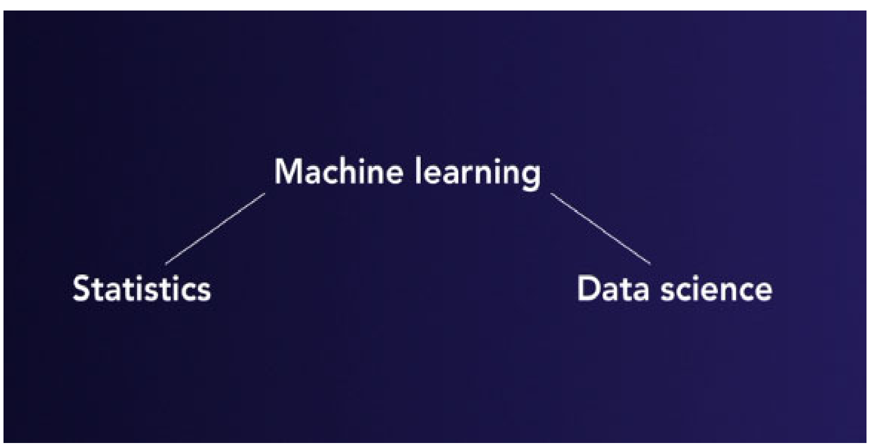
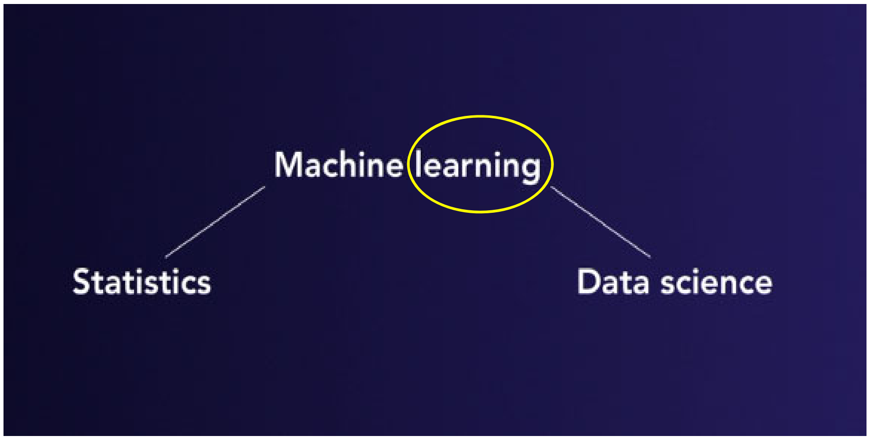
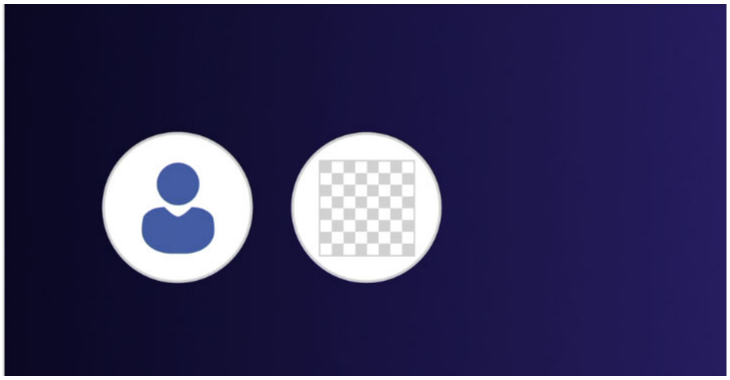
Machine Learning Overview



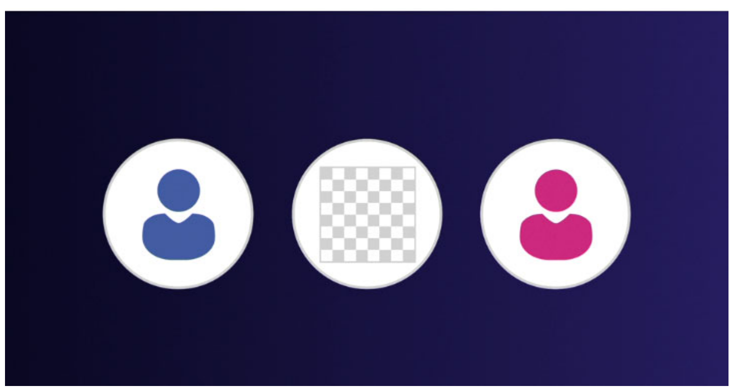


How do humans learn?

