Instance Based Learning (KNN Classifier)

Course Title: Machine Learning

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:		**Week No:**	**04**	**Semester:**	**Summer 2022-23**
Lecturer:	*Md Saef Ullah Miah (saef@aiub.edu)*				

Instance-Based Learning

- 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?

Instance-Based Classifiers

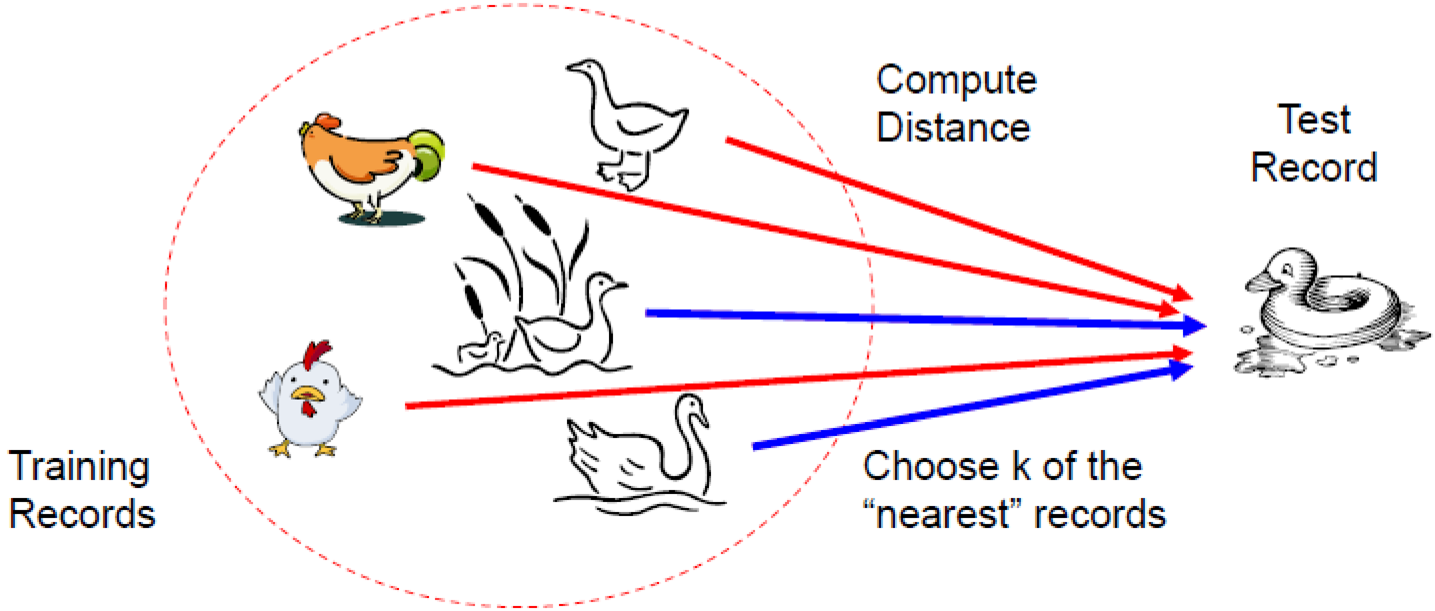
Examples:

