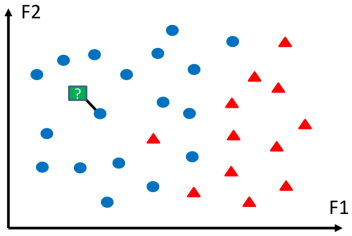
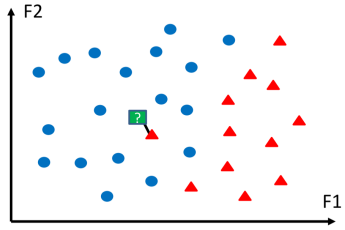
K-Nearest Neighbor Classifier (KNN Classifier)

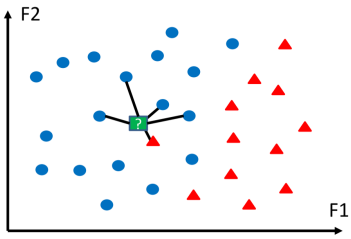
- Classification algo that classifies objects based on closest training samples in the feature space



← conclude ? is blue since nearest neighbor is blue



- Making decision based on single sample is risky
  - not reliable, vulnerable to noise or mistakes in training set



- KNN classifier classifies objects based on majority of K closest training samples in feature space e.g. K=5

Feature Table

- **Training Feature Table:**
  - In this example we have *6 training data samples (6 different flowers)*, *4 features* and label has *3 classes (3 different types of Iris)*.

D

	sepal length	sepal width	petal length	petal width	Label	
1	5.3	3.7	1.5	0.2	setosa	y_1
2	5	3	2	0.2	setosa	y_2
3	7.0	3.2	4.7	1.4	versicolor	⋮
4	6.4	3.2	4.5	1.5	versicolor	
5	6.3	2.7	4.9	1.8	virginica	
6	7.9	3.8	6.4	2	virginica	

N

- training dataset: $\{(x_1, y_1), \dots (x_n, y_n)\}$: n data samples used for training

	sepal length	sepal width	petal length	petal width	Label	
x_1	5.3	3.7	1.5	0.2	setosa	y_1
x_2	5	3	2	0.2	setosa	y_2
x_3	7.0	3.2	4.7	1.4	versicolor	y_3
	6.4	3.2	4.5	1.5	versicolor	
	6.3	2.7	4.9	1.8	virginica	
	7.9	3.8	6.4	2	virginica	

Defn

- Assuming we have
  - N data samples (observations)
  - D features
  - C classes

- Feature vector x (input): The vector of all features for one single observation: $x \in \mathbb{R}^p$
- Label y (output): Value of label for one observation $y \in \{0, 1, 2, \dots C\}$
- training dataset: $\{(x_1, y_1), (x_n, y_n) \dots (x_n, y_n)\}$

* special case: Binary classification
  - Number of classes $C=2$
  - Labels: $(0, 1)$

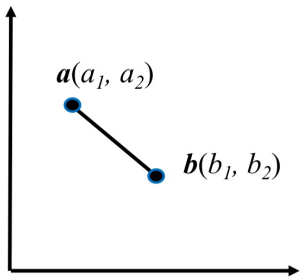
1- Nearest Neighbor Classifier

- Training dataset: $\{(x_1, y_1) \dots (x_n, y_n)\}$ with known label
- new sample with unknown label: $(x, y = ?)$

	sepal length	sepal width	petal length	petal width	Label	
x_1	5.3	3.7	1.5	0.2	setosa	y_1
x_2	5	3	2	0.2	setosa	y_2
	7.0	3.2	4.7	1.4	versicolor	
:	6.4	3.2	4.5	1.5	versicolor	:
	6.3	2.7	4.9	1.8	virginica	
x_N	7.9	3.8	6.4	2	virginica	y_N
x	7	3.9	5.9	1.3	???	$y=?$

