Artificial Intelligence

Classification: Nearest Neighbor

tita@pens.ac.id

Supervised learning and classification

- Given : dataset of instances with known categories
- Goal : using the "knowledge" in the dataset, classify a given instance
  - predict the category of the given instance that is rationally consistent with the dataset

Algoritma klasifikasi

Tahapan dalam algoritma klasifikasi

- **Konstruksi model:** menguraikan suatu himpunan kelas yang ditentukan sebelumnya.
- **Penggunaan model:** setelah dibuat, model digunakan untuk mengklasifikasikan tuple data yag label kelasnya tidak diketahui

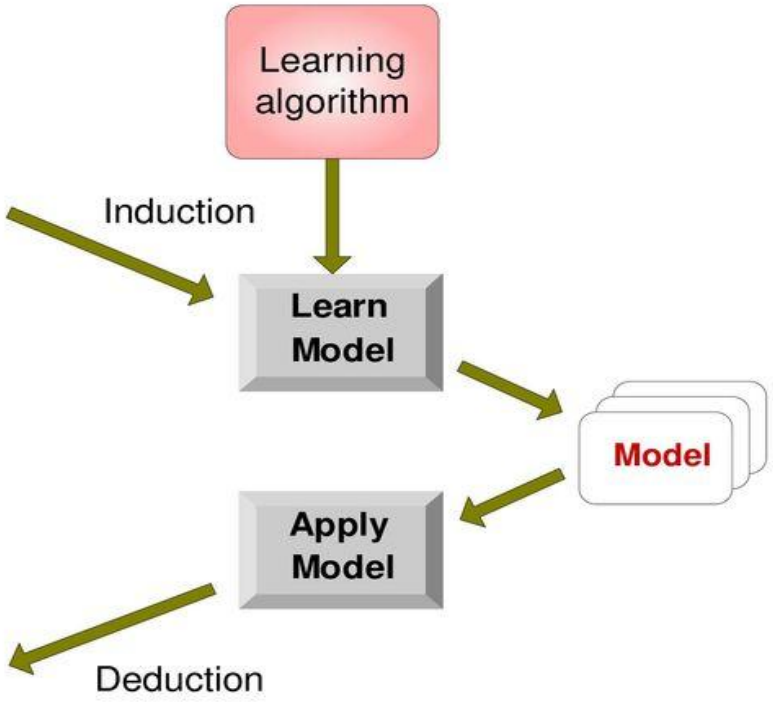
Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Contoh algoritma klasifikasi

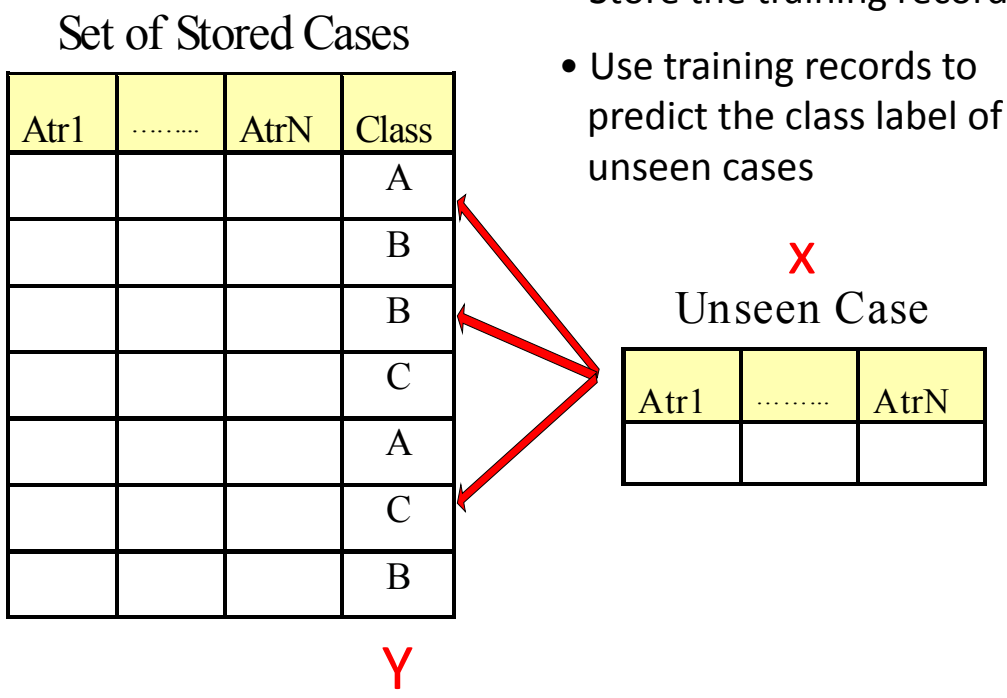
- Nearest neighbour
- Decision tree
- Bayesian network
- Adaptive bayesian network
- Naïve bayes
- dll

