Geometric View of Machine Learning Nearest Neighbor Classification

Slides adapted from Prof. Carpuat

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!!

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:
    - Identity $\mathbf{0}$ $\quad \mathbf{v} + \mathbf{0} = \mathbf{v}$
    - Inverse **-** $\quad \mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
  - Scalar multiplication:
    - Distributive rule: $\quad \alpha(\mathbf{u} + \mathbf{v}) = \alpha(\mathbf{u}) + \alpha(\mathbf{v})$

      $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$

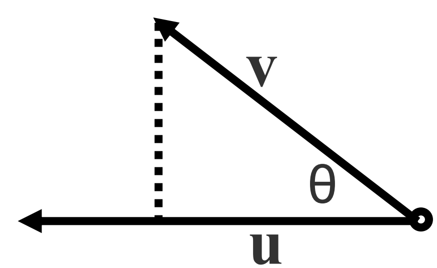
Dot Product

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

  $\mathbf{v}_1 \bullet \mathbf{v}_2 = x_1 x_2 + y_1 y_2 + z_1 z_2$ (in 3D)

- Useful for many purposes
  - Computing the Euclidean length of a vector: $length(\mathbf{v}) = \mathrm{sqrt}(\mathbf{v} \bullet \mathbf{v})$
  - *Normalizing* a vector, making it unit-length
  - Computing the angle between two vectors:

    $\mathbf{u} \bullet \mathbf{v} = |\mathbf{u}|\,|\mathbf{v}| \cos(\theta)$
  - Checking two vectors for orthogonality
  - *Projecting* one vector onto another
- Other ways of measuring length and distance are possible



Vector norms

$$v = (x_1, x_2, \ldots \quad {}_{n}$$

Two norm (Euclidean norm)

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

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

Infinity norm

$$\|v\|_\infty = \max(|x|_1, |x|_2, \ldots \quad {}_{n}$$

One norm ("Manhattan distance")

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

For a 2 dimensional vector, write down the set of vectors with two, one and infinity norm equal to unity

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

- 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 Nei… as… n

K: number of neighbors that classification is based on

Training Data

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

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

$S \leftarrow [\,]$
for $n = 1$ **to** N **do**
  $S \leftarrow S \oplus \langle d(x_n, \hat{x}), n \rangle$ // store distance to training example n
end for
$S \leftarrow \textsc{sort}(S)$ // put lowest-distance objects first
$\hat{y} \leftarrow 0$
for $k = 1$ **to** K **do**
  $\langle dist, n \rangle \leftarrow S_k$ // n this is the kth closest data point
  $\hat{y} \leftarrow \hat{y} + y_n$ // vote according to the label for the nth training point
end for
return $\textsc{sign}(\hat{y})$ // 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

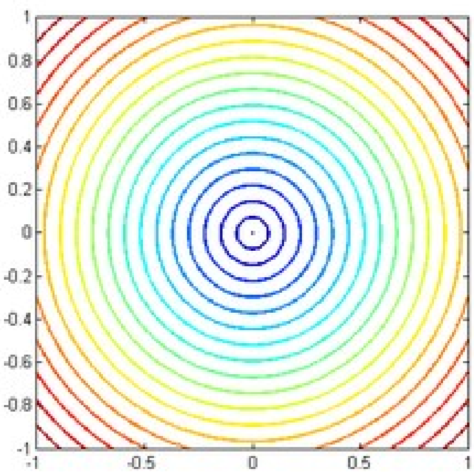
Components of a k-NN Classifier

- 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

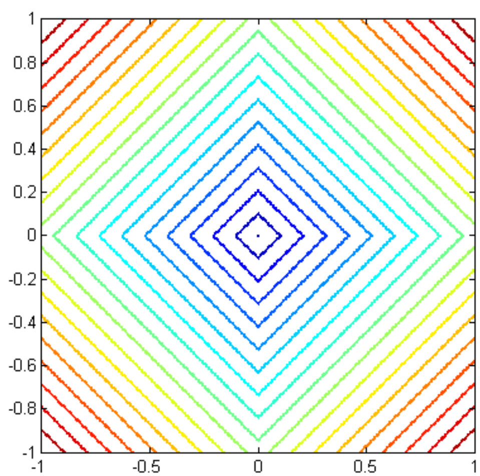
Distance metrics

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

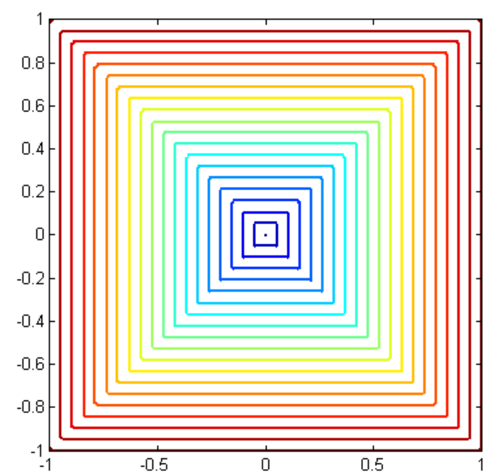
L2 distance
(= Euclidean distance)



