Lecture - 6

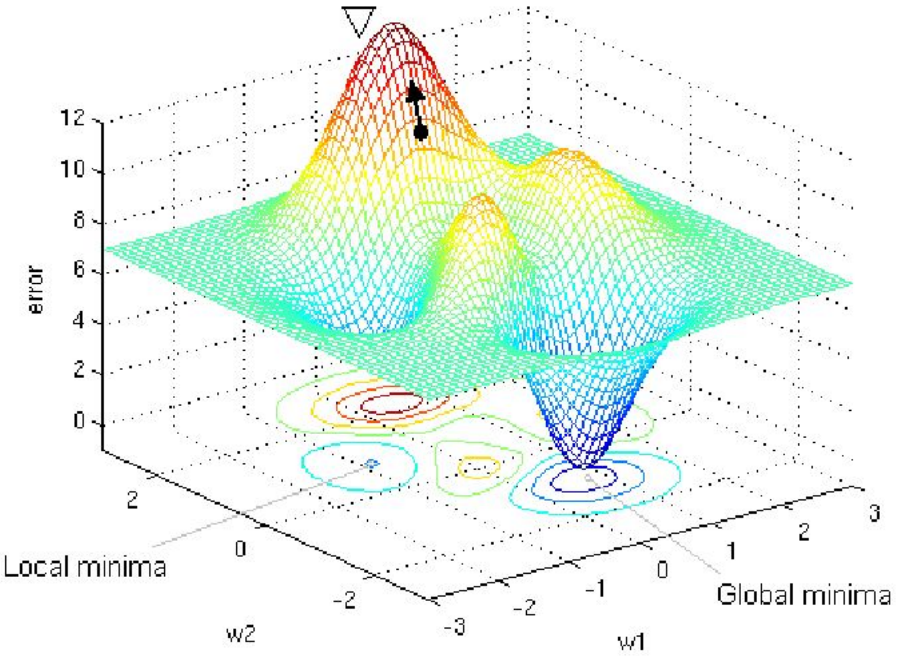
Training Neural Network (contd.)

So far ...

- Networks are trained to minimize total "error" on a training set
- We can use gradient descent to minimize the error
- The gradient of the error with respect to network parameters is computed through backpropagation
- We have implemented Neural Network from scratch.

Module 1: The error surface, convergence, learning rate

The Error Surface



The Error Surface

Popular hypothesis: **In a large network**

- Saddle points are far more common than local minima
- Local minima are not too bad (many recent studies)

- Grzegorz Swirszcz, Wojciech Marian Czarnecki, Razvan Pascanu: **Local minima in training of deep networks.** CoRR abs/1611.06310 (2016)
- Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, Yann LeCun: **The Loss Surfaces of Multilayer Networks.** AISTATS 2015

The Error Surface

Popular hypothesis: **In a large network**

- Saddle points are far more common than local minima
- Local minima are not too bad

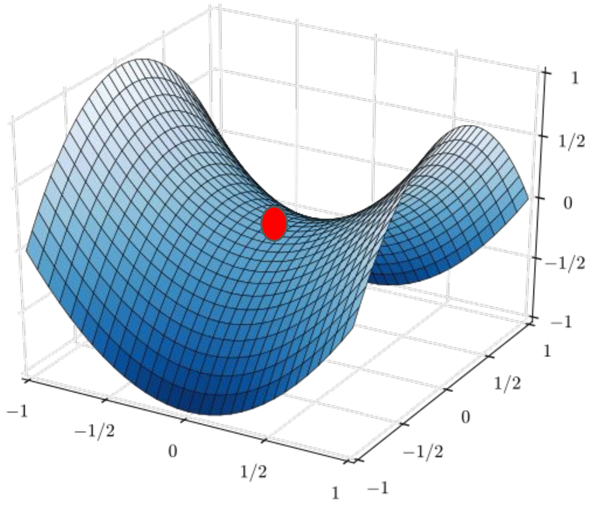
What is a Saddle Point

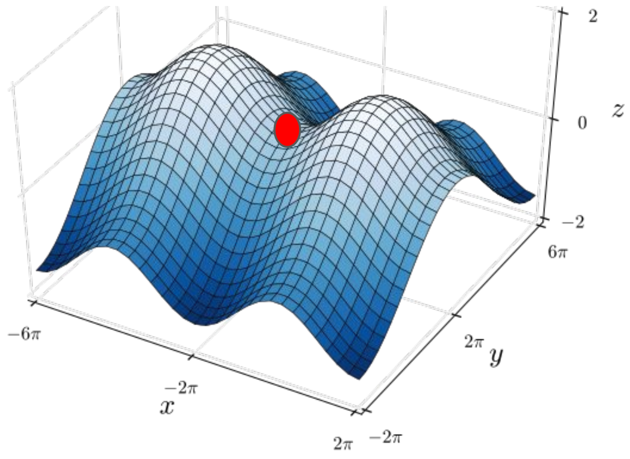
- A point where gradient is zero, and the value of the error surface increases in some directions but decreases in some other directions.

The Error Surface

What is a Saddle Point

- A point where gradient is zero, and the value of the error surface increases in some directions but decreases in some other directions.





The Error Surface

What is a Saddle Point

- A point where gradient is zero, and the value of the error surface increases in some directions but decreases in some other directions.
- Gradient descent often stuck at saddle point

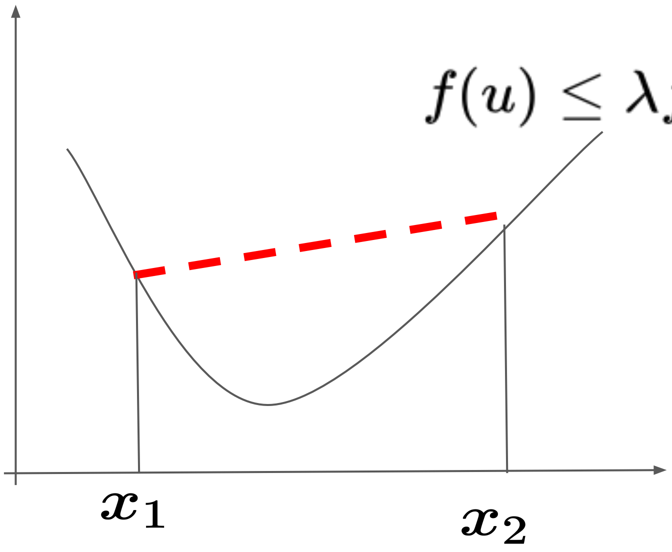
So far ...

- Neural nets can be trained via gradient descent that minimizes a loss function
- Backpropagation can be used to derive the derivatives of the loss
- For large networks, the loss function may have a large number of unpleasant saddle points
  – Which backpropagation may find

Convergence of gradient descent

- In the discussion so far we have assumed the training arrives at a local minimum
- Does it always converge?
- How long does it take?
- Hard to analyze for an MLP, but we can look at the problem through the lens of convex optimization

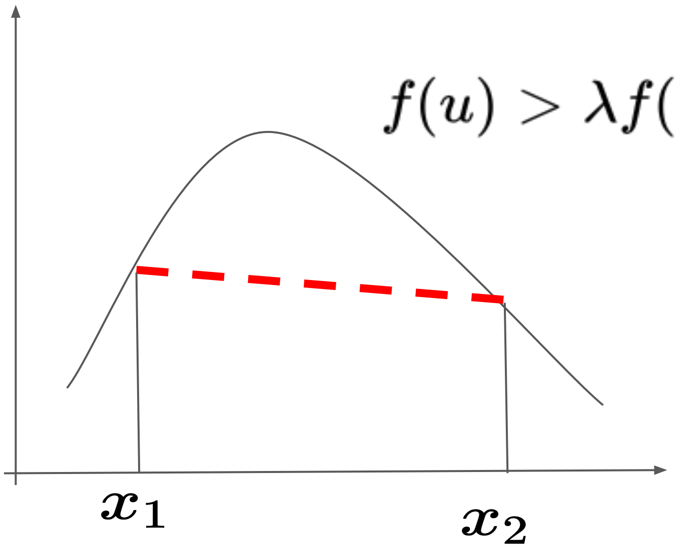
Convex Function



For any two points x_1 and x_2:

$$f(u) \leq \lambda f(x_1) + (1-\lambda) f(x_2), \forall u \in [x_1, x_2], \lambda \in \{0, 1\}$$

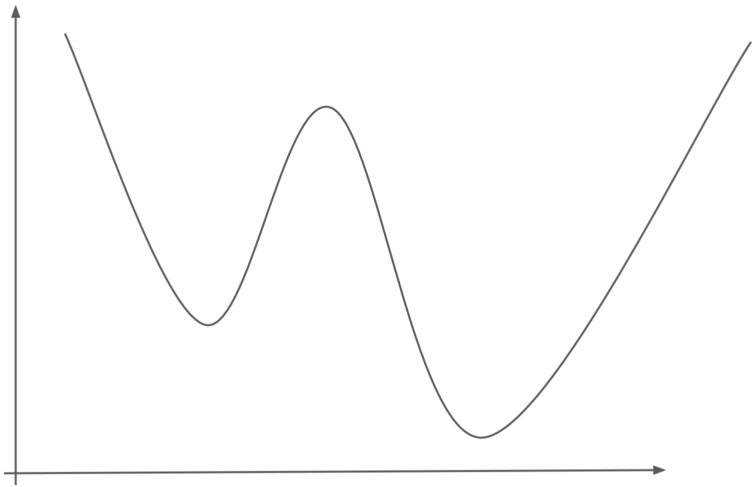
Concave Function



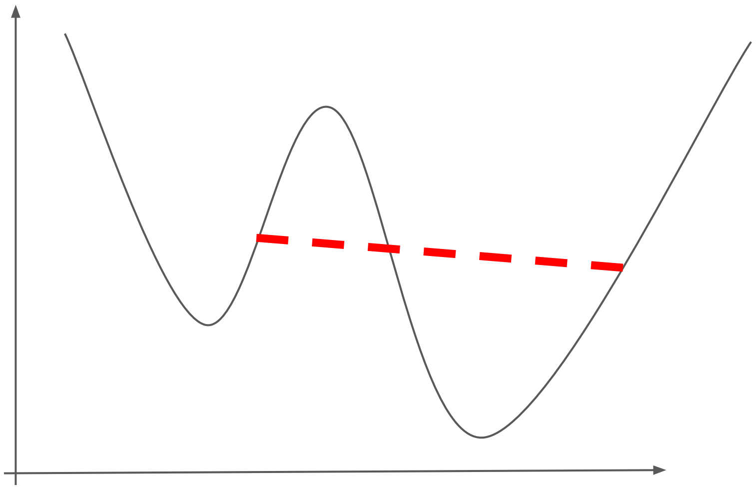
For any two points x_1 and x_2:

$$f(u) > \lambda f(x_1) + (1-\lambda) f(x_2), \forall u \in [x_1, x_2], \lambda \in \{0, 1\}$$

