CS221 Problem Workout

Week 2

Introduction

Samantha Liu

General OH: Thursdays 2:00-4:00 Online
HW OH: Tuesdays 2:00-4:00 Online

Kevin Li

General OH: Wednesdays 5:00-7:00 Online
HW OH: Fridays 4:30-6:30 Online

Last Week: 3 Design Decisions in ML

Example: **linear regression** $\phi(x) \in \mathbb{R}^d, y \in \mathbb{R}$

1. Define a model:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

2a. Define a loss function for each example:

$$\mathsf{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$$

2b. Decide how to aggregate the per-example losses:

$$\mathsf{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \mathsf{Loss}(x, y, \mathbf{w})$$

3. Use GD / SGD to compute w:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathsf{TrainLoss}(\mathbf{w})$$

This Week: Key Takeaways

Example: **linear regression**, $\phi(x) \in \mathbb{R}^d, y \in \mathbb{R}$

1. Define a model:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

What is ϕ? How can we get non-linear decision boundaries?

2a. Define a loss function for each example:

$$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$$

2b. Decide how to aggregate the per-example losses:

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \mathbf{w})$$

What are the fairness implications of minimizing the average group loss? What about the maximum group loss?

3. Use GD / SGD to compute w:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$$

Can we break down the steps of taking gradients s.t. we can write algorithms (e.g. backpropagation) to automatically compute gradients for us?

Nonlinear Features

Motivation: Linear Classification

x_1	x_2	$f(x)$
0	2	1
-2	0	1
1	-1	-1

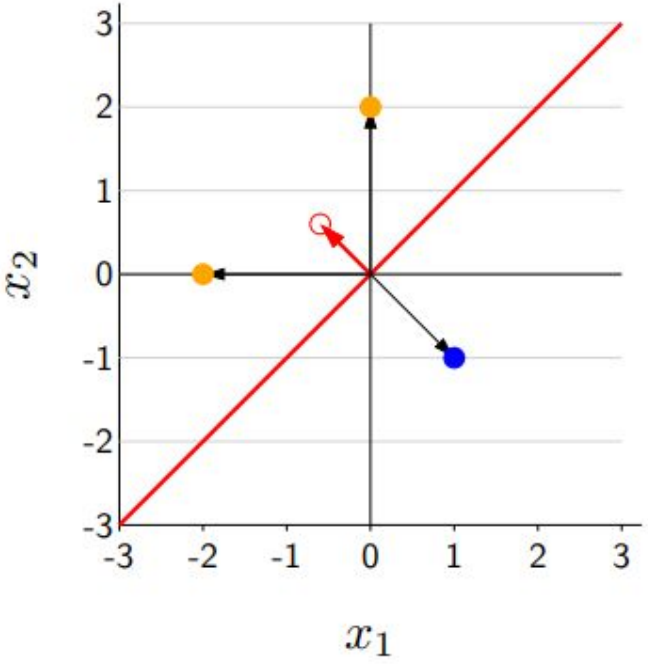
$$f(x) = \text{sign}(\overbrace{[-0.6, 0.6]}^{\mathbf{w}} \cdot \overbrace{[x_1, x_2]}^{\phi(x)})$$

$$f([0,2]) = \text{sign}([-0.6, 0.6] \cdot [0, 2]) = \text{sign}(1.2) = 1$$

$$f([-2,0]) = \text{sign}([-0.6, 0.6] \cdot [-2, 0]) = \text{sign}(1.2) = 1$$

$$f([1,-1]) = \text{sign}([-0.6, 0.6] \cdot [1, -1]) = \text{sign}(-1.2) = -1$$

Decision boundary: x such that $\mathbf{w} \cdot \phi(x) = 0$



Motivation: Linear Classification

x_1	x_2	$f(x)$
0	2	1
-2	0	1
1	-1	-1

$$f(x) = \mathrm{sign}(\overbrace{[-0.6, 0.6]}^{\mathbf{w}} \cdot \overbrace{[x_1, x_2]}^{\phi(x)})$$