Euclidean Distance: Between 2 points in 2-D Space

$$Dis(a, b) = \|a - b\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$



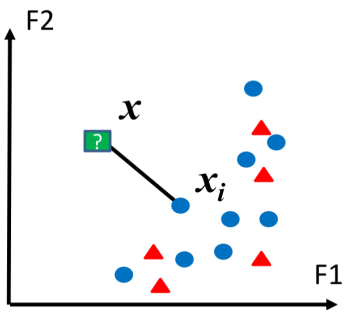
Euclidean Distance: for D- dimensional Space:

$$Dis(x_i, x) = \|x_i - x\| = \sqrt{\sum_{d=1}^{D} (x_{id} - x_d)^2}$$

1- Nearest Neighbor Classification

- training dataset: $\{(x_1, y_1) \dots (x_n, y_n)\}$
- new sample, unknown label: $(x, y = ?)$

- Euclidean Distance between x and a known training point x_i:



- Nearest Neighbor to x ($NN(x)$)

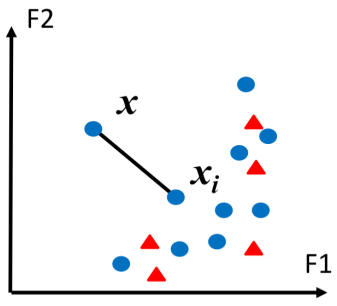
$$NN(x) = \arg\min_{i \in \{1,\dots,n\}} (Dis(x_i, x))$$

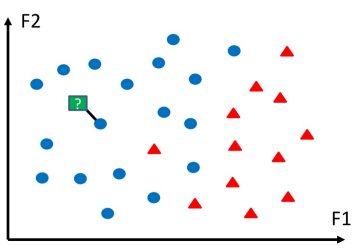
$$= \arg\min_{i \in \{1,\dots,n\}} (\|x_i - x\|^2)$$

$$= \arg\min_{i \in \{1,\dots,n\}} \left(\sum_{d=1}^{D} (x_{id} - x_d)^2\right)$$

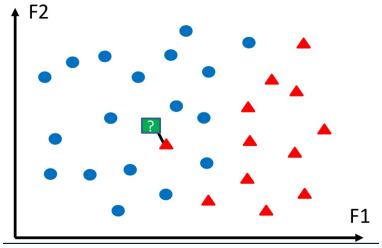
- After finding NN of x, label of x will be determined as label of its NN

$$y = y_{NN}(x)$$





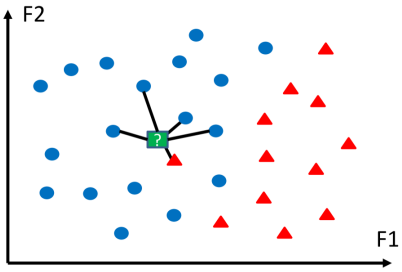
1-NN works on this example, but will it always work?



← In this ex. NN classification would be wrong

- making decision on a single sample is risky

K-NN Classification



- Classifies objects based on K closest training samples in the feature space

e.g. K=5

- 1st NN: $NN_1(x)$
- 2nd NN: $NN_2(x)$
- ⋮
- k^{th} NN: $NN_K(x)$

- set of K-NNs: $KNN(x) = \{NN_1(x) \dots NN_K(x)\}$

↑ Out of 5 NN, 4 blue, 1 red, ∴ prediction is blue

- Classification Rule: Select the label with the majority in $KNN(x)$

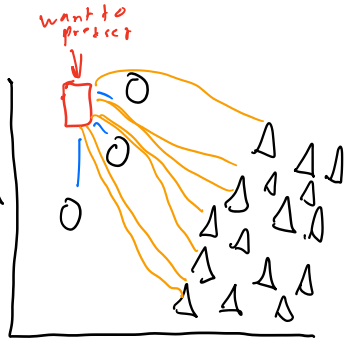
Benefits large K

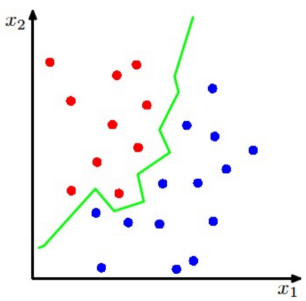
- ignoring the effect of outliers
- Better decision making

