CPSC 340: Machine Learning and Data Mining

K-Means Clustering

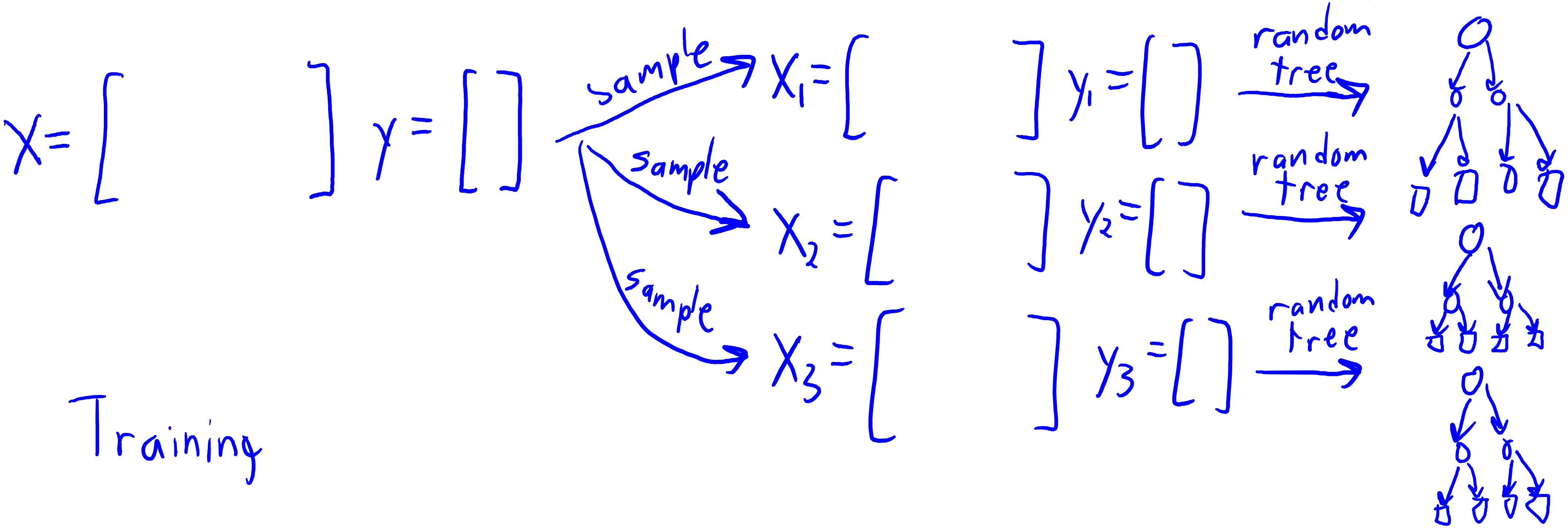
Fall 2017

Admin

- Assignment 1 is due next Friday.
  - Needs Julie 0.6 (you can use JuliaBox if you can’t get Julia/PyPlot working).
  - There was a bug in the decision tree predict function.
  - There was a minor error in the example_knn.jl function.

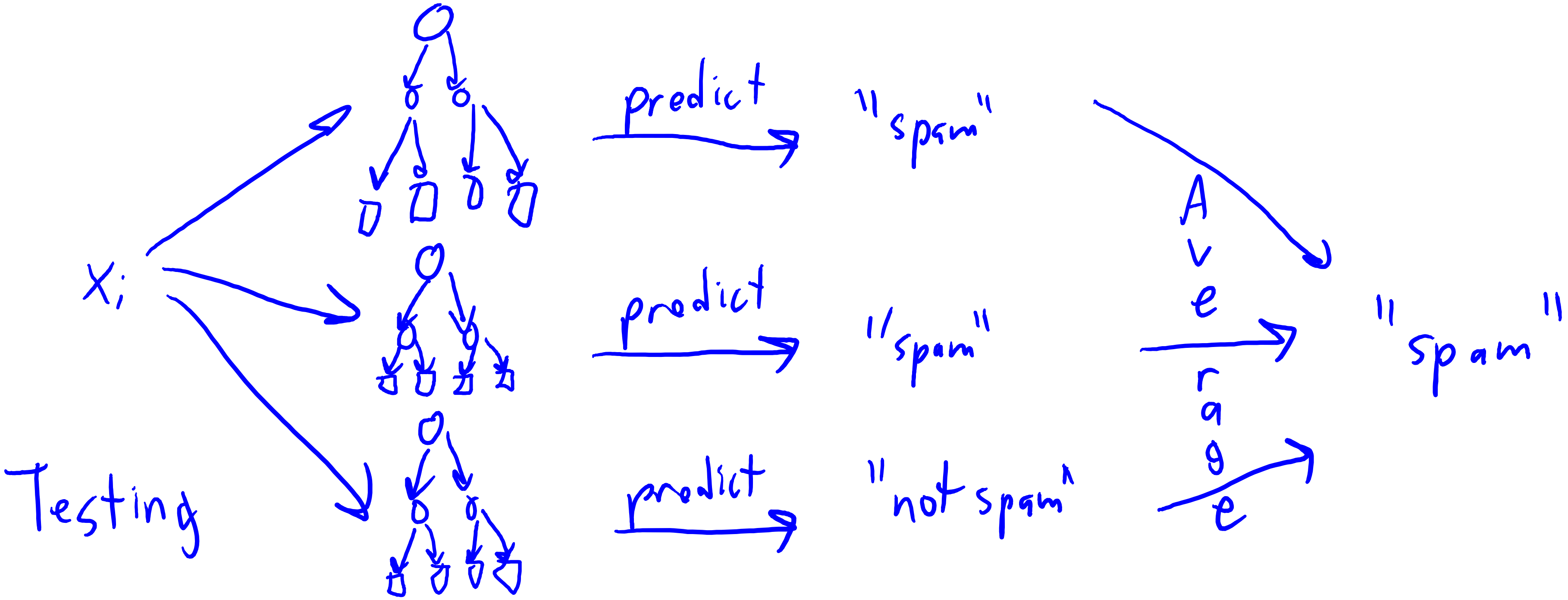
Random Forests

- Random forests are one of the best ‘out of the box’ classifiers.
- Fit deep decision trees to random bootstrap samples of data, base splits on random subsets of the features, and classify using mode.



Random Forests

- Random forests are one of the best ‘out of the box’ classifiers.
- Fit deep decision trees to random bootstrap samples of data, base splits on random subsets of the features, and classify using mode.



End of Part 1: Key Concepts

- Fundamental ideas:
  - Training vs. test error (memorization vs. learning).
  - IID assumption (examples come independently from same distribution).
  - Golden rule of ML (test set should not influence training).
  - Fundamental trade-off (between training error vs. approximation error).
  - Validation sets and cross-validation (can approximate test error)
  - Optimization bias (we can overfit the training set and the validation set).
  - Decision theory (we should consider costs of predictions).
  - Parametric vs. non-parametric (whether model size depends on ‘n’).
  - No free lunch theorem (there is no “best” model).

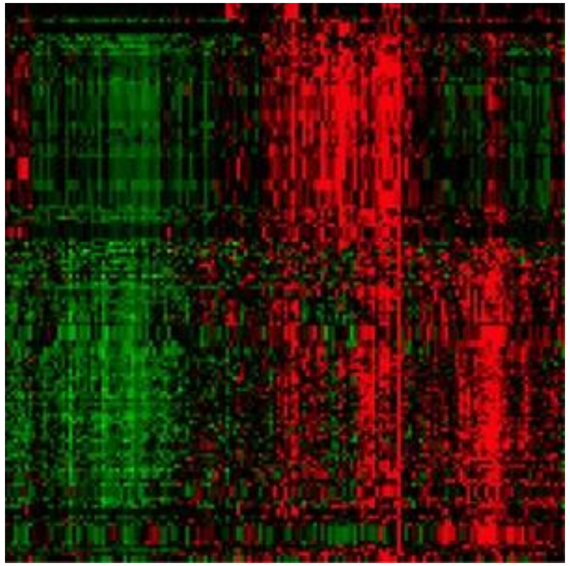
End of Part 1: Key Concepts

- We saw 3 ways of “learning”:
  - Searching for rules.
    - Decision trees (greedy recursive splitting using decision stumps).
  - Counting frequencies.
    - Naïve Bayes (probabilistic classifier based on conditional independence).
  - Measuring distances.
    - K-nearest neigbbours (non-parametric classifier with universal consistency).
- We saw 2 generic ways of improving performance:
  - Encouraging invariances with data augmentation.
  - Ensemble methods (combine predictions of several models).
    - Random forests (averaging plus randomization to reduce overfitting).

Application: Classifying Cancer Types

- "I collected gene expression data for 1000 different types of cancer cells, can you tell me the different classes of cancer?"

X =



- We are not given the class labels y, but want meaningful labels.
- An example of unsupervised learning.

https://corelifesciences.com/human-long-non-coding-rna-expression-microarray-service.html

Unsupervised Learning

- Supervised learning:
  - We have features x_i and class labels y_i.
  - Write a program that produces y_i from x_i.
- Unsupervised learning:
  - We only have x_i values, but no explicit target labels.
  - You want to do "something" with them.
- Some unsupervised learning tasks:
  - Outlier detection: Is this a 'normal' x_i?
  - Similarity search: Which examples look like this x_i?
  - Association rules: Which x^j occur together?
  - Latent-factors: What 'parts' are the x_i made from?
  - Data visualization: What does the high-dimensional X look like?
  - Ranking: Which are the most important x_i?
  - Clustering: What types of x_i are there?

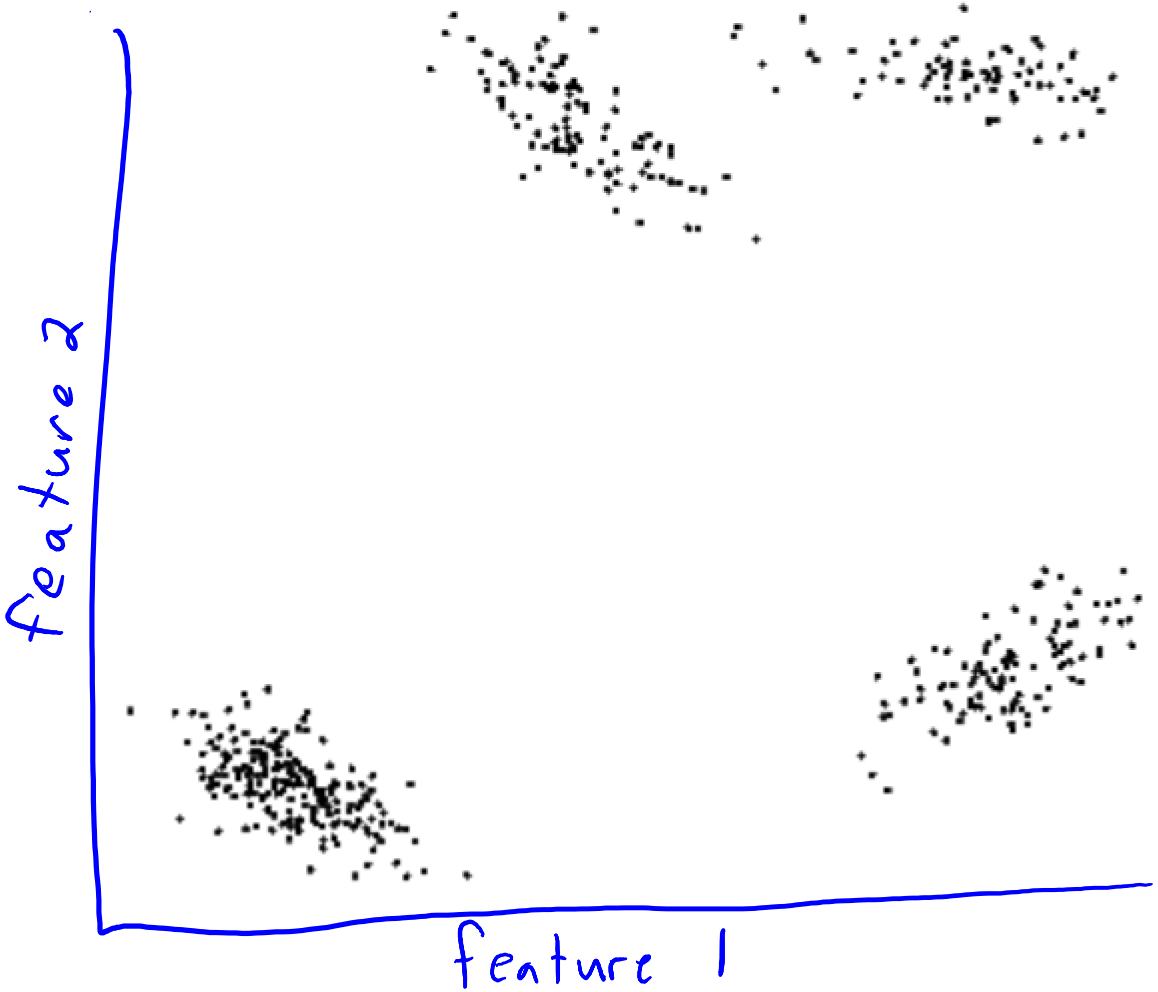
Clustering

- Clustering:
  - Input: set of objects described by features x_i.
  - Output: an assignment of objects to 'groups'.
- Unlike classification, we are not given the 'groups'.
  - Algorithm must discover groups.
- Example of groups we might discover in e-mail spam:
  - 'Lucky winner' group.
  - 'Weight loss' group.
  - 'Nigerian prince' group.
  - 'Russian bride' group.

Clustering Example

Input: data matrix 'X'.

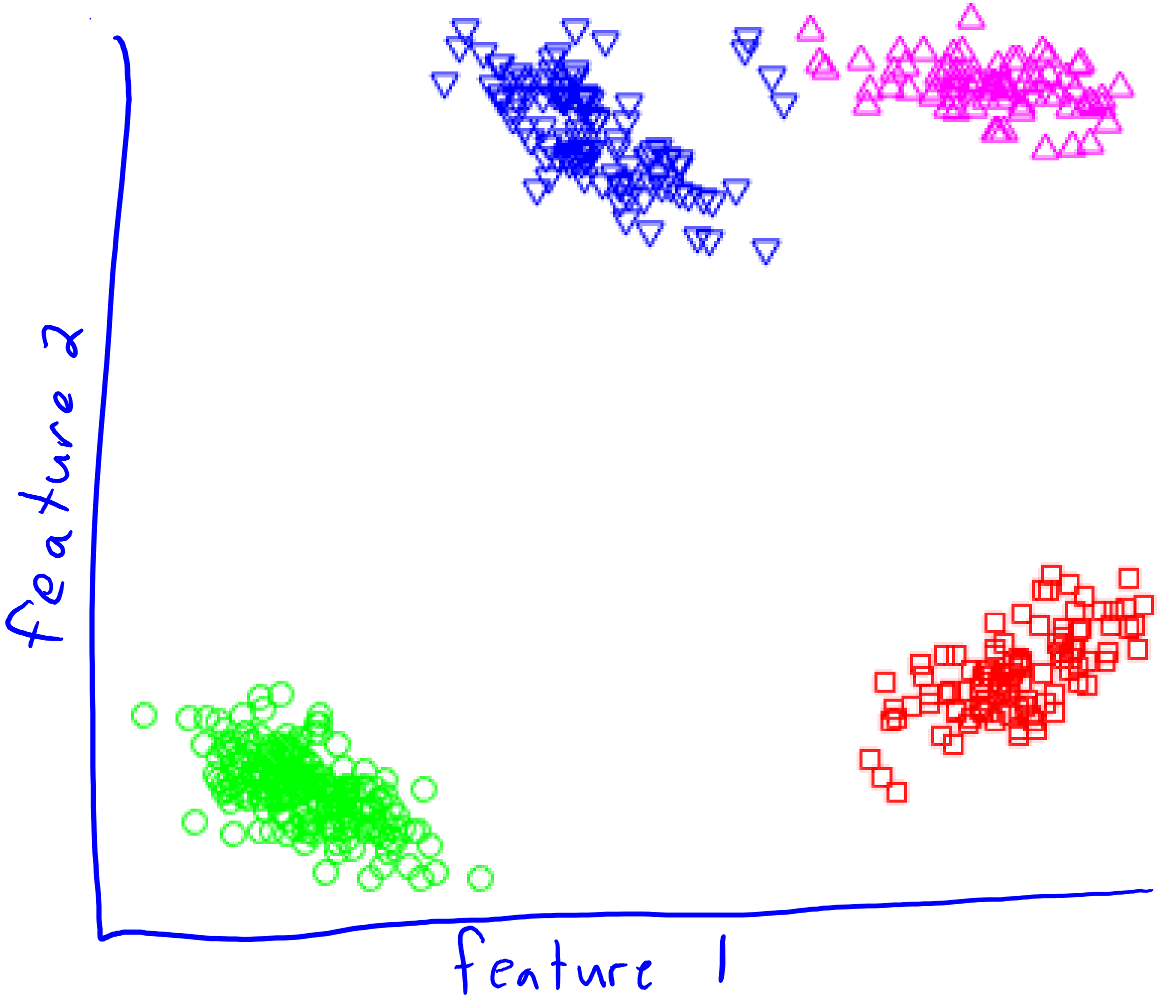
$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$



Clustering Example

Input: data matrix 'X'.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$



Output: clusters $\hat{y}$.

$$\hat{y} = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 3 \\ 1 \\ \vdots \end{bmatrix}$$

Data Clustering

- General goal of clustering algorithms:
  - Objects in the same group should be 'similar'.
  - Objects in different groups should be 'different'.
- But the 'best' clustering is hard to define:
  - We don't have a test error.
  - Generally, there is no 'best' method in unsupervised learning.
    - So there are lots of methods: we'll focus on important/representative ones.
- Why cluster?
  - You could want to know what the groups are.
  - You could want a 'prototype' example for each group.
  - You could want to find the group for a new example x_i.
  - You could want to find objects related to a new example x_i.

