Artificial Intelligence/Machine Learning/Deep Learning: 'Bridging the Skills Gap'

Lesson 2: Gradient Descent

Hello, welcome back! This lesson is dedicated to the most widely used optimization technique: Gradient Descent!

1. Gradient Descent, Multivariate Gradient Descent, Stochastic Gradient Descent, Mini Batch Stochastic Gradient Descent,
2. Optimizers: Momentum, RMSprop, Adam, Adagrad

Prerequisite for this session:

Differentiation, Gradient (∇), Taylor Series, Vector. These items are explained in the math refresher on Calculus and Linear Algebra.

Last session we looked at a regression problem, trying to predict the house price given de size of the house. We introduced the concept of a Cost Function C(w,b) and every time that ($y \neq y_p$) the model incurs 'cost'.

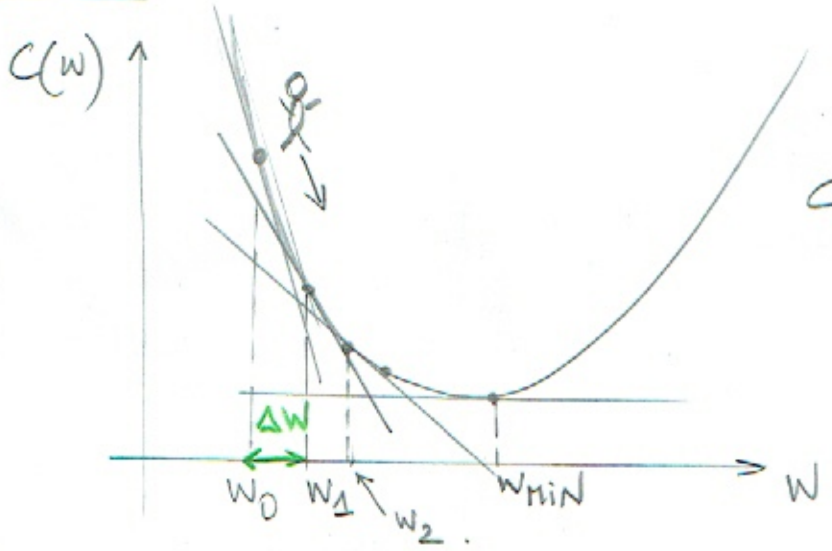
Our goal is to 'learn' the model parameters w and b as to minimize that cost for a test point (x_{new}) $\rightarrow$ **minimization problem** $\rightarrow$ **differentiation problem** $\rightarrow$ **Gradient Descent**

The cost function used for our single variate regression problem was MSE

C(w,b) = $\frac{1}{6}\sum_{i=1}^{6}(y_p^i - y^i)$ with y_p the predicted housing price and y the actual housing price

C(w,b) is quadratic function $\rightarrow$ always positive and penalizes outliers

LESSON 2



MSE = quadratic function

$$C(w,b) = \frac{1}{m}\sum_{i=1}^{m}\left(y_p^{(i)} - y^{(i)}\right)^2$$

START ① pick w_0 randomly.

② calculate $\frac{dc}{dW}(w_0) \rightarrow$ sign $\rightarrow -$

move downwards with step $\alpha \rightarrow w_1$

calculate $\frac{dc}{dW}(w_1) \rightarrow$ sign $\rightarrow -$

move downwards with step $\alpha \rightarrow w_2$

$\xrightarrow{\text{Repeat}}$ till $\frac{dc}{dW} \approx 0 \rightarrow$ MINIMUM

↳ or change in cost $< 10^{-3}$

$$\boxed{w_{i+1} = w_i - \alpha \frac{dc}{dW}(w_i)}$$

LEARNING Rate = hyperparameter!

chose AS A DATA Scientist

→ ANDREW Ng

0.001, 0.003, 0.01, ...