Benefits of a small K

- avoid overfitting
- Simpler model
- low computational complexity
- No bias towards popular labels

if K=3, predict O
if K=10 predict Δ





Decision Boundary

- ∀ possible point in the space, can determine its label using KNN rule
- this gives a decision boundary that partitions the space into different regions

Advantages KNN Classifier

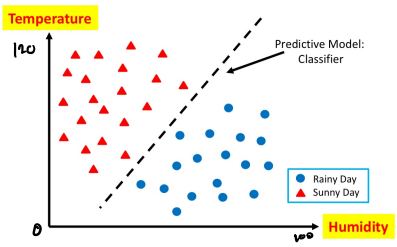
- easy & simple to implement
- low computational complexity

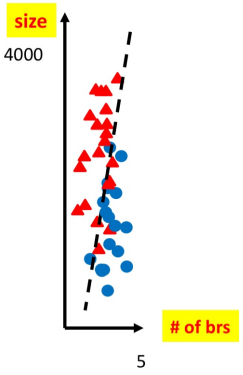
Disadvantages

- Computationally intensive for large scale problems:
  - inefficient for Big Data (need to find millions of distances, find shortest each time)
- Choosing best K is challenging

- Practical hint about KNN
  - distances depend on unit of feature

← in ex. both features almost in same range so no problem





Ex Predicting if a house will be sold within 30 days based on size & # of bedrooms

feature 1: size (0-4000 ft²)
feature 2: # of bedrooms (0-5)

$NN(x) = \arg\min_{i \in \{1,\dots,N\}} (\|x_i - x\|_2^2)$

$= \arg\min_{i \in \{1,\dots,N\}} [(x_{i1} - x_1)^2 + (x_{i2} - x_2)^2]$

negligable — significant

∴ we need to scale (normalize) the features before training / testing

Easy way to scale features:

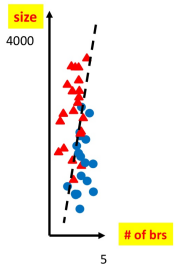
F1_Scaled = Size / max(size)

F2_Scaled = # of bedrooms / max # of rooms

After normalization:

$(0 \le F1\_Scaled \le 1)$

$(0 \le F2\_Scaled \le 1)$



Before Scaling

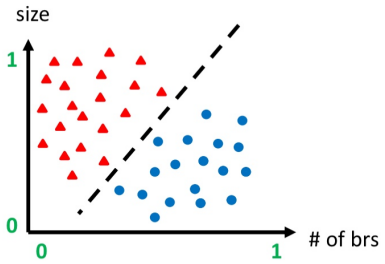
After Scaling

- Before Scaling:

	Number of Bedrooms	Size (sqr footage)	Label
1	1	1300	Sold
2	3	2400	Not-Sold
3	3	2270	Not-Sold
4	2	1450	Sold
5	2	1400	Sold
6	4	2900	Not-Sold

- After Scaling:

	Number of Bedrooms	Size (sqr footage)	Label
1	1/4	1300/2900	Sold
2	3/4	2400/2900	Not-Sold
3	3/4	2270/2900	Not-Sold
4	2/4	1450/2900	Sold
5	2/4	1400/2900	Sold
6	4/4	2900/2900	Not-Sold



More advanced technique for normalization:

- We can Normalize data samples to have zero mean and unit standard deviation in each dimension (i.e. for each feature)
  - We should normalize for each column of the feature table individually

- Compute mean & std deviation for each feature (x_{nd} is the d^{th} feature of n^{th} data sample):

$$mean(x_d) = \bar{x}_d = \frac{1}{N}\sum_{n=1}^{N} x_{nd} \qquad std(x_d) = s_d = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_{nd} - \bar{x}_d)^2}$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \left(\frac{x_{nd} - \bar{x}_d}{s_d}\right)$$