EEP 596: Adv Intro ML || Lecture 4

Dr. Karthik Mohan

Univ. of Washington, Seattle

January 12, 2023

Logistics

- Conceptual 1 is assigned
- Any questions on logistics?

→ OH: - F, Sa

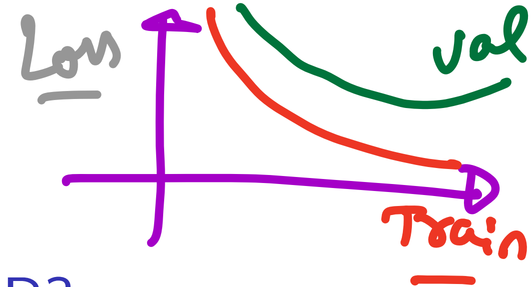
Quiz Section

Today's class!

- Recap of Overfitting and Regularization
- Gradient Descent and SGD Algorithm] Algo Foundation of ML
- Introduction to Classification in ML

ICE #1 (2 mins)

Train error
Val Error

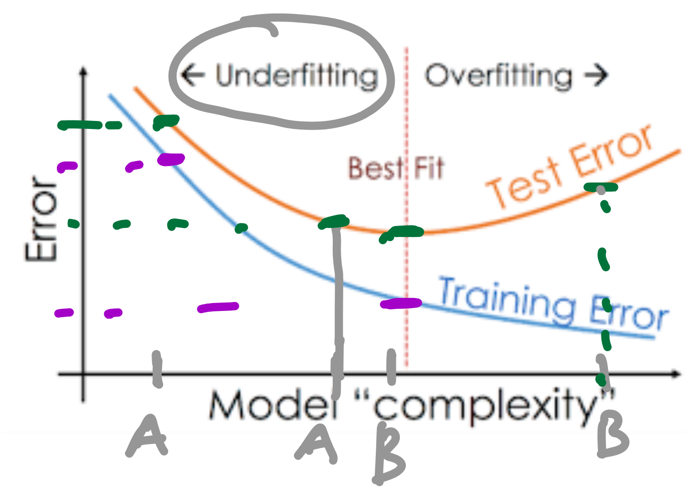


When is Model A under-fitting as compared to Model B?

Let A_{train} be train error of model A and A_{val} be validation error of model A and the same notation for model B.

- a) $A_{train} < B_{train}$ and $A_{val} > B_{val}$
- b) $A_{train} > B_{train}$ and $B_{val} < A_{val}$
- c) $A_{train} < B_{train}$ and $A_{val} < B_{val}$
- d) $A_{train} > B_{train}$ and $B_{val} > A_{val}$

ICE #1 graphed



$\Rightarrow B_{train} < A_{train}$

$\Rightarrow B_{val} < A_{val}$

Over-fitting and Remedies

Regularize ← Penalty + Loss = Regularized Loss

Remedy	Name	Benefits
$\|w\|_2^2$ ℓ_2 Reg.	Ridge Regression	No large weights
$\|w\|_1$ ℓ_1 Reg.	Lasso	Removes un-important features
$\ell_1 - \ell_2$ Reg.	Elastic Net	Combined benefits
Feature Selection		Reduces d so that $d << N$
Increase dataset size	Data Aug.	Increases N so that $N >> d$

(+)

1) Real Data Request

2) Synthetic Data Aug

3) Re-sampling

overfitting ← [$d >> N$]

d: Feature dim

N: #examples / #datapts

a) Increase $N\uparrow$ b) Decrease $d\downarrow$

Understanding ℓ_1 and ℓ_2 norms better

Interesting Geometry of these norm $- \ell_1, \ell_2, \ell_p$

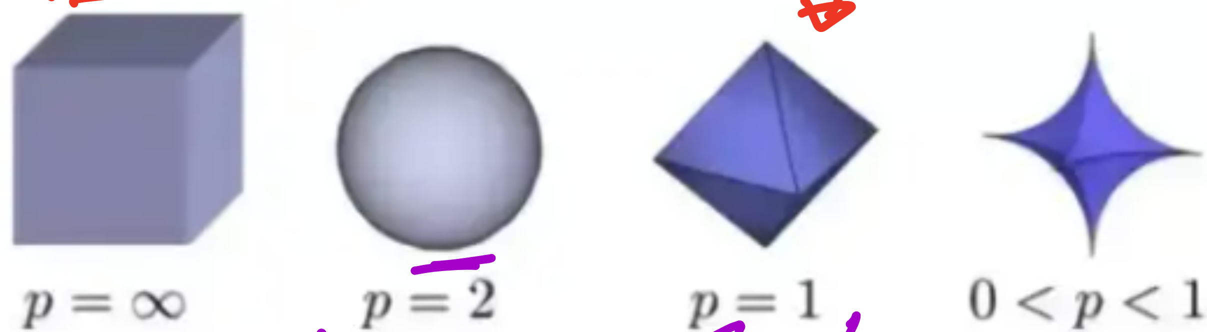
$0 < p < \infty$

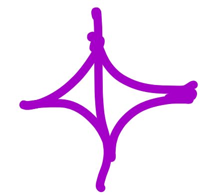
Norm Balls

Cube

Louvre museum

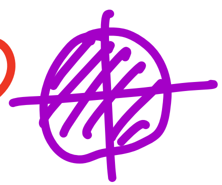
3d :-





2d :-

$\max(|w_1|, |w_2|) \leq 1$



$p = 2$

Euclidean norm

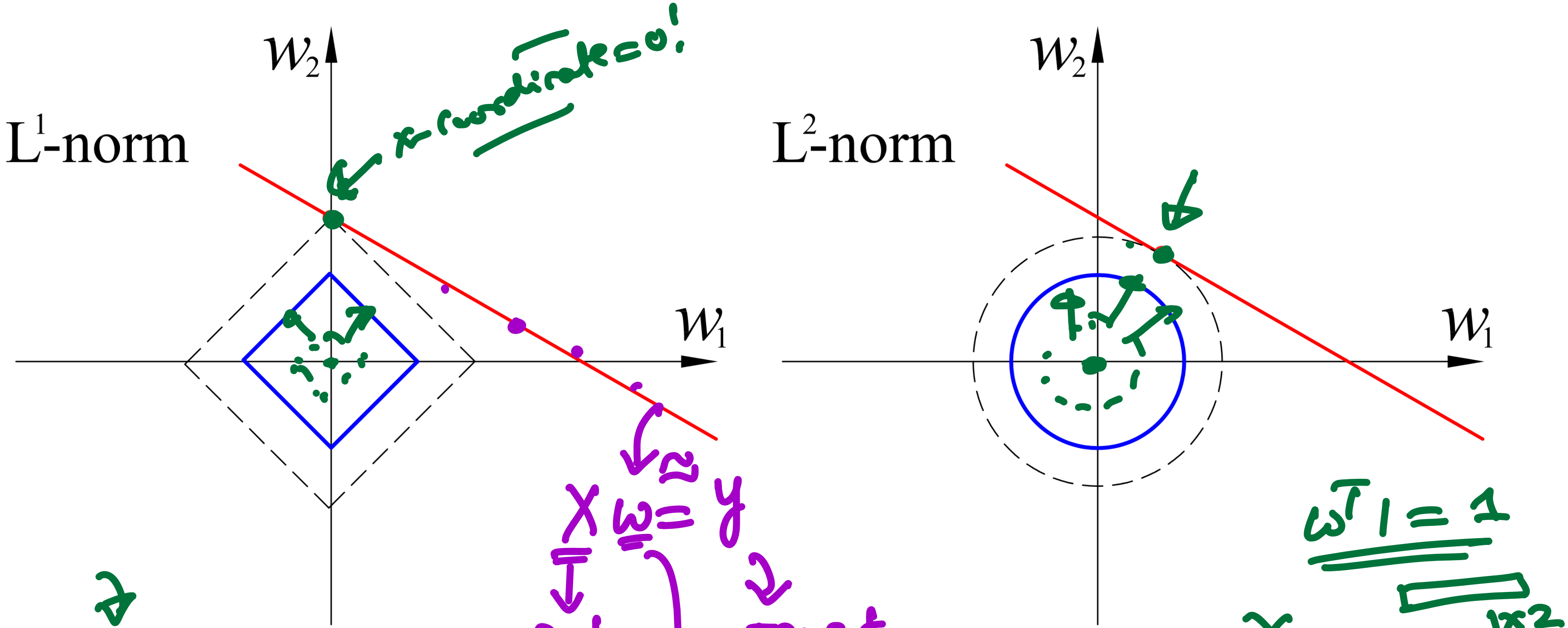
$\|w\|_2^2 < 1 \equiv w_1^2 + w_2^2 \leq 1$

$\|w\|_1 \leq 1$

$|w_1| + |w_2| \leq 1$

Understanding ℓ_1 and ℓ_2 norms better

Linear Regression ⟺ overfitting



$X \underline{w} \approx y$

X → Data, w → parameters/weights, y → Target

$w^T 1 = 1$

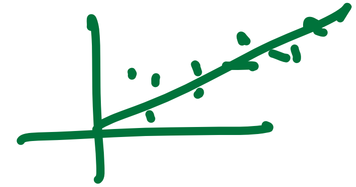
X (n×d), (1×2)

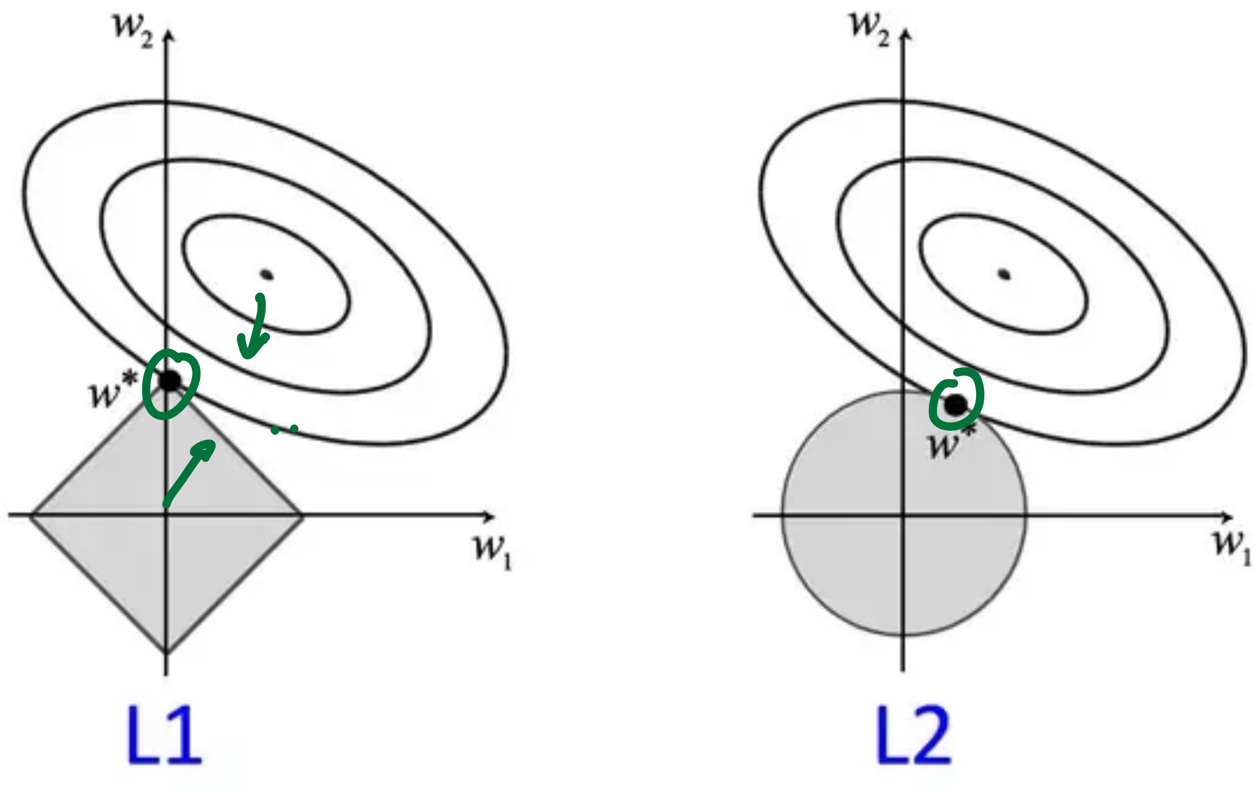
$\min \|w\|$ s.t. $Xw = y$

$N \ll d$ → Infinitely many solns when overfitting!