$\times 3$ $\times 3$

$i=0 \rightarrow w_1 = w_0 - \alpha \frac{dc}{dW}(w_0)$

if $\frac{dc}{dW}(w_0) < 0$

$$\underbrace{-\alpha \frac{dc}{dW}(w_0) > 0}$$

$w_1 = w_0 +$ something. → move to the Right on W-AXIS

α too small → slow convergence

α too big → Risk of oscillations → NO convergence

So far $\rightarrow$ 1 feature (size of the house)

Multivariate Regression

m samples

Size	# Rooms	Age	Price

$\rightarrow$ d features $(d=3)$

set $x_0 = 1$.

$$\underset{\text{1 feature}}{y = wx + b} \longrightarrow \underset{\text{d features}}{y = w_0 x_0 + w_1 x_1 + \dots + w_d x_d} \quad (1)$$

$$X = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \qquad W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

feature vector

weight vector

$$W^T X = (w_0 \; w_1 \cdots w_d) \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} = (1)$$

$\Downarrow$

i-th sample

$$\boxed{MSE = C(W) = \frac{1}{m} \sum_{i=1}^{m} \left(W^T X^{(i)} - y^{(i)} \right)^2}$$

<u>now</u>: $C(W_0 + \Delta W) \approx C(W_0) + \underbrace{\Delta W \cdot \nabla C(W_0)}_{\substack{\text{dot product} \\ = \text{scalar}}} + \cancel{\dots}$

TAYLOR

if ΔW small enough $\rightarrow$ linear!

and minimum if ΔW and $\nabla C(W_0)$ are opposite directions

$\rightarrow \Delta W \sim -\nabla C(W_0) \rightarrow \frac{\Delta W}{\nabla C(W_0)} = -\alpha$

$\xrightarrow{+W_0} W_0 + \Delta W = W_0 - \alpha \nabla C(W_0)$

$$\boxed{w_{i+1} = w_i - \alpha \nabla C(w_i)}$$

Gradient Descent for multivariate regression!

example: 2 features → 2 weights

$x_1 \; x_2$ $\qquad$ $w_1 \; w_2$.

