Gradient Descent

Numerical derivative

Random Search

Save weight if current loss better than best loss

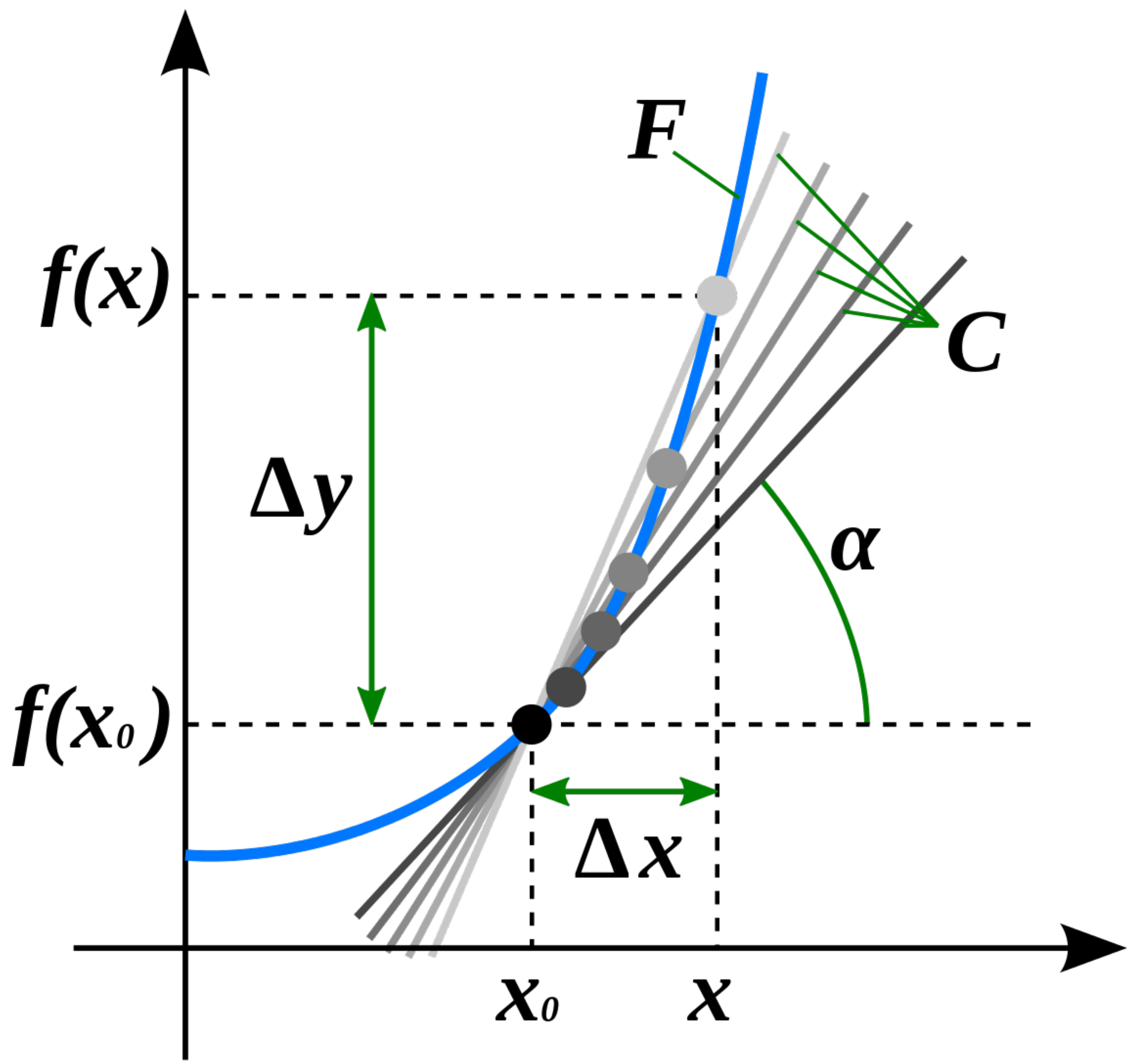
Gradient Descent

Numerical derivative

$$\frac{\partial L}{\partial w} \approx \frac{L(w+\epsilon) - L(w-\epsilon)}{2\epsilon}$$