Instance Based Classifiers

- Also known as “memory-based” learning.
- Instance-based learning is often termed *lazy* learning, as there is typically no “transformation” of training instances into more general “statements”.
- Instead, the presented training data is simply stored and, when a new query instance is encountered, a set of similar, related instances is retrieved from memory and used to classify the new query instance
- Examples:
  - Nearest neighbor
    - Uses k “closest” points (nearest neighbors) for performing classification

Instance-Based Classifiers

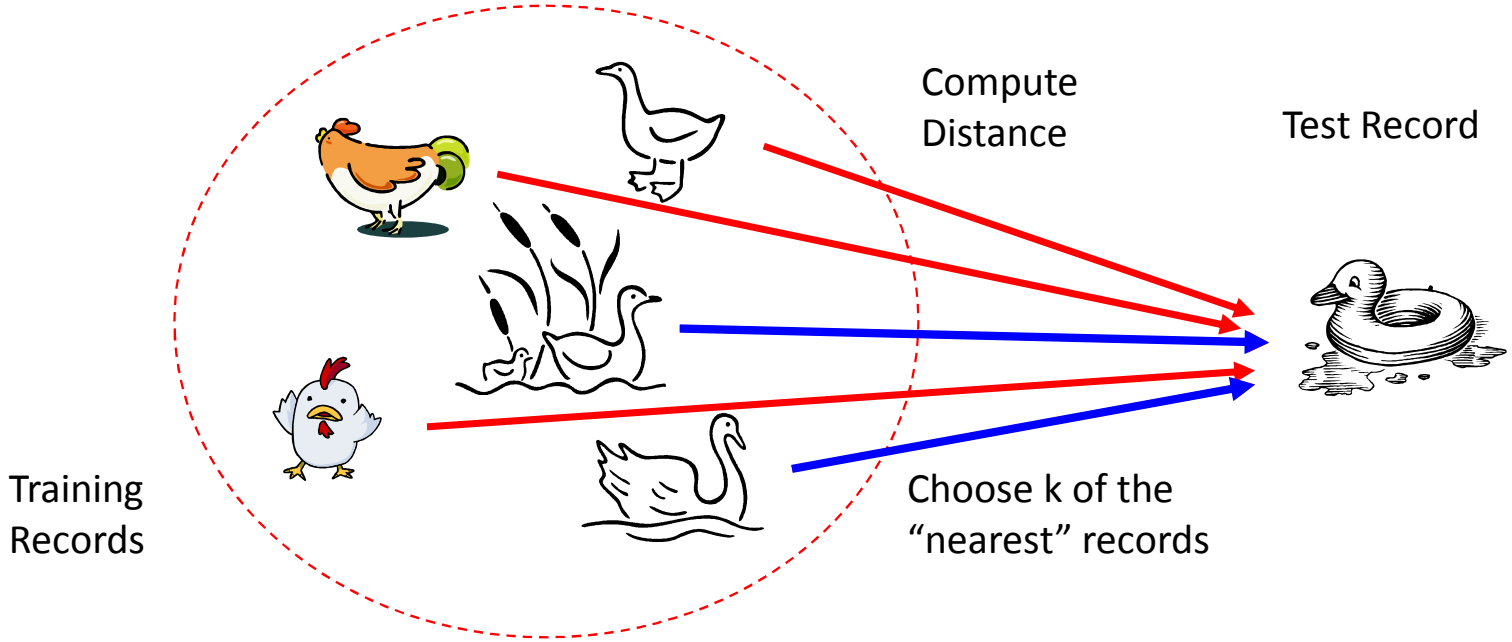
- Idea:
  - Similar examples have similar label.
  - Classify new examples like similar training examples.
- Algorithm:
  - Given some new example x for which we need to predict its class y
  - Find most similar training examples
  - Classify *x* "like" these most similar examples
- Questions:
  - How to determine similarity?
  - How many similar training examples to consider?
  - How to resolve inconsistencies among the training examples?



