



CMSC 422 Introduction to Machine Learning
Lecture 4 Geometry and Nearest Neighbors

Furong Huang / furongh@cs.umd.edu



UNIVERSITY OF
MARYLAND

What we know so far

Decision Trees

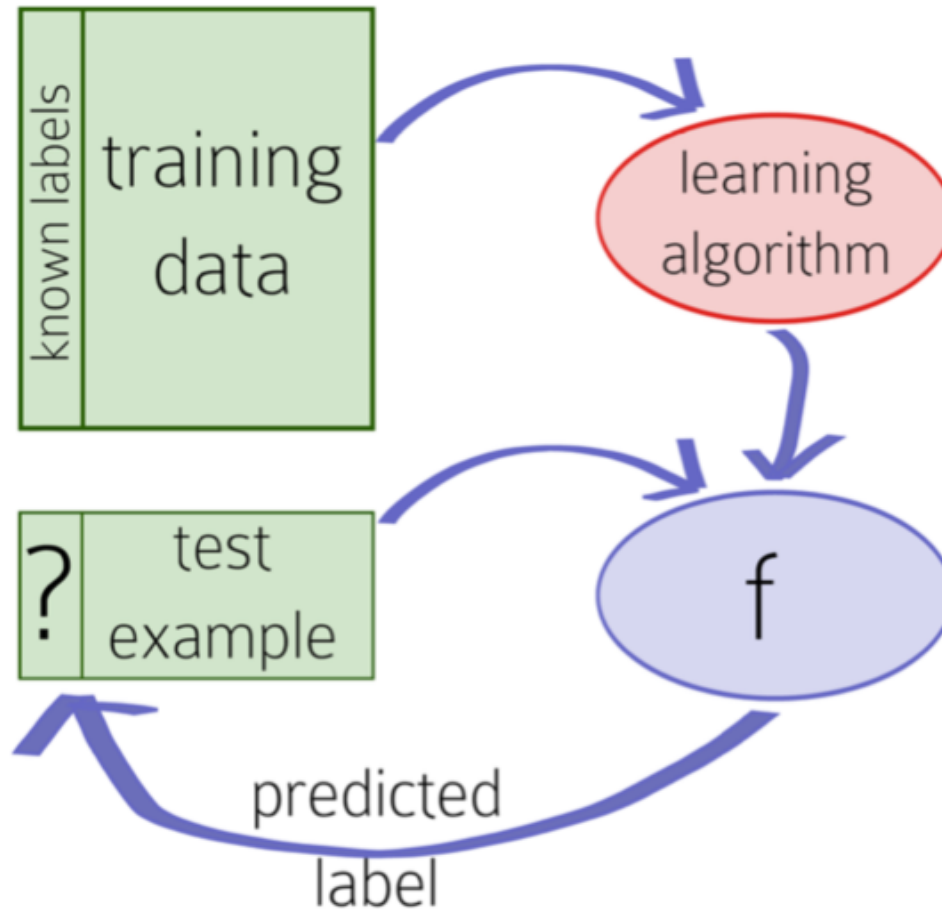
- What is a decision tree, and how to induce it from data

Fundamental Machine Learning Concepts

- Difference between memorization and generalization
- What inductive bias is, and what is its role in learning
- What underfitting and overfitting means
- How to take a task and cast it as a learning problem
- **Why you should never ever touch your test data!!**

Today's Topics

- Nearest Neighbors (NN) algorithms for classification
 - K-NN, Epsilon ball NN
- Fundamental Machine Learning Concepts
 - Decision boundary



Linear Algebra

- Provides compact representation of data
 - For a given example, all its features can be represented as a single **vector**
 - An entire dataset can be represented as a single **matrix**
- Provide ways of manipulating these objects
 - **Dot products, vector/matrix operations**, etc
- Provides formal ways of describing and discovering patterns in data
 - Examples are points in a **Vector Space**
 - We can use **Norms and Distances** to compare them
- Some are valid for feature data types
- Some can be made valid, with generalization ...