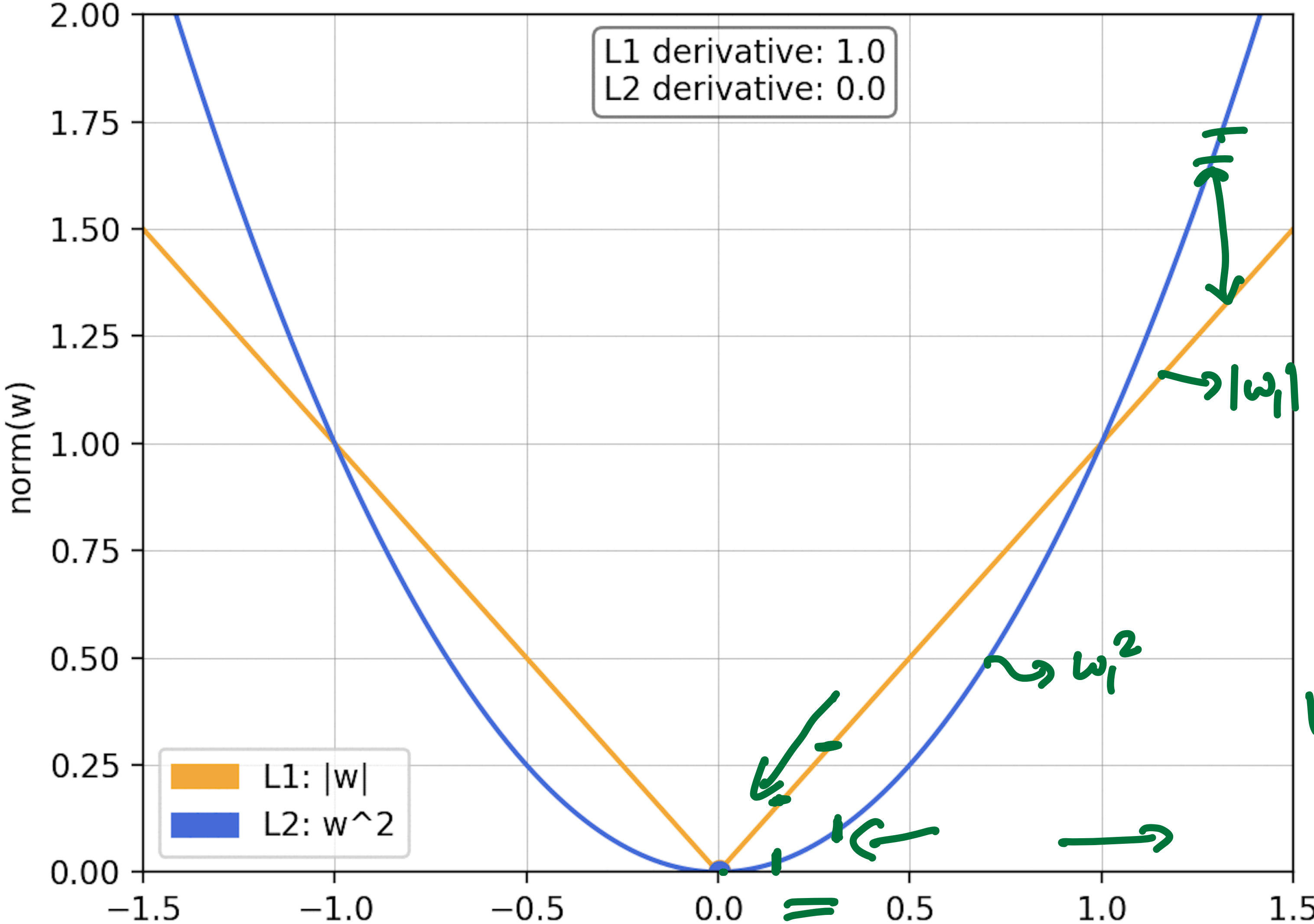
Understanding ℓ_1 and ℓ_2 norms better

$\|X\hat{w} - y\|_2 = \underline{\varepsilon}$ ⟶





Understanding ℓ_1 and ℓ_2 norms in one dimension



w_1

Example:

$w_1 = 1.5$

$w_1^2 = 2.25$

$|w_1| = 1.5$

$w_1 = 0.1$

$w_1^2 = 0.01$!

$|w_1| = 0.1$

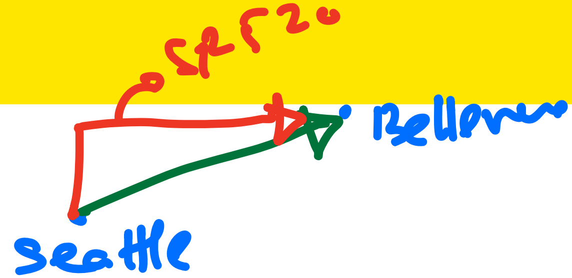
Over-fitting and Remedies

Remedy	Name	Benefits
ℓ_2 Reg.	Ridge Regression	No large weights
ℓ_1 Reg.	Lasso	Removes un-important features
$\ell_1 - \ell_2$ Reg.	Elastic Net	Combined benefits
Feature Selection		Reduces d so that $d << N$
Increase dataset size	Data Aug.	Increases N so that $N >> d$

$$\min_{w} \underbrace{\|Xw - y\|_2^2}_{\text{Loss}} + \underbrace{\lambda \|w\|_2^2}_{\ell_2 \text{ penalty}} \quad \Big\} \text{ Ridge Reg.}$$

$$\min_{w} \|Xw - y\|_2^2 + \underbrace{\lambda \|w\|_1}_{\ell_1 \text{ penalty}} \quad \Big\} \text{ Lasso}$$

ICE #2



Manhattan and Euclidean Distance

Every **norm** of a vector (or a matrix) gives rise to a **distance metrics**. Norm is a measure of magnitude of a vector (or matrix) while distance metric is a measure of well, distance between two vectors. Consider for instance the distance between Seattle and Bellevue. If you drew a straight-line between the two cities, that would be the **Euclidean distance**. However, if you start in downtown seattle, and take SR-520, that is equivalent to the ℓ_1 distance or **Manhattan distance**. Compute the Euclidean and Manhattan distance between two vectors, $x = [1, 2, 3], y = [2, 4, -1]$. The distances are closest to:

1. 7 and 4
2. 4 and 7
3. 7 and 5
4. 5 and 7

$$\|x-y\|_2 = \sqrt{(x_1-y_1)^2+(x_2-y_2)^2+(x_3-y_3)^2}$$

$$\|x-y\|_1 = |x_1-y_1|+|x_2-y_2|+|x_3-y_3|$$

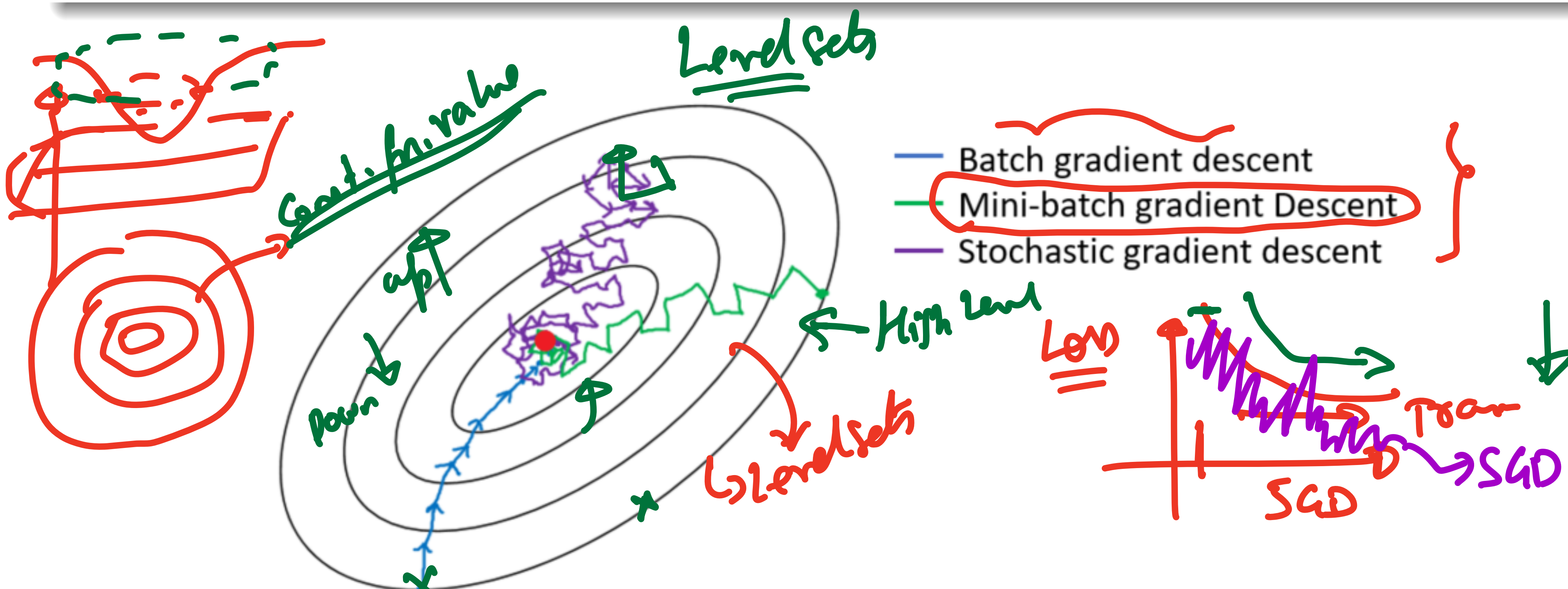
Understanding regularization better

Conceptual Assignment 2

We will look at the numerical impact of ℓ_1 and ℓ_2 norms (used in Lasso and Ridge Regression) on the weights learned in one of the conceptual assignments.