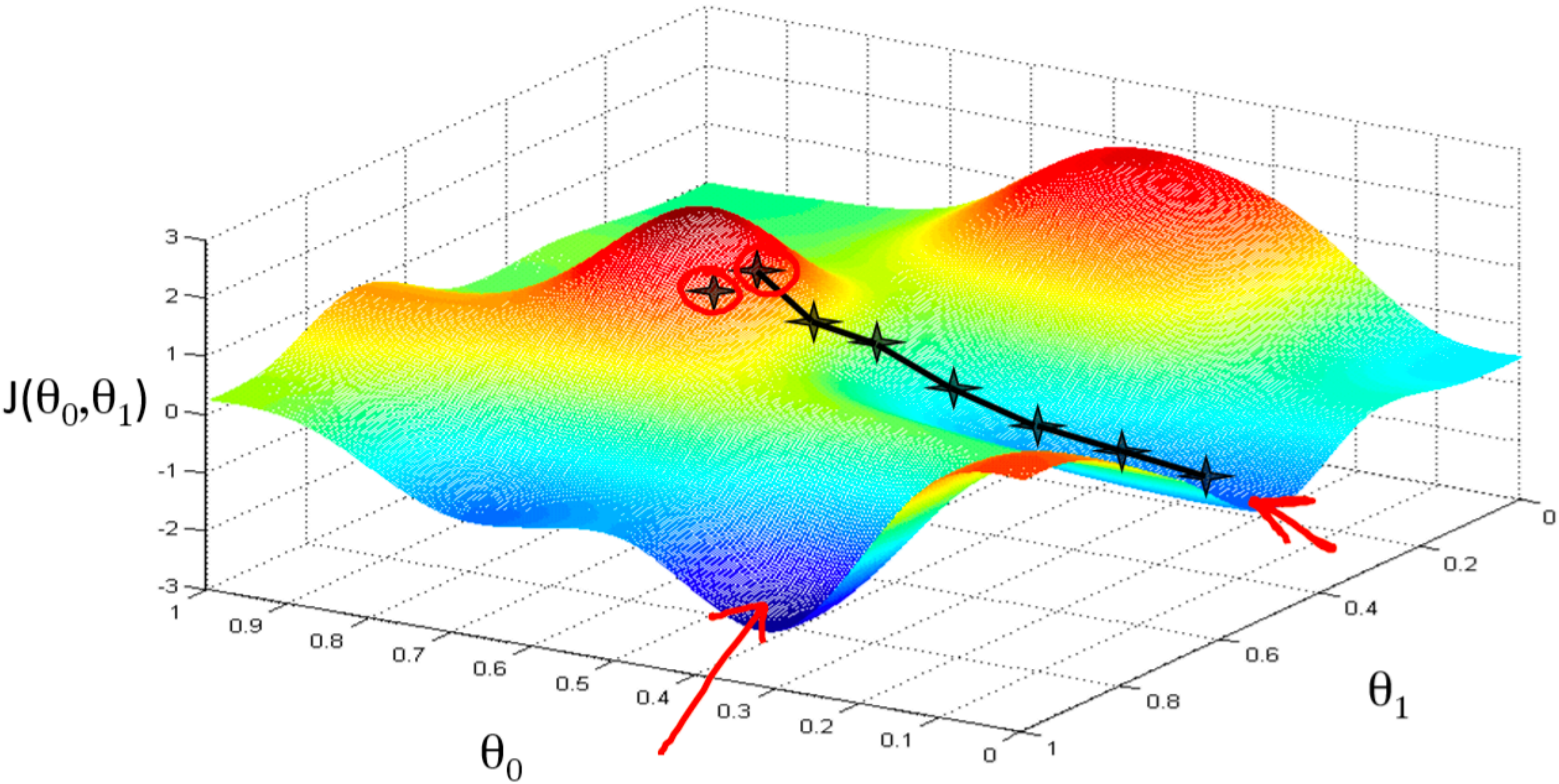
Gradient Descent

Derivative



Gradient Descent

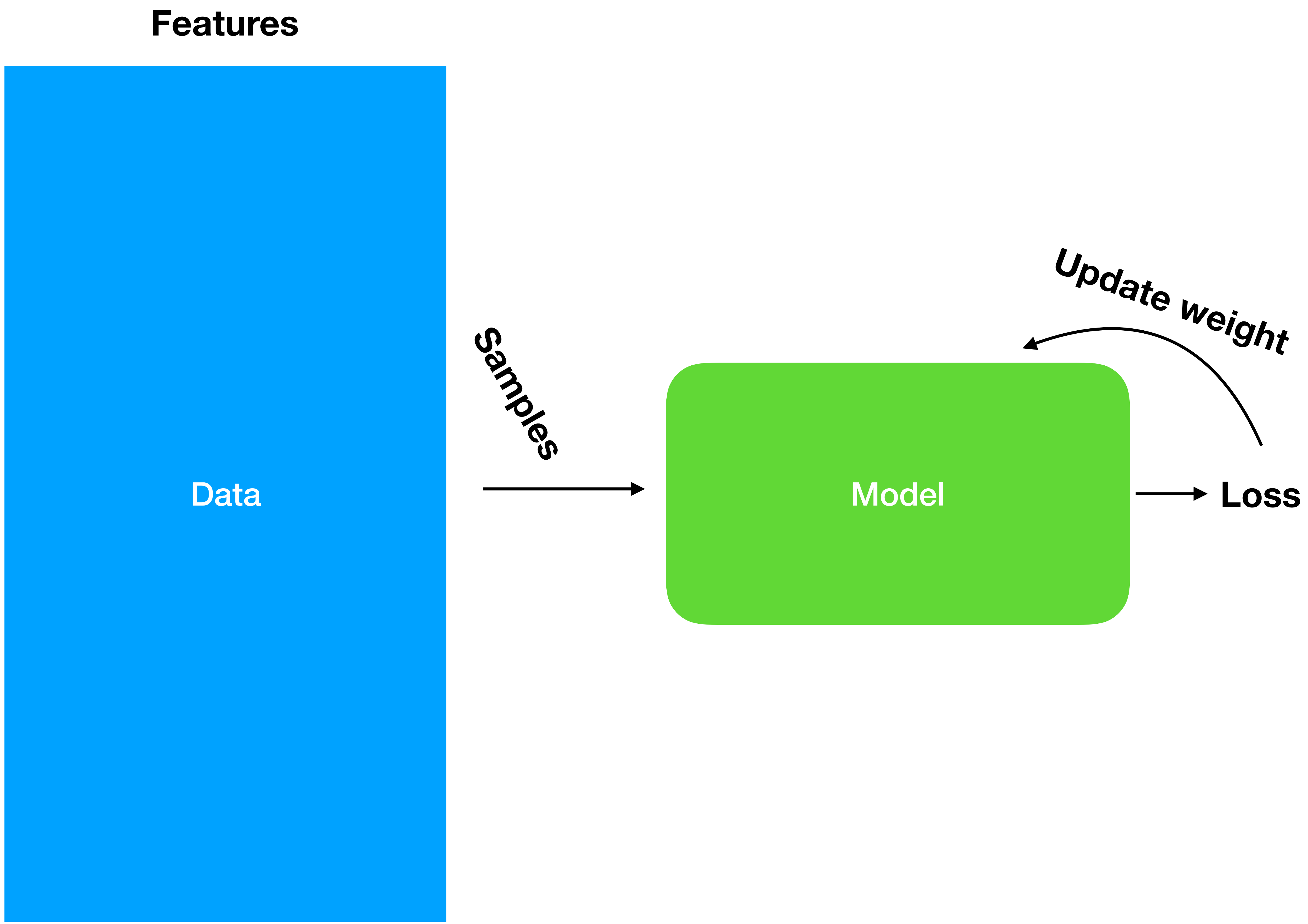