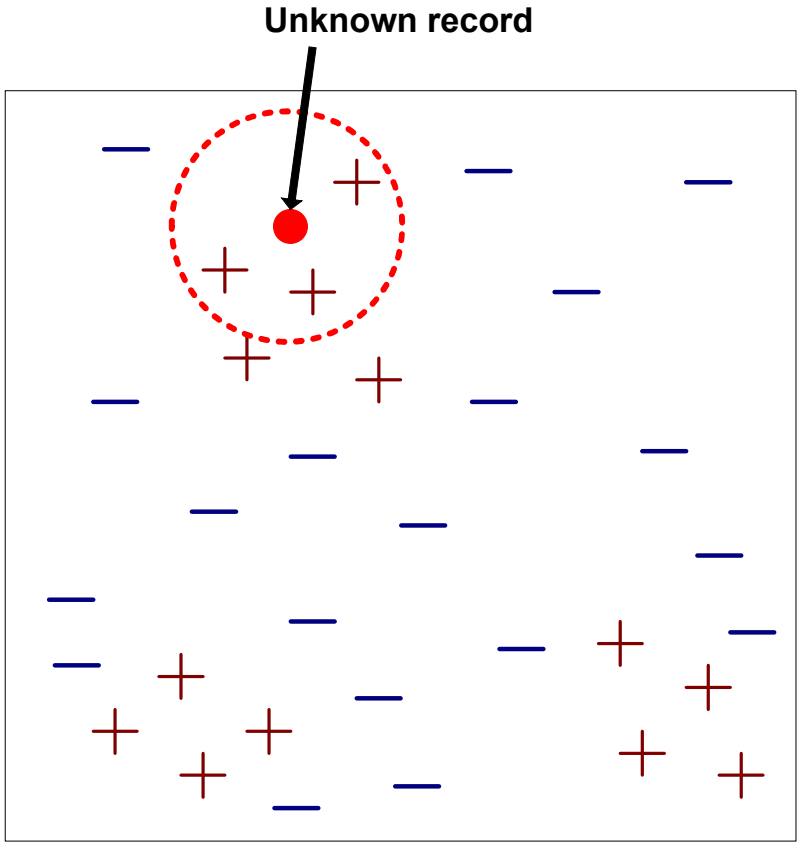
- Store the training records
- Use training records to predict the class label of unseen cases

Nearest Neighbor Classifiers

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



Nearest-Neighbor Classifiers

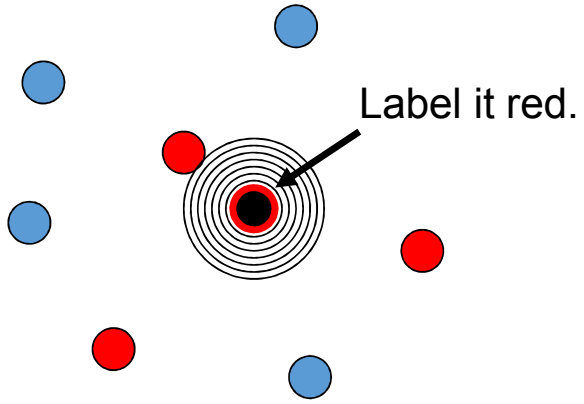


- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

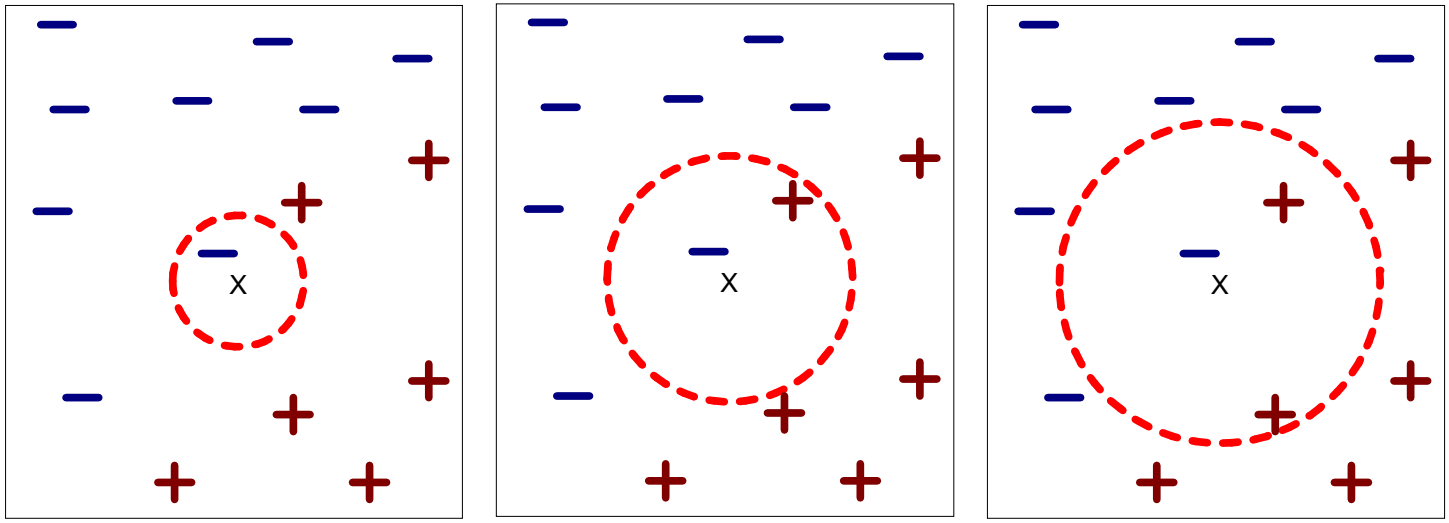
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., **by taking majority vote**)

1-Nearest Neighbor

- One of the simplest of all machine learning classifiers
- **Simple idea:** label a new point the same as the closest known point