How can you learn to play chess?

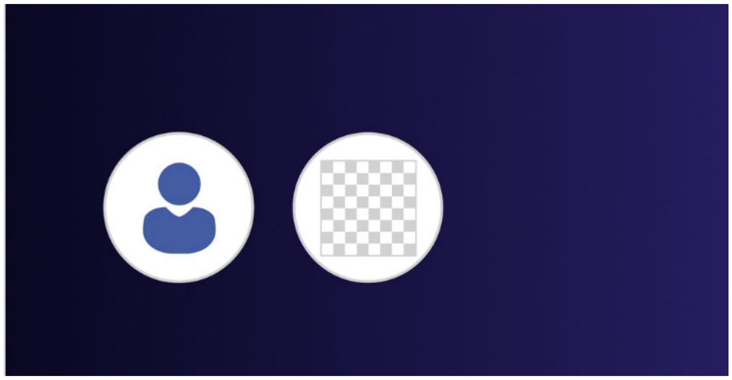


How can you learn to play chess?

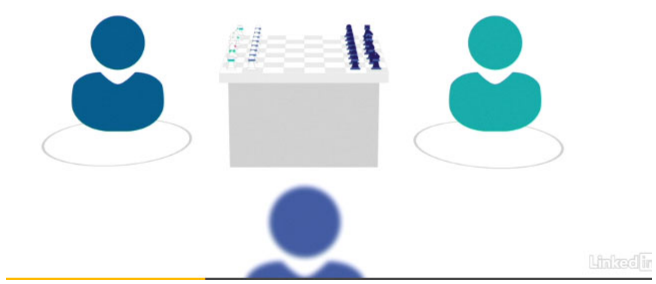


Hire a chess tutor – who will supervise your learning

Suppose you don’t have a tutor or chess book?



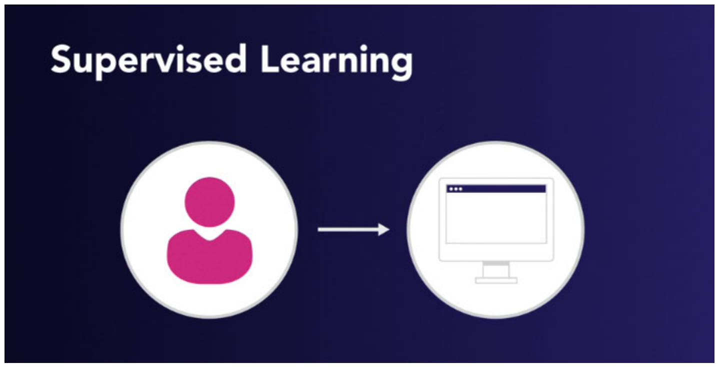
Go to the park and observe games



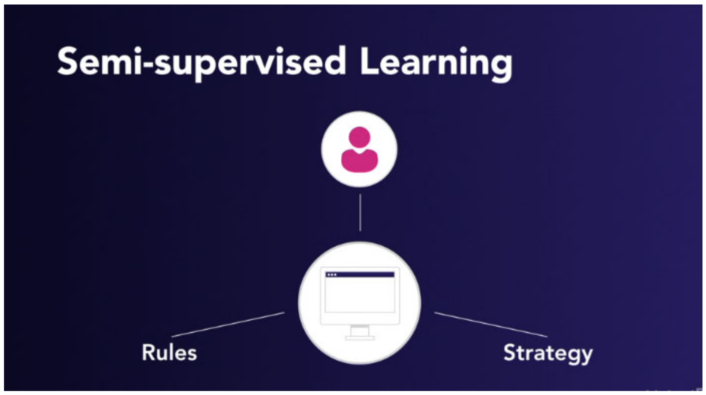
Maybe try a combination

- Use a tutor for game basics
- Observe games and refine your skills

How does human learning relate to Machine Learning?


