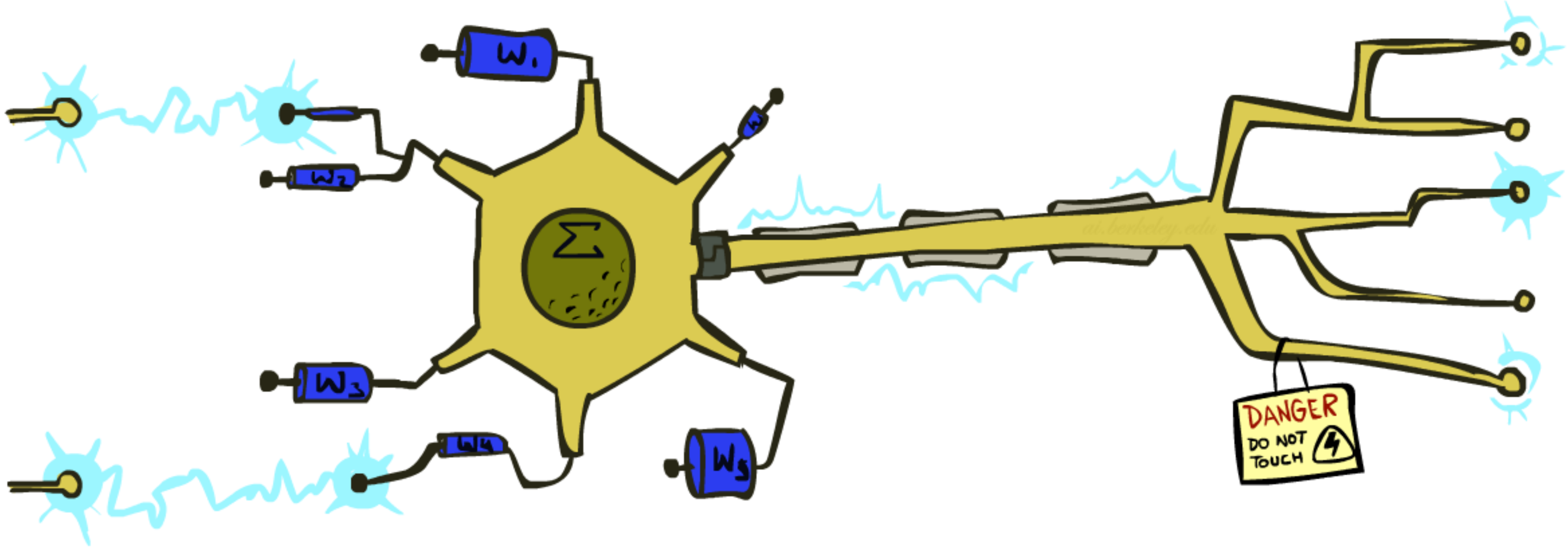
# CS 343: Artificial Intelligence

## Perceptrons and Clustering




Prof. Yuke Zhu — The University of Texas at Austin

[These slides based on those of Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

# Announcements

- Project 6: Classification
  - Released now!
  - Due Wednesday 4/19 at 11:59 pm
- Homework 6: Perceptons, Neural Networks, Gradient Descent
  - Released today! Due Monday 4/17 at 11:59 pm
  - You can start now, but the NN topic will be covered on April 13.
- CTF Contest
  - Qualification deadline: Wednesday 4/12 at 11:59 pm
  - * Now beating the baseline agent is all you need to qualify!
- Guest Lecture next Tuesday: Prof. Bruce Porter
  - Please prepare to attend in person!

# Error-Driven Classification

# Errors, and What to Do

- Examples of errors

```
Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the
latest version of OmniPage Pro, for just $99.99* - the regular
list price is $499! The most common question we've received
about this offer is - Is this genuine? We would like to assure
you that this offer is authorized by ScanSoft, is genuine and
valid. You can get the . . .
```

```
. . . To receive your $30 Amazon.com promotional certificate,
click through to

  http://www.amazon.com/apparel

and see the prominent link for the $30 offer. All details are
there. We hope you enjoyed receiving this message. However, if
you'd rather not receive future e-mails announcing new store
launches, please click . . .
```

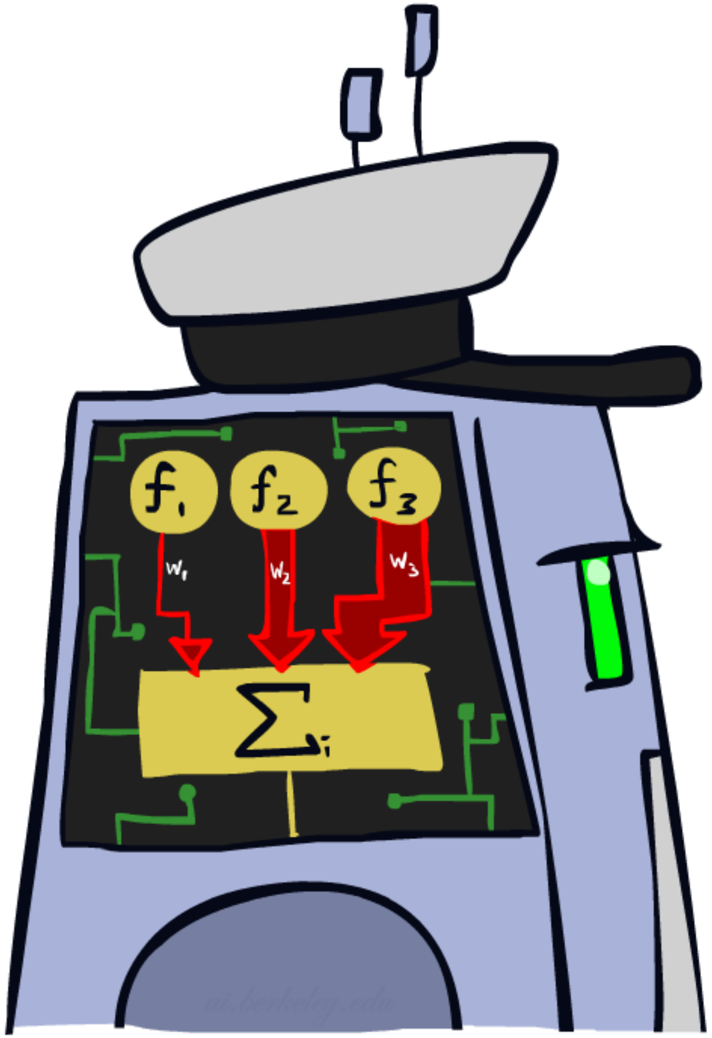
# What to Do About Errors

- Problem: there's still spam in your inbox

- Need more features – words aren't enough!
  - Have you emailed the sender before?
  - Have 1M other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?

- Naïve Bayes models can incorporate a variety of features, but tend to do best when homogeneous (e.g. all features are word occurrences) and/or roughly independent

# Linear Classifiers



# Feature Vectors

$x$ $\quad$ $f(x)$ $\quad$ $y$

```
Hello,

Do you want free printr
cartriges?  Why pay more
when you can get them
ABSOLUTELY FREE!  Just
```

```
# free        : 2
YOUR_NAME     : 0
MISSPELLED    : 2
FROM_FRIEND   : 0
...
```