$$C(w) = w_0 + w_1 x_1 + w_2 x_2$$

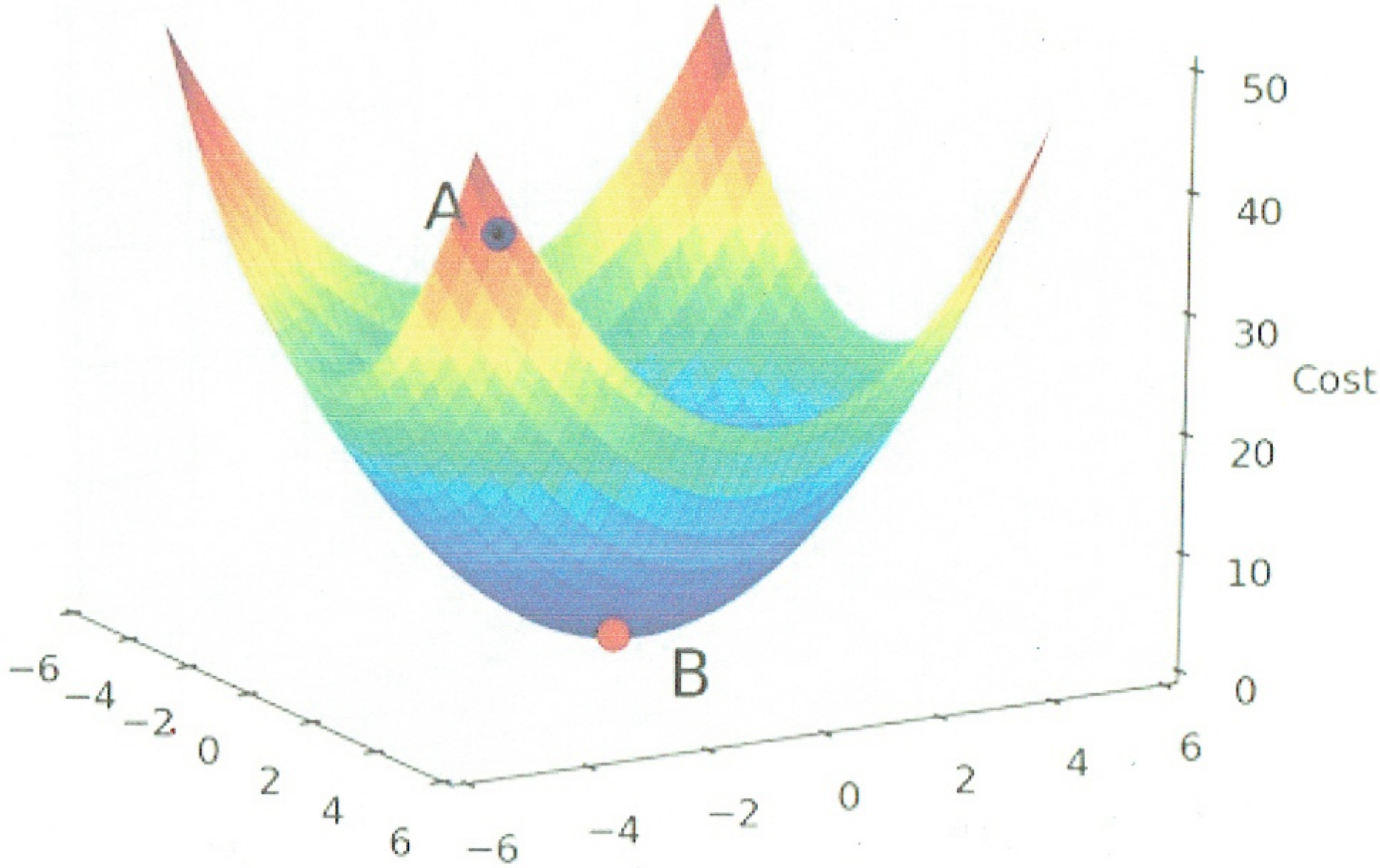
↓

WAS BIAS b

↳ $w_0 x_0 \ldots$

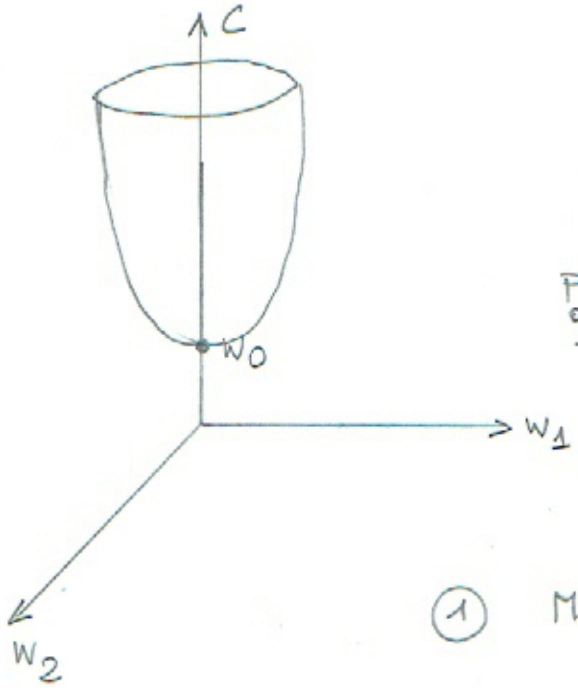
↓

$$x_0 \overset{\text{set}}{=} 1.$$



4

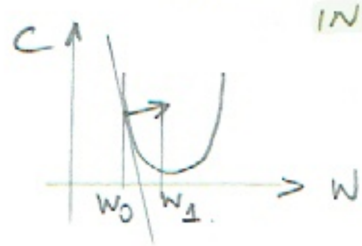
PROJECTION ON W_1, W_2 plane → CONTOUR Plots