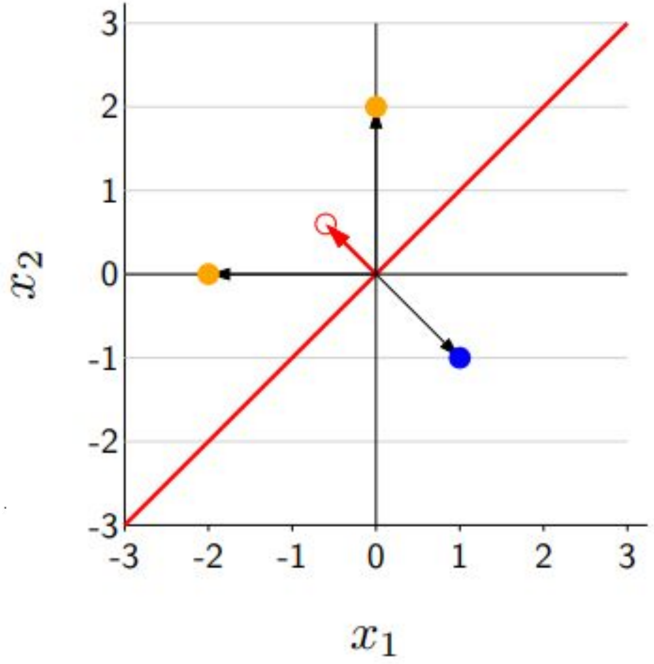
Linear in what?

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

Linear in $\mathbf{w}$? Yes

Linear in $\phi(x)$? Yes

Linear in x? No!



Motivation: Linear Classification

Linear in what?

$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$

Linear in $\mathbf{w}$? Yes

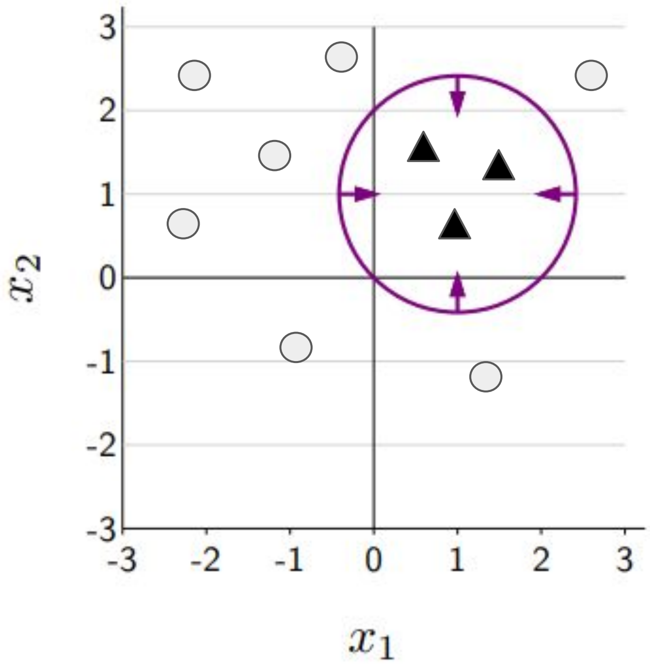
Linear in $\phi(x)$? Yes

Linear in x? No!

$$\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$$
$$f(x) = \text{sign}([2, 2, -1] \cdot \phi(x))$$

Equivalently:

$$f(x) = \begin{cases} 1 & \text{if } \{(x\_1 - 1)^2 + (x\_2 - 1)^2 \leq 2\} \\ -1 & \text{otherwise} \end{cases}$$



The Non-Linear Feature Map can be anything!

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \rightarrow \quad \phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ \cos(x_1) \end{bmatrix}$$

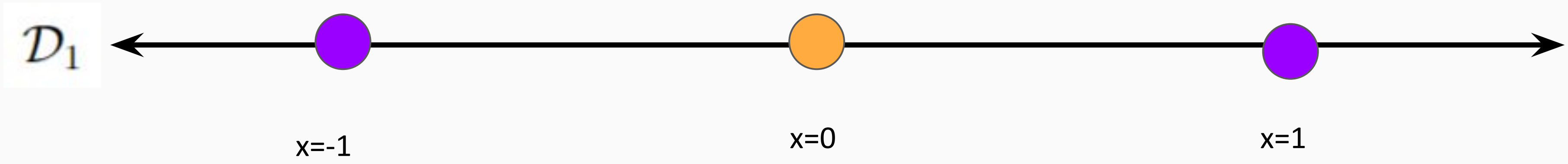
$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \rightarrow \quad \phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ 1\{x_1 < 0\} \end{bmatrix}$$

$x =$  $\in \mathbb{R}^{3 \times W \times H} \quad \rightarrow \quad \phi(x) = \sigma(V_2 \sigma(V_1 x))$

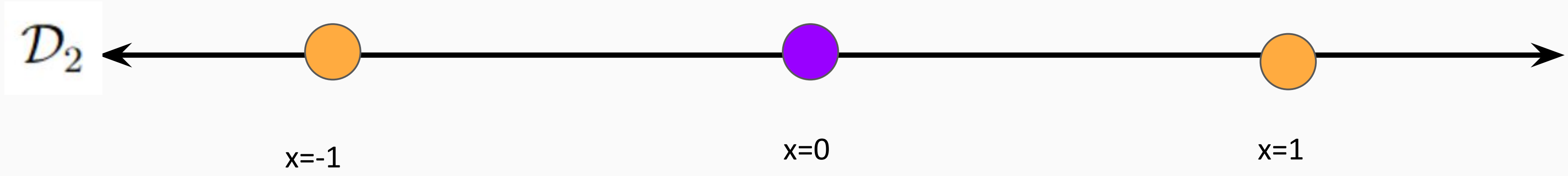
Problem 1:

Legend: y=+1 y=-1

- $\mathcal{D}_1 = \{(-1, +1), (0, -1), (1, +1)\}$.

