Similarity and Dissimilarity Measures

- Used by a number of data mining techniques:
  - Nearest neighbors
  - Clustering
  - Anomaly detection

How to measure "proximity"?

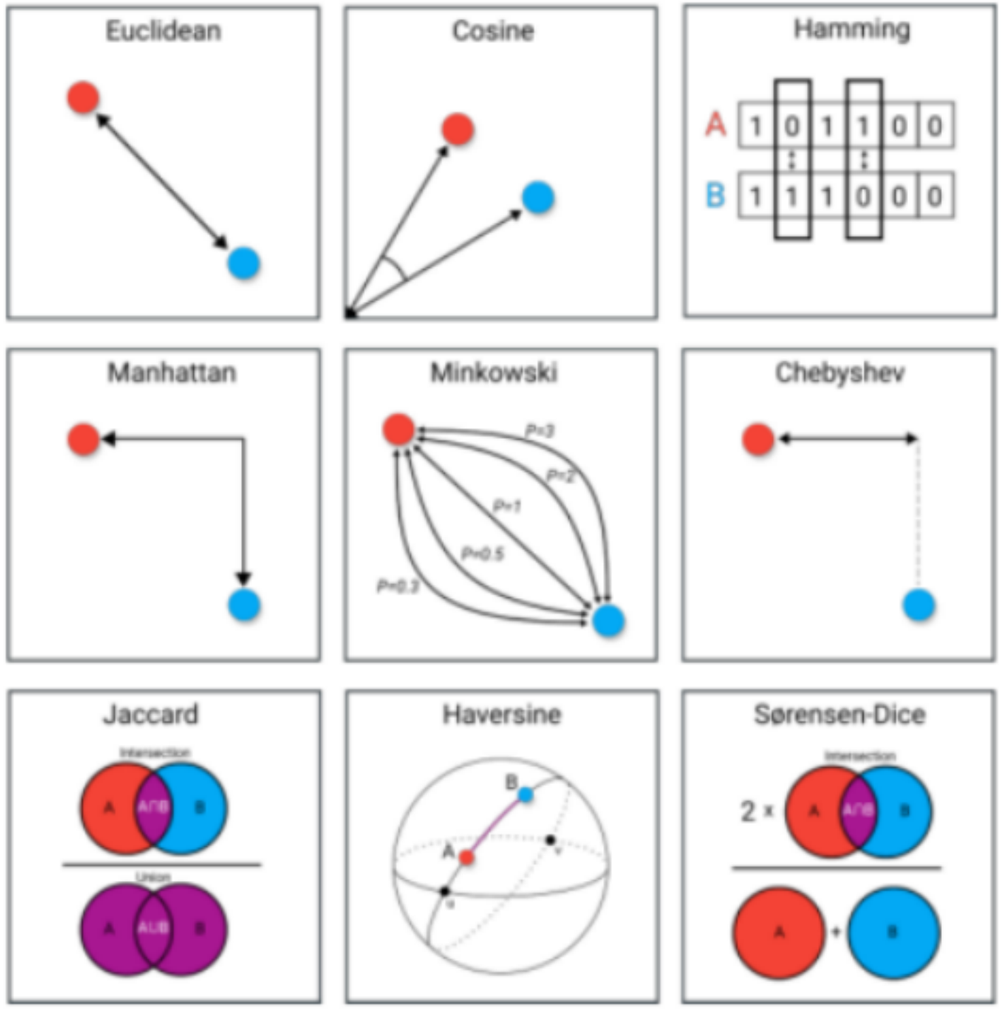
- Proximity: similarity or dissimilarity between two objects
  - Similarity: numerical measure of the degree to which two objects are *alike*
    - Usually in range [0,1]
      - 0 = no similarity
      - 1 = complete similarity
  - Dissimilarity: measure of the degree in which two objects are *different*