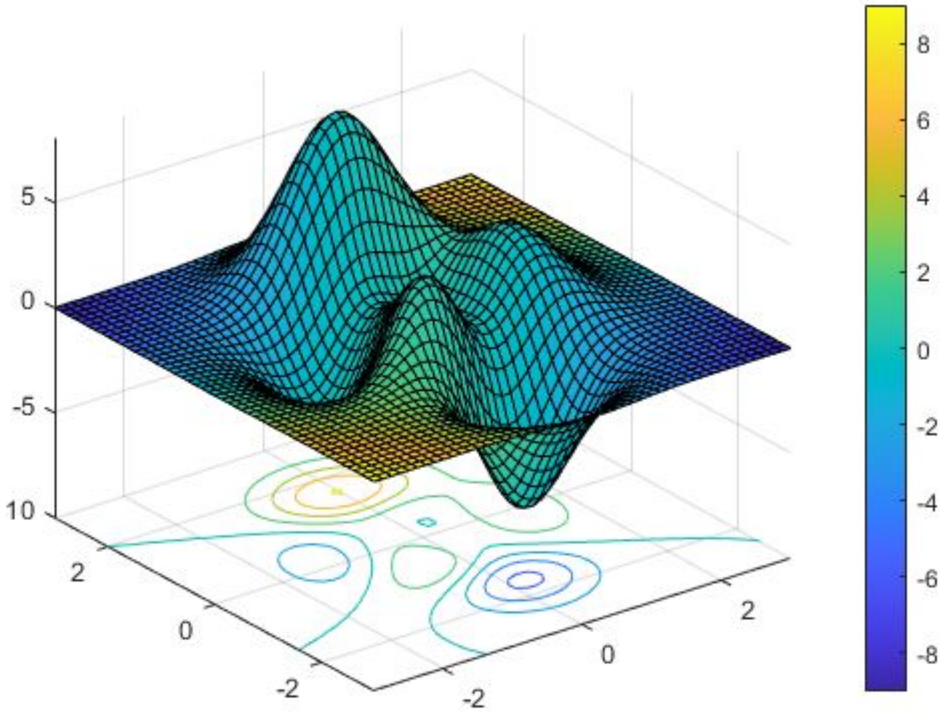
Non-convex Function



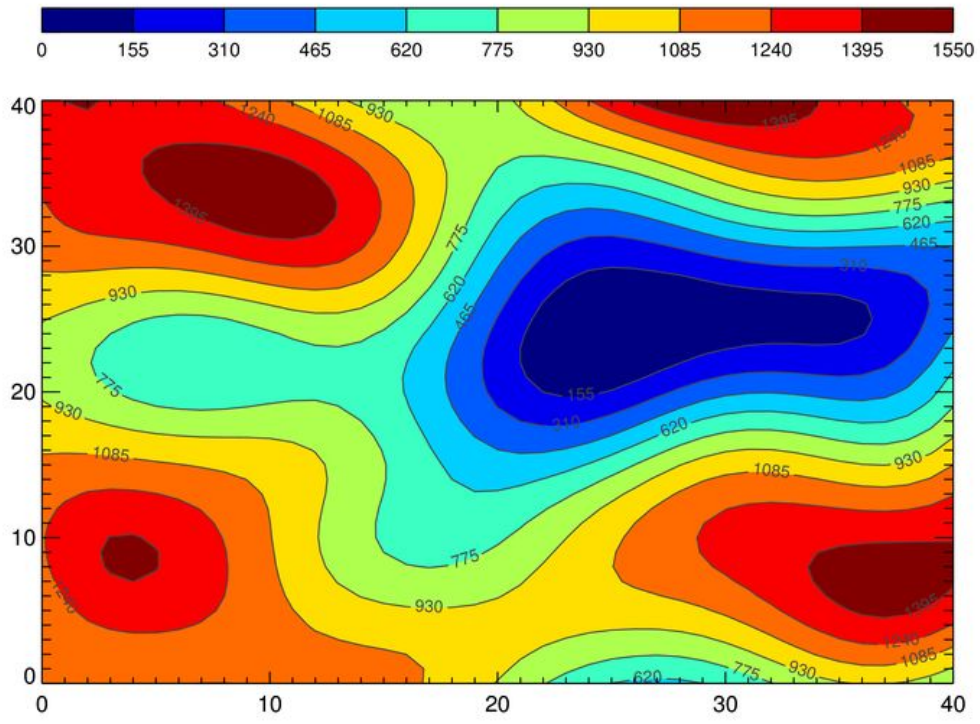
Non-convex Function



Contour representation



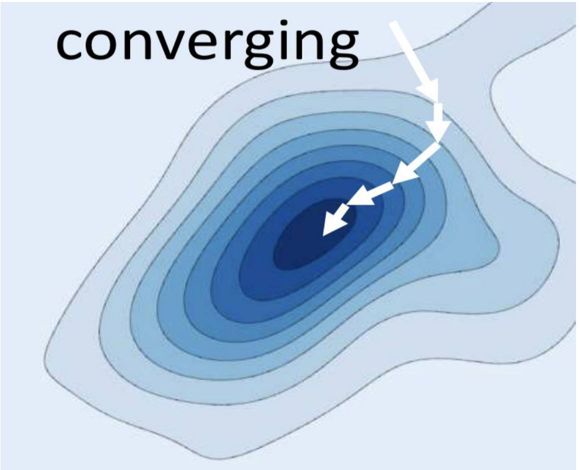
Contour representation

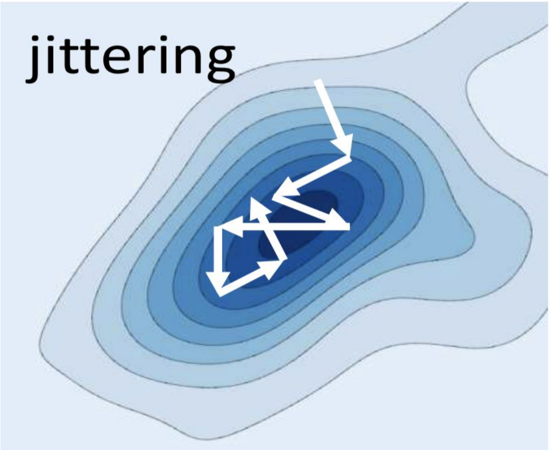


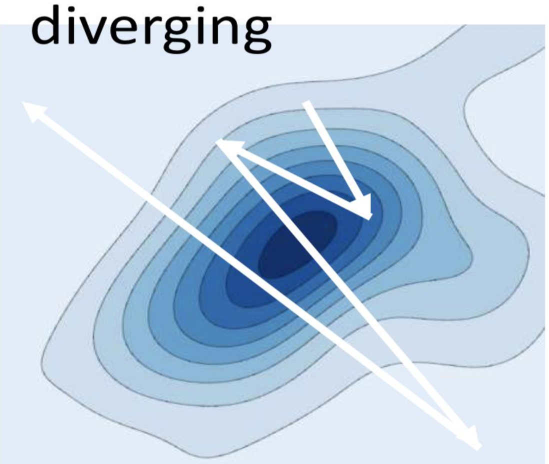
Convergence of Gradient Descent

- An iterative algorithm is said to converge to a solution if the value updates arrive at a fixed point
  – Where the gradient is 0 and further updates do not change the estimate

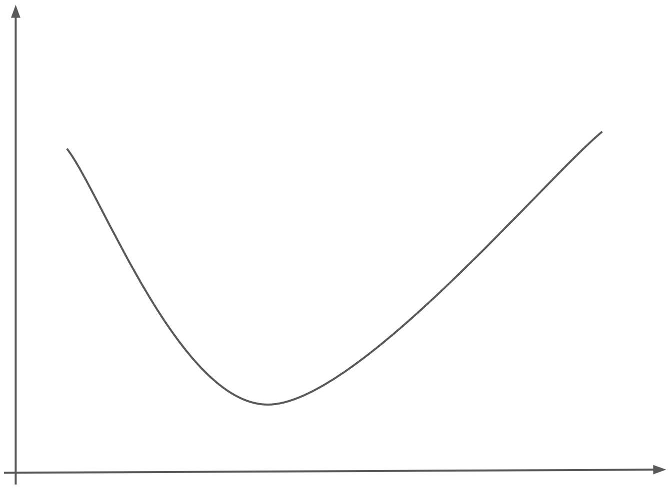
Convergence of Gradient Descent



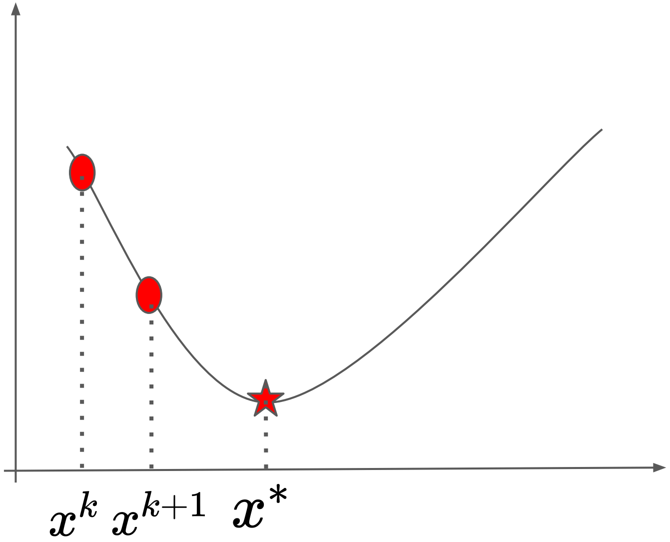




Convergence Rate

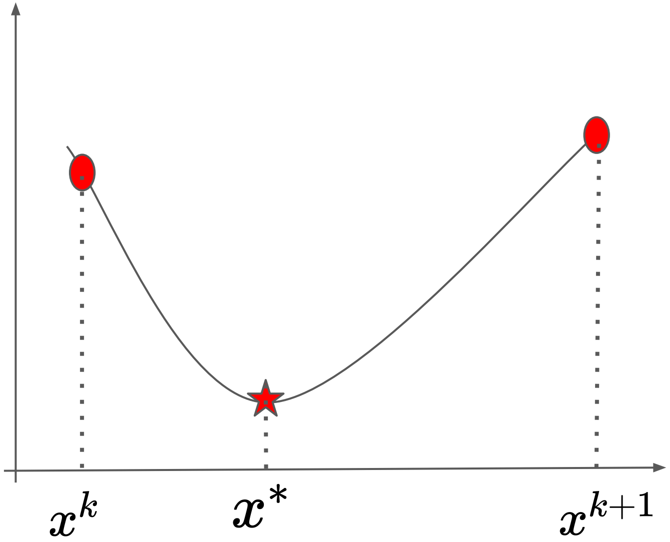


Convergence Rate



$$R = \frac{|f(x^*) - f(x^{k+1})|}{|f(x^*) - f(x^k)|}$$

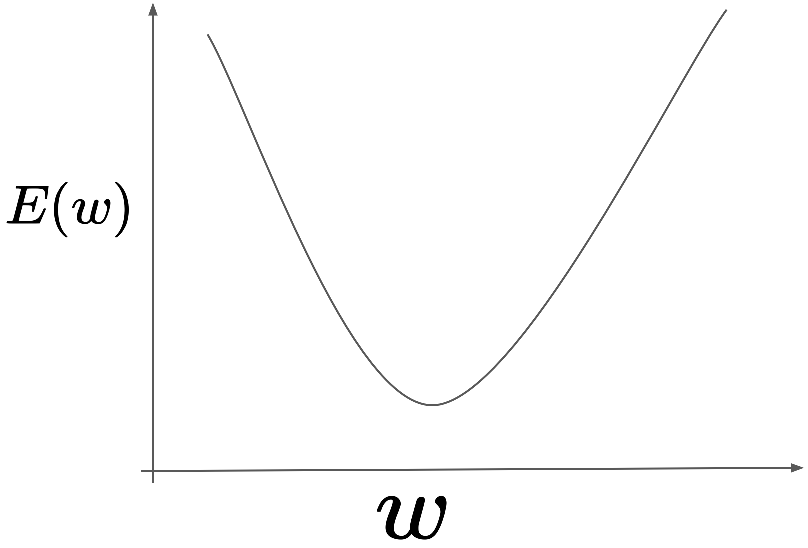
Convergence Rate



$$R = \frac{|f(x^*) - f(x^{k+1})|}{|f(x^*) - f(x^k)|}$$

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

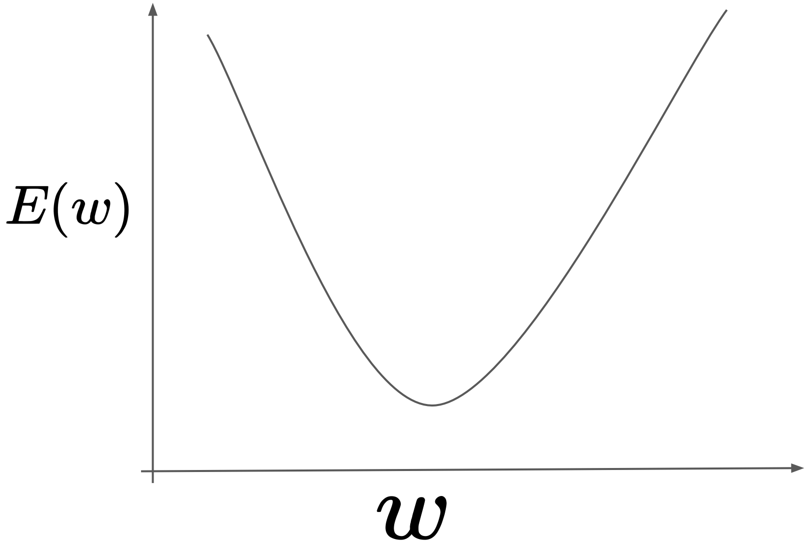


Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}$$

Gradient descent update rule

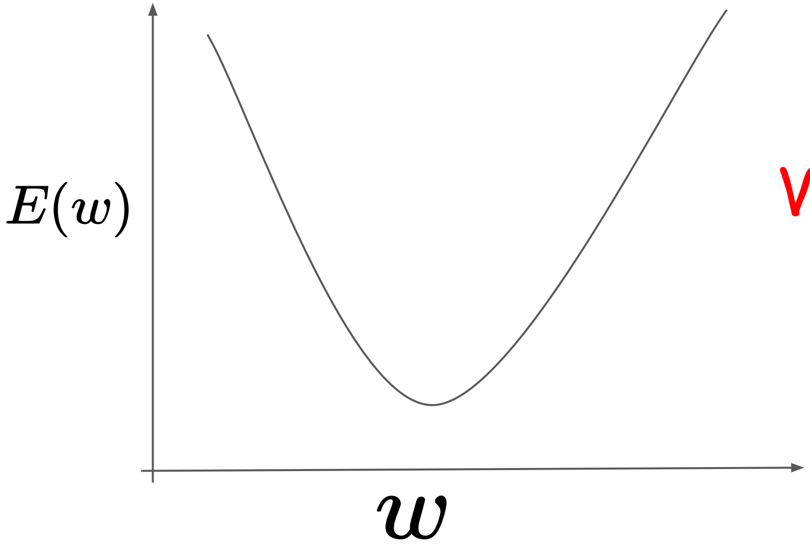


Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \boxed{\eta}\frac{dE(w^t)}{dw} \longrightarrow$$