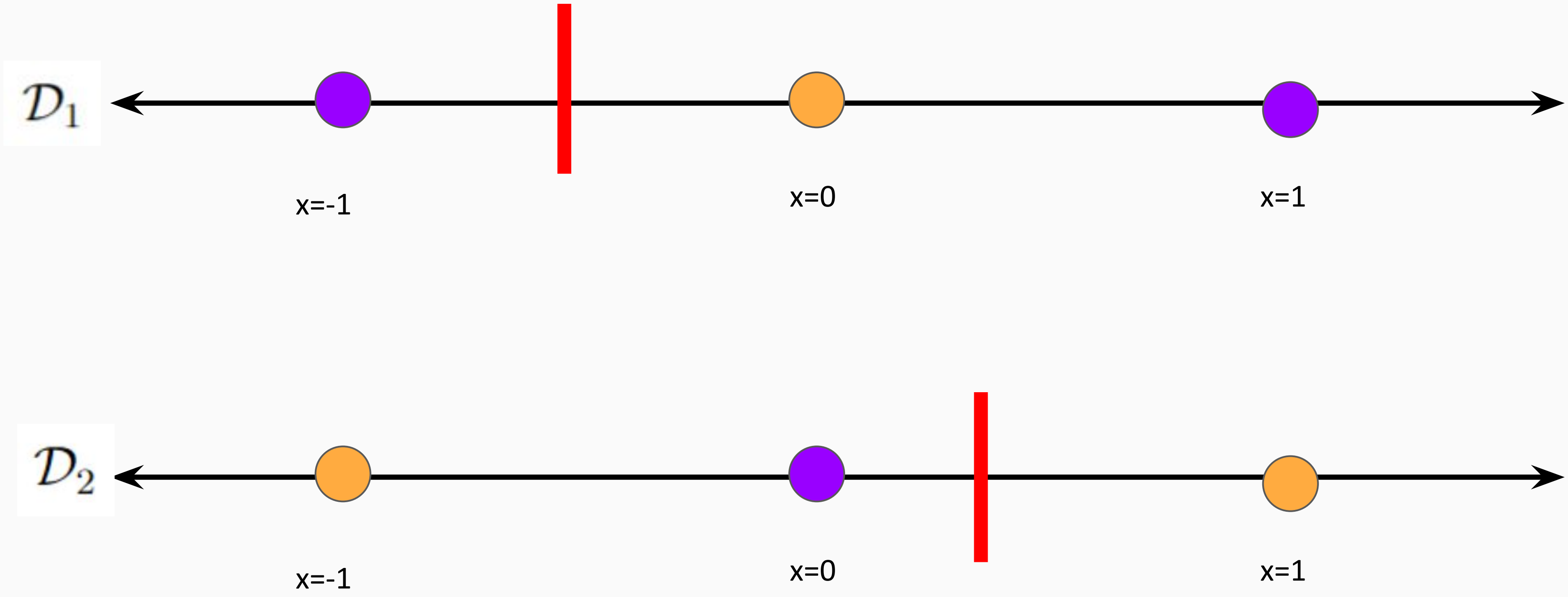


- $\mathcal{D}_2 = \{(-1, -1), (0, +1), (1, -1)\}$.