Similarity/Dissimilarity for Simple Attributes

- p and q are the attribute values for two data

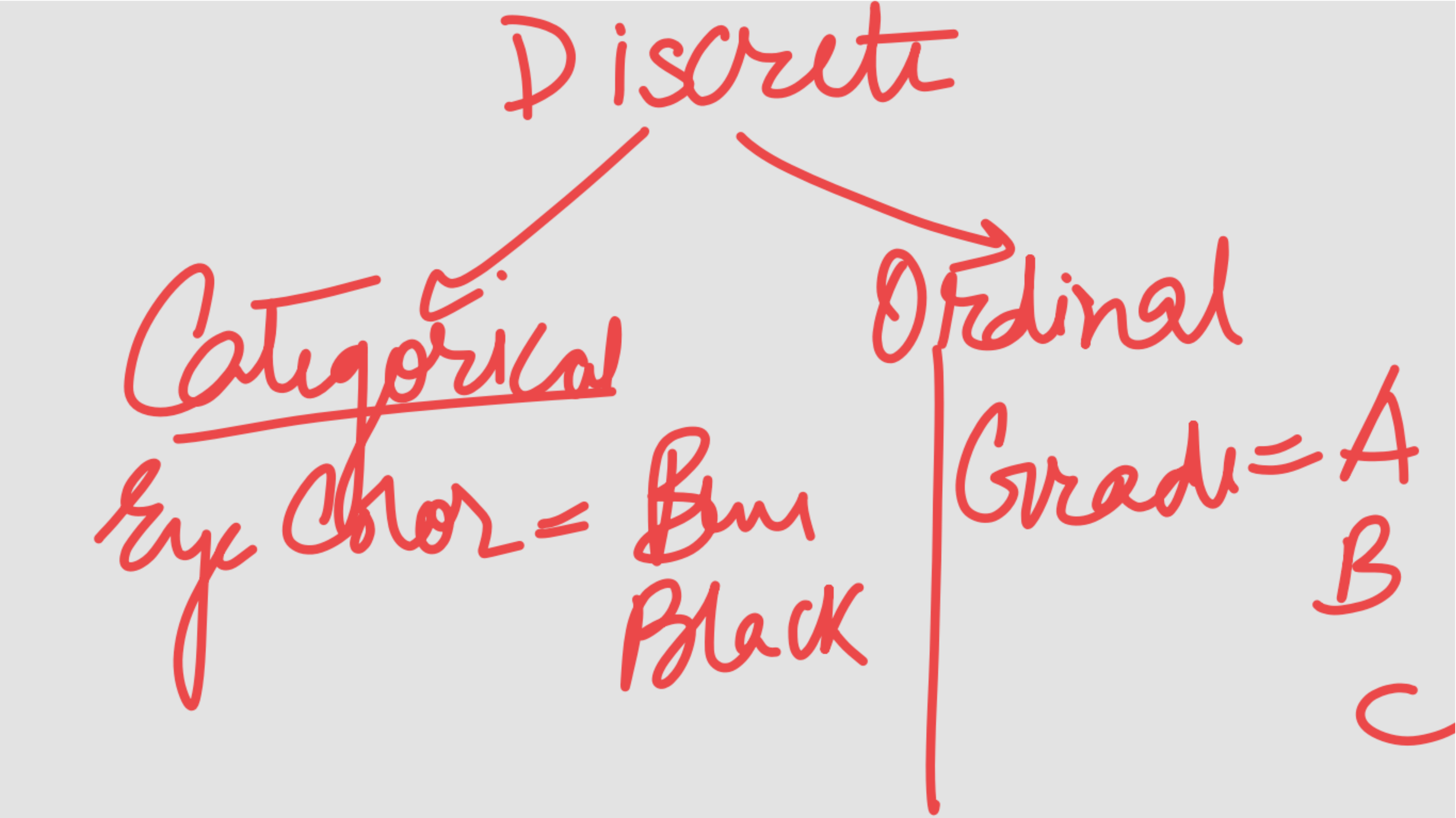
Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or

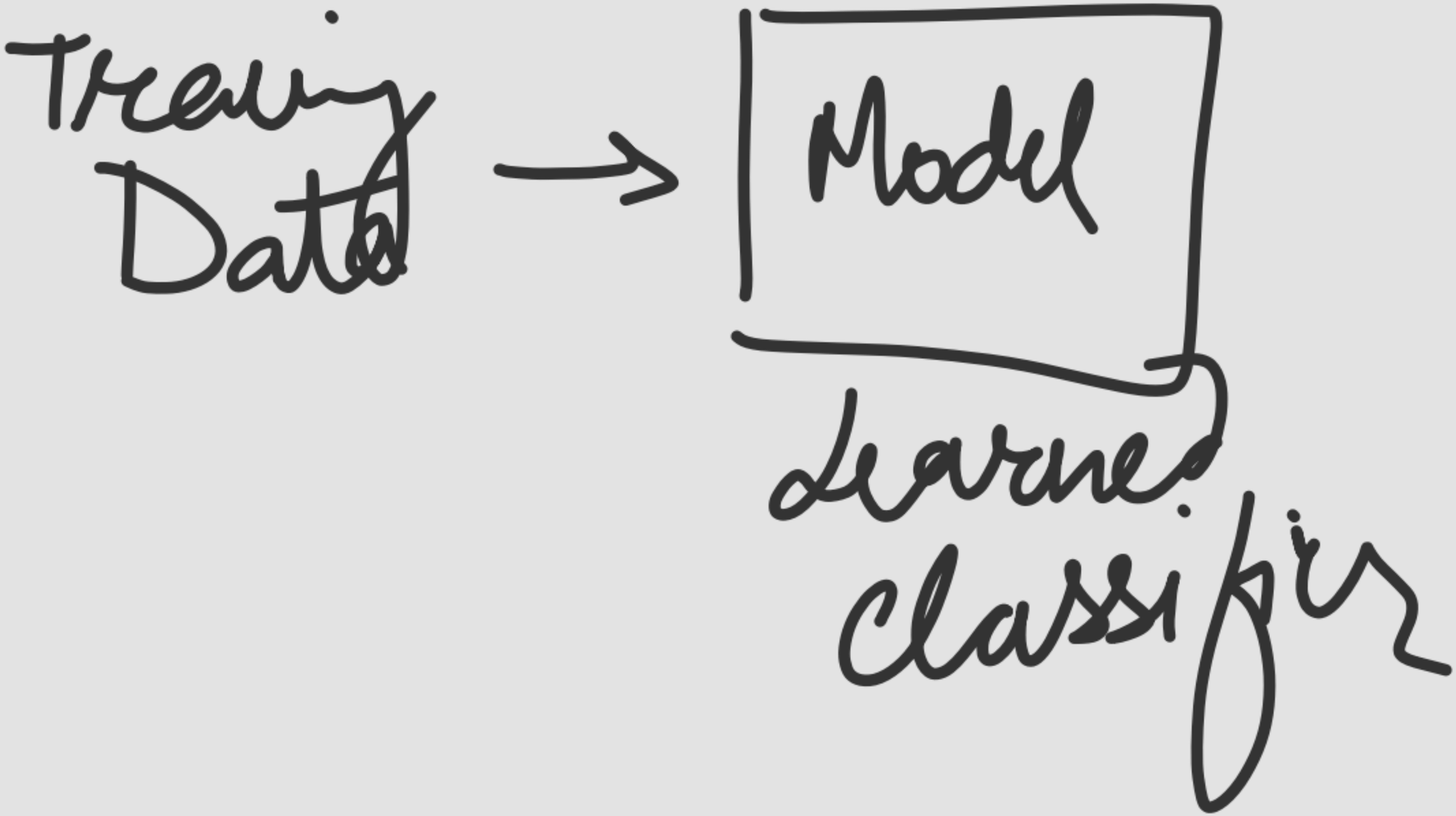
FEW POPULAR DISTANCE MEASURES!