Clustering of Epstein-Barr Virus

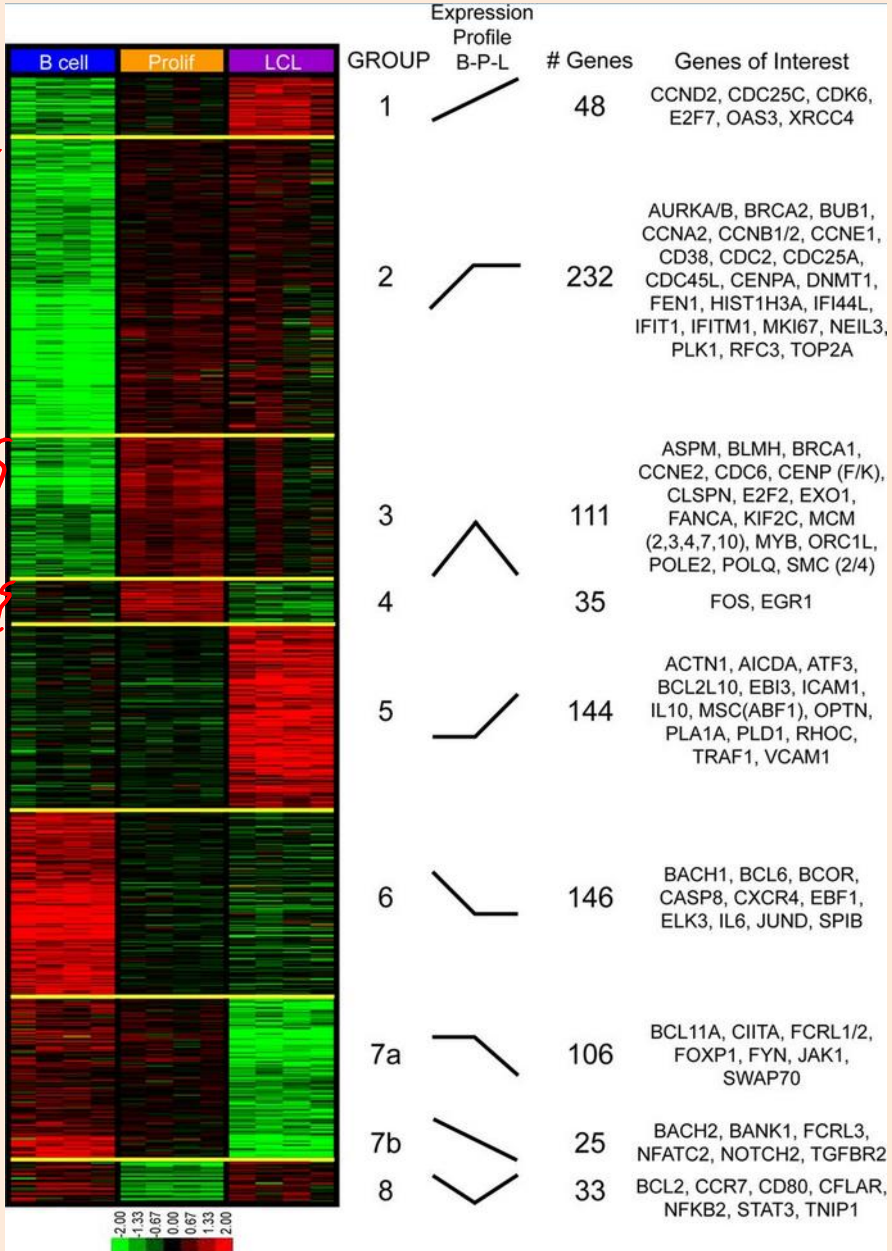
$\hat{y}_i=1$

$\hat{y}_i=2$

$\hat{y}_i=3$

$\hat{y}_i=4$

re-ordered

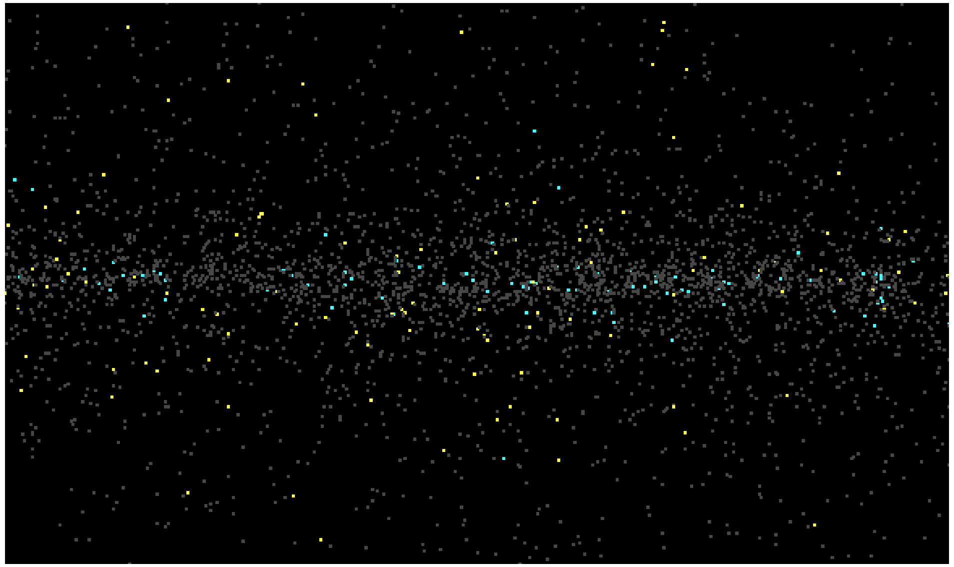


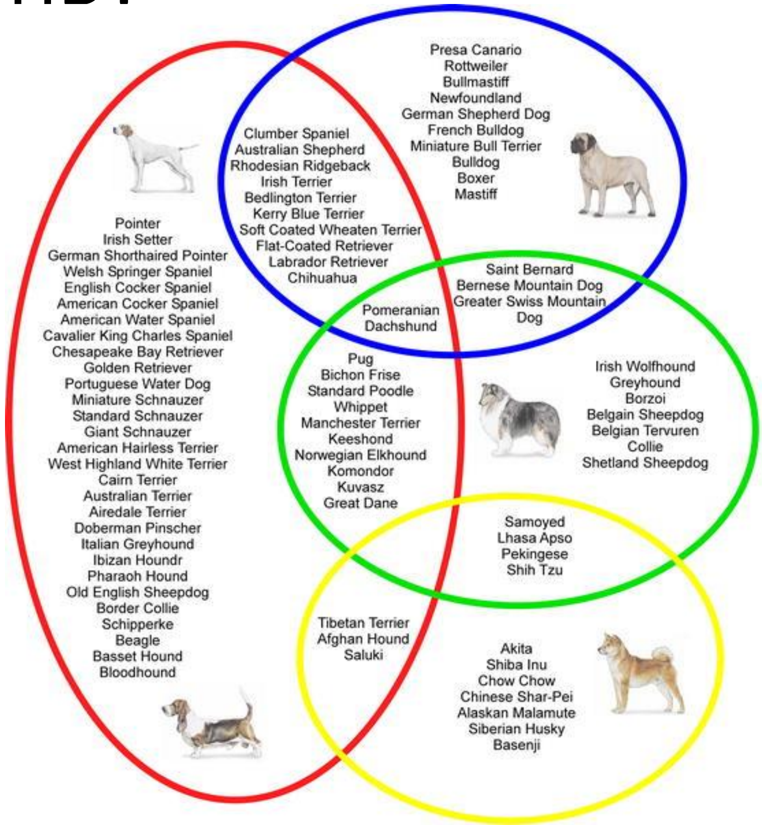
GROUP	# Genes	Genes of Interest
1	48	CCND2, CDC25C, CDK6, E2F7, OAS3, XRCC4
2	232	AURKA/B, BRCA2, BUB1, CCNA2, CCNB1/2, CCNE1, CD38, CDC2, CDC25A, CDC45L, CENPA, DNMT1, FEN1, HIST1H3A, IFI44L, IFIT1, IFITM1, MKI67, NEIL3, PLK1, RFC3, TOP2A
3	111	ASPM, BLMH, BRCA1, CCNE2, CDC6, CENP (F/K), CLSPN, E2F2, EXO1, FANCA, KIF2C, MCM (2,3,4,7,10), MYB, ORC1L, POLE2, POLQ, SMC (2/4)
4	35	FOS, EGR1
5	144	ACTN1, AICDA, ATF3, BCL2L10, EBI3, ICAM1, IL10, MSC(ABF1), OPTN, PLA1A, PLD1, RHOC, TRAF1, VCAM1
6	146	BACH1, BCL6, BCOR, CASP8, CXCR4, EBF1, ELK3, IL6, JUND, SPIB
7a	106	BCL11A, CIITA, FCRL1/2, FOXP1, FYN, JAK1, SWAP70
7b	25	BACH2, BANK1, FCRL3, NFATC2, NOTCH2, TGFBR2
8	33	BCL2, CCR7, CD80, CFLAR, NFKB2, STAT3, TNIP1

http://jvi.asm.org/content/86/20/11096.abstract

Other Clustering Applications

- NASA: what types of stars are there?
- Biology: are there sub-species?
- Documents: what kinds of documents are on my HD?
- Commercial: what kinds of customers do I have?





http://www.eecs.wsu.edu/~cook/dm/lectures/l9/index.html
http://www.biology-online.org/articles/canine_genomics_genetics_running/figures.html

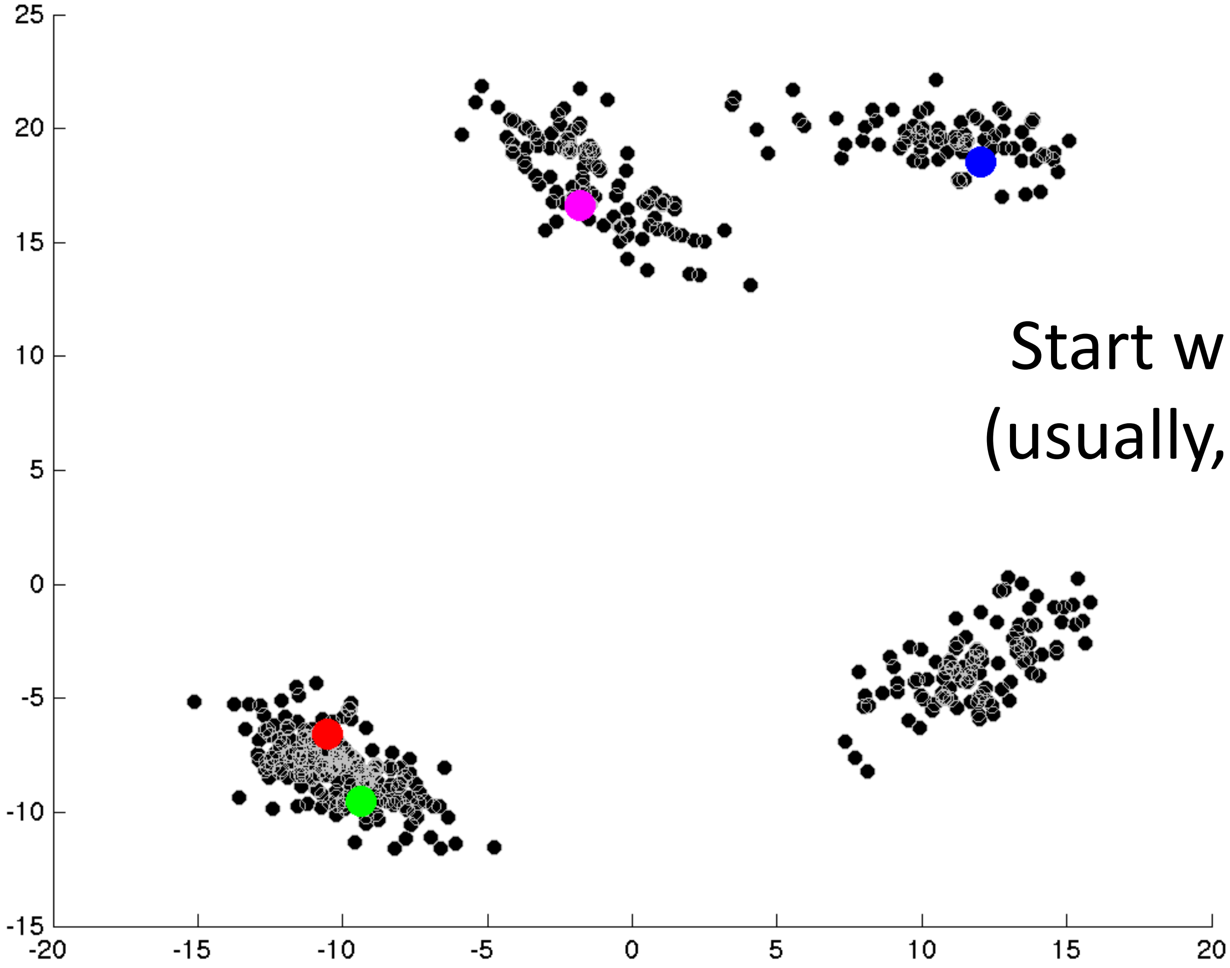
K-Means

- Most popular clustering method is k-means.
- Input:
  - The number of clusters ‘k’ (hyper-parameter).
  - Initial guess of the center (the “mean”) of each cluster.
- Algorithm:
  - Assign each x_i to its closest mean.
  - Update the means based on the assignment.
  - Repeat until convergence.

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

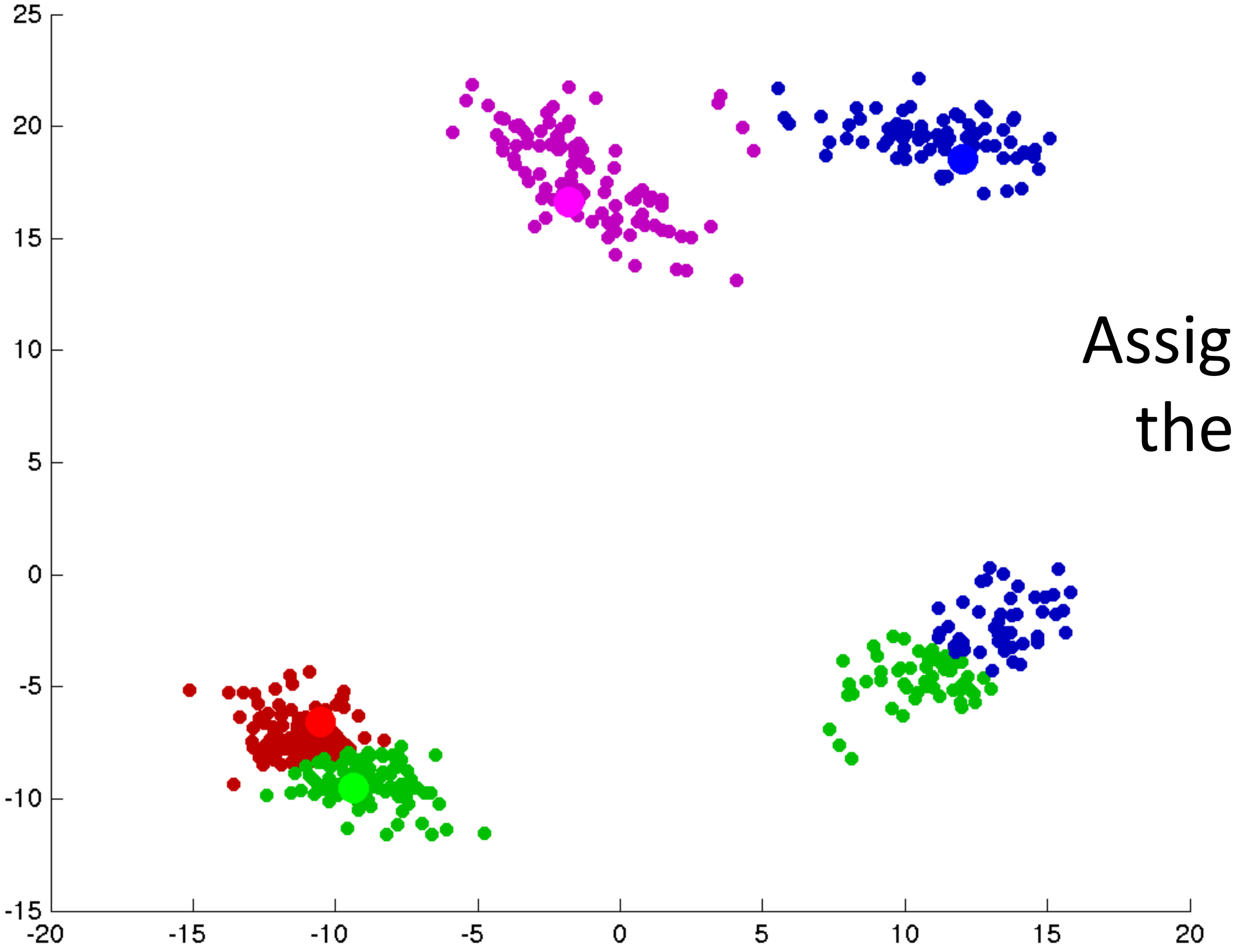


Start with ‘k’ initial ‘means’
(usually, random data points)