Definition of Nearest Neighbor

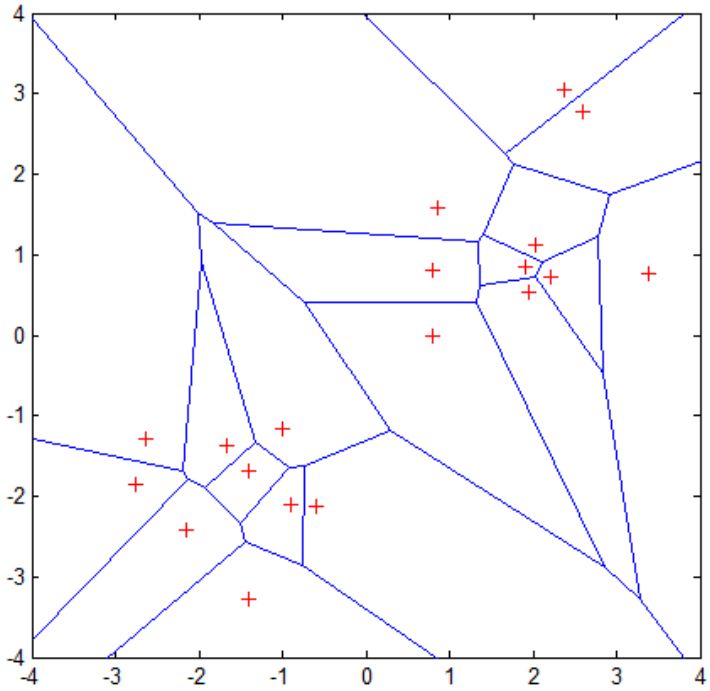


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

1 Nearest-Neighbor

Voronoi tessellation diagram defines the classification boundary



Properties:

1) All possible points within a sample's Voronoi cell are the nearest neighboring points for that sample
2) For any sample, the nearest sample is determined by the closest Voronoi cell edge

Distance Metric

- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Distance Metrics

Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r \right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic:

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T Q(\boldsymbol{x} - \boldsymbol{y}) = \sum_{j=1}^{m} \left(\sum_{i=1}^{m} (x_i - y_i) q_{ji} \right) (x_j - y_j)$$

Q is a problem-specific positive definite $m \times m$ weight matrix

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m} (\boldsymbol{x} - \boldsymbol{y})^{\mathrm{T}} V^{-1} (\boldsymbol{x} - \boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m} (x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m} (x_i - \overline{x_i})^2 \sum_{i=1}^{m} (y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i} \left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}} \right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_x$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)} \sum_{i=1}^{m} \sum_{j=1}^{i-1} \mathrm{sign}(x_i - x_j)\,\mathrm{sign}(y_i - y_j)$$

sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

Figure 1. Equations of selected distance functions.
($\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of m attribute values).

Issues with Distance Metrics

- Most distance measures were designed for linear/real-valued attributes
- Two important questions in the context of machine learning:
  - How best to handle nominal attributes
  - What to do when attribute types are mixed

Distance for Nominal Attributes

Value Difference Metric (VDM)

[Stanfill & Waltz, 1986]

Providing appropriate distance measurements for nominal attributes.

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2$$

$N_{a,x}$ = # times attribute a had value x

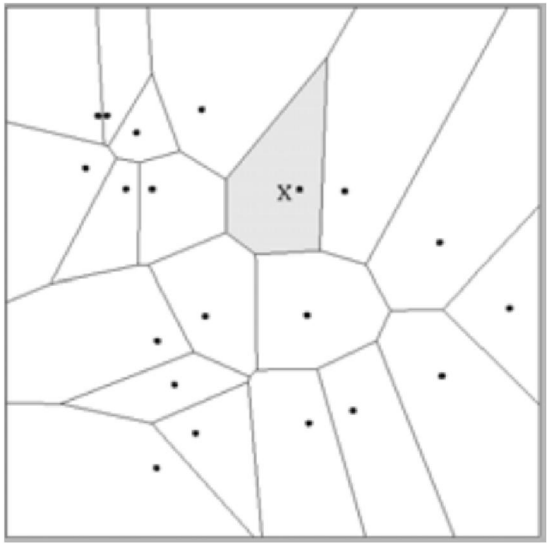
$N_{a,x,c}$ = # times attribute a had value x and class was c

C = # output classes

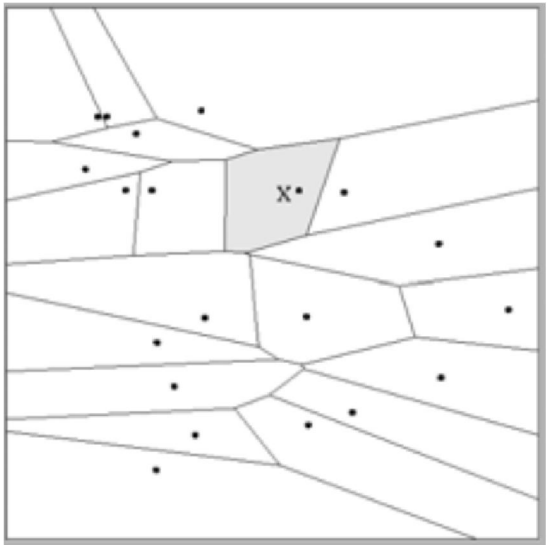
Two values are considered closer if they have more similar classifications, i.e., if they have more similar correlations with the output classes.