Try a 1-Dimensional Feature Function

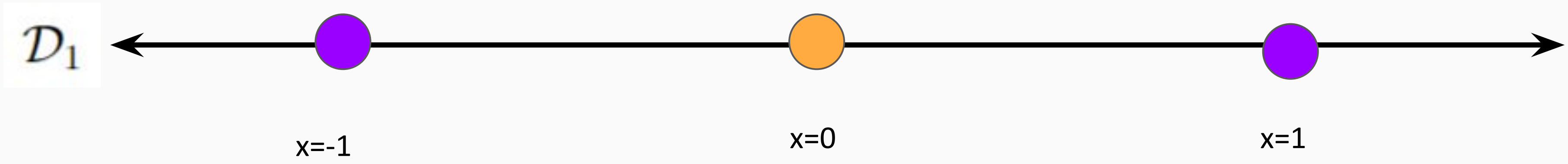
Legend: y=+1 y=-1



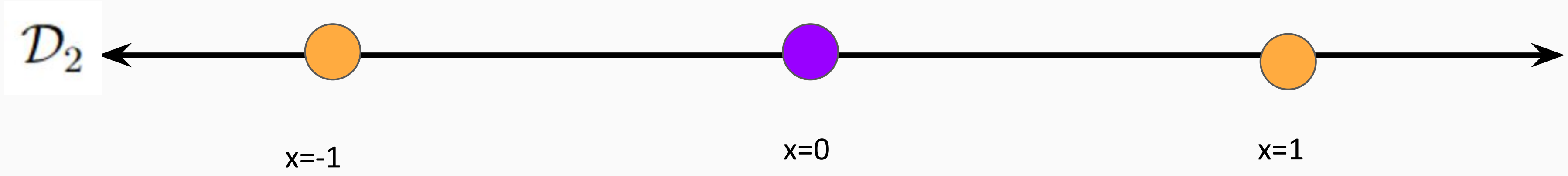
Need at least a Two-Dimensional Feature

Legend: y=+1 y=-1

- $\mathcal{D}_1 = \{(-1, +1), (0, -1), (1, +1)\}$.



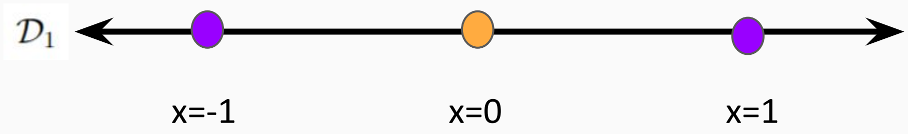
- $\mathcal{D}_2 = \{(-1, -1), (0, +1), (1, -1)\}$.



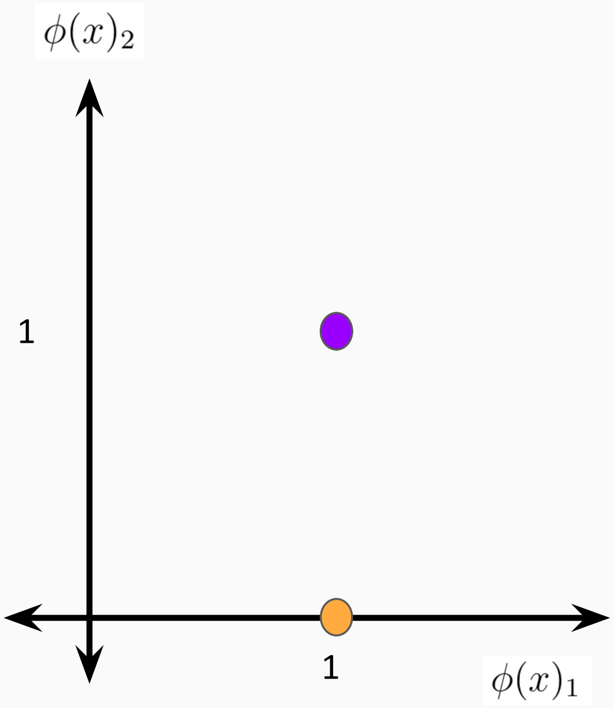
Many Solutions!

Legend: y=+1 y=-1

One option is $\phi(x) = [1, x^2]$, and using $\mathbf{w}_1 = [-1, 2]$ and $\mathbf{w}_2 = [1, -2]$.



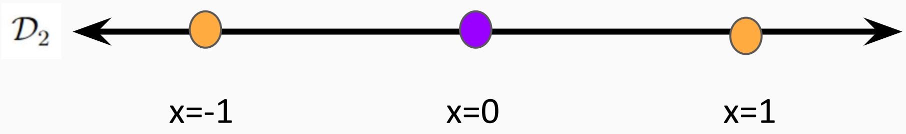
- For $x = -1$, $\mathbf{w}_1 \cdot \phi(x) = [-1, 2] \cdot [1, 1] = 1 > 0$
- For $x = 0$, $\mathbf{w}_1 \cdot \phi(x) = [-1, 2] \cdot [1, 0] = -1 < 0$
- For $x = 1$, $\mathbf{w}_1 \cdot \phi(x) = [-1, 2] \cdot [1, 1] = 1 > 0$