Gradient descent update rule



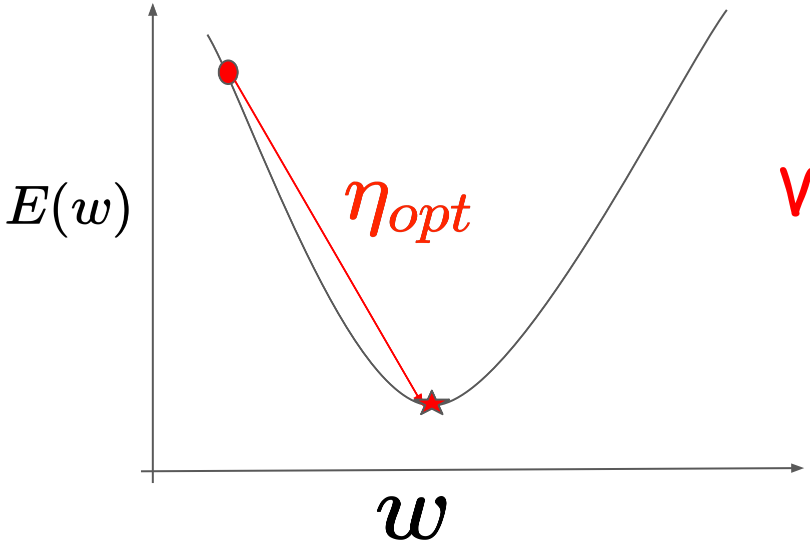
What is the best η?

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \boxed{\eta}\frac{dE(w^t)}{dw}$$

Gradient descent update rule

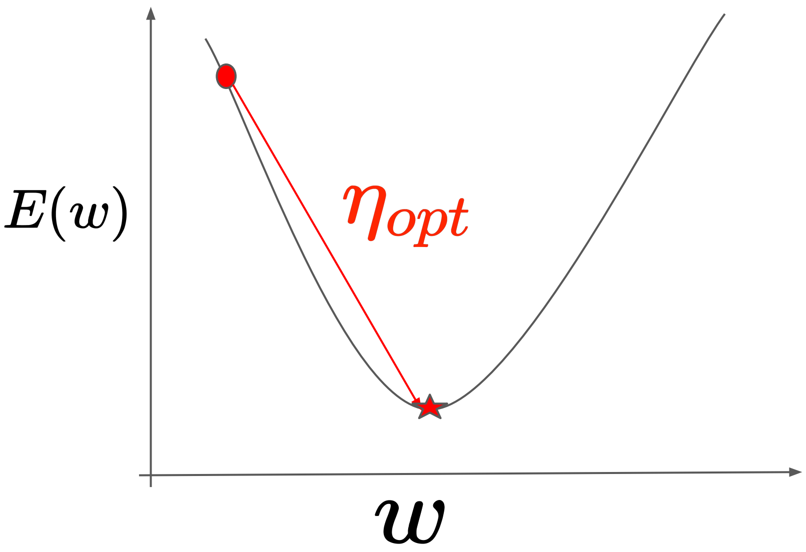


What is the best η?

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}$$



Let us find minima of E(w) using Newton's method.

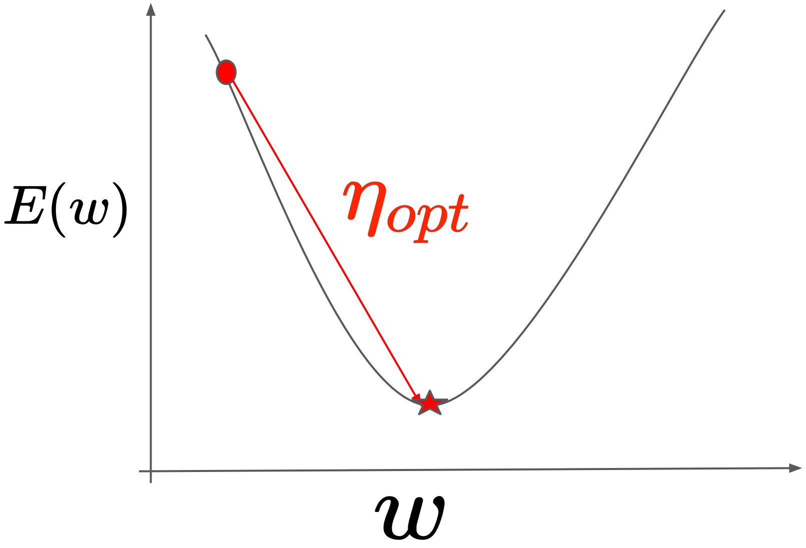
Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}$$

Taylor series

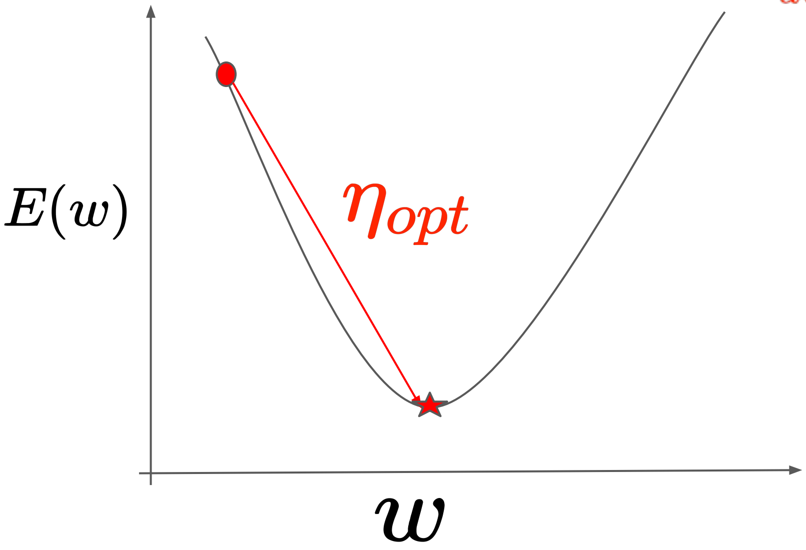
$$E(w) = E(w^t) + (w - w^t)E'(w^t) + \frac{(w-w^t)^2}{2}E''(w^t)$$



Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}$$



Taylor series

$$E(w) = E(w^t) + (w - w^t)E'(w^t) + \frac{(w-w^t)^2}{2}E''(w^t)$$

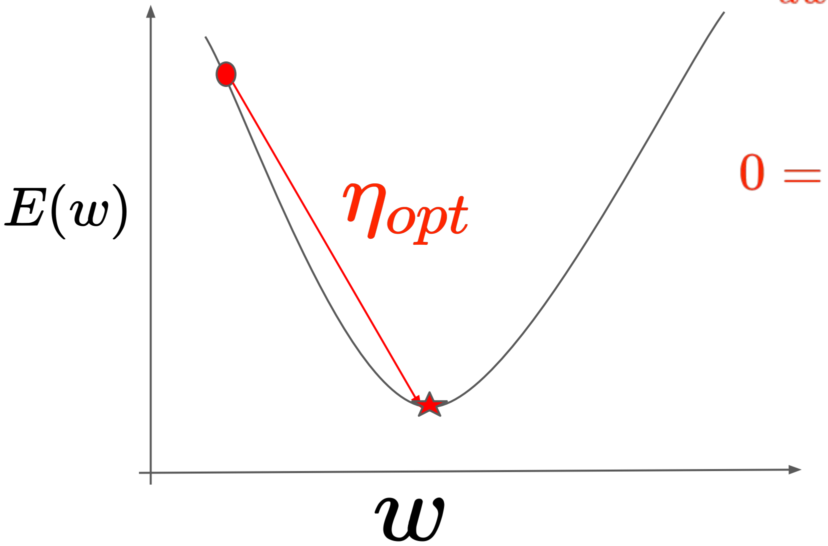
$$\frac{dE(w)}{dw} = E'(w^t) + (w - w^t)E''(w^t) + E'(w^t)$$

$$+ \frac{2(w-w^t)}{2}E''(w^t)$$

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}$$



Taylor series

$$E(w) = E(w^t) + (w - w^t)E'(w^t) + \frac{(w-w^t)^2}{2}E''(w^t)$$

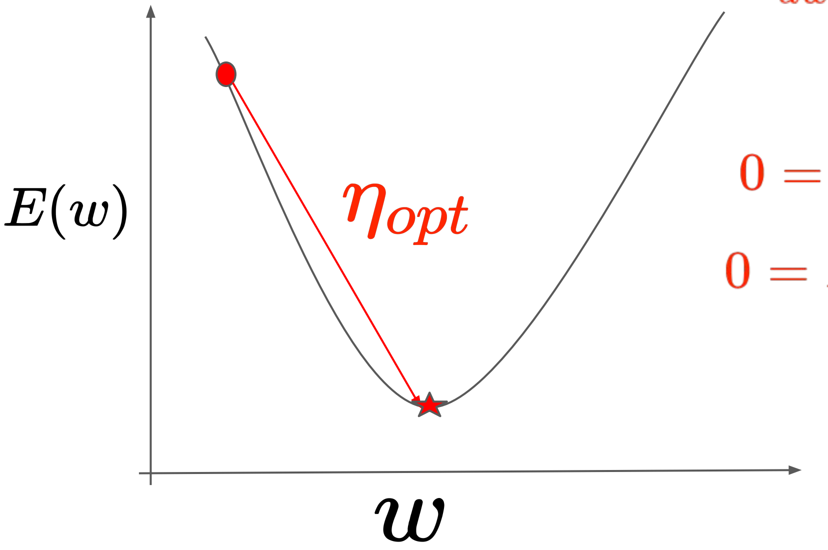
$$\frac{dE(w)}{dw} = E'(w^t) + (w - w^t)E''(w^t) + E'(w^t) + \frac{2(w-w^t)}{2}E''(w^t)$$