Local minima



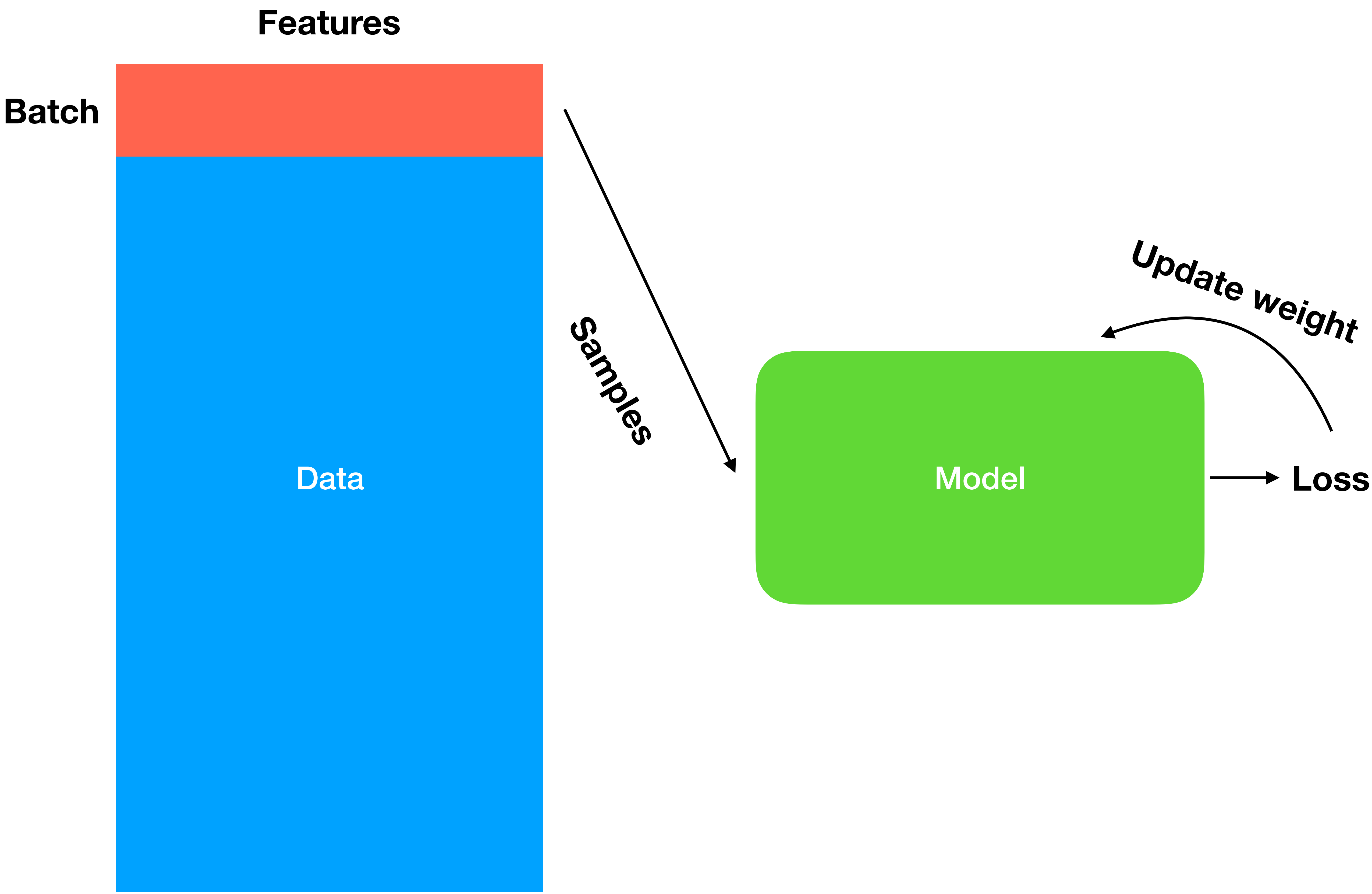
Gradient Descent

```
W = W - learning_rate * dLdW
b = b - learning_rate * dLdb
```