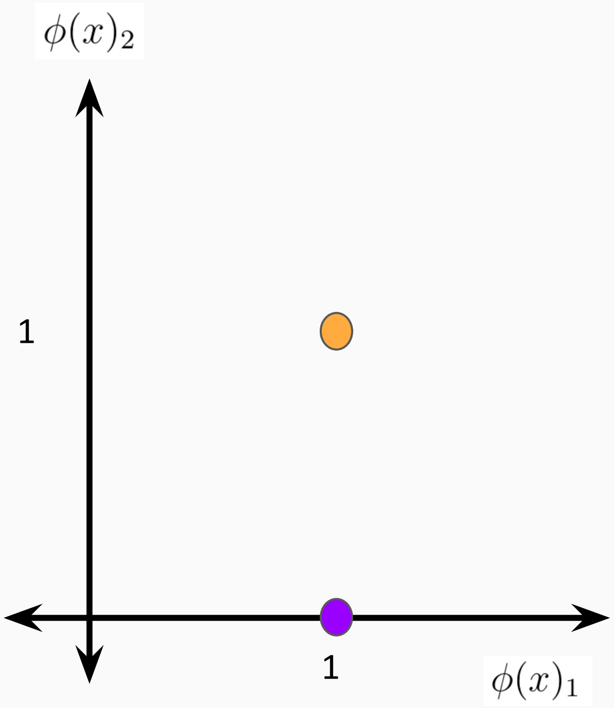
Many Solutions!

Legend: y=+1 y=-1

One option is $\phi(x) = [1, x^2]$, and using $\mathbf{w}_1 = [-1, 2]$ and $\mathbf{w}_2 = [1, -2]$.



- For $x = -1$, $\mathbf{w}_2 \cdot \phi(x) = [1, -2] \cdot [1, 1] = -1 < 0$
- For $x = 0$, $\mathbf{w}_2 \cdot \phi(x) = [1, -2] \cdot [1, 0] = 1 > 0$
- For $x = 1$, $\mathbf{w}_2 \cdot \phi(x) = [1, -2] \cdot [1, 1] = -1 < 0$



Linearly Separating EVERY Dataset – is it possible?

- Given pairs: $(x_1, y_1), ..., (x_n, y_n)$

- Design a feature map $\phi(x)$ such that we can classify

$$(\phi(x_1), y_1), ..., (\phi(x_n), y_n)$$

  for some weight vector w

- **<u>Note: We never said that this has to be a good feature map!</u>**

Linearly Separating EVERY Dataset – is it possible?

$$\phi(x) = \begin{cases} \text{lookup y in training set}(x) & x \in \mathcal{D}_{\text{train}} \\ 0 & \text{o.w.} \end{cases}$$

And use w = 1. This is "memorizing the training set."

Moral of the story: 100% train accuracy != good test accuracy!

More Practice

- The (optional) Problem 4 from this Problem Session
- Problem 1b on the HW 2

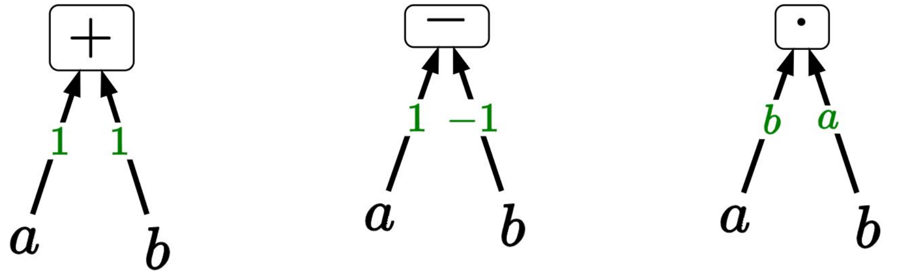
Backpropagation

Drawing a Computational Graph

Preparation:

- Write every function f(a, b, …) as a vertex with one incoming edge per input variable.

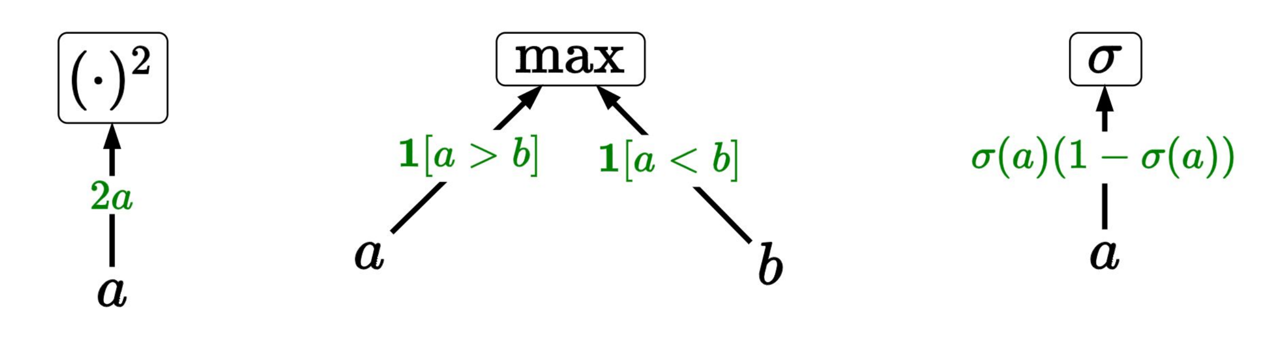
- Write down the gradients (green) of f with respect to each input (a,b) as functions of (a,b) along each respective edge.