$$0 = 2E'(w^t) + 2(w - w^t)E''(w^t)$$

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}$$



Taylor series

$$E(w) = E(w^t) + (w - w^t)E'(w^t) + \frac{(w-w^t)^2}{2}E''(w^t)$$

$$\frac{dE(w)}{dw} = E'(w^t) + (w - w^t)E''(w^t) + E'(w^t)$$

$$+ \frac{2(w-w^t)}{2}E''(w^t)$$

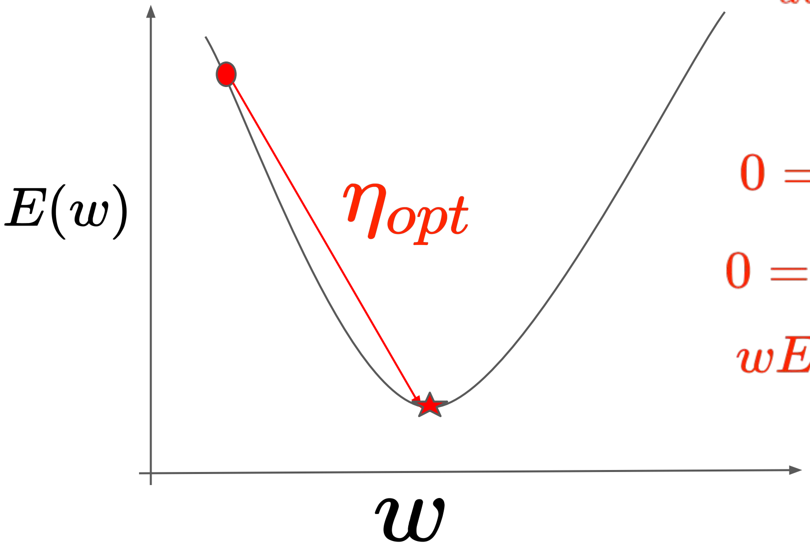
$$0 = 2E'(w^t) + 2(w - w^t)E''(w^t)$$

$$0 = E'(w^t) + wE''(w^t) - w^t E''(w^t)$$

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}$$



Taylor series

$$E(w) = E(w^t) + (w - w^t)E'(w^t) + \frac{(w-w^t)^2}{2}E''(w^t)$$

$$\frac{dE(w)}{dw} = E'(w^t) + (w - w^t)E''(w^t) + E'(w^t)$$

$$+ \frac{2(w-w^t)}{2}E''(w^t)$$

$$0 = 2E'(w^t) + 2(w - w^t)E''(w^t)$$

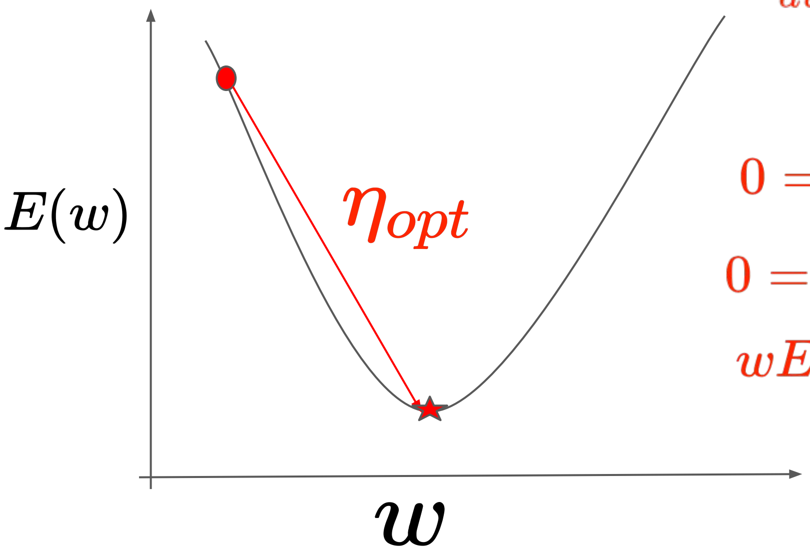
$$0 = E'(w^t) + wE''(w^t) - w^t E''(w^t)$$

$$wE''(w^t) = w^t E''(w^t) - E'(w^t)$$

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}$$



Taylor series

$$E(w) = E(w^t) + (w - w^t)E'(w^t) + \frac{(w-w^t)^2}{2}E''(w^t)$$

$$\frac{dE(w)}{dw} = E'(w^t) + (w - w^t)E''(w^t) + E'(w^t)$$

$$+ \frac{2(w-w^t)}{2}E''(w^t)$$

$$0 = 2E'(w^t) + 2(w - w^t)E''(w^t)$$

$$0 = E'(w^t) + wE''(w^t) - w^t E''(w^t)$$

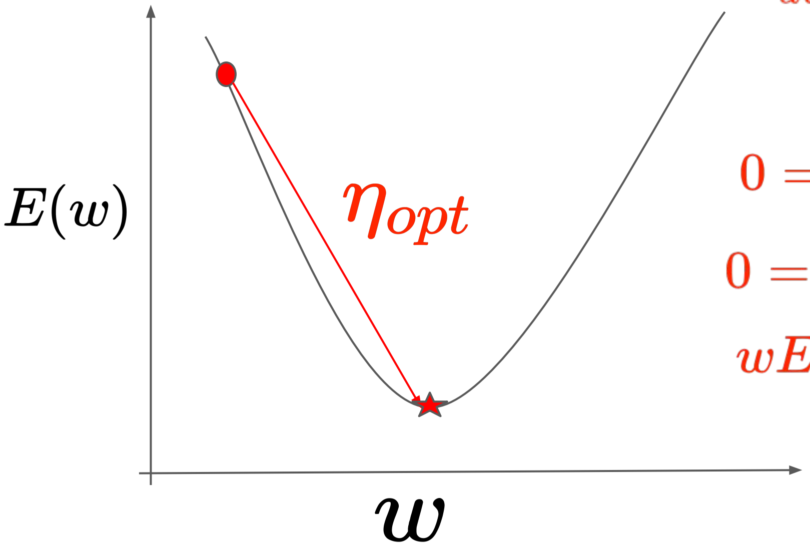
$$wE''(w^t) = w^t E''(w^t) - E'(w^t)$$

$$w = w^t - \frac{E'(w^t)}{E''(w^t)}$$

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$



Taylor series

$$E(w) = E(w^t) + (w - w^t)E'(w^t) + \frac{(w-w^t)^2}{2}E''(w^t)$$

$$\frac{dE(w)}{dw} = E'(w^t) + (w - w^t)E''(w^t) + E'(w^t)$$

$$+ \frac{2(w-w^t)}{2}E''(w^t)$$

$$0 = 2E'(w^t) + 2(w - w^t)E''(w^t)$$

$$0 = E'(w^t) + wE''(w^t) - w^t E''(w^t)$$

$$wE''(w^t) = w^t E''(w^t) - E'(w^t)$$

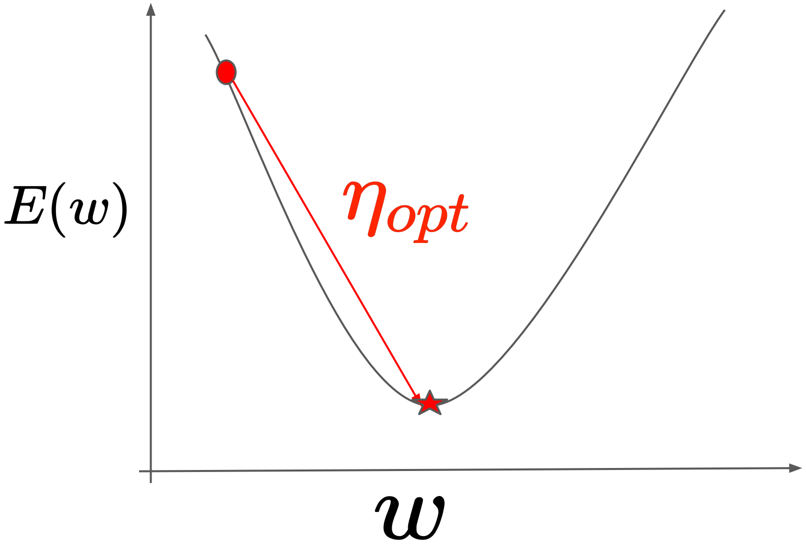
$$\boxed{w = w^t - \frac{E'(w^t)}{E''(w^t)}}$$

Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$

$$\boxed{w = w^t - \frac{E'(w^t)}{E''(w^t)}}$$



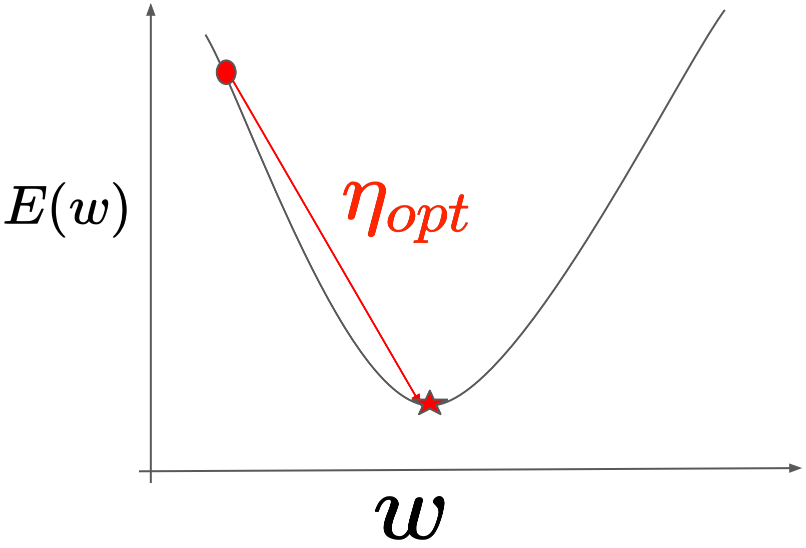
Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$

$$\boxed{w = w^t - \frac{E'(w^t)}{E''(w^t)}}$$

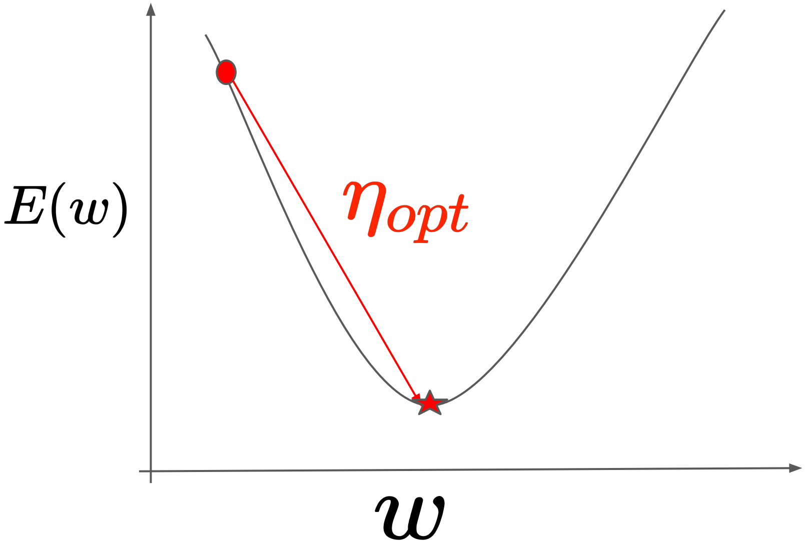
$$\eta_{opt} = \frac{1}{E''(w^t)}$$



Convergence for quadratic surface

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$



$$\boxed{w = w^t - \frac{E'(w^t)}{E''(w^t)}}$$

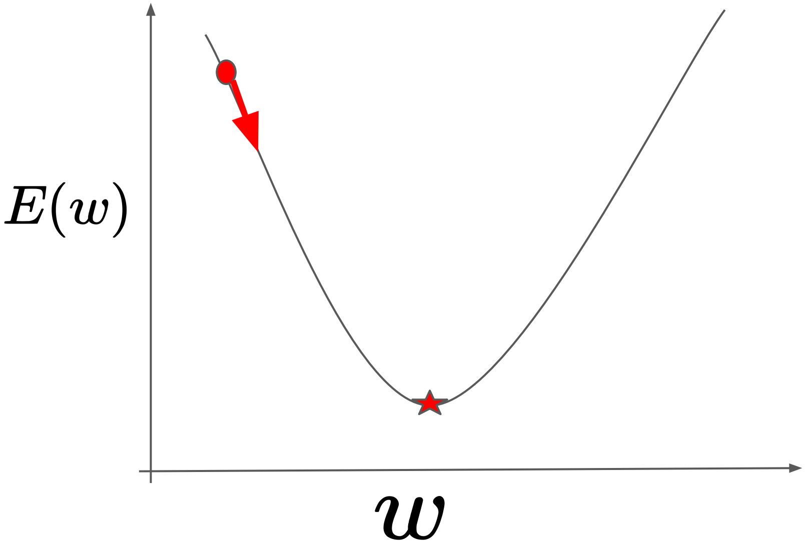
$$\eta_{opt} = \frac{1}{E''(w^t)}$$

$$\eta_{opt} = \frac{1}{a}$$

Case 1: $\eta < \eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

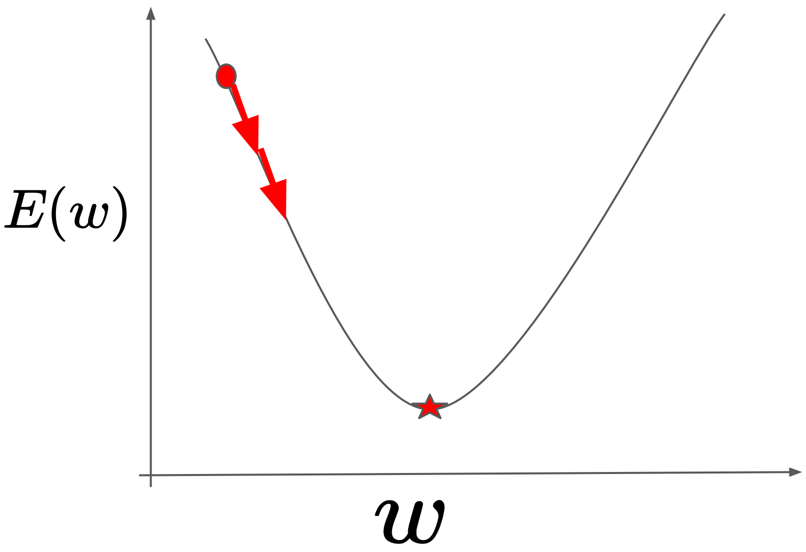
$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$