Supervised Learning


Unsupervised Learning
Rules
Strategy



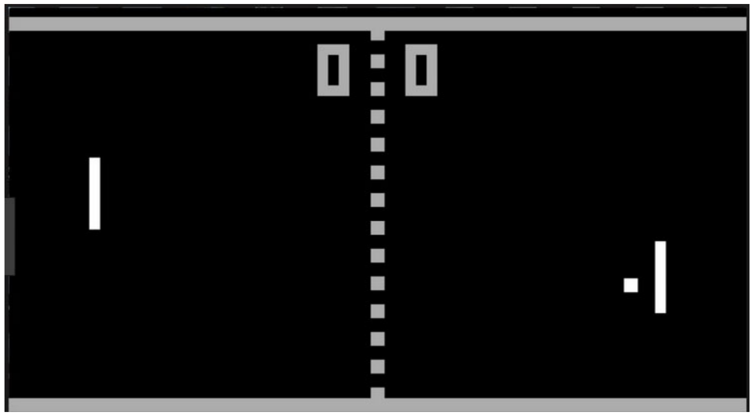
Which approach is best?

....It depends

- Do you have a good tutor?
  - supervised works
- Do you have lots of observations but no tutor?
  - unsupervised
- Do you have both?
  - semi-supervised
- Downsides?
  - bad tutor
  - poor observations

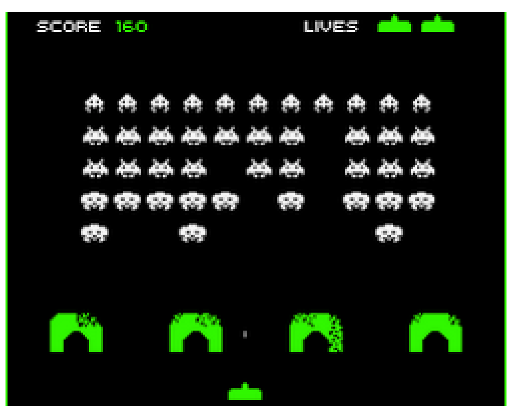
Reinforcement Learning

PONG



Give rewards – when paddle hits ball = +
- when opponent misses = +

Space Invaders



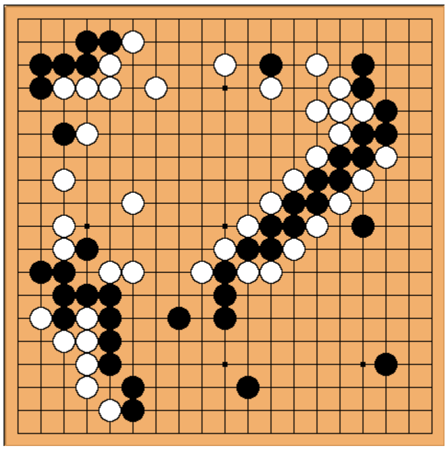
More complex game...

Q-Learning

- States
- Actions
- Q = Quality of outcome

- What actions lead to better states – get a higher Q value
- Unsupervised learning based on playing numerous games

GO



Alpha Go

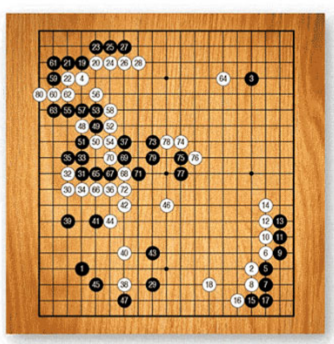
- AlphaGo's first formal match was against the reigning 3-times European Champion, Mr Fan Hui, in October 2015.
- Unsupervised Learning – studied thousands of Go Games

Alpha Go Zero

- 5 months later
- Alpha Go Zero outplays Alpha GO
- Uses Q-Learning

3 hours

AlphaGo Zero plays like a human beginner, forgoing long term strategy to focus on greedily capturing as many stones as possible.