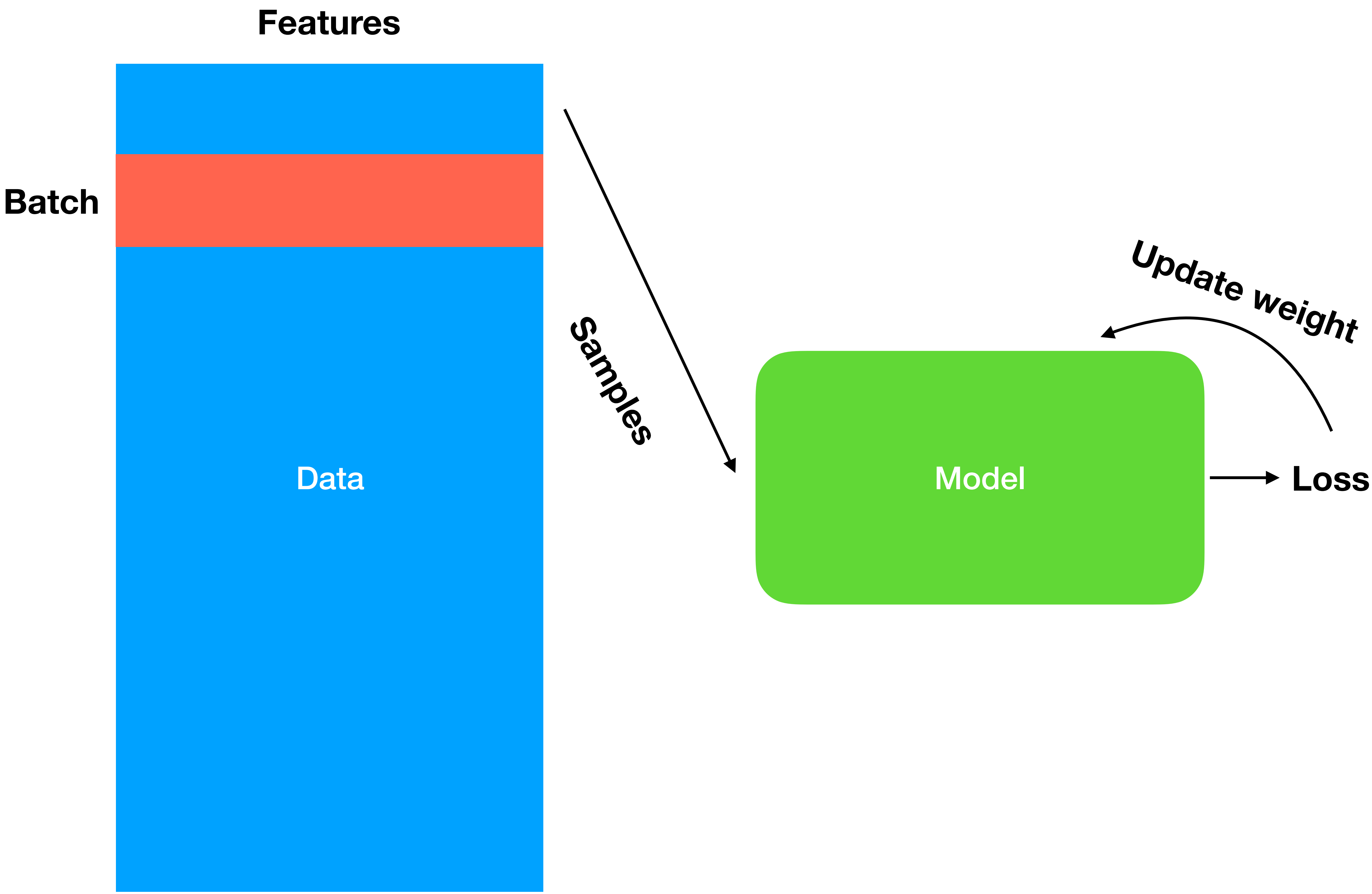
Gradient Descent



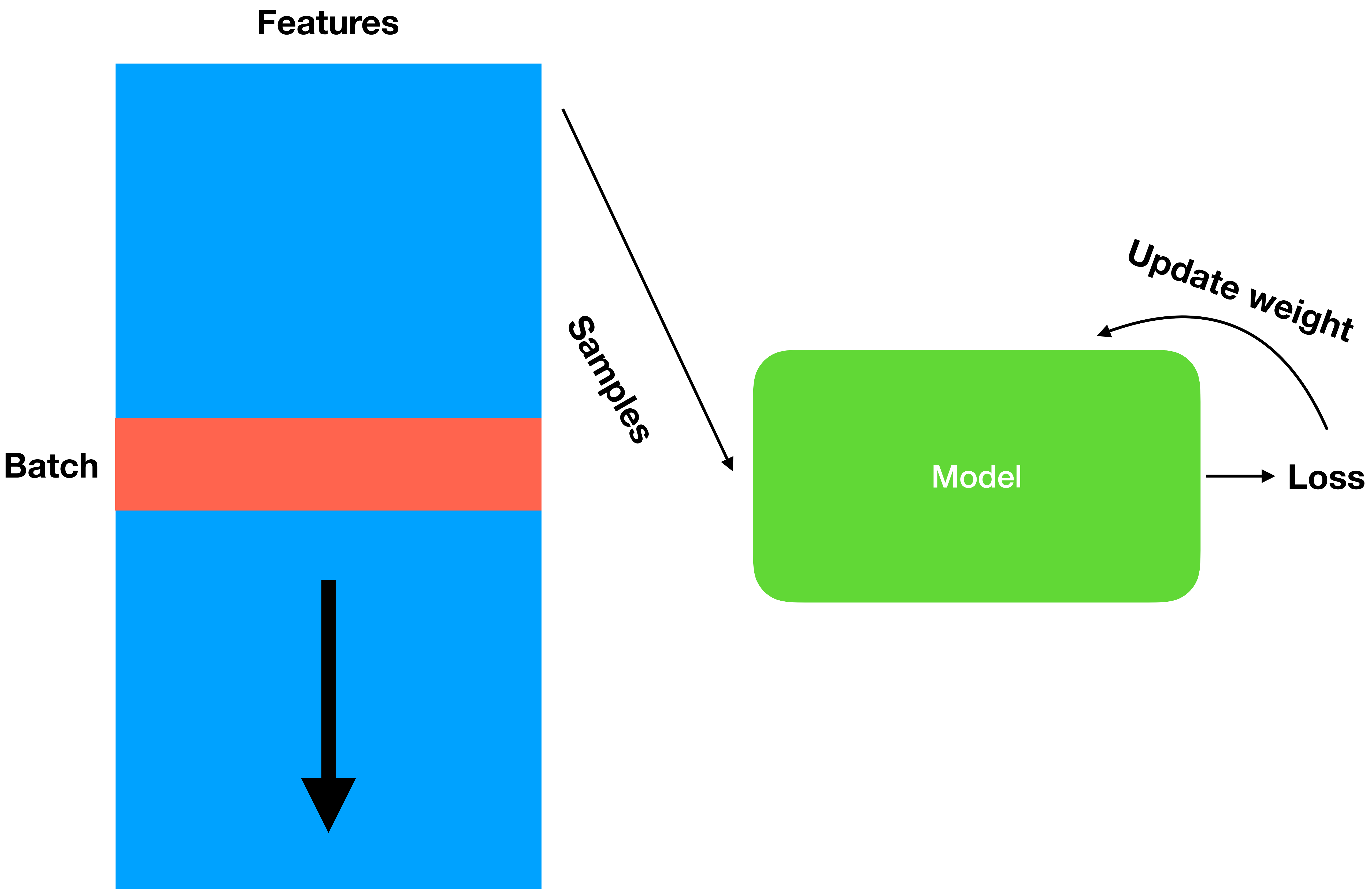
Stochastic Gradient Descent



Stochastic Gradient Descent

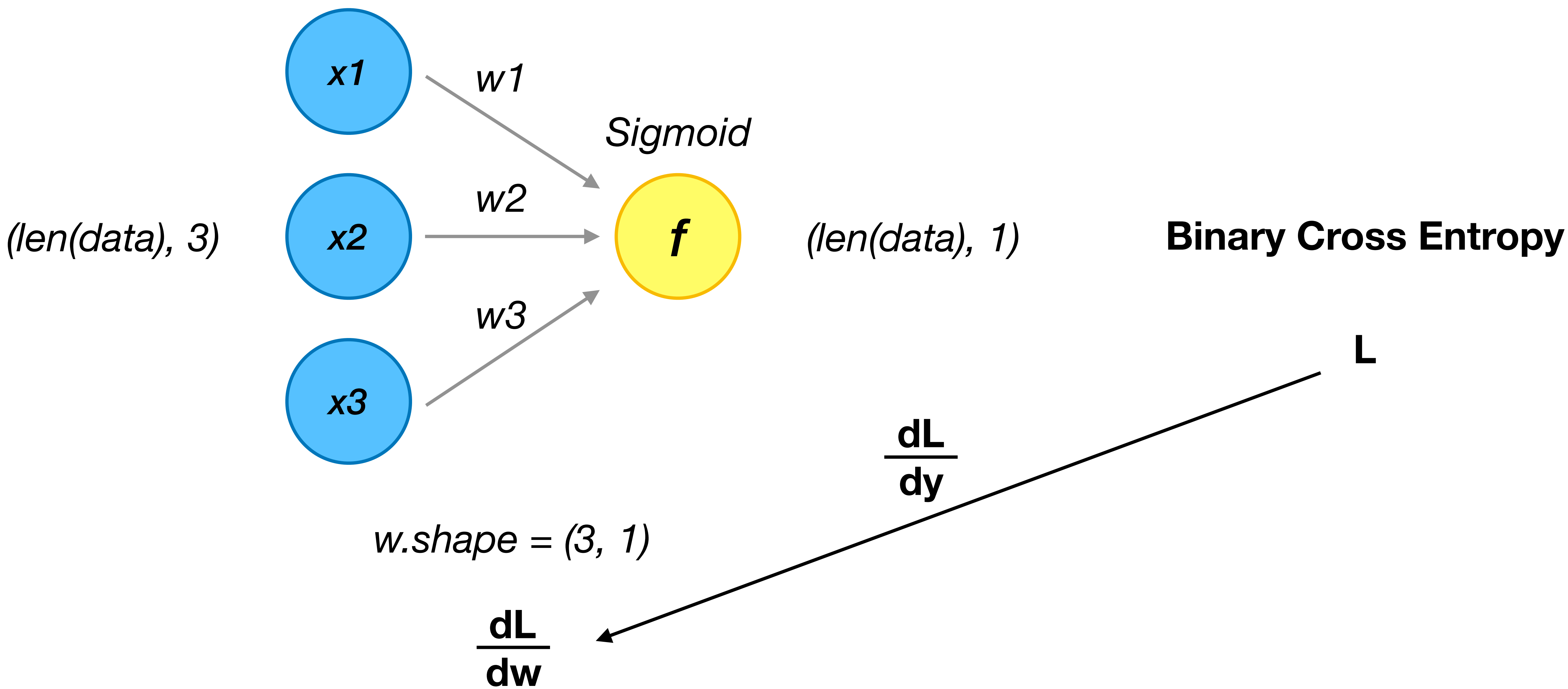


Stochastic Gradient Descent