Case 1: $\eta < \eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

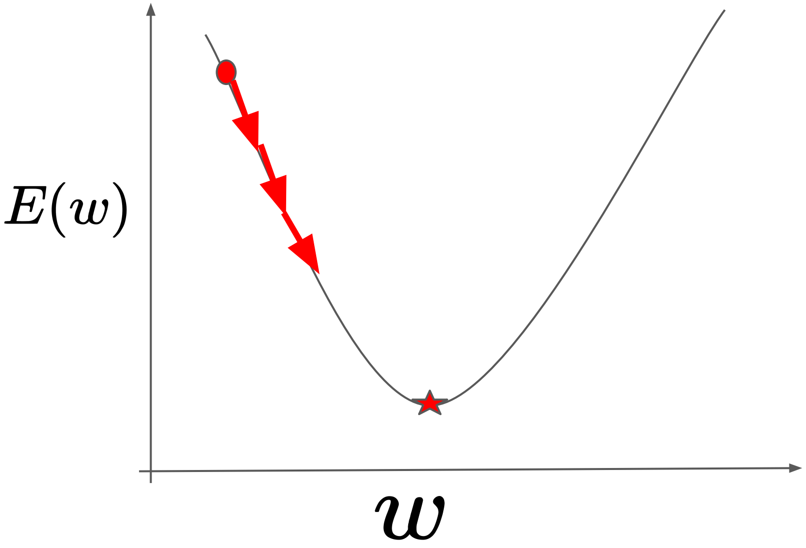
$$\boxed{w^{t+1} = w^t - \eta\frac{dE(w^t)}{dw}}$$



Case 1: $\eta < \eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

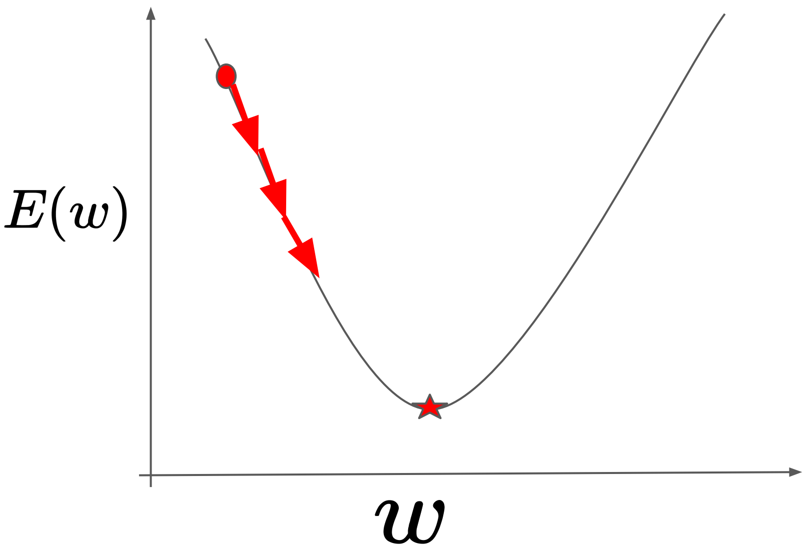
$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$



Case 1: $\eta < \eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

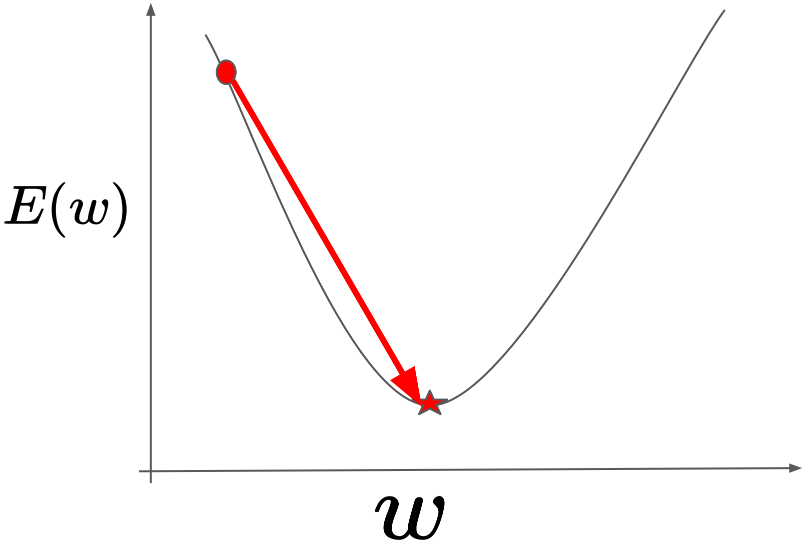
$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$



Case 2: $\eta = \eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

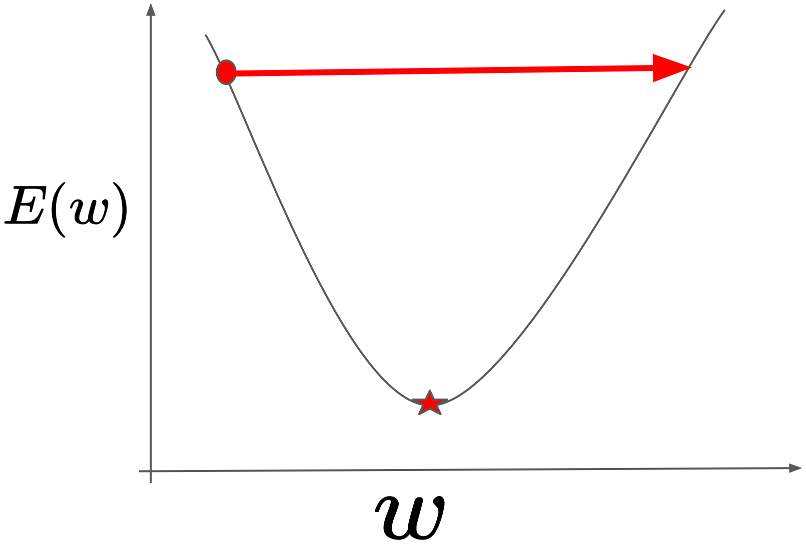
$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$



Case 3: $\eta = 2\eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

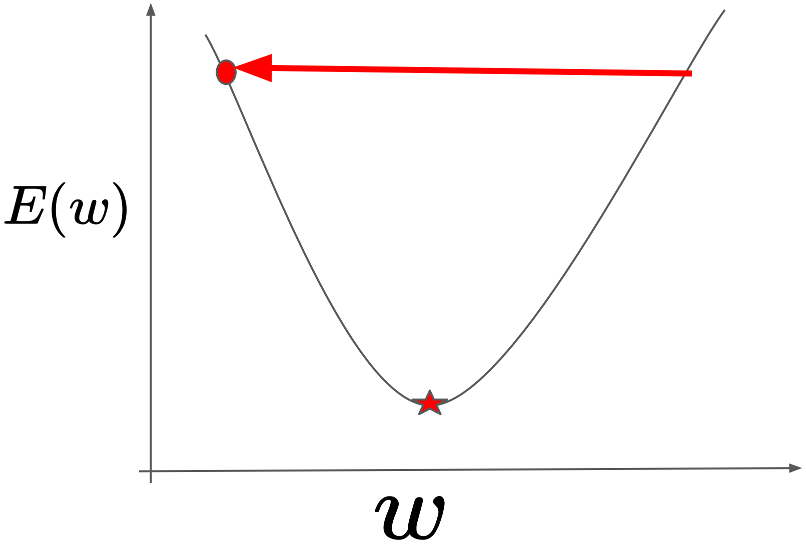
$$\boxed{w^{t+1} = w^t - \eta\frac{dE(w^t)}{dw}}$$



Case 3: $\eta = 2\eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

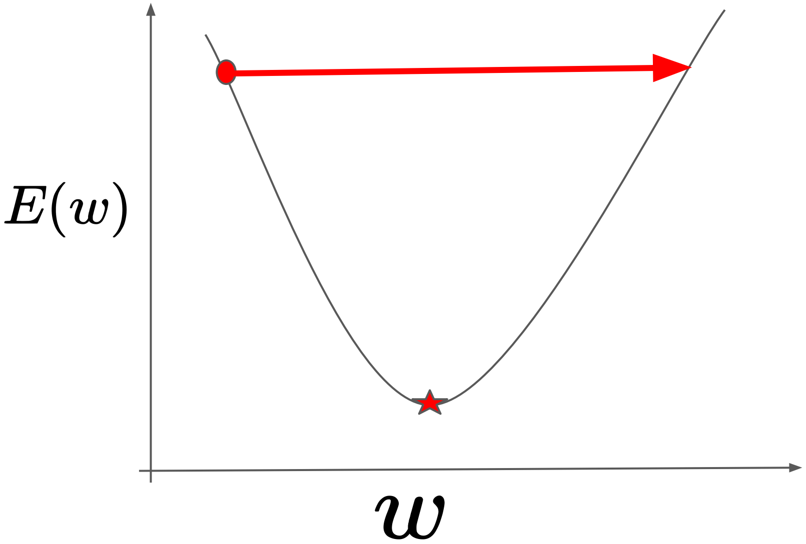
$$\boxed{w^{t+1} = w^t - \eta \frac{dE(w^t)}{dw}}$$



Case 3: $\eta = 2\eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

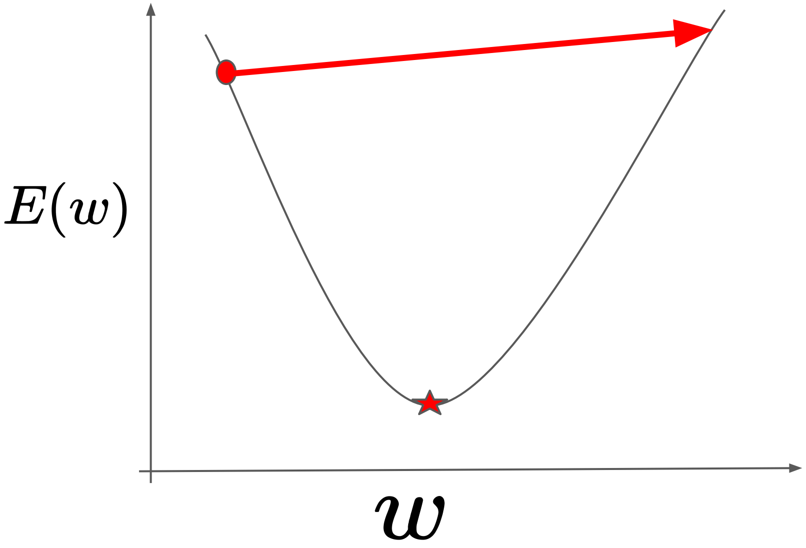
$$\boxed{w^{t+1} = w^t - \eta\frac{dE(w^t)}{dw}}$$