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

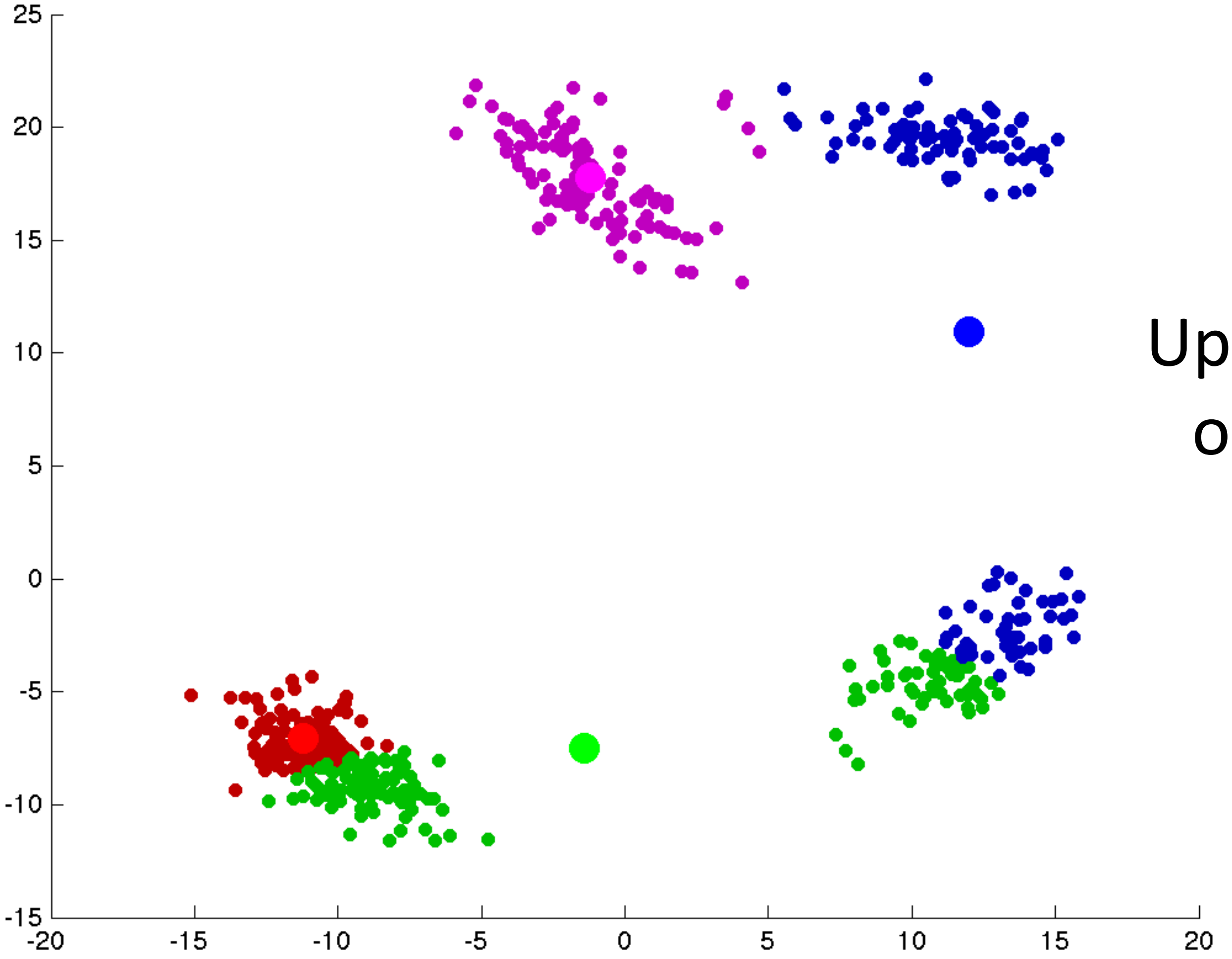


Assign each object to the closest mean.

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

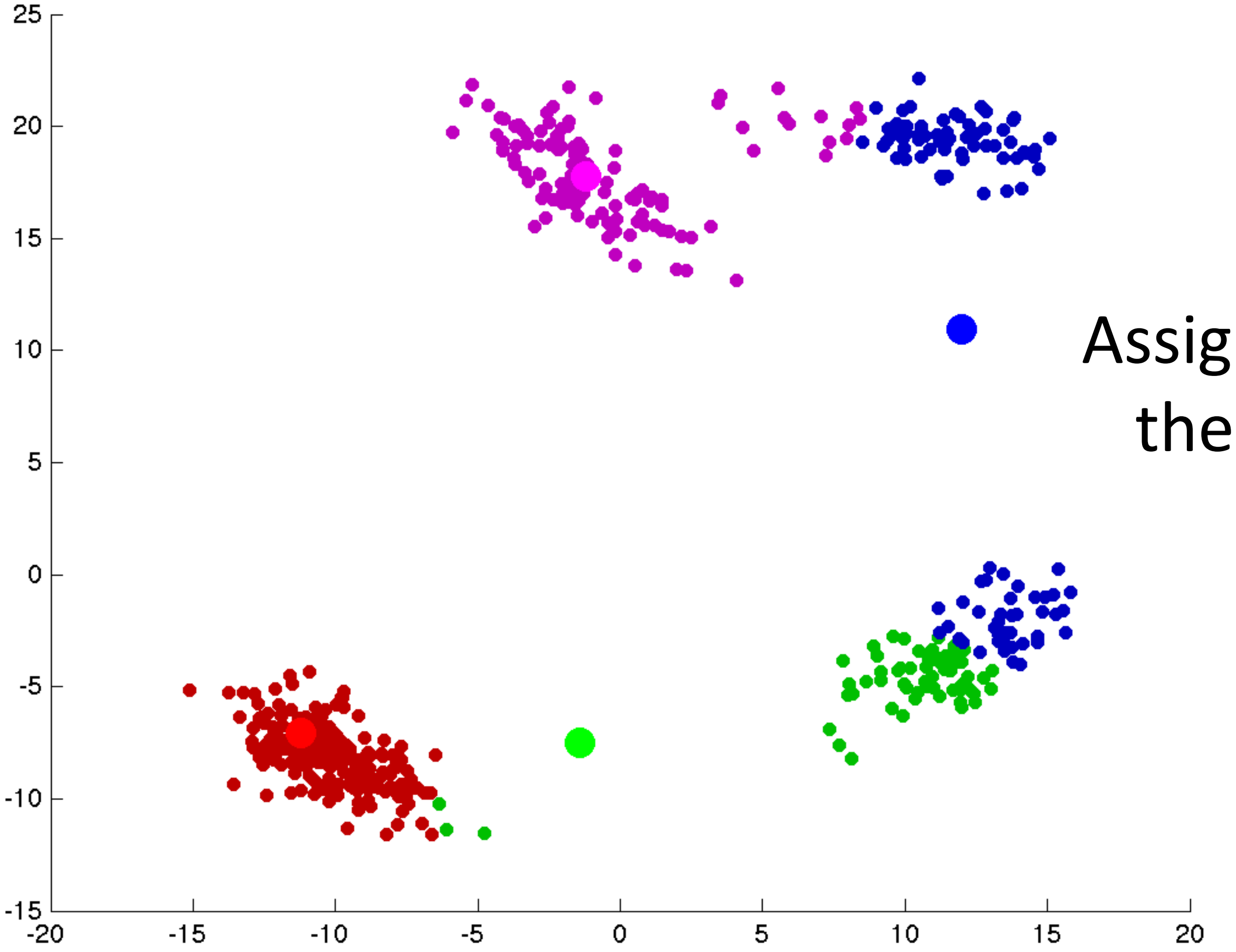


Update the mean of each group.

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

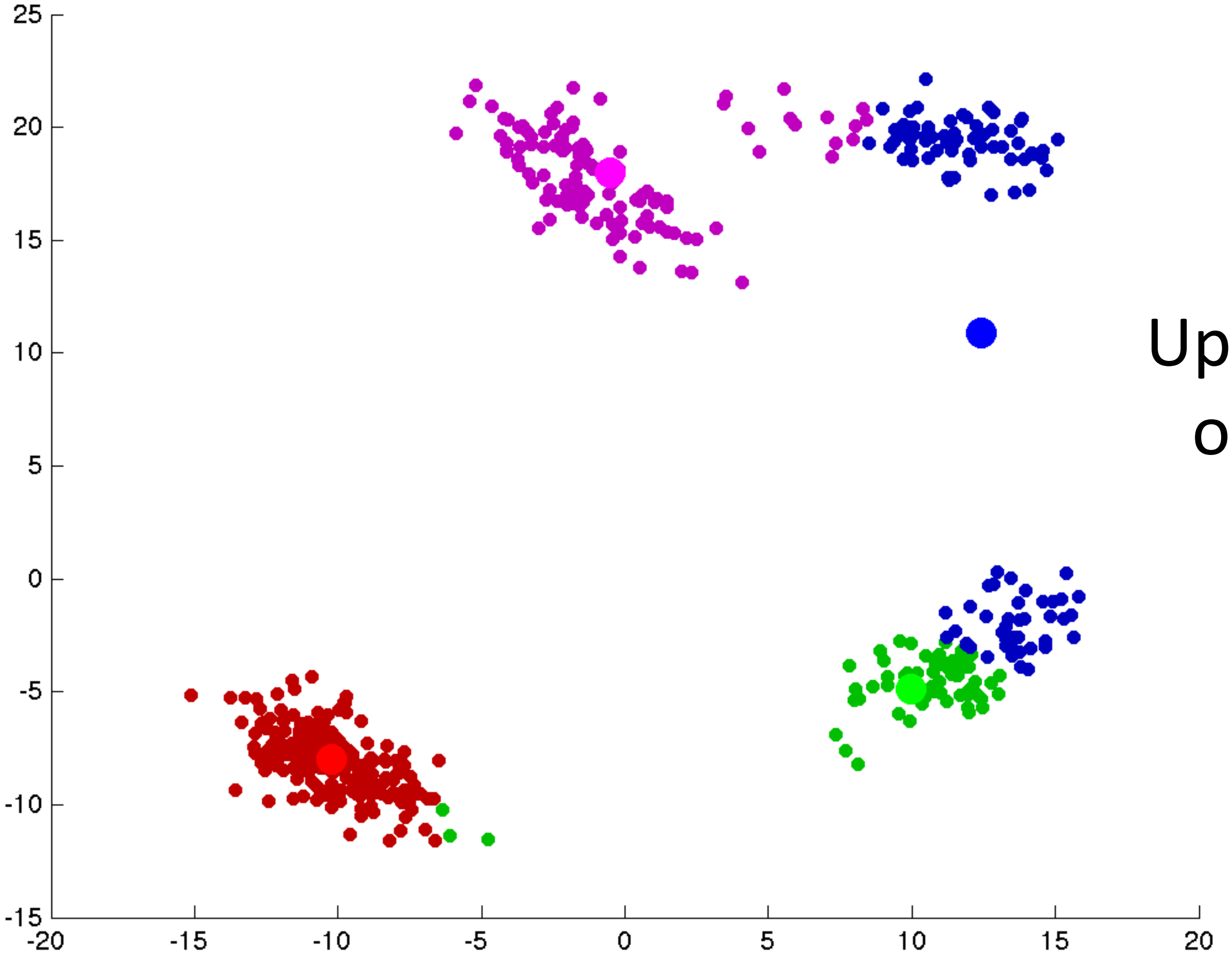


Assign each object to the closest mean.

K-Means Example

Input: data matrix 'X'.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

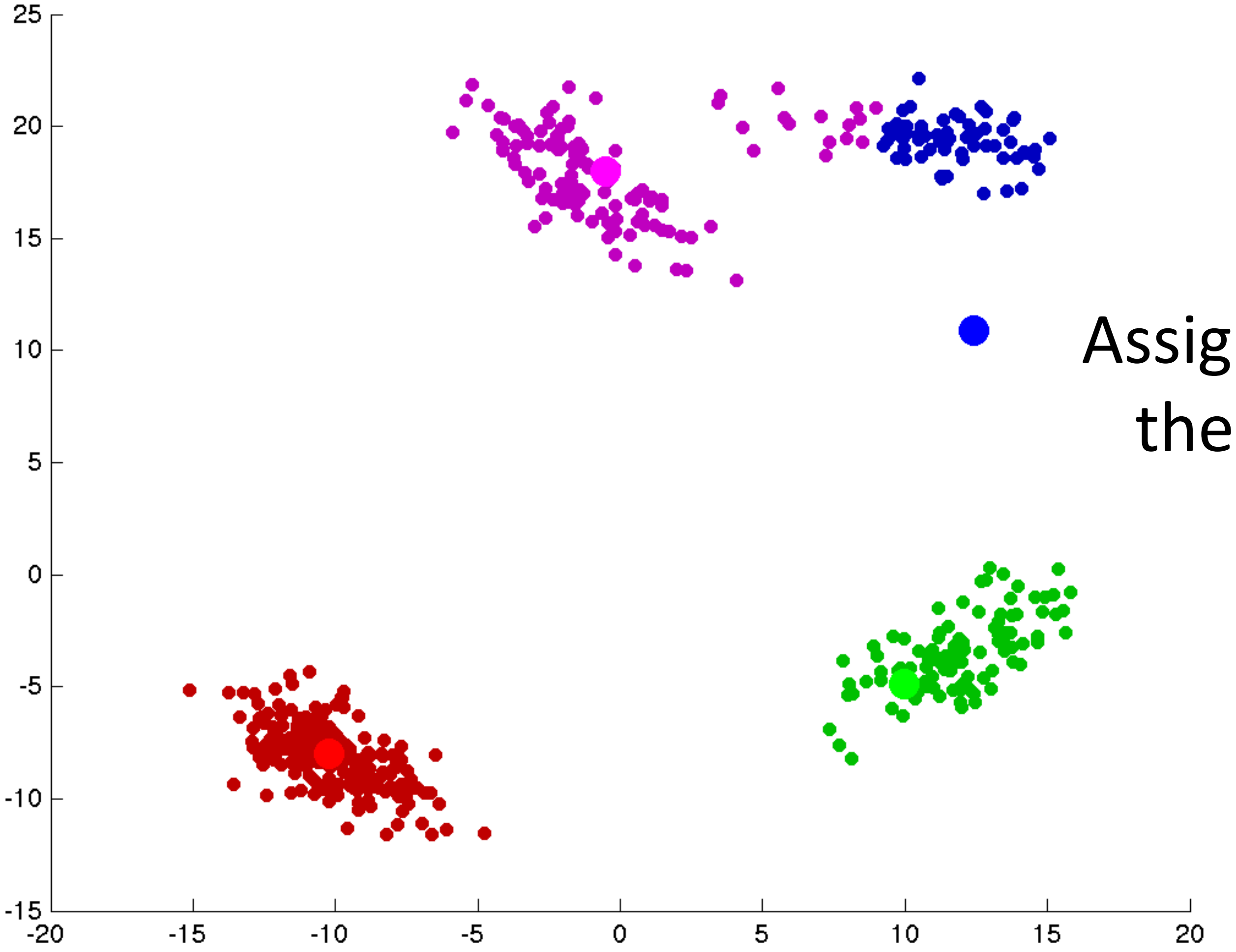


Update the mean of each group.

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

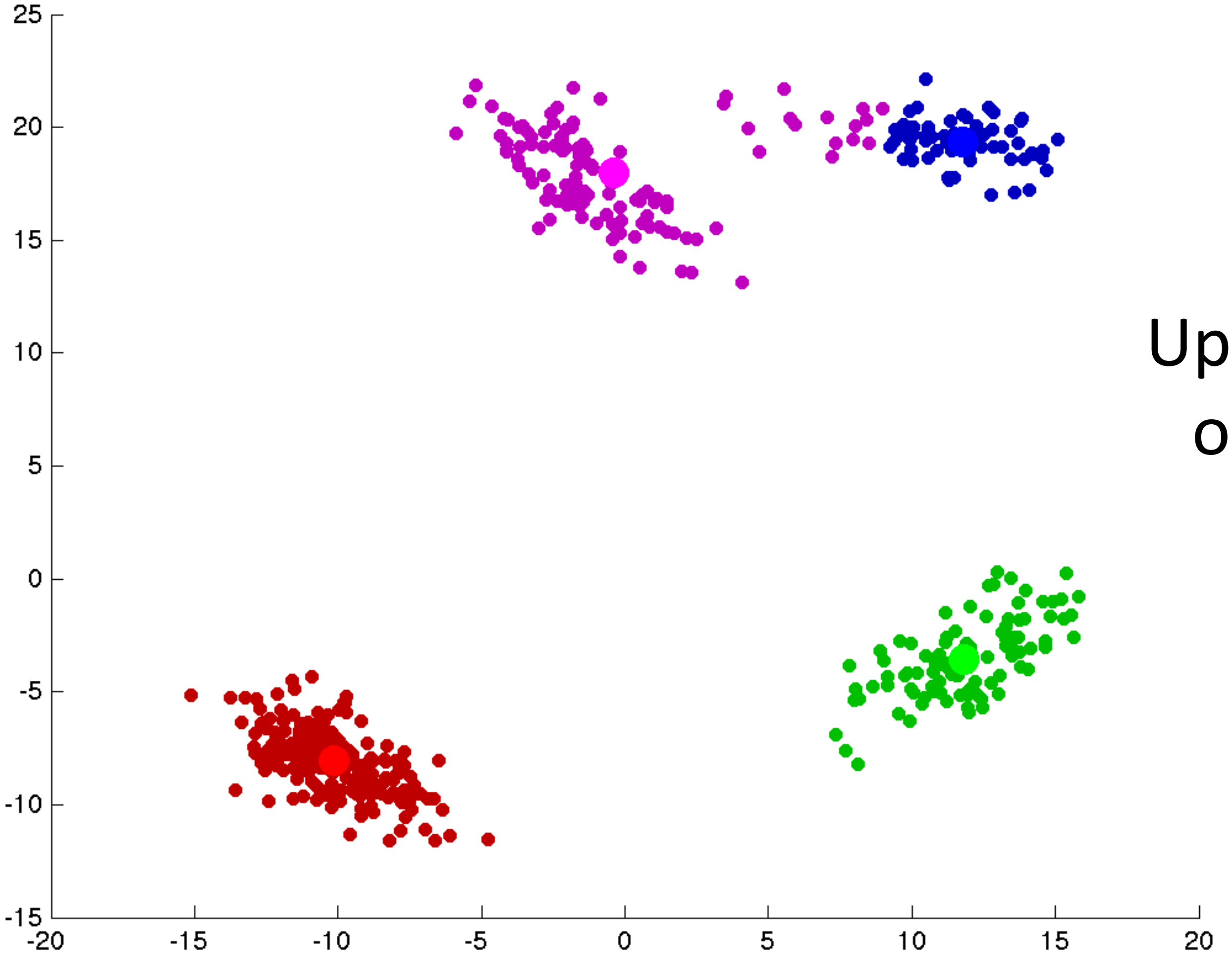


Assign each object to the closest mean.