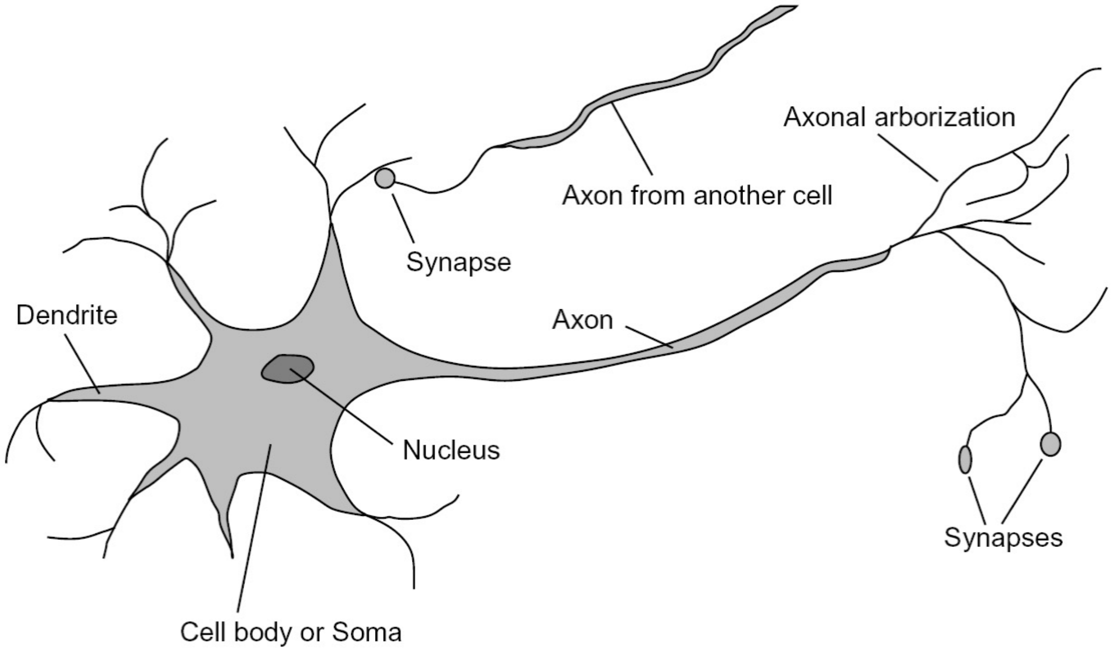
SPAM
or
+

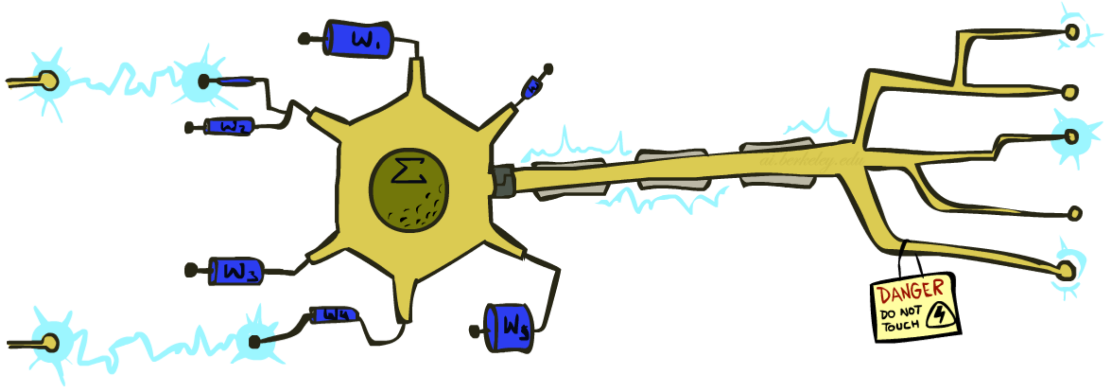
```
PIXEL-7,12    : 1
PIXEL-7,13    : 0
...
NUM_LOOPS     : 1
...
```

“2”

# Some (Simplified) Biology

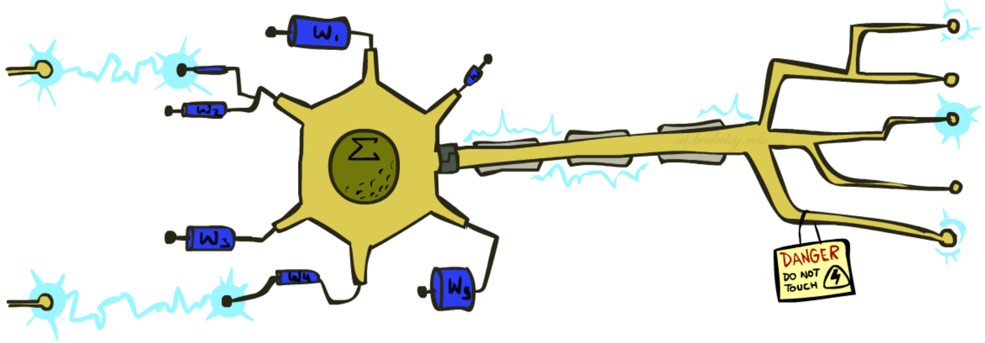
- Very loose inspiration: human neurons



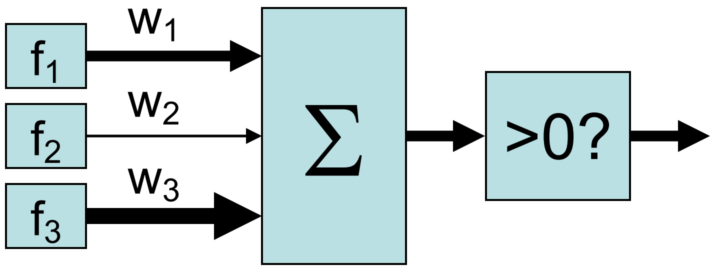
# Linear Classifiers

- Inputs are feature values
- Each feature has a weight
- Sum is the activation




$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- If the activation is:
  - Positive, output +1
  - Negative, output -1

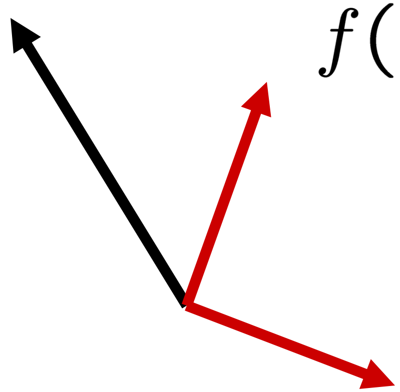


# Weights

- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples

```
# free      : 4
YOUR_NAME   :-1
MISSPELLED  : 1
FROM_FRIEND :-3
...
```

$w$



$f(x_1)$

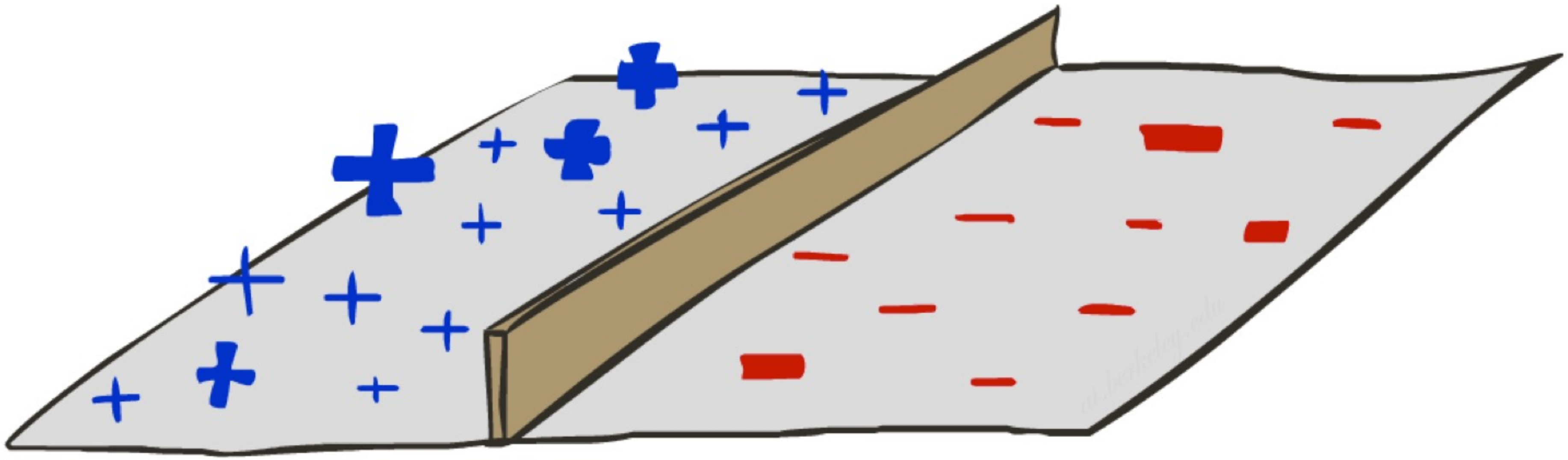
```
# free      : 2
YOUR_NAME   : 0
MISSPELLED  : 2
FROM_FRIEND : 0
...
```

$f(x_2)$

```
# free      : 0
YOUR_NAME   : 1
MISSPELLED  : 1
FROM_FRIEND : 1
...
```

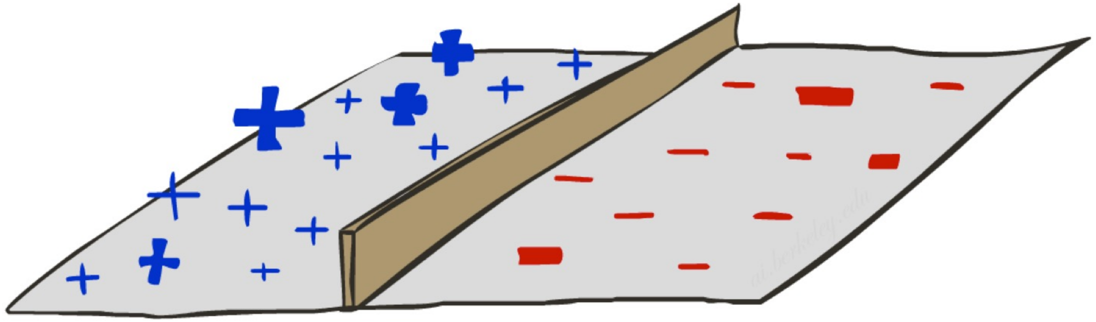
*Dot product* $w \cdot f$ *positive means the positive class*

