

# 10-301/601: Introduction to Machine Learning

## Lecture 24: Clustering & Bagging

Hoda Heidari, Henry Chai & Matt Gormley

4/15/24

# Front Matter

- Announcements
  - HW8 released 4/8, due 4/19 (Friday) at 11:59 PM
  - HW9 released 4/19 (Friday), due 4/25 at 11:59 PM
    - HW9 is a written-only homework
    - **You may only use at most 2 late days on HW9**

# Clustering

- Goal: split an *unlabeled* data set into groups or clusters of “similar” data points

- Use cases:

- Organizing data
- Discovering patterns or structure ★
- Preprocessing for downstream machine learning tasks

- Applications:

- customer segmentation / recommender system
- image segmentation / object detection
- topic identification
-

# Recall: Similarity for $k$ NN

- Intuition: ~~predict the label of a data point to be the label of the “most similar” training point~~ two points are “similar” if the distance between them is small
- Euclidean distance:  $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2$

# Partition-Based Clustering

- Given a desired number of clusters,  $K$ , return a partition of the data set into  $K$  groups or clusters,  $\{C_1, \dots, C_K\}$ , that optimize some objective function
  1. What objective function should we optimize?
  2. How can we perform optimization in this setting?



# Example Clusterings



Option A



Option B



# Example Clusterings



A.



B. TOXIC



C.



Poll Question 1: Which option do you prefer?

Poll Question 2: Justify your response to the previous question



# Recipe for K-means

- Define a model and model parameters

– Assume  $K$  clusters

– Use the Euclidean distance

– Parameters: cluster center  $\{ \mu_1, \dots, \mu_K \}$ , cluster assignment  $\{ z^{(1)}, \dots, z^{(N)} \}$

- Write down an objective function

$$\text{minimize } \sum_{n=1}^N \|x^{(n)} - \mu_{z^{(n)}}\|_2$$

- Optimize the objective w.r.t. the model parameters

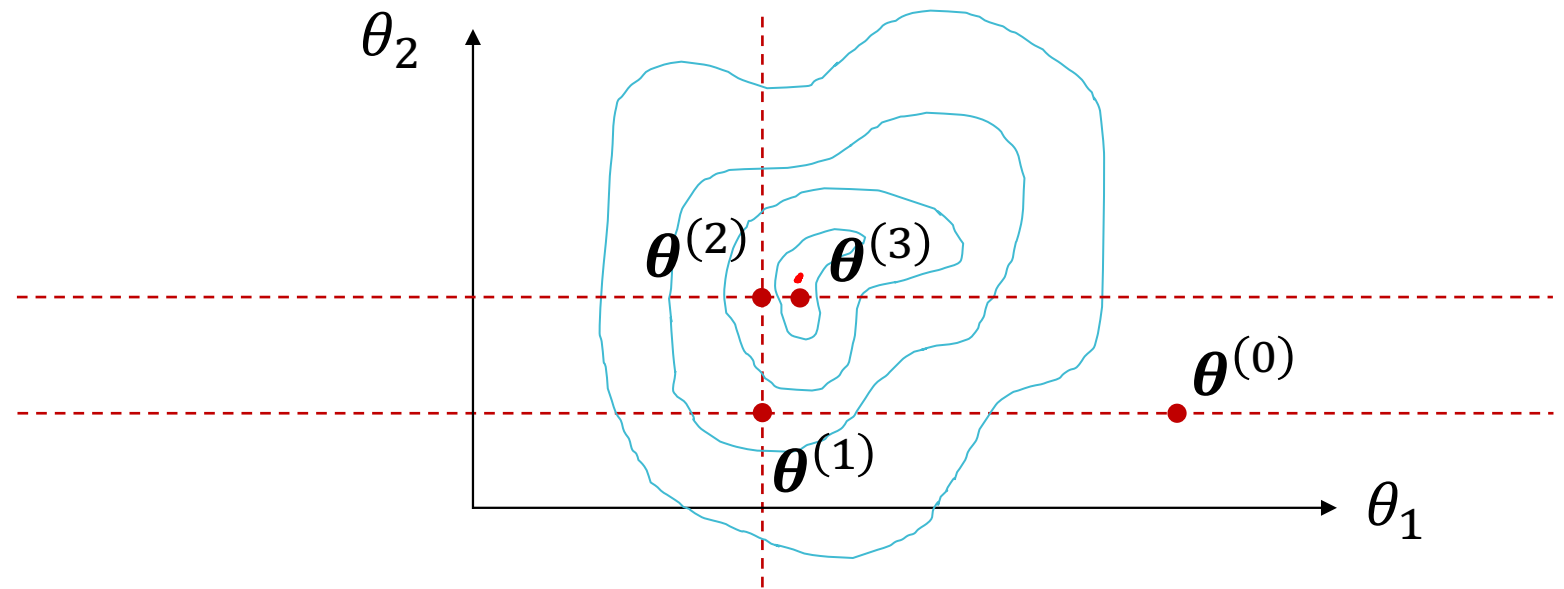
(Block) coordinate descent

# Coordinate Descent

- Goal: minimize some objective

$$\hat{\theta} = \operatorname{argmin} J(\theta)$$

- Idea: iteratively pick one variable and minimize the objective w.r.t. just that variable, *keeping all others fixed*.



# Block Coordinate Descent

- Goal: minimize some objective

$$\hat{\alpha}, \hat{\beta} = \operatorname{argmin} J(\alpha, \beta)$$

- Idea: iteratively pick one *block* of variables ( $\alpha$  or  $\beta$ ) and minimize the objective w.r.t. that block, keeping the other(s) fixed.
  - Ideally, blocks should be the largest possible set of variables *that can be efficiently optimized simultaneously*

# Optimizing the $K$ -means objective

$$\hat{\mu}_1, \dots, \hat{\mu}_K, z^{(1)}, \dots, z^{(N)} = \operatorname{argmin} \underbrace{\sum_{n=1}^N \|x^{(n)} - \mu_{z^{(n)}}\|_2}_{\text{objective}}$$

- If  $\mu_1, \dots, \mu_K$  are fixed

$$z^{(n)} = \operatorname{argmin}_{z \in \{1, \dots, K\}} \|x^{(n)} - \mu_z\|_2$$

- If  $z^{(1)}, \dots, z^{(N)}$  are fixed

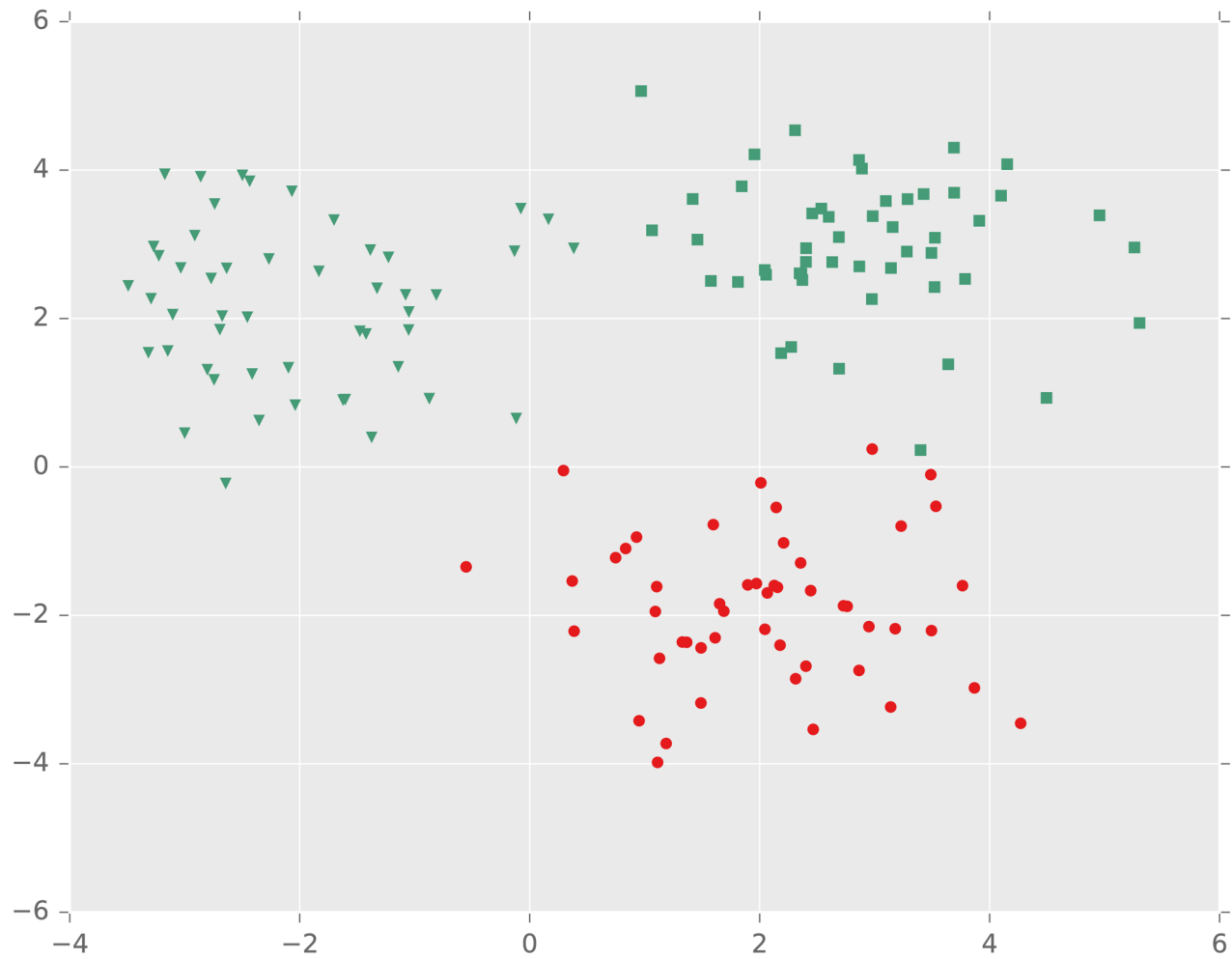
$$\hat{\mu}_k = \frac{1}{N_k} \sum_{n: z^{(n)} = k} x^{(n)}$$

where  $N_k$  is the # of data points in cluster  $k$

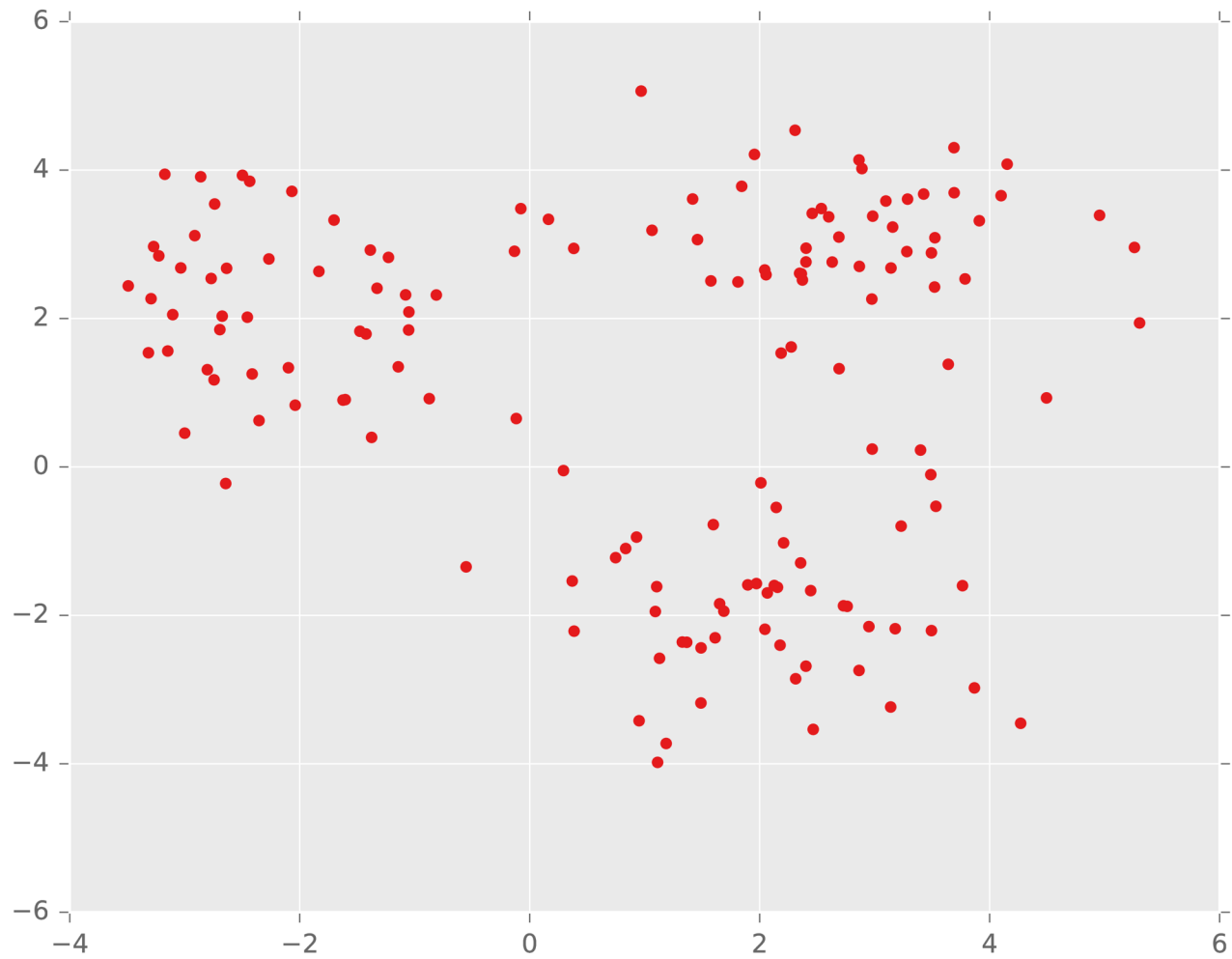
# K-means Algorithm

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(n)})\}_{n=1}^N, K$ 
  1. Initialize cluster centers  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$
  2. While NOT CONVERGED
    - a. Assign each data point to the cluster with the nearest cluster center:
$$z^{(n)} = \underset{k}{\operatorname{argmin}} \|\mathbf{x}^{(n)} - \boldsymbol{\mu}_k\|_2$$
    - b. Recompute the cluster centers:
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n: z^{(n)}=k} \mathbf{x}^{(n)}$$
where  $N_k$  is the number of data points in cluster  $k$
- Output: cluster centers  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$  and cluster assignments  $z^{(1)}, \dots, z^{(N)}$

