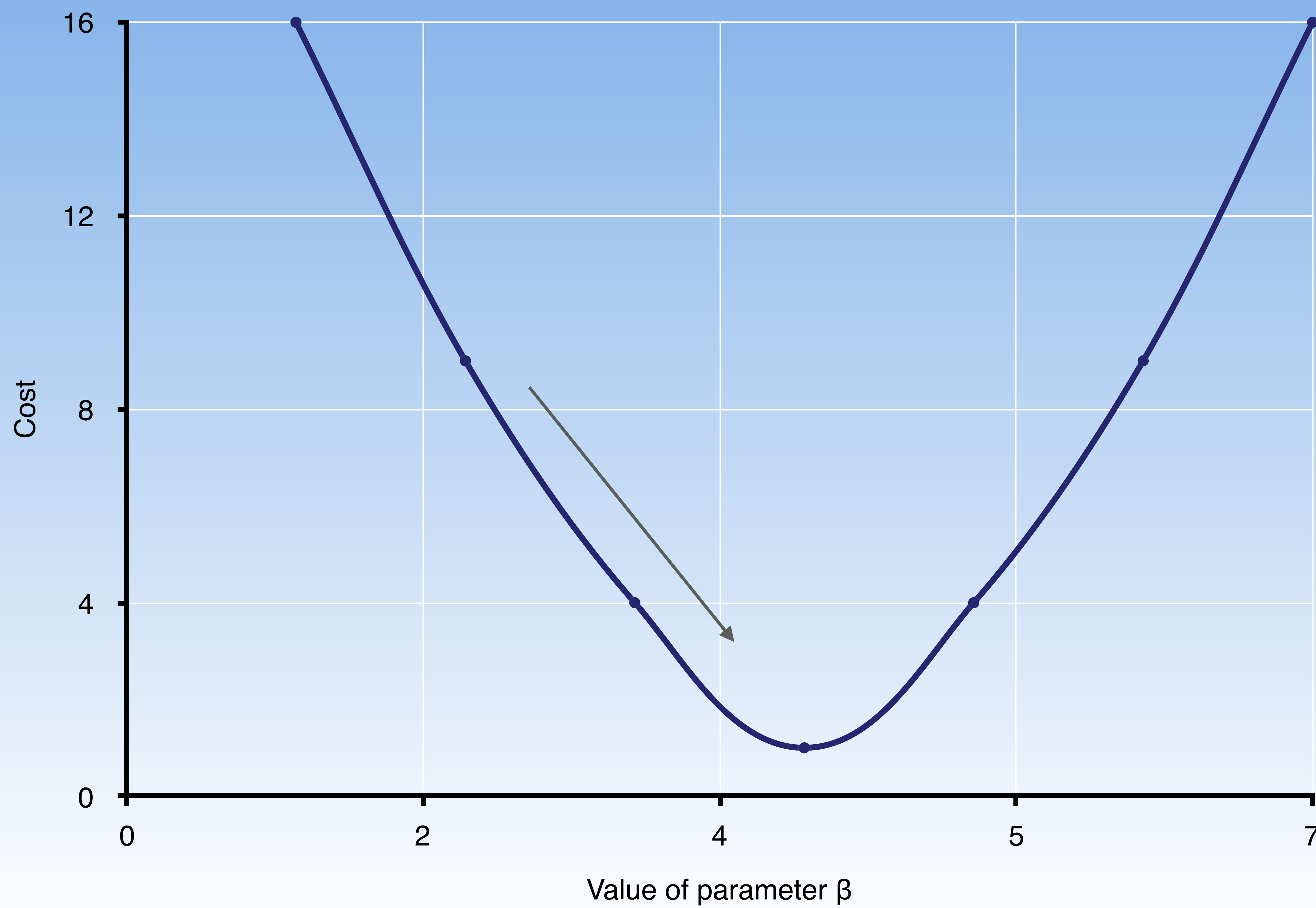
BACKPROP SUMMARY

- **Input:** Set activation for each node of input layer
- **Feedforward:** For each layer, calculate the output
- **Output error δ:** Compute loss function at output layer
- **Backpropagate:** output error back using **gradient**
- **Adjust weights:** At each stage apply correction to weights