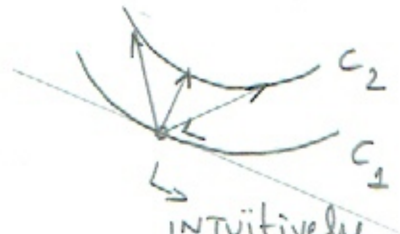
projection ON W_1, W_2 plane → CONTOUR Plots showing points with equal cost

see p 6

1. MINIMIZATION → gradient points INWARD



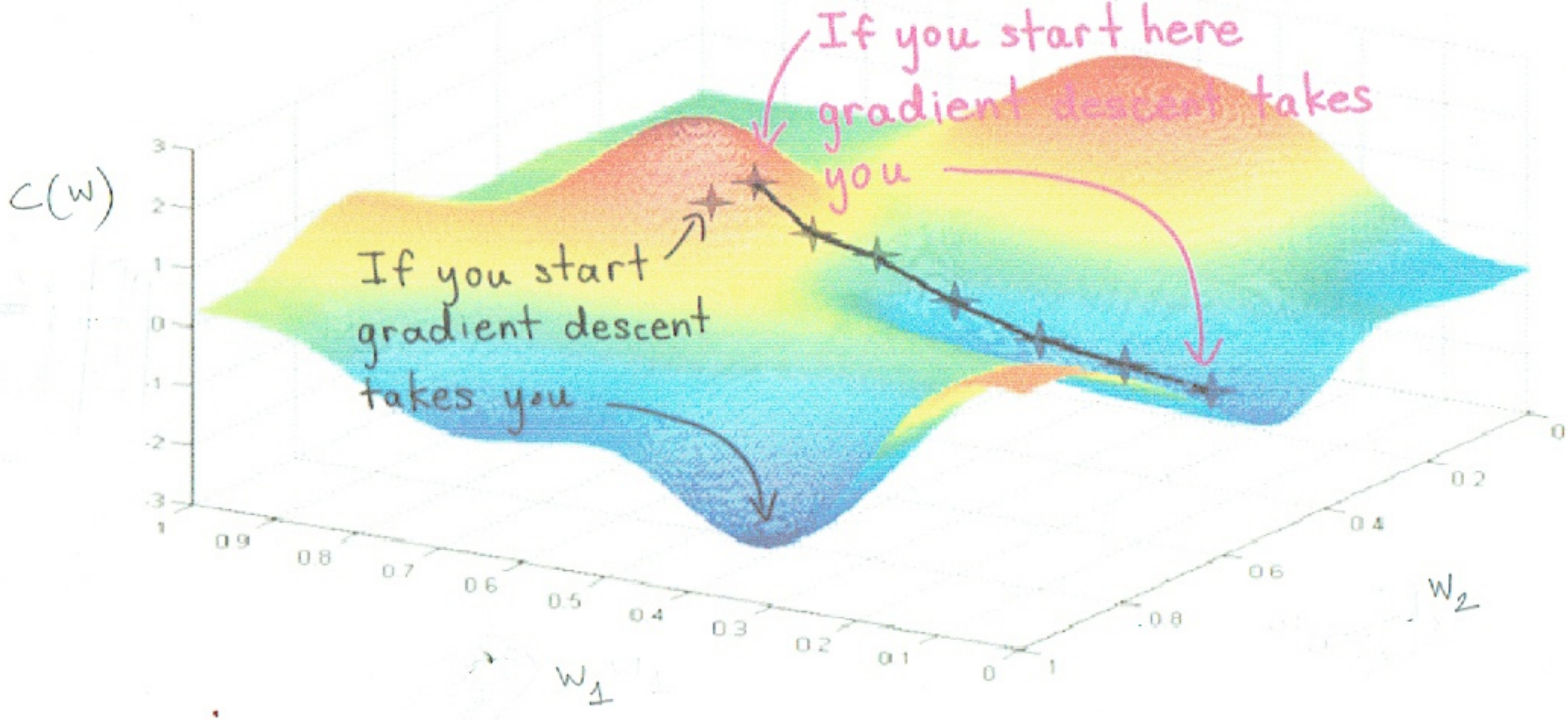
2. Gradient is ⊥ ON CONTOUR lines And is IN the direction of steepest descent