Underlying Engine behind ML Training

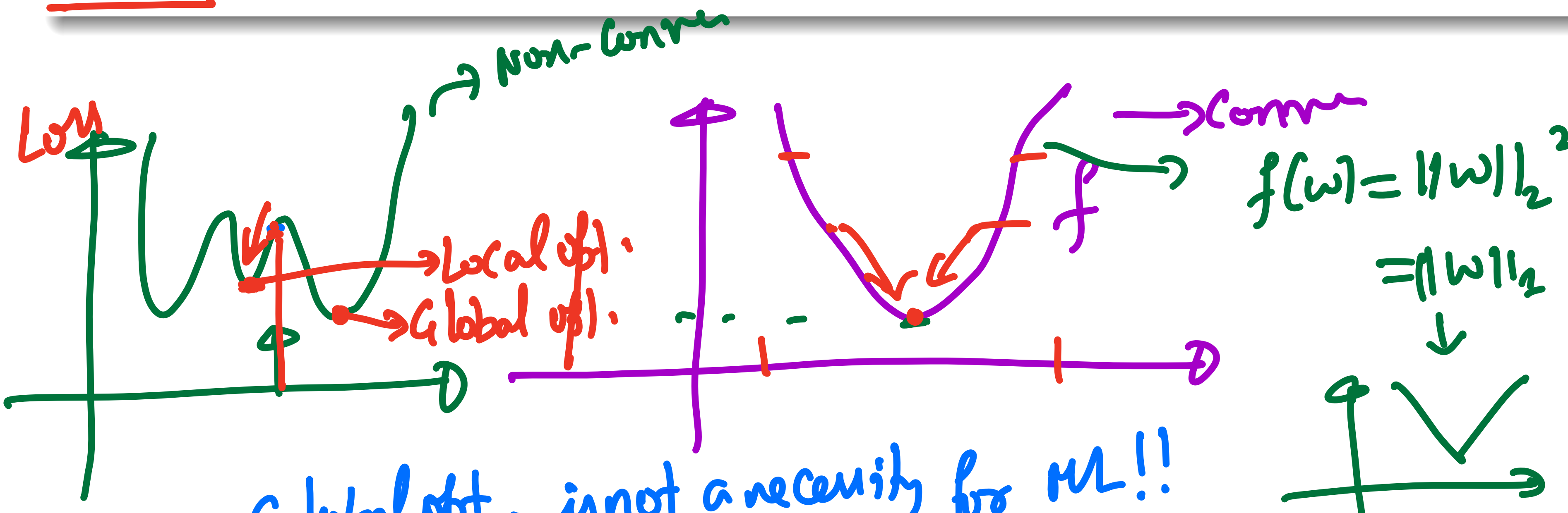
`(Mini-batch) Stochastic Gradient Descent` Almost every model and problem-space in ML uses SGD of some kind - Clustering, Regression, Deep Learning, Computer Vision and NLP to name a few. Almost every algorithm in every library - `Scikit-learn`, `Keras`, `Pytorch`, etc uses **mini-batch SGD under the hood**.



So what is Gradient Descent?

Fundamentally

Take a convex/non-convex function, f. GD allows you to find a local optimum to f.



Global opt. is not a necessity for ML!!
(over-fitting issues!)

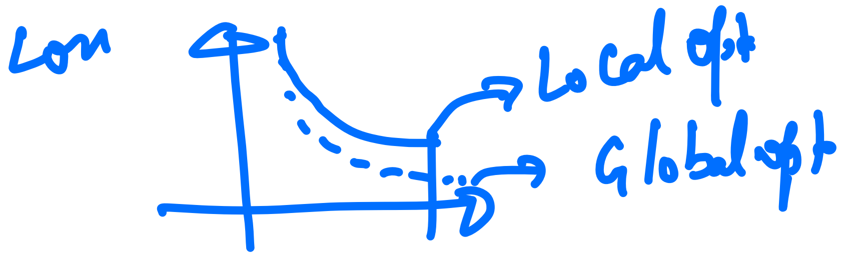
So what is Gradient Descent?

Fundamentally

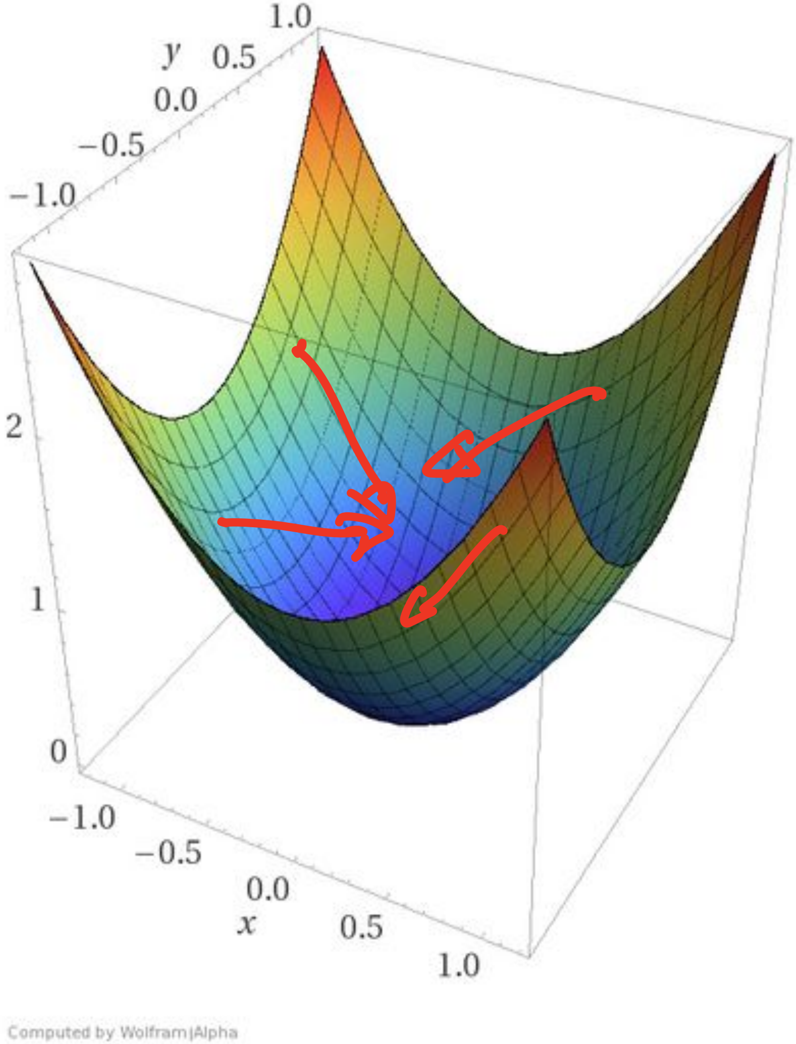
Take a convex/non-convex function, f. GD allows you to find a local optimum to f.

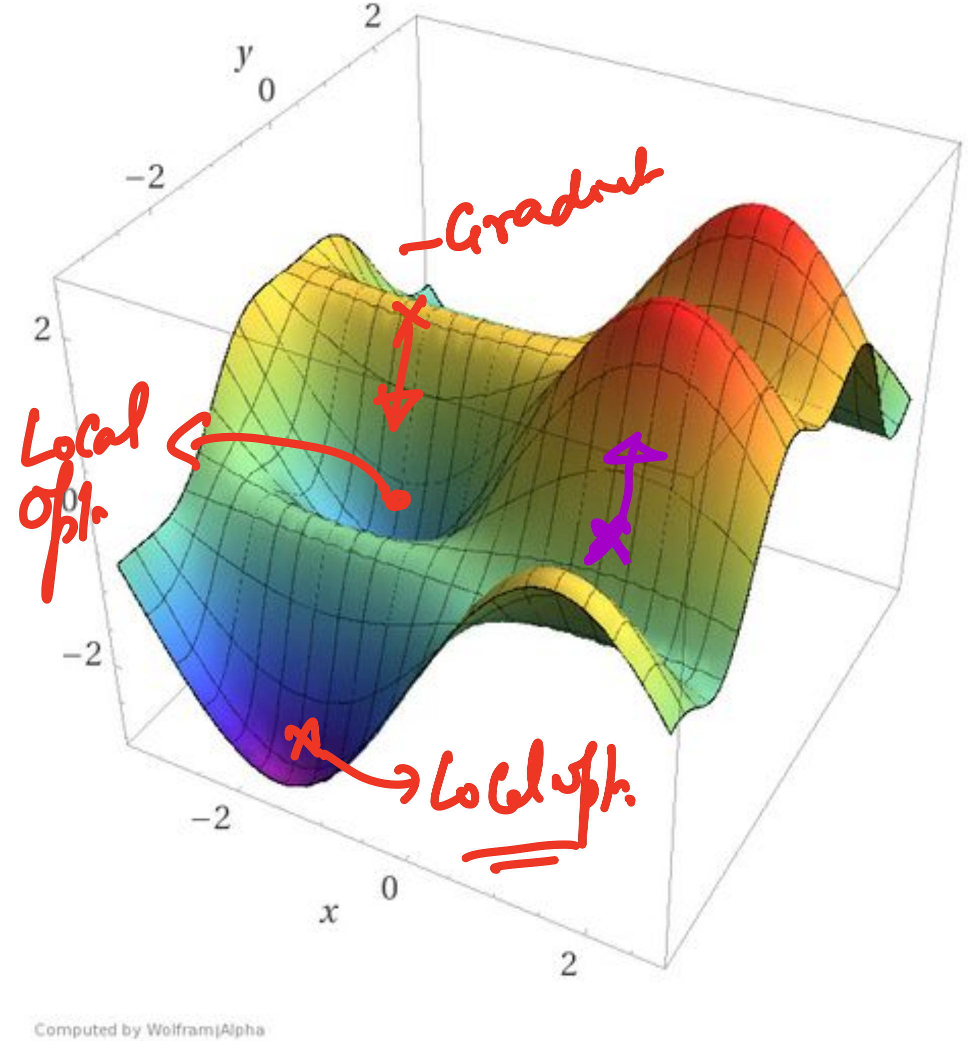
Why is this important?

Consider the Linear Regression problem. $\hat{w}$ is a local optimum to the function $f(w) = \frac{1}{2}\|Xw - y\|_2^2 + \lambda\|w\|_2^2$ } Ridge Regrn

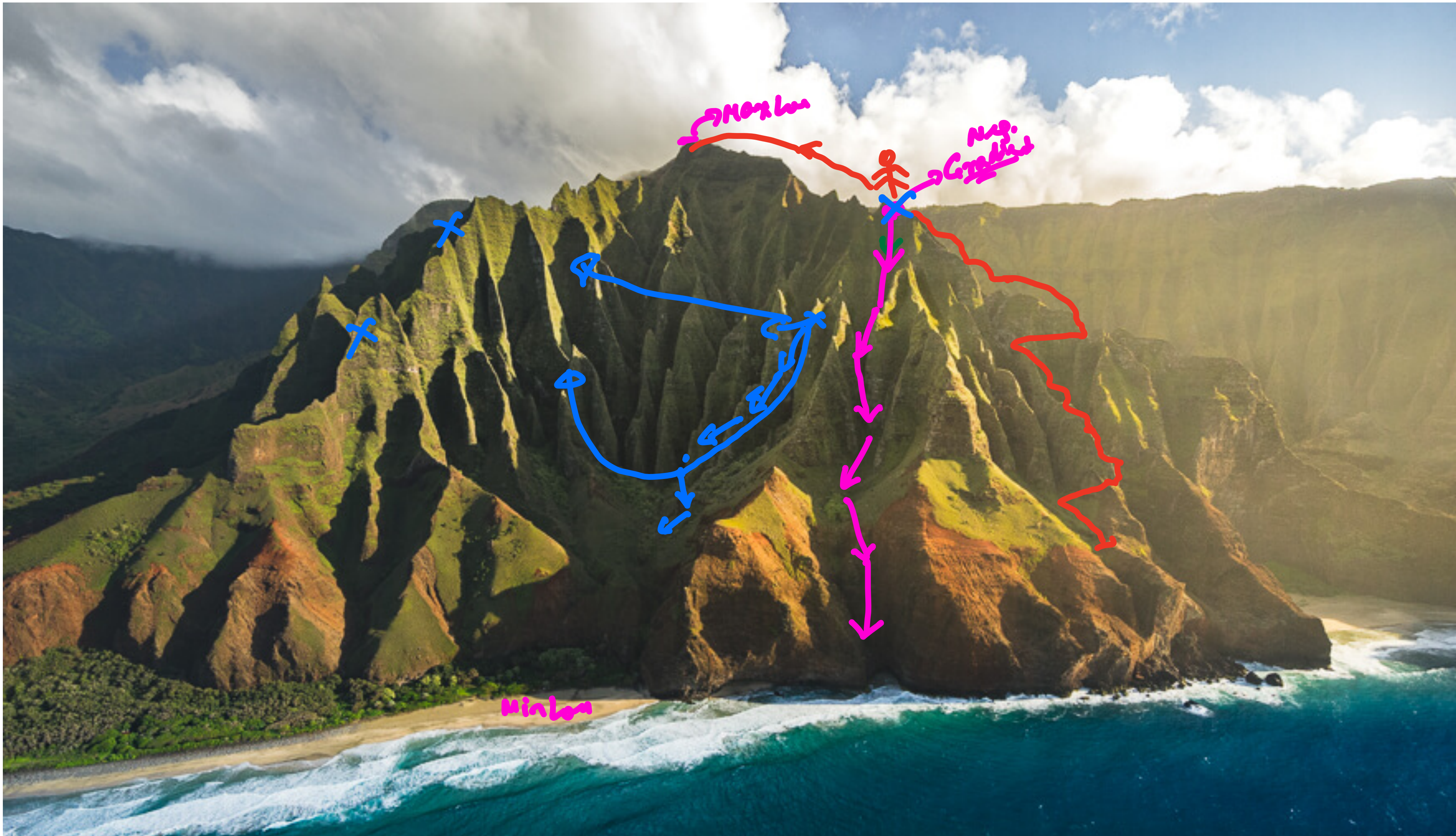


Negative Gradient helps you view the direction of descent





Negative Graidents on a Kauai peak!