Standardization

- In other situations, may want all features to be treated "equally"
  - One feature doesn't dominate another (income vs age)
- Common treatment to all variables:
  - Standardize each variable:
    - Mean = 0
    - Standard Deviation = 1


Discrete
Categorical
Eye Color = Brown
Black
Ordinal
Grade = A
B
C


Training Data
Model
learned classifier

Eager Learner Models

- So far in this course, we've performed prediction by:
  1. Taking a dataset
  2. Learning a model
  3. Using model to classify/predict test instances

- Sometimes called eager learners:
  - Designed to learn a ~~model that maps~~ the input attributes to the class label, as soon as training data becomes available.

Eg: Decision Tree, Logistic Regr

No model building

Lazy Learner Models

- Opposite strategy:
  - *Delay* process of modeling the training data, until it is necessary to classify/predict a test instance.
  - There is no "training" period, but each classification can be relatively expensive
  - Example: k-
    - Nearest neighbors

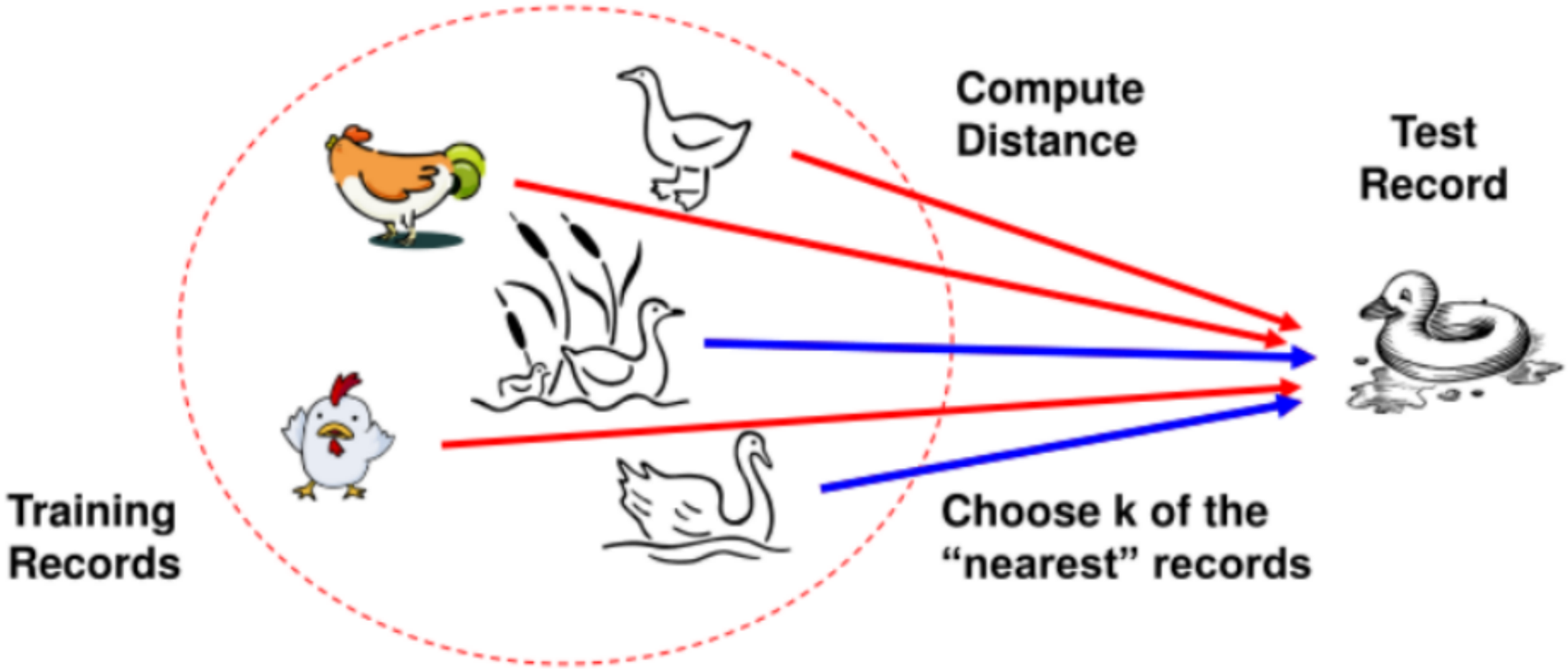
Nearest Neighbors

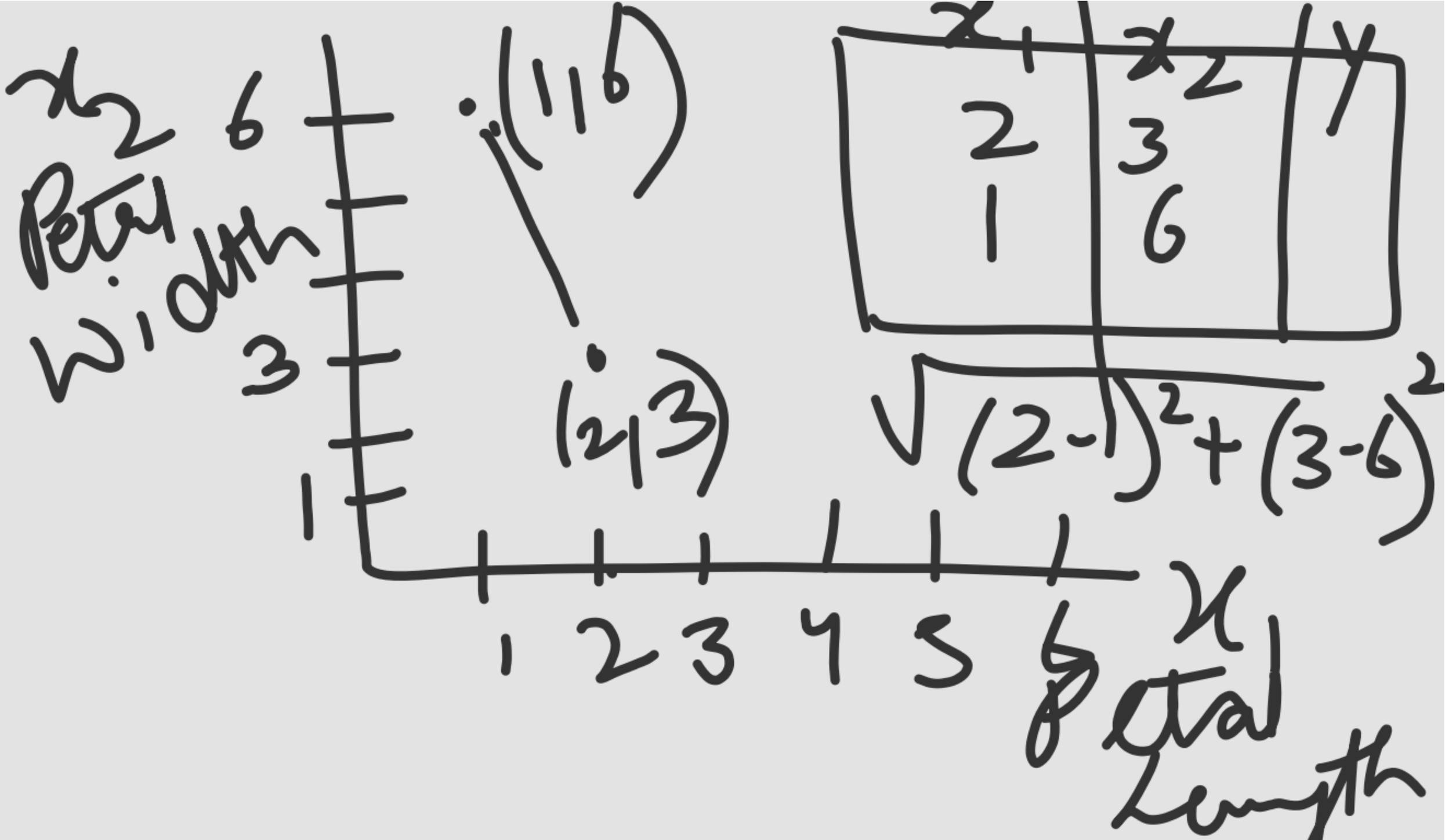
- *k*-nearest neighbors
  - *k* = parameter, chosen by analyst
  - For a given test instance, use the *k* "closest" points (nearest neighbors) for performing classification
    - "closest" points: defined by some proximity metric, such as Euclidean Distance

Find all the training examples that are relatively similar to the attributes of the test example. These examples, which are known as nearest neighbors, can be used to determine the class label of the test example!!

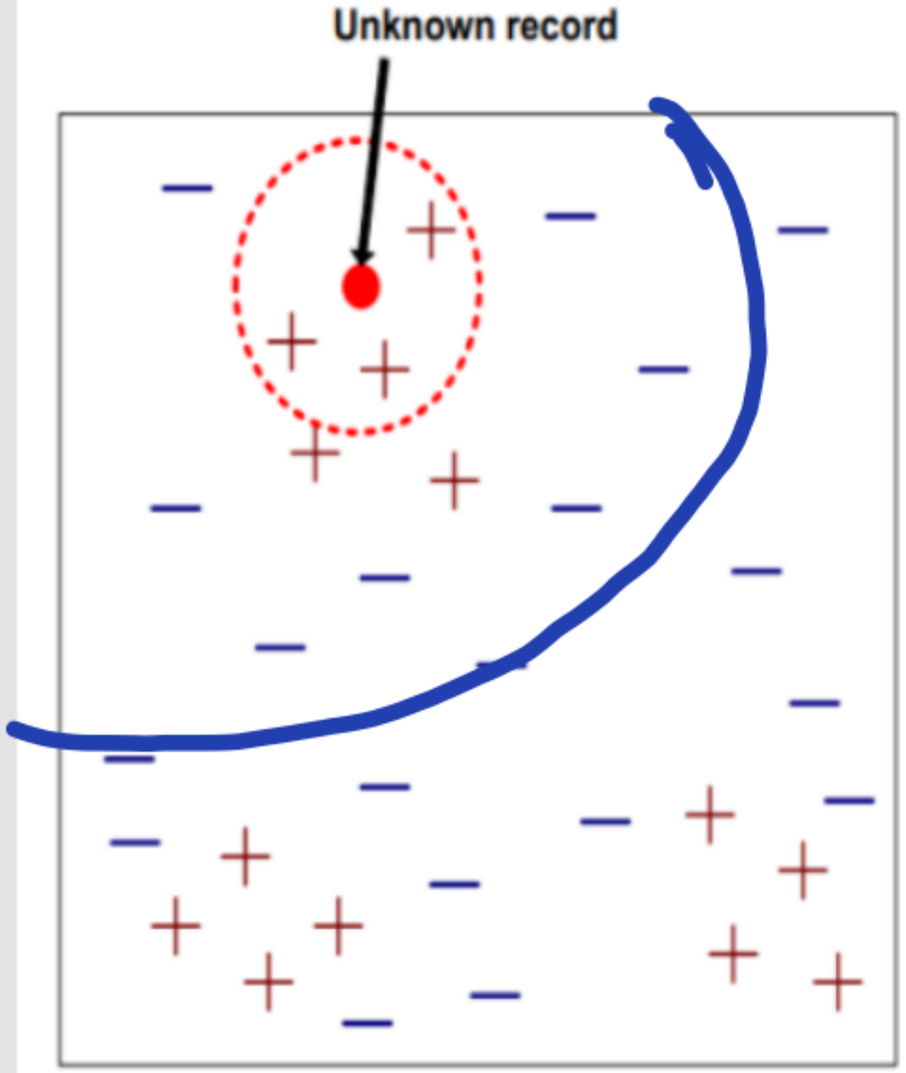
Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck




x2
Petal Width
6
3
1
(1|6)
(2|3)
1 2 3 4 5 6
x1
Total Length
x2
y
2
3
1
6
$\sqrt{(2-1)^2+(3-6)^2}$

Nearest-Neighbor Classifier

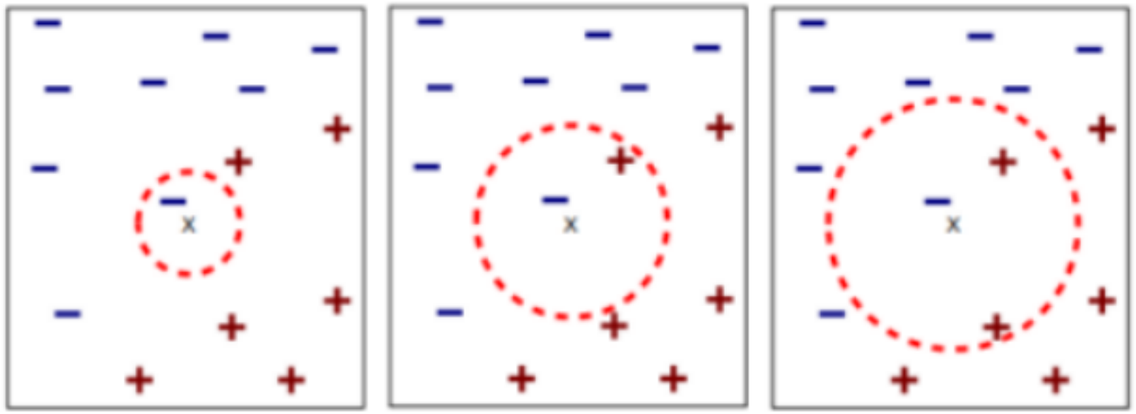


- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

- The *k*-nearest neighbors of a given example *x* are the *k* points that are closest to *x*.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

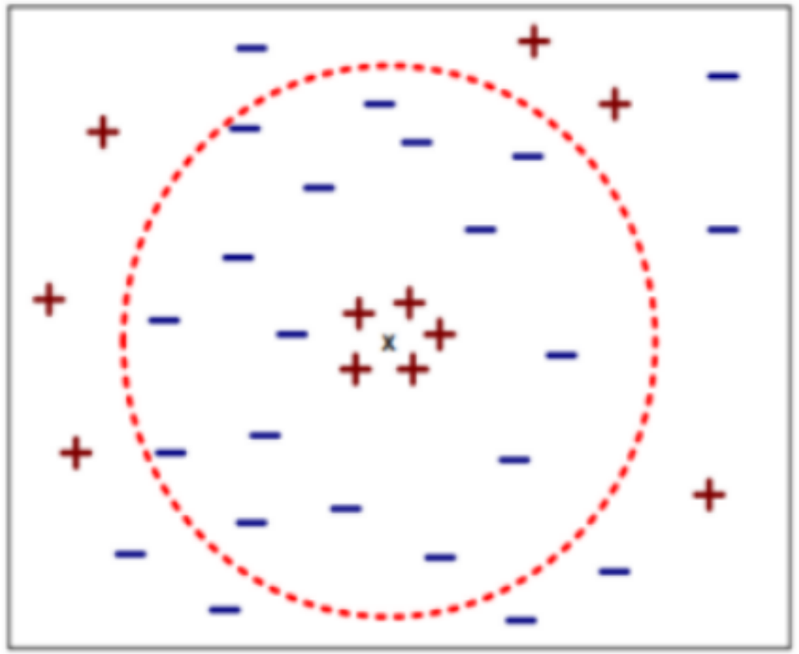
- Classification changes depending on the chosen *k*
- Majority Voting

Tie?

- Randomly choose classification.
- For binary problems, usually an odd *k* is used to avoid ties.

Choosing the right *k*

- If *k* is too small, sensitive to noise points in the training data
  - Susceptible to overfitting
- If *k* is too large, neighborhood may include points from other classes
  - Susceptible to misclassification



Algorithm

- Can't have a CS class without pseudocode!

Algorithm 5.2 The k-nearest neighbor classification algorithm.