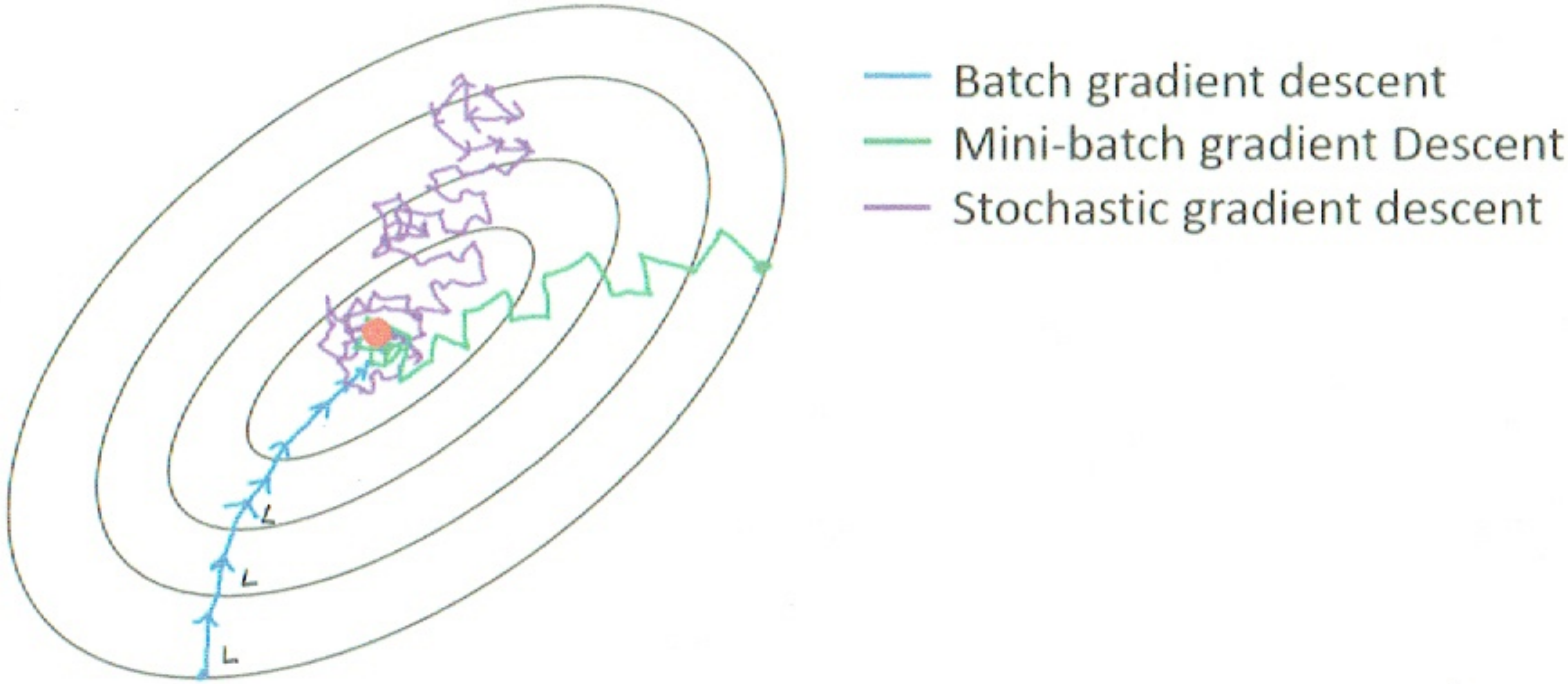


INTUITIVELY why gradient points INTO Direction of Steepest Descent

Convexity : Not ALL cost functions are convex
ex: Neural Networks

→ you can get stuk IN local MINIMUM


C(w)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
0.8
0.6
0.4
0.2
0
w_1
w_2
If you start
gradient descent
takes you
If you start here
gradient descent takes
you


Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent

How to improve GD ?

↳ 1) converge FASTER
2) Avoid getting stuck in local minima
3)

you want α small.

you want α 'bigger'

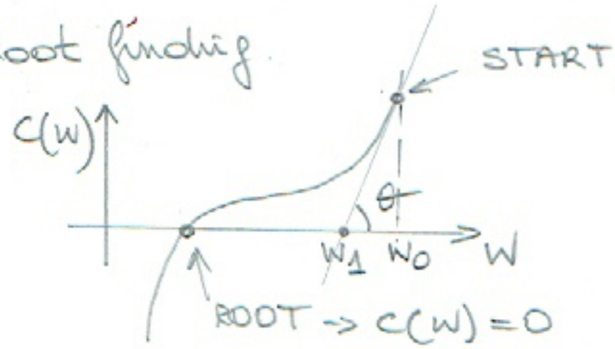
↓

ideally you have a different learning rate α for every feature!