19 hours

AlphaGo Zero has learnt the fundamentals of more advanced Go strategies such as life-and-death, influence and territory.

Reinforcement Learning

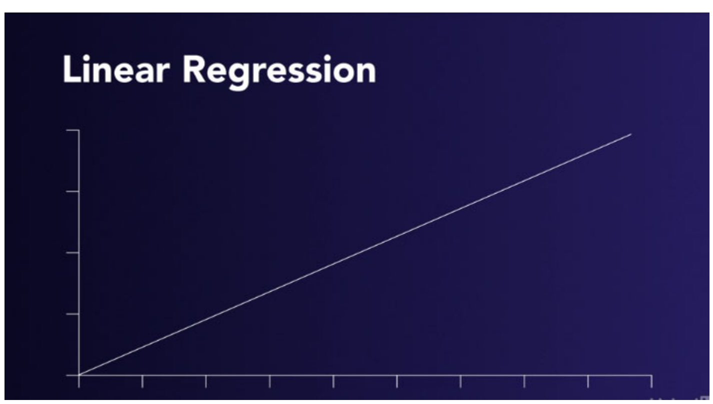
- See: https://skymind.ai/wiki/deep-reinforcement-learning

Categories of Supervised Learning

- Binary
- Multiclass
- Regression

Binary Classification


Multiclass Classification


Linear Regression

Approaches to Machine Learning

- Decision Trees
- k-nearest neighbor
- K-mean clustering
- Regression
  - we have looked at single variable and multiple variable regression

Decision Trees

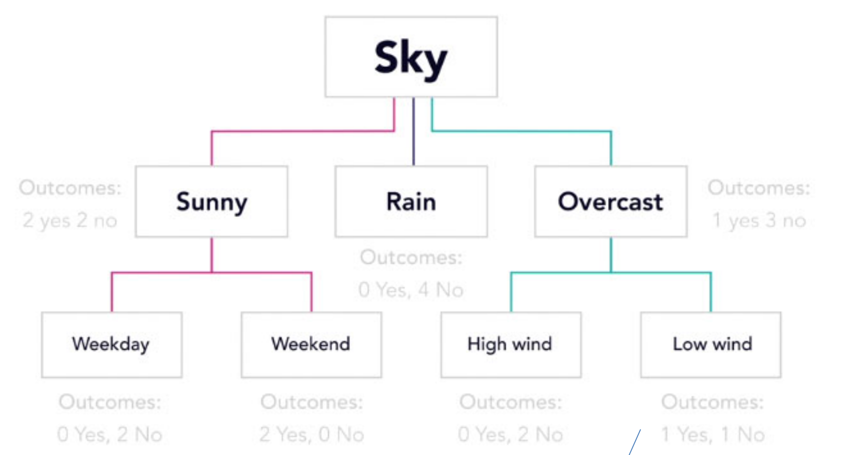
- Use for binary classification with supervised machine learning. Should Yash go to the beach?

Predictors	Outcome
• Sky • Weekend • Wind speed	• Yash goes to the beach

Supervised Training Set

Predictors			Outcome
Sky	Weekend	Wind	Yash goes to the beach
Sunny	Weekday	Low	No
Sunny	Weekday	High	No
Overcast	Weekday	Low	Yes
Rain	Weekday	Low	Yes
Rain	Weekend	Low	Yes
Rain	Weekend	High	No
Overcast	Weekend	High	Yes
Sunny	Weekday	Low	No
Sunny	Weekend	Low	Yes
Rain	Weekend	Low	Yes
Sunny	Weekend	High	Yes
Overcast	Weekday	High	Yes
Overcast	Weekend	Low	Yes
Rain	Weekday	High	No

Use data to build decision tree



WIND is not a good predictor – may want to remove or find another predictor..