- Rote-learner
  - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

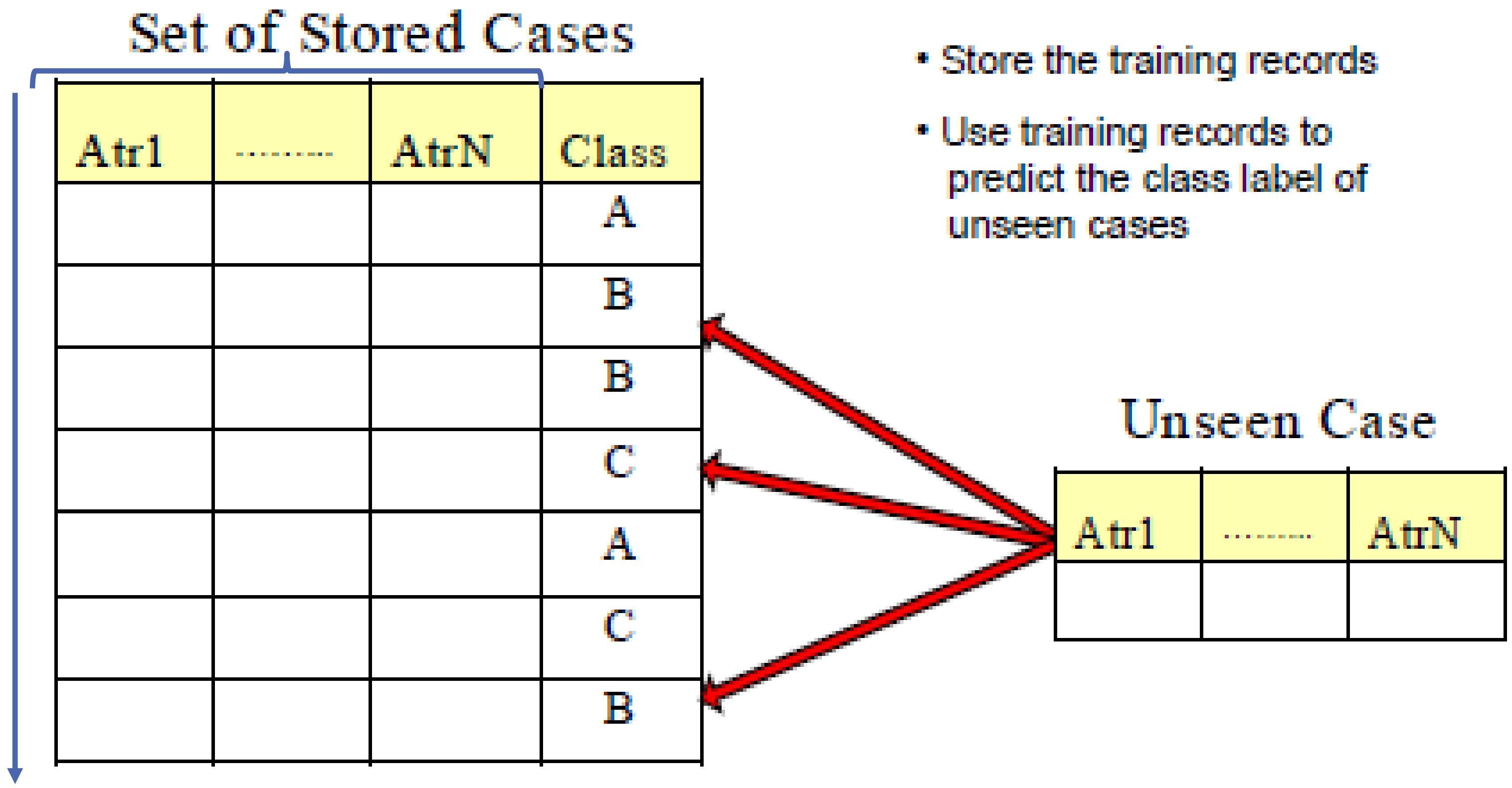
- Nearest neighbor
  - Uses k “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