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

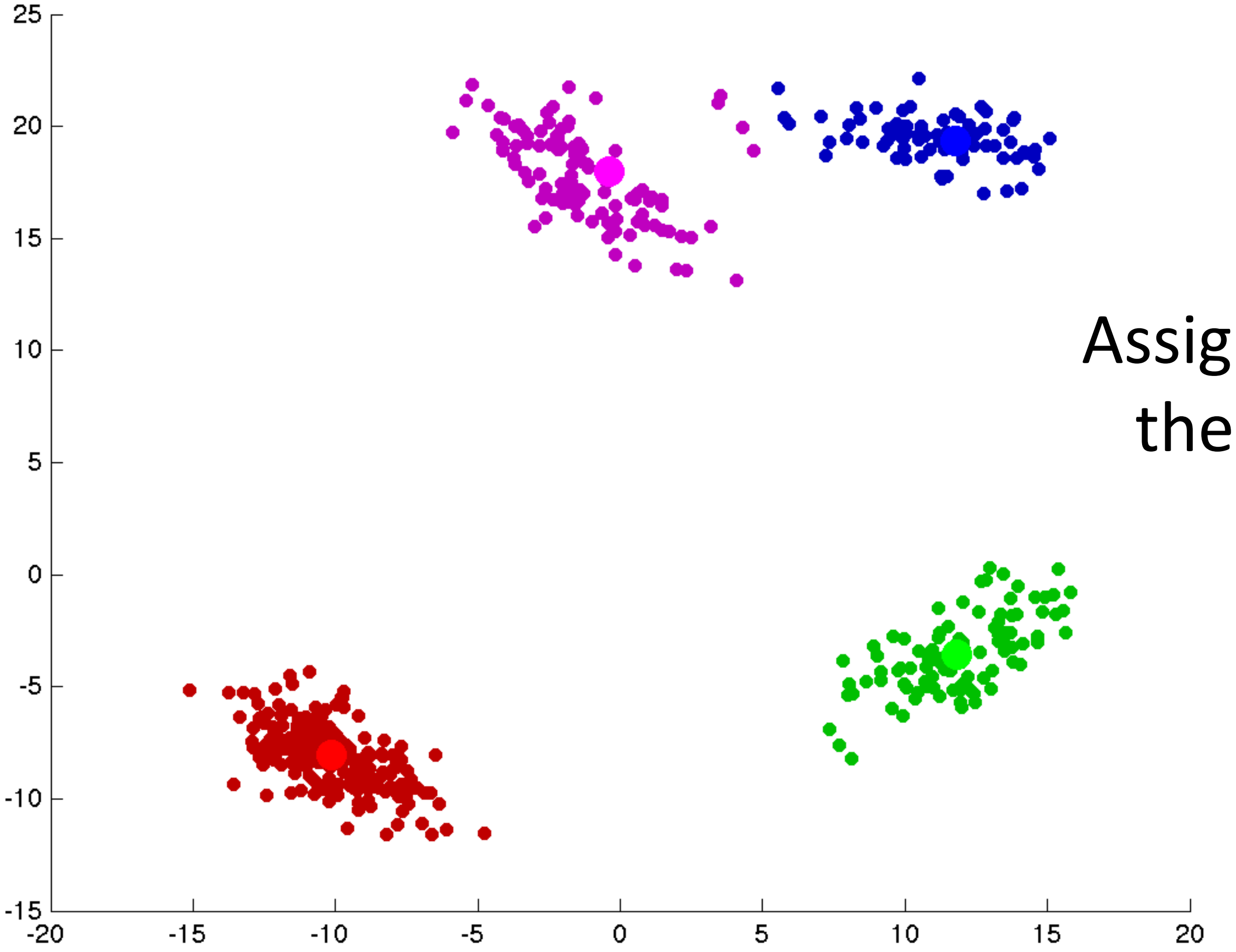


Update the mean of each group.

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

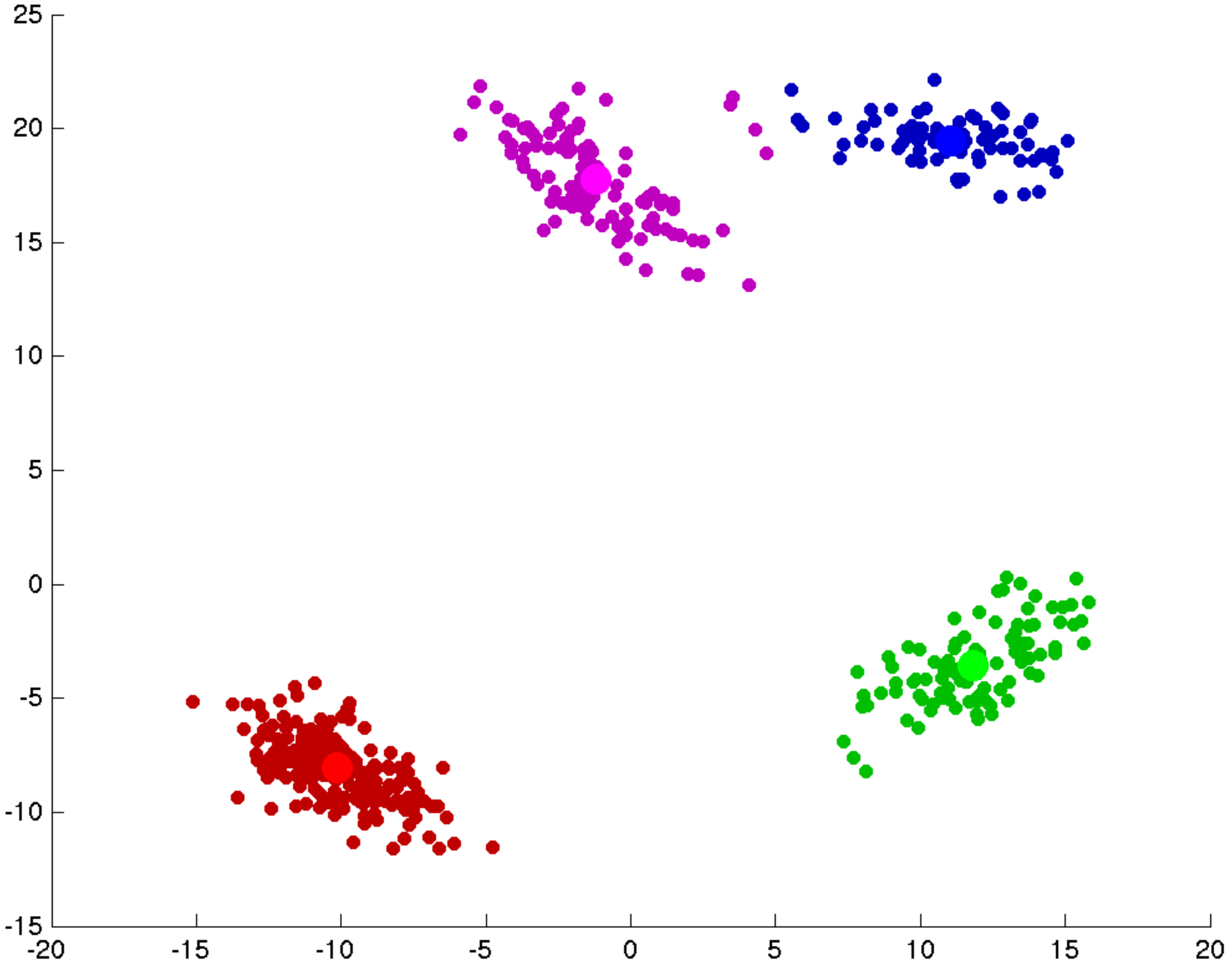


Assign each object to the closest mean.

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$

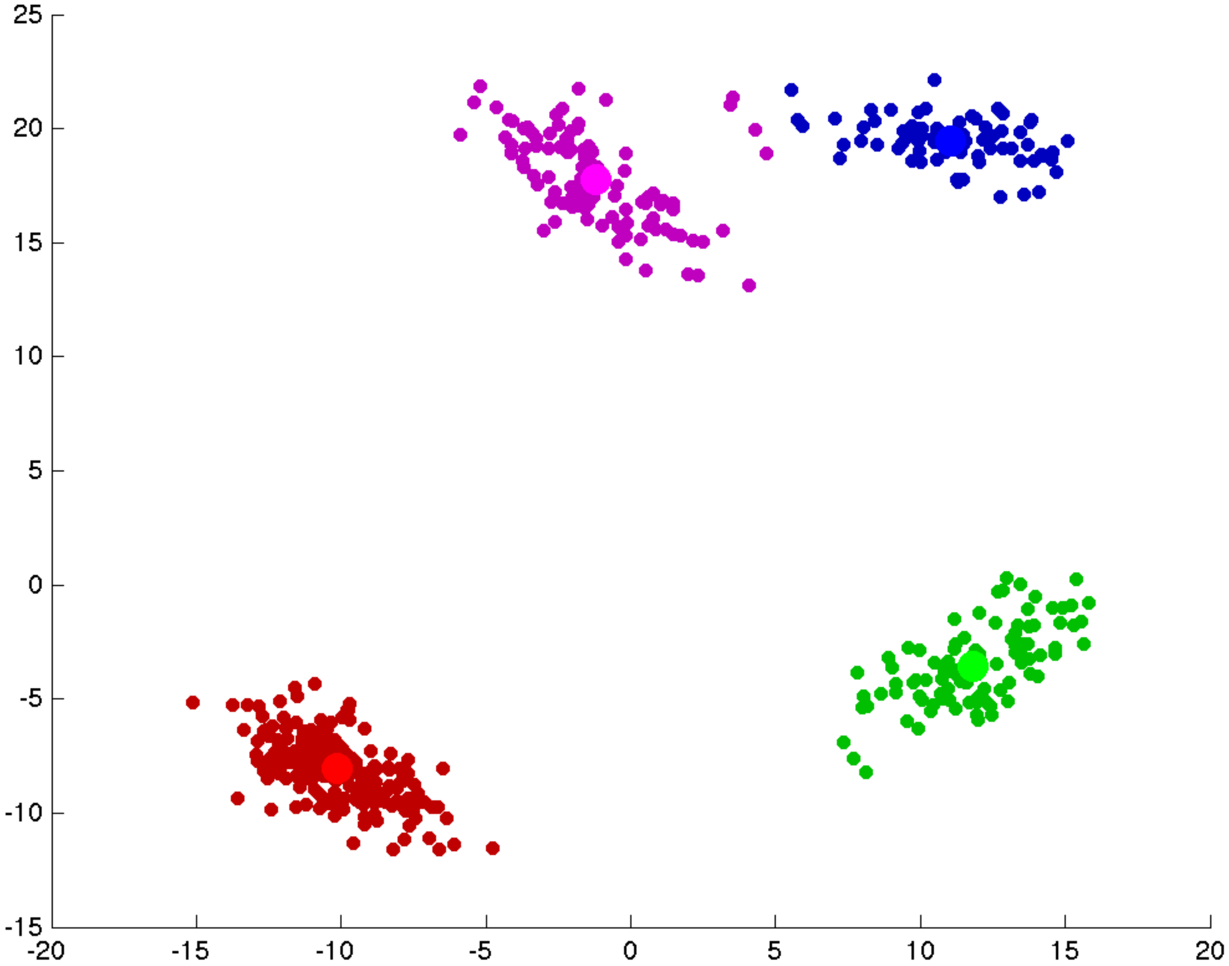


Stop if no objects change groups.

K-Means Example

Input: data matrix ‘X’.

$$X = \begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}$$



Output:

- Clusters ‘$\hat{y}$’.
- Means ‘W’.

$$\hat{y} = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 3 \\ 1 \\ \vdots \end{bmatrix}$$

$$W = \left.\underbrace{\begin{bmatrix} -1.2 & 17.8 \\ -10.2 & -8.0 \\ 11.0 & 19.5 \\ 11.8 & -3.6 \end{bmatrix}}_{d}\right\} k$$

Interactive demo:
https://www.naftaliharris.com/blog/visualizing-k-means-clustering

K-Means Issues

- Guaranteed to converge when using Euclidean distance.

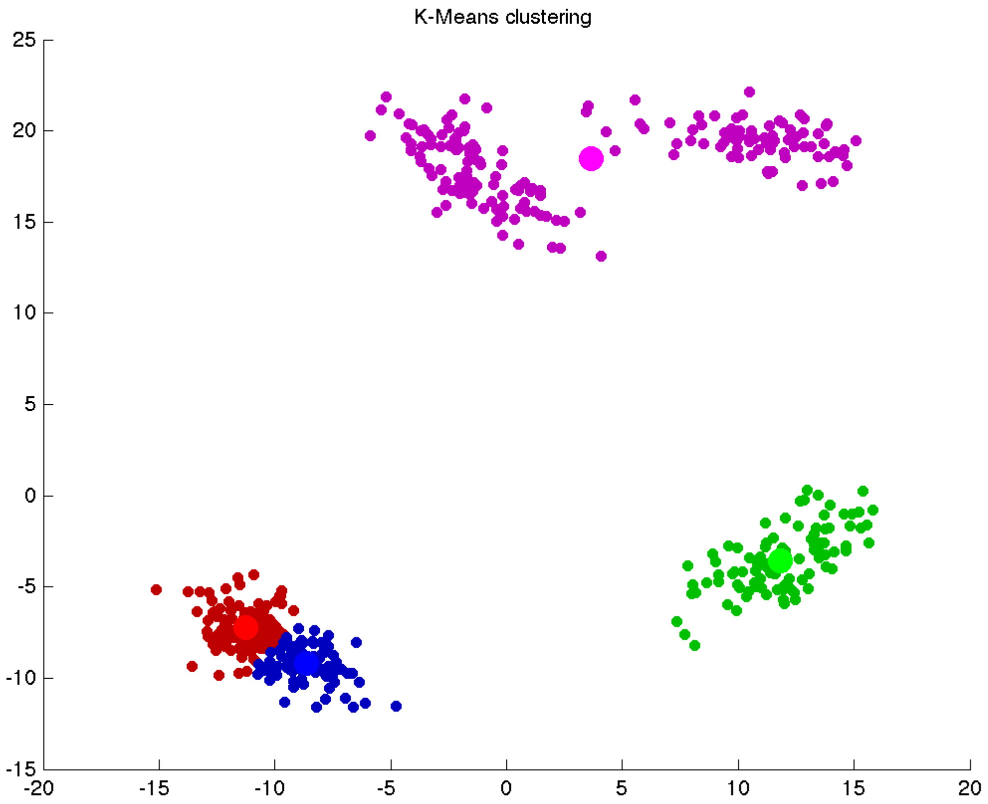
- Given a new test object:
  - Assign it to the nearest mean to cluster it.

- Assumes you know number of clusters ‘k’.
  - Lots of heuristics to pick ‘k’, none satisfying:
    - https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set

- Each object is assigned to one (and only one) cluster:
  - No possibility for overlapping clusters or leaving objects unassigned.

- It may converge to sub-optimal solution…

K-Means Clustering with Different Initialization



- Classic approach to dealing with sensitivity to initialization: random restarts.
  - Try several different random starting points, choose the "best".
- See bonus slides for a more clever approach called k-means++.

KNN vs. K-Means

- Don’t confuse KNN classification and k-means clustering:

Property	**KNN Classification**	**K-Means Clustering**
Task	Supervised learning (given y_i)	Unsupervised learning (no given y_i).
Meaning of ‘k’	Number of neighbours to consider (not number of classes).	Number of clusters (always consider single nearest mean).
Initialization	No training phase.	Training that is sensitive to initialization.
Model complexity	Model is complicated for small ‘k’, simple for large ‘k’.	Model is simple for small ‘k’, complicated for large ‘k’.

What is K-Means Doing?

- We can interpret K-Means steps as trying to minimize an objective:
  - Total sum of squared distances from object x_i to their centers $w_{\hat{y}_i}$:

$$f(w_1, w_2, \dots, w_k, \hat{y}_1, \hat{y}_2, \dots, \hat{y}_n) = \sum_{i=1}^{n} \|w_{\hat{y}_i} - x_i\|^2$$

$\longrightarrow$ Cluster of example 'i',

$\hat{y}_i \in \{1, 2, \dots, k\}$

- The k-means steps:
  - Minimize 'f' in terms of the $\hat{y}_i$ (update cluster assignments).
  - Minimize 'f' in terms of the w_c (update means).
- Termination of the algorithm follows because:
  - Each step does not increase the objective.
  - There are a finite number of assignments to k clusters.