→ you need info over curvature
→ HESSIAN

Newton's method : 2nd order optimization

↳ ① Root finding



$$\tan\theta = \frac{C(W_0)}{W_0 - W_1} \quad (1)$$

↓

$$\frac{dC}{dW_0} = C'(W_0) \quad (2)$$

$$(1) \text{ and } (2) \rightarrow C'(W_0) = \frac{C(W_0)}{W_0 - W_1}$$

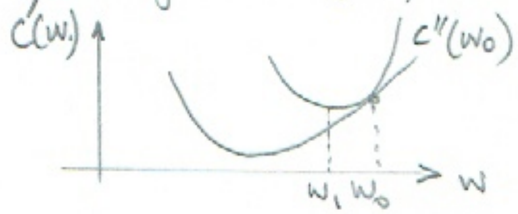
$$\rightarrow (W_0 - W_1)\, C'(W_0) = C(W_0)$$

$$\rightarrow \cancel{W_0 C'(W_0)} - W_1 C'(W_0) = C(W_0) - W_0 C'(W_0)$$

$$\rightarrow W_1 C'(W_0) = W_0 C'(W_0) - C(W_0)$$

$$\rightarrow W_1 = \frac{W_0 C'(W_0)}{C'(W_0)} - \frac{C(W_0)}{C'(W_0)} \rightarrow \boxed{W_1 = W_0 - \frac{C(W_0)}{C'(W_0)}}$$

Analogy → we want to find $C'(W_0) = 0$ → $\boxed{W_1 = W_0 - \frac{C'(W_0)}{C''(W_0)}}$



· NEWTON converges faster!
computationally more expensive → $\mathcal{H}^{-1}$
no hyperparameter α

better when # features is small ($d < 1000$)

STOCHASTIC GRADIENT Descent

Full batch GD → for every little step → massive simult. calculations

$$MSE = c(w) = \frac{1}{m} \sum_{i=1}^{m} (w^T x^{(i)} - y^{(i)})^2$$

↳ $m = 300 \cdot 10^6$ training samples

IDEA: ① shuffle dataset
② execute GD on 1 sample and update!
③ Repeat for next sample

↳ Noisy!
SGD doesnt really converge but ok if we are near minimum

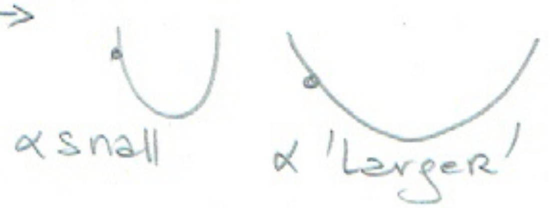
Mini-bath Gradient Descent

↳ GOTO method for large datasets
① split data in mini-batches and process batch per batch!

size batch = hyperparameter!
typically 64, 128, 256, 512, ...

Optimizers

ADAGRAD: Adaptive GRAdient (2011)



↳ dynamically adjust α for each update and for every weight individually

$$\boxed{w_{i+1} = w_i - \frac{\alpha}{\sqrt{v_i + \varepsilon}} \nabla C(w_i)} \quad (1)$$

↳ Avoid division by zero

$$v_i = v_{i-1} + (\nabla C(w_i))^2$$

START: $v_0 = 0$

$$v_1 = v_0 + (\nabla C(w_1))^2 = (\nabla C(w_1))^2$$

$$v_2 = v_1 + (\nabla C(w_2))^2 = (\nabla C(w_1))^2 + (\nabla C(w_2))^2$$

$$\vdots$$

$$v_i = (\Delta C(w_1))^2 + (\nabla C(w_2))^2 + \dots + (\nabla C(w_i))^2 \quad (2)$$