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



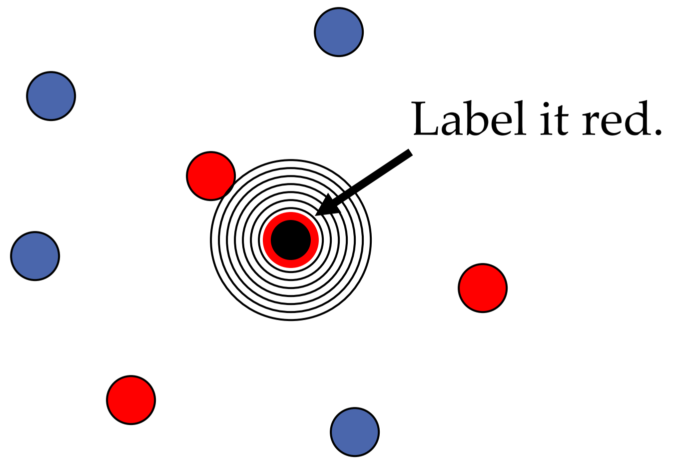
Instance-Based Classifiers



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

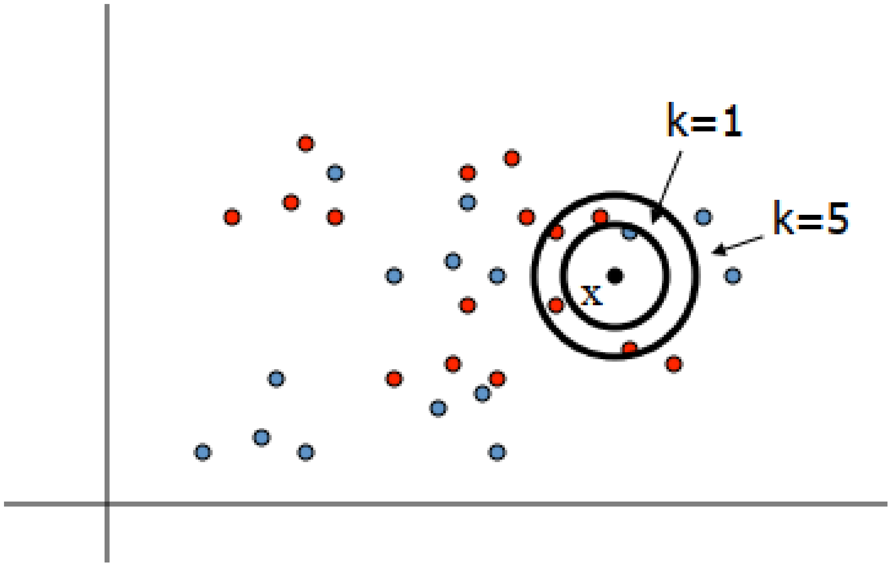
1-Nearest Neighbor

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



K-Nearest Neighbor Methods

- To classify a new input vector x, examine the k-closest training data points to x and assign the object to the most frequently occurring class