MAIN POINT

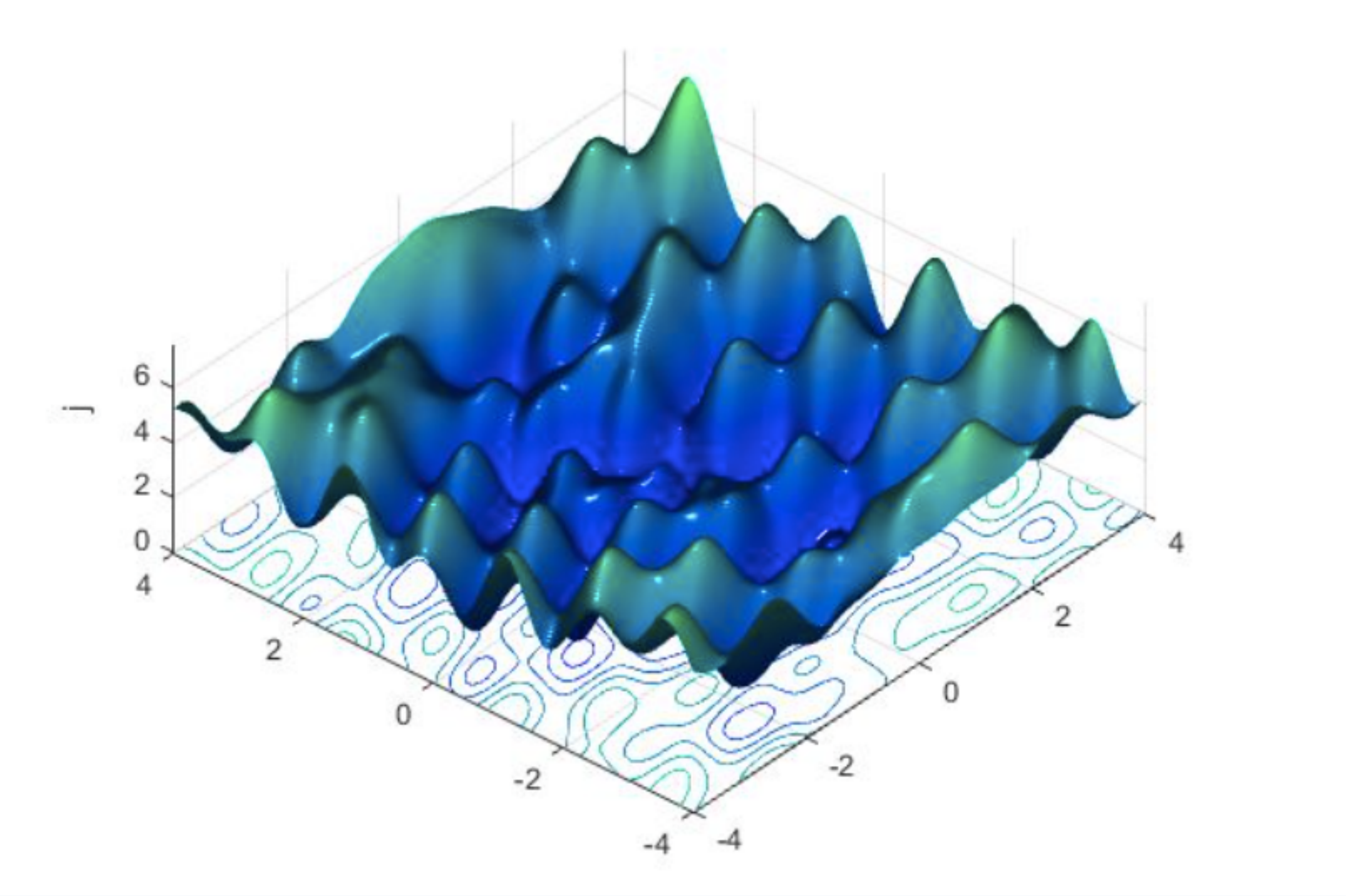
- Gradient -> allows to use error to correct the weights
- (NN is a differentiable graph)