Logistic Regression

Train



$$\frac{dL}{dw} = \frac{dL}{dy}\frac{dy}{dw}$$

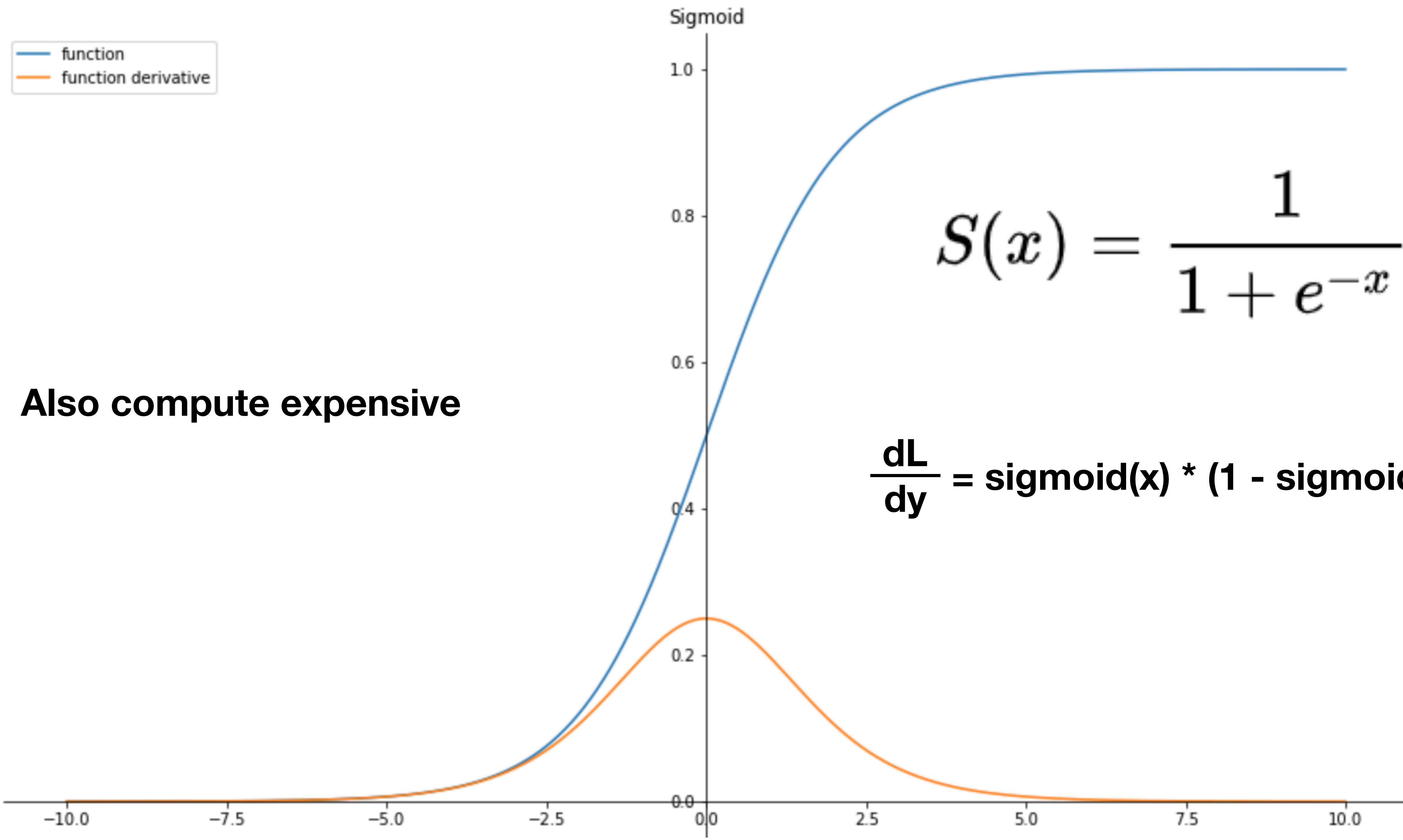
y = sigmoid( )

Regularization

BCELoss = - (y * log(pred) + (1 - y) * log(1 - pred)) + reg * R(W)