Gradient Descent

Linear Reg.

$L(w) = \frac{1}{N} \|\tilde{X}w - \tilde{y}\|_2^2$

Batch Gradient Descent

Let us say we want to minimize $L(w)$ - Loss Function and find the best $\hat{w}$ that does that.

1. **Initialize** $w = w_0$ (maybe randomize)

Gradient Descent

Batch Gradient Descent

Let us say we want to minimize $L(w)$ - Loss Function and find the best $\hat{w}$ that does that.

1. **Initialize** $w = w_0$ (maybe randomize)
2. **Gradient Descent** $w \leftarrow w - lr * \nabla L(w)$ } Take a step in the direction of -ve Gradient

"Learning Rate" (Step Size)

"Learning Rate Schedulers"

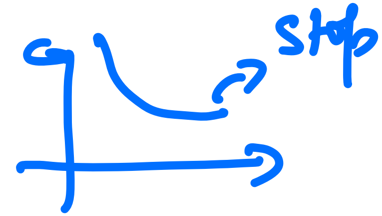
Gradient Descent

Batch Gradient Descent

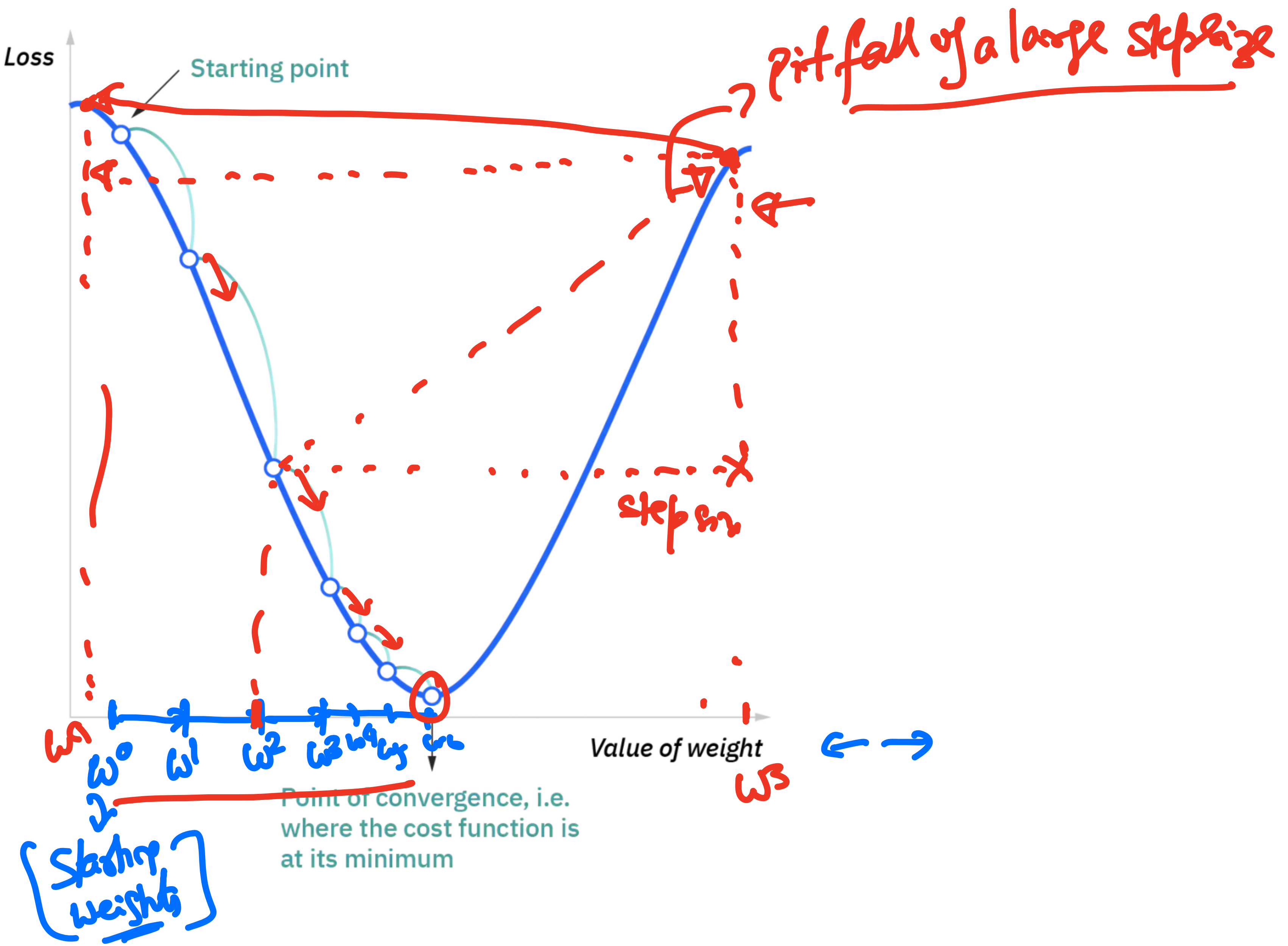
Let us say we want to minimize $L(w)$ - Loss Function and find the best $\hat{w}$ that does that.

1. **Initialize** $w = w_0$ (maybe randomize)
2. **Gradient Descent** $w \leftarrow w - lr * \nabla L(w)$
3. **Iterate** Repeat step 2 until w converges, i.e.

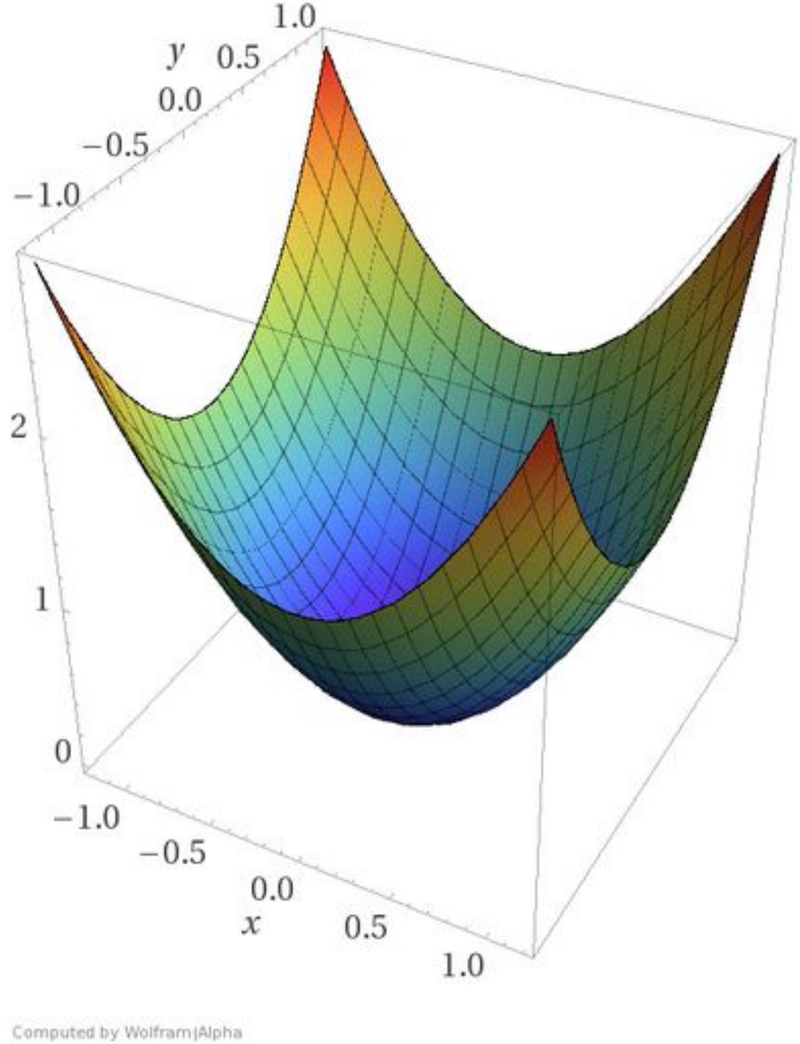
$$\|w^{k+1} - w^k\| / \|w^k\| \leq 10^{-3}$$

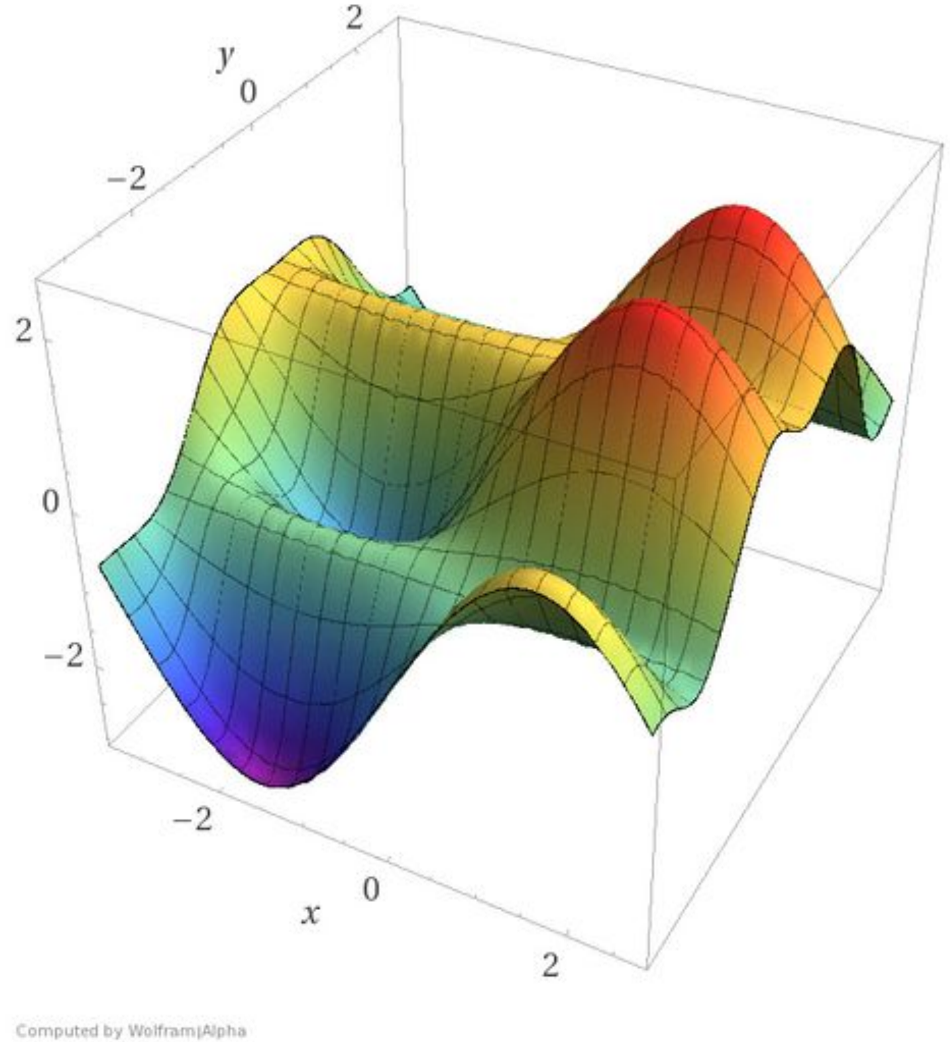


GD in one dimension



Loss function in 2 dimensions





ICE #3

Gradient of Ridge Regularizer (2 mins)

Find the gradient of the regularization function, $R(w) = \underbrace{\lambda \|w\|_2^2}_{\lambda(w_1^2 + w_2^2 + \dots + w_d^2)}$. I.e. obtain the expression for, $\nabla_w R(w)$?

