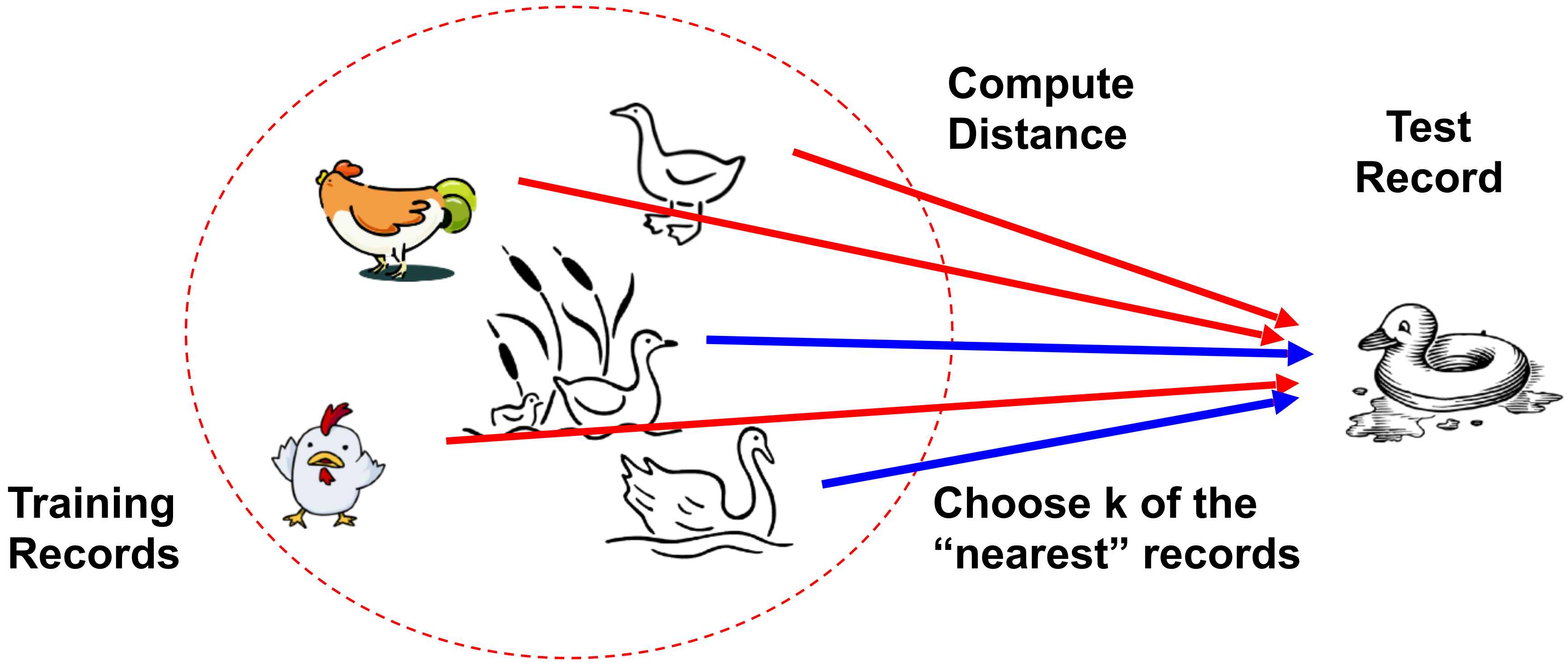
Classifiers: kNN

आई आई टी हैदराबाद
IIT Hyderabad

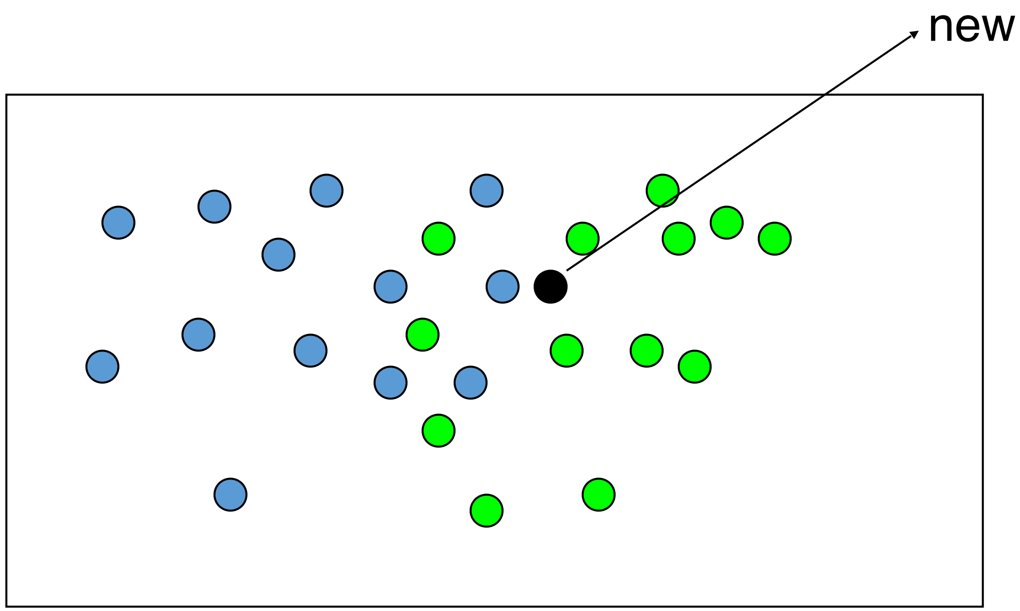
k-Nearest Neighbors

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



k-Nearest Neighbors

- Majority vote within the k nearest neighbors



K= 1: blue
K= 3: green

k-Nearest Neighbors

- An arbitrary instance is represented by $(a_1(x), a_2(x), a_3(x),.., a_n(x))$
  - $a_i(x)$ denotes features
- Euclidean distance between two instances
  - $d(x_i, x_j)=sqrt(sum\ for\ r=1\ to\ n\ (a_r(x_i) - a_r(x_j))^2)$
- L_p distance
  - p=2: Euclidean distance
  - p=1: Manhattan distance
  - $p = \infty$: Max distance
  - p= 0: Count non-zero distance
- In case of continuous-valued target function
  - Mean value of k nearest training examples

Other Distance Metrics

- Cosine Distance Metric $\rho(\vec{x}_1, \vec{x}_2) = \cos(\angle(\vec{x}_1, \vec{x}_2)) = \dfrac{\vec{x}_1 \cdot \vec{x}_2}{\|\vec{x}_1\|_2 \ \|\vec{x}_2\|_2}$