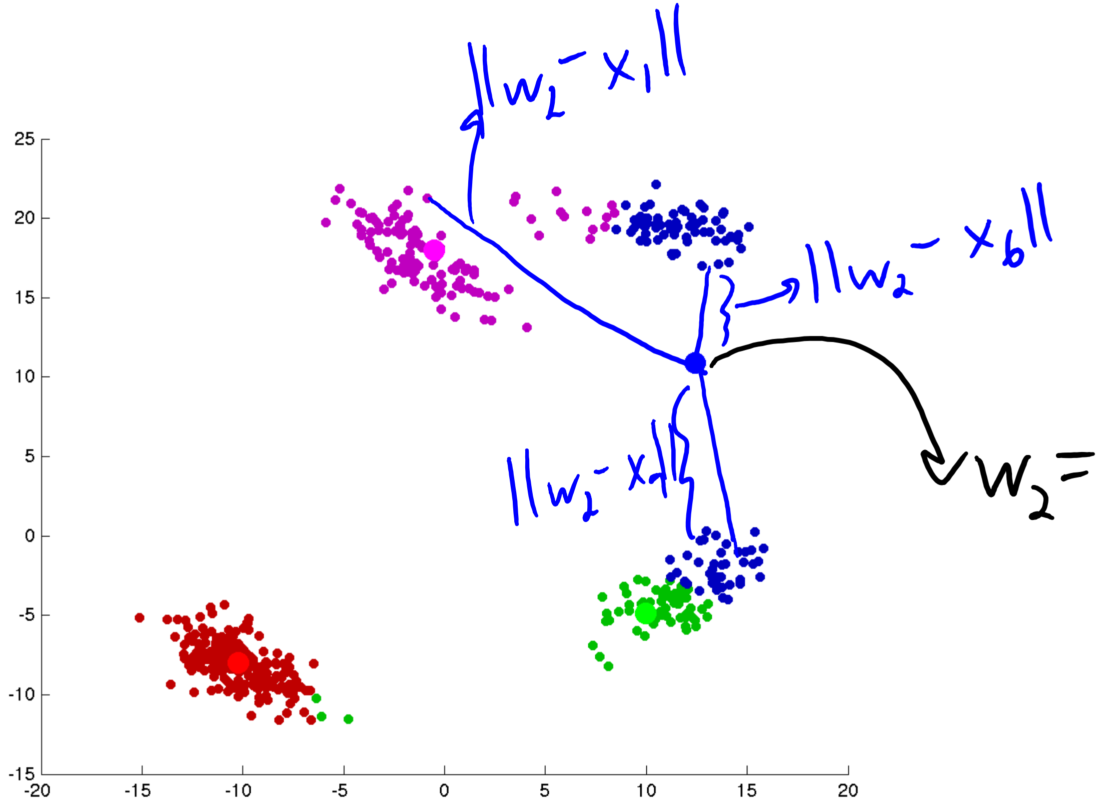
$$W = \underbrace{\begin{bmatrix} \text{---} \; w_1 \; \text{---} \\ \text{---} \; w_2 \; \text{---} \\ \vdots \\ \text{---} \; w_k \; \text{---} \end{bmatrix}}_{d} \left.\vphantom{\begin{bmatrix} a \\ a \\ a \\ a \end{bmatrix}}\right\} k$$

Cost of K-means

- Bottleneck is calculating distance from each x_i to each mean w_c:

$$\|w_c - x_i\|^2 = \sum_{j=1}^{d} (w_{cj} - x_{ij})^2$$

d-dimensional vector



$$w_2 = [13 \;\; 11]$$

Cost of K-means

- Bottleneck is calculating distance from each x_i to each mean w_c:

d-dimensional vector

$$\|w_c - x_i\|^2 = \sum_{j=1}^{d} (w_{cj} - x_{ij})^2$$

  - Each time we do this costs O(d).
- We need to compute distance from ‘n’ objects to ‘k’ clusters.
- Total cost of assigning objects to clusters is O(ndk).
  - Fast if k is not too large.
- Updating means is cheaper: O(nd).
  - For each cluster 'c', compute $w_c = \frac{1}{n_c} \sum_{i \in C} x_i$

Object in cluster.

Loop over objects in cluster.

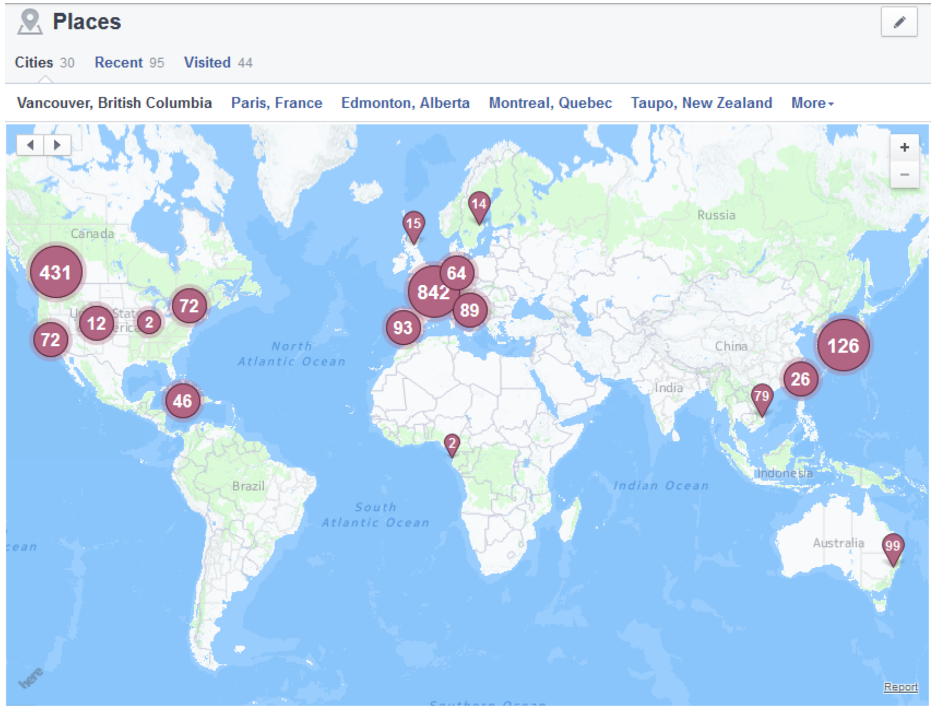
Number of objects in cluster 'c'

Vector Quantization

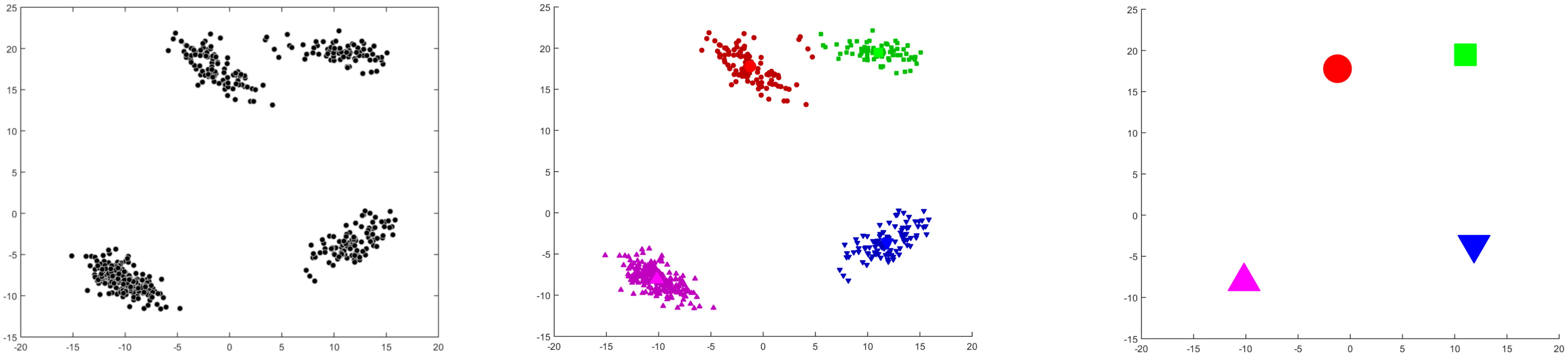
- K-means originally comes from signal processing.
- Designed for vector quantization:
  - Replace objects with the mean of their cluster ("prototype").

- Example:
  - Facebook places: 1 location summarizes many.
  - What sizes of clothing should I make?

http://wannabite.com/wp-content/uploads/2014/10/ragu-pasta-sauce-printable-coupon.jpg



Vector Quantization



$$X = \overset{n \times d}{\begin{bmatrix} -9.0 & -7.3 \\ -10.9 & -9.0 \\ 13.7 & 19.3 \\ 13.8 & 20.4 \\ 12.8 & 20.6 \\ \vdots & \vdots \end{bmatrix}} \xrightarrow{\text{Run k-means}} W = \overset{k \times d}{\begin{bmatrix} -1.2 & 17.8 \\ -10.2 & -8.0 \\ 11.0 & 19.5 \\ 11.8 & -3.6 \end{bmatrix}} \quad \hat{y} = \overset{n \times 1}{\begin{bmatrix} \boxed{2} \\ 2 \\ 3 \\ 3 \\ 1 \\ \vdots \end{bmatrix}} \xrightarrow[\text{with means.}]{\text{Approximate objects}} \hat{X} = \begin{bmatrix} -10.2 & -8.0 \\ -10.2 & -8.0 \\ 11.0 & 19.5 \\ 11.0 & 19.5 \\ -1.2 & 17.8 \\ \vdots & \end{bmatrix}$$

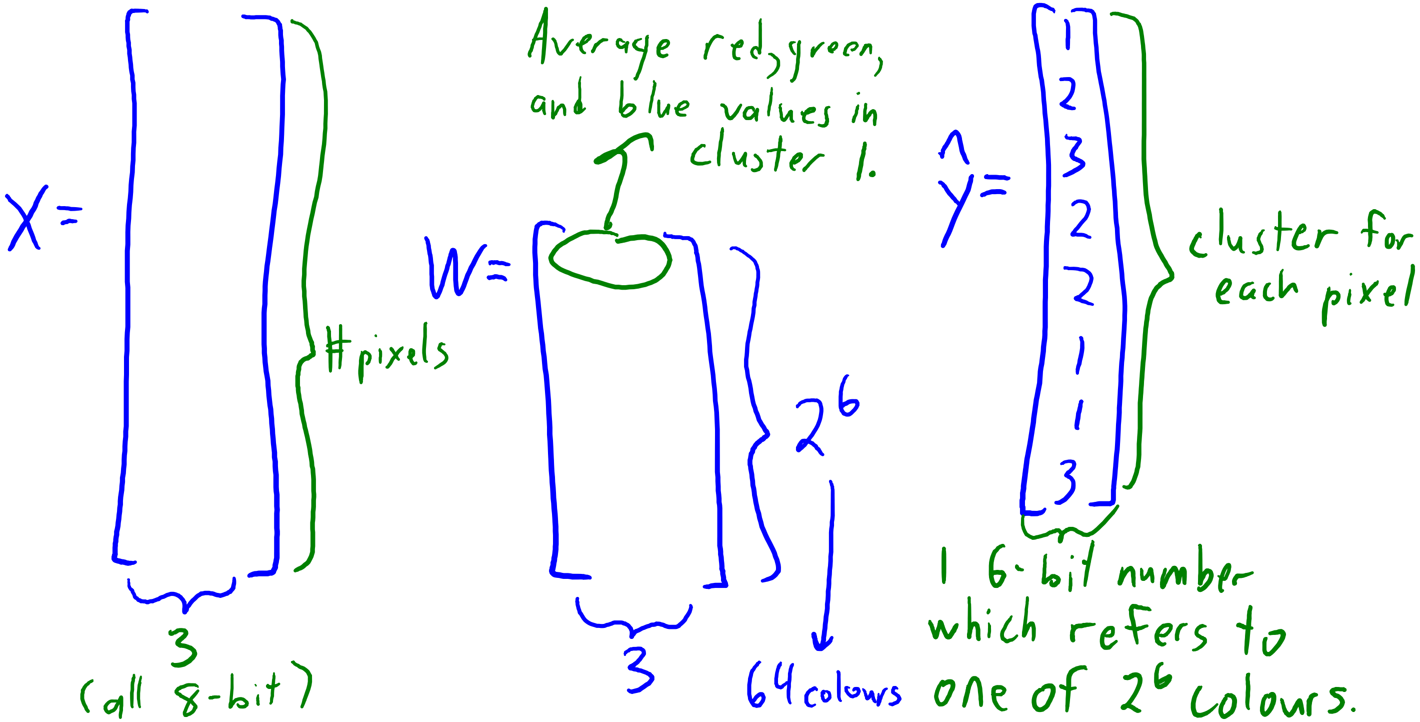
Vector Quantization: Image Colors

- Usual RGB representation of a pixel’s color: three 8-bit numbers.
  - For example, [241 13 50] = 🟥.
  - Can apply k-means to find set of prototype colours.

Original:
(24-bits/pixel)

Run k-means with
2^6 clusters:

K-means predictions:
(6-bits/pixel)



Vector Quantization: Image Colors

- Usual RGB representation of a pixel's color: three 8-bit numbers.
  - For example, [241 13 50] = 🟥.
  - Can apply k-means to find set of prototype colours.

Original:
(24-bits/pixel)

$$X = \left.\underbrace{\begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}}_{\substack{3 \\ (\text{all 8-bit})}}\right\} \#\text{pixels}$$

Run k-means with
2^6 clusters:

Average red, green, and blue values in cluster 1.

$$W = \left.\underbrace{\begin{bmatrix} \bigcirc \\ \\ \\ \\ \end{bmatrix}}_{3}\right\} 2^3 = 8$$

64 colours

K-means predictions:
(3-bits/pixel)

$$\hat{y} = \left.\underbrace{\begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \\ 2 \\ 1 \\ 1 \\ 3 \end{bmatrix}}\right\} \text{cluster for each pixel}$$

1 3-bit number which refers to one of 2^3 colours.

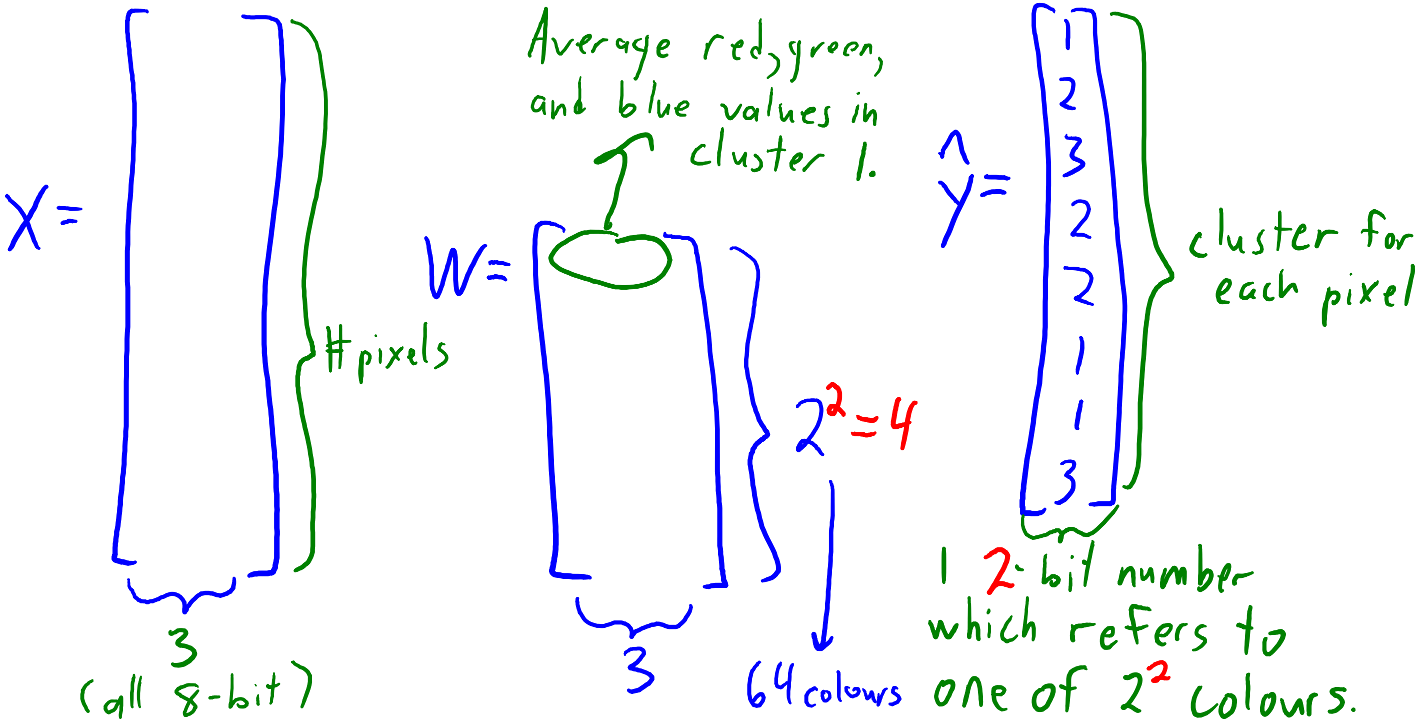
Vector Quantization: Image Colors

- Usual RGB representation of a pixel's color: three 8-bit numbers.
  - For example, [241 13 50] = 🟥.
  - Can apply k-means to find set of prototype colours.

Original:
(24-bits/pixel)

Run k-means with
2^6 clusters:

K-means predictions:
(2-bits/pixel)



Average red, green, and blue values in cluster 1.

1 2-bit number which refers to one of 2^2 colours.