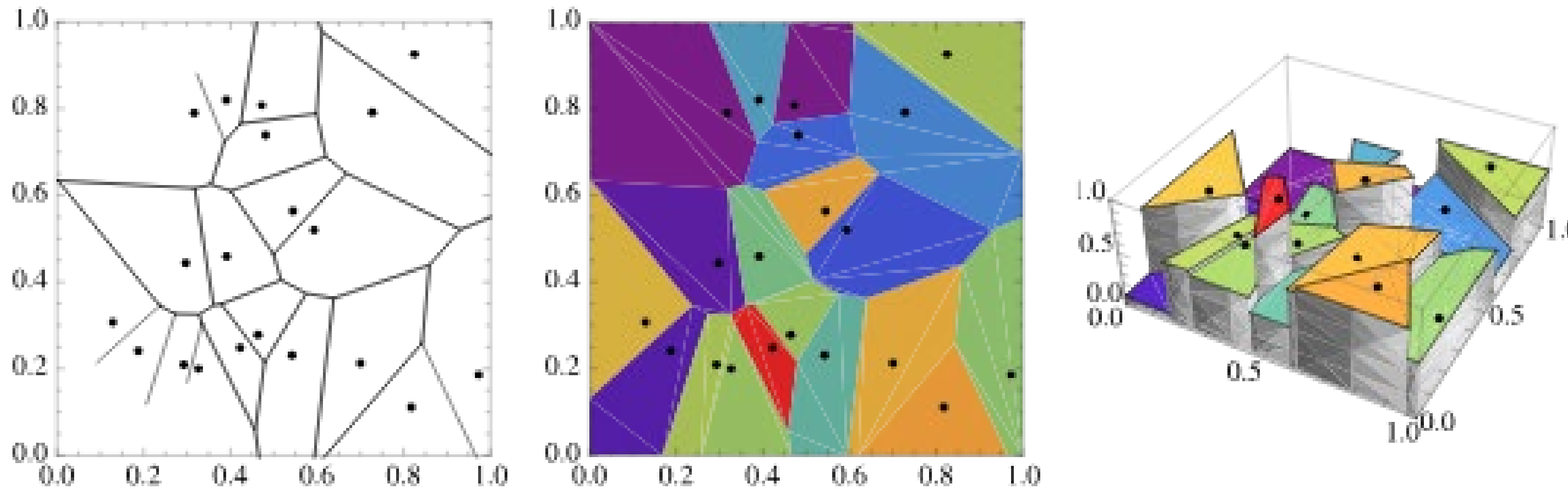


common values for k: 3, 5

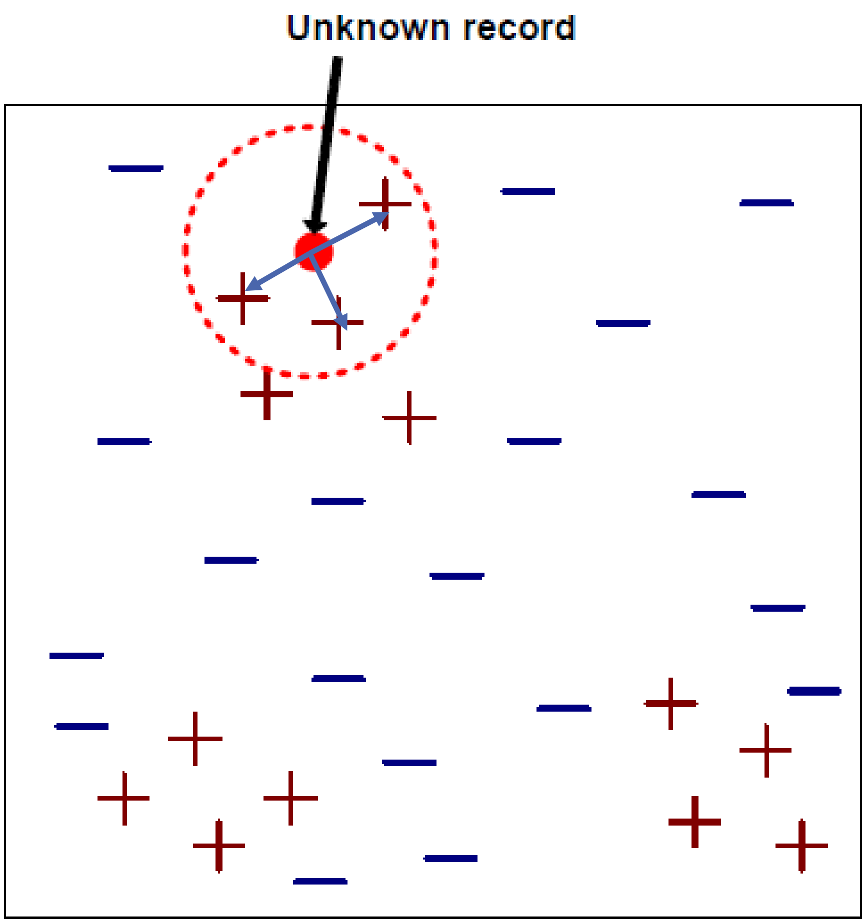
Decision Boundaries

- The nearest neighbor algorithm does not explicitly compute decision boundaries. However, the decision boundaries form a subset of the Voronoi diagram for the training data.

- The more examples that are stored, the more complex the decision boundaries can become



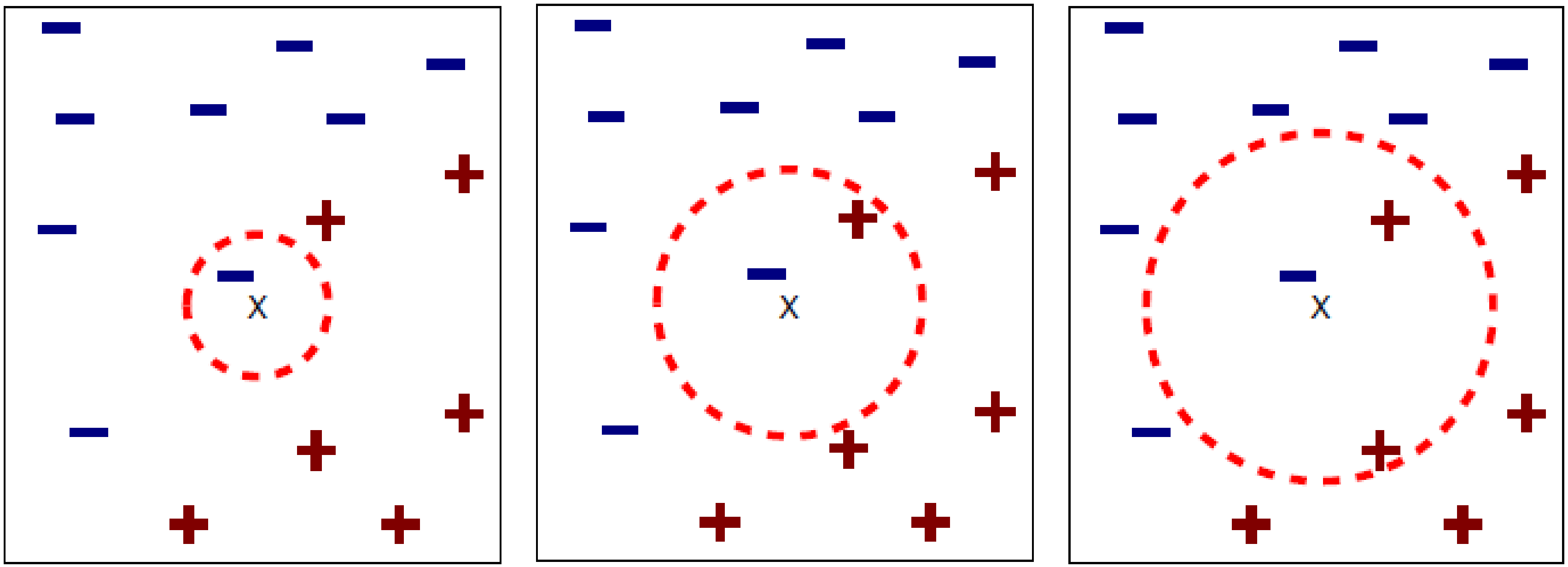
Nearest-Neighbor Classifiers



- 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)

