

Classification with Nearest Neighbors

CMSC 422

SOHEIL FEIZI

sfeizi@cs.umd.edu

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
- **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

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 most similar training examples

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

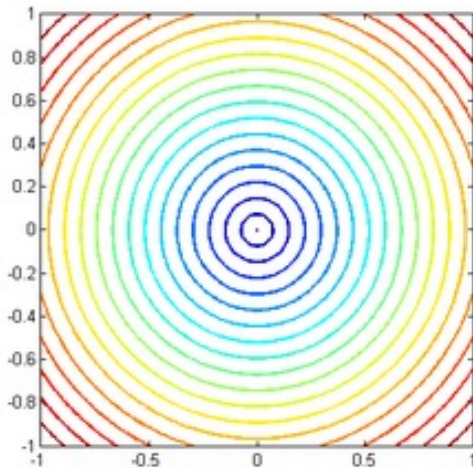
Components of a k-NN Classifier

- 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

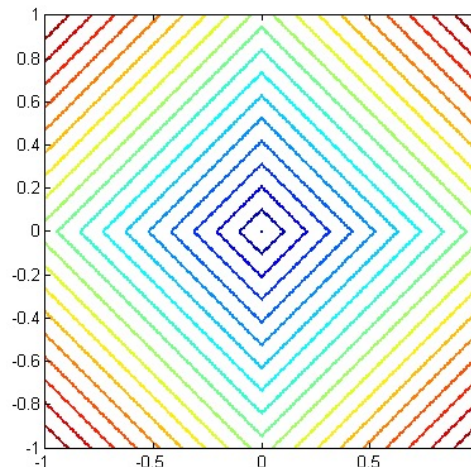
Distance metrics

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

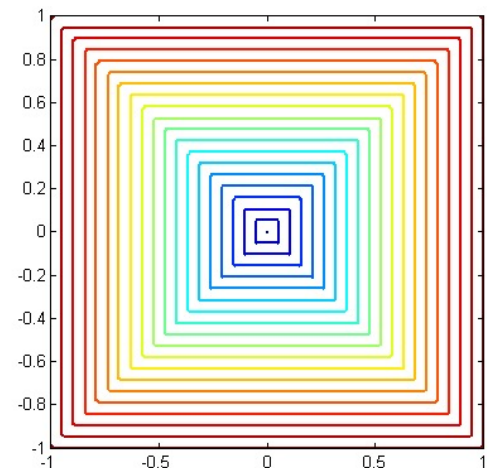
L2 distance
(= Euclidean distance)