Mathematical view of vectors

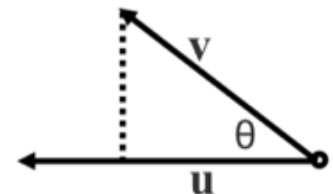
- Ordered set of numbers: (1,2,3,4)
- Example: (x,y,z) coordinates of a point in space.
- The 16384 pixels in a 128×128 image of a face
- List of choices in the tennis example
- Vectors usually indicated with bold lower case letters.
Scalars with lower case
- Usual mathematical operations with vectors:
 - Addition operation $\mathbf{u} + \mathbf{v}$, with:
 - ❖ Zero $\mathbf{0}$ $\mathbf{v} + \mathbf{0} = \mathbf{v}$
 - ❖ Inverse — $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
 - Scalar multiplication:
 - ❖ Distributive rule: $\alpha(\mathbf{u} + \mathbf{v}) = \alpha\mathbf{u} + \alpha\mathbf{v}$
 $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$

Dot Product

- The *dot product* or, more generally, *inner product* of two vectors is a scalar:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = x_1x_2 + y_1y_2 + z_1z_2 \quad (\text{in 3D})$$

- Useful for many purposes
 - Computing the Euclidean length of a vector:
 $\text{length}(\mathbf{v}) = \sqrt{\mathbf{v} \cdot \mathbf{v}}$
 - Normalizing a vector, making it unit-length
 - Computing the angle between two vectors: $\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}| |\mathbf{v}| \cos(\theta)$
 - Checking two vectors for orthogonality
 - Projecting one vector onto another
- Other ways of measuring length and distance are possible



Vector Norms

$$\mathbf{v} = (x_1, x_2, \dots, x_n)$$

- Two norm (Euclidean norm)

$$\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$$

If $\|\mathbf{v}\|_2 = 1$, $\mathbf{v}$ is a unit vector

- Infinity norm

$$\|\mathbf{v}\|_\infty = \max(|x_1|, |x_2|, \dots, |x_n|)$$

- One norm (Manhattan distance)

$$\|\mathbf{v}\|_1 = \sum_{i=1}^n |x_i|$$

Law of Large Numbers

- Suppose that $v_1, v_2, \dots, v_N$ are **independent** and **identically distributed** random variables
- The empirical sample average approaches the population average as the number of sample goes to infinity

$$\Pr\left(\lim_{N \rightarrow \infty} \frac{1}{N} \sum_n v_n = \mathbb{E}[v]\right) = 1$$

Nearest Neighbor

- Intuition—points close in a feature space are likely to belong to the same class
 - Choosing right features is very important
- Nearest Neighbors (NN) algorithms for Classification
 - K-NN, Epsilon ball NN
- Fundamental Machine Learning Concepts
 - Decision boundary

Intuition for Nearest Neighbor Classification

This “**rule of nearest neighbor**” has considerable elementary intuitive appeal and probably corresponds to practice in many situations. For example, it is possible that much medical diagnosis is influenced by the doctor’s **recollection** of the subsequent history of an earlier patient whose symptoms **resemble** in some way those of the current patient.

(Fix and Hodges, 1952)

Intuition for Nearest Neighbor Classification

- Simple idea
 - Store all training examples
 - Classify new examples based on label for K closest training examples
 - Training may just involve making structures to make computing closest examples cheaper

K Nearest Neighbors

Training Data

K: number of neighbors that classification is based on

Test instance with unknown class in $\{-1; +1\}$

Algorithm 3 KNN-PREDICT($\mathbf{D}, K, \hat{x}$)

```
1:  $S \leftarrow []$ 
2: for  $n = 1$  to  $N$  do
3:    $S \leftarrow S \oplus \langle d(x_n, \hat{x}), n \rangle$            // store distance to training example  $n$ 
4: end for
5:  $S \leftarrow \text{SORT}(S)$                            // put lowest-distance objects first
6:  $\hat{y} \leftarrow 0$ 
7: for  $k = 1$  to  $K$  do
8:    $\langle \text{dist}, n \rangle \leftarrow S_k$                    //  $n$  this is the  $k$ th closest data point
9:    $\hat{y} \leftarrow \hat{y} + y_n$                        // vote according to the label for the  $n$ th training point
10: end for
11: return  $\text{SIGN}(\hat{y})$                              // return  $+1$  if  $\hat{y} > 0$  and  $-1$  if  $\hat{y} < 0$ 
```