Case 4: $\eta > 2\eta_{opt}$

$$E(w) = \frac{1}{2}aw^2 + bw + c$$

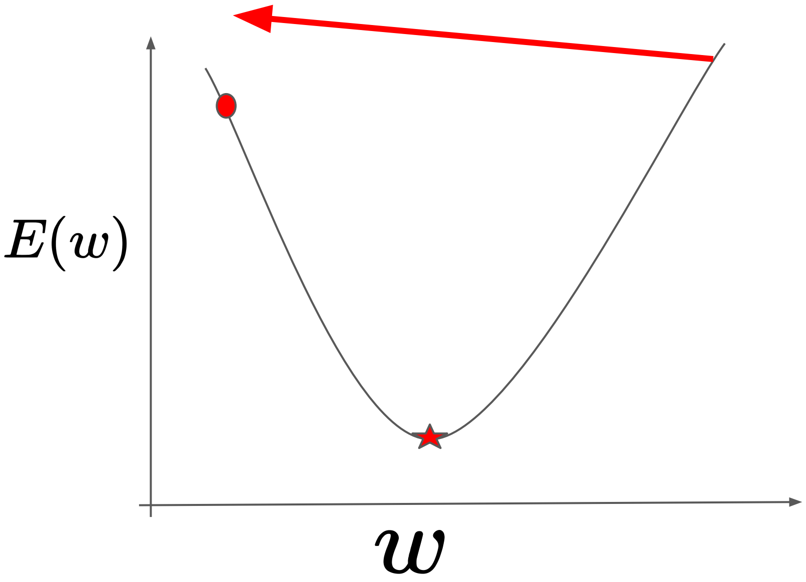
$$\boxed{w^{t+1} = w^t - \eta\frac{dE(w^t)}{dw}}$$



Case 4: $\eta > 2\eta_{opt}$

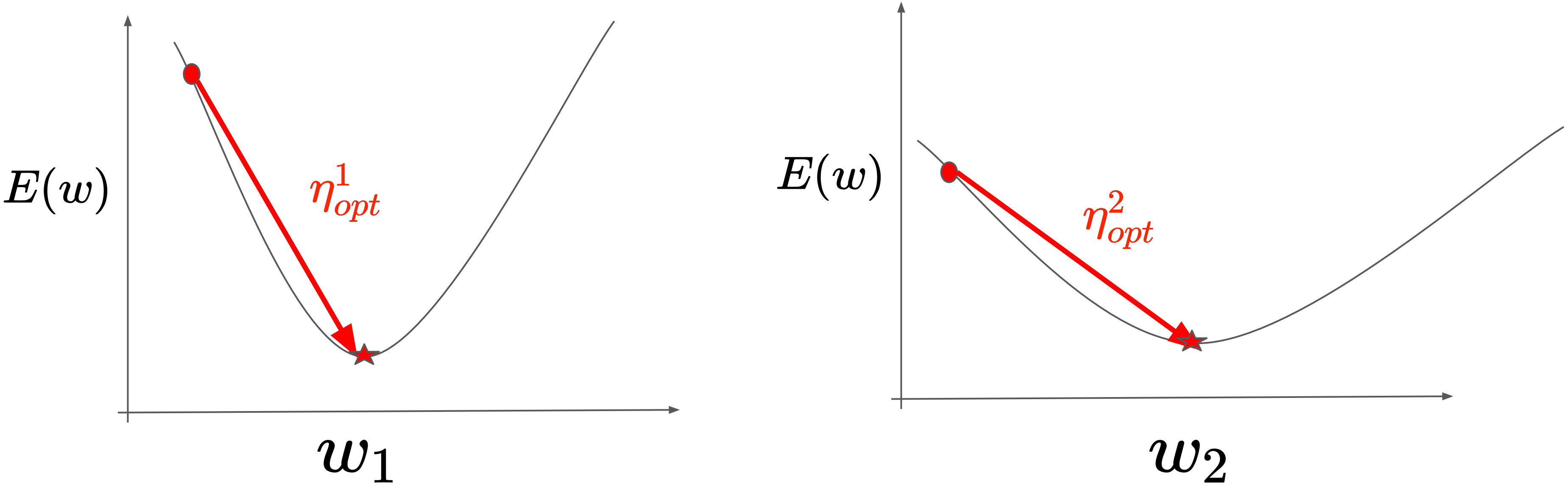
$$E(w) = \frac{1}{2}aw^2 + bw + c$$

$$\boxed{w^{t+1} = w^t - \eta\frac{dE(w^t)}{dw}}$$

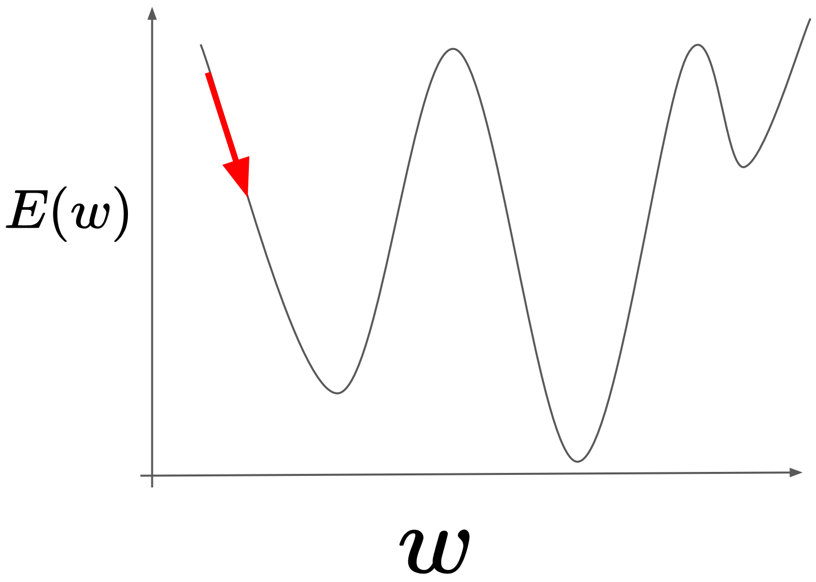


So far we have analyzed only single variable and convex functions

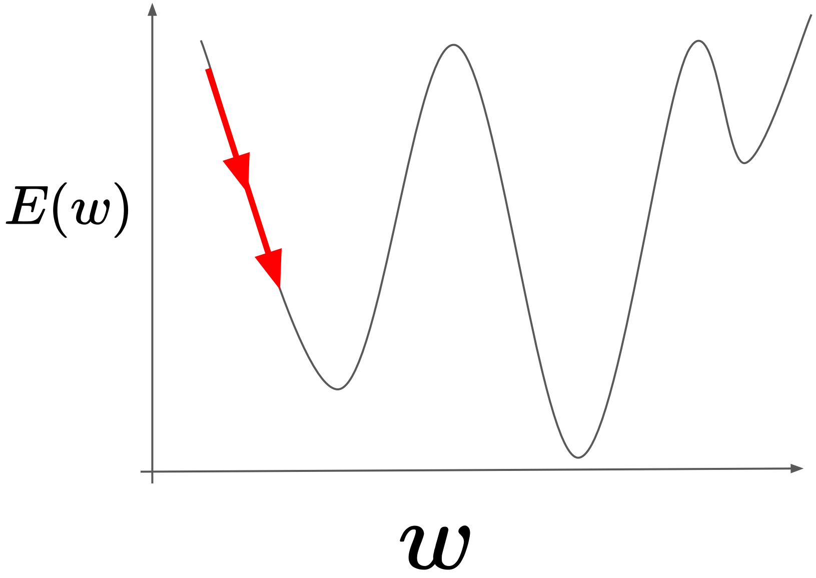
Problem 1: Multi-variable cost function



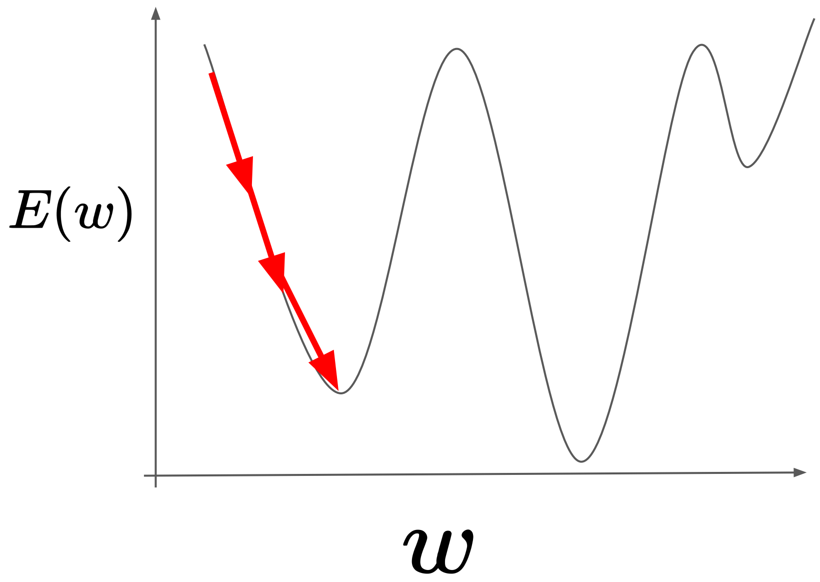
Problem 2: Non-convex cost function



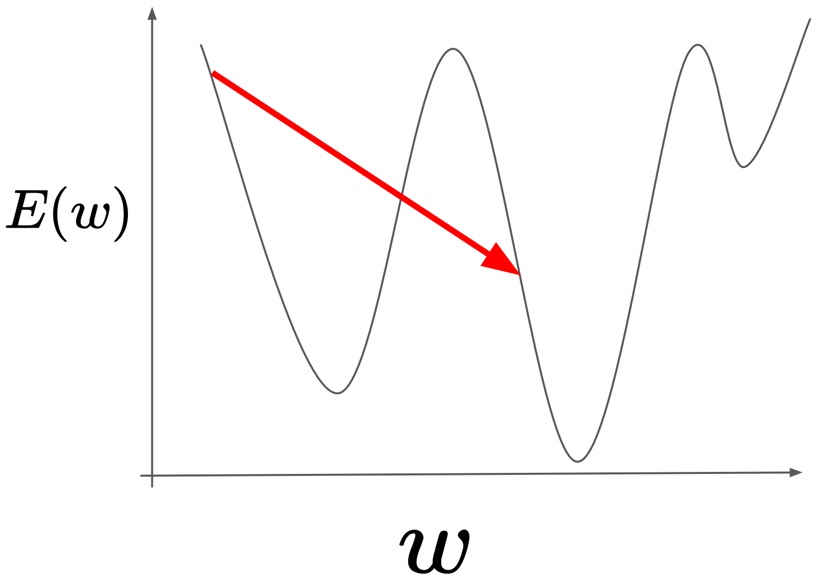
Problem 2: Non-convex cost function



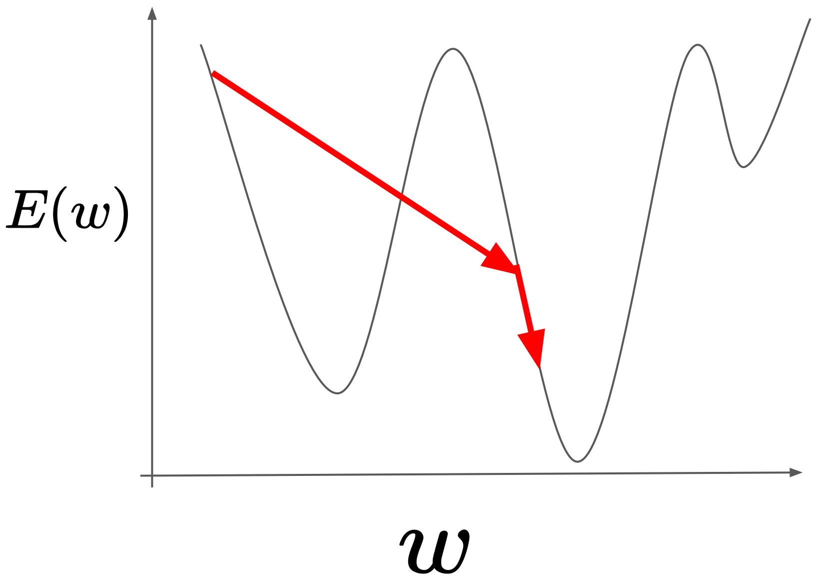
Problem 2: Non-convex cost function



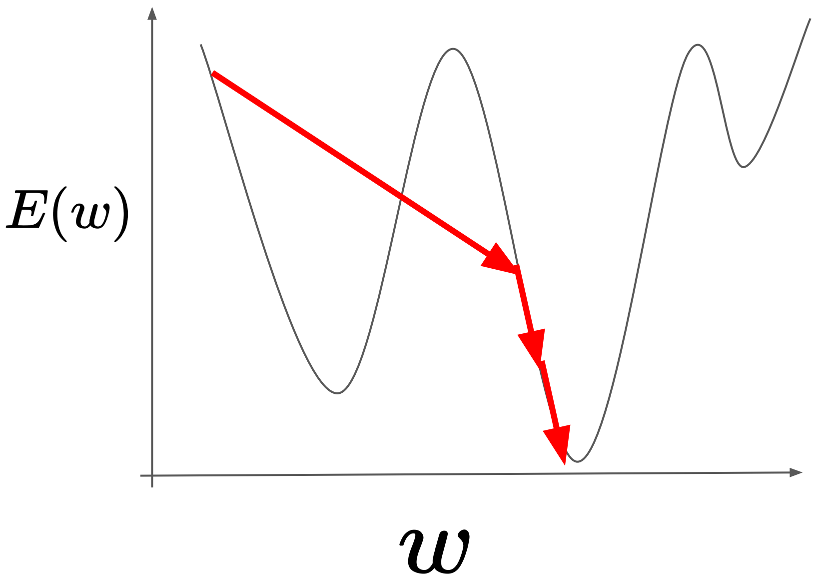
Problem 2: Non-convex cost function



Problem 2: Non-convex cost function



Problem 2: Non-convex cost function



Decaying Learning rate

- Linear decay

$$\eta_t = \frac{\eta_0}{t+1}$$

- Quadratic decay

$$\eta_t = \frac{\eta_0}{(t+1)^2}$$

- Exponential decay

$$\eta_t = \eta_0 e^{-\beta t}, \beta > 0$$

Module 2: variants of GD

Batch Gradient Descent

- Uses the whole batch of training data at every step.
- Calculates the error for each record and takes an average to determine the gradient.