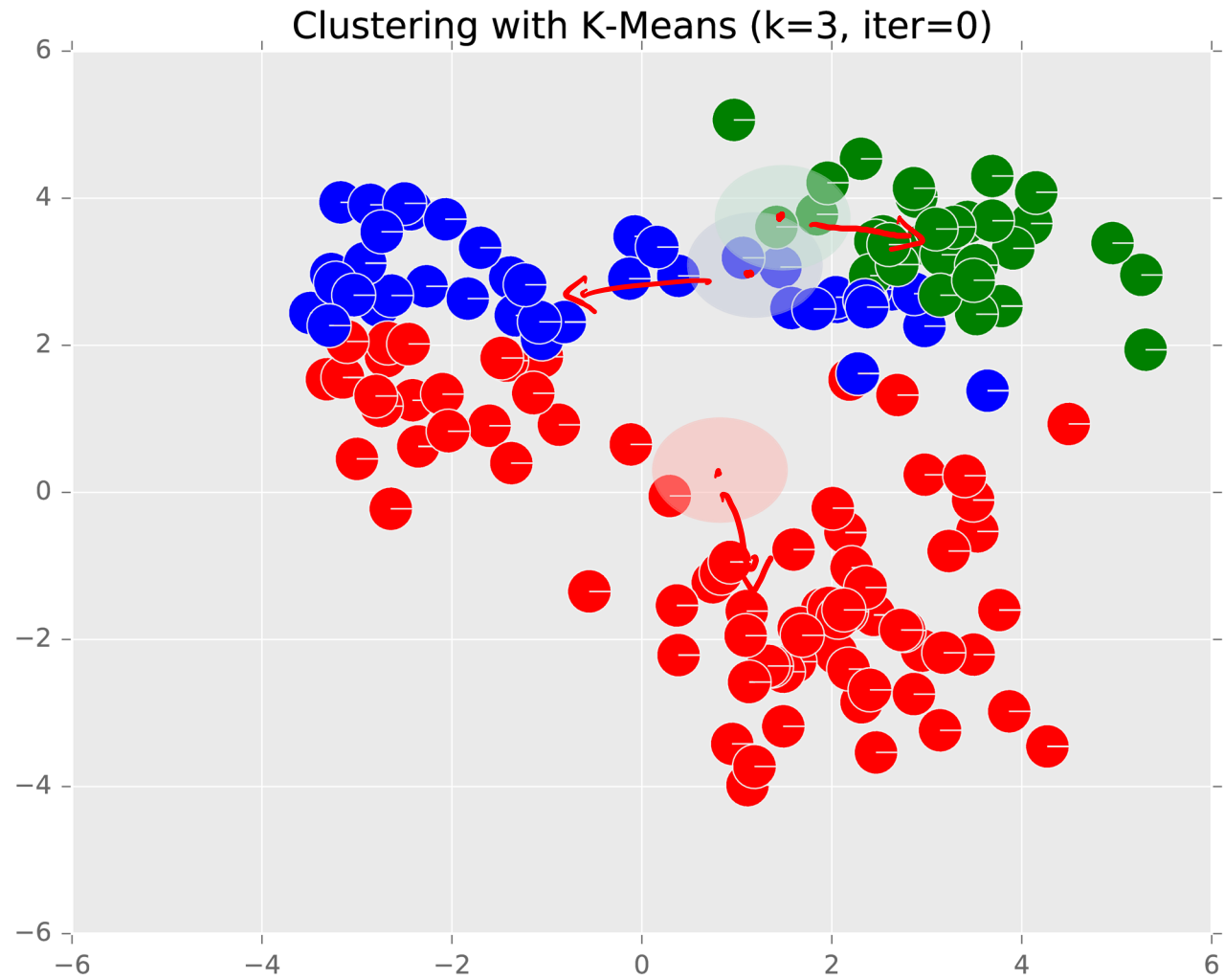
# $K$ -means: Example ( $K = 3$ )



# $K$ -means: Example ( $K = 3$ )

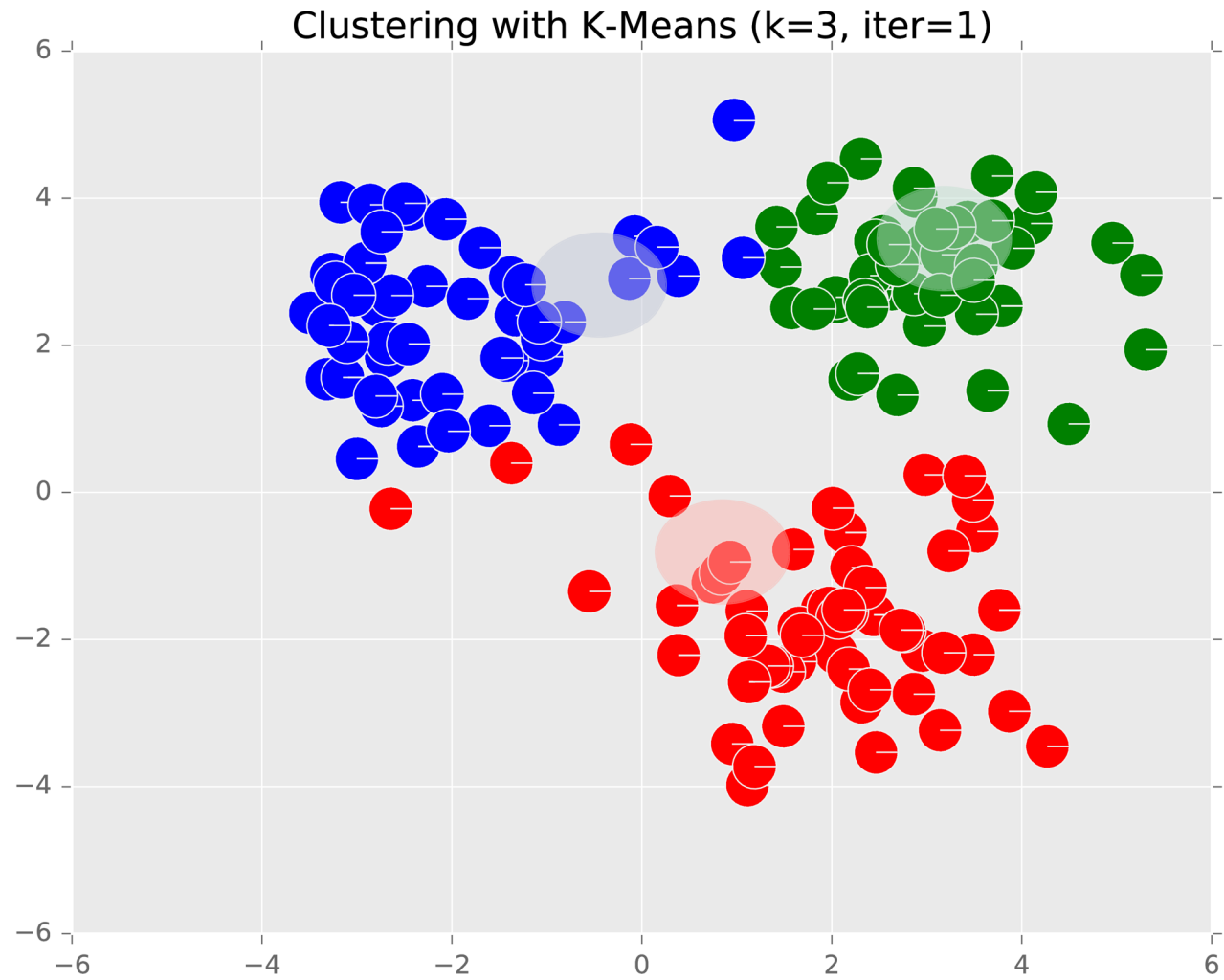


*K*-means:  
Example  
( $K = 3$ )

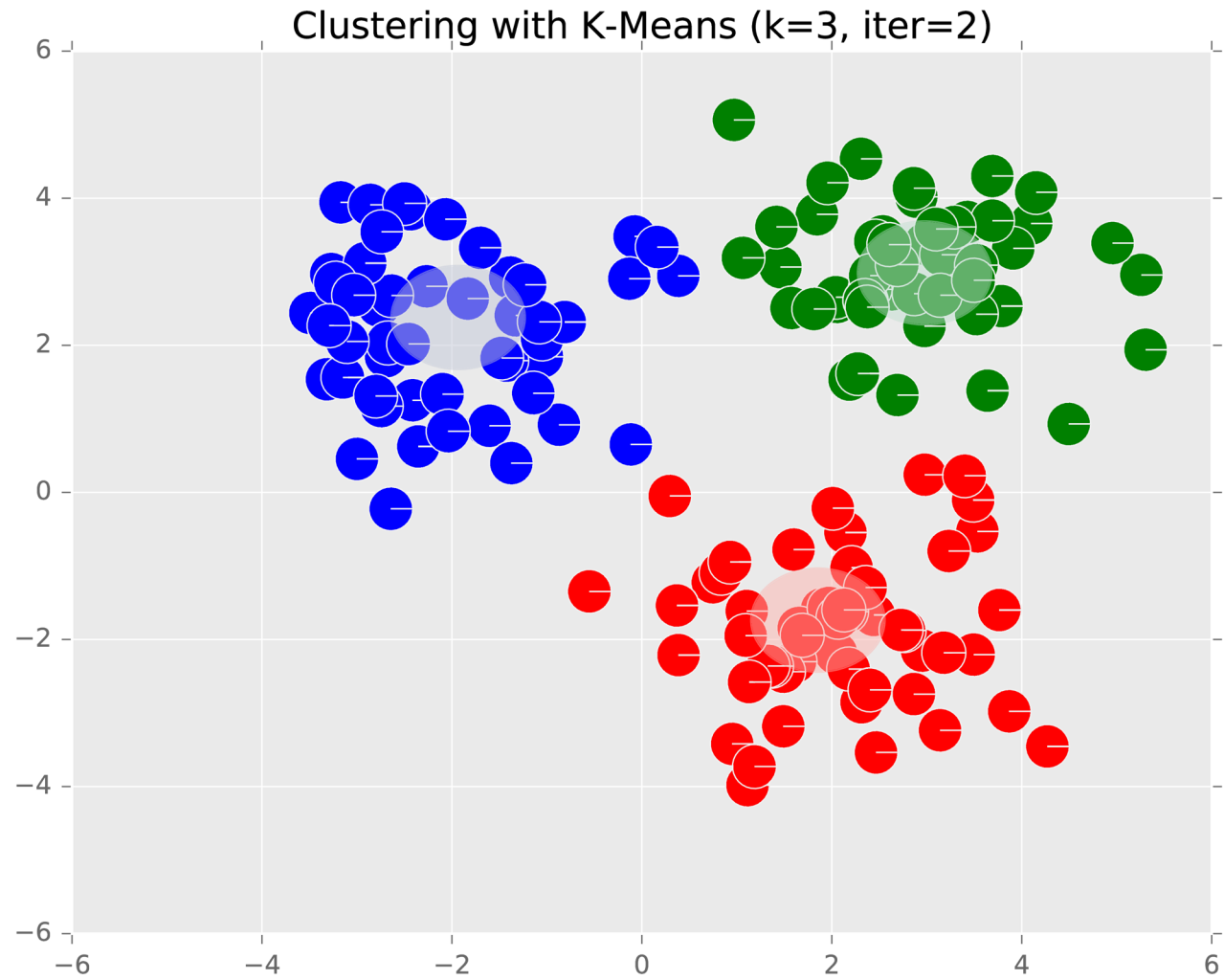




*K*-means:  
Example  
( $K = 3$ )