2 approaches to learning

Eager learning (eg decision trees)

- Learn/Train
 - Induce an **abstract model** from data
- Test/Predict/Classify
 - Apply learned model to new data

Lazy learning (eg nearest neighbors)

- Learn
 - **Just store data** in memory
- Test/Predict/Classify
 - Compare new data to stored data
- **Properties**
 - Retains all information seen in training
 - Complex hypothesis space
 - Classification can be very slow

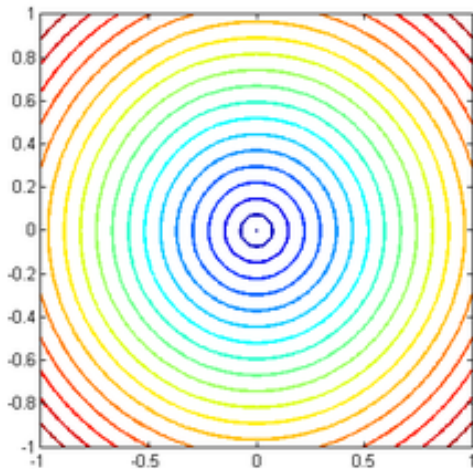
Components of a k-NN Classifier

- Distance metric
 - How do we measure distance between instances?
 - Determines the layout of the example space
- The k hyperparameter
 - How large a neighborhood should we consider?
 - Determines the complexity of the hypothesis space

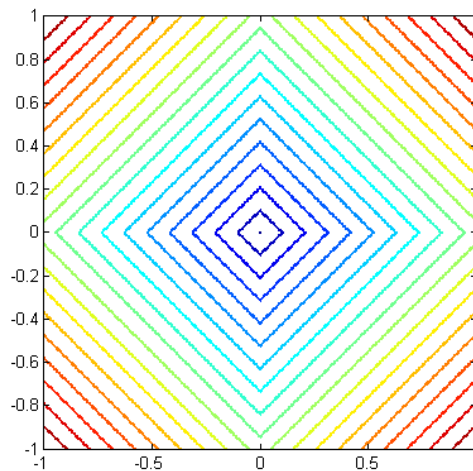
Distance metrics

- We can use any distance function to select nearest neighbors.
- Different distances yield different neighborhoods

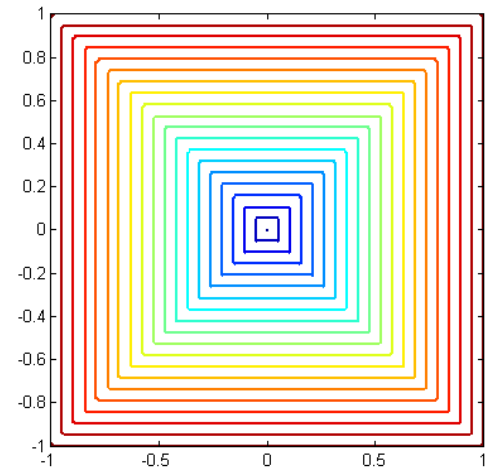
L2 distance
(= Euclidean distance)