Drawing a Computational Graph

Review from Lecture:

$$f(a) = a^2 \quad f(a,b) = \max(a,b) \quad f(a) = \sigma(a)$$



Problem 2

Consider the following function

$$\mathrm{Loss}(x, y, z, w) = 2(xy + \max\{w, z\})$$

Run the backpropagation algorithm to compute the four gradients (each with respect to one of the individual variables) at $x = 3,\ y = -4,\ z = 2$ and $w = -1$. Use the following nodes: addition, multiplication, max, multiplication by a constant.

Problem 2

Step 1: Draw a computational graph for the function:

$$\mathrm{Loss}(x, y, z, w) = 2(xy + \max\{w, z\})$$

Next Step: Forward Pass

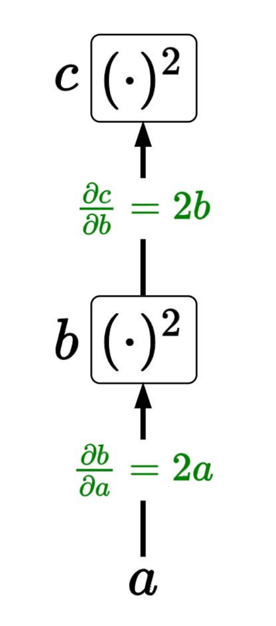
Step 2: After getting your data, flow up the graph and compute values at each function node (in yellow) based on the input(s) from the incoming edge(s).

Example:

c = 81

b = 9

a = 3



Problem 2

Step 2: After getting your data, flow up the graph and compute values at each function node (in yellow) based on the input(s) from the incoming edge(s).

$$\text{Loss}(x, y, z, w) = 2(xy + \max\{w, z\})$$

$$x = 3, \; y = -4, \; z = 2 \text{ and } w = -1.$$

Next Step: Backward Pass for the Gradients

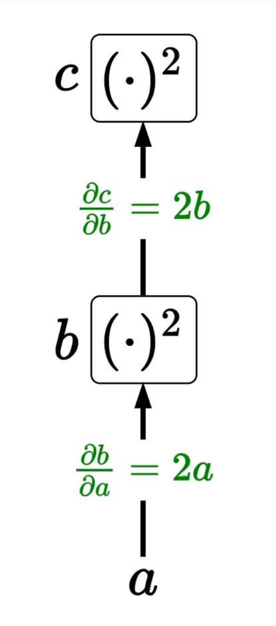
Step 2: Flow down the graph now and compute your gradients (in purple) based on the input(s) from the incoming edge(s) and the chain rule.

Example:

c = 81

b = 9

a = 3



dc/db = 2(9) = 18

dc/da = 18 x db/da = 18 x 2(3) = 18 x 6 = 108

Problem 2

Step 2: Flow down the graph now and compute your gradients (in purple) based on the input(s) from the incoming edge(s) and the chain rule.

$$\text{Loss}(x, y, z, w) = 2(xy + \max\{w, z\})$$

$$x = 3, \; y = -4, \; z = 2 \text{ and } w = -1.$$

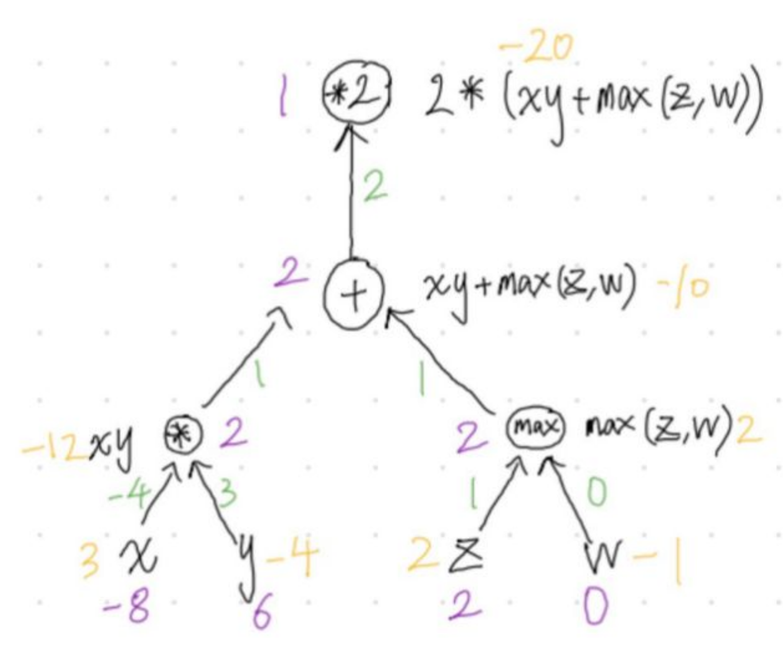
What are each of the following?