L1 distance

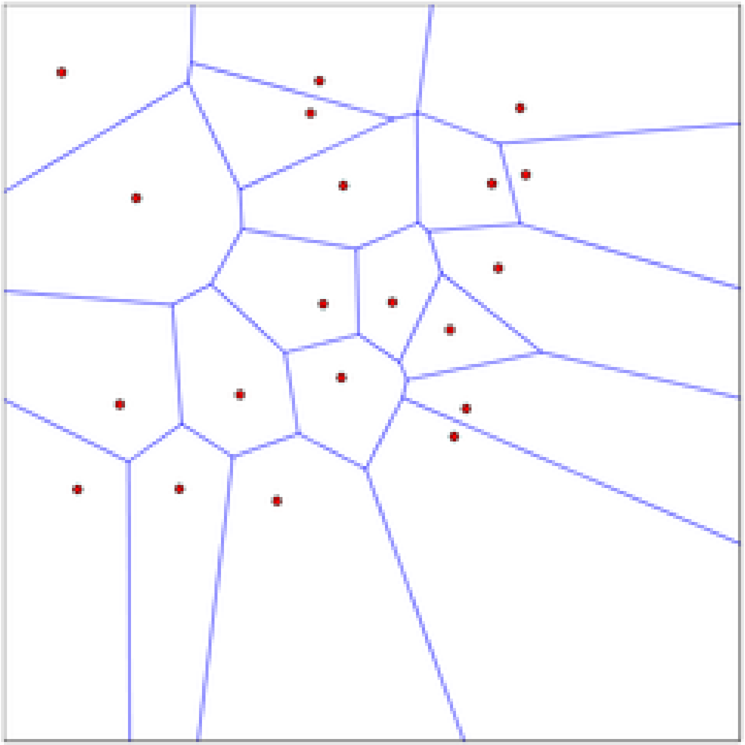


Max norm



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

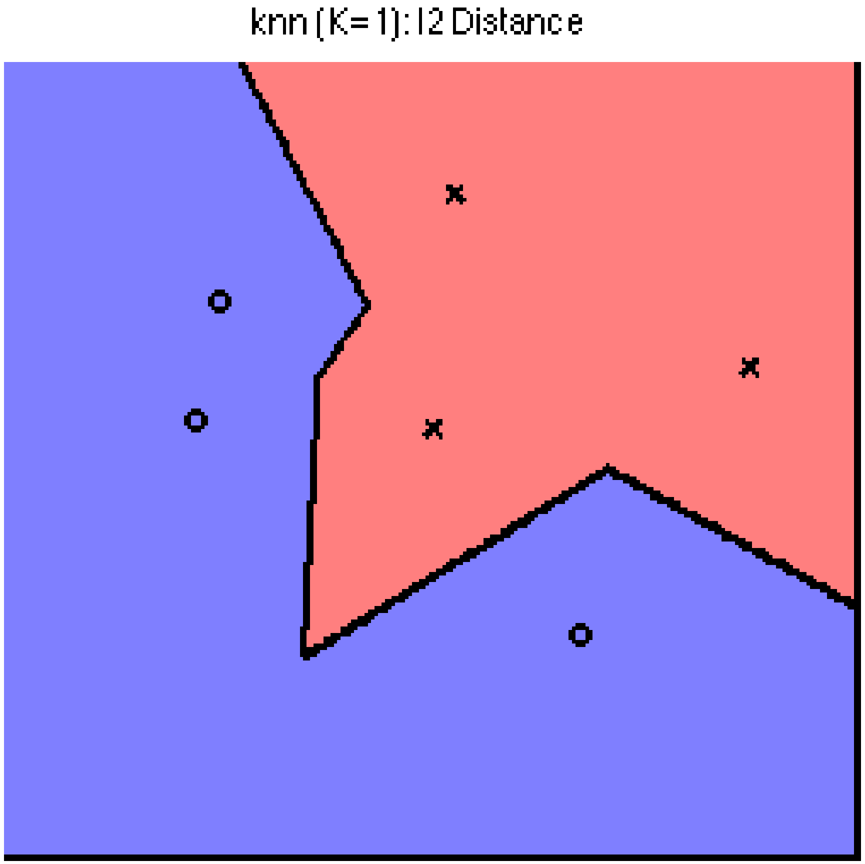


Decision Boundary of a Classifier

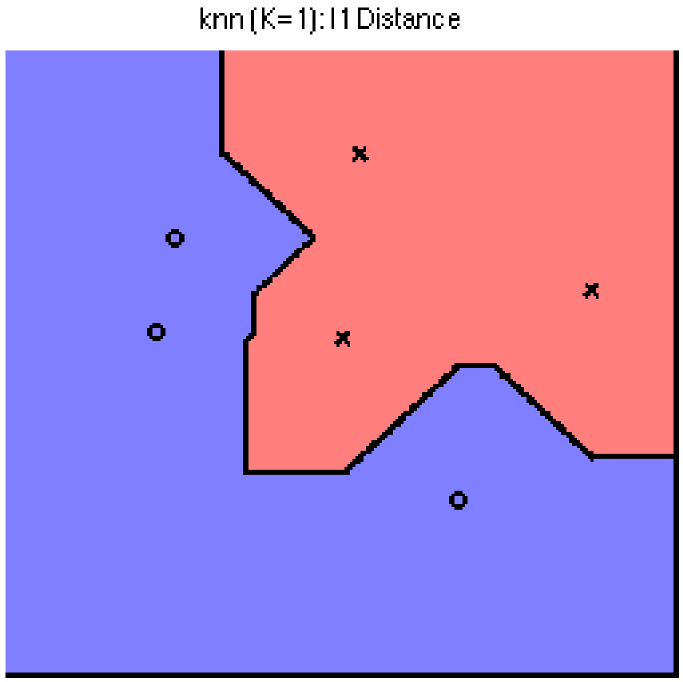
- It is simply the line that separates positive and negative regions in the feature space