L1 distance

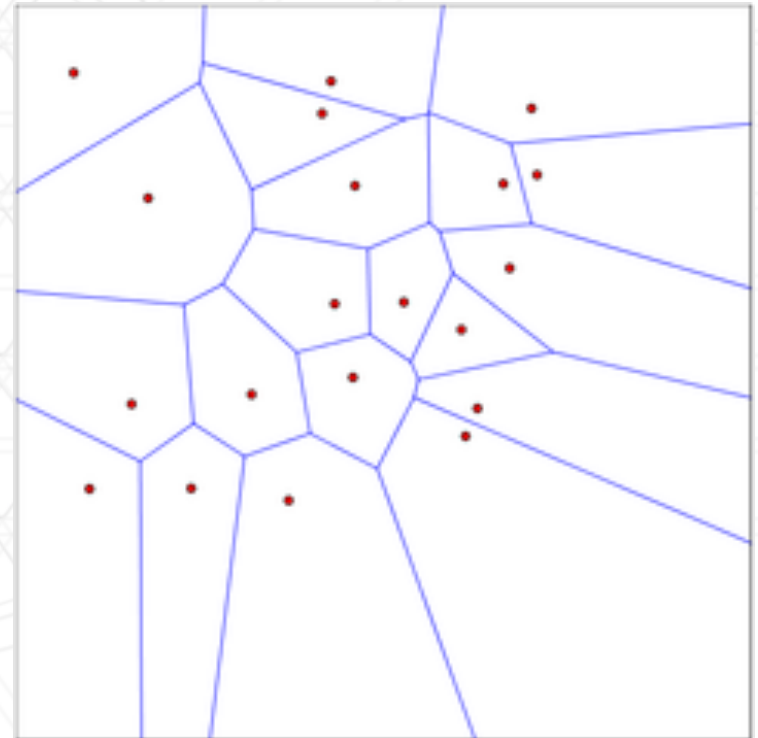


Max norm



K=1 and Voronoi Diagrams

- Imagine we are given a bunch of training examples
- Find regions in the feature space which are closest to every training example
- Algorithm—if our test point is in the region corresponding to a given input point—return its label

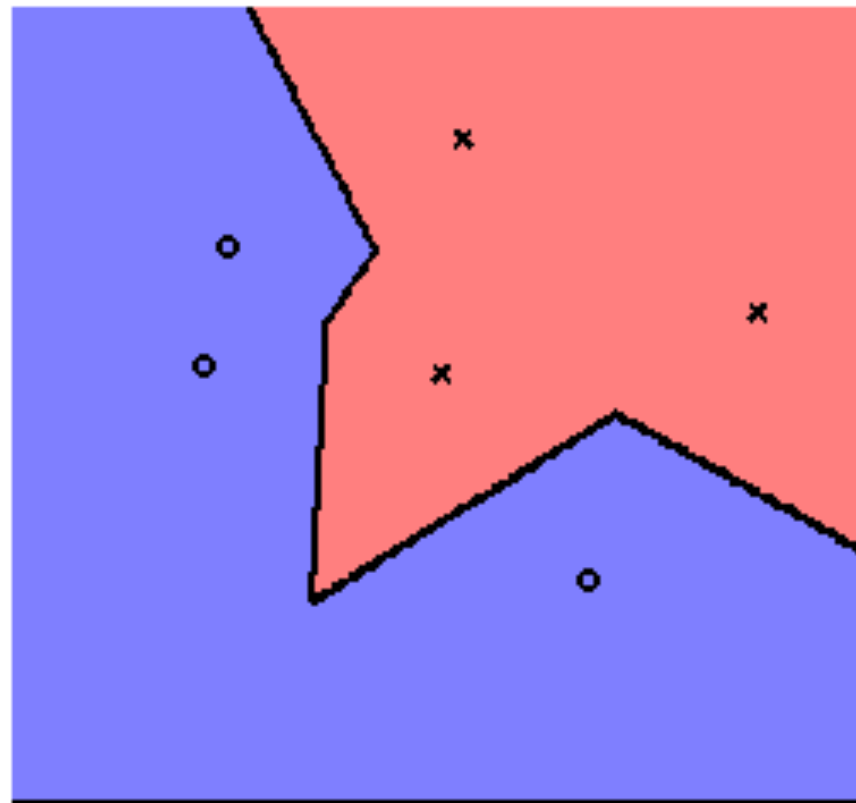


Decision Boundary of a Classifier

- It is simply the line that separates positive and negative regions in the feature space
- Why is it useful?
 - it helps us visualize how examples will be classified for the entire feature space
 - it helps us visualize the complexity of the learned model