- Edit Distance

$x_1 =$ AAATCCCGTAA

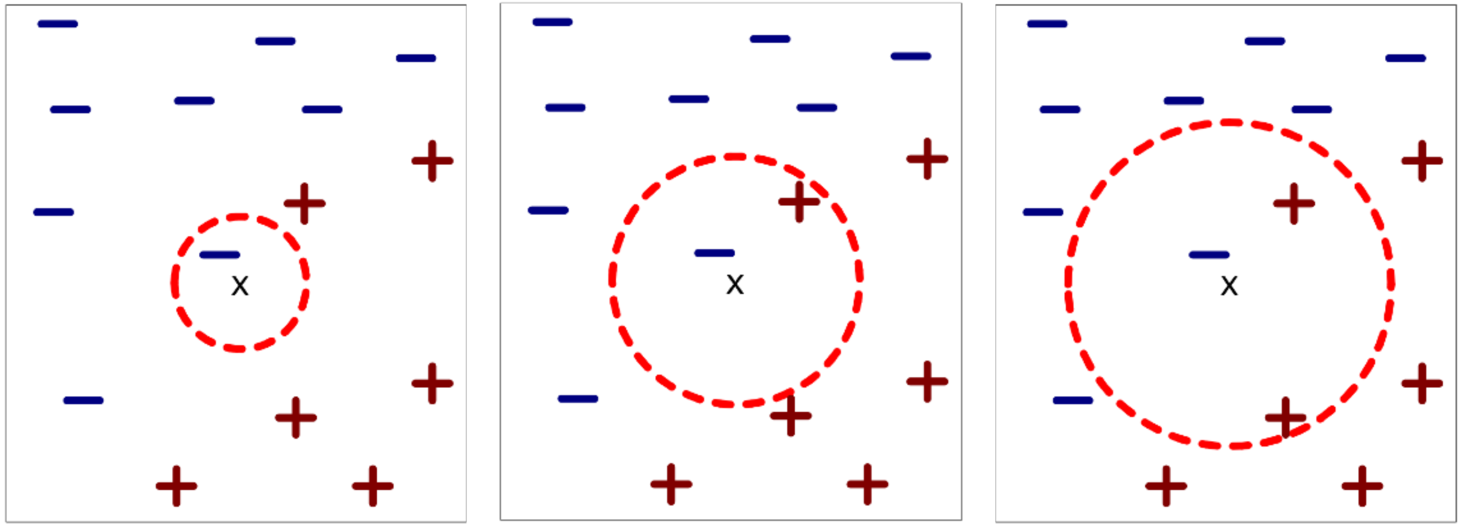
$x_2 =$ AATCGCGTAA

Minimum number of insertions, deletions and mutations needed

$$\rho(x_1, x_2) = 2$$

k-Nearest Neighbors

- Choosing k is important
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



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

How to determine k

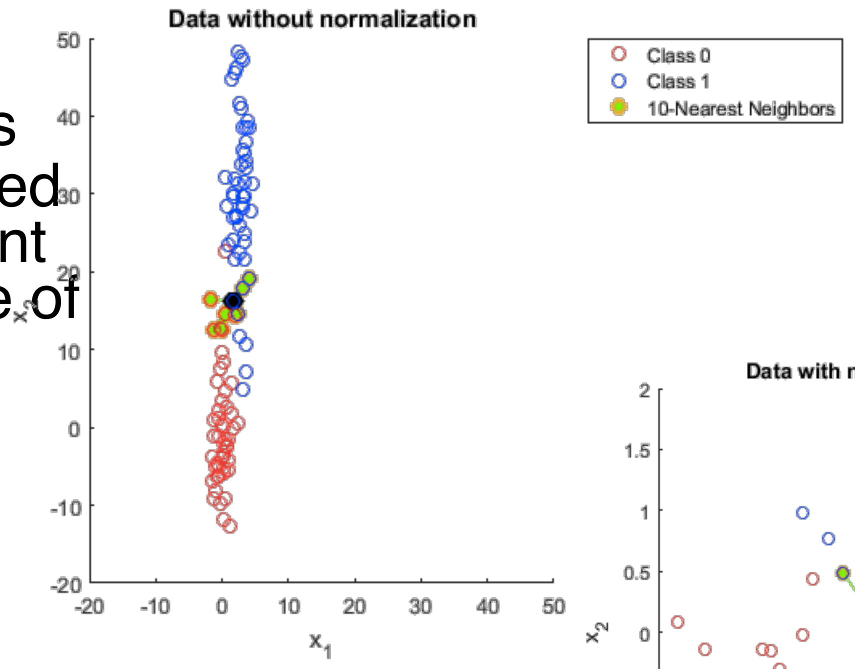
- Determined experimentally (think cross-validation!)
  - Start with k=1 and use a test set to validate the error rate of the classifier
  - Repeat with k=k+2
  - Choose the value of k for which the error rate is minimum
  - Note: k typically an odd number to avoid ties in binary classification