3

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

Steps to determine a class using KNN

Let's assume you have a dataset with N training examples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where x_i represents the feature vector of the ith training example and y_i represents the corresponding class label. Given a new data point x_{new} for which you want to determine the class, the steps for KNN classification are as follows:

Steps to determine a class using KNN

Step 1: Calculate Distances

Calculate the distance between the new data point x_{new} and all the training examples using a distance metric such as Euclidean distance:

$$\text{Distance}(x_{\text{new}}, x_i) = \|x_{\text{new}} - x_i\|$$

Step 2: Find Nearest Neighbors

Sort the calculated distances in ascending order and select the k smallest distances. These k training examples are the nearest neighbors of x_{new}.

Step 3: Count Class Occurrences

Count the occurrences of each class among the k nearest neighbors.

Steps to determine a class using KNN

Step 4: Determine the Class

Assign the class label to x_{new} based on the majority class among its k nearest neighbors. If there is a tie, you can resolve it using various methods, such as considering more neighbors or using a distance-weighted approach.

Mathematically, if $y_{i_1}, y_{i_2}, \ldots, y_{i_k}$ are the class labels of the k nearest neighbors, you can determine the class y_{new} for the new data point x_{new} as follows:

$$y_{\text{new}} = \text{argmax}_y \sum_{j=1}^{k} I(y = y_{i_j})$$

Where:

- y_{new} is the predicted class label for x_{new}.
- $I(\text{condition})$ is an indicator function that equals 1 if the condition inside the parentheses is true, and 0 otherwise.
- y iterates over all possible class labels.

Example

Dataset:

Consider a dataset with three training examples:

1. $x_1 = (2, 3)$ with class label $y_1 = \text{Class A}$
2. $x_2 = (5, 6)$ with class label $y_2 = \text{Class B}$
3. $x_3 = (8, 9)$ with class label $y_3 = \text{Class A}$

New Data Point:

Let's say we have a new data point $x_{\text{new}} = (6, 7)$ for which we want to determine the class using KNN with $k = 1$ (considering only the nearest neighbor).

Example

Step 1: Calculate Distances:

Calculate the Euclidean distances between x_{new} and the training examples:

- Distance from x_{new} to x_1: $||x_{\text{new}} - x_1|| = \sqrt{(6-2)^2 + (7-3)^2} = \sqrt{16+16} = \sqrt{32} \approx 5.66$
- Distance from x_{new} to x_2: $||x_{\text{new}} - x_2|| = \sqrt{(6-5)^2 + (7-6)^2} = \sqrt{1+1} = \sqrt{2} \approx 1.41$
- Distance from x_{new} to x_3: $||x_{\text{new}} - x_3|| = \sqrt{(6-8)^2 + (7-9)^2} = \sqrt{4+4} = \sqrt{8} \approx 2.83$

Example

Step 2: Find Nearest Neighbor:

The smallest distance is $\sqrt{2}$, which corresponds to the distance between x_{new} and x_2. Therefore, x_2 is the nearest neighbor.

Step 3: Count Class Occurrences:

The class label of x_2 is Class B. There is 1 occurrence of Class B.

Step 4: Determine the Class:

Since $k = 1$, the class of x_{new} is Class B because the nearest neighbor is in Class B.

Therefore, using KNN with $k = 1$, the algorithm predicts that the class of the new data point $x_{\text{new}} = (6, 7)$ is Class B.

Example

- What about K=3?

Step 1: Calculate Distances:

As calculated previously:

- Distance from x_{new} to x_1: $\sqrt{32} \approx 5.66$
- Distance from x_{new} to x_2: $\sqrt{2} \approx 1.41$
- Distance from x_{new} to x_3: $\sqrt{8} \approx 2.83$

Step 2: Find Three Nearest Neighbors:

The three smallest distances correspond to x_2, x_3, and x_1, in that order.