# Decision Rules



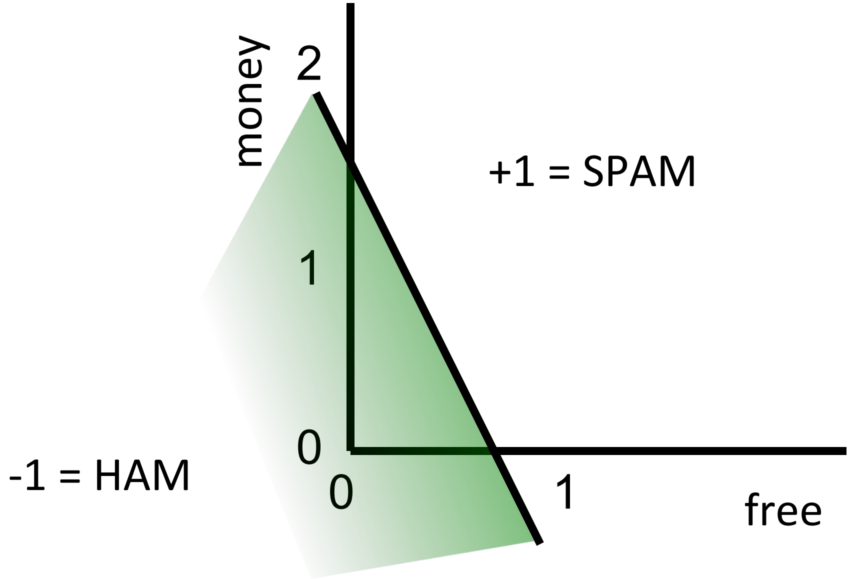
# Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



$$w$$

```
BIAS  : -3
free  :  4
money :  2
...
```

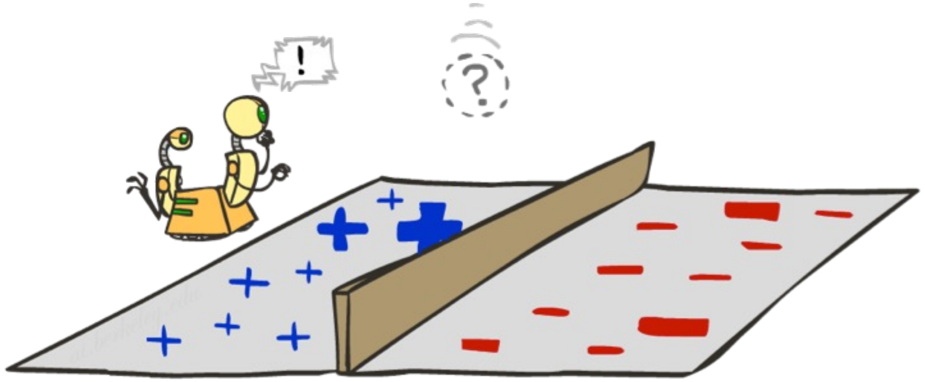



$$f \cdot w = 0$$

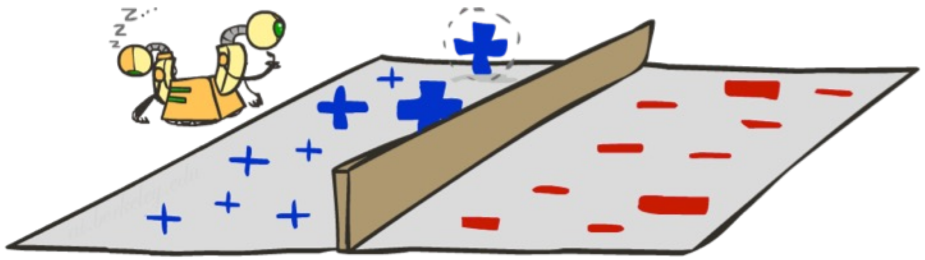
# Weight Updates

# Learning: Binary Perceptron

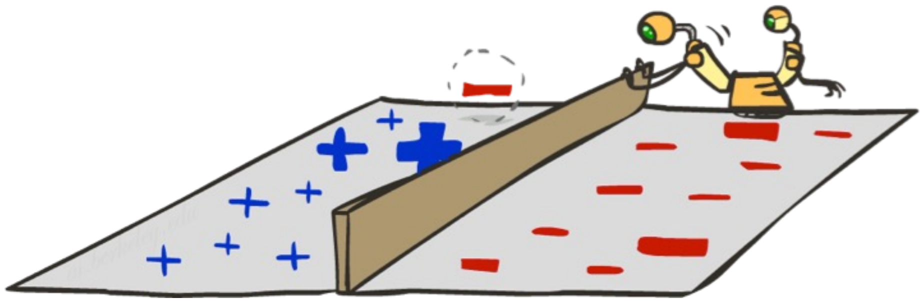
- Start with weights = 0
- For each training instance:
  - Classify with current weights




  - If correct (i.e., y=y*), no change!



  - If wrong: adjust the weight vector



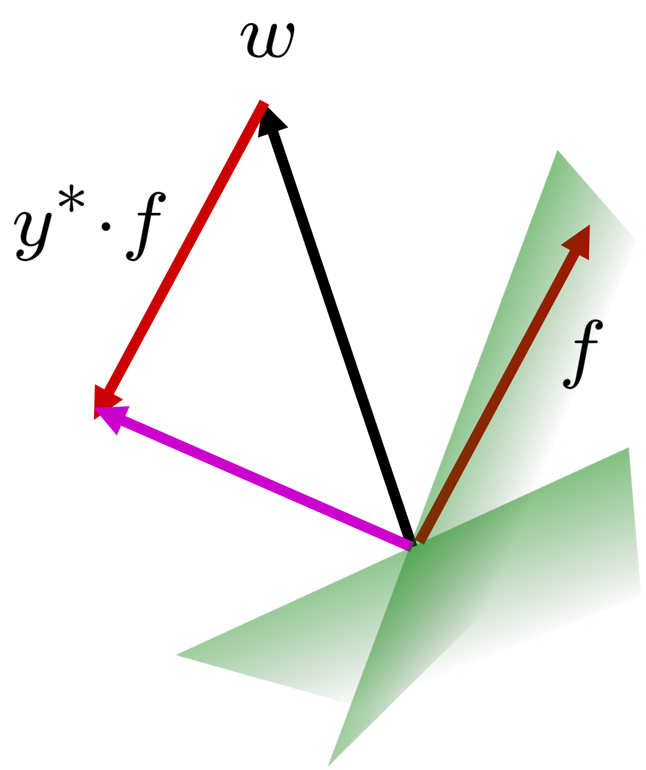
# Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights

$$y = \begin{cases} +1 & \text{if } \; w \cdot f(x) \geq 0 \\ -1 & \text{if } \; w \cdot f(x) < 0 \end{cases}$$

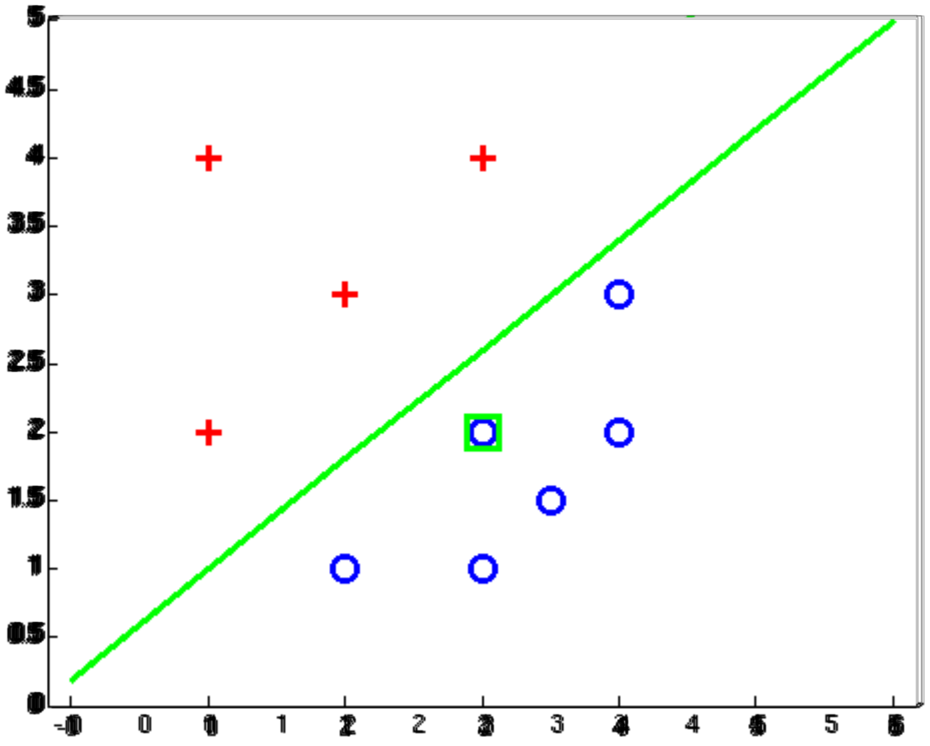
  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y* is -1.