1: Let k be the number of nearest neighbors and D be the set of training examples.
2: **for** each test example $z = (\mathbf{x}', y')$ **do**
3: Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every example, $(\mathbf{x}, y) \in D$.
4: Select $D_z \subseteq D$, the set of k closest training examples to z.
5: $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
6: **end for**

Computational Issues?

- Computation can be costly if the number of training examples is large.
- Efficient indexing techniques are available to reduce the amount of computations needed when finding the nearest neighbors of a test example

Majority Voting

where v is a class label, y_i is the class label for one of the nearest neighbors, and I(.) is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

- Simply majority: Every neighbor has the same impact on the classification:

$$\text{Majority Voting: } y' = \arg\max_v \sum_{(x_i, y_i) \in D_z} I(v = y_i)$$

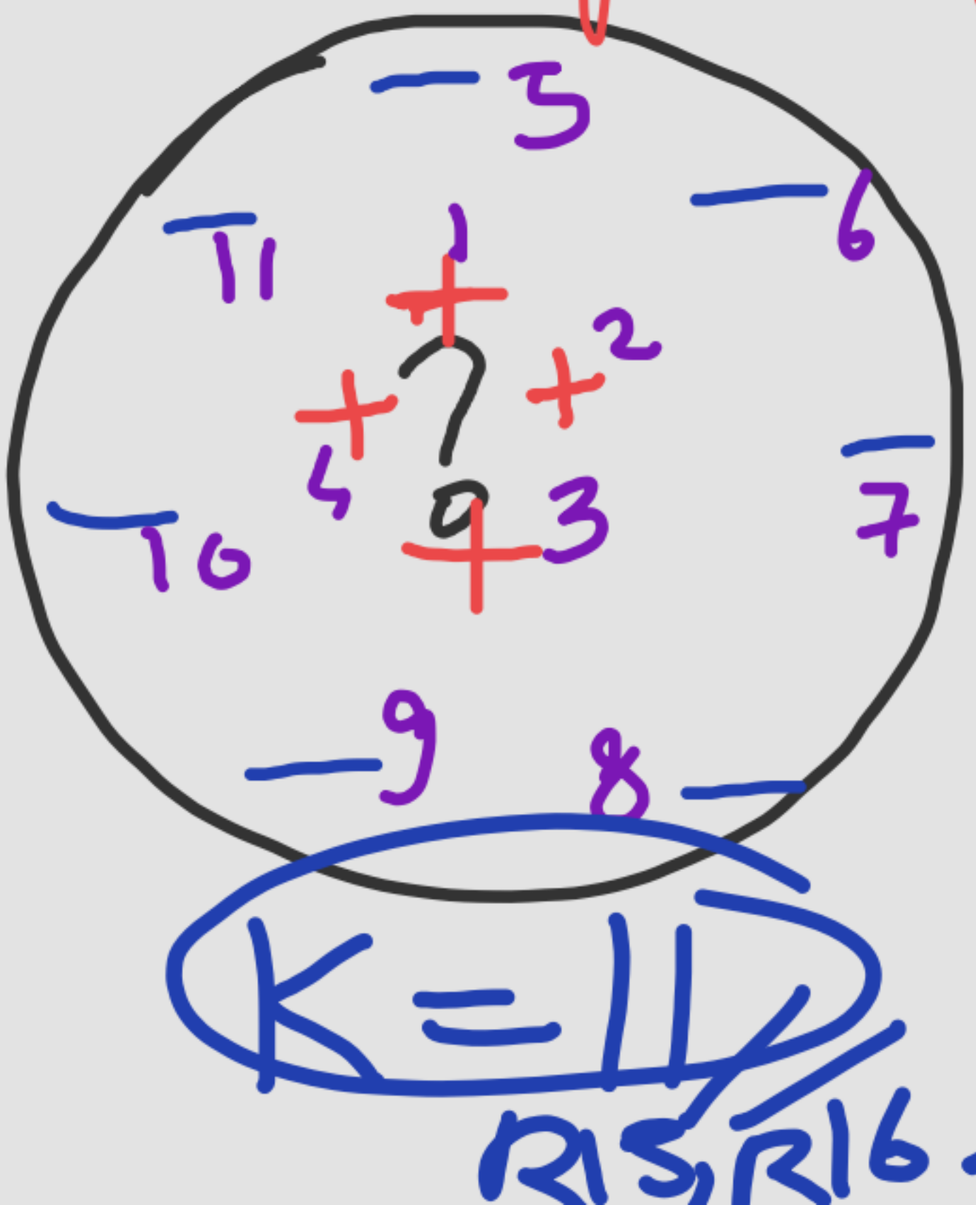
- Distance-weighted:
  - Far away neighbors have a weaker impact on the classification.

$$\text{Distance-Weighted Voting: } y' = \text{argmax}_v \sum_{(x_i, y_i) \in D_z} \omega_i \times I(v = y_i)$$

Way to reduce the impact of k is to weight the influence of each nearest neighbor x_i according to its distance: $w_i = {}^1/_{d(x', x_i)^2}$

$$w_i = \frac{1}{d(?, x(1))^2}$$

Majority Voting



	d(?, Ri)	w
R1	2	1
R2	2	1
R3	2	1
R4	2	.
R5	6	.
R6	6	.
R7	6	.
R8	6	.
R9	6	
R10	6	1
R11	6	