Example

Step 2: Find Three Nearest Neighbors:

The three smallest distances correspond to x_2, x_3, and x_1, in that order.

Step 3: Count Class Occurrences:

Among the three nearest neighbors:

- x_2 is in Class B.
- x_3 is in Class A.
- x_1 is in Class A.

So, there is 1 occurrence of Class B and 2 occurrences of Class A.

Step 4: Determine the Class:

Since $k = 3$ and two out of three nearest neighbors belong to Class A, the majority class among the three nearest neighbors is Class A. Therefore, using KNN with $k = 3$, the algorithm predicts that the class of the new data point $x_{\text{new}} = (6, 7)$ is Class A.

Nearest Neighbor Classification

- Compute distance between two points:
  - Euclidean distance

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

  - Manhatten distance

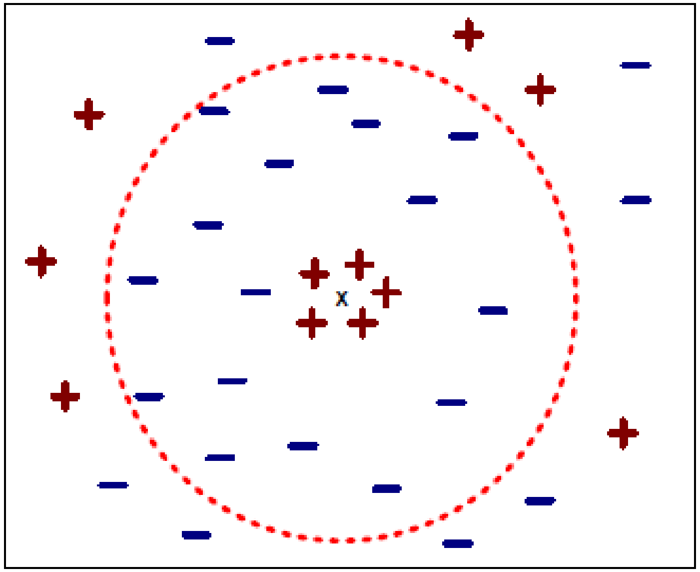
$$d(p,q) = \sum_i |p_i - q_i|$$

  - q norm distance

$$d(p,q) = \left(\sum_i |p_i - q_i|^q\right)^{1/q}$$

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



Nearest Neighbor Classification…

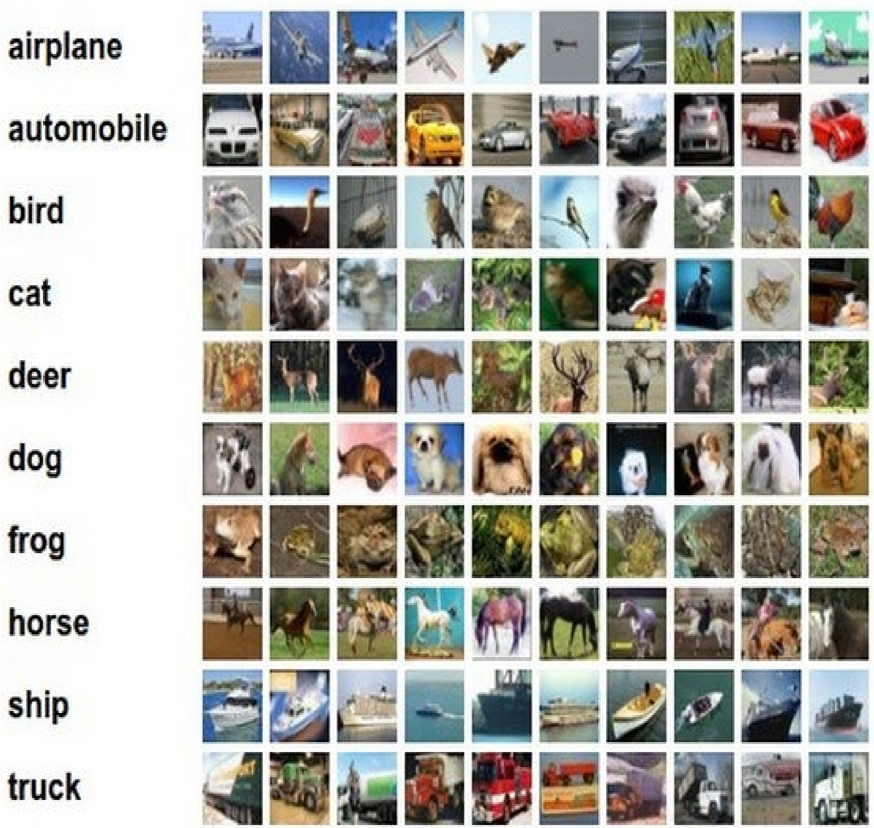
- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 60 KG to 100KG
    - income of a person may vary from Rs10K to Rs 2 Lakh