- a) $2\lambda\|w\|_2$
- b) $\lambda\|w\|_2 w$
- c) $2\lambda w$
- d) $2\lambda\|w\|_2 w$

$\frac{\partial}{\partial w_1} R(w) = 2\lambda w_1$

$\frac{\partial}{\partial w_j} = 2\lambda w_2$

$2\lambda \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix}$

Gradient = vector of partial derivatives

$\begin{bmatrix} \frac{\partial}{\partial w_1} R(w) \\ \frac{\partial}{\partial w_2} R(w) \\ \vdots \\ \frac{\partial}{\partial w_d} R(w) \end{bmatrix}$

$[\quad] \rightarrow \nabla_w R(w)$

ICE #3

Gradient of Ridge Regularizer (2 mins)

Find the gradient of the regularization function, $R(w) = \lambda \|w\|_2^2$. I.e. obtain the expression for, $\nabla_w R(w)$?

- a) $2\lambda \|w\|_2$
- b) $\lambda \|w\|_2 w$
- c) $2\lambda w$
- d) $2\lambda \|w\|_2 w$

In Assignment 2

We will have a question comparing GD and exact solution for Ridge Regression! Comparison on computation time and accuracy and how both the methods scale?

Gradient Descent Properties

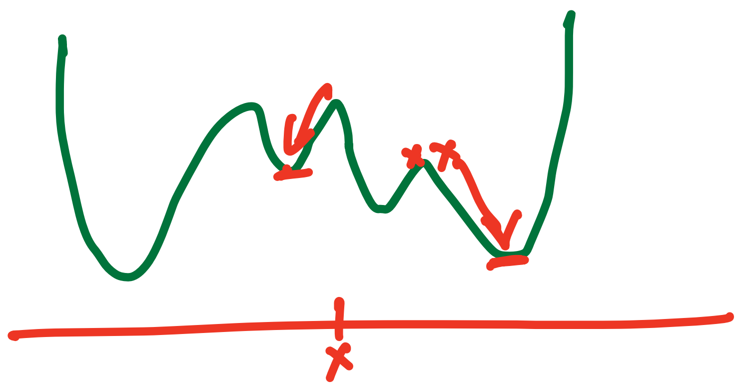
1. Gradient Descent converges to a local minimum

Gradient Descent Properties

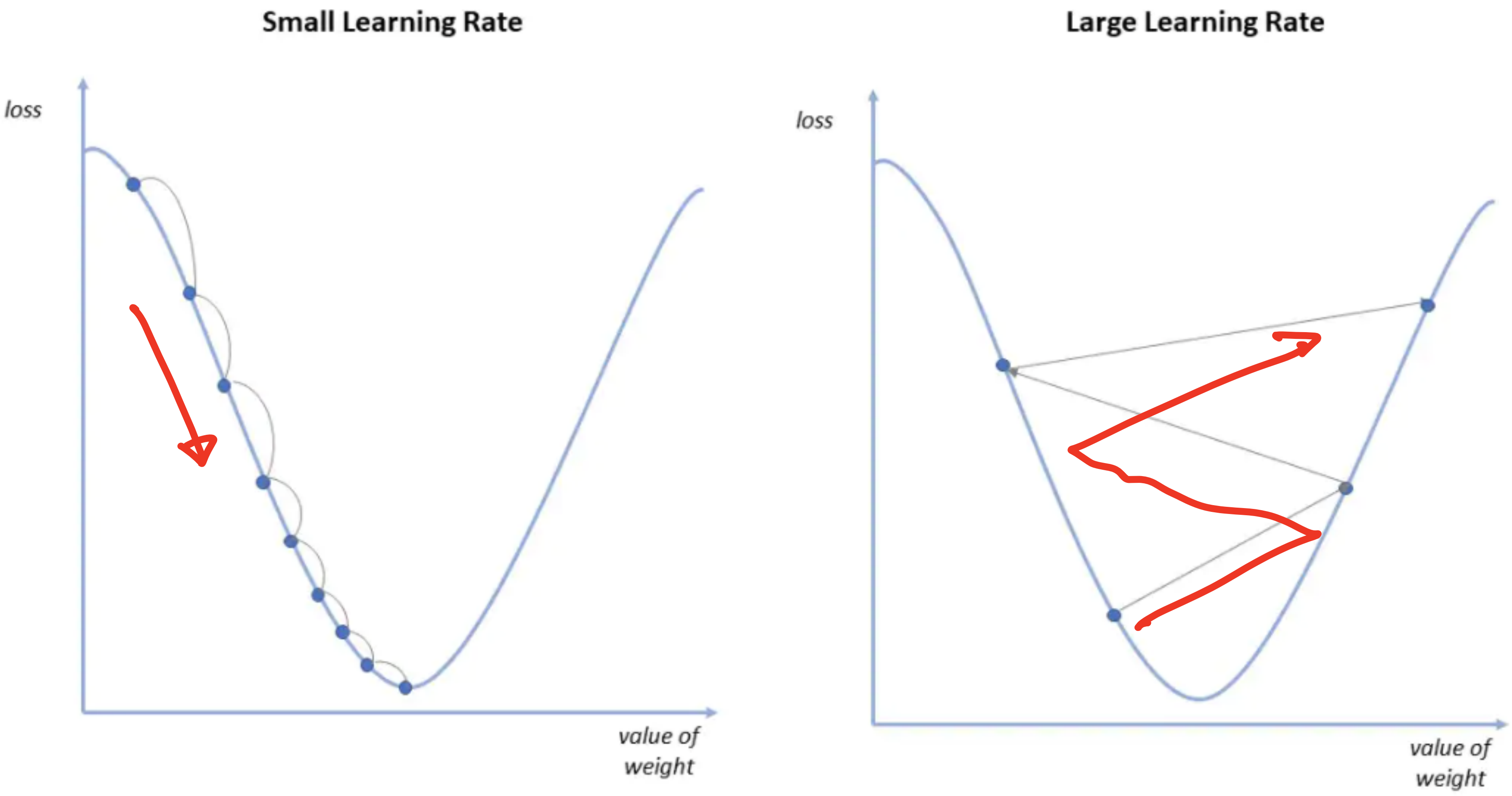
1. Gradient Descent converges to a local minimum
2. If L is a convex function, all local minima become a global minima!

Gradient Descent Properties

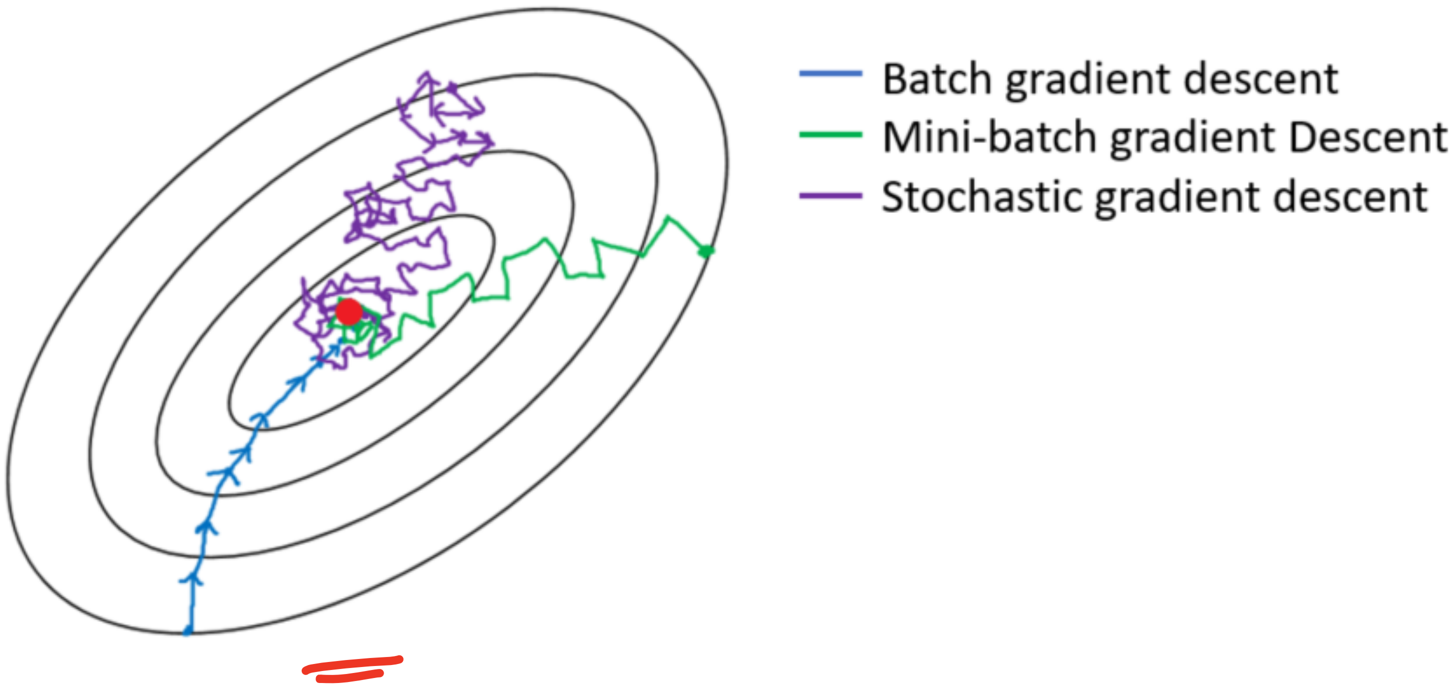
1. Gradient Descent converges to a local minimum
2. If L is a convex function, all local minima become a global minima!
3. Wherever we start, gradient descent usually finds a local minima closest to the start.



Effect of Learning Rate



GD behavior in the search space



Gradient descent in practice - SGD!

$N = 100$

$L(w) = \frac{1}{100}\left(L_1(w) + L_2(w) + \dots + L_{100}(w) \right)$

SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w.

1. **Initialize** w^0 (randomize)

Gradient descent in practice - SGD!

SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w.

1. **Initialize** w^0 (randomize) Pick index i at random between 1 and N!

20

Gradient descent in practice - SGD!

SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w.