Distance Metrics

- Different metrics can change the decision surface



Dist(**a**,**b**) =$(a_1 - b_1)^2 + (a_2 - b_2)^2$



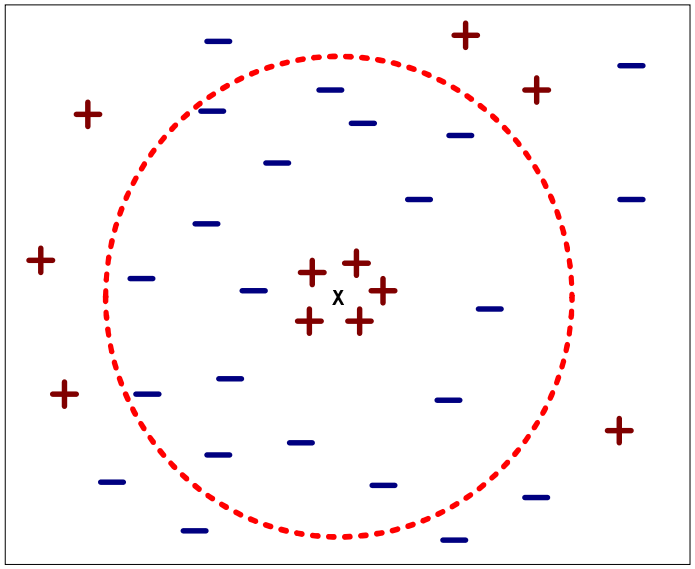
Dist(**a**,**b**) =$(a_1 - b_1)^2 + (3a_2 - 3b_2)^2$

- Standard Euclidean distance metric:
  - Two-dimensional: Dist(a,b) = sqrt($(a_1 - b_1)^2 + (a_2 - b_2)^2$)
  - Multivariate: Dist(a,b) = sqrt($\sum (a_i - b_i)^2$)

Adapted from "Instance-Based Learning" lecture slides by Andrew Moore, CMU.

Nearest Neighbor Classification…

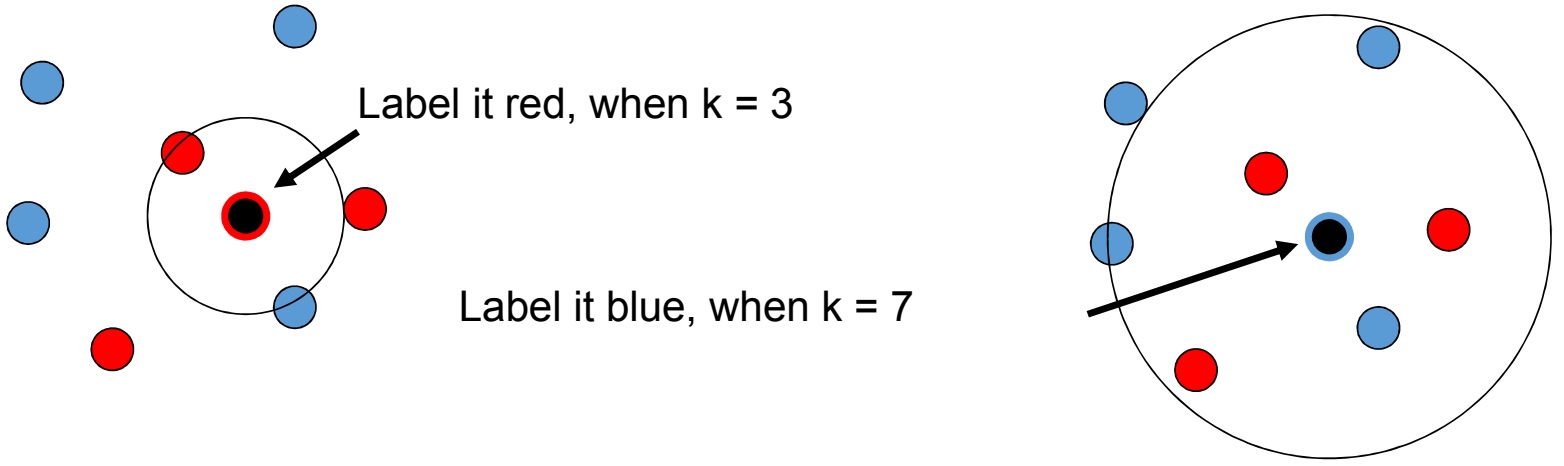
- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



k – Nearest Neighbor

(Avoiding noises)