reg - hyperparameter

Sigmoid

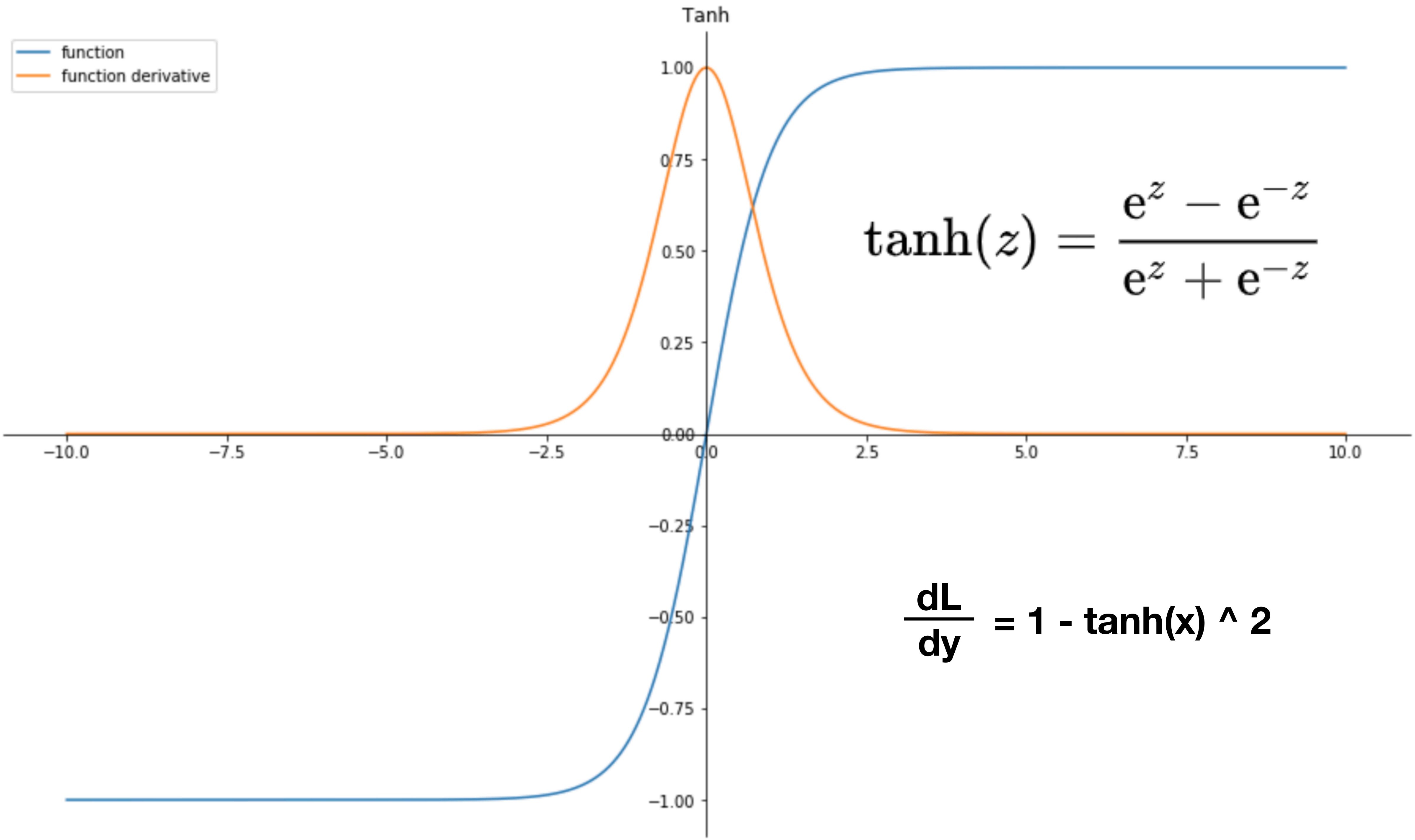


$$S(x) = \frac{1}{1 + e^{-x}}$$

Also compute expensive

$$\frac{dL}{dy} = \text{sigmoid}(x) * (1 - \text{sigmoid}(x))$$

Tanh

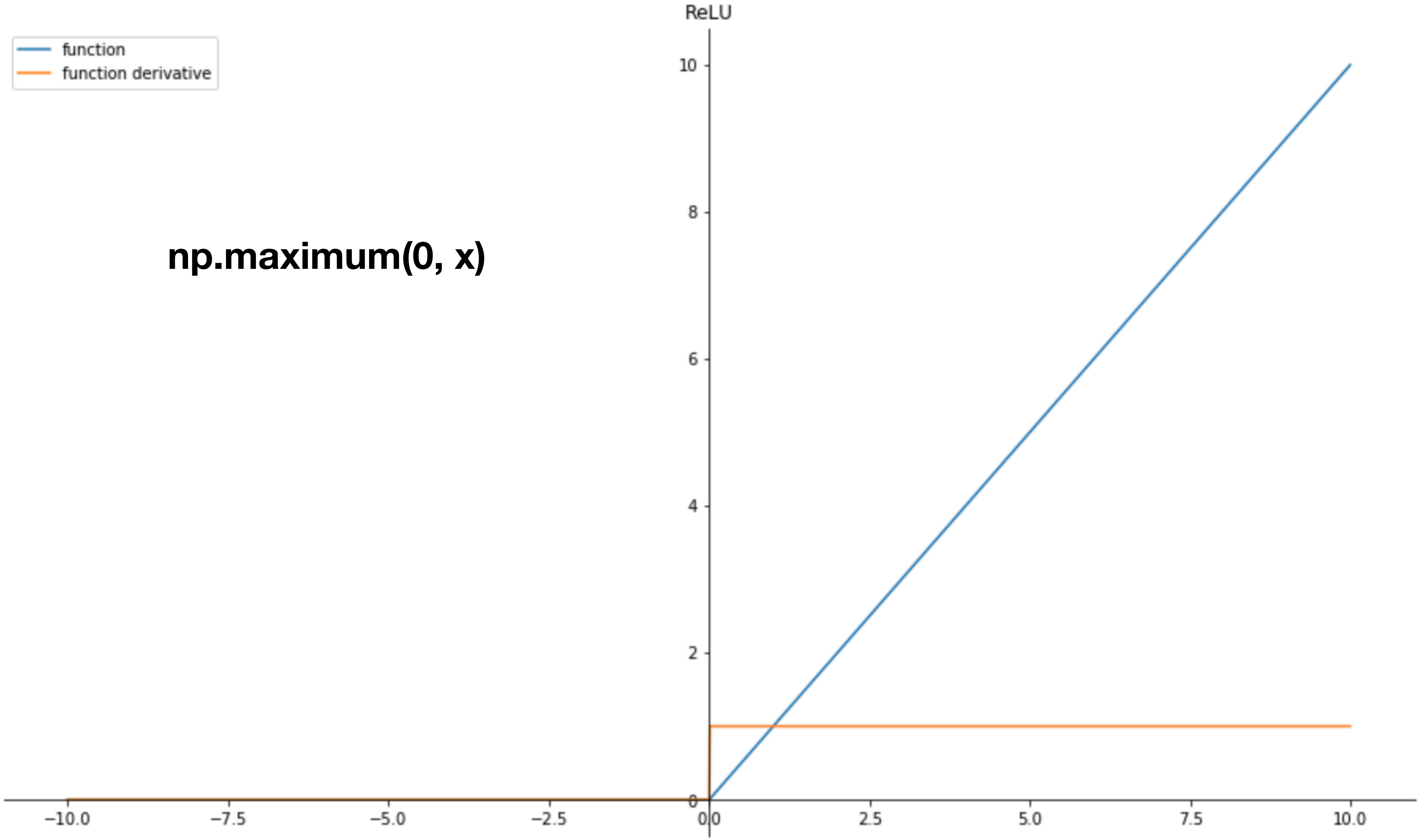