$$w = w + y^* \cdot f$$



# Examples: Perceptron

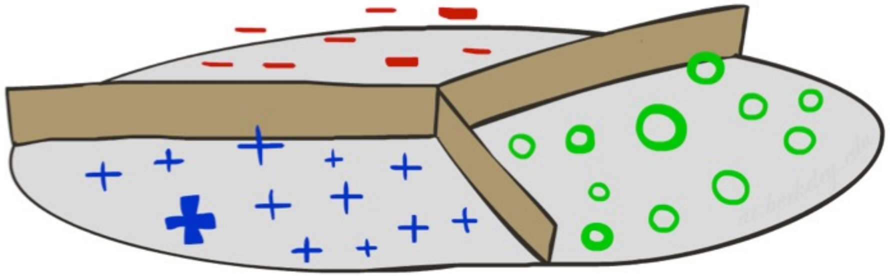
- Separable Case

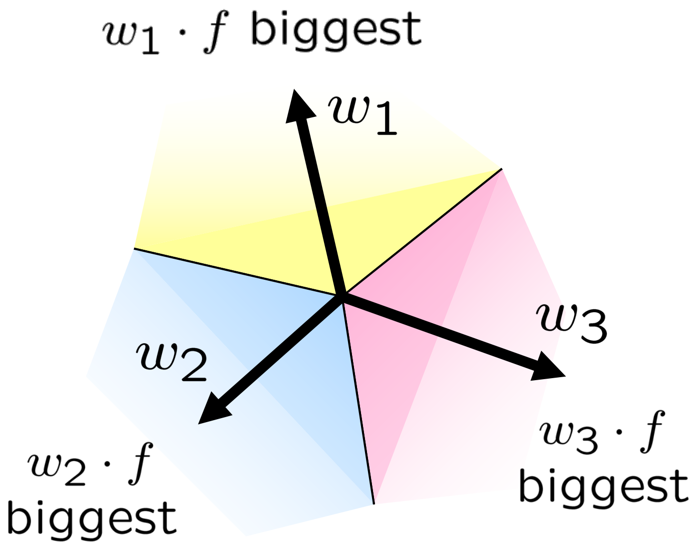


# Multiclass Decision Rule

- If we have multiple classes:
  - A weight vector for each class: $w_y$
  - Score (activation) of a class y: $w_y \cdot f(x)$
  - Prediction highest score wins

$$y = \arg\max_y \; w_y \cdot f(x)$$






*Binary = multiclass where the negative class has weight zero*

# Learning: Multiclass Perceptron

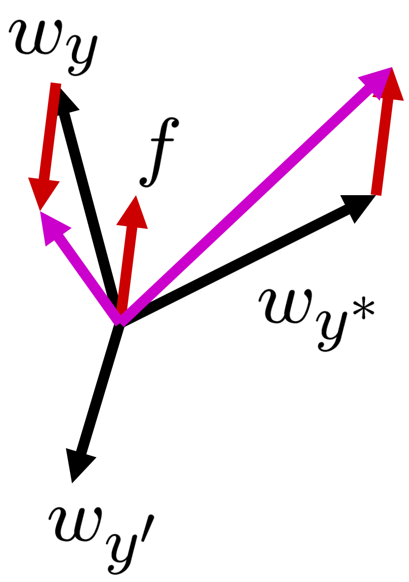
- Start with all weights = 0
- Pick up training examples one by one
- Predict with current weights

$$y = \arg\max_y \; w_y \cdot f(x)$$

- If correct, no change!
- If wrong: lower score of wrong answer, raise score of right answer

$$w_y = w_y - f(x)$$

$$w_{y^*} = w_{y^*} + f(x)$$

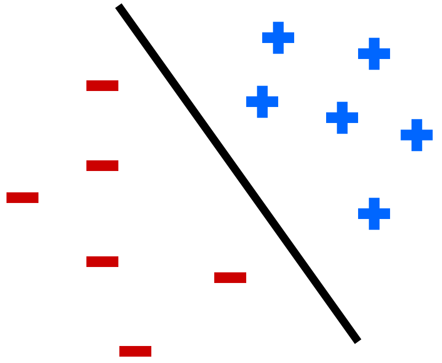


# Properties of Perceptrons

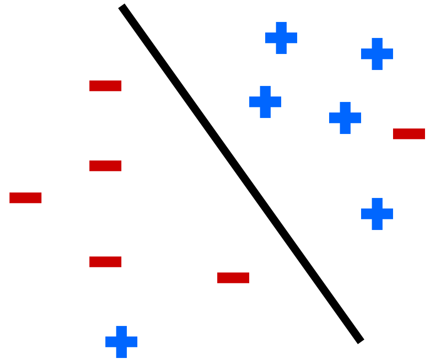
- Separability: true if some parameters get the training set perfectly correct
- Convergence: if the training is separable, perceptron will eventually converge (binary case)
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

$$\text{mistakes} < \frac{k}{\delta^2}$$

Separable



Non-Separable

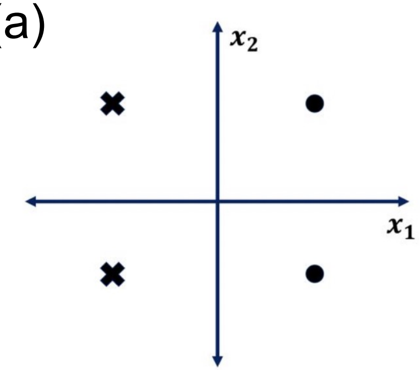


# Perceptron Exercises

For each of the datasets represented by the graphs below, please select the feature maps for which the perceptron algorithm can perfectly classify the data.

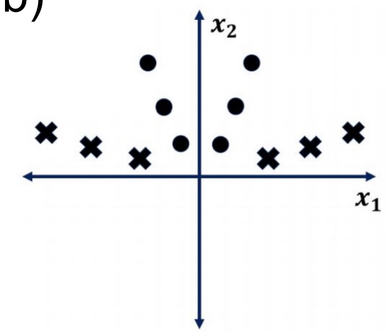
Each data point is in the form ($x_1$, $x_2$), and has some label Y, which is either a 1 (dot) or −1 (cross).

(a)



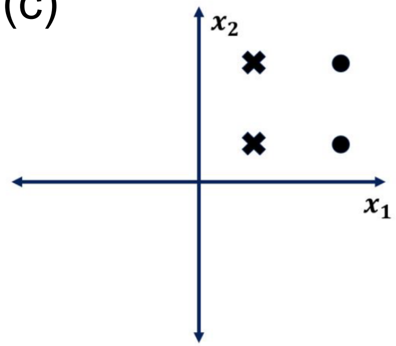

- ☐ $[x_1 \quad x_2 \quad 1]$
- ☐ $[x_1 \quad x_2 \quad x_1^2]$
- ☐ $[x_1 \quad x_2 \quad |x_1|]$
- ☐ $[x_1 \quad x_2 \quad Y]$
- ☐ $[x_1 \quad x_2]$

(b)




- ☐ $[x_1 \quad x_2 \quad 1]$
- ☐ $[x_1 \quad x_2 \quad x_1^2]$
- ☐ $[x_1 \quad x_2 \quad |x_1|]$
- ☐ $[x_1 \quad x_2 \quad Y]$
- ☐ $[x_1 \quad x_2]$

(c)



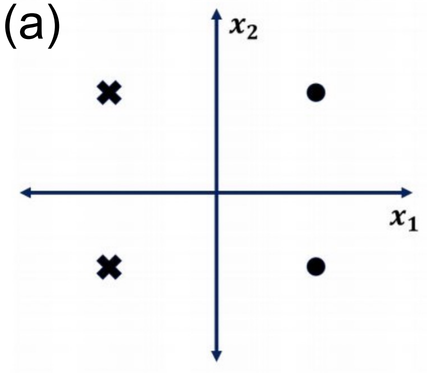

- ☐ $[x_1 \quad x_2 \quad 1]$
- ☐ $[x_1 \quad x_2 \quad x_1^2]$
- ☐ $[x_1 \quad x_2 \quad |x_1|]$
- ☐ $[x_1 \quad x_2 \quad Y]$
- ☐ $[x_1 \quad x_2]$

# Perceptron Exercises

For each of the datasets represented by the graphs below, please select the feature maps for which the perceptron algorithm can perfectly classify the data.