k-nearest neighbor

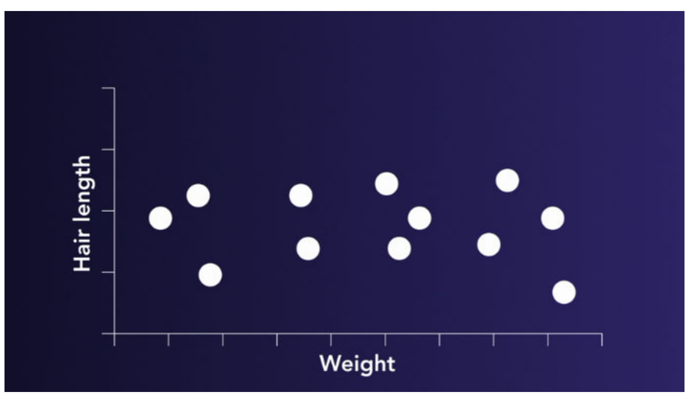
k-nearest neighbor

- KNN – one of simplest and most commonly used classifiers
- Does not actually train a model to predict
- An observation is predicted based on the class with the largest proportion of nearest neighbors
- If observation is surrounded by class Z, then we conclude our observation is class Z

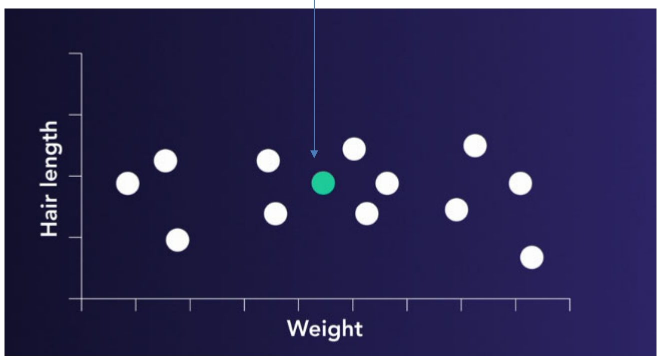
make predictions by classifying your data with what you already know

a form of supervised learning

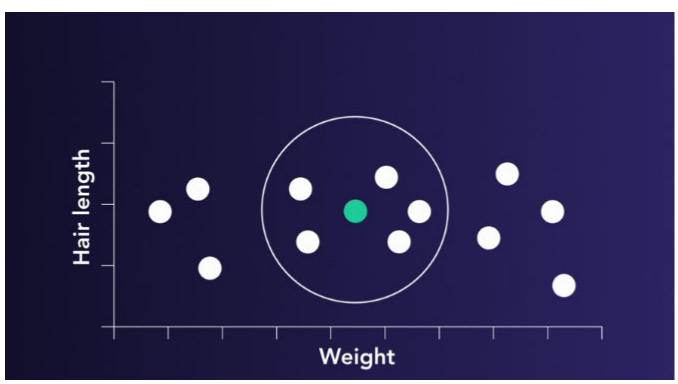
We have dog data – we know breeds and measure hair length and weight