- Generalizes 1-NN to smooth away noise in the labels
- A new point is now assigned the most frequent label of its *k* nearest **neighbors**



KNN Example

	Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
1	great	yes	yes	normal	no	yes
2	great	no	yes	normal	no	yes
3	mediocre	yes	no	high	no	no
4	great	yes	yes	normal	yes	yes

Similarity metric: Number of matching attributes (k=2)

- New examples:
  - Example 1 (great, no, no, normal, no) Yes

    → most similar: number 2 (1 mismatch, 4 match) → yes

    →Second most similar example: number 1 (2 mismatch, 3 match) → yes

  - Example 2 (mediocre, yes, no, normal, no) Yes/No

    → Most similar: number 3 (1 mismatch, 4 match) → no

    →Second most similar example: number 1 (2 mismatch, 3 match) → yes

Selecting the Number of Neighbors

- Increase k:
  - Makes KNN less sensitive to noise

- Decrease k:
  - Allows capturing finer structure of space

- ➔Pick k not too large, but not too small (depends on data)

Curse-of-Dimensionality

- Prediction accuracy can quickly degrade when number of attributes grows.
  - Irrelevant attributes easily “swamp” information from relevant attributes
  - When many irrelevant attributes, similarity/distance measure becomes less reliable

- Remedy
  - Try to remove irrelevant attributes in pre-processing step
  - Weight attributes differently
  - Increase k (but not too much)

Advantages and Disadvantages of KNN

- Need distance/similarity measure and attributes that “match” target function.

- For large training sets
  - Must make a pass through the entire dataset for each classification.
  - This can be prohibitive for large data sets.

- Prediction accuracy can quickly degrade when number of attributes grows.

Simple to implement algorithm;

Requires little tuning;

Often performs quite weel!

(Try it first on a new learning problem).

Scale Effects

- Different features may have different measurement scales
  - E.g., patient weight in kg (range [50,200]) vs. blood protein values in ng/dL (range [-3,3])
- Consequences
  - Patient weight will have a much greater influence on the distance between samples
  - May bias the performance of the classifier

Nearest Neighbor Classification…

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - 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

Standardization

- Transform raw feature values into z-scores

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

  - x_{ij} is the value for the i^{th} sample and j^{th} feature
  - μ_j is the average of all x_{ij} for feature j
  - σ_j is the standard deviation of all x_{ij} over all input samples
- Range and scale of z-scores should be similar

  (providing distributions of raw feature values are alike)

Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
- Classifying unknown records are relatively expensive
  - Naïve algorithm: O(n)
  - Need for structures to retrieve nearest neighbors fast.
    - The Nearest Neighbor Search problem.

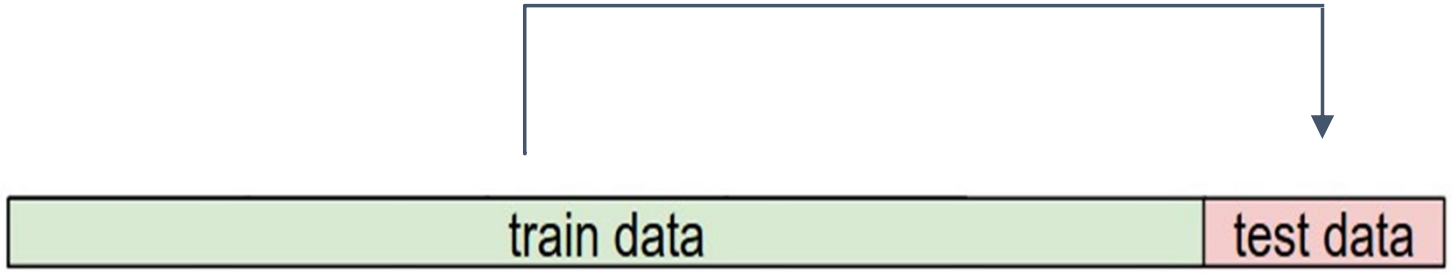
Hyperparameter tuning:

What is the best **distance** to use?
What is the best value of **k** to use?

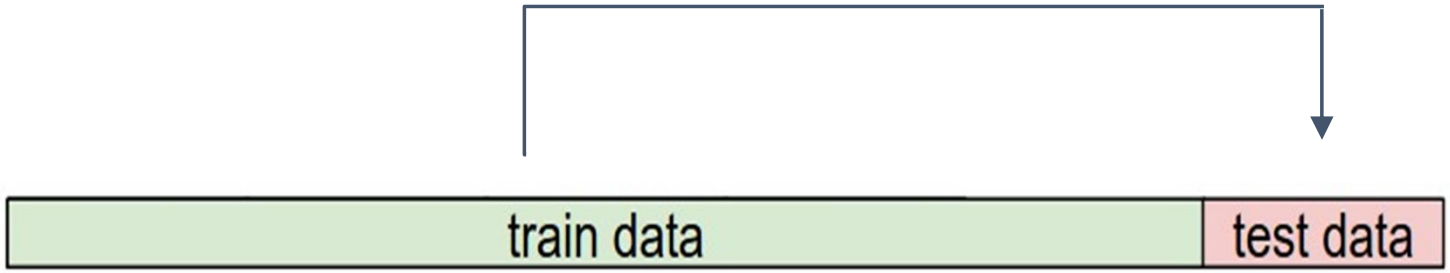
i.e. how do we set the **hyperparameters**?