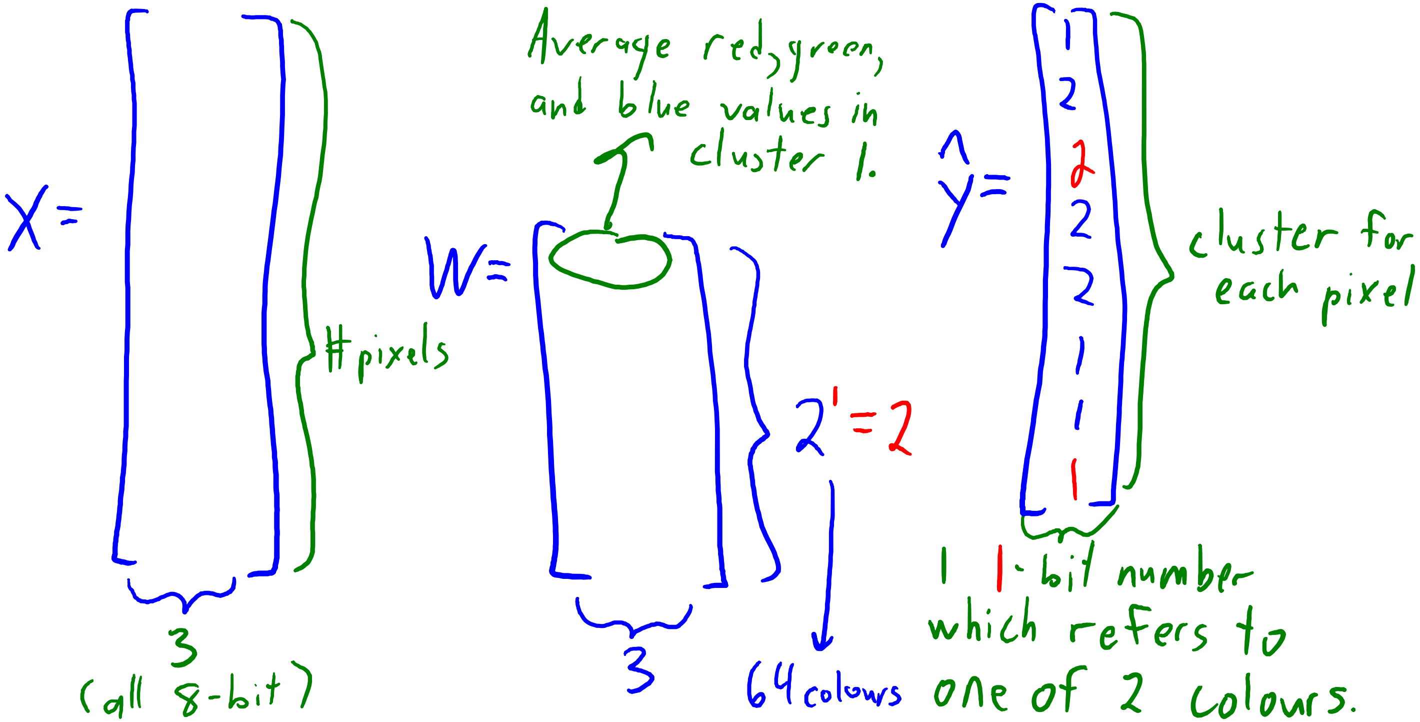
Vector Quantization: Image Colors

- Usual RGB representation of a pixel’s color: three 8-bit numbers.
  - For example, [241 13 50] = 🟥.
  - Can apply k-means to find set of prototype colours.

Original:
(24-bits/pixel)

Run k-means with
2^6 clusters:

K-means predictions:
(1-bit/pixel)



Summary

- Unsupervised learning: fitting data without explicit labels.
- Clustering: finding ‘groups’ of related objects.
- K-means: simple iterative clustering strategy.
  - Fast but sensitive to initialization.
- Vector quantization:
  - Compressing objects by replacing them with the mean of their cluster.

- Next time:
  - John Snow and non-parametric clustering.

What is K-Means Doing?

- How are are k-means step decreasing this objective?

$$f(w_1, w_2, \dots, w_k, \hat{y}_1, \hat{y}_2, \dots, \hat{y}_n) = \sum_{i=1}^{n} \|w_{\hat{y}_i} - x_i\|^2$$

- If we just write as function of a particular $\hat{y}_i$, we get:

$$f(\hat{y}_i) = \|w_{\hat{y}_i} - x_i\|^2 + (\text{constant})$$

  - The “constant” includes all other terms, and doesn’t affect location of min.
  - We can minimize in terms of $\hat{y}_i$ by setting it to the ‘c’ with w_c closest to x_i.

What is K-Means Doing?

- How are are k-means step decreasing this objective?

$$f(w_1, w_2, \ldots, w_k, \hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n) = \sum_{i=1}^{n} \|w_{\hat{y}_i} - x_i\|^2$$

- If we just write as function of a particular w_{cj} we get:

$$f(w_{cj}) = \underbrace{\sum_{i \in c'}}_{\text{set of examples with } \hat{y}_i = c'} \sum_{j'=1}^{d} (w_{c'j'} - x_{ij'})^2 + (\text{constant})$$

- Derivative is given by: $f'(w_{cj}) = 2 \sum_{i \in c} (w_{cj} - x_{ij})$
- Setting equal to 0 and solving for w_{cj} gives: $\sum_{i \in c} w_{cj} = \sum_{i \in c} x_{ij}$ or $w_{cj} * n_c = \sum_{i \in c} x_{ij}$

or $w_{cj} = \frac{1}{n_c} \sum_{i \in c} x_{ij}$

K-Medians Clustering

- With other distances k-means may not converge.
  - But we can make it converge by changing the updates so that they are minimizing an objective function.
- E.g., we can use the L1-norm objective: $\sum_{i=1}^{n} \|w_{\hat{y}_i} - x_i\|_1$
- Minimizing the L1-norm objective gives the 'k-medians' algorithm:
  - Assign points to clusters by finding "mean" with smallest L1-norm distance.
  - Update 'means' as median value (dimension-wise) of each cluster.
    - This minimizes the L1-norm distance to all the points in the cluster.
- This approach is more robust to outliers.

k-means will put a cluster here.

What is the "L1-norm and median" connection?

- Point that minimizes the sum of squared L2-norms to all points:

$$f(w) = \sum_{i=1}^{n} \|w - x_i\|^2$$

  - Is given by the mean (just take derivative and set to 0):

$$w = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- Point that minimizes the sum of L1-norms to all all points:

$$f(w) = \sum_{i=1}^{n} \|w - x_i\|_1$$

  - Is given by the median (derivative of absolute value is +1 if positive and -1 if negative, so any point with half of points larger and half of points smaller is a solution).

K-Medoids Clustering

- A disadvantage of k-means in some applications:
  - The means might not be valid data points.
  - May be important for vector quantiziation.
- E.g., consider bag of words features like [0,0,1,1,0].
  - We have words 3 and 4 in the document.
- A mean from k-means might look like [0.1 0.3 0.8 0.2 0.3].
  - What does it mean to have 0.3 of word 2 in a document?
- Alternative to k-means is k-medoids:
  - Same algorithm as k-means, except the means must be data points.
  - Update the means by finding example in cluster minimizing squared L2-norm distance to all points in the cluster.

K-Means Initialization

- K-means is fast but sensitive to initialization.

- Classic approach to initialization: random restarts.
  - Run to convergence using different random initializations.
  - Choose the one that minimizes average squared distance of data to means.

- Newer approach: k-means++
  - Random initialization that prefers means that are far apart.
  - Yields provable bounds on expected approximation ratio.

K-Means++

- Steps of k-means++:
  1. Select initial mean w_1 as a random x_i.
  2. Compute distance d_{ic} of each object x_i to each mean w_c.

     $$d_{ic} = \sqrt{\sum_{j=1}^{d} (x_{ij} - w_{cj})^2} = \|x_i - w_c\|_2$$

  3. For each object 'i' set d_i to the distance to the closest mean.

     $$d_i = \min_c \{d_{ic}\}$$

  4. Choose next mean by sampling an example 'i' proportional to $(d_i)^2$.

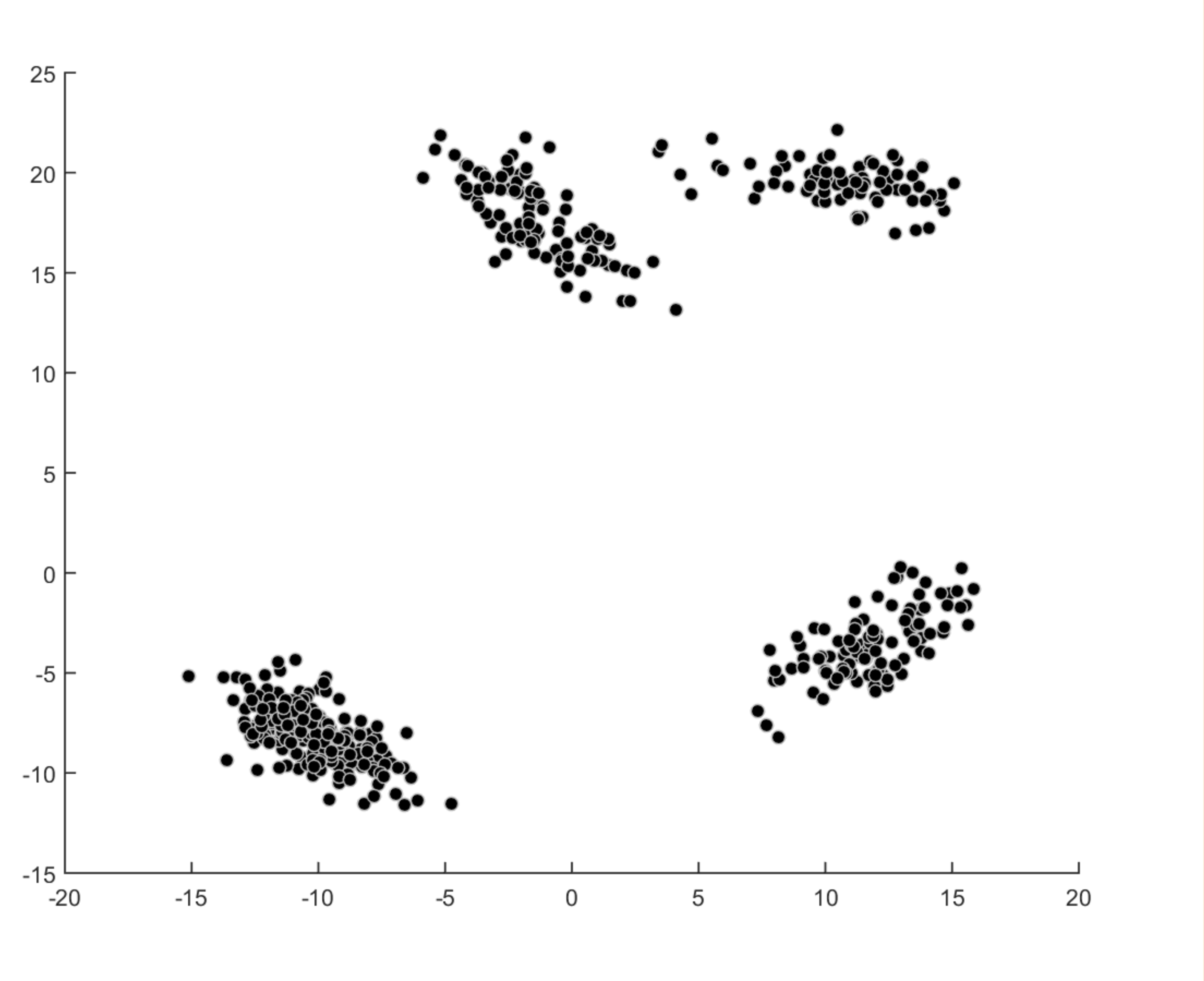
     $$p_i \propto d_i^2 \Rightarrow p_i = \frac{d_i^2}{\sum_{j=1}^{n} d_j^2}$$

     "probability that we choose x_i as next mean"