$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

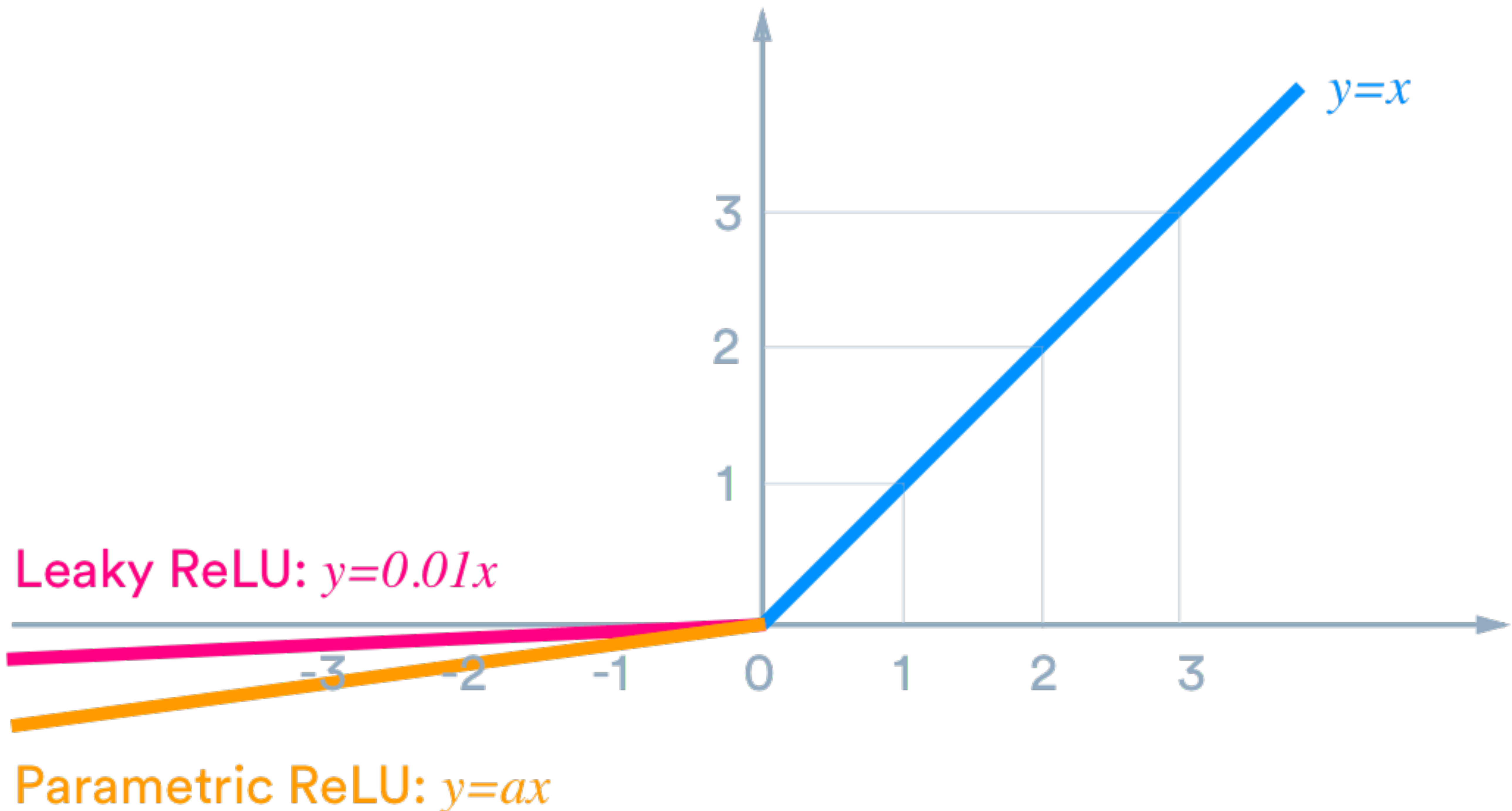
$$\frac{dL}{dy} = 1 - \tanh(x) \wedge 2$$

ReLU

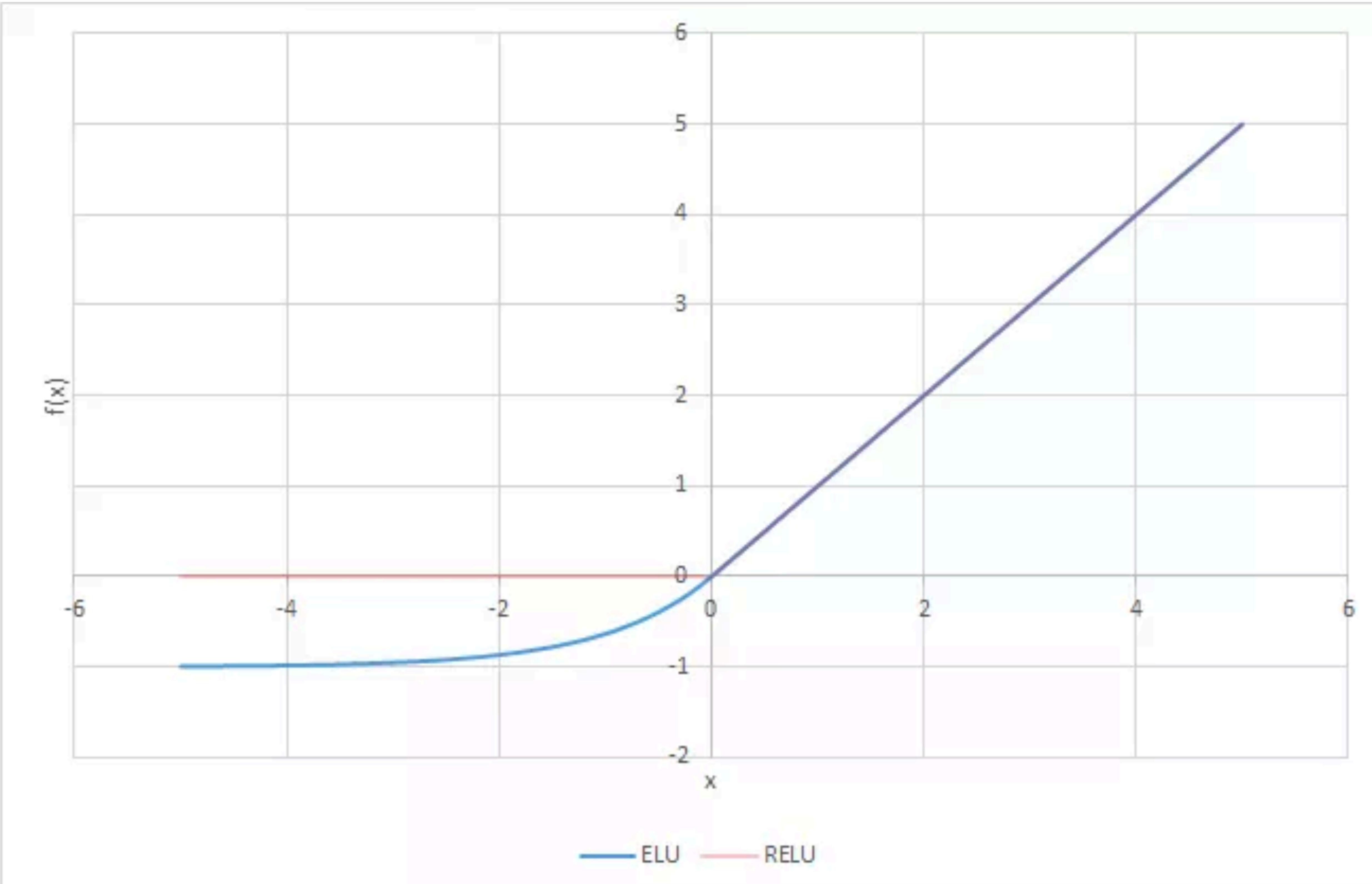
np.maximum(0, x)