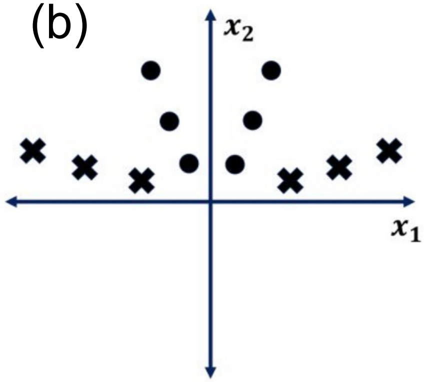
Each data point is in the form $(x_1, x_2)$, and has some label Y, which is either a 1 (dot) or −1 (cross).

(a)



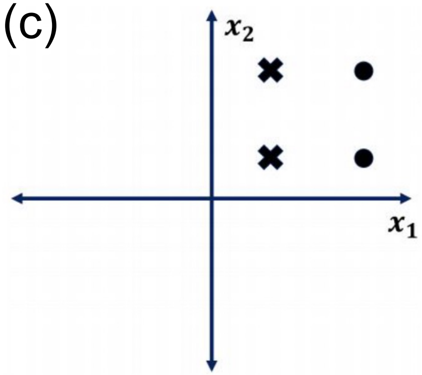

- ■ $[x_1 \quad x_2 \quad 1]$
- ■ $[x_1 \quad x_2 \quad x_1^2]$
- ■ $[x_1 \quad x_2 \quad |x_1|]$
- ■ $[x_1 \quad x_2 \quad Y]$
- ■ $[x_1 \quad x_2]$

(b)




- □ $[x_1 \quad x_2 \quad 1]$
- ■ $[x_1 \quad x_2 \quad x_1^2]$
- ■ $[x_1 \quad x_2 \quad |x_1|]$
- ■ $[x_1 \quad x_2 \quad Y]$
- □ $[x_1 \quad x_2]$

(c)




- ■ $[x_1 \quad x_2 \quad 1]$
- ■ $[x_1 \quad x_2 \quad x_1^2]$
- □ $[x_1 \quad x_2 \quad |x_1|]$
- ■ $[x_1 \quad x_2 \quad Y]$
- □ $[x_1 \quad x_2]$

# Recap: Classification

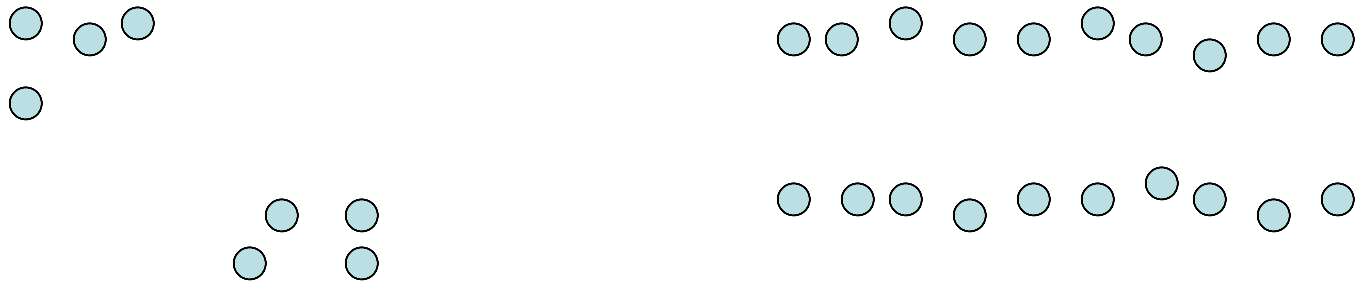
- Classification systems:
  - Supervised learning
  - Make a prediction given evidence
  - We’ve seen several methods for this
  - Useful when you have labeled data

# Clustering

- Clustering systems:
  - Unsupervised learning
  - Detect patterns in unlabeled data
    - E.g. group emails or search results
    - E.g. find categories of customers
    - E.g. detect anomalous program executions
  - Useful when don't know what you're looking for
  - Requires data, but no labels
  - Often get gibberish

# Clustering

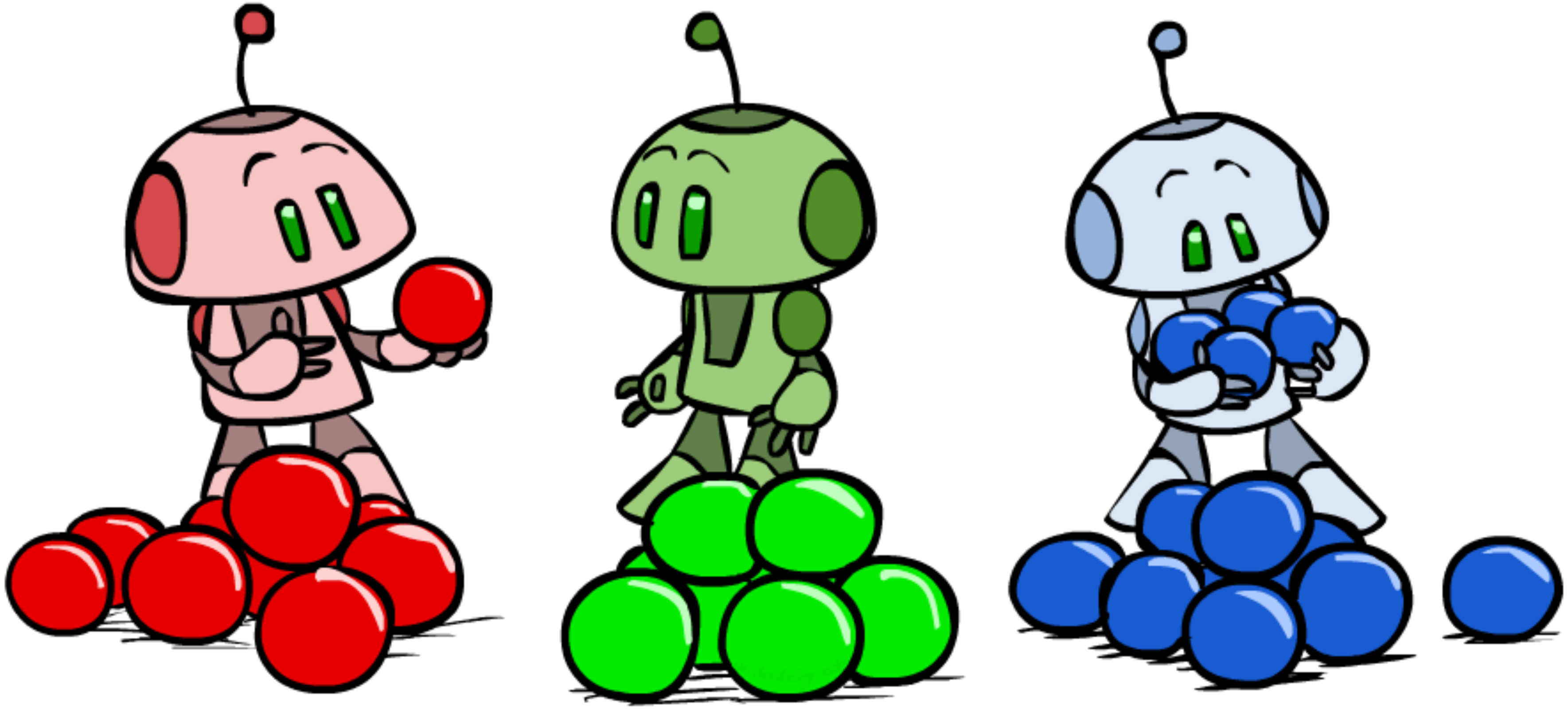
- Basic idea: group together similar instances
- Example: 2D point patterns