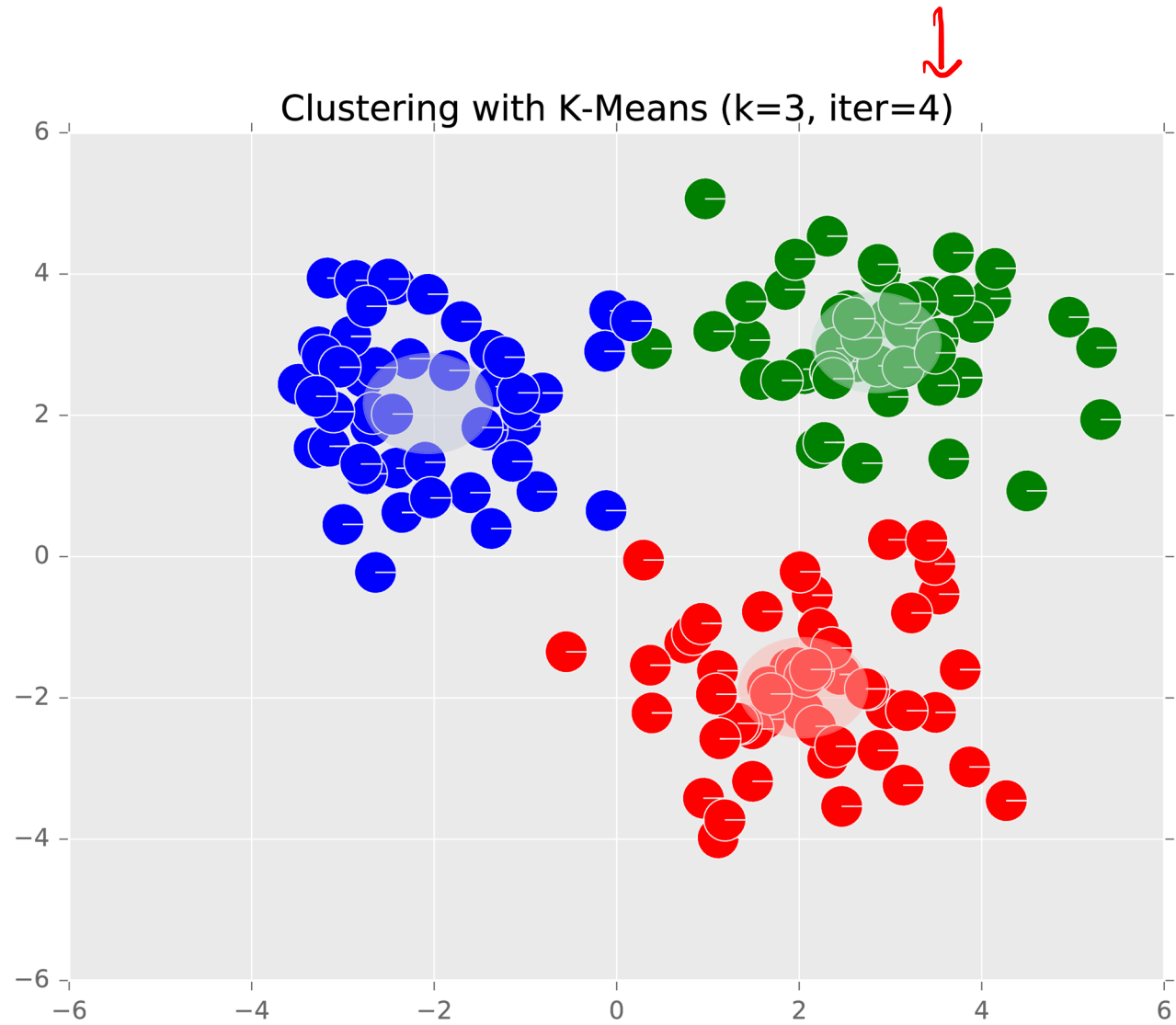
*K*-means:  
Example  
( $K = 3$ )



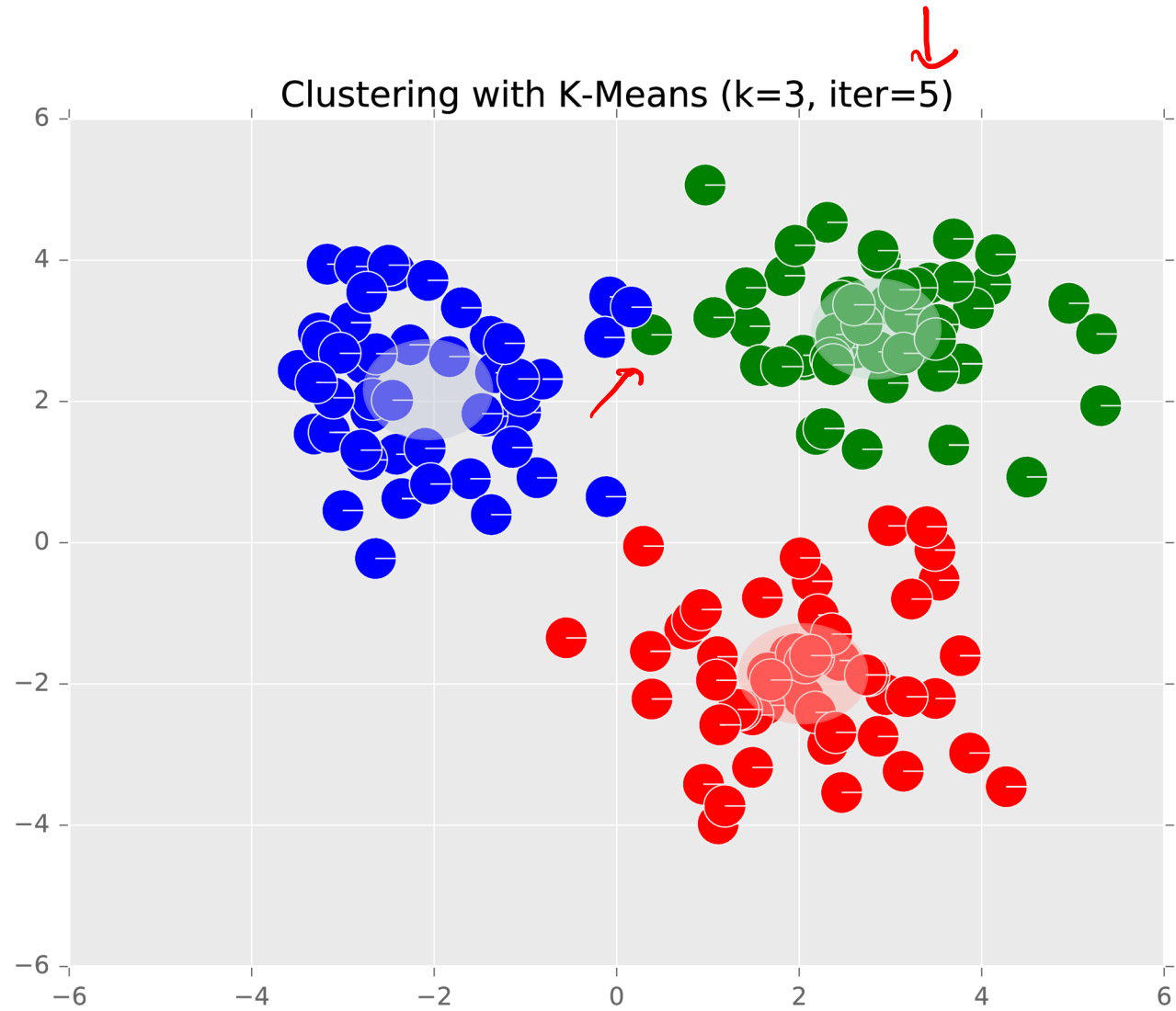
*K*-means:  
Example  
( $K = 3$ )



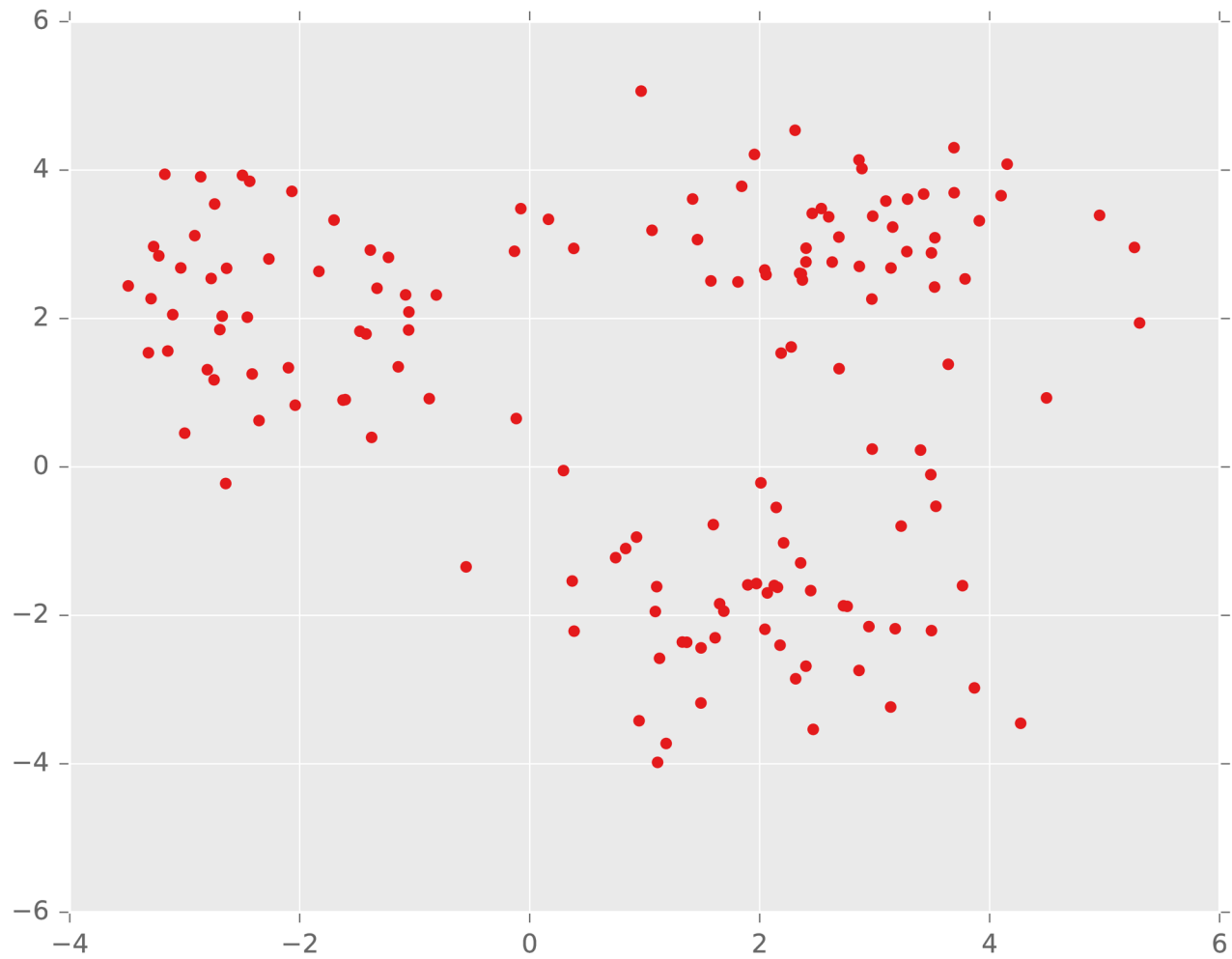
*K*-means:  
Example  
( $K = 3$ )



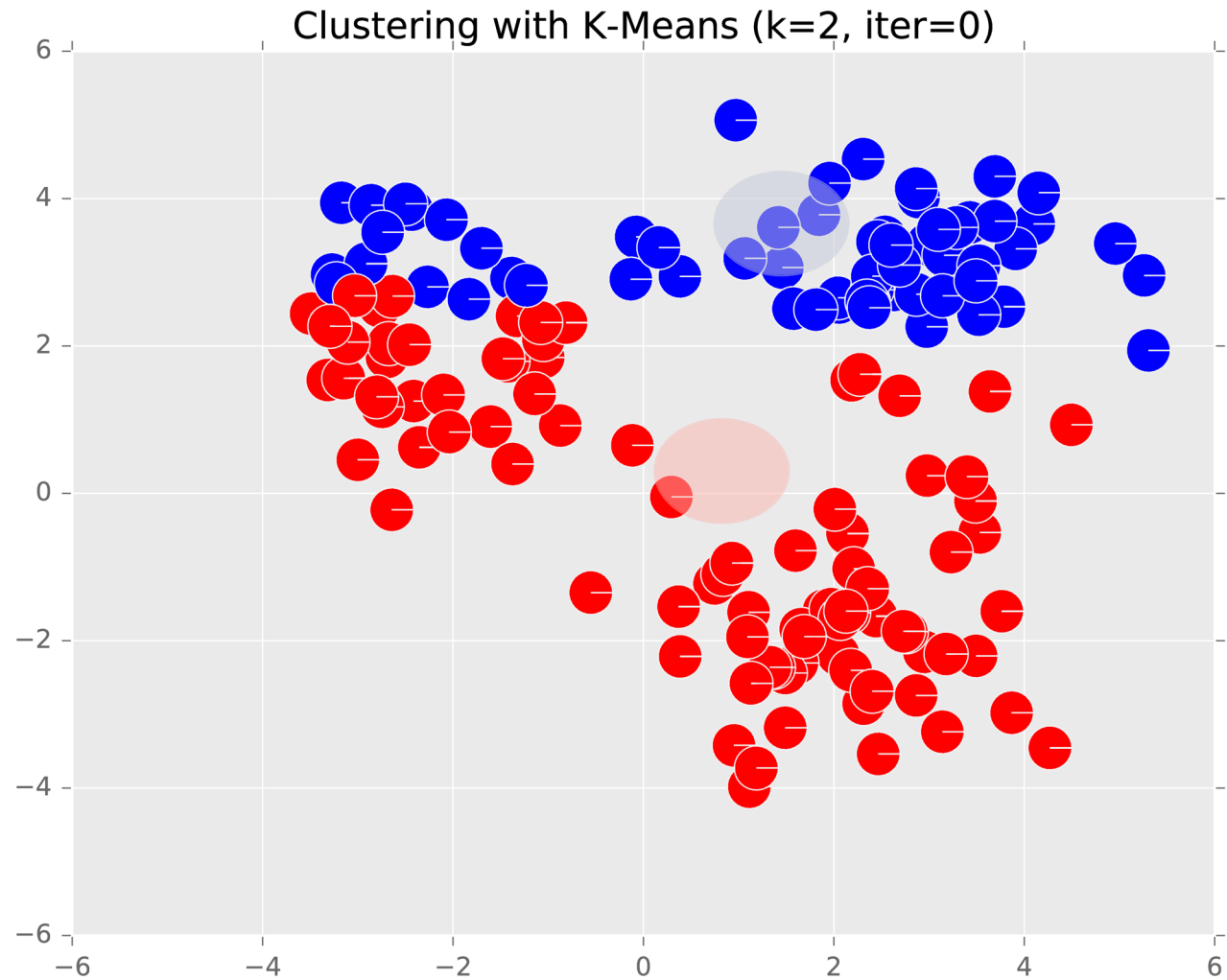
*K*-means:  
Example  
( $K = 3$ )