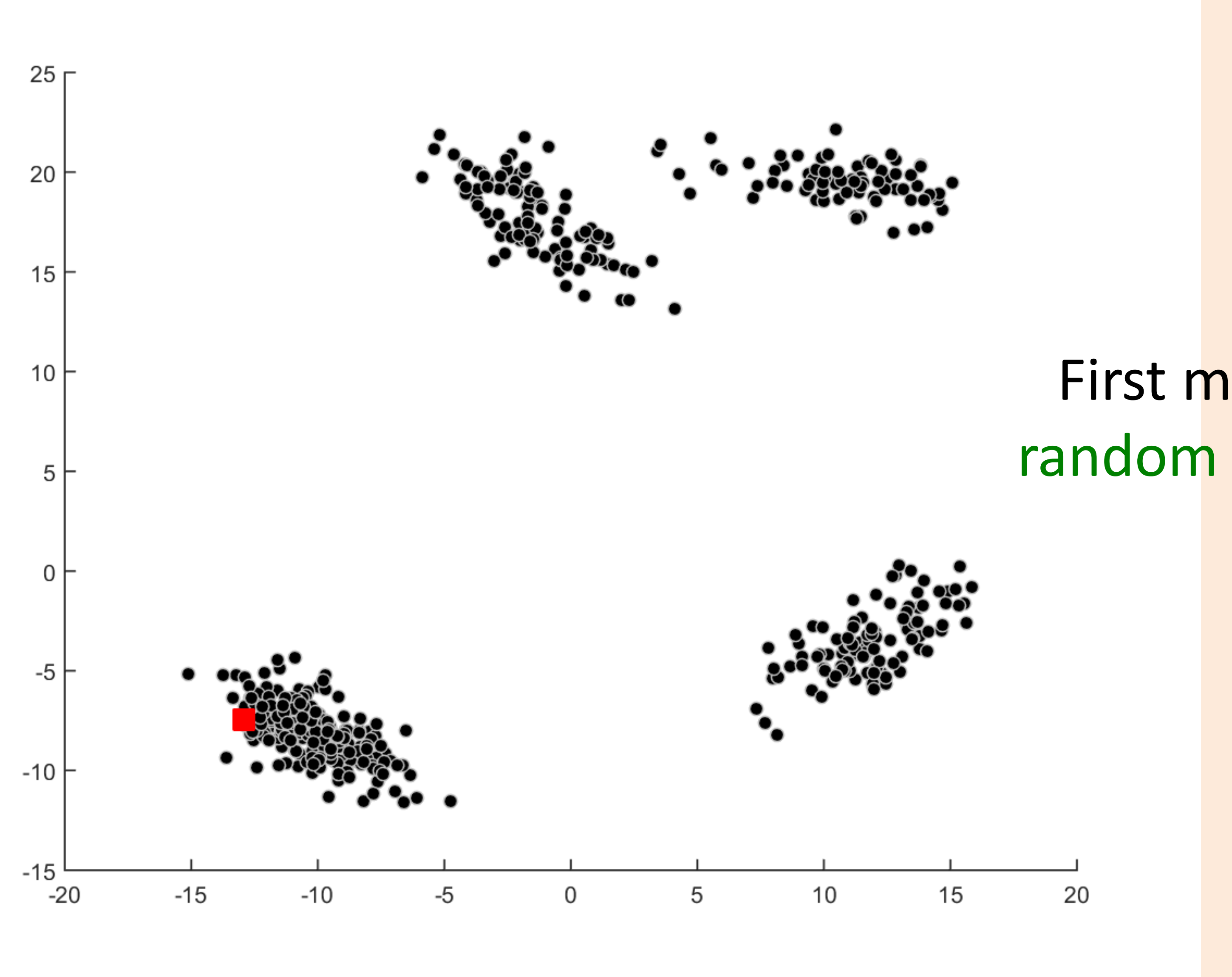
     Can be done in $O(n)$

  5. Keep returning to step 2 until we have k-means.
- Expected approximation ratio is O(log(k)).

K-Means++

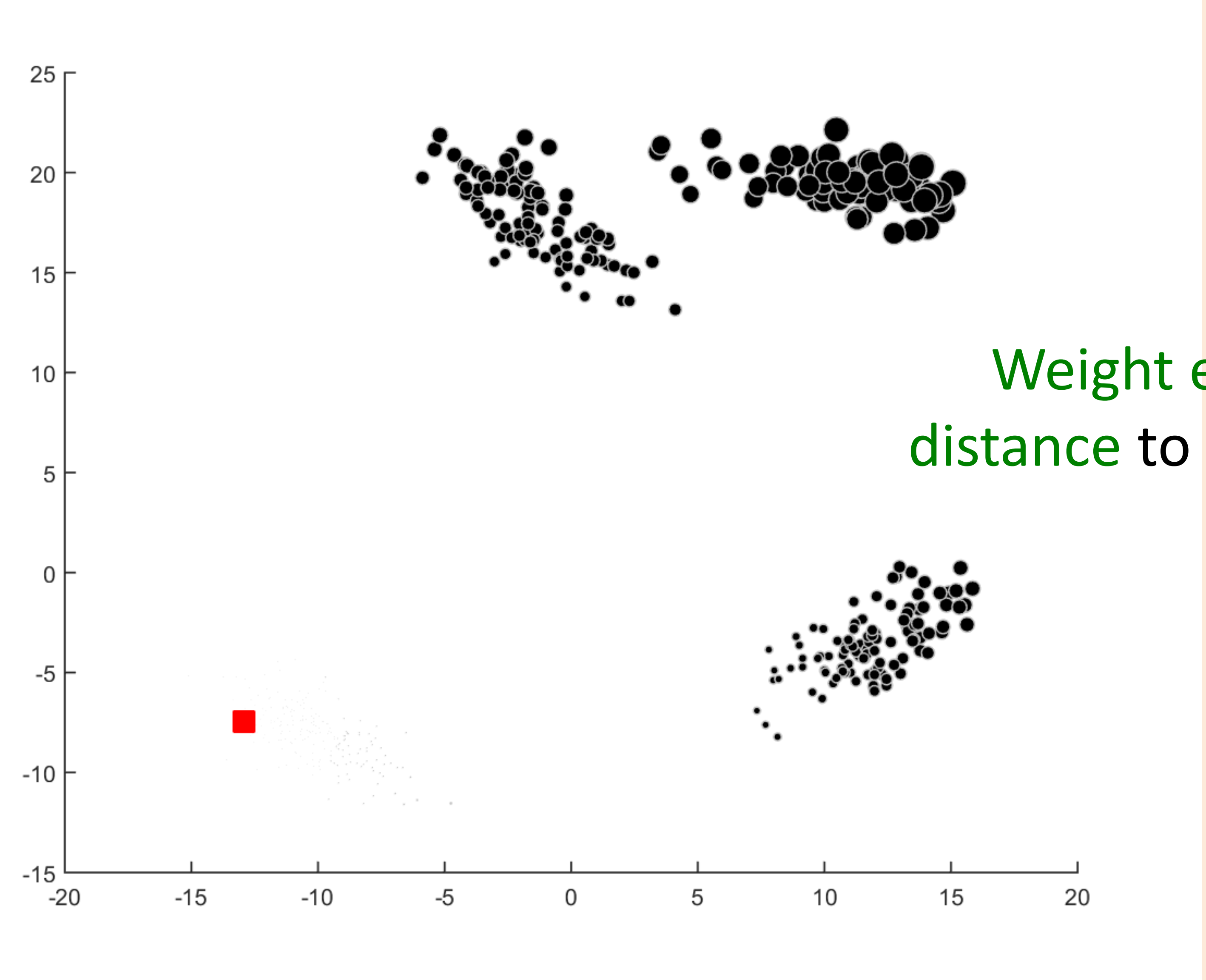


K-Means++



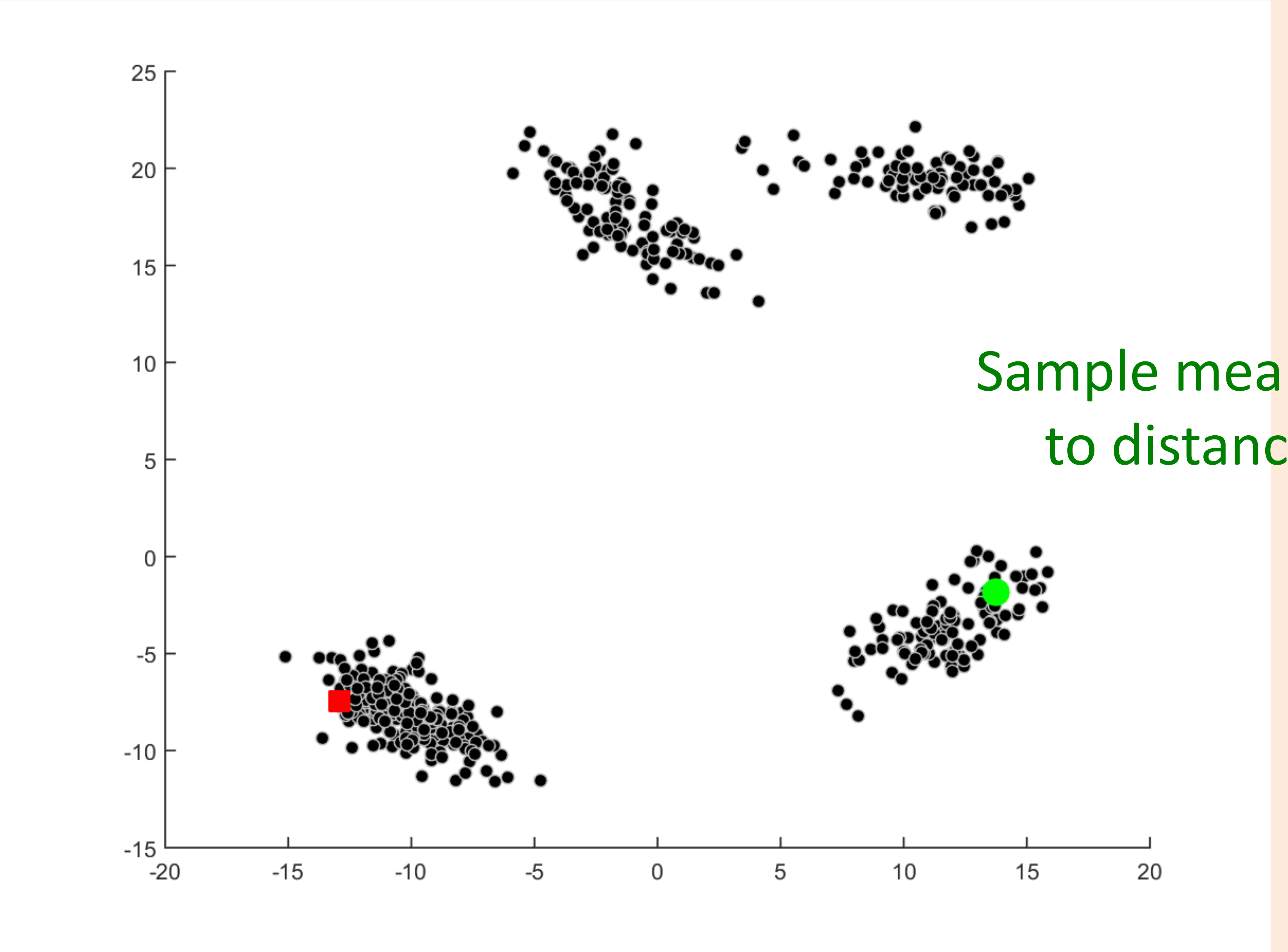
First mean is a random example.

K-Means++



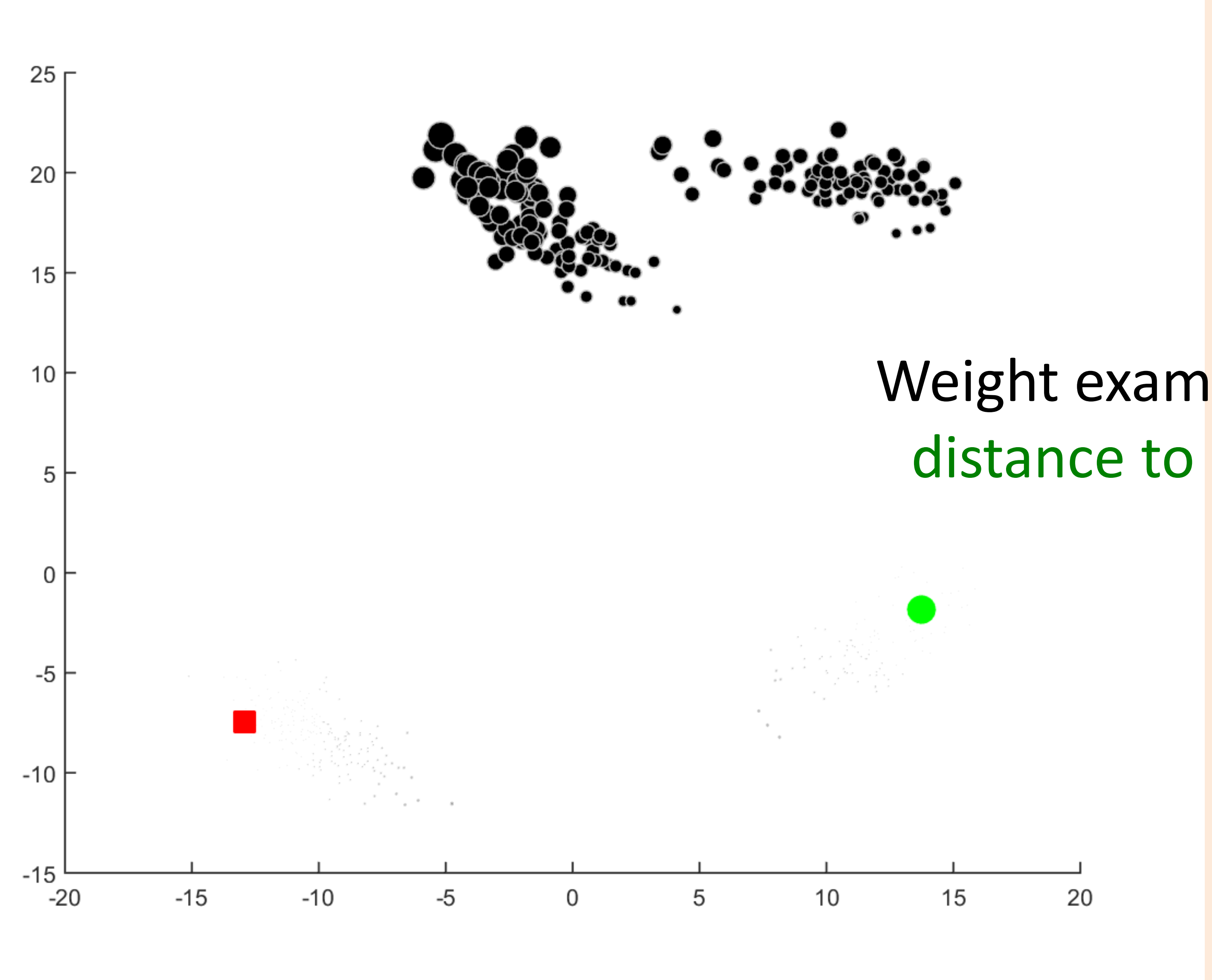
Weight examples by distance to mean squared.

K-Means++



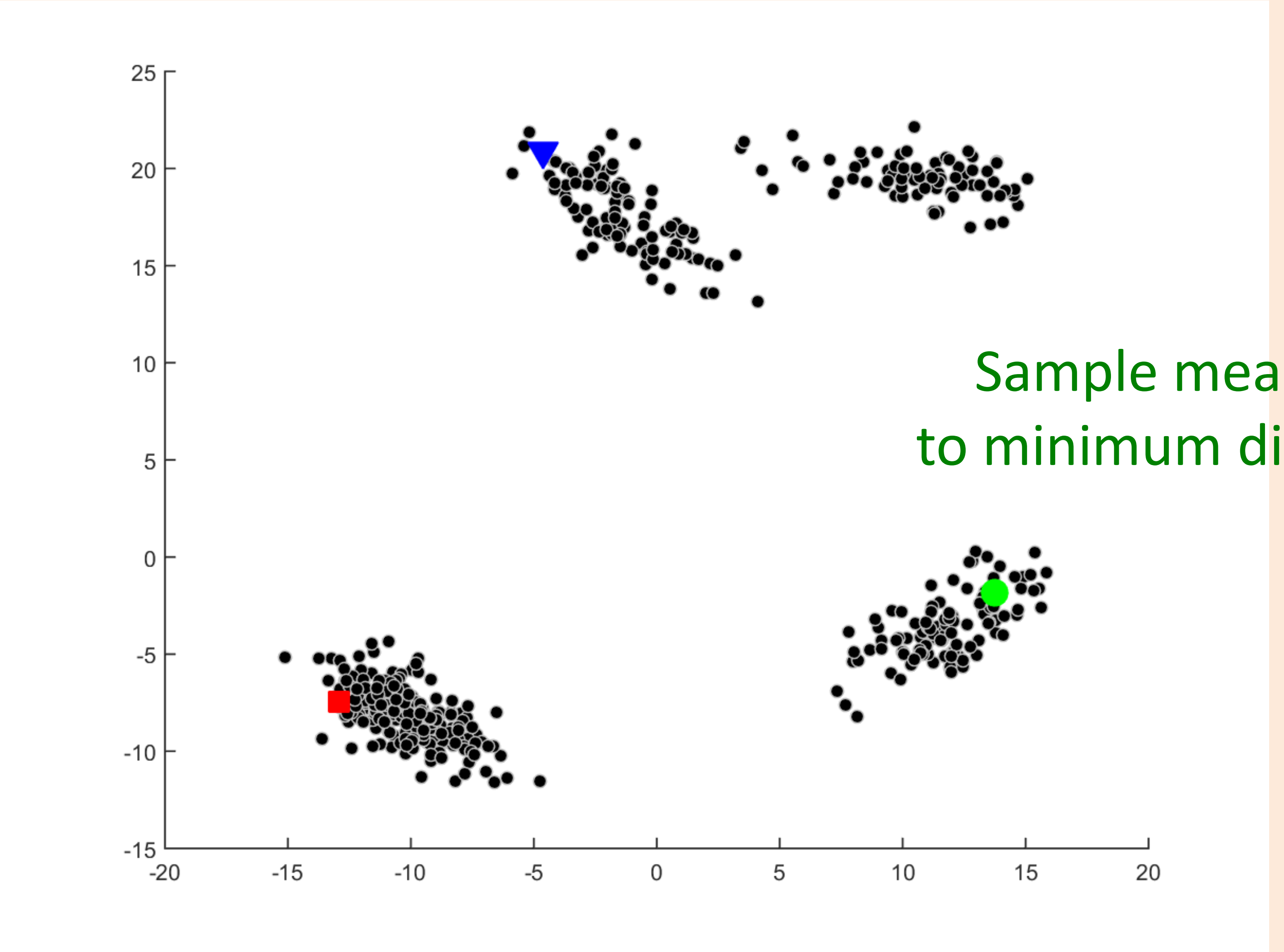
Sample mean proportional to distances squared.

K-Means++



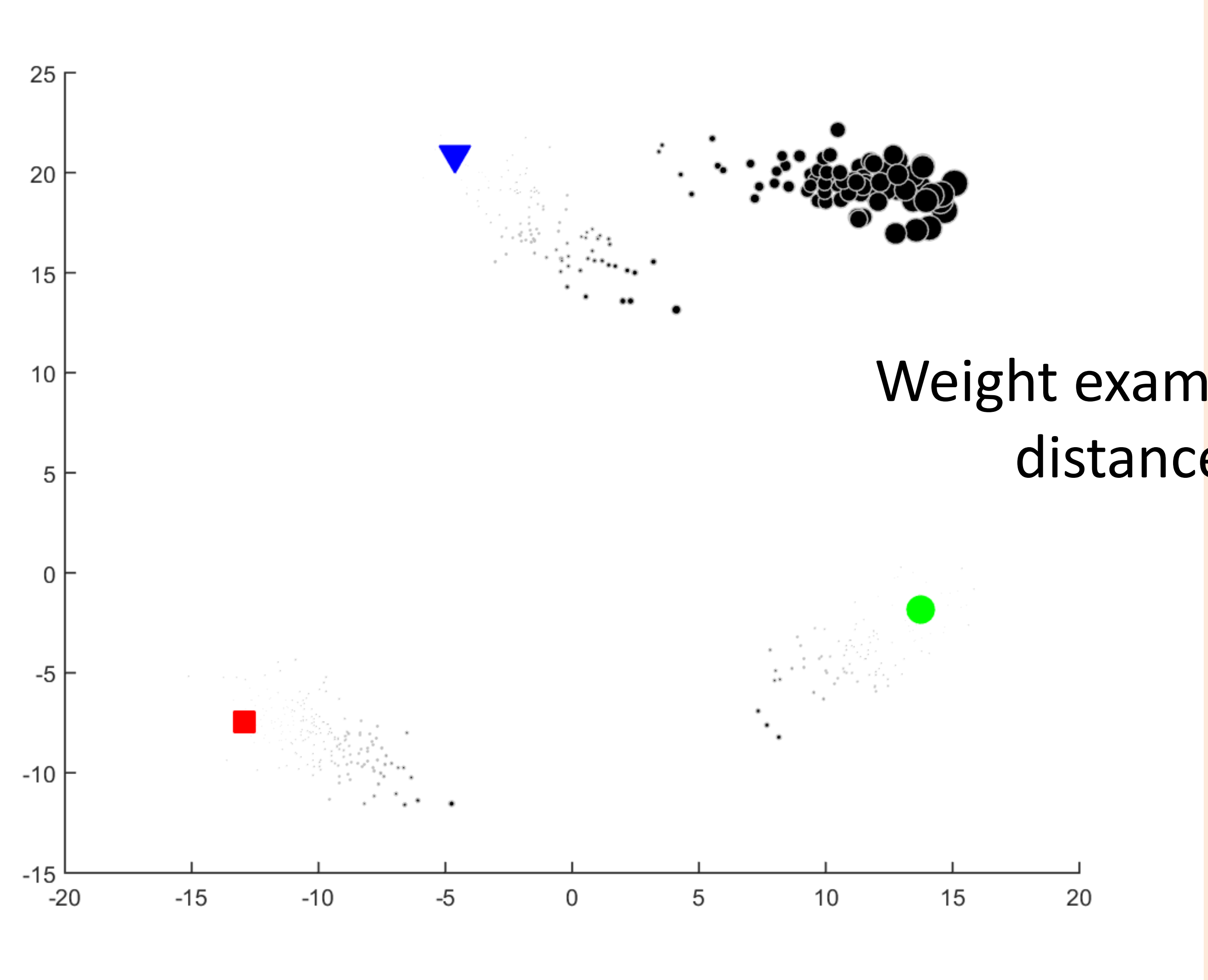
Weight examples by squared distance to nearest mean.