GRADIENT DESCENT

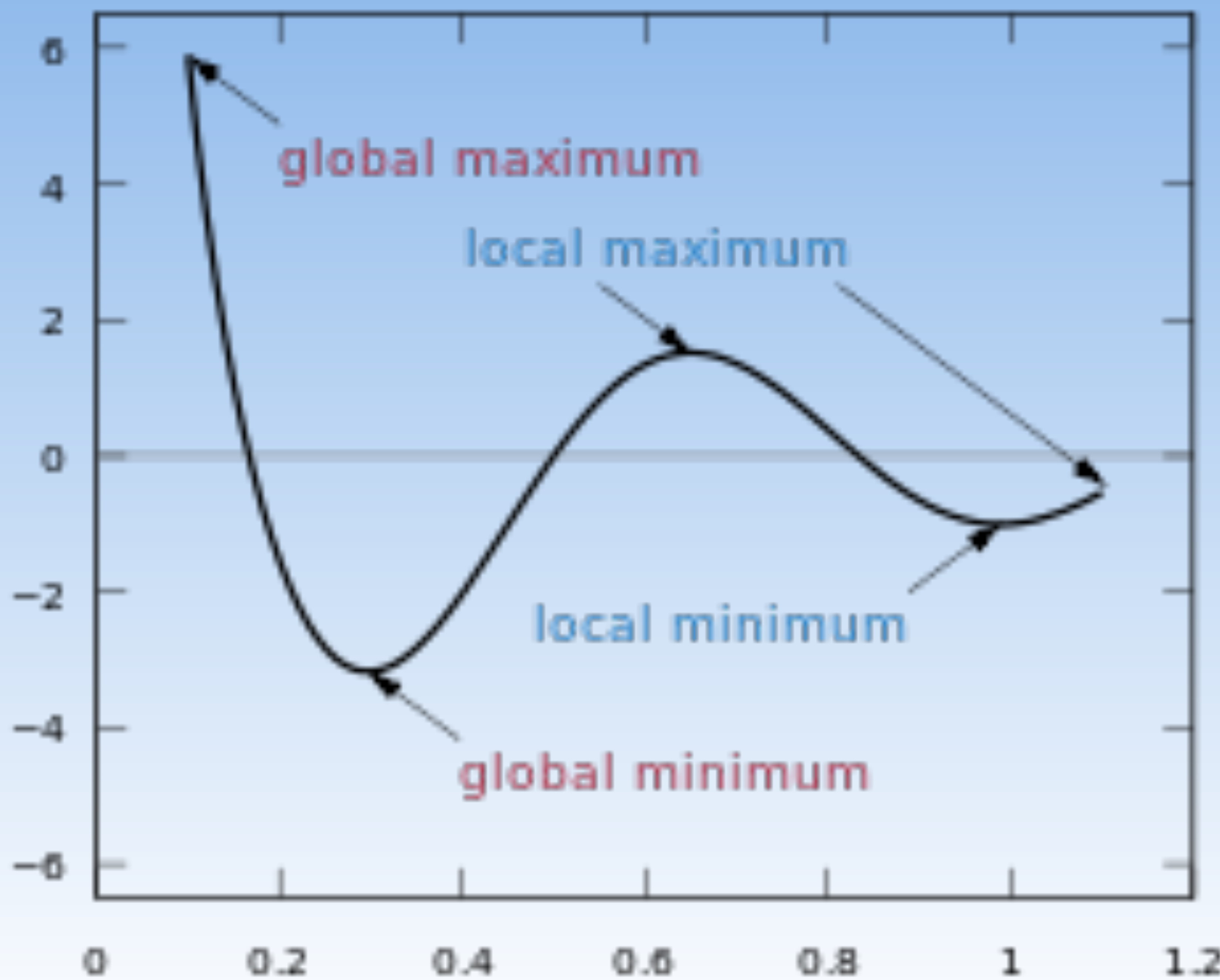


Catalit LLC

ACTUAL LOSS SURFACE



LOCAL MINIMA