- **dLoss/dx**
- **dLoss/dy**
- **dLoss/dw**
- **dLoss/dz**

Problem 2

$$\mathrm{Loss}(x, y, z, w) = 2(xy + \max\{w, z\})$$

$x = 3,\ y = -4,\ z = 2$ and $w = -1.$



K-Means Clustering

K-Means

- K-Means is an unsupervised algorithm to split data points into K clusters.
- We want to minimize the sum of distances from each point to its assigned centroid.

Initialize $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K]$ randomly.

For $t = 1, \ldots, T$:

  Step 1: set assignments $\mathbf{z}$ given $\boldsymbol{\mu}$

    For each point $i = 1, \ldots, n$:

$$z_i \leftarrow \arg\min_{k=1,\ldots,K} \|\phi(x_i) - \mu_k\|^2$$

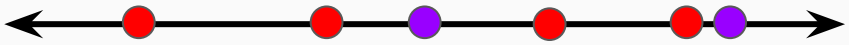
  Step 2: set centroids $\boldsymbol{\mu}$ given $\mathbf{z}$

    For each cluster $k = 1, \ldots, K$:

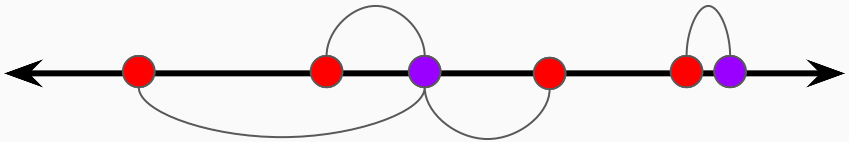
$$\mu_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i:z_i=k} \phi(x_i)$$

Example: 2-Means for 4 data points (in red)

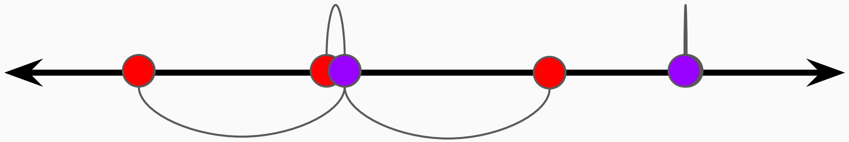
Initialize random centroids (in purple)



For each point, assign to the closest centroid



Move the centroids to min sum of distances



K-Means

- K-Means is an unsupervised algorithm to split data points into K clusters.
- We want to minimize the sum of distances from each point to its assigned centroid.

Initialize $\boldsymbol{\mu} = [\mu_1, \dots, \mu_K]$ randomly.

For $t = 1, \dots, T$:

Step 1: set assignments $\mathbf{z}$ given $\boldsymbol{\mu}$

For each point $i = 1, \dots, n$:

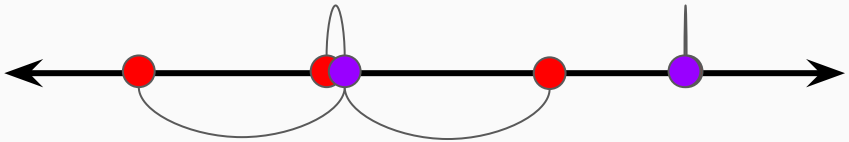
$$z_i \leftarrow \arg\min_{k=1,\dots,K} \|\phi(x_i) - \mu_k\|^2$$

Step 2: set centroids $\boldsymbol{\mu}$ given $\mathbf{z}$

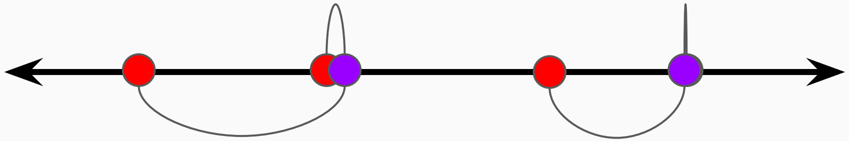
For each cluster $k = 1, \dots, K$:

$$\mu_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i:z_i=k} \phi(x_i)$$

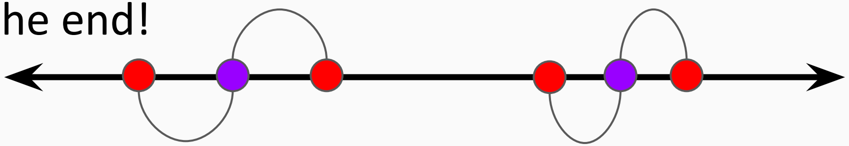
Example: 2-Means for 4 data points (in red)



Repeat until the clusters no longer change



The 3rd point from the left changes clusters in the end!