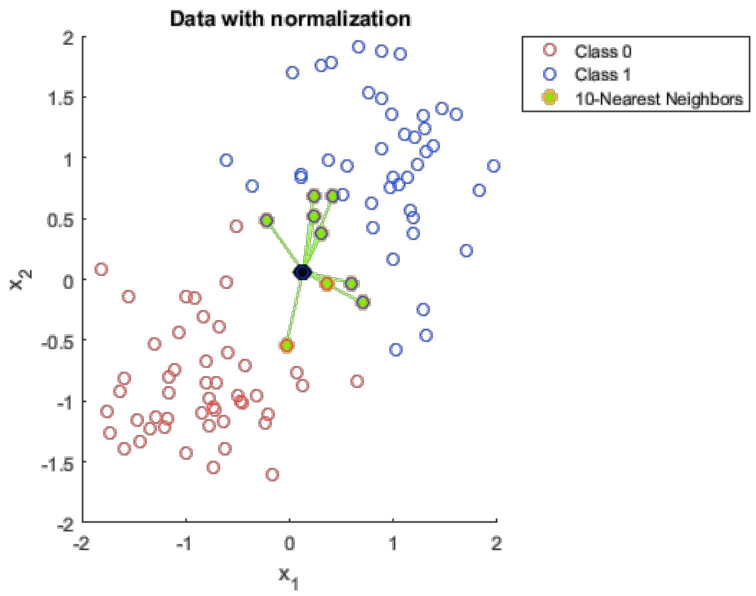
Pros and Cons

- Pros
  - Highly effective and simple method
  - Trains very fast ("Lazy" learner)
- Cons
  - Curse of dimensionality
    - In higher dimensions, all data points lie on the surface of the unit hypersphere!
  - Closeness in raw measurement space may not be good for the task
  - Storage: all training examples are saved in memory
    - A decision tree or linear classifier is much smaller
  - Slow at query time
    - Can be overcome and presorting and indexing training samples

Improvements

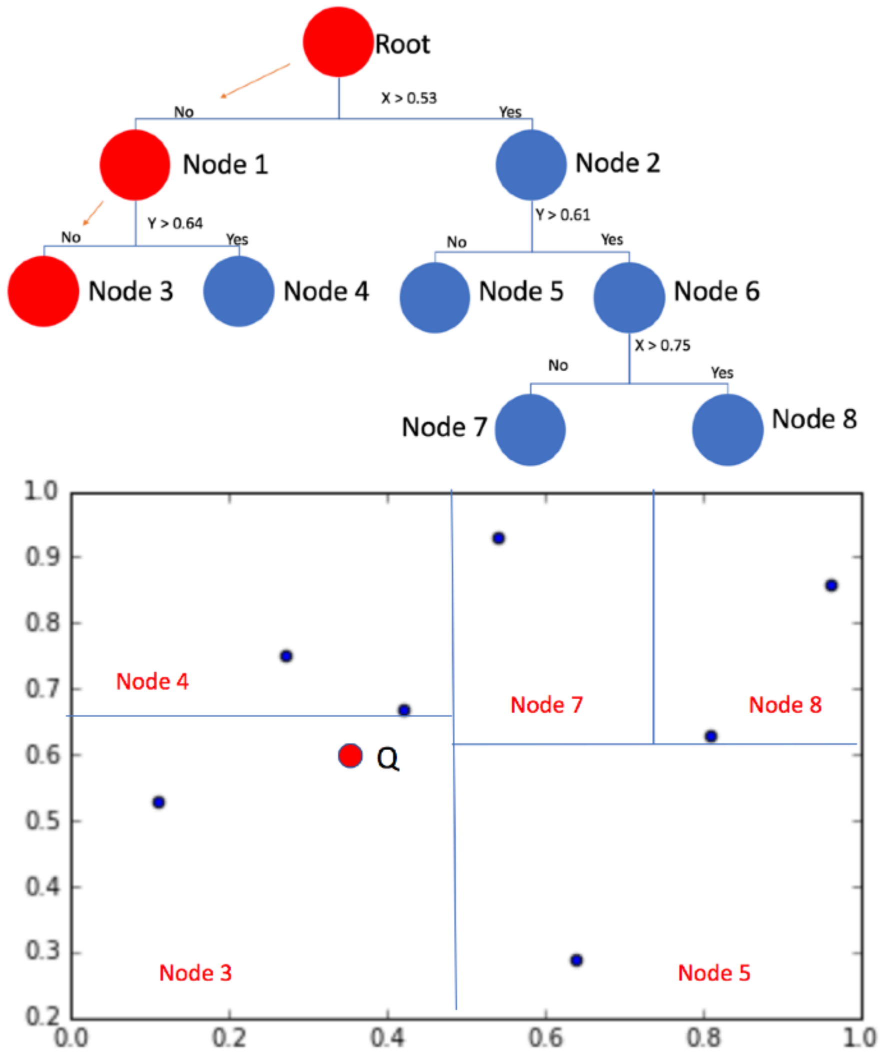
- Distance-Weighted Nearest Neighbors
  - Assign weights to the neighbors based on their ‘distance’ from the query point (E.g., weight ‘may’ be inverse square of the distances)
  - Can also learn this -> “**Metric Learning**”
- Scaling (**normalization**) attributes for fair computation of distances