1. **Initialize** w^0 (randomize) Pick index i at random between 1 and N!
2. **Gradient Descent** $w^{k+1} \leftarrow w - lr * \nabla L_i(w^k)$

Gradient descent in practice - SGD!

stochastic

stochastic

Random Permute:-

N=6

1 2 3 4 5 6

5 6 2 4 1 3

Pass = Going through the dataset once

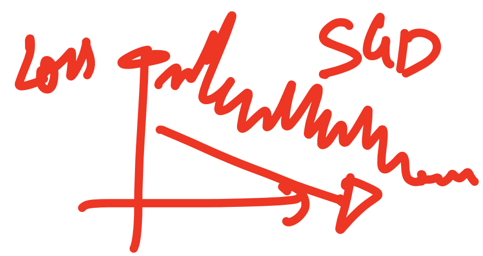
k passes

SGD

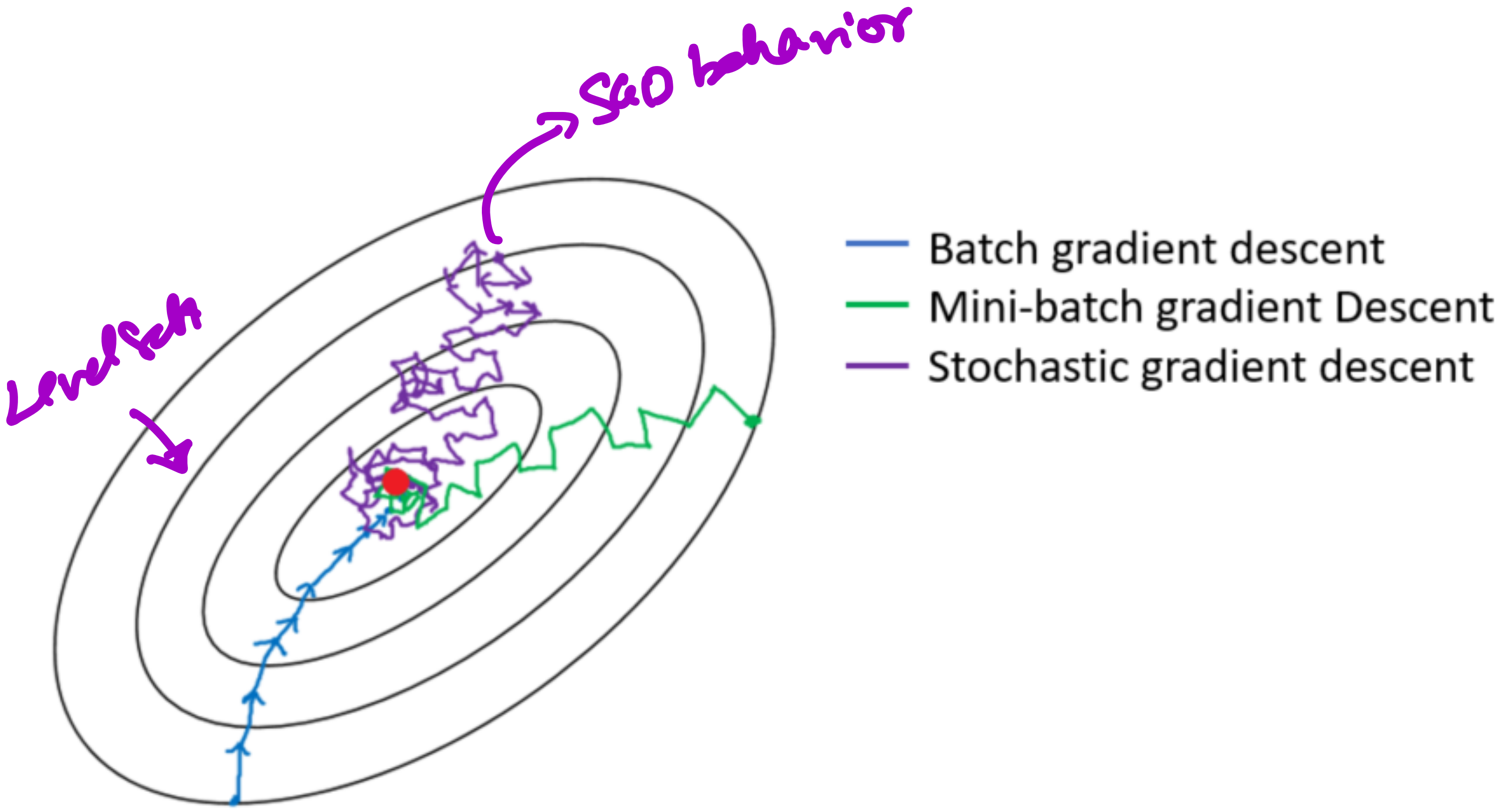
Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w.

1. **Initialize** w^0 (randomize) 2 Pick index i at random between 1 and N!
2. 3 **Gradient Descent** $w^{k+1} \leftarrow w - lr * \nabla L_i(w^k)$
3. 4 **Iterate** Repeat step 2 and 3 until w converges, i.e.

$$\|w^{k+1} - w^k\| / \|w^k\| \leq 10^{-3}$$



SGD behavior in search space



SGD in practice - mini-batch SGD!

mini-batch SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w. Let B be the number of batches and k be the batch size.

$k = 150$

1. **Initialize** $w = w_0$ (randomize)

SGD in practice - mini-batch SGD!

mini-batch SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w. Let B be the number of batches and k be the batch size.

1. **Initialize** $w = w_0$ (randomize) 2 Pick a batch of k data points at random between 1 and N: $i_1, i_2, \ldots, i_k$!

SGD in practice - mini-batch SGD!

mini-batch SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w. Let B be the number of batches and k be the batch size.

1. **Initialize** $w = w_0$ (randomize) Pick a batch of k data points at random between 1 and N: $i_1, i_2, \ldots, i_k$!
2. **Gradient Descent** $w^{k+1} \leftarrow w^k - lr * \underbrace{\sum_{j=1}^{k} \nabla_w L_{i_j}(w^k)}_{\text{Learning from k data pts.}}$

(In SGD - Learn from 1 data pt.)

SGD in practice - mini-batch SGD!

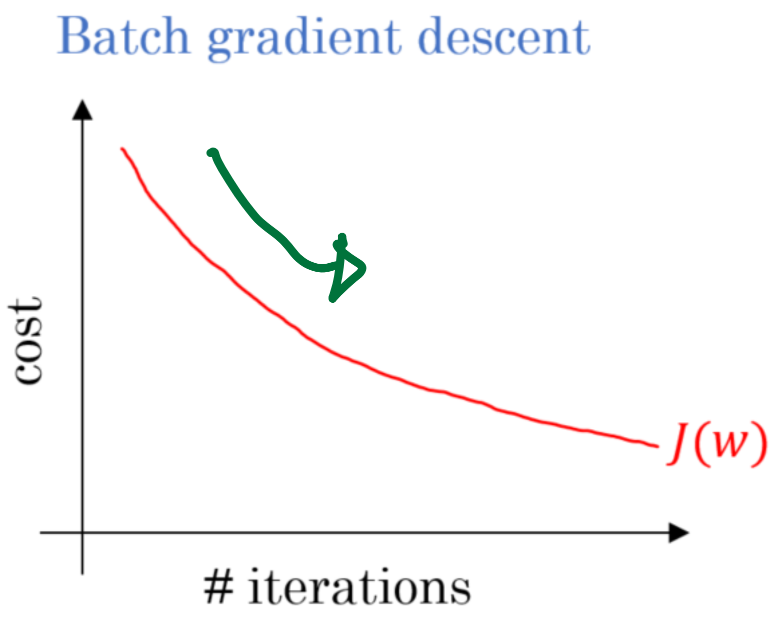
mini-batch SGD

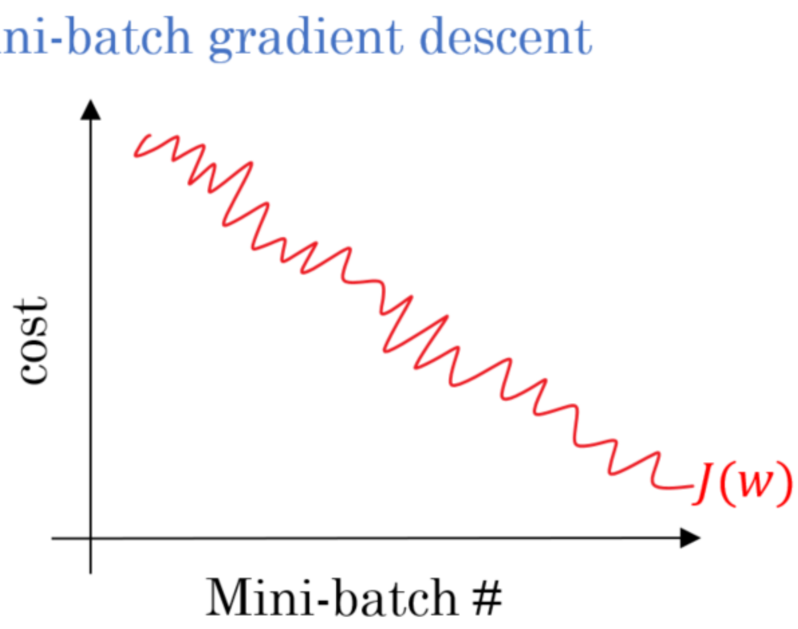
Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w. Let B be the number of batches and k be the batch size.

1. **Initialize** $w = w_0$ (randomize) 2 Pick a batch of k data points at random between 1 and N: $i_1, i_2, \ldots, i_k$!
2. 3 **Gradient Descent** $w^{k+1} \leftarrow w^k - lr * \sum_{j=1}^{k} \nabla_w L_{i_j}(w^k)$
3. 4 **Iterate** Repeat step 2 and 3 until w converges, i.e.

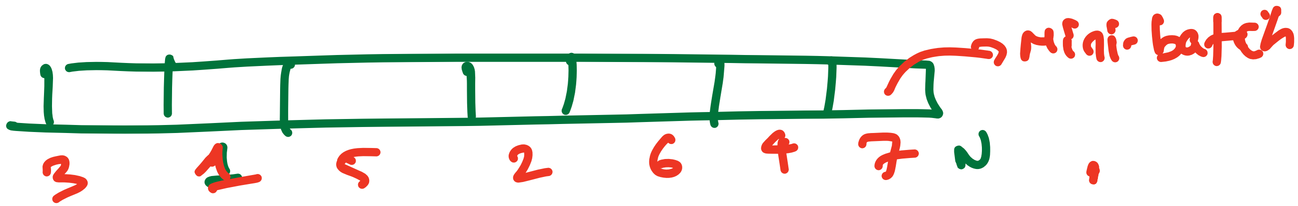
$$\|w^{k+1} - w^k\| / \|w^k\| \leq 10^{-3}$$

GD vs Mini-batch convergence behavior





GD vs mini-batch SGD



Factor	GD	Mini-batch SGD
Data	All per iteration	Mini-batch (usually 128 or 256)
Randomness	Deterministic	Stochastic
Error reduction	Monotonic	Stochastic
Computation	High	Low
Memory big data	Intractable	Tractable
Convergence	Low relative error	Few "passes" on data
Local Minima traps	Yes	No

Mini-batch SGD "generalize" better on un-seen data than GD!