Advantage: the algorithm is more computational efficient and it produces a stable learning path, so it is easier to convergence.

Disadvantage: The entire training set can be too large to process in the memory

Stochastic Gradient Descent (SGD)

- Uses the single training data at every step.
- Calculates the error for each record and update the weight for every record

Advantage: fits into memory.

Disadvantage: Computationally expensive

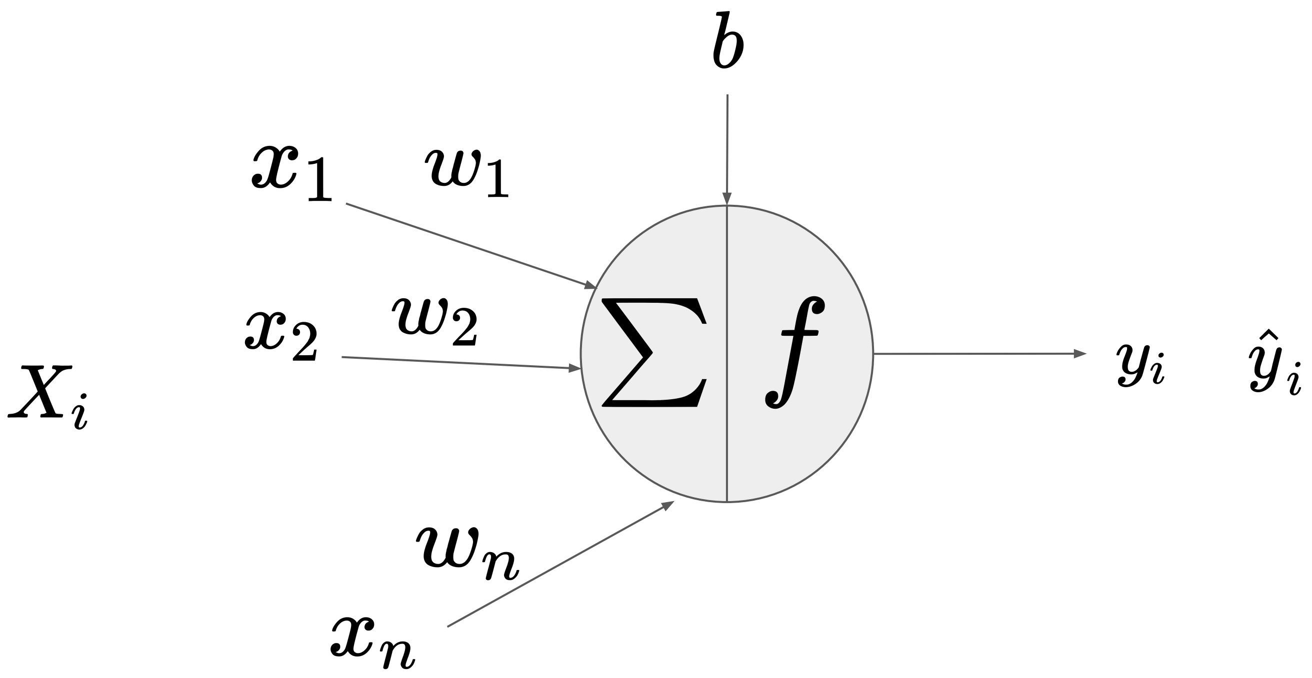
Mini Batch Gradient Descent

- Combines batch GD and SGD
- The training set is divided into multiple groups called batches.
- At a time a single batch is passed through the network which computes the loss of every sample in the batch and uses their average to update the parameters of the neural network.

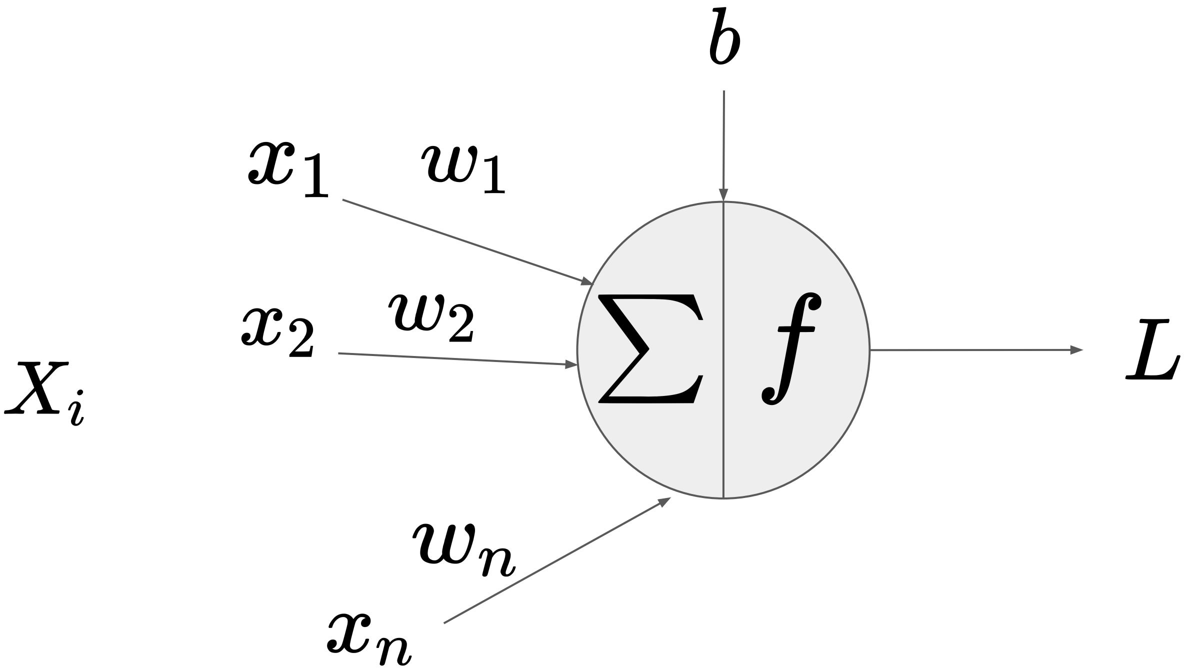
Advantages: fits in memory, computationally attractive.

Why is data coming into picture in GD

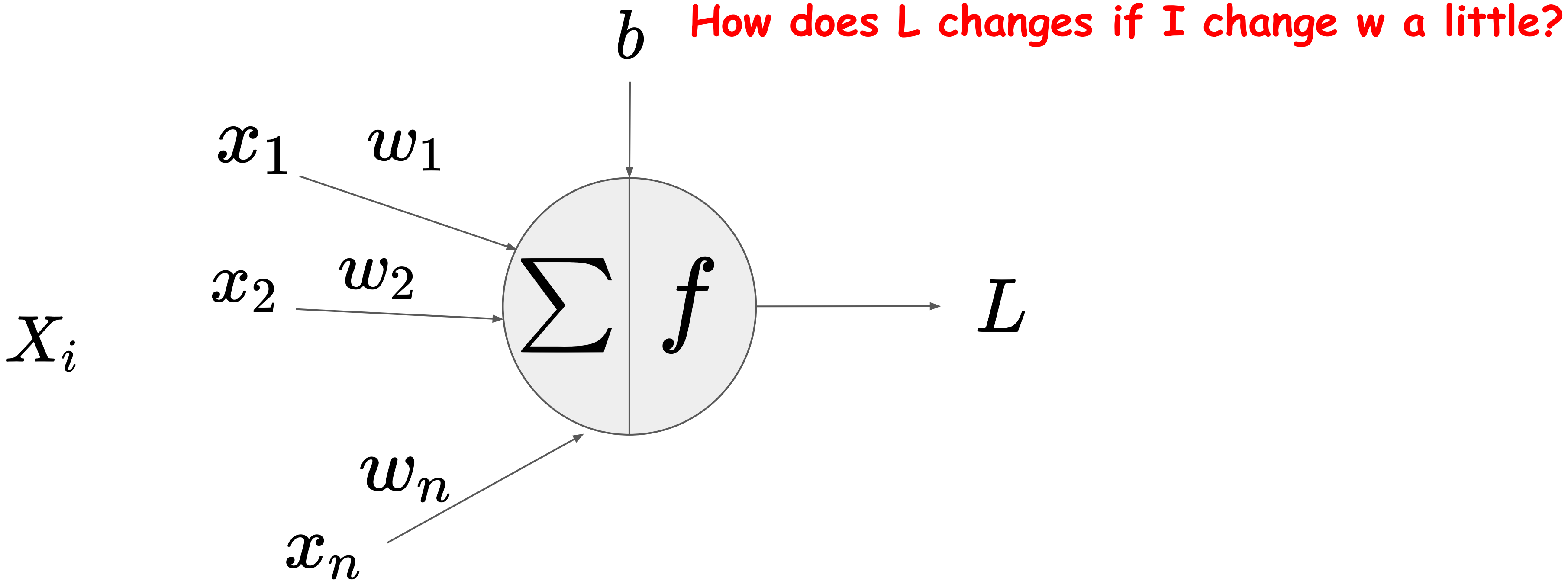
Let us work on a simplest example, One Neuron!



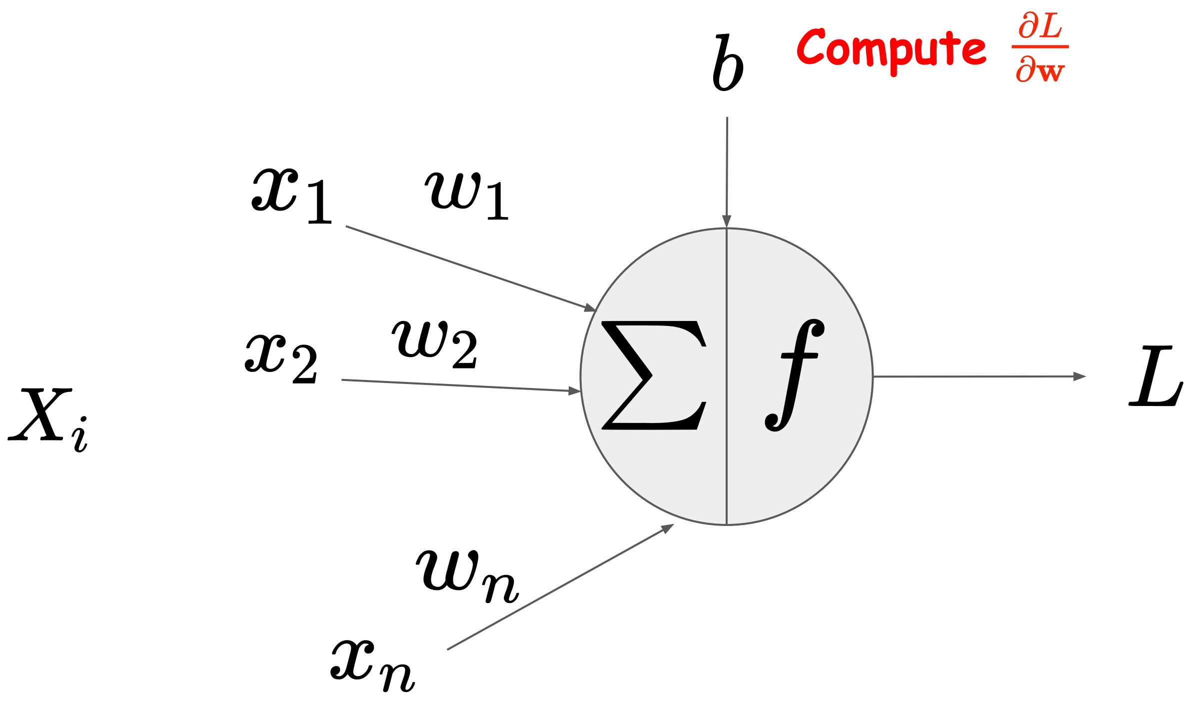
Let us work on a simplest example, One Neuron!