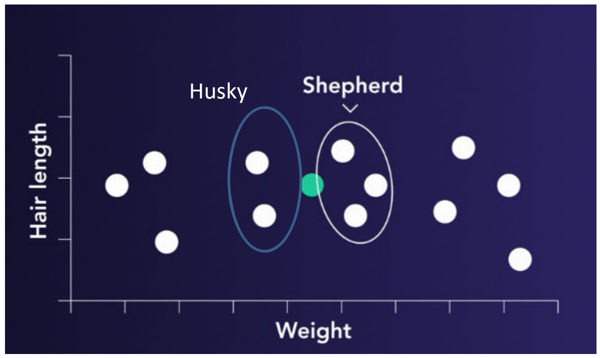


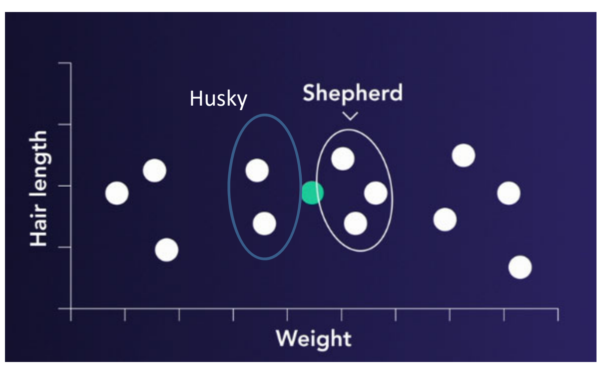
unknown dog breed



k = 5 nearest neighbors







Conclude Shepherd

https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

Code – k-nearest neighbor

Iris - dataset

The data set consists of 50 samples from each of three species of *Iris* (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other.

Ronald Fisher

K-Nearest Neighbor

```python
# Load libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import datasets
```

```python
# load data
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

```python
# create standardizer
standardizer = StandardScaler()

# standarize features
X_std = standardizer.fit_transform(X)
```

```python
# train KNN classifier with 5 neighbors
knn = KNeighborsClassifier(n_neighbors=5, n_jobs=-1).fit(X_std,y)
```

```python
# create two observations
new_observations = [[0.75, 0.75, 0.75, 0.75],
                    [1, 1, 1, 1]]

# predict the class
knn.predict(new_observations)

```

```
array([1, 2])
```

```python
# view probability of our observations is in one of three classes
knn.predict_proba(new_observations)
```