- What could "similar" mean?
  - One option: small (squared) Euclidean distance

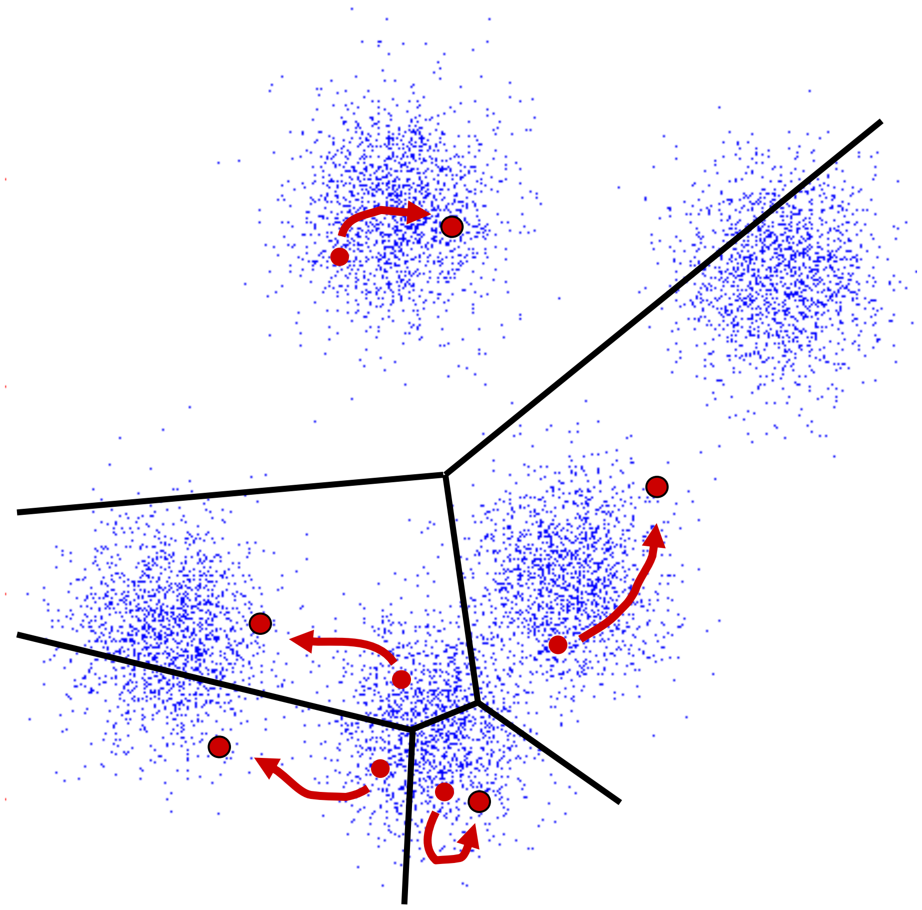
$$\text{dist}(x, y) = (x-y)^\top(x-y) = \sum_i (x_i - y_i)^2$$

# K-Means

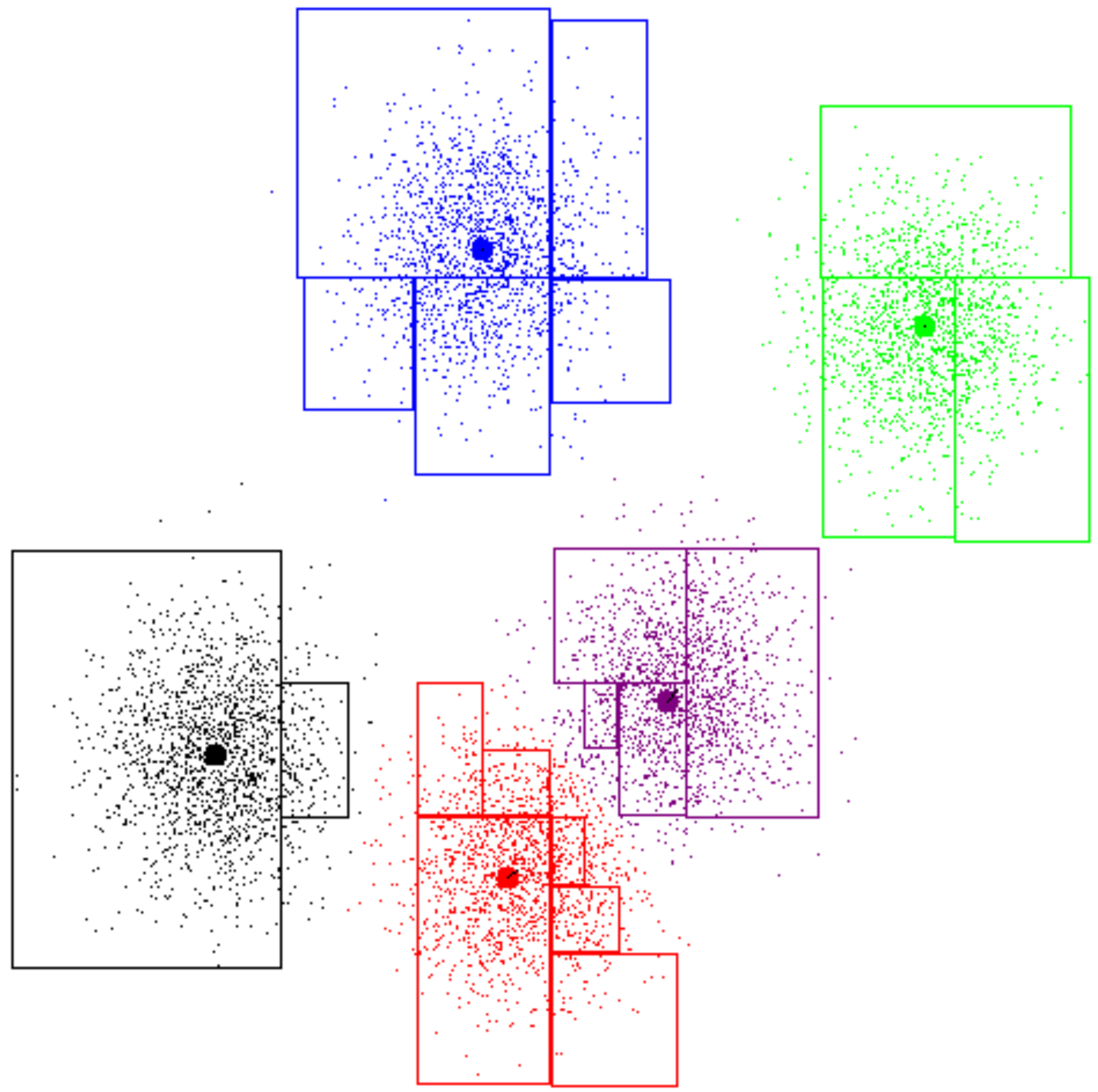


# K-Means

- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Alternate:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
  - Stop when no points' assignments change



# K-Means Example



# K-Means as Optimization

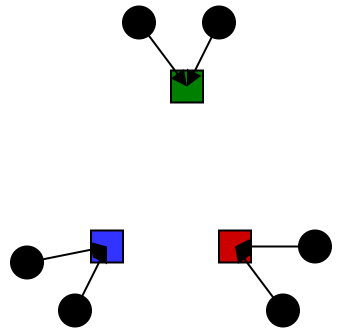
- Consider the total distance to the means:

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

points

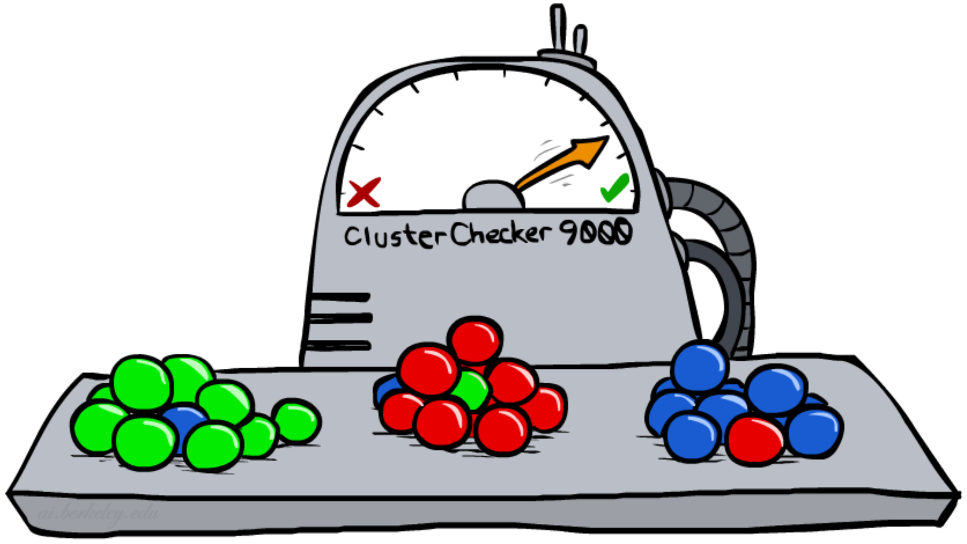
means

assignments



- Each iteration reduces phi

- Two stages each iteration:
  - Update assignments: fix means c, change assignments a
  - Update means: fix assignments a, change means c