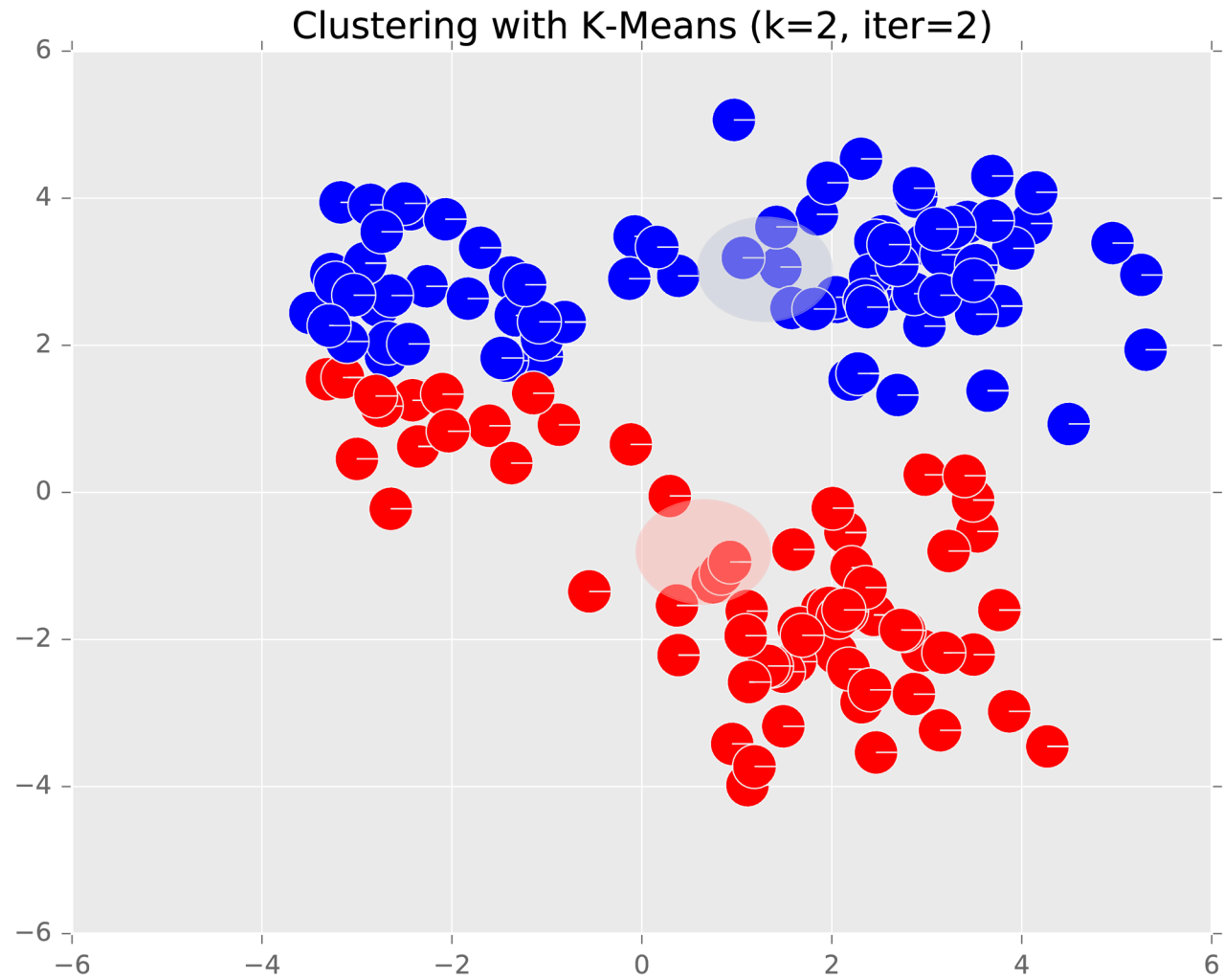
# $K$ -means: Example ( $K = 2$ )



*K*-means:  
Example  
( $K = 2$ )

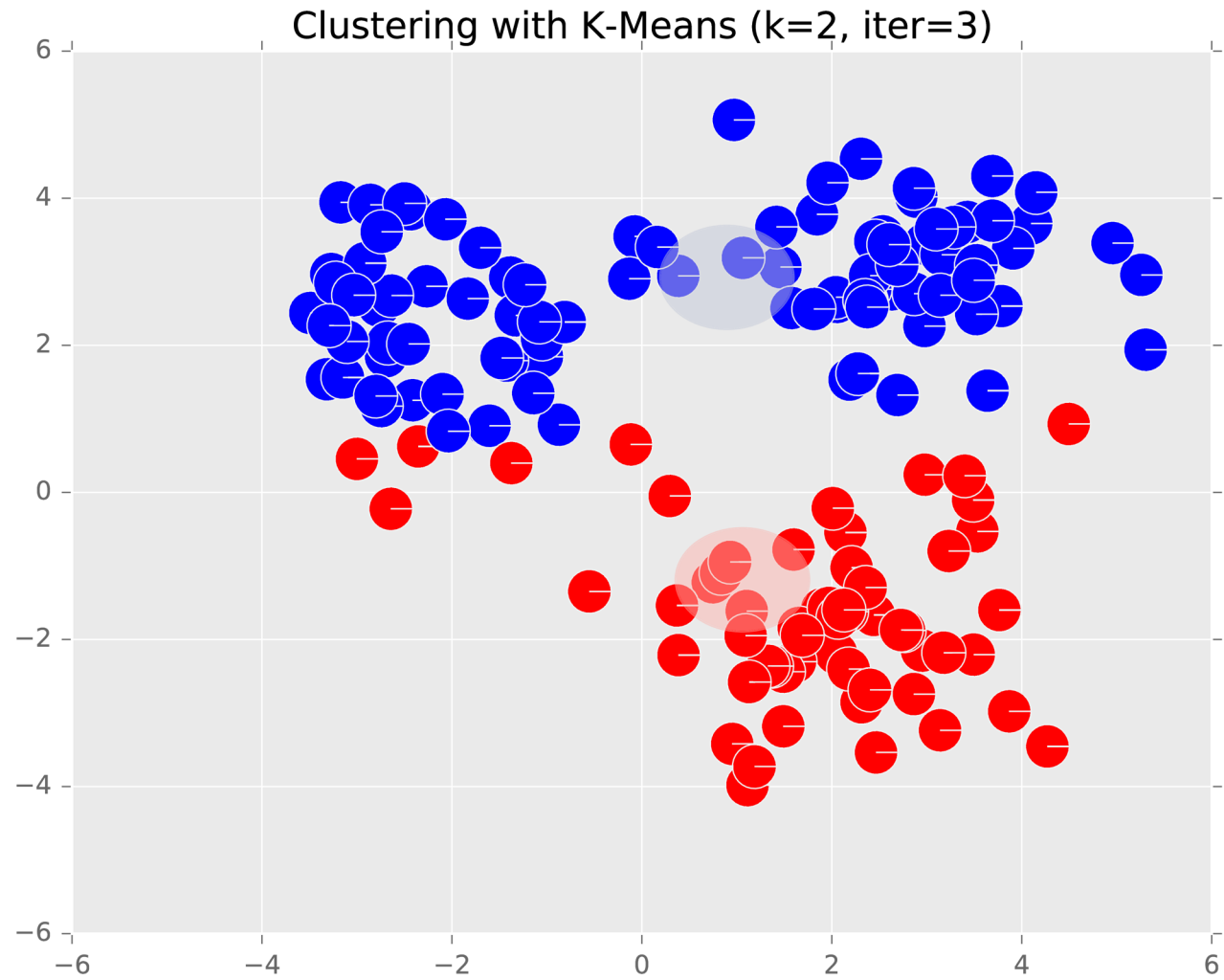


*K*-means:  
Example  
( $K = 2$ )

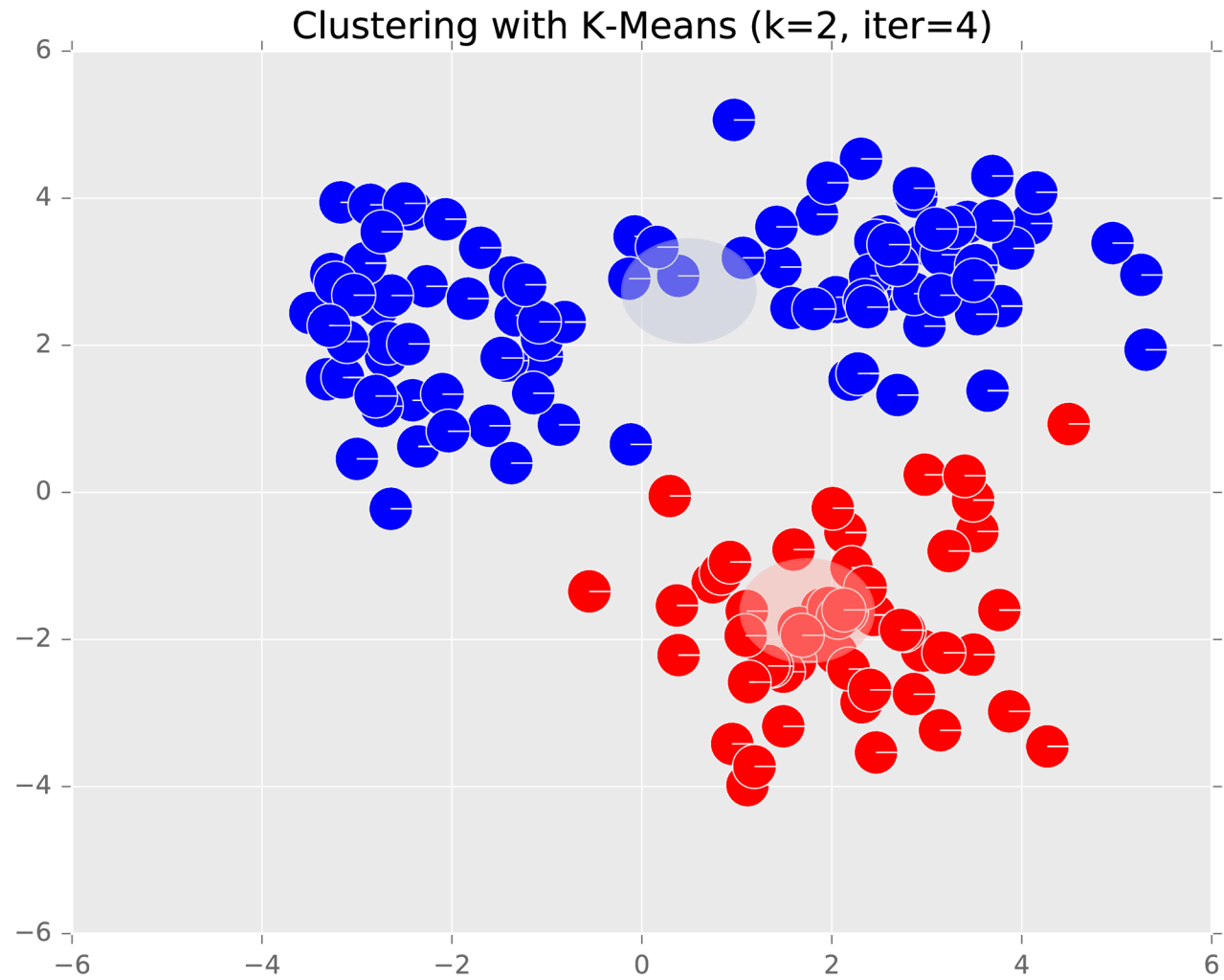




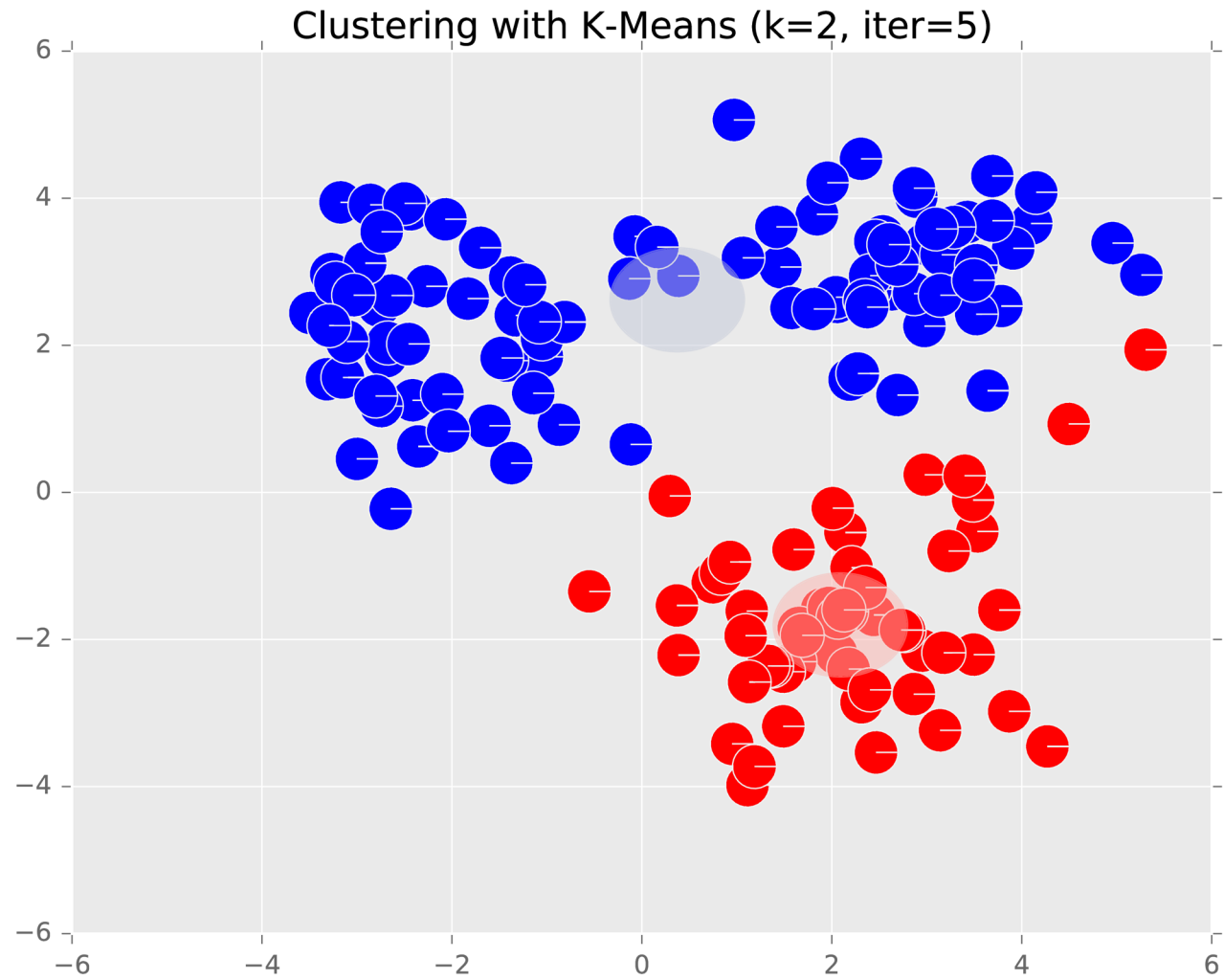
*K*-means:  
Example  
( $K = 2$ )