K-Means++



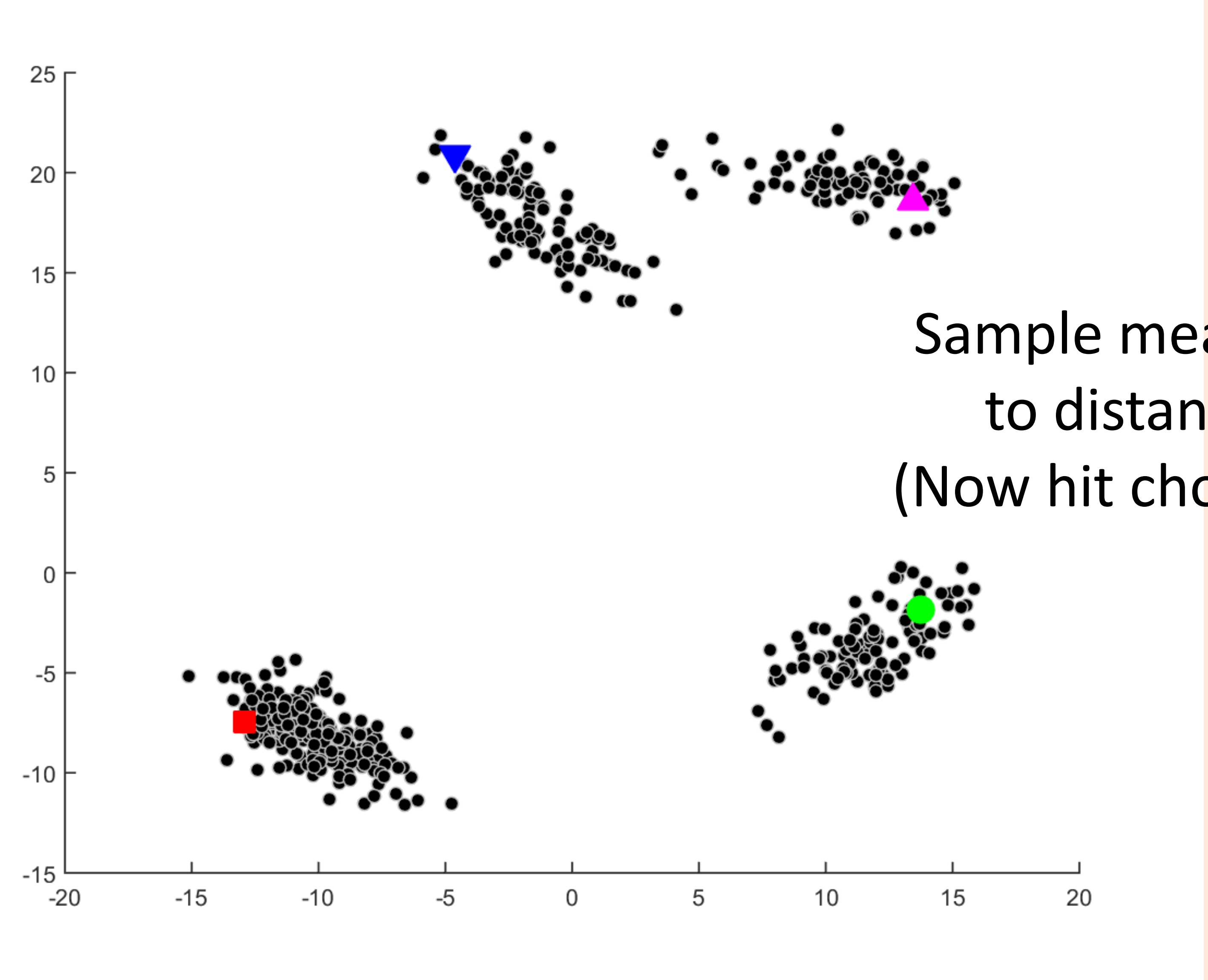
Sample mean proportional
to minimum distances squared

K-Means++



Weight examples by squared distance to mean.

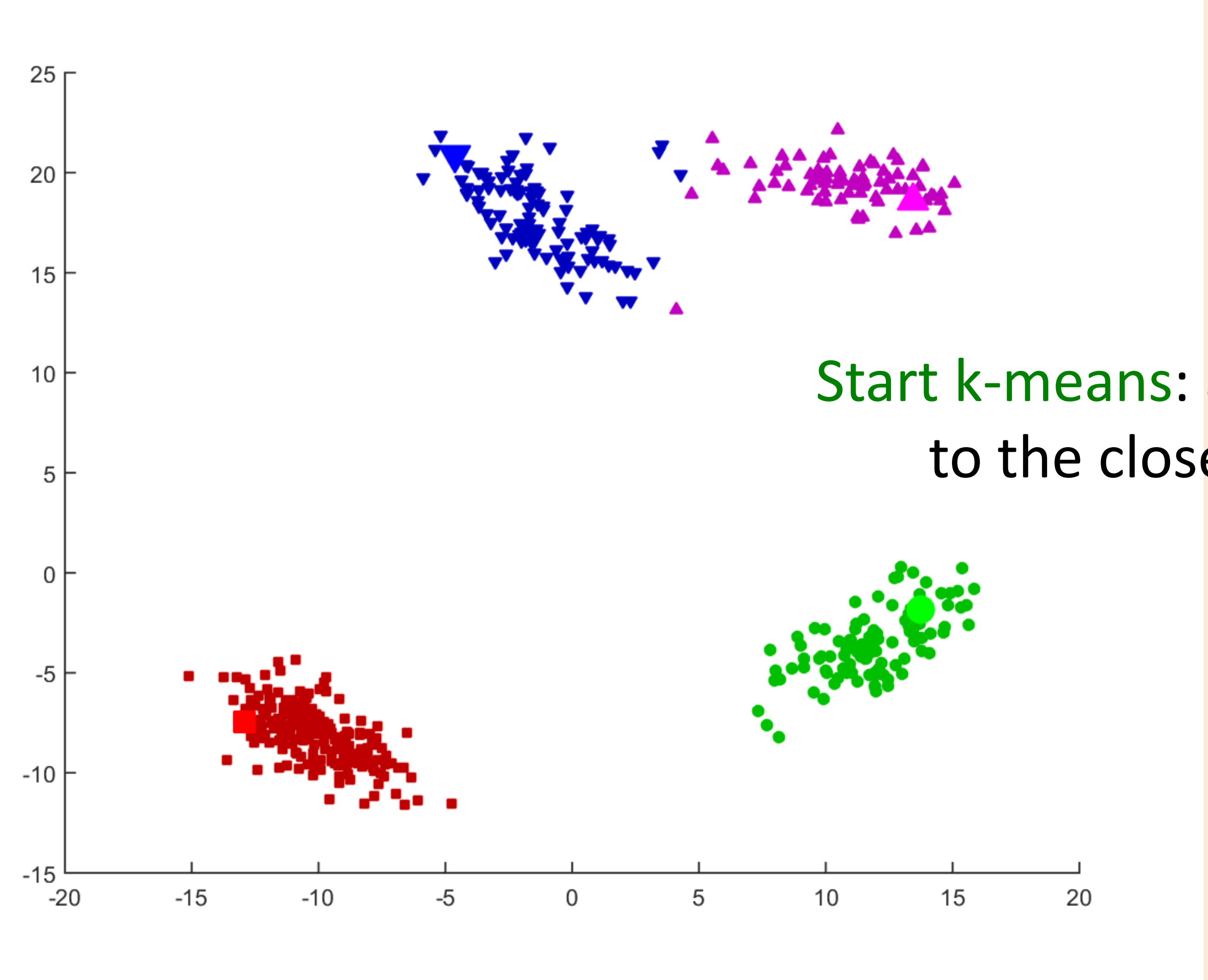
K-Means++



Sample mean proportional to distances squared.

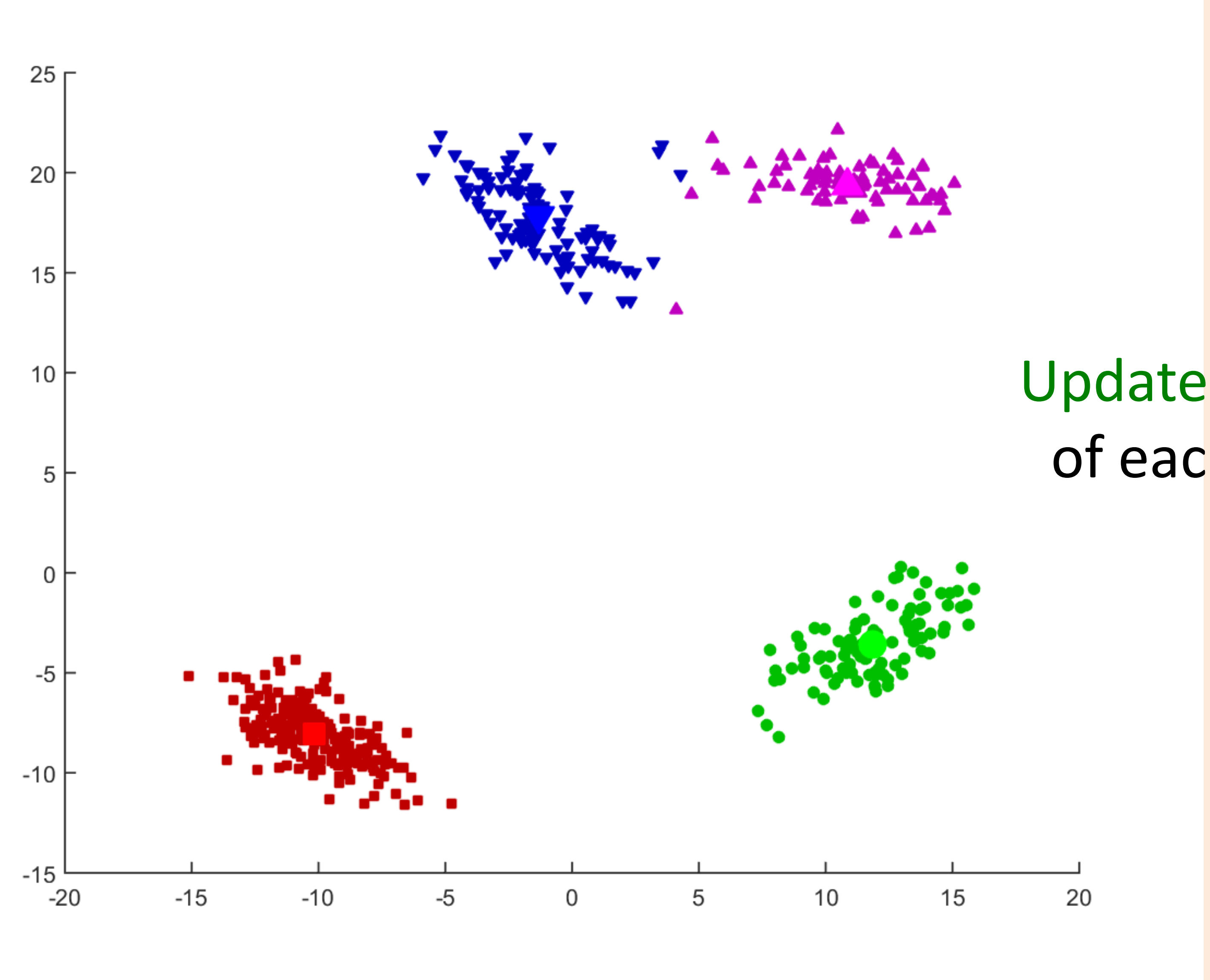
(Now hit chosen target k=4.)

K-Means++



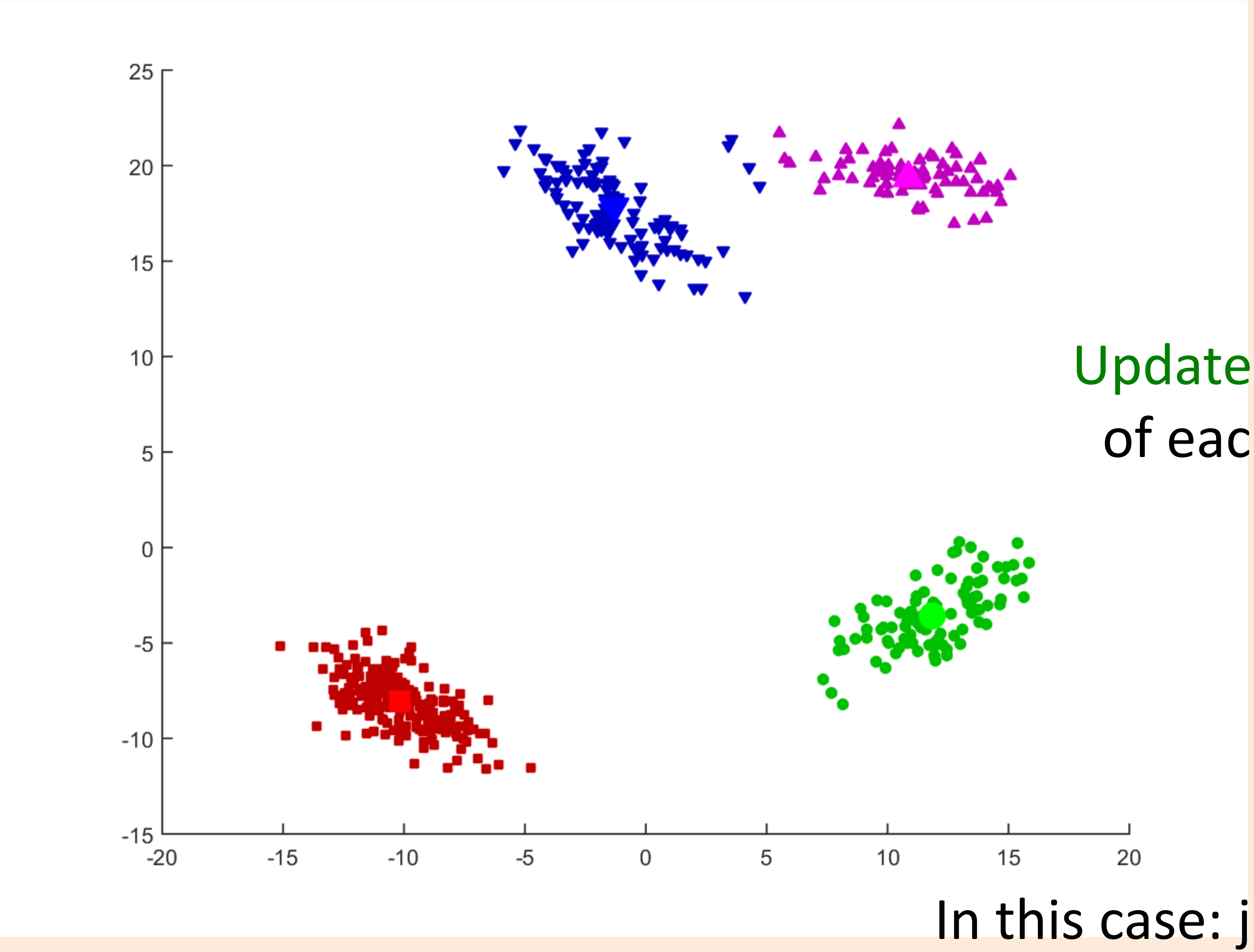
Start k-means: assign objects to the closest mean.

K-Means++



Update the mean
of each cluster.

K-Means++



Update the mean
of each cluster.

In this case: just 2 iterations!

Discussion of K-Means++

- Recall the objective function k-means tries to minimize:

$$f(\underset{\text{all means}}{w}, \underset{\text{all assigments}}{c}) = \sum_{i=1}^{n} \| x_i - w_{c(i)} \|_2^2$$

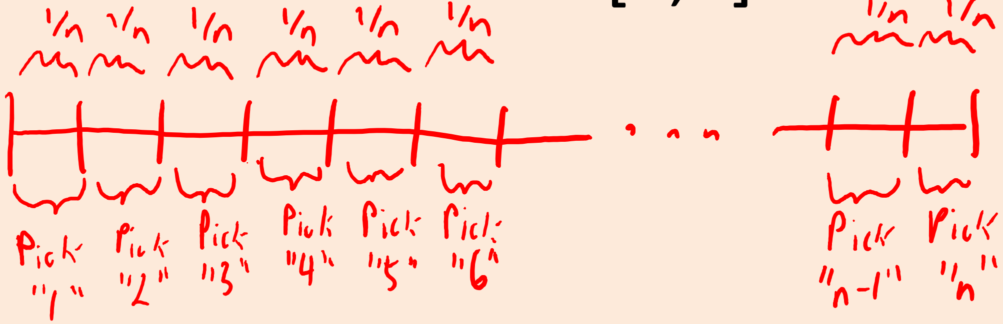
- The initialization of ‘W’ and ‘c’ given by k-means++ satisfies:

$$\frac{\underset{\text{expectation over random samples}}{E}[f(w,c)]}{f(\underbrace{w^*, c^*}_{\text{"Best" mean and clustering according to objective.}})} \le O(\log(k))$$

- Get good clustering with high probability by re-running.
- However, there is no guarantee that c^* is a good clustering.

Uniform Sampling

- Standard approach to generating a random number from $\{1,2,\ldots,n\}$:
  1. Generate a uniform random number ‘u’ in the interval [0,1].
  2. Return the largest index ‘i’ such that $u \leq i/n$.
- Conceptually, this divides interval [0,1] into ‘n’ equal-size pieces:

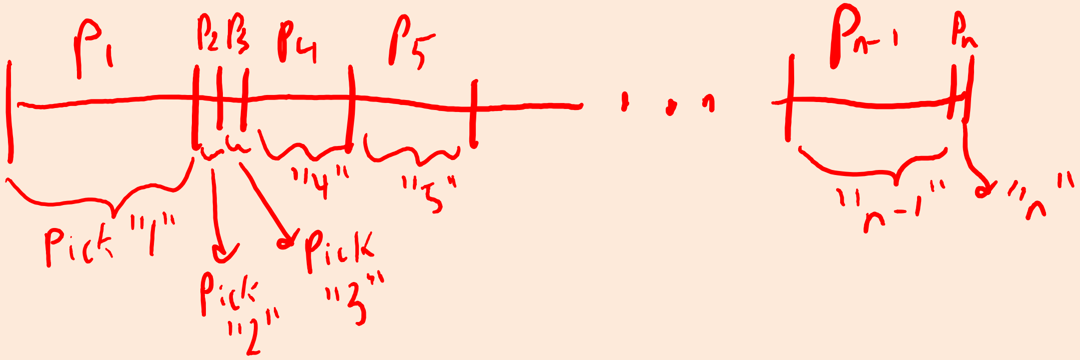


- This assumes $p_i = 1/n$ for all ‘i’.

probability of picking number 'i'.

Non-Uniform Sampling

- Standard approach to generating a random number for general p_i.
  1. Generate a uniform random number 'u' in the interval [0,1].
  2. Return the largest index 'i' such that u ≤ $\sum_{j=1}^{i} p_i$
- Conceptually, this divides interval [0,1] into non-equal-size pieces:



- Can sample from a generic discrete probability distribution in O(n).
- If you need to generate 'm' samples:
  – Cost is O(n + m log (n)) with binary search and storing cumulative sums.

How many iterations does k-means take?

- Each update of the ‘$\hat{y}_i$’ or ‘w_c’ does not increase the objective ‘f’.
- And there are k^n possible assignments of the $\hat{y}_i$ to ‘k’ clusters.
- So within k^n iterations you cannot improve the objective by changing $\hat{y}_i$, and the algorithm stops.

- Tighter-but-more-complicated “smoothed” analysis:
  - https://arxiv.org/pdf/0904.1113.pdf