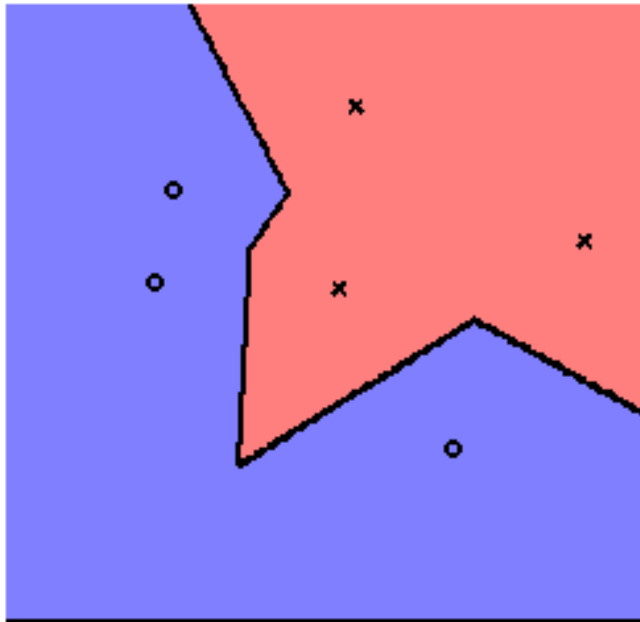
Decision Boundaries for 1-NN

knn (K=1): L2 Distance

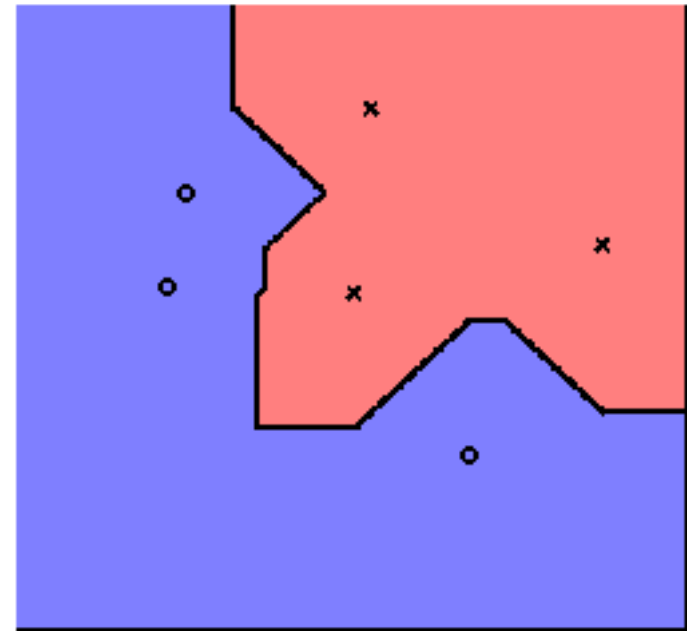


Decision Boundaries change with the distance function

knn (K=1): L2 Distance

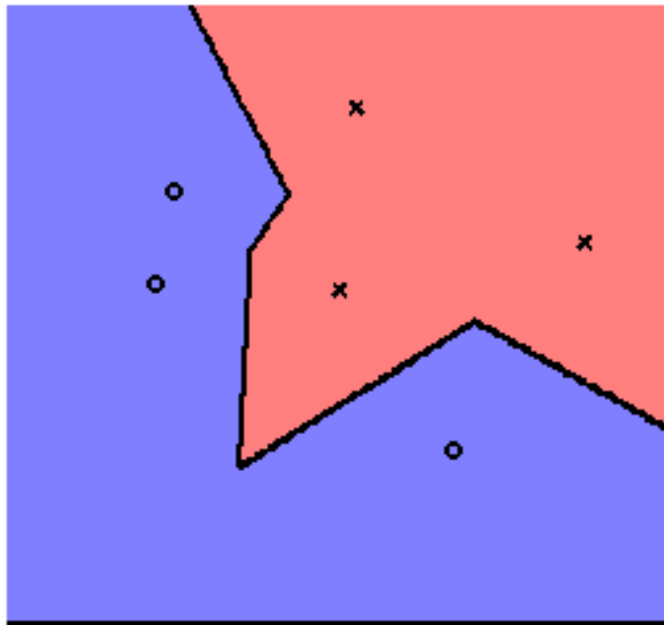


knn (K=1): L1 Distance

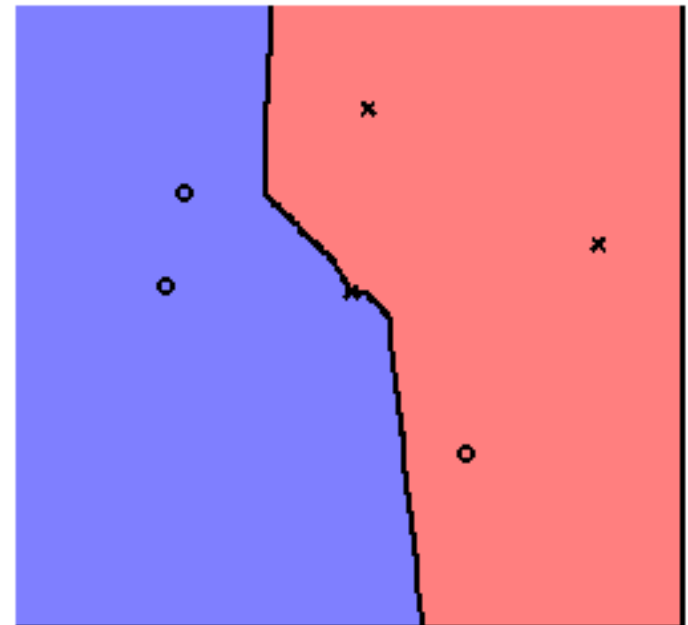


Decision Boundaries change with K

knn (K=1): 12 Distance



knn (K=3): 12 Distance



The k hyperparameter

- Tunes the complexity of the hypothesis space
 - If $k = 1$, every training example has its own neighborhood
 - If $k = N$, the entire feature space is one neighborhood!
- Higher k yields smoother decision boundaries
- How would you set k in practice?

What is the inductive bias of k-NN?

- Nearby instances should have the same label
- All features are equally important
- Complexity is tuned by the k parameter

Variations on k-NN:

Weighted voting

- Weighted voting
 - Default: all neighbors have equal weight
 - Extension: weight neighbors by (inverse) distance

Variations on k-NN:

Epsilon Ball Nearest Neighbors

- Same general principle as K-NN, but change the method for selecting which training examples vote