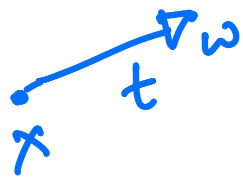
Proof for Gradient being direction of steepest ascent

$g(t) = f(x + tw)$

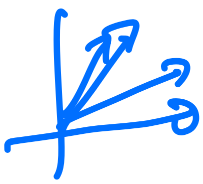
$g'(t) = (\nabla f(x + tw))^T w$

$\max_{w} g'(0) = \max_{w} \nabla f(x)^T w$

$\Rightarrow w = \nabla f(x)$



Because of Cauchy-Schwarz Ineq



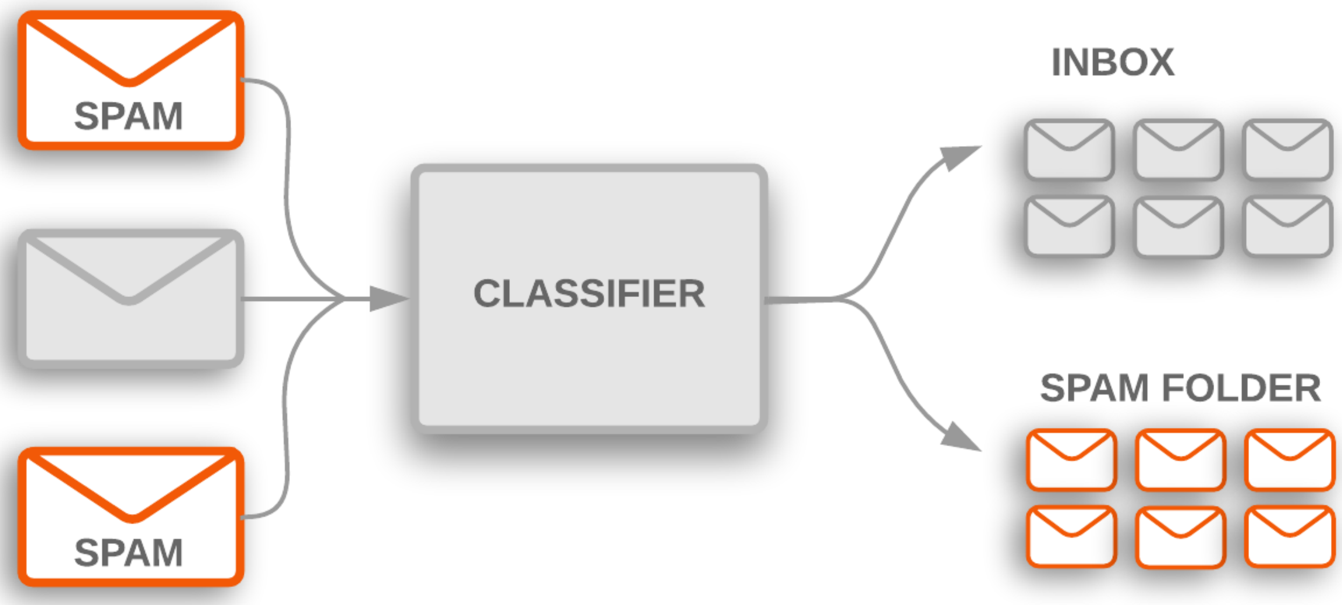
$x^T y \leq \|x\|_2 \|y\|_2$

Equality when $x = y$!

Course Outline

Week	Lecture Material	Assignment
1	Linear Regression	Housing Price Prediction
2	**Classification**	**Spam classification (Kaggle)**
3	Classification	Flower/Leaf classification
4	Clustering	MNIST digits clustering
5	Anomaly Detection	Crypto Prediction (Kaggle + P)
6	Data Visualization	Crypto Prediction (Kaggle + P)
7	Deep Learning	Visualizing 1000 images
8	Deep Learning (DL)	ECG Arrythmia Detection
9	DL in NLP	TwitterSentiment Analysis (Kaggle + P)
10	DLs in Vision	TwitterSentiment Analysis (Kaggle + P)

Classification in Machine Learning



Difference between Classification and Regression

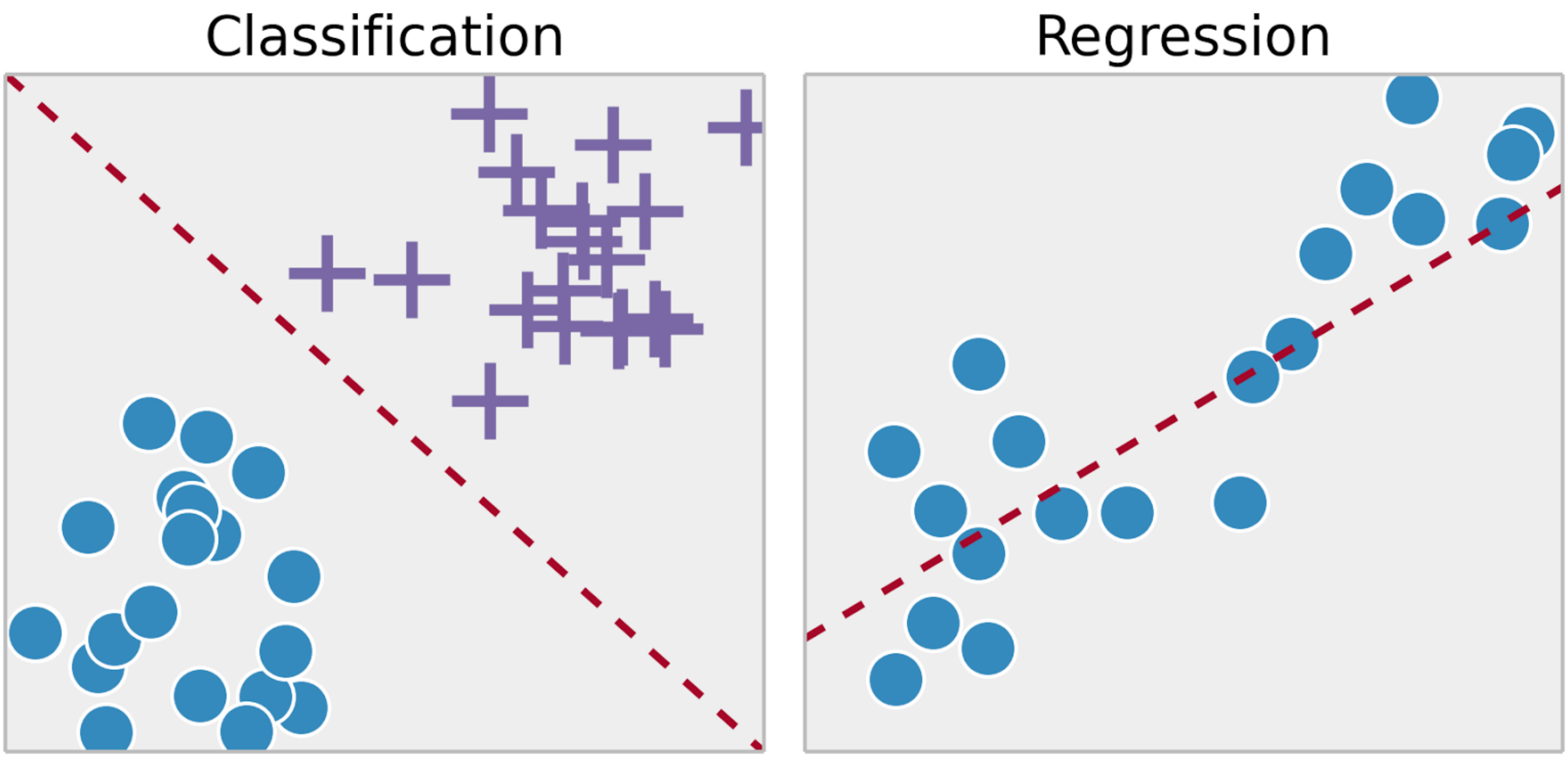
Simple difference

The target type in Regression is **numeric** whereas that in classification is **categorical**

Difference between Classification and Regression

Simple difference

The target type in Regression is **numeric** whereas that in classification is **categorical**



Types of Classification

Binary vs Multi-class classification

With binary categories, its a binary classification problem and with multiple categories, we have a multi-class classification.

Types of Classification

Binary vs Multi-class classification

With binary categories, its a binary classification problem and with multiple categories, we have a multi-class classification.

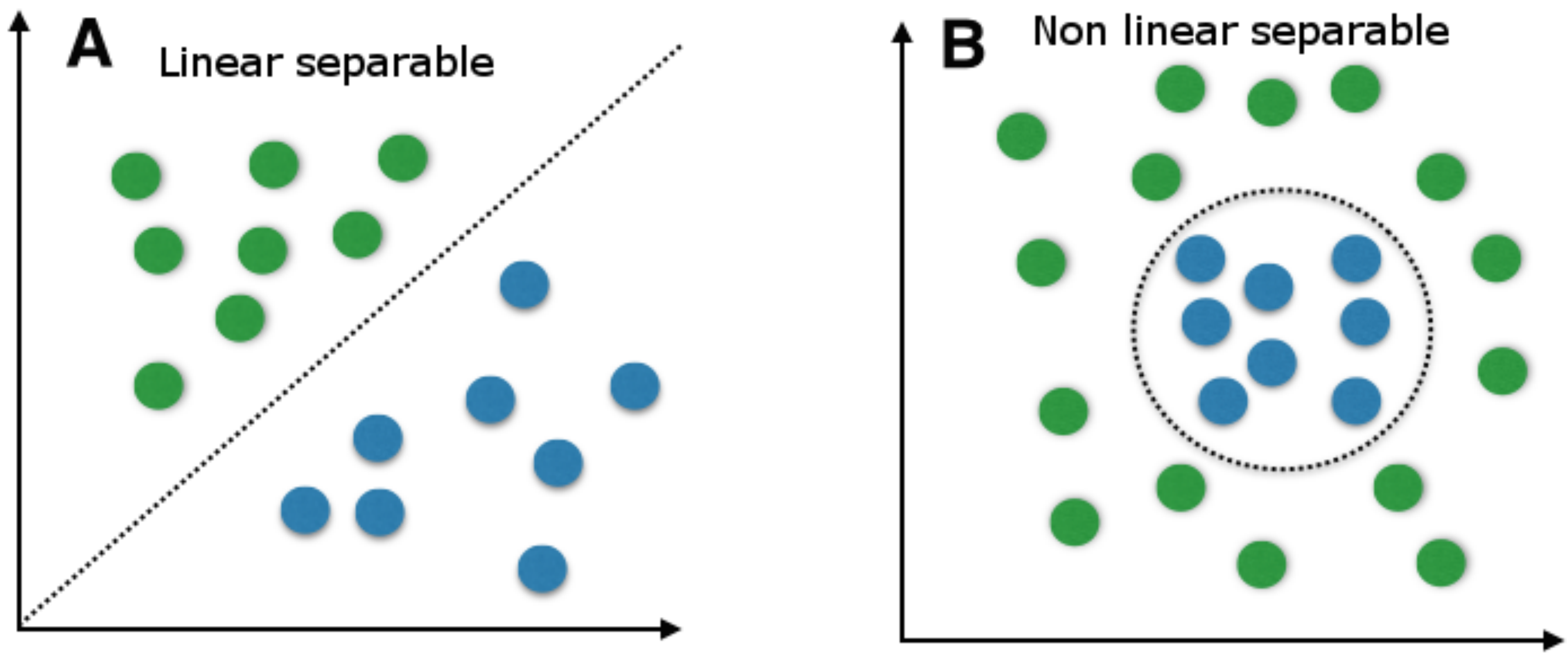
Target is called Label

For binary classification, the convention is to label the target as positive or negative. Example: Positive for spam and negative for not-spam