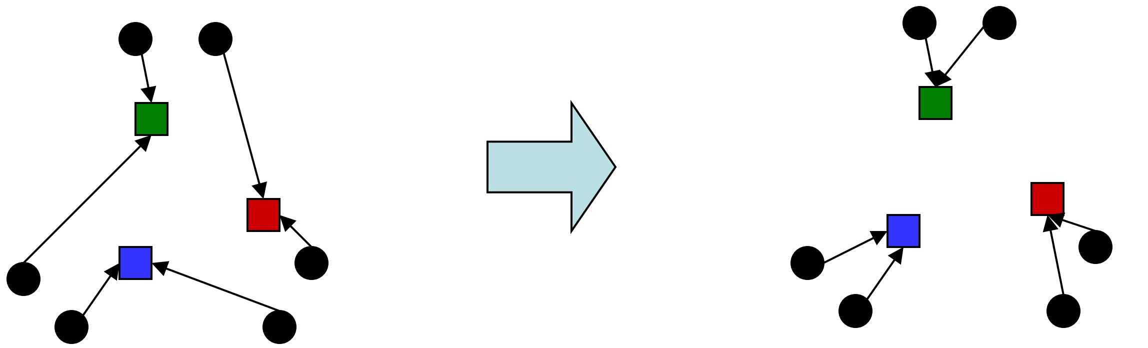
# Phase I: Update Assignments

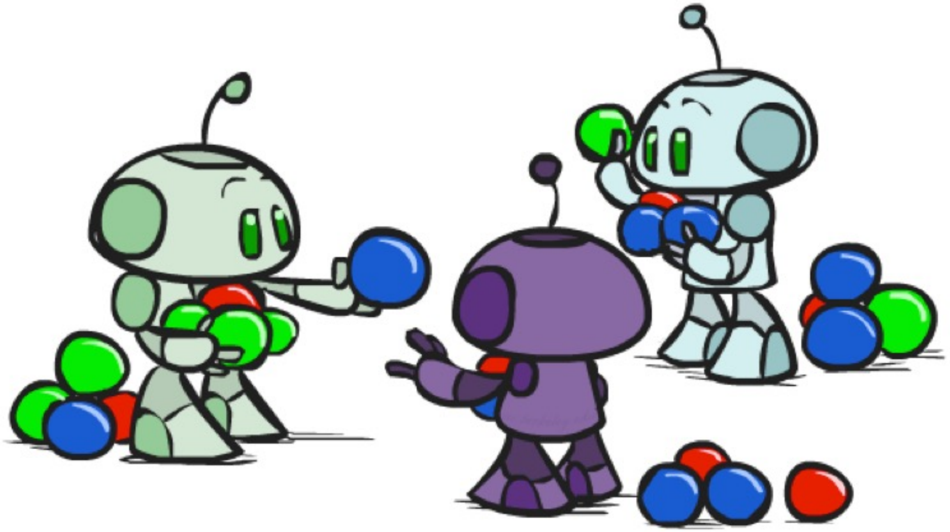
- For each point, re-assign to closest mean:

$$a_i = \underset{k}{\mathrm{argmin}}\, \mathrm{dist}(x_i, c_k)$$

- Can only decrease total distance phi!

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \mathrm{dist}(x_i, c_{a_i})$$

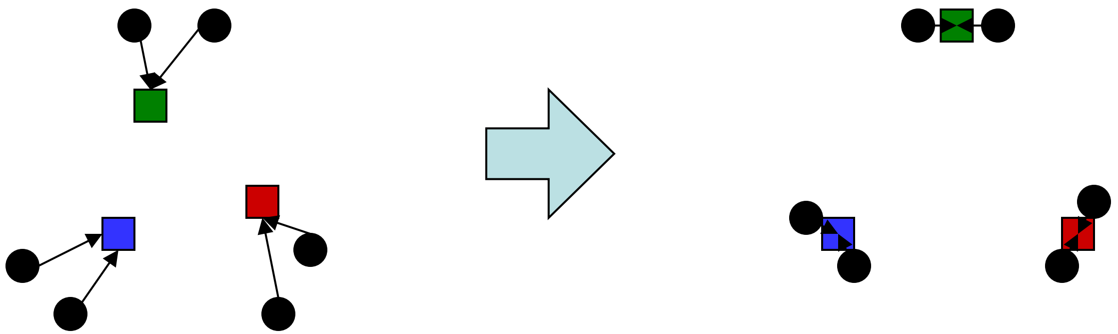




# Phase II: Update Means

- Move each mean to the average of its assigned points:

$$c_k = \frac{1}{|\{i : a_i = k\}|} \sum_{i:a_i=k} x_i$$

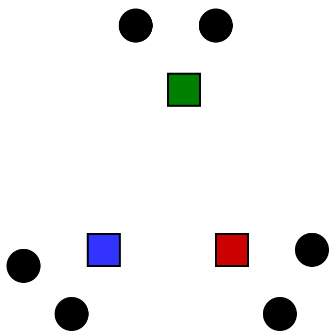


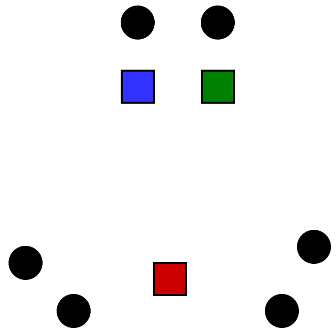
- Also can only decrease total distance… (Why?)

- Fun fact: the point y with minimum squared Euclidean distance to a set of points {x} is their mean

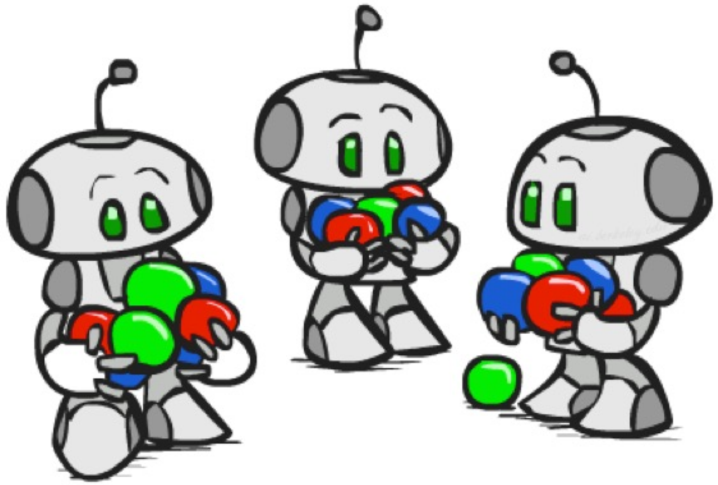
# Initialization

- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?



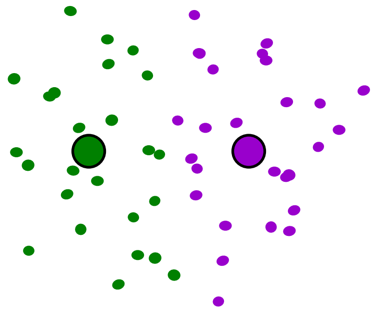


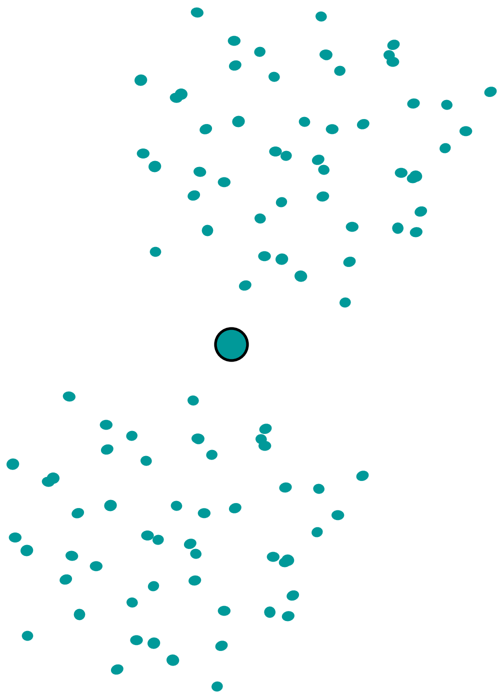
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics



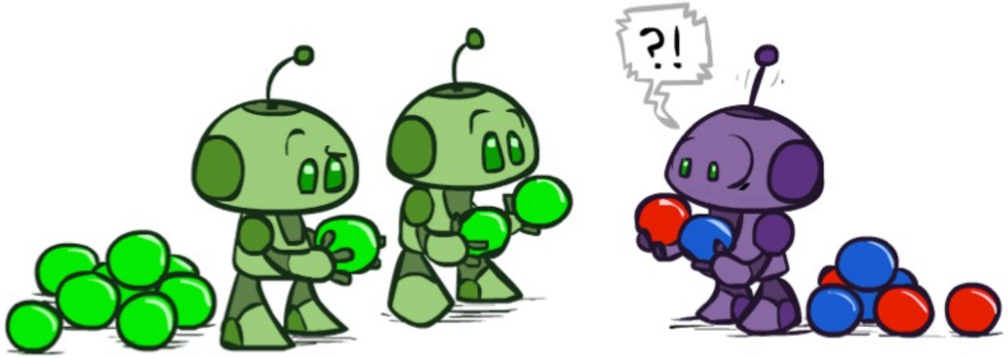
# K-Means Getting Stuck

- A local optimum:



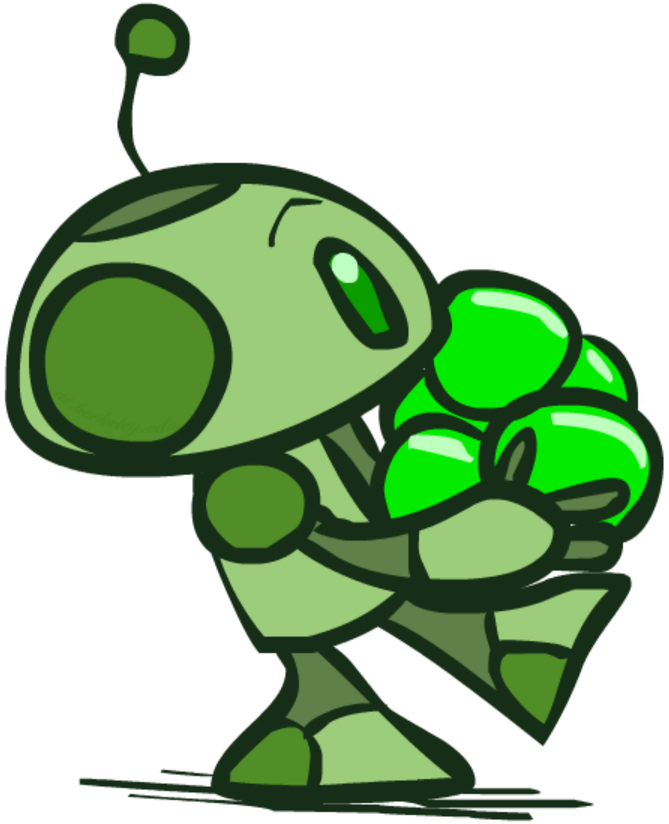


*Why doesn't this work out like the earlier example, with the purple taking over half the blue?*



# K-Means Questions

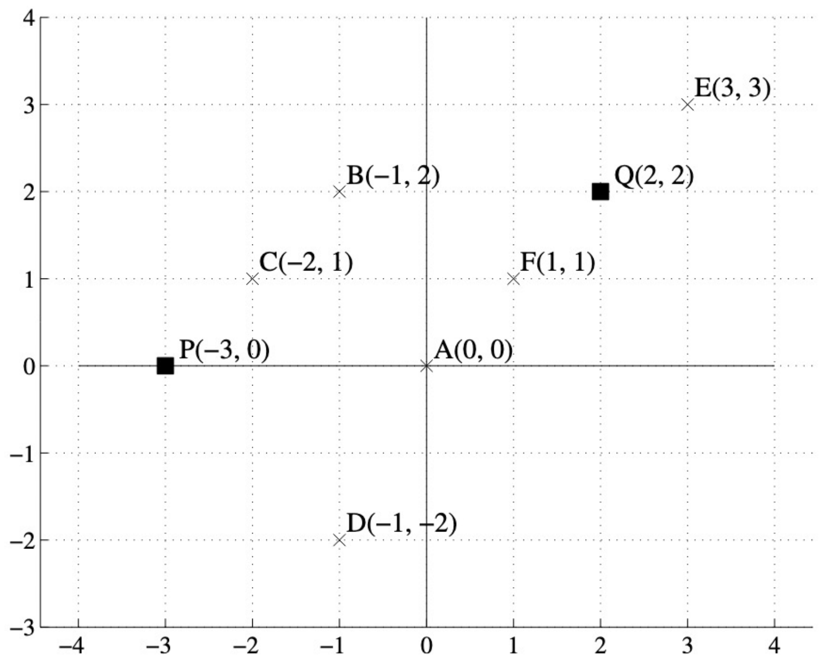
- Will K-means converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
  - If the patterns are very very clear?
- Will it find something interesting?
- Do people ever use it?
- How many clusters to pick?



# K-Means Exercises

In this question, we will do $k$-means clustering to cluster the points $A, B \ldots F$ (indicated by $\times$'s in the figure on the right) into 2 clusters. The current cluster centers are $P$ and $Q$ (indicated by the ■ in the diagram on the right). Recall that $k$-means requires a distance function. Given 2 points, $A = (A_1, A_2)$ and $B = (B_1, B_2)$, we use the following distance function $d(A, B)$ that you saw from class,

$$d(A, B) = (A_1 - B_1)^2 + (A_2 - B_2)^2$$




**(a)** [2 pts] **Update assignment step:** Select all points that get assigned to the cluster with center at $P$:

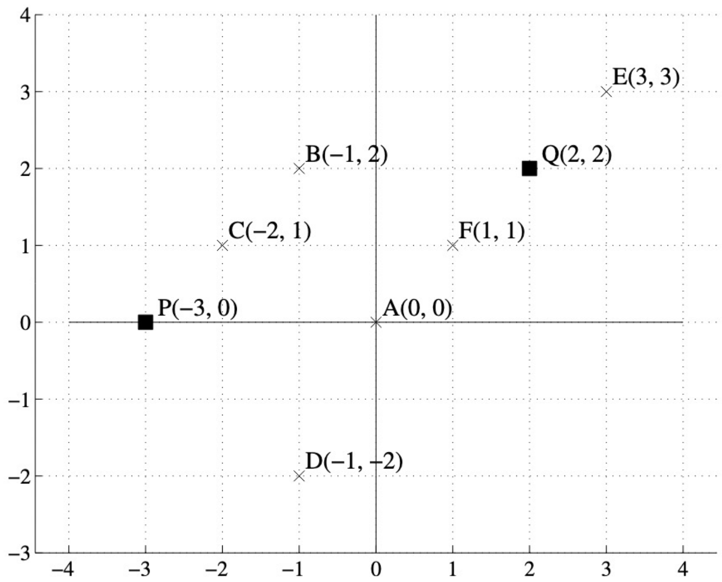
○ $A$ ○ $B$ ○ $C$ ○ $D$ ○ $E$ ○ $F$ ○ No point gets assigned to cluster P

**(b)** [2 pts] **Update cluster center step:** What does cluster center $P$ get updated to?

# K-Means Exercises

In this question, we will do $k$-means clustering to cluster the points $A, B \ldots F$ (indicated by $\times$'s in the figure on the right) into 2 clusters. The current cluster centers are $P$ and $Q$ (indicated by the ■ in the diagram on the right). Recall that $k$-means requires a distance function. Given 2 points, $A = (A_1, A_2)$ and $B = (B_1, B_2)$, we use the following distance function $d(A, B)$ that you saw from class,

$$d(A, B) = (A_1 - B_1)^2 + (A_2 - B_2)^2$$




**(a)** [2 pts] **Update assignment step:** Select all points that get assigned to the cluster with center at $P$:

○ $A$ ● $B$ ● $C$ ● $D$ ○ $E$ ○ $F$ ○ No point gets assigned to cluster P

**(b)** [2 pts] **Update cluster center step:** What does cluster center $P$ get updated to?

The cluster center gets updated to the point, $P'$ which minimizes, $d(P', B) + d(P', C) + d(P', D)$, which in this case turns out to be the centroid of the points, hence the new cluster center is

$$\left(\frac{-1-2-1}{3}, \frac{2+1-2}{3}\right) = \left(\frac{-4}{3}, \frac{+1}{3}\right)$$