- Instead of using K nearest neighbors, use all examples x such that

$$distance(\hat{x}, x) \leq \varepsilon$$

Exercise: How would you modify KNN-Predict to perform Epsilon Ball NN?

Algorithm 3 KNN-PREDICT($\mathbf{D}, K, \hat{x}$)

```
1:  $S \leftarrow []$ 
2: for  $n = 1$  to  $N$  do
3:    $S \leftarrow S \oplus \langle d(x_n, \hat{x}), n \rangle$            // store distance to training example  $n$ 
4: end for
5:  $S \leftarrow \text{SORT}(S)$                            // put lowest-distance objects first
6:  $\hat{y} \leftarrow 0$ 
7: for  $k = 1$  to  $K$  do
8:    $\langle \text{dist}, n \rangle \leftarrow S_k$                  //  $n$  this is the  $k$ th closest data point
9:    $\hat{y} \leftarrow \hat{y} + y_n$                        // vote according to the label for the  $n$ th training point
10: end for
11: return  $\text{SIGN}(\hat{y})$                              // return  $+1$  if  $\hat{y} > 0$  and  $-1$  if  $\hat{y} < 0$ 
```



Furong Huang

3251 A.V. Williams, College Park, MD 20740

301.405.8010 / furongh@cs.umd.edu