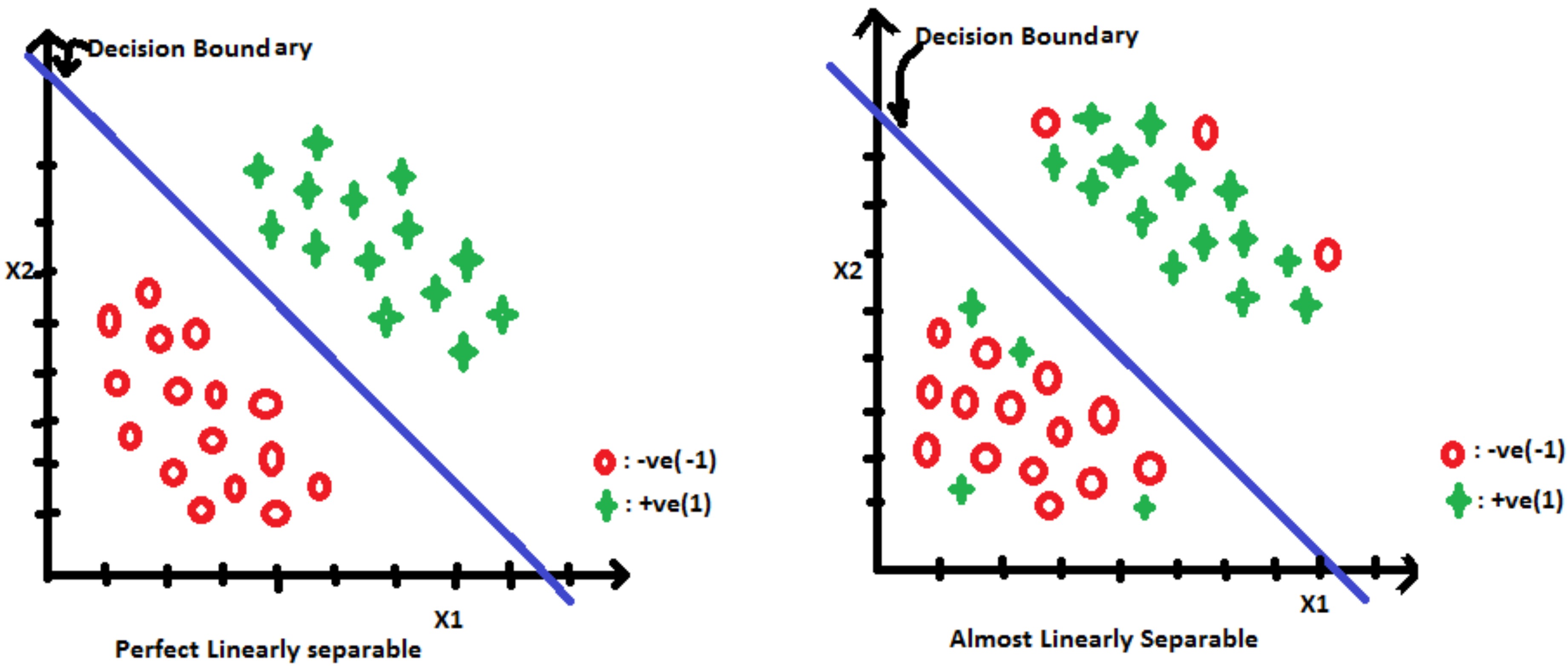
Spam Classification Example

Email excerpt	Type	Label
Could you please respond by tomorrow?	Not-spam	-1
Congratulations!!! You have been selected...	Spam	$+1$
Looking forward to your presentation...	Not-spam	-1
$\dots$	$\dots$	$\dots$

Linear Separability



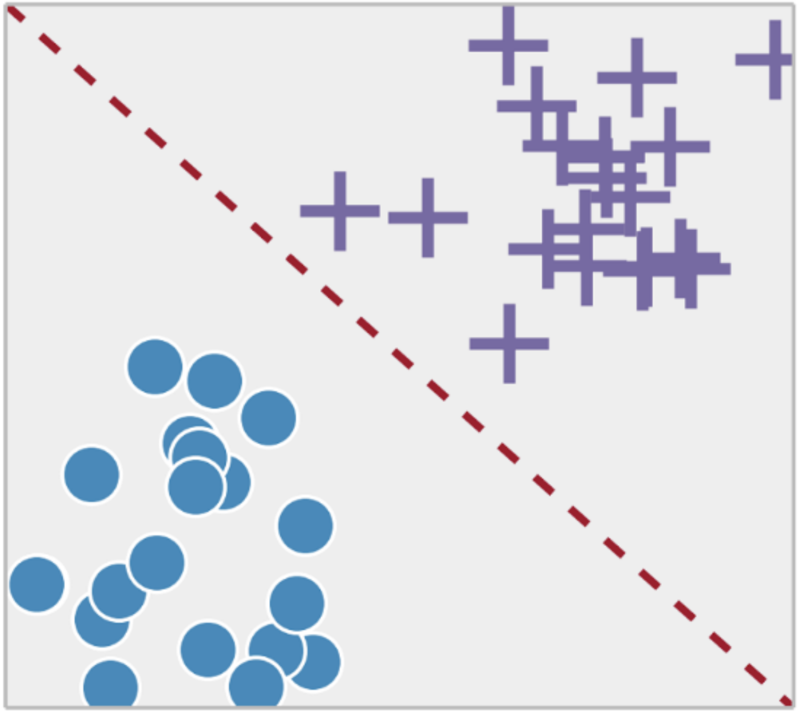
Approximate Linear Separability



ICE #4

Which of the following data sets is the closest to being linearly separable?

Logistic Regression



LR fundamentals

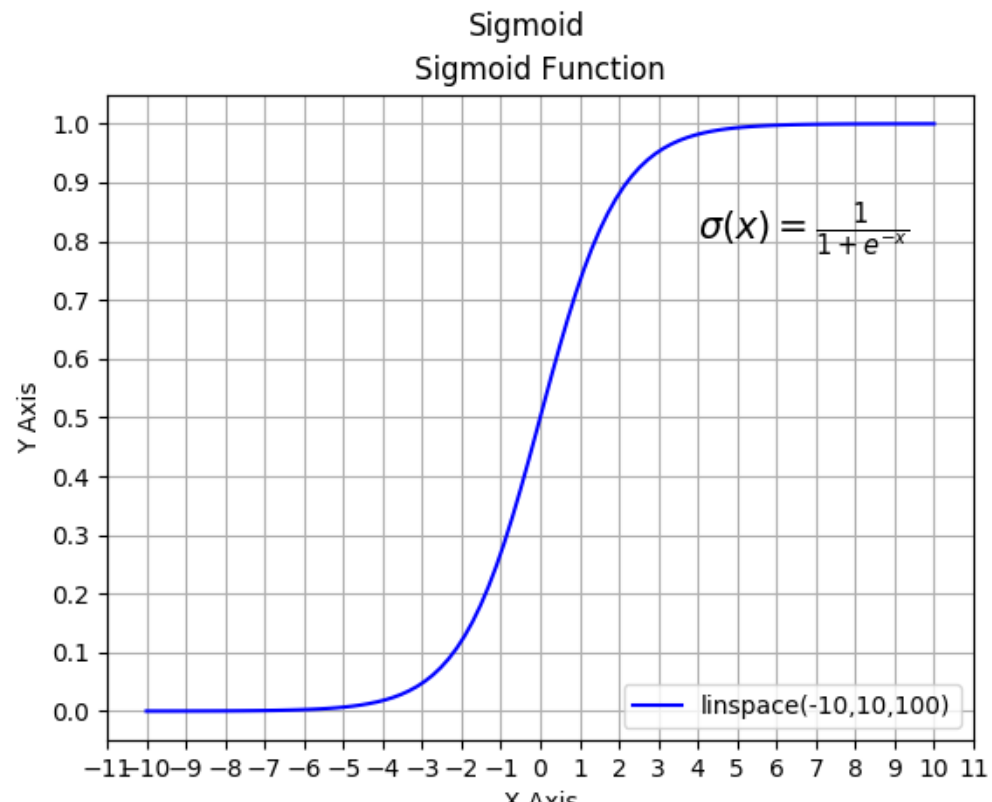
- Linear Model
- Want score $w^T x^i > 0$ for $y_i = +1$ and $w^T x_i < 0$ for $y_i = -1$!
- If linearly separable data, above is feasible. Else, minimize error in separability!!

Logistic Regression

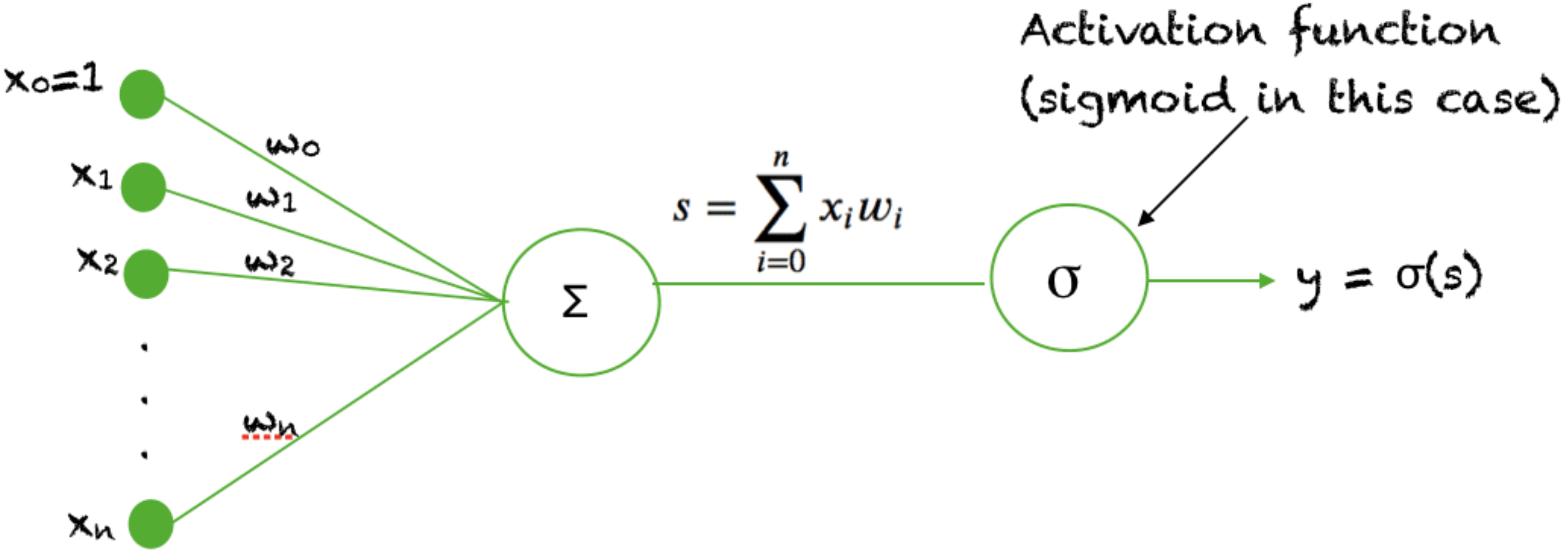
Probability for a class

In LR, the score, $w^T x$ is converted to a probability through the sigmoid function. So we can talk about $P(\hat{y}^i = +1)$ or $P(\hat{y}^i = -1)$

Sigmoid Function



LR represented Graphically



Logistic Regression

LR Prediction

$$\hat{y}_i = \frac{1}{1 + e^{-\hat{w}^T x^i}}$$

LR Loss

Assume that $y_i = 0$ or $y_i = 1$ (i.e. the negative class has a label 0). Then the binary cross-entropy loss applies to LR:

$$\min_{w} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

Summary

- Why gradients are important?
- GD vs SGD vs Mini-batch SGD
- Why mini-batch SGD is preferred?
- Regression vs Classification
- Decision Boundary and Linear Separability
- Logistic Regression