+04
-07

K. Majority class

R15, R16 - R50

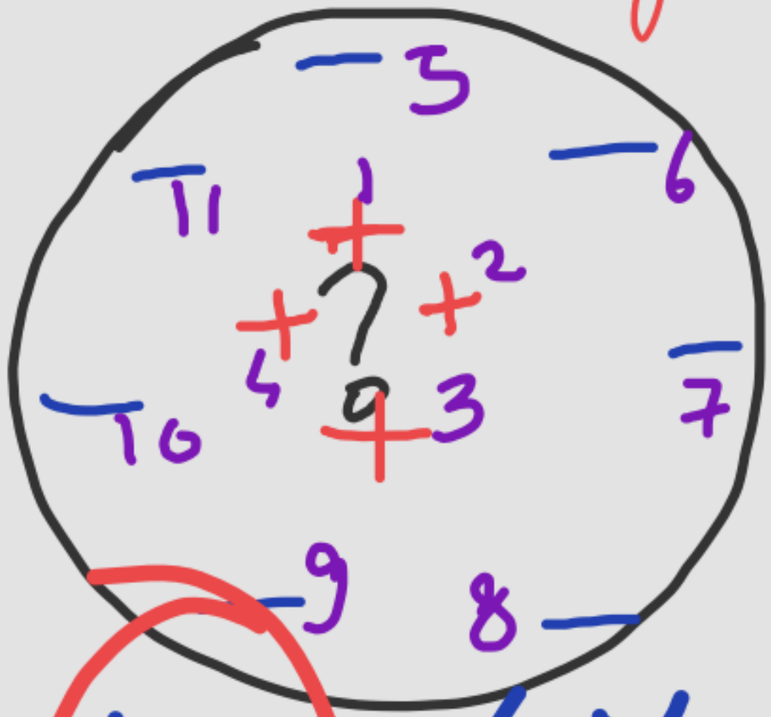
1. (+) + 1.(+) + 1(+) + 1(+)

= 4(+)

1.(−) + 1.(−) + 1(−) + 1(−) . . . .

= 7(−) → Majority Voting

Distance-Weighted Voting



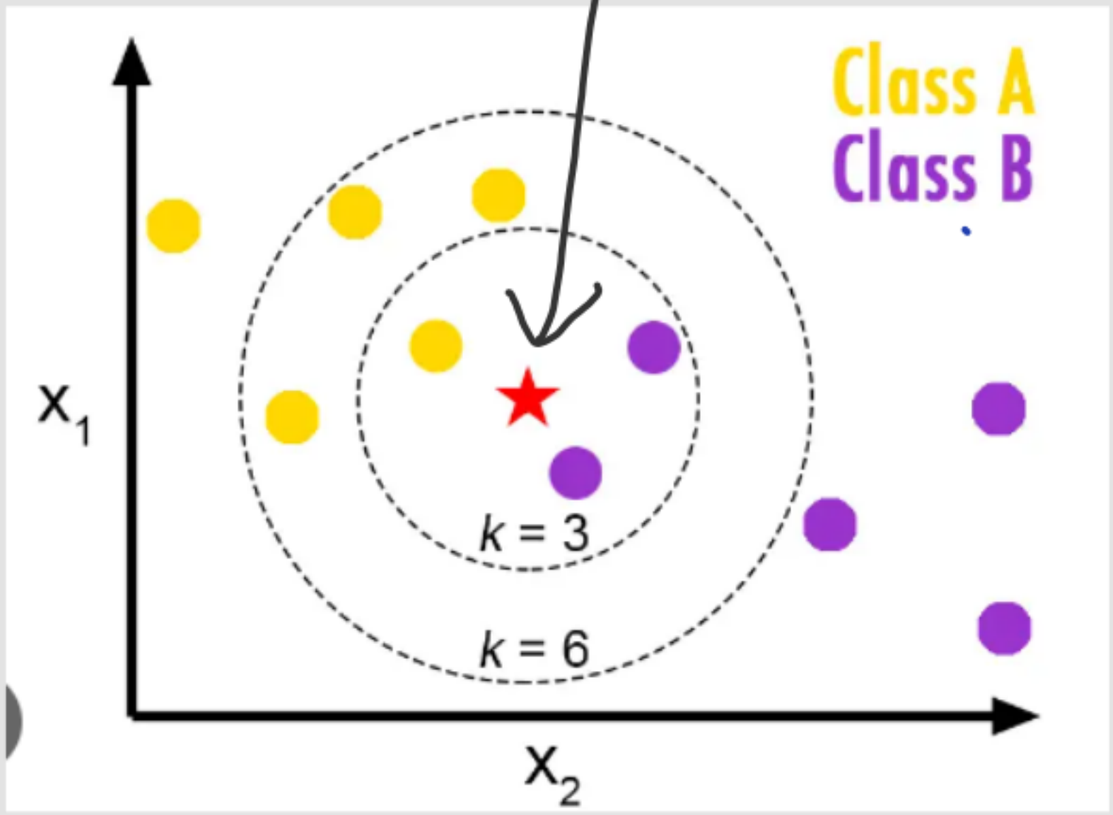
$$\frac{1}{d^2}$$

+: $4 \times \frac{1}{4}$

−: $7 \times \frac{1}{36}$

	d(?, Ri)	Weight
R1	2	1/4
R2	2	1/4
R3	2	1/4
R4	2	1/4
R5	6	1/36
R6	6	
R7	6	
R8	6	
R9	6	
R10	6	
R11	6	

K- nearest neighbors for this test instance



PROBLEM

Q. Given the dataset, classify the following data point:

(T, F, 1)

a1	a2	a3	Class
T	T	5	Y
T	T	7	Y
T	F	8	N
F	F	3	Y
F	T	7	N
F	T	4	N
F	F	5	N
T	F	6	Y
F	T	1	N