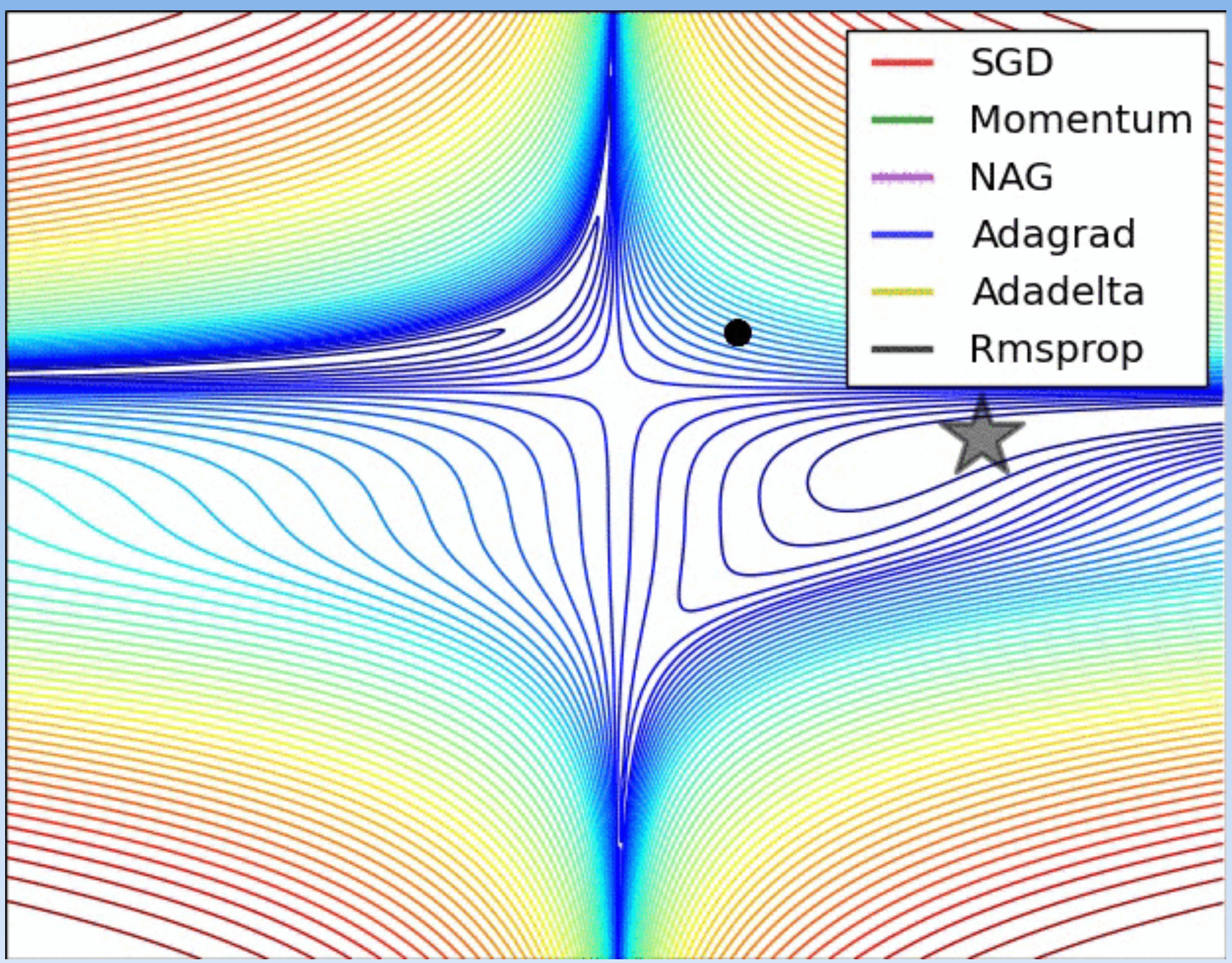


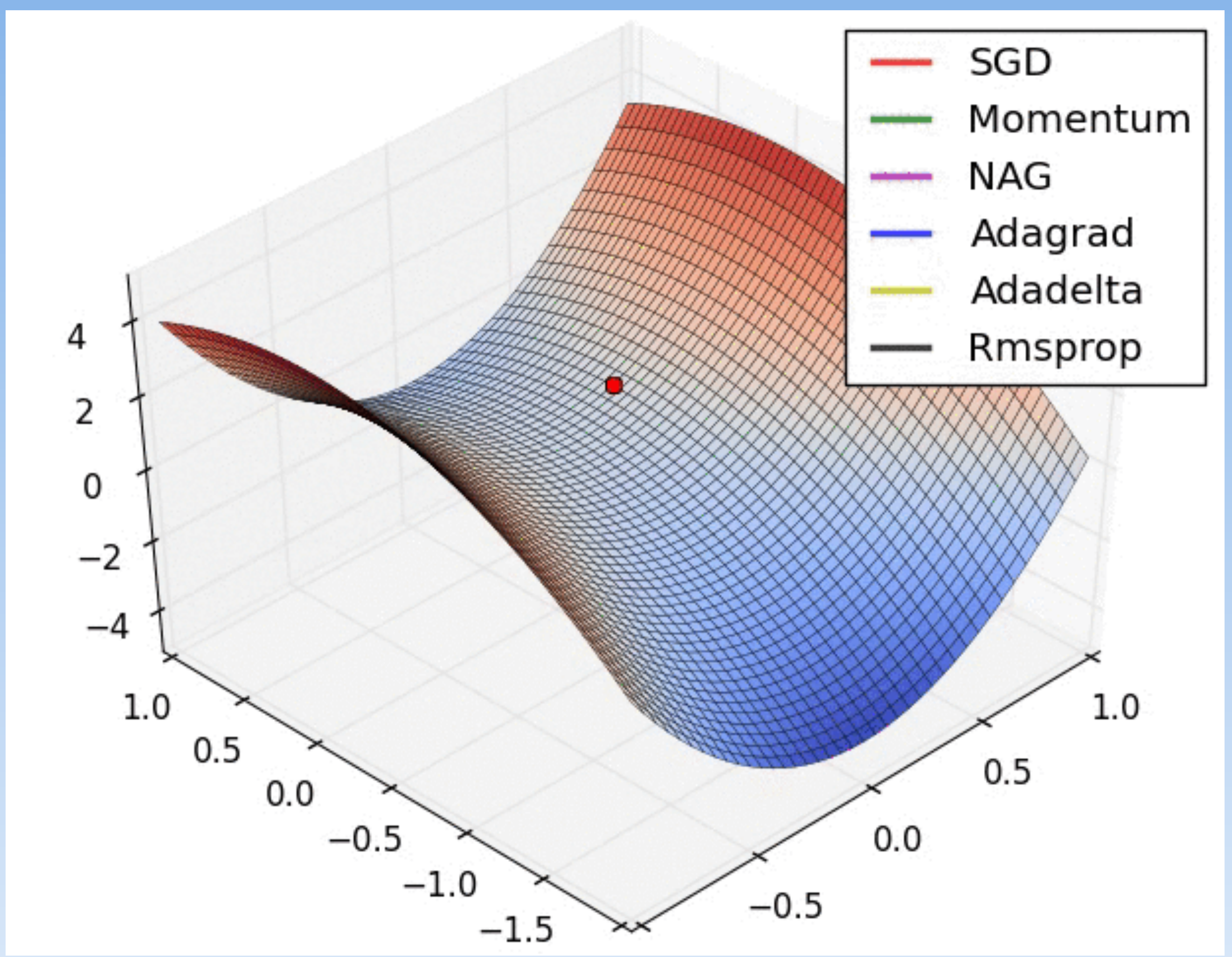
Catalit LLC

GRADIENT DESCENT

- Batch => Use whole dataset at each update
- Stochastic => Use 1 sample at each update
- Mini-batch => Use N samples at each update