```
array([[0. , 0.6, 0.4],
       [0. , 0. , 1. ]])
```

k-Means Clustering

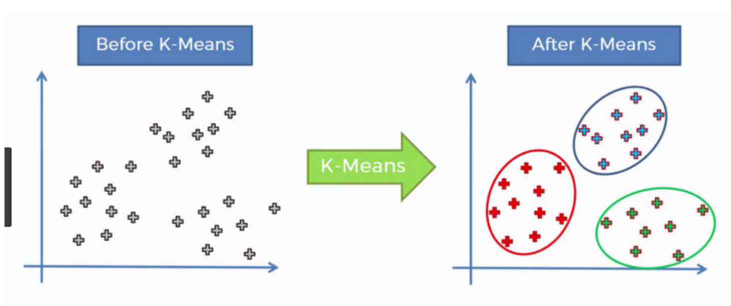
k-Means Clustering

k-Means Clustering ≠ k-NN

k-means clustering is an unsupervised machine learning algorithm – developed in 1967

Use it to create clusters based on what the machine sees in the data

Assumes there are groups that are not labeled that have different characteristics

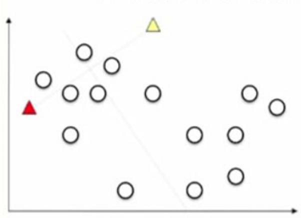


K-means clustering algorithm

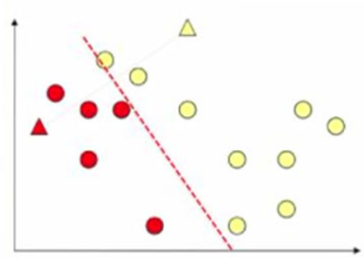
- Input: K, set of points $x_1 \dots x_n$
- Place centroids $c_1 \dots c_K$ at random locations
- Repeat until convergence:
  - for each point x_i:
    - find nearest centroid c_j $\quad \underbrace{\arg\min_j D(x_i, c_j)}_{\text{distance (e.g. Euclidian) between instance } x_i \text{ and cluster center } c_j}$
    - assign the point x_i to cluster j
  - for each cluster j = 1 ... K: $\quad c_j(a) = \frac{1}{n_j} \sum_{x_i \to c_j} x_i(a) \quad$ for a = 1..d
    - new centroid c_j = mean of all points x_i assigned to cluster j in previous step

STOP when none of the cluster assignments change

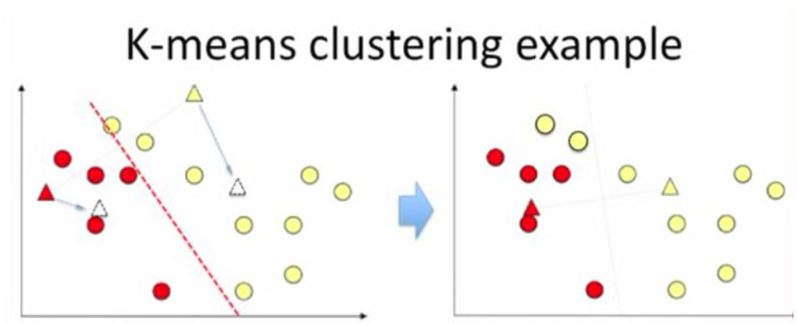
K-means clustering example



Pick k centroids at random

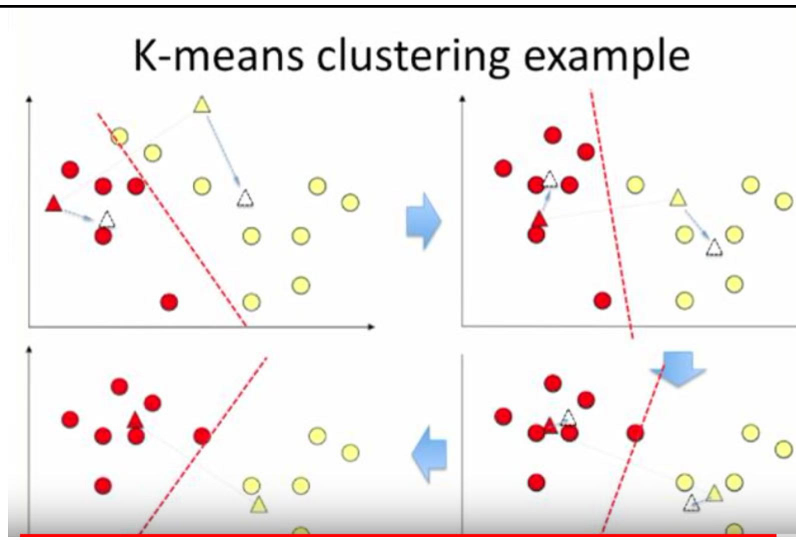


Categorize data based on how close to the centroid



Move centroid to the mean of the class

Rebalance



Repeat until the position of the centroid does not change

k-means assumptions

- Assumes clusters are convex shapes (circles, ellipses)
- All features equally scaled
- Groups are balanced – have roughly the same number of observations

See: https://www.datascience.com/blog/k-means-clustering

Code

```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

```
#load data
iris = datasets.load_iris()
features = iris.data
```

```
# standardize features
scaler = StandardScaler()
features_std = scaler.fit_transform(features)

```

```
# create K-means object
cluster = KMeans(n_clusters=3, random_state=0, n_jobs= -1)

```

```
# Train model
model = cluster.fit(features_std)
```

```
# view predict class
model.labels_
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 2, 0,
       2, 2, 2, 2, 0, 2, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0,
       0, 0, 0, 2, 2, 0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0,
       0, 2, 2, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2])
```

```
# view true class
iris.target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Very close to actual data – note that different category numbers

```
# create new observation
new_observation = [[0.8, 0.8, 0.8, 0.8 ]]

# predict
model.predict (new_observation)
```

```
array([0])
```

closest match is category 0

```
# view cluster centers
model.cluster_centers_
```

```
array([[ 1.13597027,  0.09659843,  0.996271  ,  1.01717187],
       [-1.01457897,  0.84230679, -1.30487835, -1.25512862],
       [-0.05021989, -0.88029181,  0.34753171,  0.28206327]])
```

Readings

- Reinforcement Learning:
  - See: https://skymind.ai/wiki/deep-reinforcement-learning
- K-Means Clustering
  - See: https://www.datascience.com/blog/k-means-clustering
- KNN – K-Nearest Neighbor
  - https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7
- Decision Trees
  - https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052