Example dataset: CIFAR-10

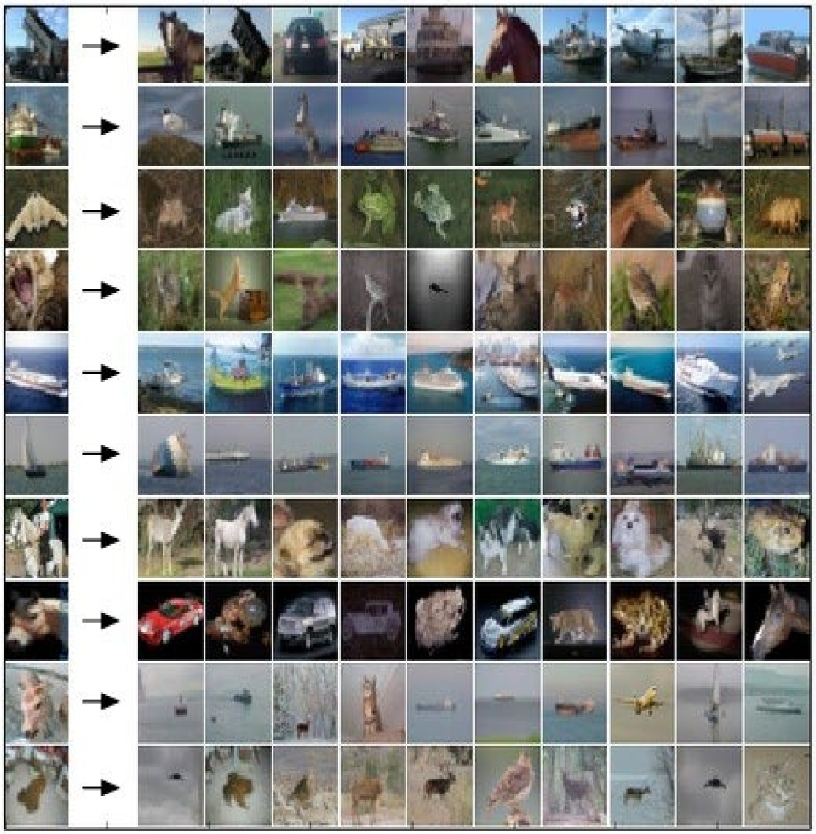
10 labels
50,000 training images, each image is tiny: 32x32
10,000 test images.

Example dataset: CIFAR-10

10 labels
50,000 training images
10,000 test images.



For every test image (first column), examples of nearest neighbors in rows



The choice of distance is a **hyperparameter** common choices:

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p \left| I_1^p - I_2^p \right|$$

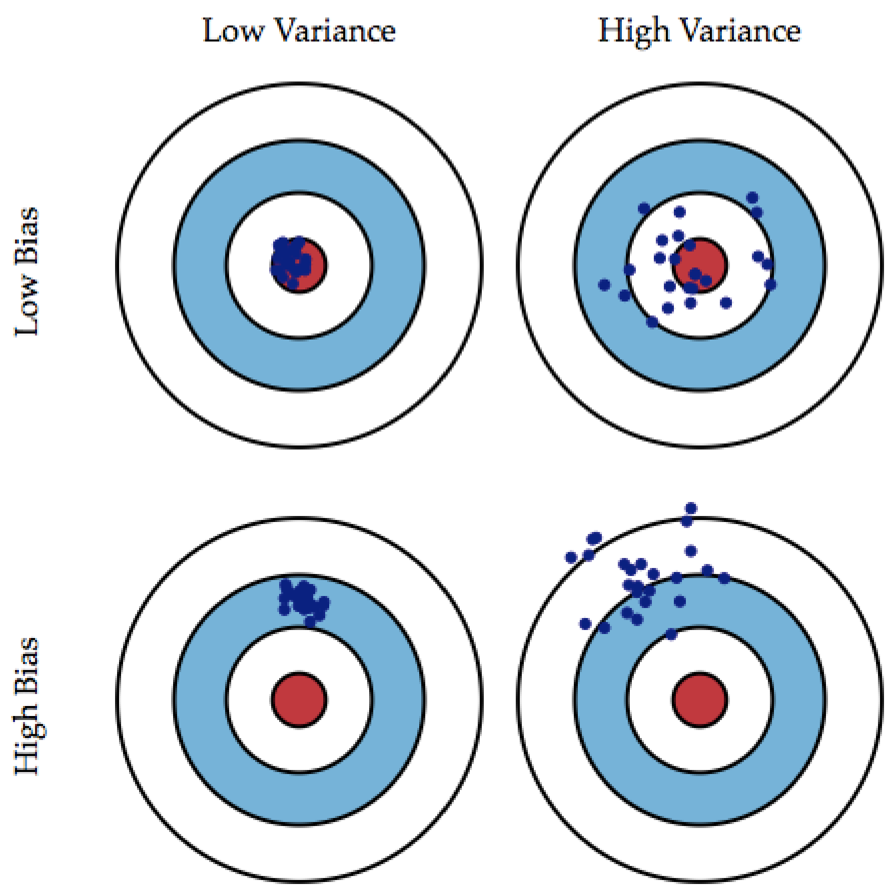
L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p \right)^2}$$