L1 distance



Max norm

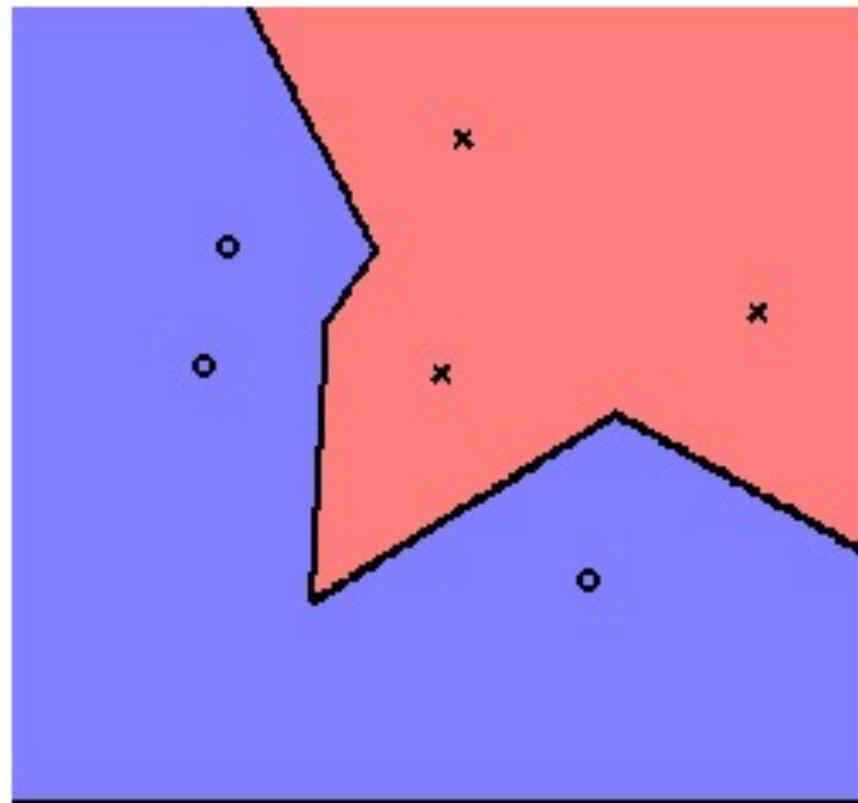


Decision Boundary of a Classifier

- It is 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

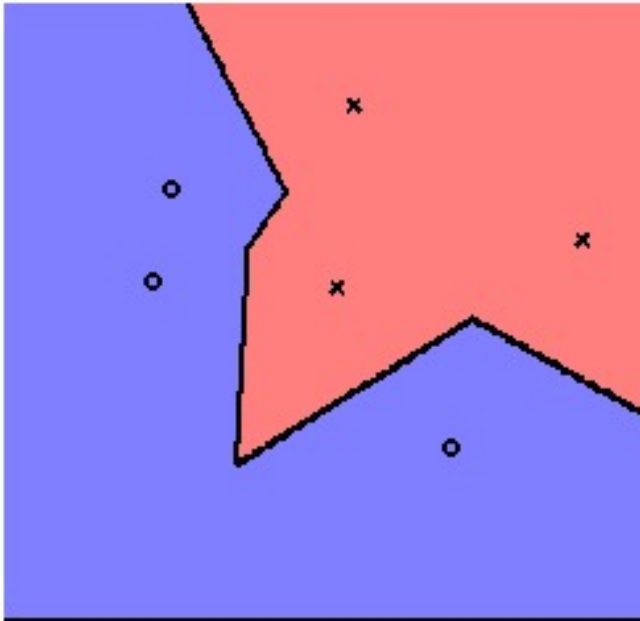
Decision Boundaries for 1-NN

knn (K=1): 12 Distance

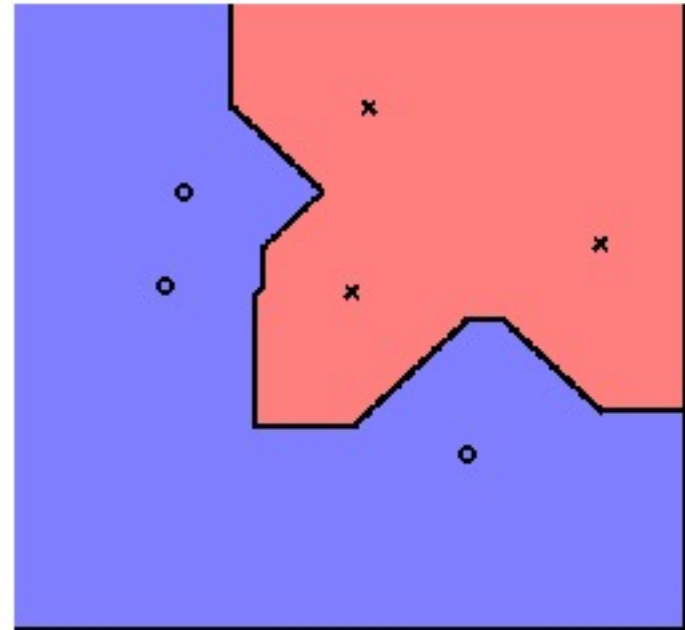


Decision Boundaries change with the distance function

knn (K=1): L2 Distance

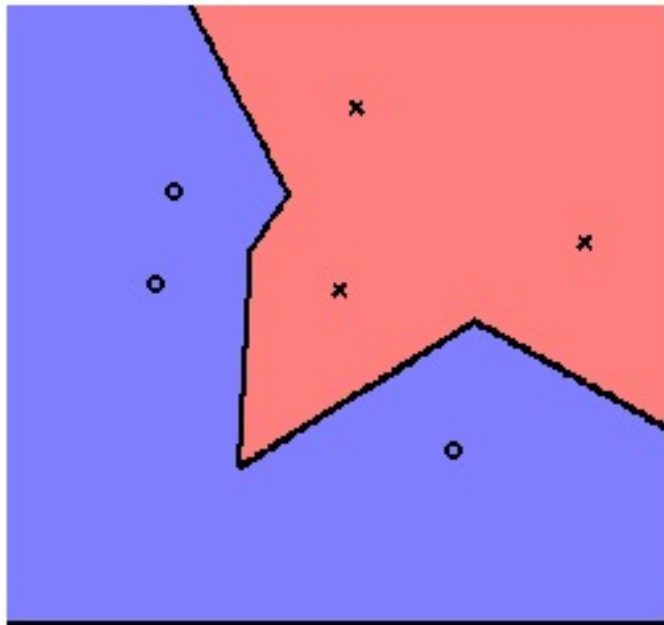


knn (K=1): L1 Distance

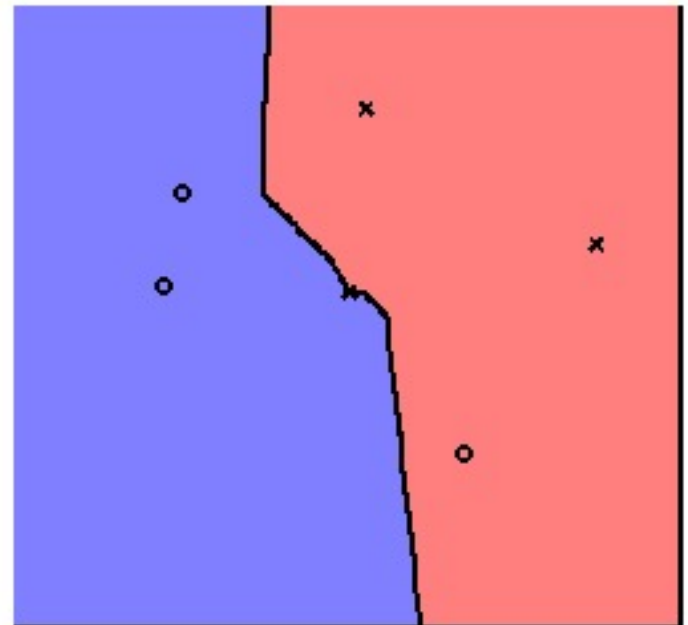


Decision Boundaries change with K

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

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

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

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

```
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  $\hat{y} < 0$ 
```

Recap

- Nearest Neighbors (NN) algorithms for classification
 - K-NN, Epsilon ball NN
 - Take a geometric view of learning
- Fundamental Machine Learning Concepts
 - Decision boundary
 - Visualizes predictions over entire feature space
 - Characterizes complexity of learned model
 - Indicates overfitting/underfitting