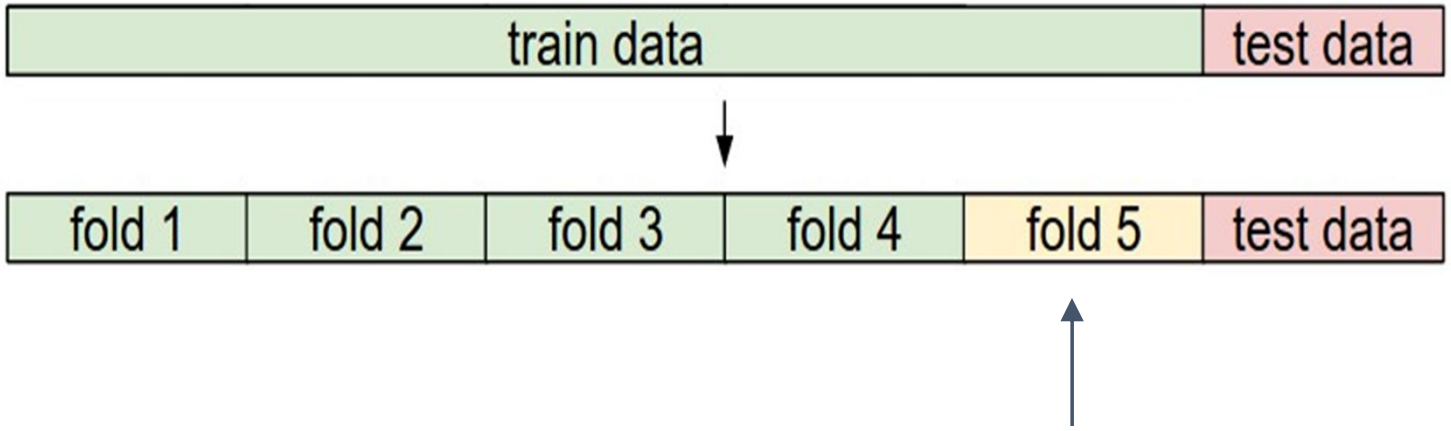
Very problem-dependent.
Must try them all out and see what works best.

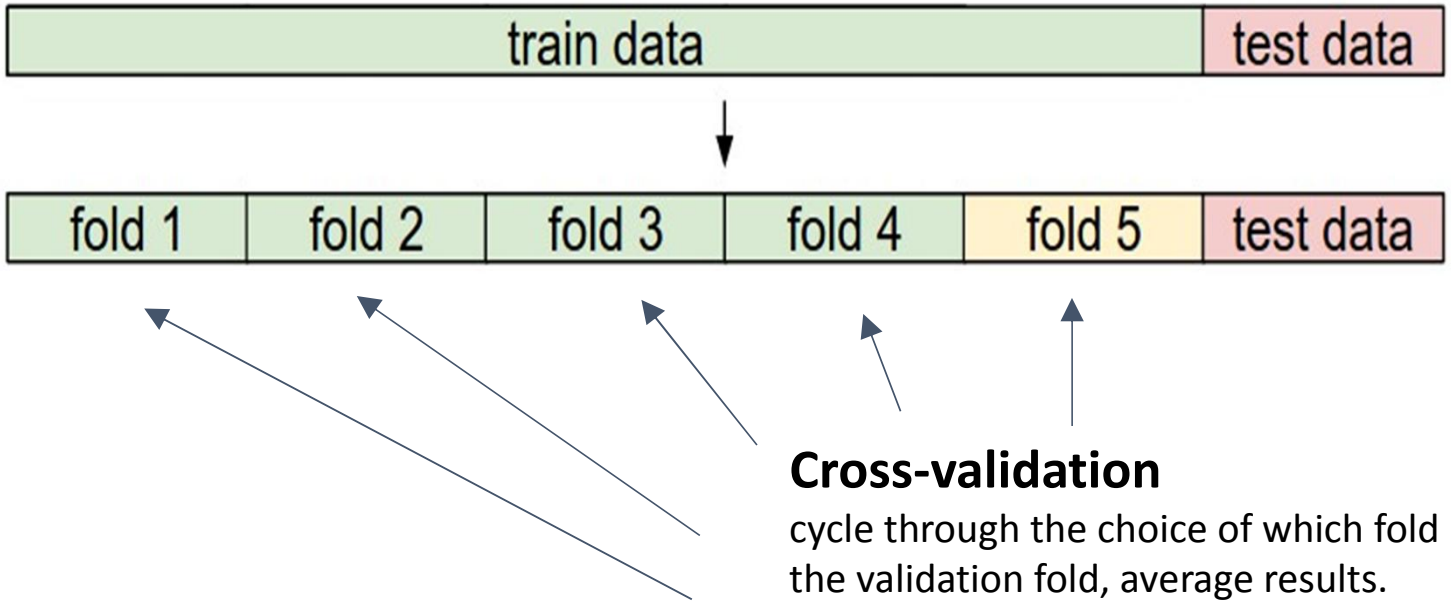
Try out what hyperparameters work best on test set.