Bias and Variance

*The **bias** is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs.*

*The **variance** is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs.*


Low Variance
High Variance
Low Bias
High Bias

Bias and Variance

Increasing k in the kNN algorithm should have what effect on:

Bias: ?

Variance: ?

Bias and Variance

Increasing k in the kNN algorithm should have what effect on:

Bias: Should **increase**. The large k is, the further (on average) are the points being aggregated from x

Variance: ?

Bias and Variance

Increasing k in the kNN algorithm should have what effect on:

Bias: Should increase. The large k is, the further (on average) are the points being aggregated from x. i.e. the value depends on f(x’) for x’ further from x.

Variance: Should **decrease**. The average or majority vote of a set of equal-variance values has lower variance than each value.

Bias and Variance Rule of Thumb

Compared to simpler (fewer parameters), complex models have what kind of Bias and Variance?:

Bias: ?

Variance:?

Bias and Variance Rule of Thumb

Compared to simpler (fewer parameters), complex models have what kind of Bias and Variance?:

Bias: ? Lower, because complex models can better model local variation in f(x).

Variance:?

Bias and Variance Rule of Thumb

Compared to simpler (fewer parameters), complex models have what kind of Bias and Variance?:

Bias: ? Lower, because complex models can better model local variation in f(x).

Variance:? Higher, because the parameters are generally more sensitive to few data values.

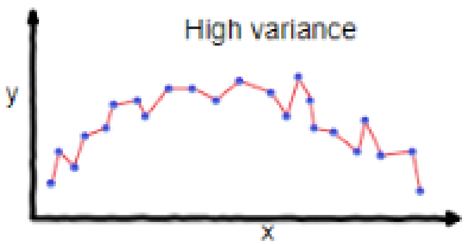
Bias and Variance Rule of Thumb

Compared to simpler (fewer parameters), complex models have what kind of Bias and Variance?:

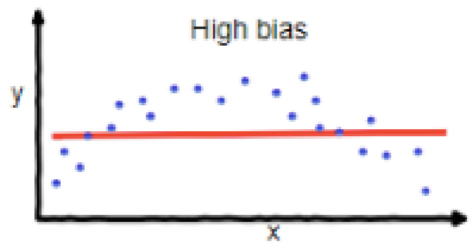
Bias: ? Lower, because complex models can better model local variation in f(x).

Variance:? Higher, because the parameters are generally more sensitive to few data values.

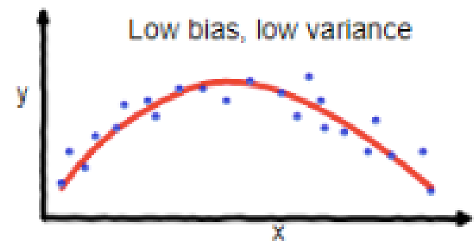
Complex models are prone to overfitting. They require/benefit from large training datasets in order to reduce variance. This includes most **Deep Networks**.



overfitting



underfitting



Good balance

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.

Comparison of Methods

Linear discriminant analysis Nearest centroid KNN	Neural networks Support vector machines
Simple method	Advanced methods
Based on distance calculation	Involve machine learning
Good for simple problems	Several adjustable parameters
Good for few training samples	Many training samples required (e.g.. 50-100)
Distribution of data assumed	Flexible methods

KNN Advantages

- Easy to program
- No optimization or training required
- Classification accuracy can be very good; can outperform more complex models