Let us work on a simplest example, One Neuron!

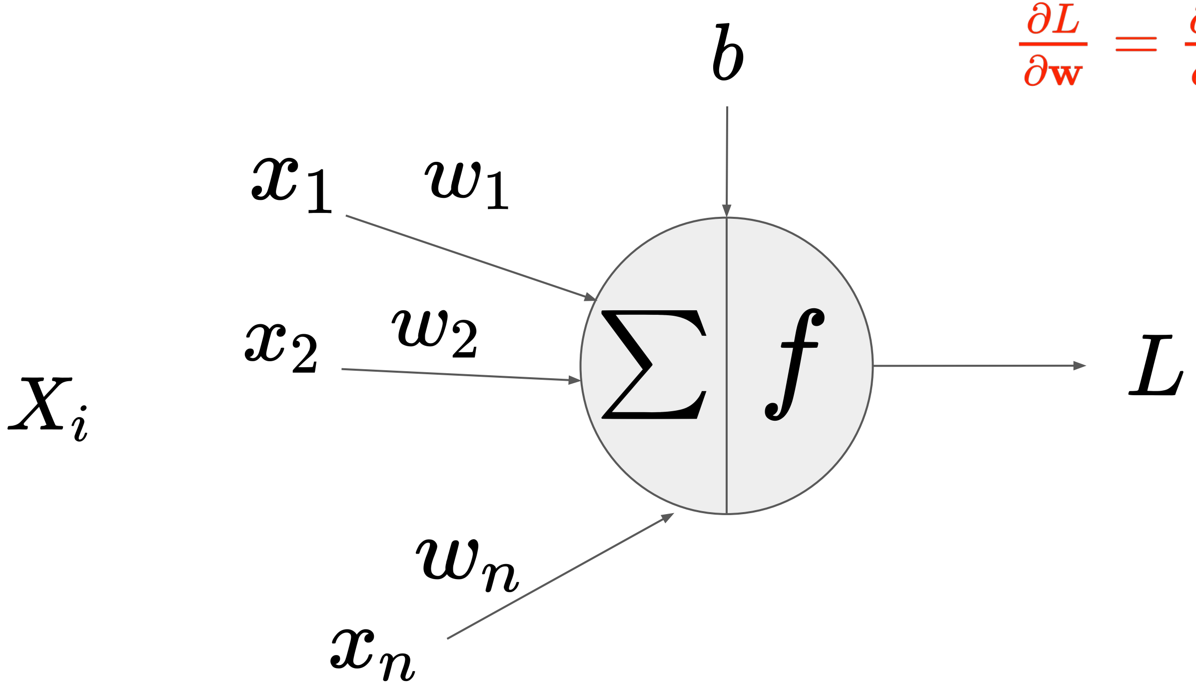


Let us work on a simplest example, One Neuron!



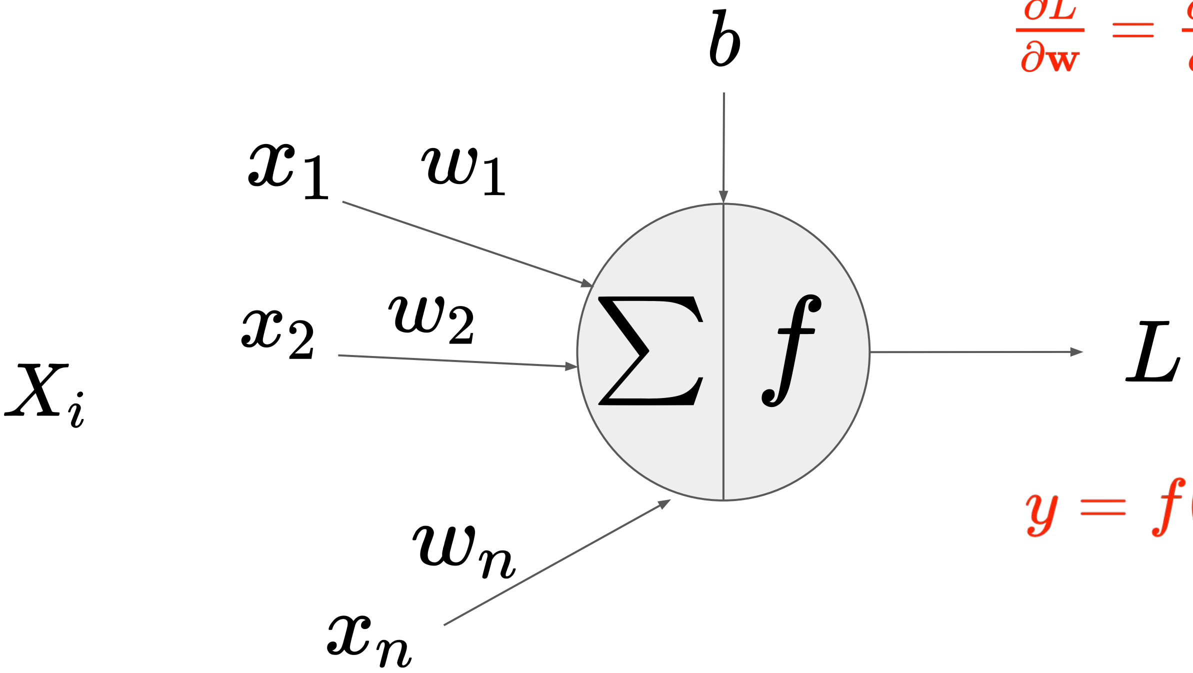
Let us work on a simplest example, One Neuron!

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial y}{\partial \mathbf{w}}$$



Let us work on a simplest example, One Neuron!

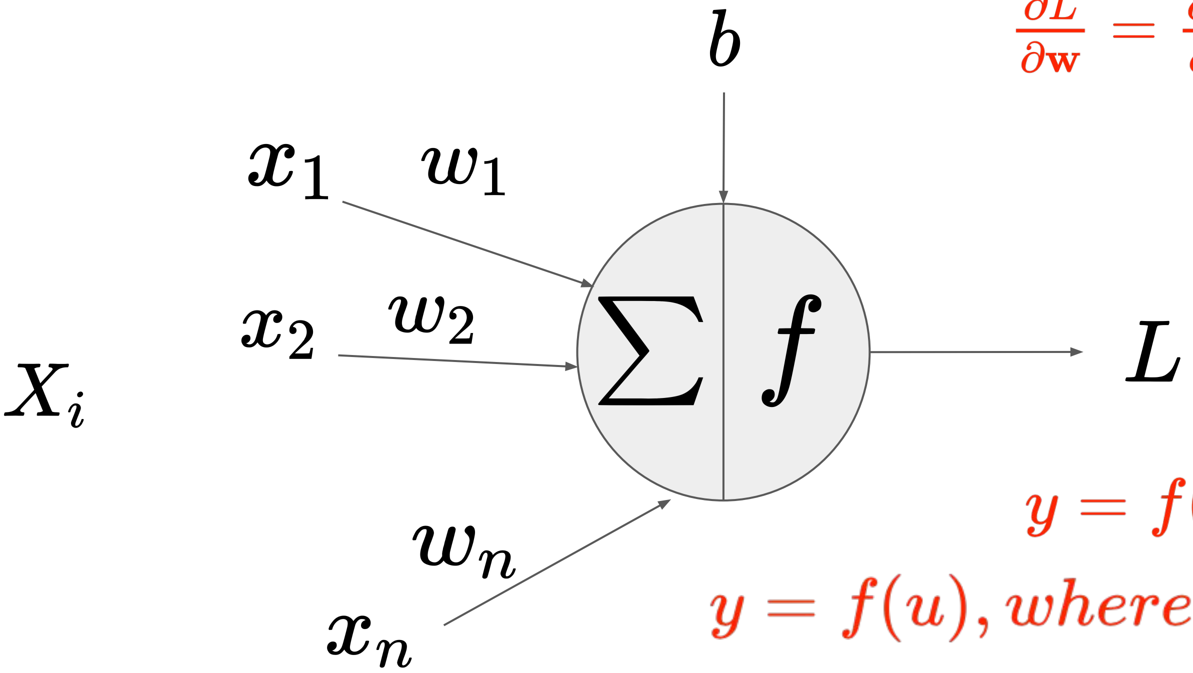
$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial y}{\partial \mathbf{w}}$$



$$y = f(\textstyle\sum_{i=1}^{n} w_i x_i + b)$$

Let us work on a simplest example, One Neuron!

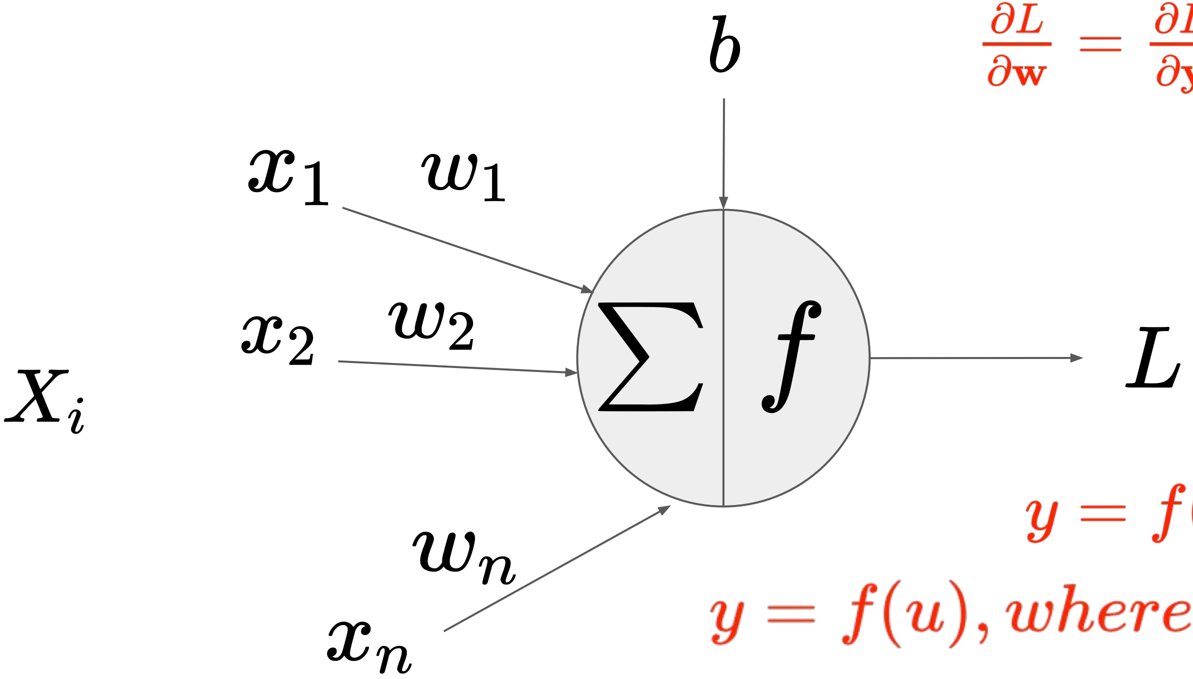
$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{w}}$$



$$y = f(\textstyle\sum_{i=1}^{n} w_i x_i + b)$$

$$y = f(u), where\ u = \textstyle\sum_{i=1}^{n} w_i x_i + b$$

Let us work on a simplest example, One Neuron!



$$\frac{\partial L}{\partial \mathbf{w}} = \frac{\partial L}{\partial \mathbf{y}} \times \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \times \frac{\partial \mathbf{u}}{\partial \mathbf{w}}$$

$$y = f(\sum_{i=1}^{n} w_i x_i + b)$$

$$y = f(u), where\ u = \sum_{i=1}^{n} w_i x_i + b$$

Summary

- Variants of GD
- How we compute derivative of loss wrt weights for single neuron
- How a single neuron is trained using data

More reading:
https://ruder.io/optimizing-gradient-descent/