Very bad idea. The test set is a proxy for the generalization performance!
Use only **VERY SPARINGLY,** at the end.



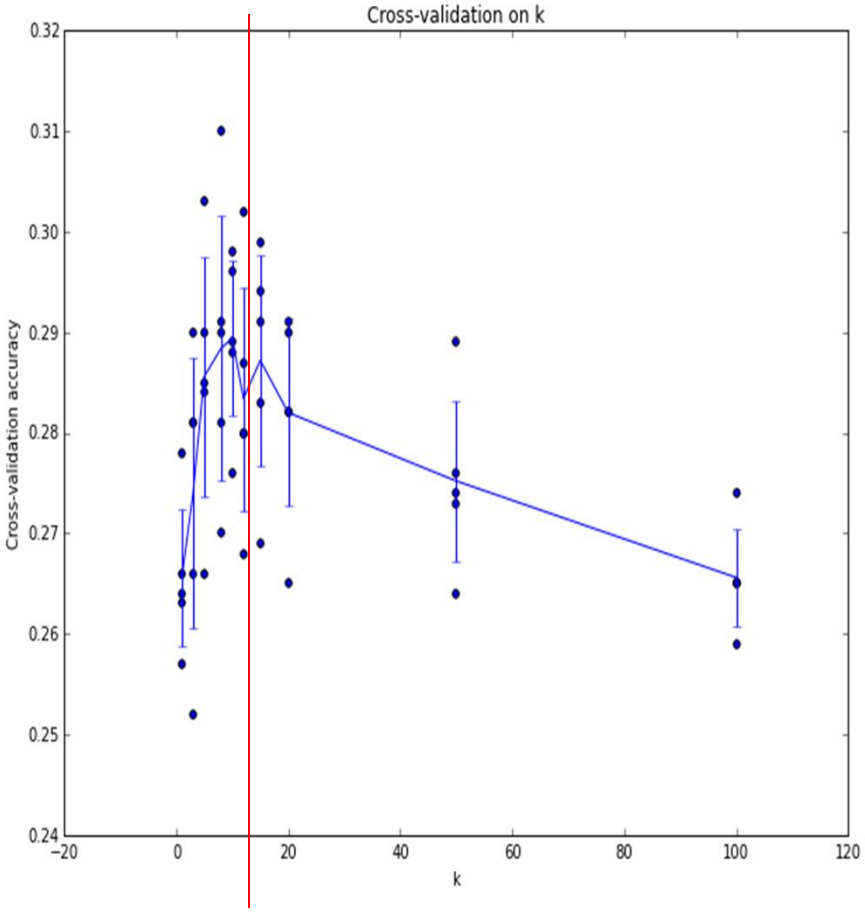

train data
test data
fold 1
fold 2
fold 3
fold 4
fold 5
test data
Validation data
use to tune hyperparameters



Cross-validation

cycle through the choice of which fold is the validation fold, average results.

Hyperparameter tuning:



Example of
5-fold cross-validation
for the value of **k.**

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best
for this data)

Summary

- **Classification:** given a **Training Set** of labeled data, predict labels on **Test Set**. Common to report the **Accuracy** of predictions (fraction of correct predictions)
- We introduced the **k-Nearest Neighbor Classifier**, which predicts labels based on nearest data in the training set
- We saw that the choice of distance and the value of k are **hyperparameters** that are tuned using a **validation set**, or through **cross-validation** if the size of the data is small.
- Once the best set of hyperparameters is chosen, the classifier is evaluated once on the test set, and reported as the performance of kNN on that data.

References

- Instanced based learning, BYU Data Mining Lab, http://dml.cs.byu.edu, Bringham Young University, 2013.
- Carla P. Gomes, Nearest Neighbor Models, CS4700: Foundations of Artificial Intelligence, http://www.cs.cornell.edu.
- Data Mining Lecture 10, Classification, www.cs.uoi.gr.
- John Canny, CS294-129: Designing, Visualizing and Understanding Deep Neural Networks, Lecture 2: Computer Vision and Image Classification, 2016.