*K*-means:  
Example  
( $K = 2$ )



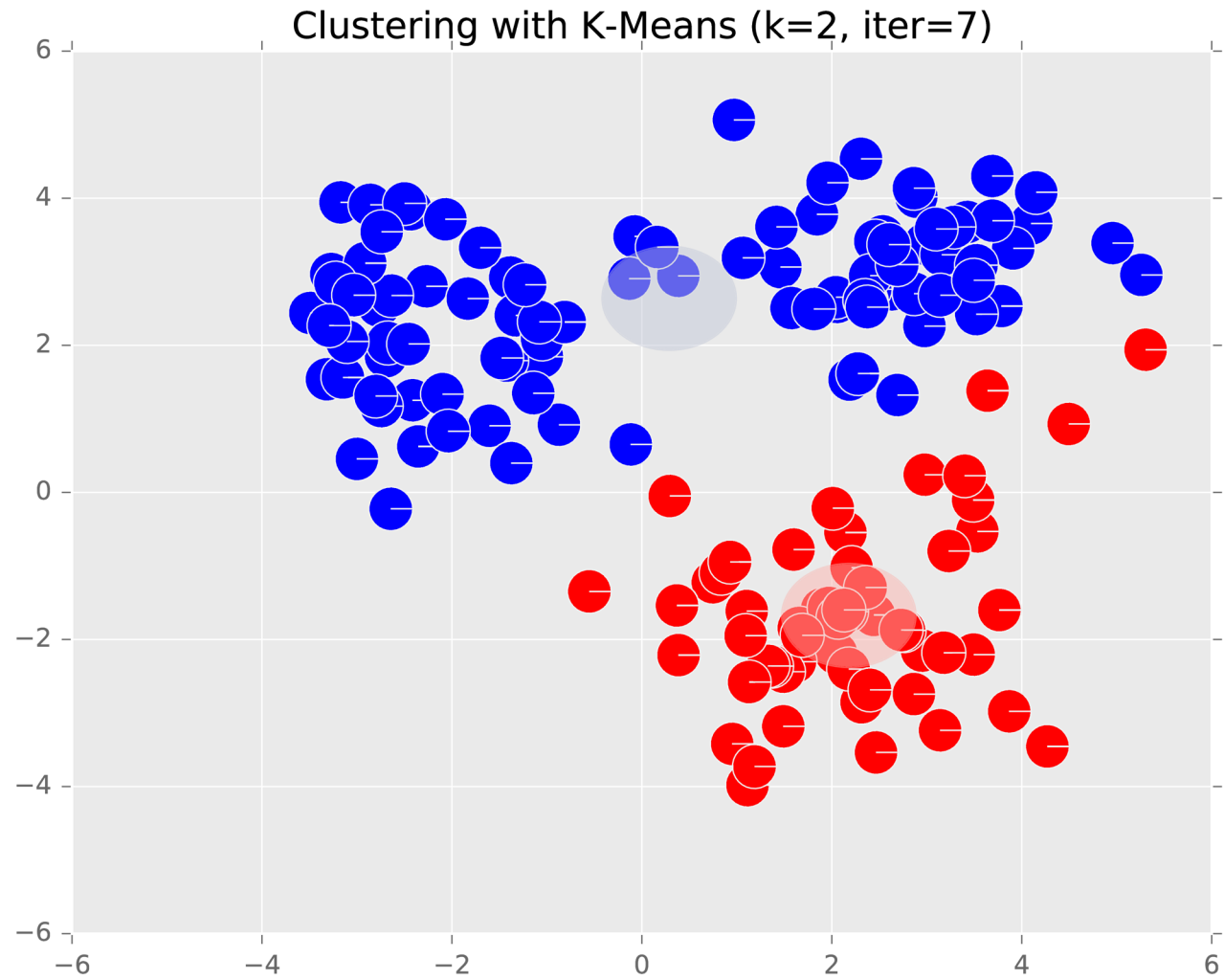
*K*-means:  
Example  
( $K = 2$ )



*K*-means:  
Example  
( $K = 2$ )



*K*-means:  
Example  
( $K = 2$ )

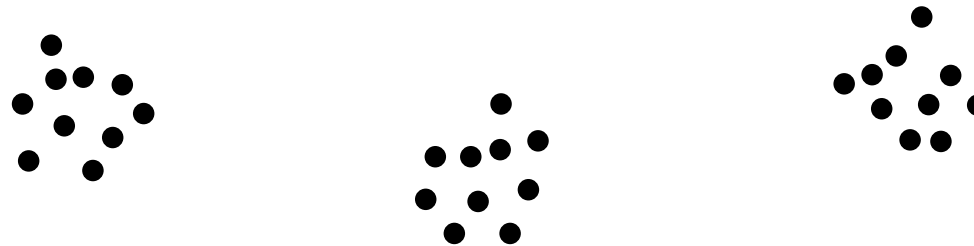


## Setting $K$

- Idea: choose the value of  $K$  that minimizes the objective function

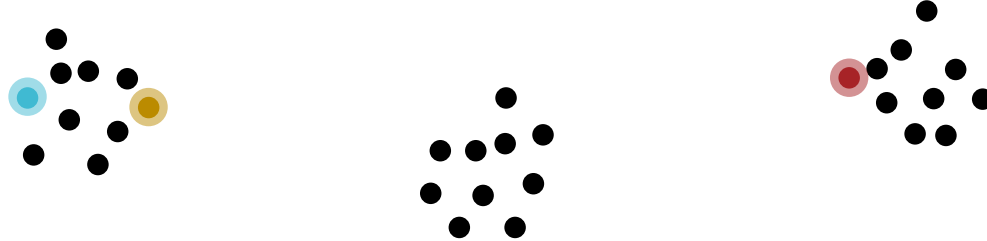
# Initializing $K$ -means

- Common choice: choose  $K$  data points at random to be the initial cluster centers (Lloyd's method)



# Initializing $K$ -means

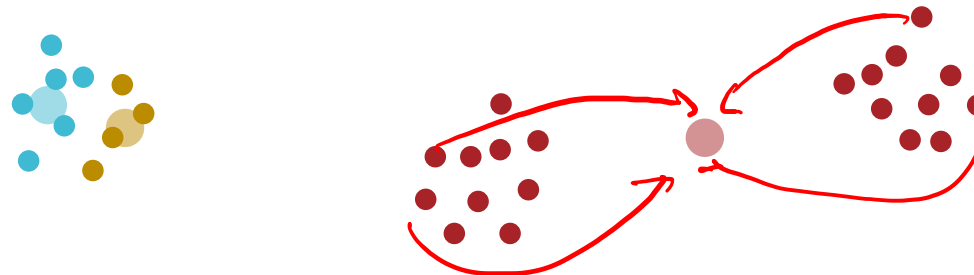
- Common choice: choose  $K$  data points at random to be the initial cluster centers (Lloyd's method)





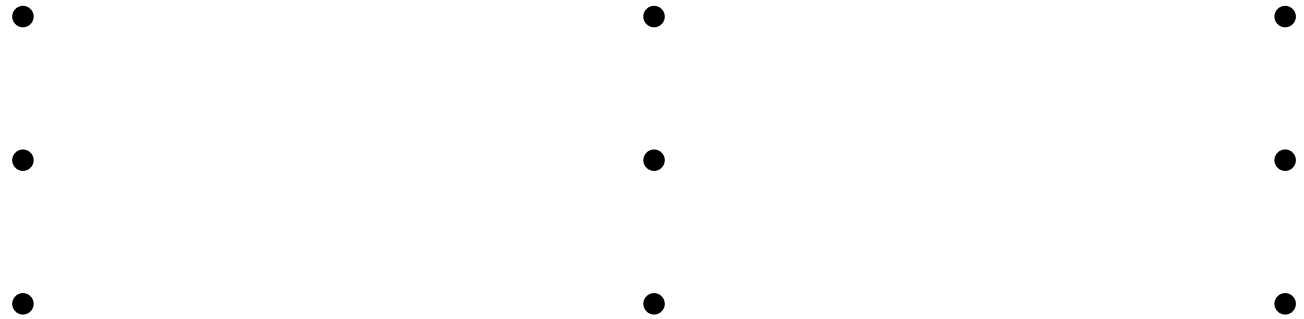
# Initializing $K$ -means

- Common choice: choose  $K$  data points at random to be the initial cluster centers (Lloyd's method)



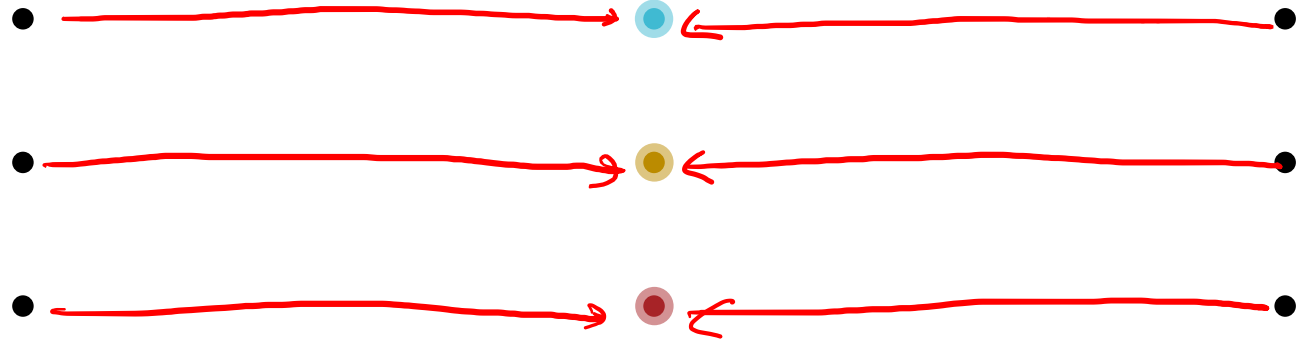
# Initializing $K$ -means

- Common choice: choose  $K$  data points at random to be the initial cluster centers (Lloyd's method)



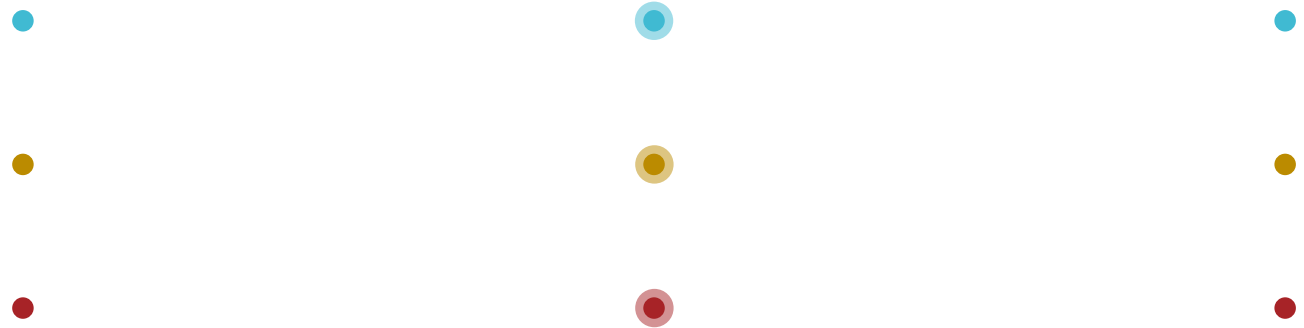
# Initializing $K$ -means

- Common choice: choose  $K$  data points at random to be the initial cluster centers (Lloyd's method)



# Initializing $K$ -means

- Common choice: choose  $K$  data points at random to be the initial cluster centers (Lloyd's method)



- Lloyd's method converges to a local minimum and that local minimum can be arbitrarily bad (relative to the optimal clusters)
  - This is because the  $K$ -means objective is nonconvex!
- Intuition: want initial cluster centers to be far apart from one another

# $K$ -means++ (Arthur and Vassilvitskii, 2007)

1. Choose the first cluster center randomly from the data points.
2. For each other data point  $\mathbf{x}$ , compute  $D(\mathbf{x})$ , the distance between  $\mathbf{x}$  and the nearest cluster center.
3. Select the next cluster center proportional to  $D(\mathbf{x})^2$ .
4. Repeat 2 and 3  $K - 1$  times.
  - $K$ -means++ achieves a  $O(\log K)$  approximation to the optimal clustering in expectation
  - Both Lloyd's method and  $K$ -means++ can benefit from multiple random restarts.