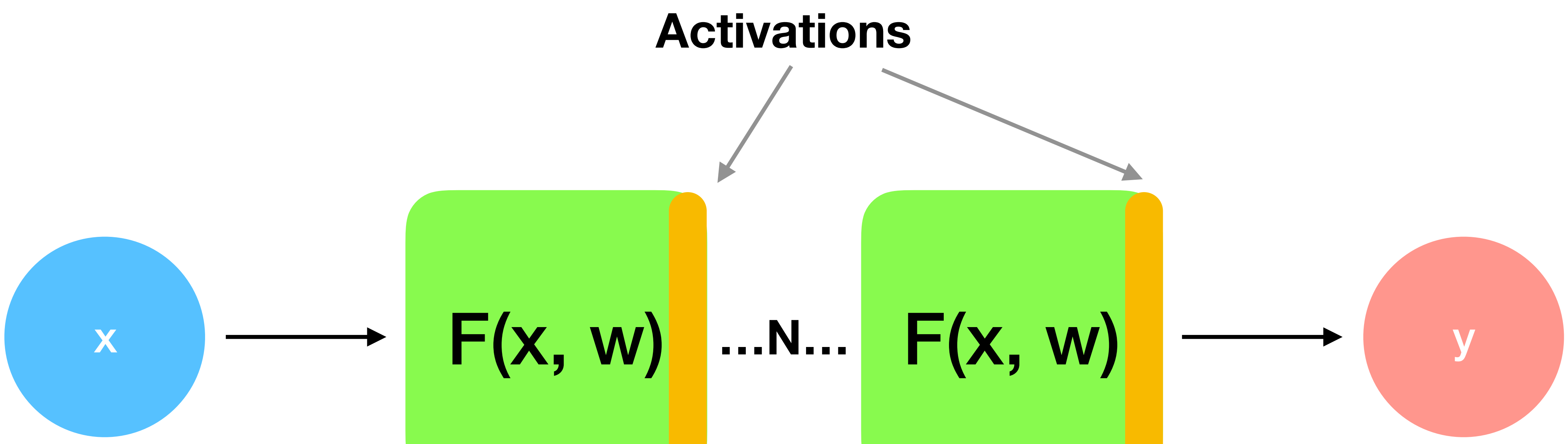
Leaky ReLU



ELU



Neural Networks

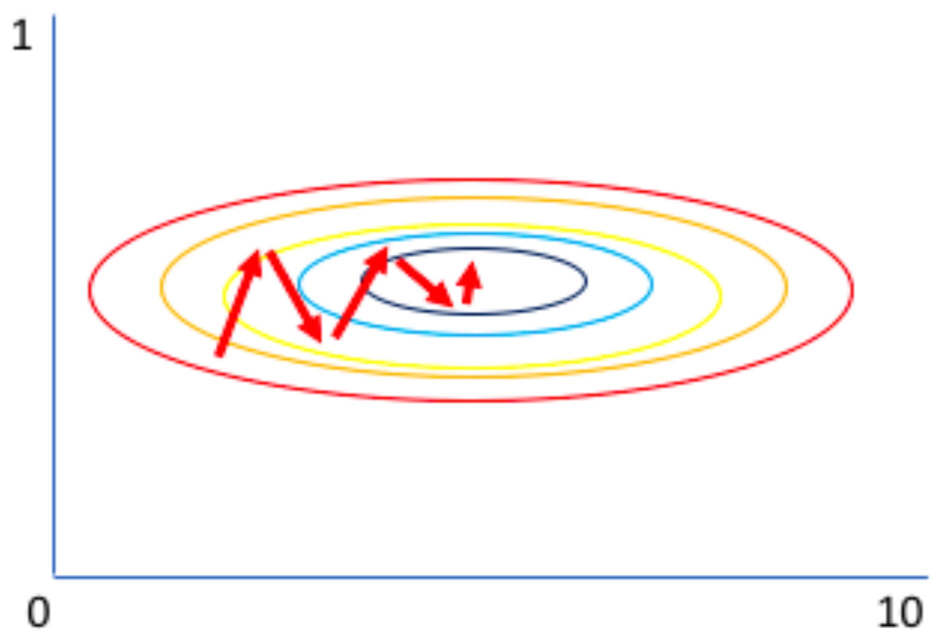


Important things

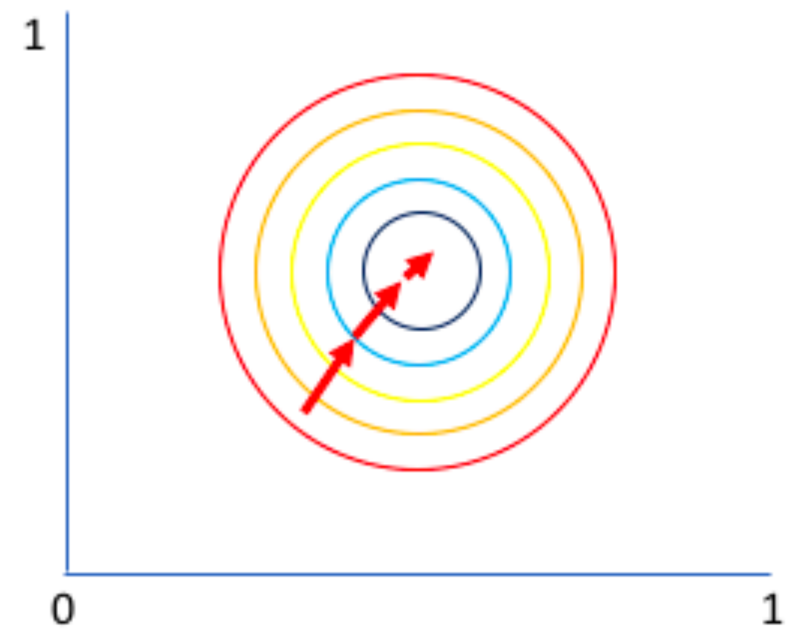
- Don’t use sigmoid as inner activation in MLP
- But then we talk about how you can use sigmoid as inner activation
- Normalize your data
- Early stopping

Important things

Why normalize?



Gradient of larger parameter dominates the update



Both parameters can be updated in equal proportions

Early Stopping