↳ keeps squares of past gradients

$$(1) \ \& \ (2) \implies -\frac{\alpha}{\sqrt{v_i + \varepsilon}}$$

↳ aggressively decays learning rate α

RMSprop:

↳ Avoid rapid growth of denominator

$$\boxed{w_{i+1} = w_i - \frac{\alpha}{\sqrt{v_i + \varepsilon}} \nabla C(w_i)}$$

EMA of squares of past gradients ← $v_i = \beta v_{i-1} + \underbrace{(1-\beta)(\nabla C(w_i))^2}$

$\beta = 0.9$

impact reduced!
→ less aggressive than ADAGRAD

MATH INTERMEZZO: Exponential weighed (moving) Average EMA

$$\boxed{EMA_{today} = \beta\, EMA_{yesterday} + (1-\beta)\, P}$$

P: current value ex: stock price

β defines how much weight is given to historical DATA.

Example:

today

4 2 5 3 1 6

past days

$$EMA_{today} = \frac{2+5+3+1+6}{5} = 17/5 = 3.4$$

$$EMA_{yesterday} = \frac{4+2+5+3+1}{5} = \frac{15}{5} \rightarrow 3 \xrightarrow{\beta = 0.9} EMA_{today} = 2.7 + 0.1 \times 6 = \underline{\underline{3.3}}$$

Momentum:

↓

Uses EMA of past gradients to update weights

Ball

GD: issue if you are in a region with gentle slope.

we want to dampen sideway movements

we want faster learning along this direction

start 1) release ball

2) ball gains speed and once it has velocity it doesnt do steepest descent anymore because momentum will keep ball going in previous direction!

3) momentum will allow ball to deal with 'smaller' local minima.

ISSUE: overshoots minimum if momentum high! → could oscillate

~~Nesterov~~

$$\boxed{W_{i+1} = W_i \underbrace{- \overset{\uparrow 0.9}{\beta}\, update_{i-1}}_{\text{momentum coeff. move according to history}} \underbrace{- \overset{\text{learning rate}}{\alpha}\, \nabla C(W_i)}_{\text{move according to current GRAD}}}$$

2 effects.

START: $\text{update}_0 = 0$

$$\text{update}_1 = \beta\, \text{update}_0 + \alpha \nabla C(w_1) = \alpha \nabla C(w_1)$$
$$\text{update}_2 = \alpha \beta \nabla C(w_1) + \alpha \nabla C(w_2)$$
$$\vdots$$
$$\text{update}_i = \beta^{i-1} \alpha \nabla(w_1) + \beta^{i-2}\alpha \nabla C(w_2) + \dots + \alpha \nabla C(w_i)$$

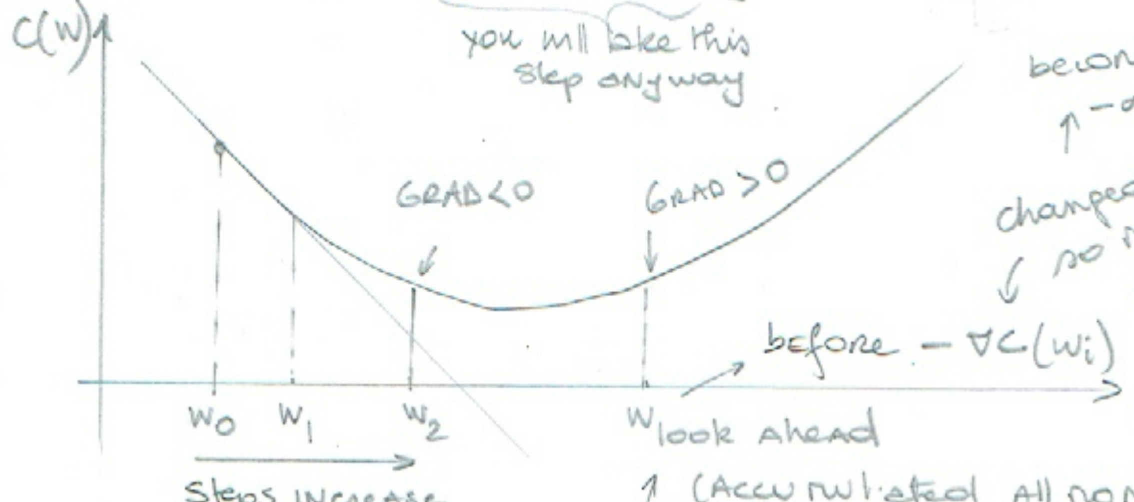
influence ↓ if we go ALONG!

as you go faster
→ take larger steps!