# *K*-means Learning Objectives

- You should be able to...
  1. Distinguish between coordinate descent and block coordinate descent
  2. Define an objective function that gives rise to a "good" clustering
  3. Apply block coordinate descent to an objective function preferring each point to be close to its nearest objective function to obtain the K-Means algorithm
  4. Implement the K-Means algorithm
  5. Connect the non-convexity of the K-Means objective function with the (possibly) poor performance of random initialization

# The Netflix Prize



**Netflix Prize**

Home Rules Leaderboard Update

- 500,000 users
- 20,000 movies
- 100 million ratings
- Goal: To obtain lower error than Netflix's existing system on 3 million held out ratings

**Congratulations!**

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

On September 21, 2009 we awarded the \$1M Grand Prize to team "BellKor's Pragmatic Chaos". Read about [their algorithm](#), checkout team scores on the [Leaderboard](#), and join the discussions on the [Forum](#).

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.

# The Netflix Prize

## Netflix Prize

COMPLETED

[Home](#) [Rules](#) [Leaderboard](#) [Update](#) [Download](#)










### Leaderboard

Showing Test Score. [Click here to show quiz score](#)

Display top  leaders.

Rank Team Name Best Test Score % Improvement Best Submit Time

Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos

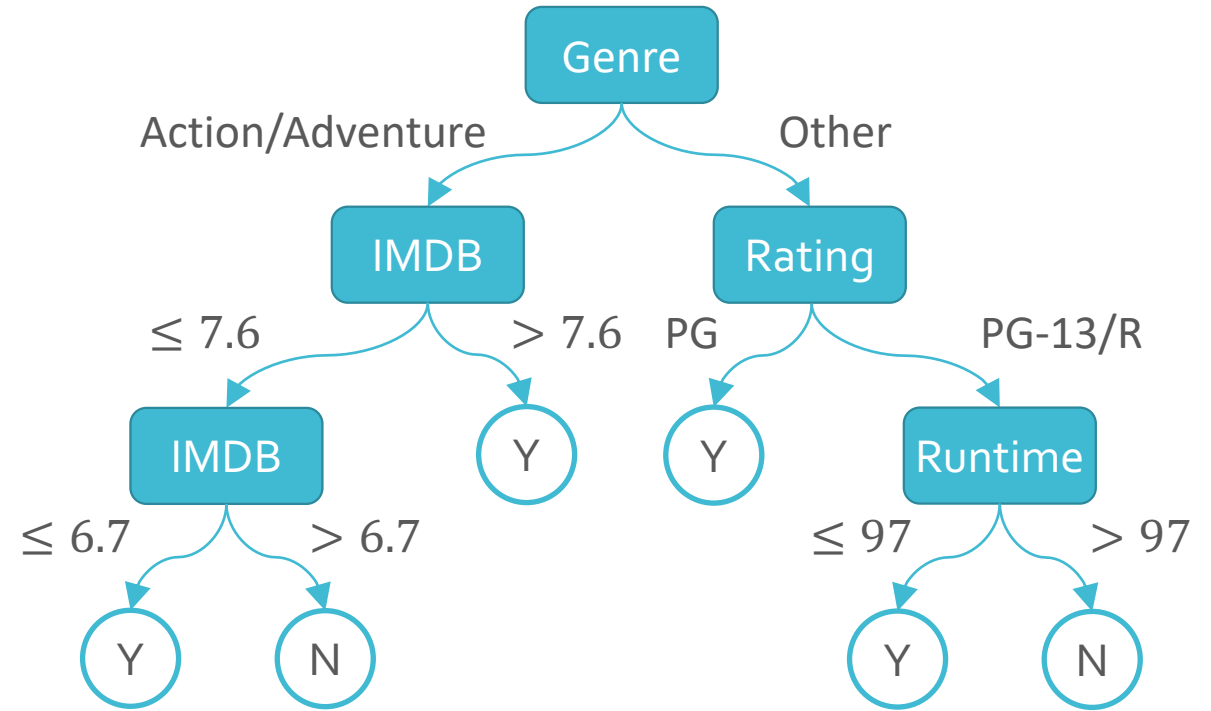
1	 <a href="#">BellKor's Pragmatic Chaos</a>	0.8567	10.06	2009-07-26 18:18:28
2	 <a href="#">The Ensemble</a>	0.8567	10.06	2009-07-26 18:38:22
3	<a href="#">Grand Prize Team</a>	0.8582	9.90	2009-07-10 21:24:40
4	 <a href="#">Opera Solutions and Vandelay United</a>	0.8588	9.84	2009-07-10 01:12:31
5	 <a href="#">Vandelay Industries !</a>	0.8591	9.81	2009-07-10 00:32:20
6	 <a href="#">PragmaticTheory</a>	0.8594	9.77	2009-06-24 12:06:56
7	 <a href="#">BellKor in BigChaos</a>	0.8601	9.70	2009-05-13 08:14:09
8	<a href="#">Dace</a>	0.8612	9.59	2009-07-24 17:18:43
9	<a href="#">Feeds2</a>	0.8622	9.48	2009-07-12 13:11:51
10	 <a href="#">BigChaos</a>	0.8623	9.47	2009-04-07 12:33:59
11	 <a href="#">Opera Solutions</a>	0.8623	9.47	2009-07-24 00:34:07
12	 <a href="#">BellKor</a>	0.8624	9.46	2009-07-26 17:19:11



MovieID	Runtime	Genre	Budget	Year	IMDB	Rating	Liked?
1	124	Action	18M	1980	8.7	PG	Y
2	105	Action	30M	1984	7.8	PG	Y
3	103	Comedy	6M	1986	7.8	PG-13	N
4	98	Adventure	16M	1987	8.1	PG	Y
5	128	Comedy	16.4M	1989	8.1	PG	Y
6	120	Comedy	11M	1992	7.6	R	N
7	120	Drama	14.5M	1996	6.7	PG-13	N
8	136	Action	115M	1999	6.5	PG	Y
9	90	Action	90M	2001	6.6	PG-13	Y
10	161	Adventure	100M	2002	7.4	PG	N
11	201	Action	94M	2003	8.9	PG-13	Y
12	94	Comedy	26M	2004	7.2	PG-13	Y
13	157	Biography	100M	2007	7.8	R	N
14	128	Action	110M	2007	7.1	PG-13	N
15	107	Drama	39M	2009	7.1	PG-13	N
16	158	Drama	61M	2012	7.6	PG-13	N
17	169	Adventure	165M	2014	8.6	PG-13	Y
18	100	Biography	9M	2016	6.7	R	N
19	130	Action	180M	2017	7.9	PG-13	Y
20	141	Action	275M	2019	6.5	PG-13	Y

# Movie Recommendations

MovieID	Runtime	Genre	Budget	Year	IMDB	Rating	Liked?
1	124	Action	18M	1980	8.7	PG	Y
2	105	Action	30M	1984	7.8	PG	Y
3	103	Comedy	6M	1986	7.8	PG-13	N
4	98	Adventure	16M	1987	8.1	PG	Y
5	128	Comedy	16.4M	1989	8.1	PG	Y
6	120	Comedy	11M	1992	7.6	R	N
7	120	Drama	14.5M	1996	6.7	PG-13	N
8	136	Action	115M	1999	6.5	PG	Y
9	90	Action	90M	2001	6.6	PG-13	Y
10	161	Adventure	100M	2002	7.4	PG	N
11	201	Action	94M	2003	8.9	PG-13	Y
12	94	Comedy	26M	2004	7.2	PG-13	Y
13	157	Biography	100M	2007	7.8	R	N
14	128	Action	110M	2007	7.1	PG-13	N
15	107	Drama	39M	2009	7.1	PG-13	N
16	158	Drama	61M	2012	7.6	PG-13	N
17	169	Adventure	165M	2014	8.6	PG-13	Y
18	100	Biography	9M	2016	6.7	R	N
19	130	Action	180M	2017	7.9	PG-13	Y
20	141	Action	275M	2019	6.5	PG-13	Y

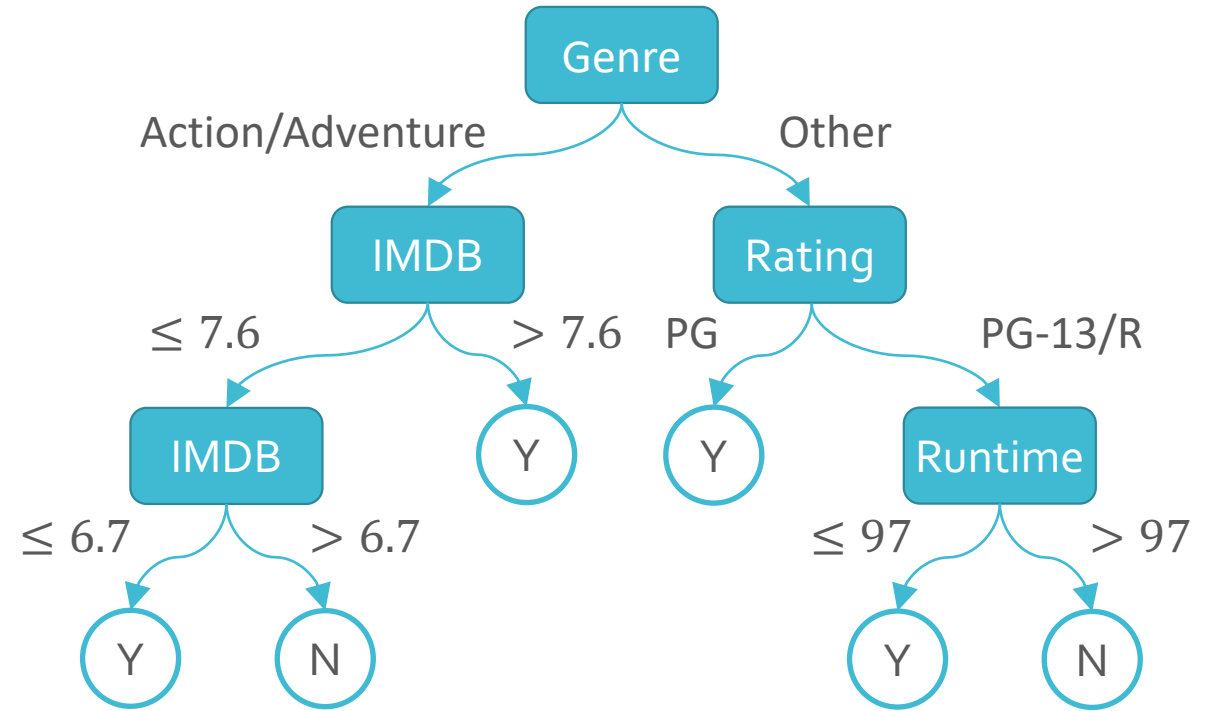


# Decision Trees

# Recall: Decision Tree Pros & Cons

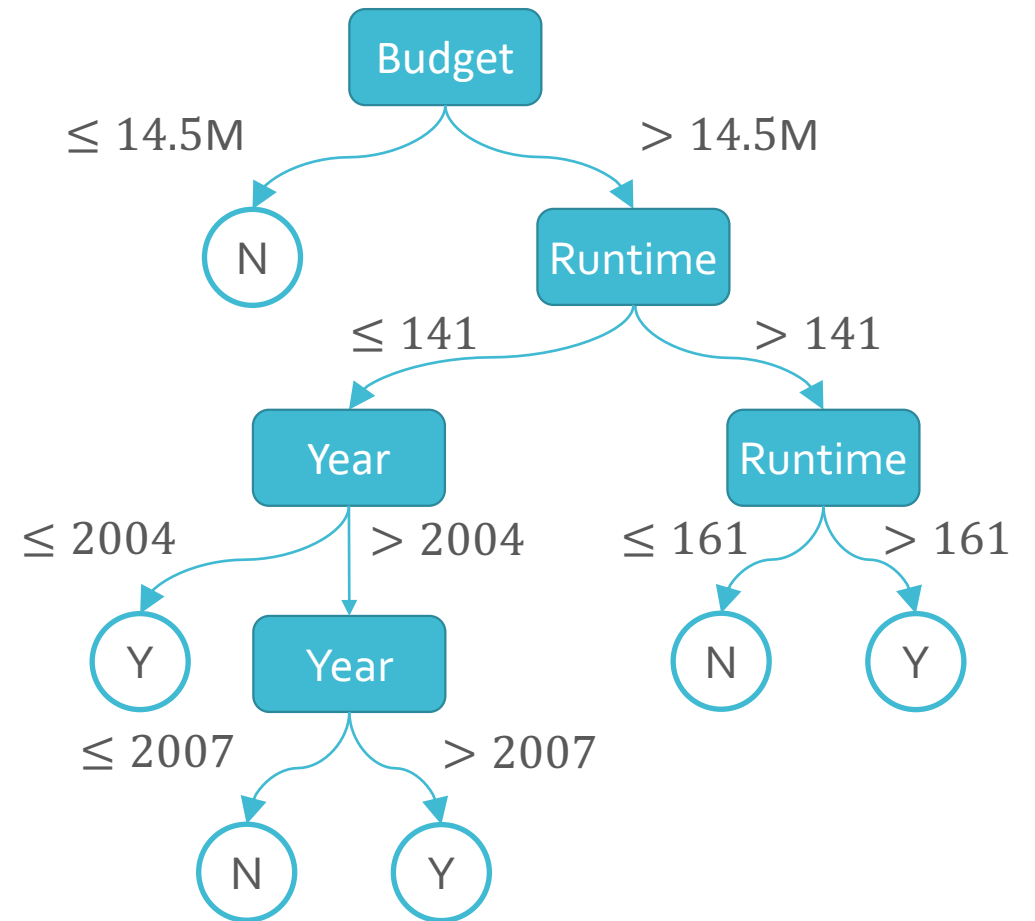
- Pros
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features
- Cons
  - Learned greedily: each split only considers the immediate impact on the splitting criterion
    - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
  - Prone to overfit
  - High variance