STEP 1: Determine K

RULE OF THUMB

Choose K = √No of training points in dataset

1. Better approach is to determine K by plotting a graph between different values of K and corresponding error rates

In this case: K = √9 = 3

STEP 2: Find distance from all the points

1. Distance based measure needs QUANTITATIVE Data
2. Convert Binary attribute to NUMERICAL form
3. Same for CATEGORICAL Attribute

a1	a1_num
T	1
T	1
T	1
F	0
F	0
F	0
F	0
F	0
T	1
F	0

Test Instance T = (T, F, 1)

= (1, 0, 1)

Dist (T, P_i)

	a1	a2	a3	Class
P1	1	1	5	Y
P2	1	1	7	Y
P3	1	0	8	N
P4	0	0	3	Y
P5	0	1	7	N
P6	0	1	4	N
P7	0	0	5	N
P8	1	0	6	Y
P9	0	1	1	N

→ $\sqrt{5}$

→ $\sqrt{11}$

→ $\sqrt{(1-0)^2+(0-1)^2+(1-1)^2}$ = $\sqrt{2}$

STEP 3:

Sort the points based on distances

STEP 4:

Gather K neighbors from the top

Point	Distance	Class
P9	√2	N
P4	√5	Y
P6	√11	N
P1	√17	Y
P7	√17	N
P8	√25	Y
P2	√37	Y
P5	√38	N
P3	√49	N

STEP 5: SIMPLE MAJORITY PREDICTION.

1. P9: N
2. P4: Y
3. P6: N

MAJORITY: 2N > 1Y

Scaling Issues

- Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
- Example, with three dimensions:
  - height of a person may vary from 1.5m to 1.8m
  - weight of a person may vary from 90lb to 300lb
  - income of a person may vary from $10K to $1M

Income will dominate if these variables aren't standardized.

We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with Default=Y.

D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> Default=Y

Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	$142,000	?		

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

Y

With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y.

Standardized Distance

One major drawback in calculating distance measures directly from the training set is in the case where variables have different measurement scales or there is a mixture of numerical and categorical variables. For example, if one variable is based on annual income in dollars, and the other is based on age in years then income will have a much higher influence on the distance calculated. One solution is to standardize the training set as shown below.

Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	0.61	?	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

Using the standardized distance on the same training set, the unknown case returned a different neighbor which is not a good sign of robustness.

Classifier Comparison

Eager Learners

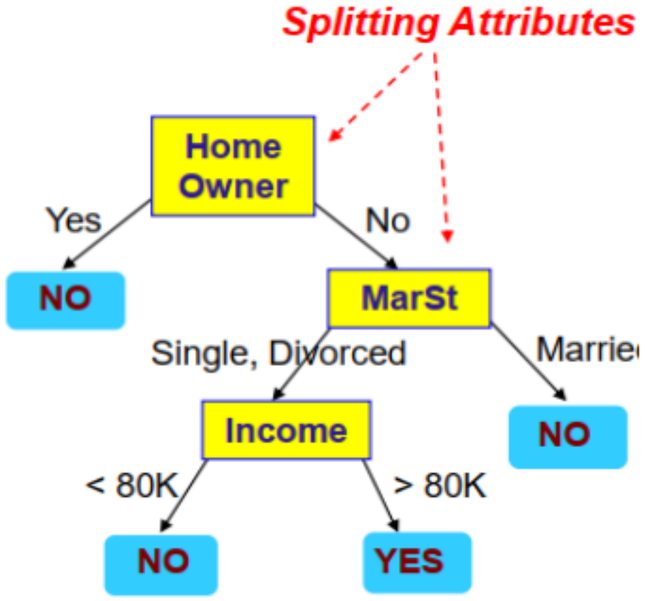
- Decision Trees, SVMs
- *finding a global model that fits the entire input space*

Lazy Learners

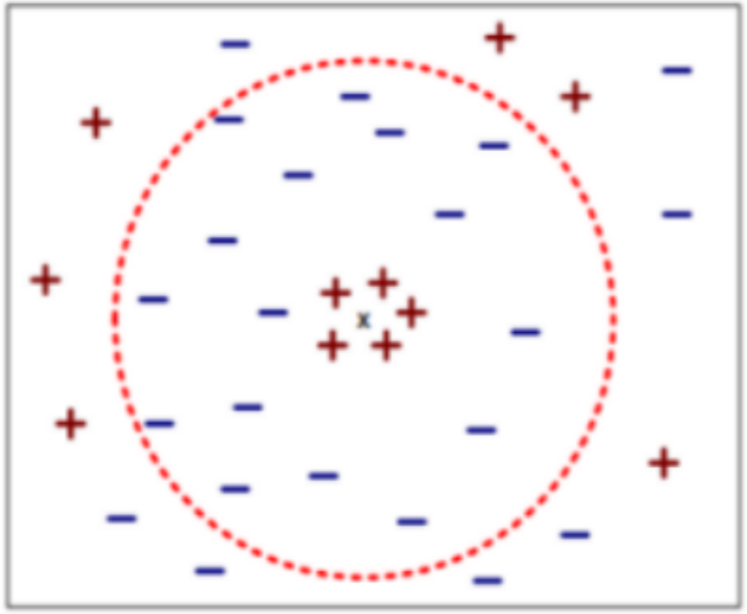
- Nearest Neighbors
- *classification decisions are made locally (small k values), are are more susceptible to noise*

	categorical	categorical	continuous	class
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree



Classifier Comparison

Eager Learners

- Decision Trees, SVMs
- Model Building: *potentially slow*
- Classifying Test Instance: *fast*

Lazy Learners

- Nearest Neighbors
- Model Building: *fast (because there is none!)*
- Classifying Test Instance: *slow*