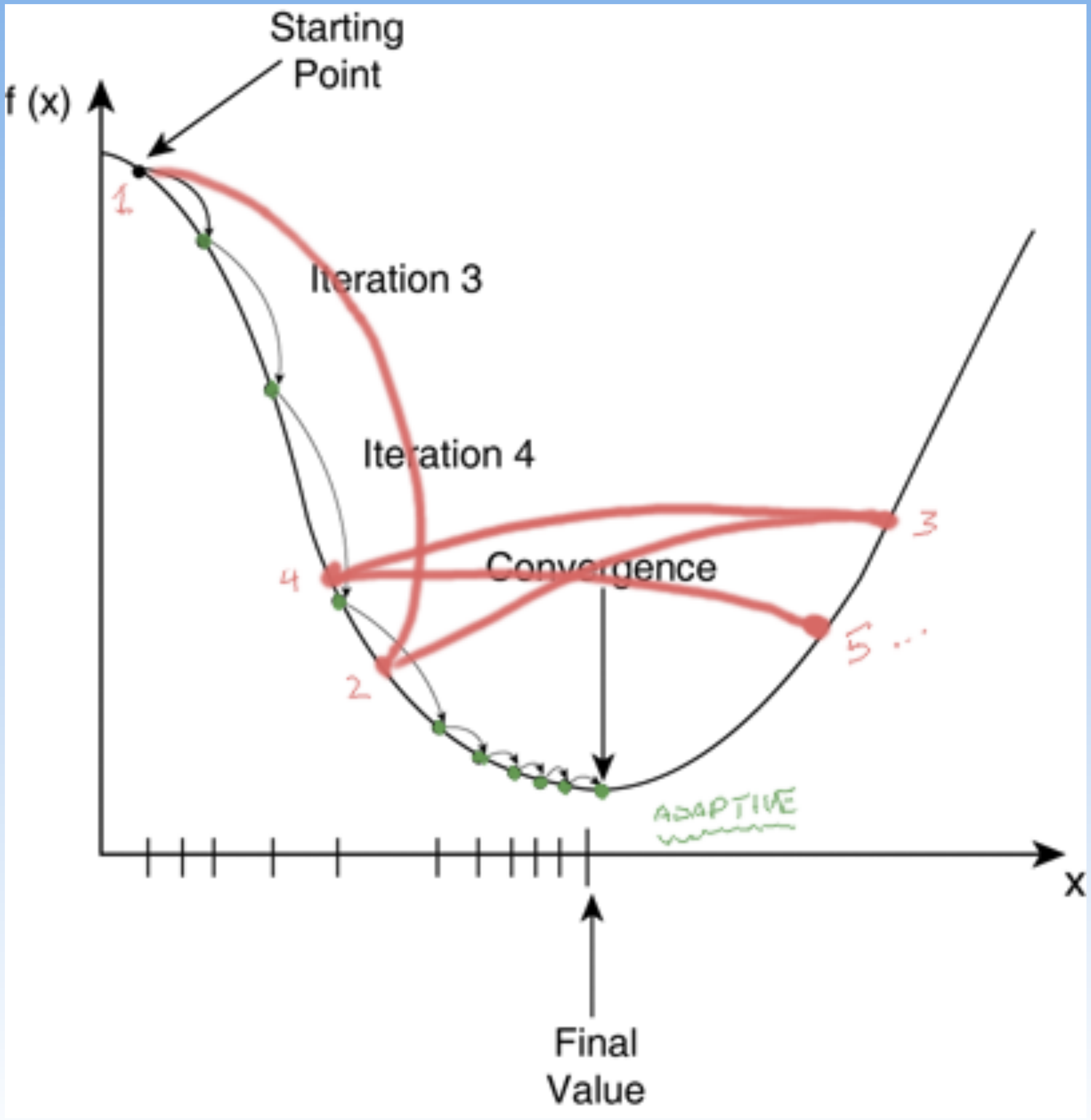
OPTIMIZERS



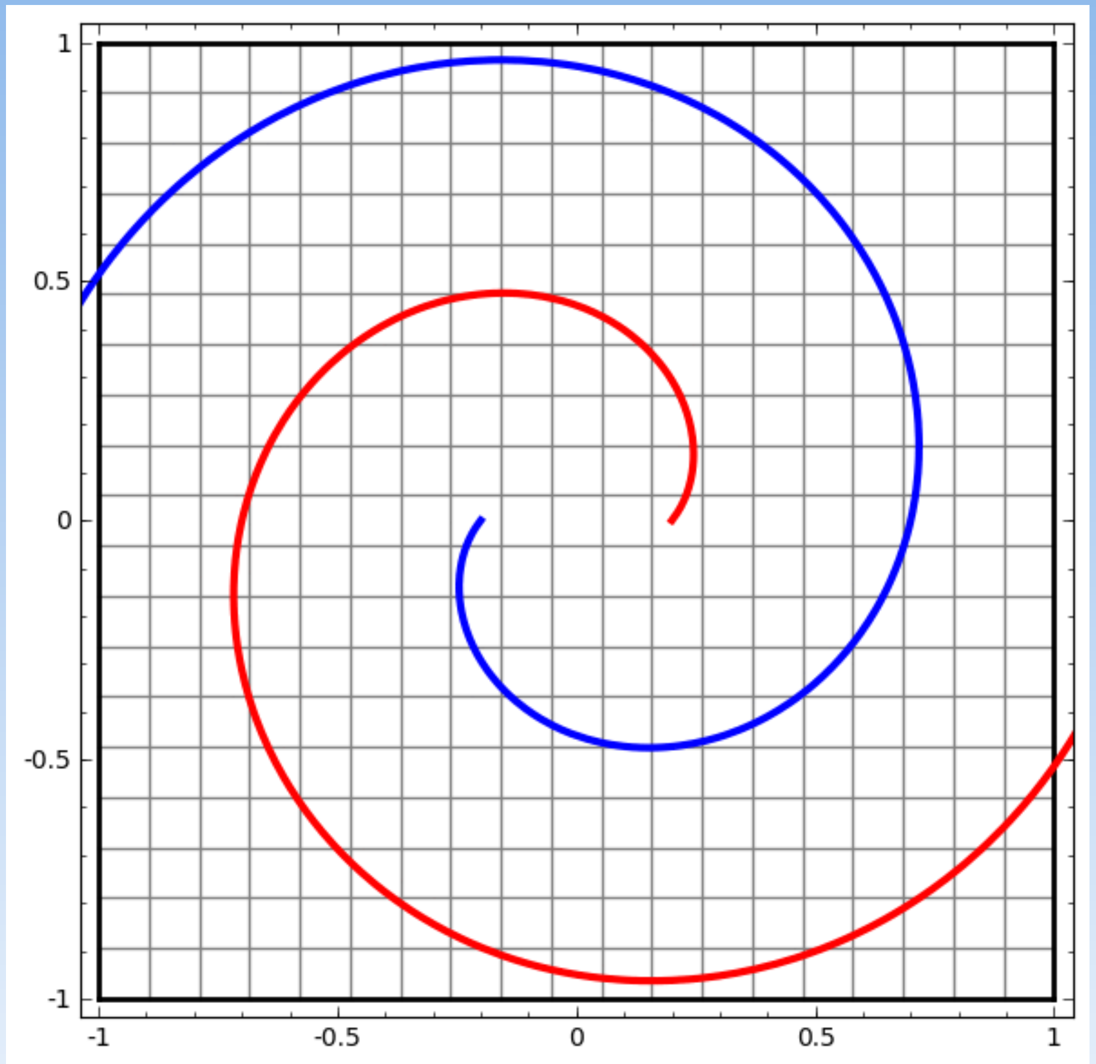