MovieID	Runtime	Genre	Budget	Year	IMDB	Rating	Liked?
1	124	Action	18M	1980	8.7	PG	Y
2	105	Action	30M	1984	7.8	PG	Y
3	103	Comedy	6M	1986	7.8	PG-13	N
4	98	Adventure	16M	1987	8.1	PG	Y
5	128	Comedy	16.4M	1989	8.1	PG	Y
6	120	Comedy	11M	1992	7.6	R	N
7	120	Drama	14.5M	1996	6.7	PG-13	N
8	136	Action	115M	1999	6.5	PG	Y
9	90	Action	90M	2001	6.6	PG-13	Y
10	161	Adventure	100M	2002	7.4	PG	N
11	201	Action	94M	2003	8.9	PG-13	Y
12	94	Comedy	26M	2004	7.2	PG-13	Y
13	157	Biography	100M	2007	7.8	R	N
14	128	Action	110M	2007	7.1	PG-13	N
15	107	Drama	39M	2009	7.1	PG-13	N
16	158	Drama	61M	2012	7.6	PG-13	N
17	169	Adventure	165M	2014	8.6	PG-13	Y
18	100	Biography	9M	2016	6.7	R	N
19	130	Action	180M	2017	7.9	PG-13	Y
20	141	Action	275M	2019	6.5	PG-13	Y

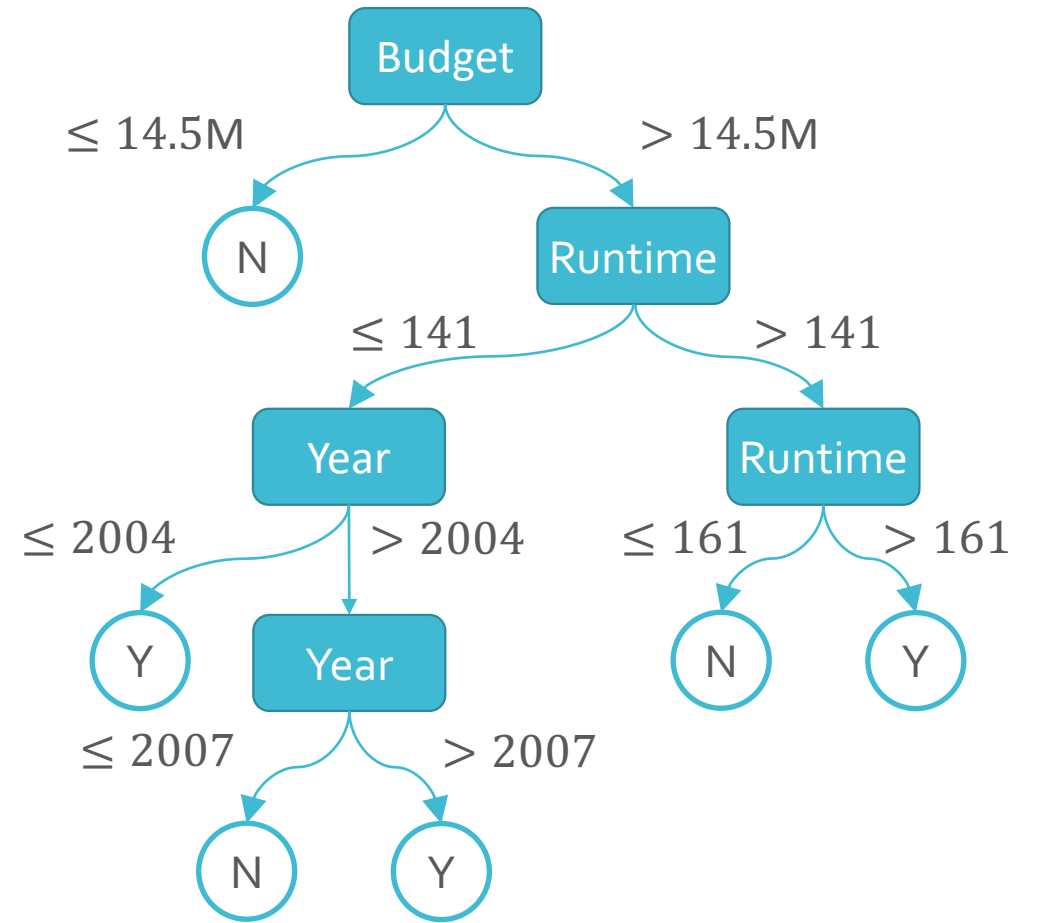
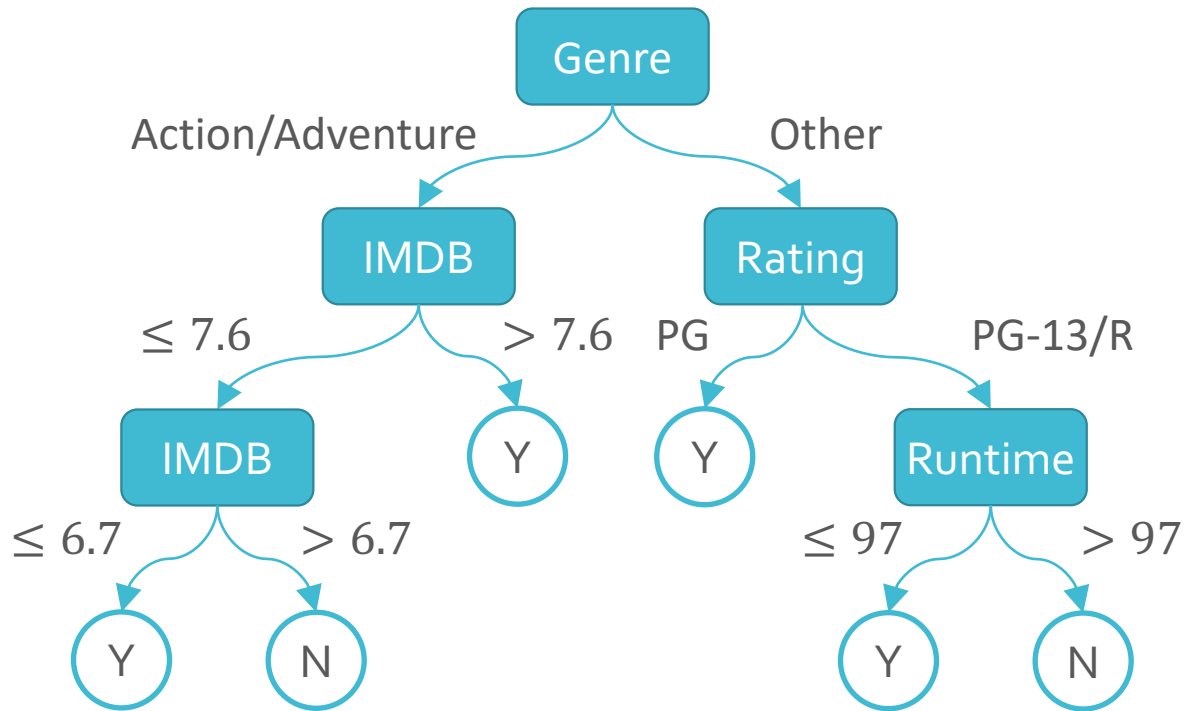


# Decision Trees

MovieID	Runtime	Genre	Budget	Year	IMDB	Rating	Liked?
1	124	Action	18M	1980	8.7	PG	Y
2	105	Action	30M	1984	7.8	PG	Y
3	103	Comedy	6M	1986	7.8	PG-13	N
4	98	Adventure	16M	1987	8.1	PG	Y
5	128	Comedy	16.4M	1989	8.1	PG	Y
6	120	Comedy	11M	1992	7.6	R	N
7	120	Drama	14.5M	1996	6.7	PG-13	N
8	136	Action	115M	1999	6.5	PG	Y
9	90	Action	90M	2001	6.6	PG-13	Y
10	161	Adventure	100M	2002	7.4	PG	N
11	201	Action	94M	2003	8.9	PG-13	Y
12	94	Comedy	26M	2004	7.2	PG-13	Y
13	157	Biography	100M	2007	7.8	R	N
14	128	Action	110M	2007	7.1	PG-13	N
15	107	Drama	39M	2009	7.1	PG-13	N
16	158	Drama	61M	2012	7.6	PG-13	Y
17	169	Adventure	165M	2014	8.6	PG-13	Y
18	100	Biography	9M	2016	6.7	R	N
19	130	Action	180M	2017	7.9	PG-13	Y
20	141	Action	275M	2019	6.5	PG-13	Y



# Decision Trees



# Decision Trees

# Decision Trees: Pros & Cons

- Pros
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features
- Cons
  - Learned greedily: each split only considers the immediate impact on the splitting criterion
    - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
  - Prone to overfit
  - High variance
    - Can be addressed via ensembles → random forests

# Random Forests

- Combines the prediction of many diverse decision trees to reduce their variability
- If  $B$  independent random variables  $x^{(1)}, x^{(2)}, \dots, x^{(B)}$  all have variance  $\sigma^2$ , then the variance of  $\frac{1}{B} \sum_{b=1}^B x^{(b)}$  is  $\frac{\sigma^2}{B}$
- Random forests = sample bagging + feature bagging  
= bootstrap aggregating + split-feature randomization



# Random Forests

- Combines the prediction of many diverse decision trees to reduce their variability
- If  $B$  independent random variables  $x^{(1)}, x^{(2)}, \dots, x^{(B)}$  all have variance  $\sigma^2$ , then the variance of  $\frac{1}{B} \sum_{b=1}^B x^{(b)}$  is  $\frac{\sigma^2}{B}$
- Random forests = sample bagging + feature bagging  
= bootstrap aggregating + split-feature randomization

# Aggregating

- How can we combine multiple decision trees,  $\{t_1, t_2, \dots, t_B\}$ , to arrive at a single prediction?
- Regression - average the predictions:

$$\bar{t}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B t_b(\mathbf{x})$$

- Classification - plurality (or majority) vote; for binary labels encoded as  $\{-1, +1\}$ :

$$\bar{t}(\mathbf{x}) = \text{sign} \left( \frac{1}{B} \sum_{b=1}^B t_b(\mathbf{x}) \right)$$

# Random Forests

- Combines the prediction of many **diverse** decision trees to reduce their variability
- If  $B$  independent random variables  $x^{(1)}, x^{(2)}, \dots, x^{(B)}$  all have variance  $\sigma^2$ , then the variance of  $\frac{1}{B} \sum_{b=1}^B x^{(b)}$  is  $\frac{\sigma^2}{B}$
- Random forests = sample bagging + feature bagging  
= bootstrap aggregating + split-feature randomization

# Bootstrapping

- Insight: one way of generating different decision trees is by changing the training data set
- Issue: often, we only have one fixed set of training data
- Idea: resample the data multiple times *with replacement*

MovieID	...
1	...
2	...
3	...
⋮	⋮
19	...
20	...

Training data

MovieID	...
1	...
1	...
1	...
⋮	⋮
14	...
19	...

Bootstrapped  
Sample 1

MovieID	...
4	...
4	...
5	...
⋮	⋮
16	...
16	...

Bootstrapped  
Sample 2

...

...

# Bootstrapping

- Idea: resample the data multiple times *with replacement*
  - Each bootstrapped sample has the same number of data points as the original data set
  - Duplicated points cause different decision trees to focus on different parts of the input space

MovieID	...
1	...
2	...
3	...
⋮	⋮
19	...
20	...

Training data

MovieID	...
1	...
1	...
1	...
⋮	⋮
14	...
19	...

Bootstrapped  
Sample 1

MovieID	...
4	...
4	...
5	...
⋮	⋮
16	...
16	...

Bootstrapped  
Sample 2

...

...

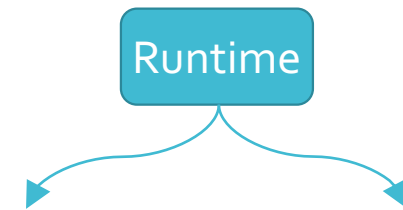
# Split-feature Randomization

- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset

Runtime	Genre	Budget	Year	IMDB	Rating
---------	-------	--------	------	------	--------

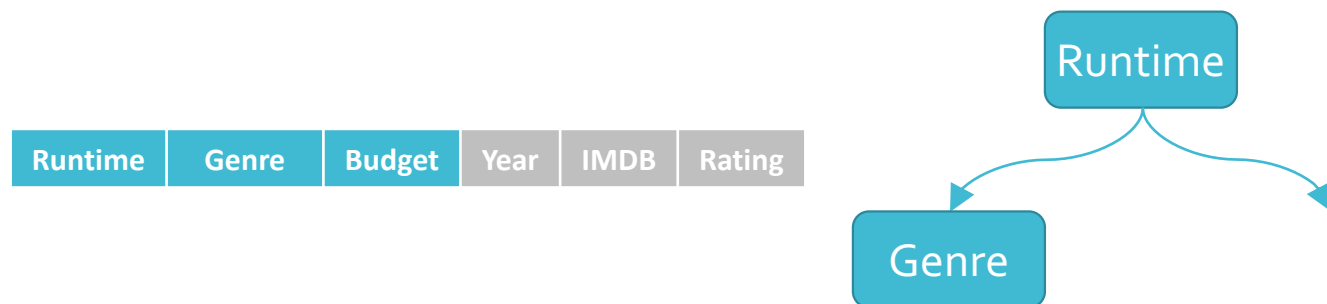
# Split-feature Randomization

- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset



# Split-feature Randomization

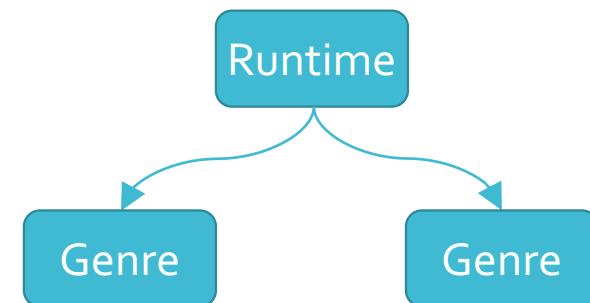
- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset





# Split-feature Randomization

- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset



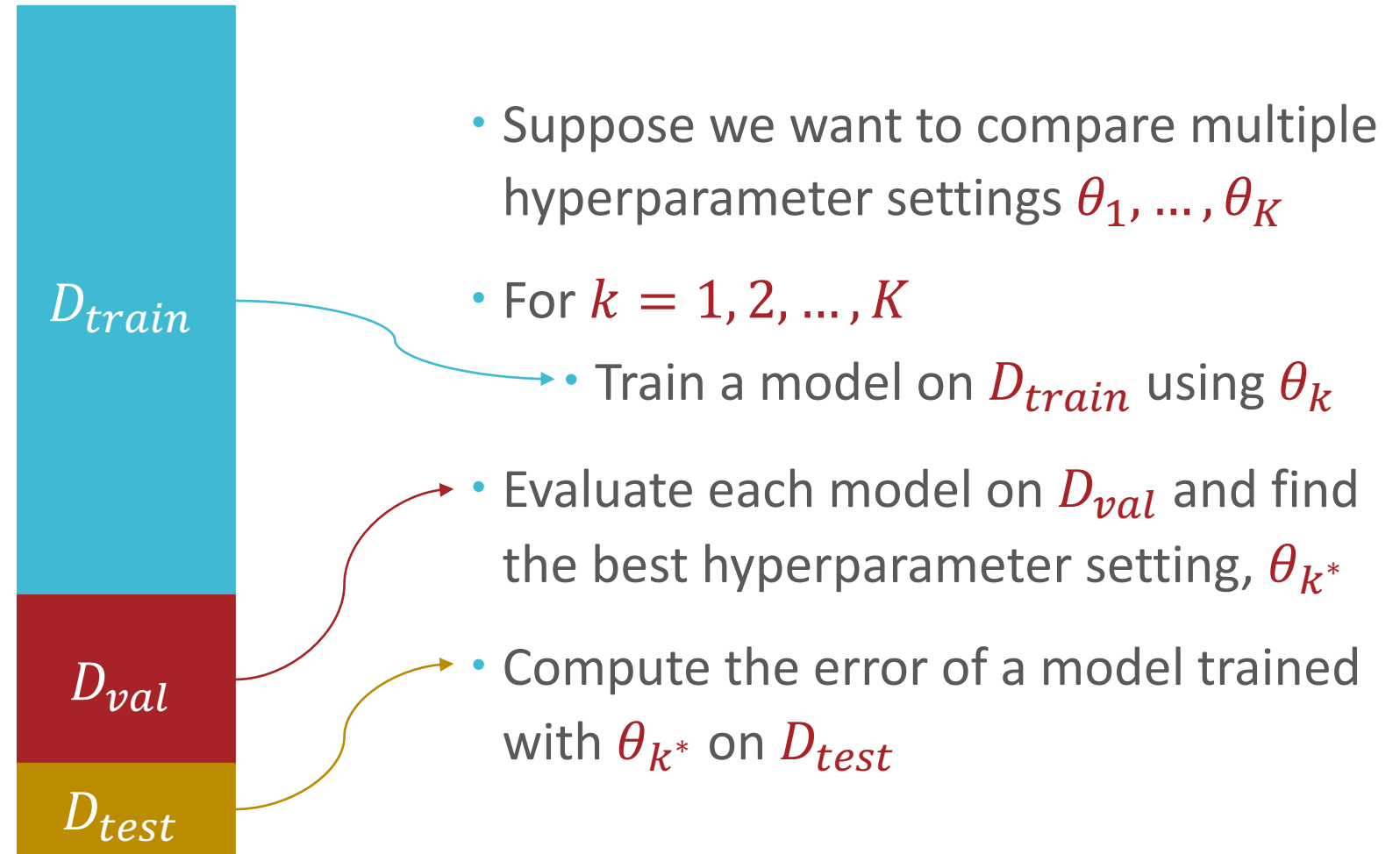
# Random Forests

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N, B, \rho$
- For  $b = 1, 2, \dots, B$ 
  - Create a dataset,  $\mathcal{D}_b$ , by sampling  $N$  points from the original training data  $\mathcal{D}$  **with replacement**
  - Learn a decision tree,  $t_b$ , using  $\mathcal{D}_b$  and the ID3 algorithm **with split-feature randomization**, sampling  $\rho$  features for each split
- Output:  $\bar{t} = f(t_1, \dots, t_B)$ , the aggregated hypothesis

## How can we set $B$ and $\rho$ ?

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N, B, \rho$
- For  $b = 1, 2, \dots, B$ 
  - Create a dataset,  $\mathcal{D}_b$ , by sampling  $N$  points from the original training data  $\mathcal{D}$  *with replacement*
  - Learn a decision tree,  $t_b$ , using  $\mathcal{D}_b$  and the ID3 algorithm *with split-feature randomization*, sampling  $\rho$  features for each split
- Output:  $\bar{t} = f(t_1, \dots, t_B)$ , the aggregated hypothesis

# Recall: Validation Sets



# Out-of-bag Error

- For each training point,  $\mathbf{x}^{(n)}$ , there are some decision trees which  $\mathbf{x}^{(n)}$  was not used to train (roughly  $B/e$  trees or 37%)
  - Let these be  $t^{(-n)} = \{t_1^{(-n)}, t_2^{(-n)}, \dots, t_{N-n}^{(-n)}\}$
- Compute an aggregated prediction for each  $\mathbf{x}^{(n)}$  using the trees in  $t^{(-n)}$ ,  $\bar{t}^{(-n)}(\mathbf{x}^{(n)})$
- Compute the out-of-bag (OOB) error, e.g., for regression

$$E_{OOB} = \frac{1}{N} \sum_{n=1}^N (\bar{t}^{(-n)}(\mathbf{x}^{(n)}) - y^{(n)})^2$$

# Out-of-bag Error

- For each training point,  $\mathbf{x}^{(n)}$ , there are some decision trees which  $\mathbf{x}^{(n)}$  was not used to train (roughly  $B/e$  trees or 37%)
  - Let these be  $t^{(-n)} = \{t_1^{(-n)}, t_2^{(-n)}, \dots, t_{N-n}^{(-n)}\}$
- Compute an aggregated prediction for each  $\mathbf{x}^{(n)}$  using the trees in  $t^{(-n)}$ ,  $\bar{t}^{(-n)}(\mathbf{x}^{(n)})$

- Compute the out-of-bag (OOB) error, e.g., for classification

$$E_{OOB} = \frac{1}{N} \sum_{n=1}^N \mathbb{1}(\bar{t}^{(-n)}(\mathbf{x}^{(n)}) \neq y^{(n)})$$

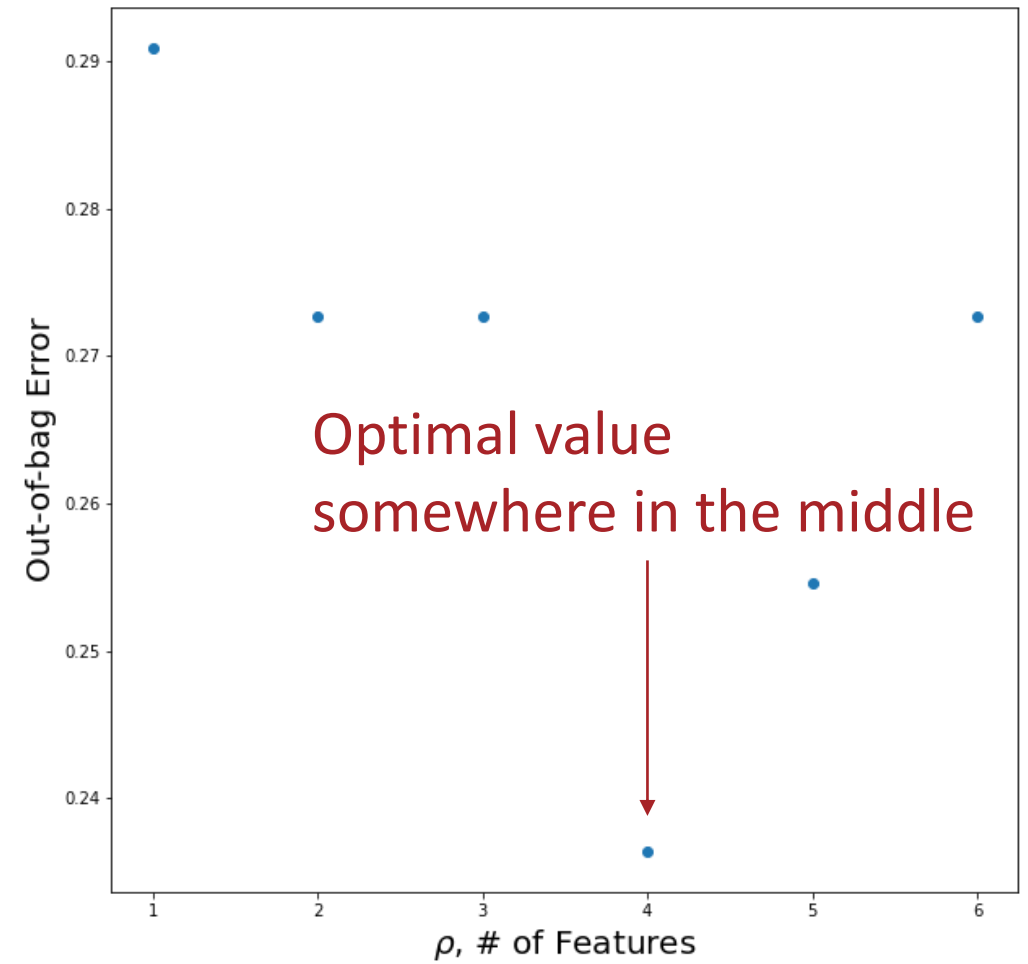
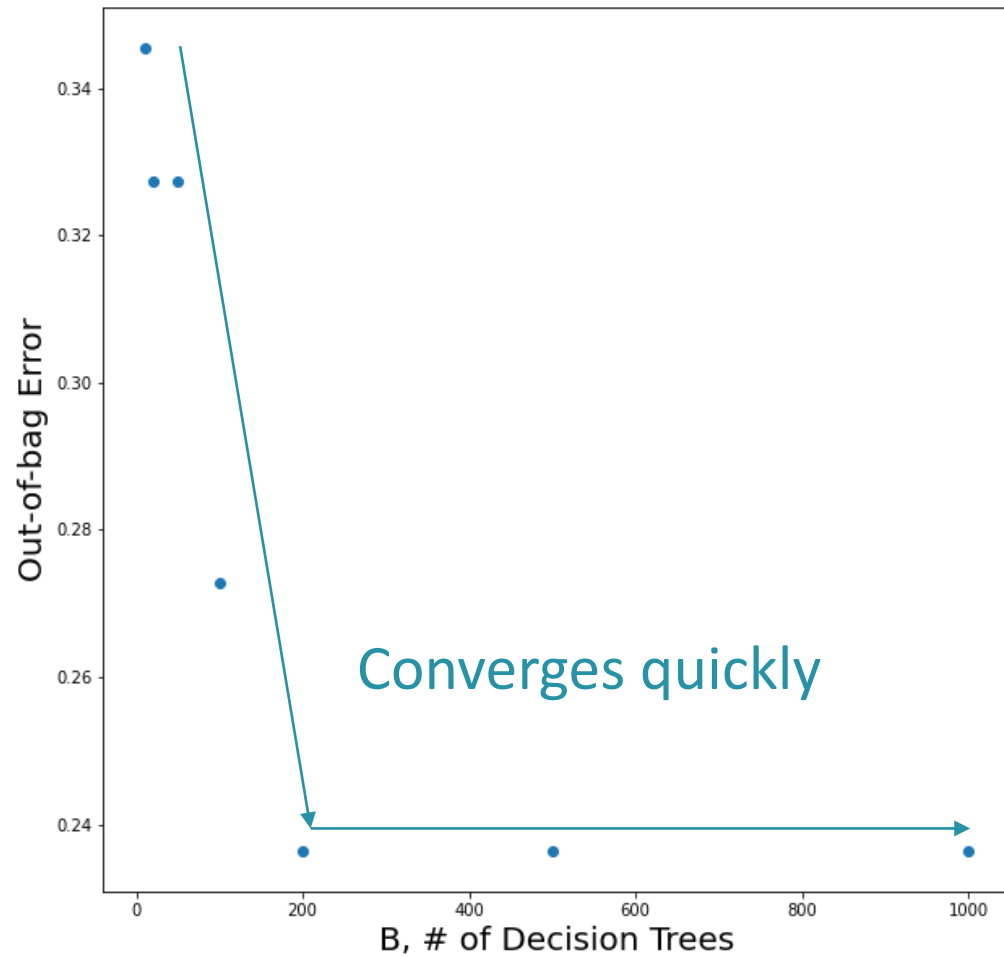
- $E_{OOB}$  can be used for hyperparameter optimization!

# Out-of-bag Error

$D_{train}$

$D_{test}$

- Suppose we want to compare different numbers of trees in our random forest  $B_1, \dots, B_K$
- For  $k = 1, 2, \dots, K$ 
  - Train a random forest on  $D_{train}$  with  $B_k$  trees
  - Compute  $E_{OOB}$  for each random forest and find the best number of trees,  $B_{k^*}$
- Evaluate the random forest with  $B_{k^*}$  trees on  $D_{test}$



# Setting Hyperparameters



# Feature Importance

- Some of the interpretability of decision trees gets lost when switching to random forests
- Random forests allow for the computation of “feature importance”, a way of ranking features based on how useful they are at predicting the target
- Initialize each feature’s importance to zero
- Each time a feature is chosen to be split on, add the reduction in entropy (weighted by the number of data points in the split) to its importance

# Feature Importance