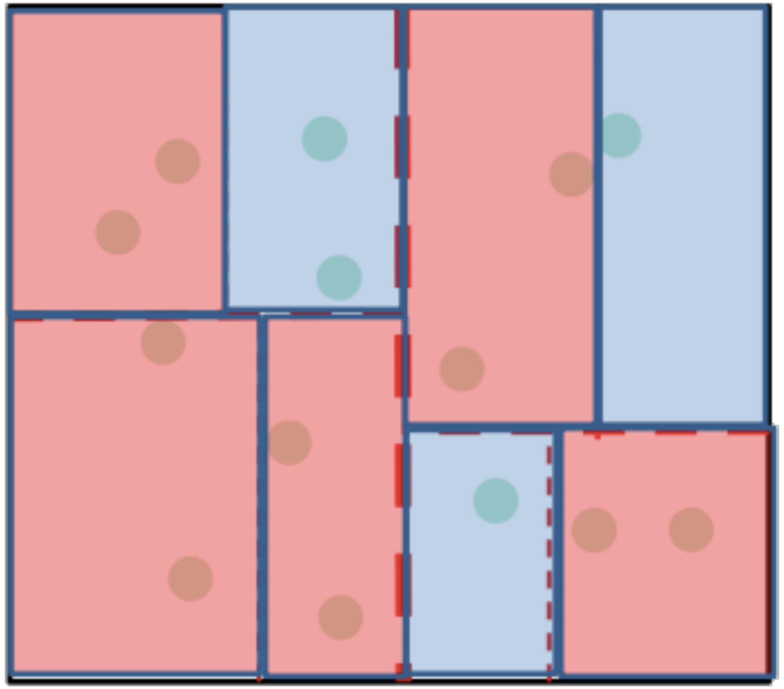
Improvements

- Finding “close” examples in a large training set quickly
  - E.g. Efficient memory indexing using kd-trees
    - In 1-dimension, can reduce complexity from O(n) to O(log n) – assuming data is sorted
  - Other methods
    - Locality-Sensitive Hashing, Clustering-based methods



Improvements

- Not storing all examples
  - We can label each cell instead and discard the training data



Classification : Evaluation metrics

Accuracy : $$\frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[y_i == \hat{y}_i]$$

		Actual Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

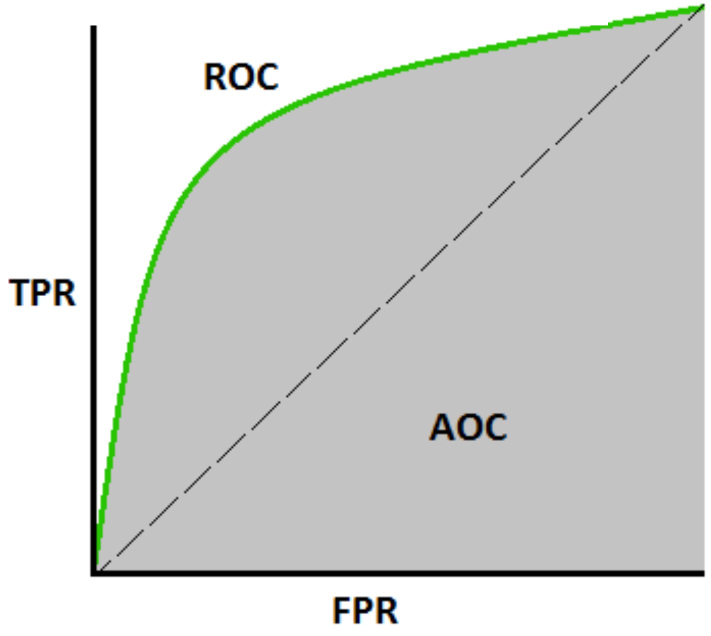
Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct
Sensitivity (Recall)	TP / (TP + FN)	The percentage of positive cases that were predicted as positive
Specificity	TN / (TN + FP)	The percentage of negative cases that were predicted as negative

Classification metric : ROC and AUC

- AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve.

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN} \qquad \text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{FPR} = 1 - \text{Specificity} = \frac{FP}{TN + FP}$$



- AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. Useful for class imbalanced data.

Readings

- Chapters 8, 9, EA Introduction to ML 2nd Edn
- Chapter 14 (Sec 14. 4) + Chapter 2 (Sec 2. 5), Bishop, PRML