Problem 3

Consider doing ordinary K-means clustering with $K = 2$ clusters on the following set of 3 one-dimensional points:

$$\{-2, 0, 10\}. \tag{2}$$

Recall that K-means can get stuck in local optima. Describe the precise conditions on the initialization $\mu_1 \in \mathbb{R}$ and $\mu_2 \in \mathbb{R}$ such that running K-means will yield the global optimum of the objective function. Notes:

- Assume that $\mu_1 < \mu_2$.
- Assume that if in step 1 of K-means, no points are assigned to some cluster j, then in step 2, that centroid μ_j is set to ∞.

Things to Consider when doing K-Means

- More clusters = lower loss
  meaning… **<u>if one centroid gets set to infinity and all 3 points end in the same cluster, then we miss the global optimum!</u>**

- So **<u>we need 2 clusters</u>** in the end – which points have to end up in different clusters?

Things to Consider when doing K-Means

- More clusters = lower loss

  meaning… if all 3 points end in the same cluster, then we miss the global optimum!

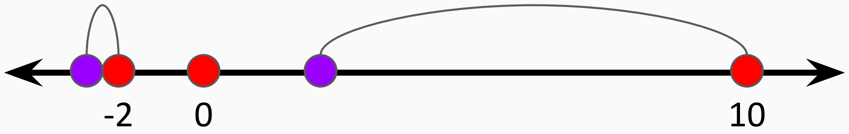
- So we need 2 clusters in the end – which points have to end up in different clusters?

  -2 and 10 must end up in different clusters

  Consider the following (random) clusters

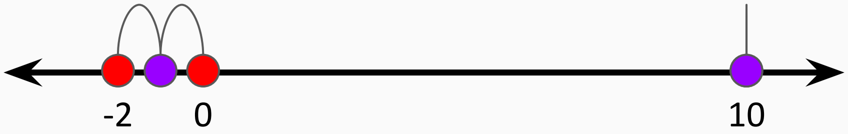
- **What about point 0?**

Problem 3



- **<u>What about point 0?</u>**
- 0 can be assigned to either cluster, and it'll work out!

- Consider what happens if -2 and 0 are assigned to the same cluster…



Cluster1 would end at -1, and Cluster2 would end at 10.

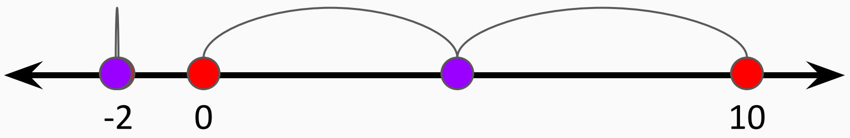
No change would happen on the second pass.

- Consider what happens if 0 and 10 are assigned to the same cluster…

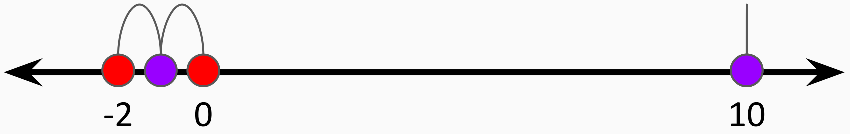
Problem 3

- Consider what happens if 0 and 10 are assigned to the same cluster…

When the clusters move to minimize the sum of distances, we get Cluster1 at -2 and Cluster2 at 5 (halfway between 0 and 10):



But, on the next pass, 0 gets assigned to Cluster1 because it's closer, leading to Cluster1 at -1 and Cluster2 at 10 (the same result as the other case!)



Problem 3

- At the end of the day, the key observation was that -2 and 10 needed to be **assigned to different clusters.**

- This condition on the initial assignments of -2 and 10 can be formally written as

$$|-2-\mu_1| < |-2-\mu_2| \text{ and } |10-\mu_1| > |10-\mu_2|$$

that is, the initial positions of the two clusters have to be such that:
-2 is initially closer to Cluster1 than Cluster2
10 is initially closer to Cluster2 than Cluster1

Final Questions?