NESTEROV

momentum ↓ , current GRAD ↓

$$w_{i+1} = \underbrace{[w_i - \text{history}]}_{\text{you will take this step anyway}} - \alpha \nabla C(w_i)$$



becomes $-\alpha \nabla C(w_{\text{look Ahead}})$

changed sign so move to left!

before $-\nabla C(w_i)$

(Accumulated All momentum)

overshot → move to left.

Adam: VERY Popular!

↳ momentum + RMSprop

EMAs →
$$m_i = \beta_1 m_{i-1} + (1-\beta_1) \nabla C(w_i) \leftarrow \text{Accumulates history}$$
$$v_i = \beta_2 v_{i-1} + (1-\beta_2) \nabla C(w_i) \leftarrow \text{takes care of } \alpha$$

$\beta_1 = 0.9$
$\beta_2 = 0.999$
$\varepsilon = 10^{-8}$

↳ hyperparameters

$$\boxed{w_{i+1} = w_i - \frac{\alpha}{\sqrt{\hat{v}_i + \varepsilon}} \cdot \hat{m}_i}$$

$$\hat{m}_i = \frac{m_i}{1-\beta_1^i}$$

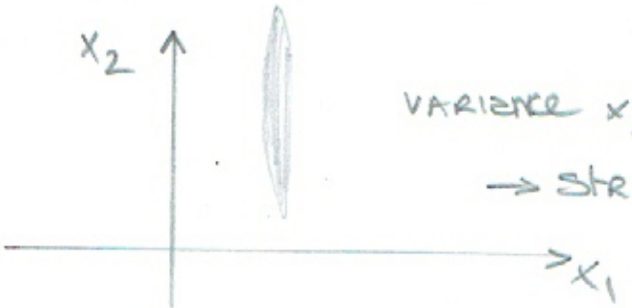
$$\hat{v}_i = \frac{v_i}{1-\beta_2^i}$$

Feature Scaling / Normalizing

Python: sklearn → StandardScaler

ISSUE: x_1 : 0 → 5

x_2 : 0 → 20.000



variance x_1 <<< variance x_2

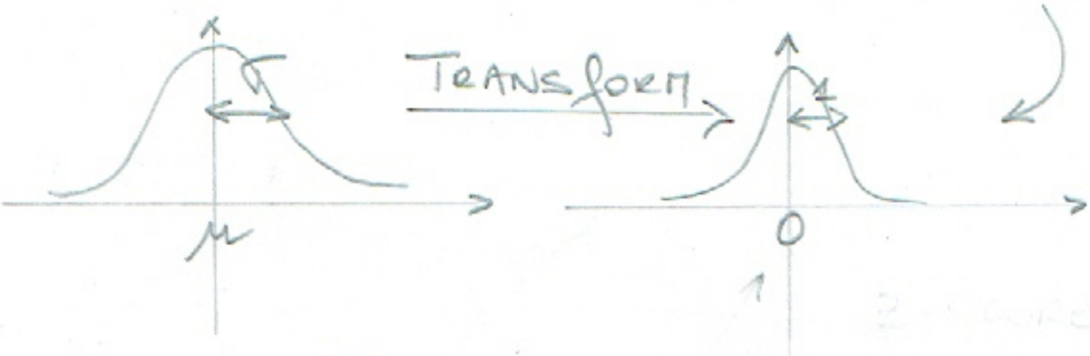
→ stretched contour lines

→ slows down convergence

↓ solution

good practice in machine learning ← normalization

MEAN IS ZERO $= \mu$
VARIANCE IS 1 $= \sigma^2$



$$\boxed{z = \frac{x - \mu}{\sigma}}$$

↳ use same μ, σ for normalization of test data