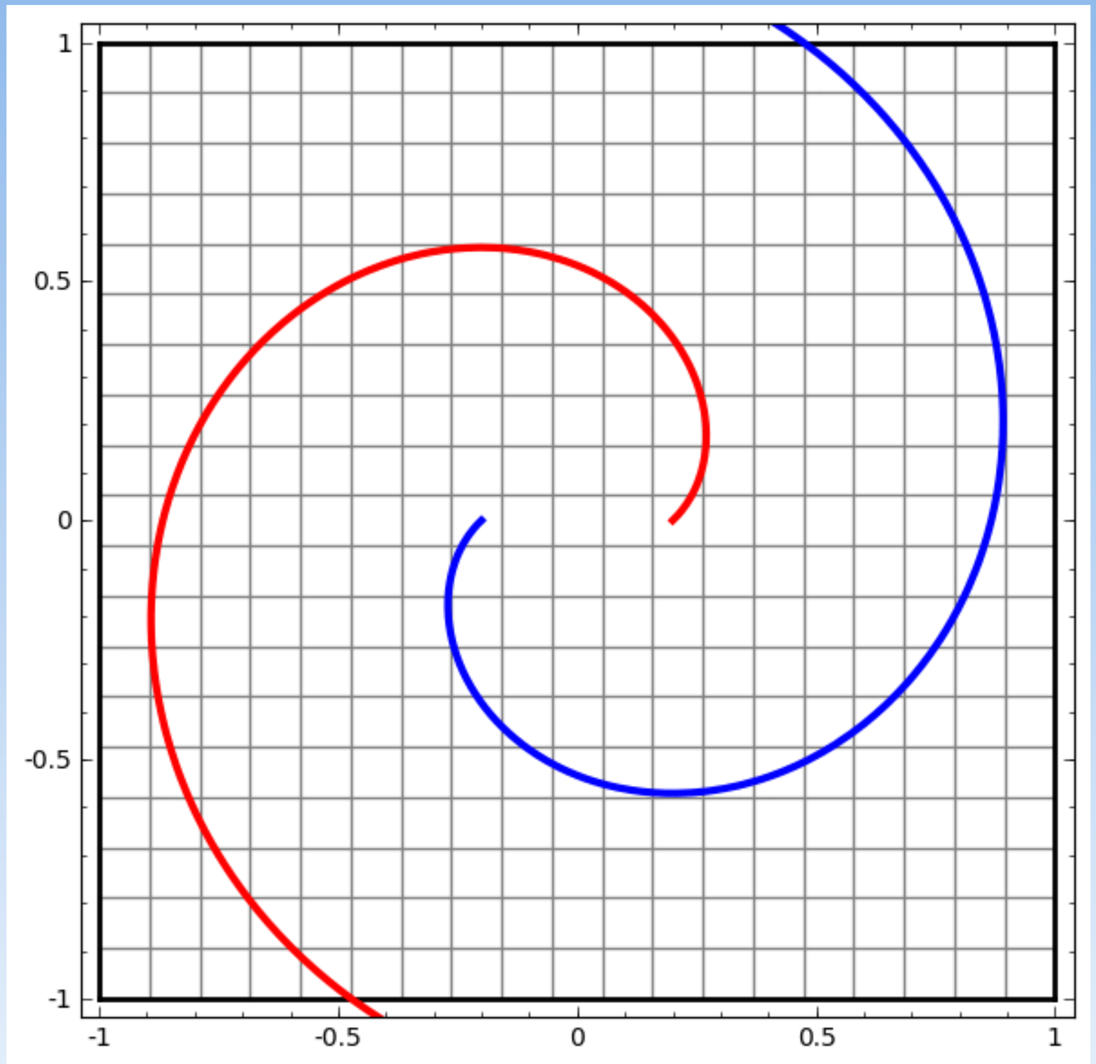
http://sebastianruder.com/optimizing-gradient-descent/

LEARNING RATE



ACTIVATION FUNCTION





LAB