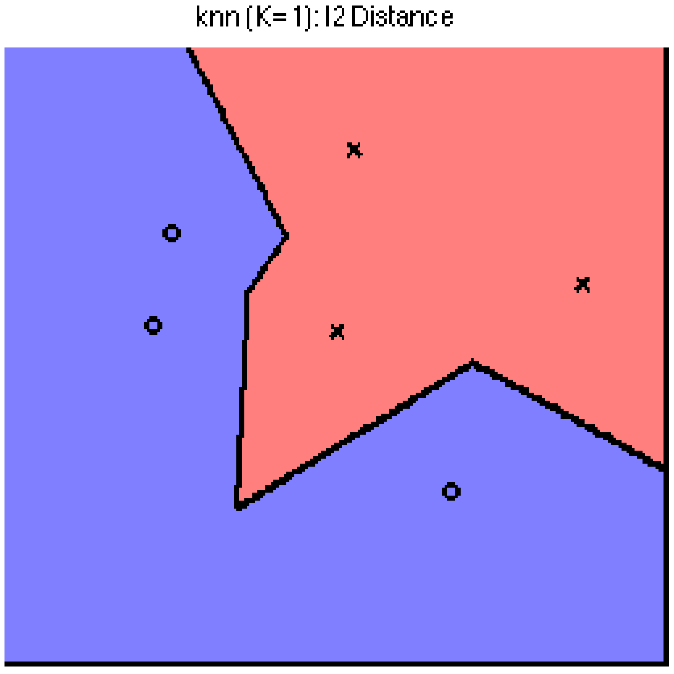
- Why is it useful?
  - it helps us visualize how examples will be classified for the entire feature space
  - it helps us visualize the complexity of the learned model

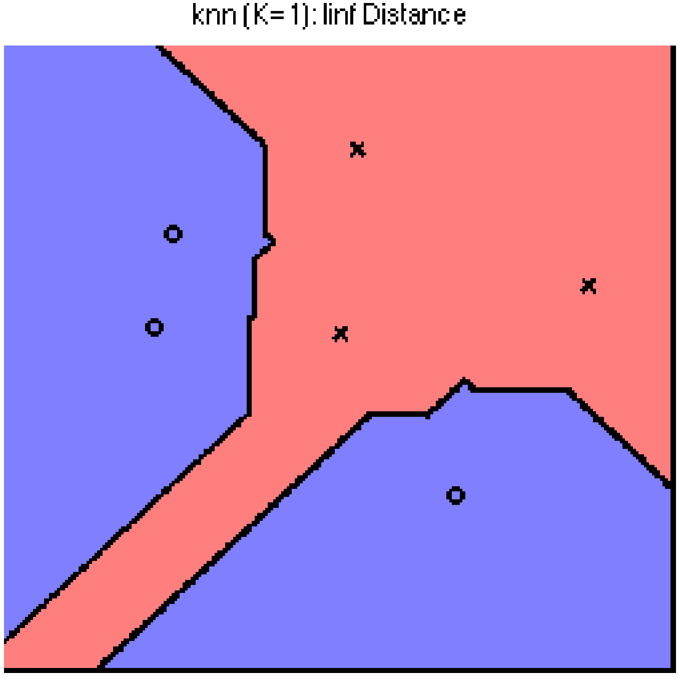
Decision Boundaries for 1-NN



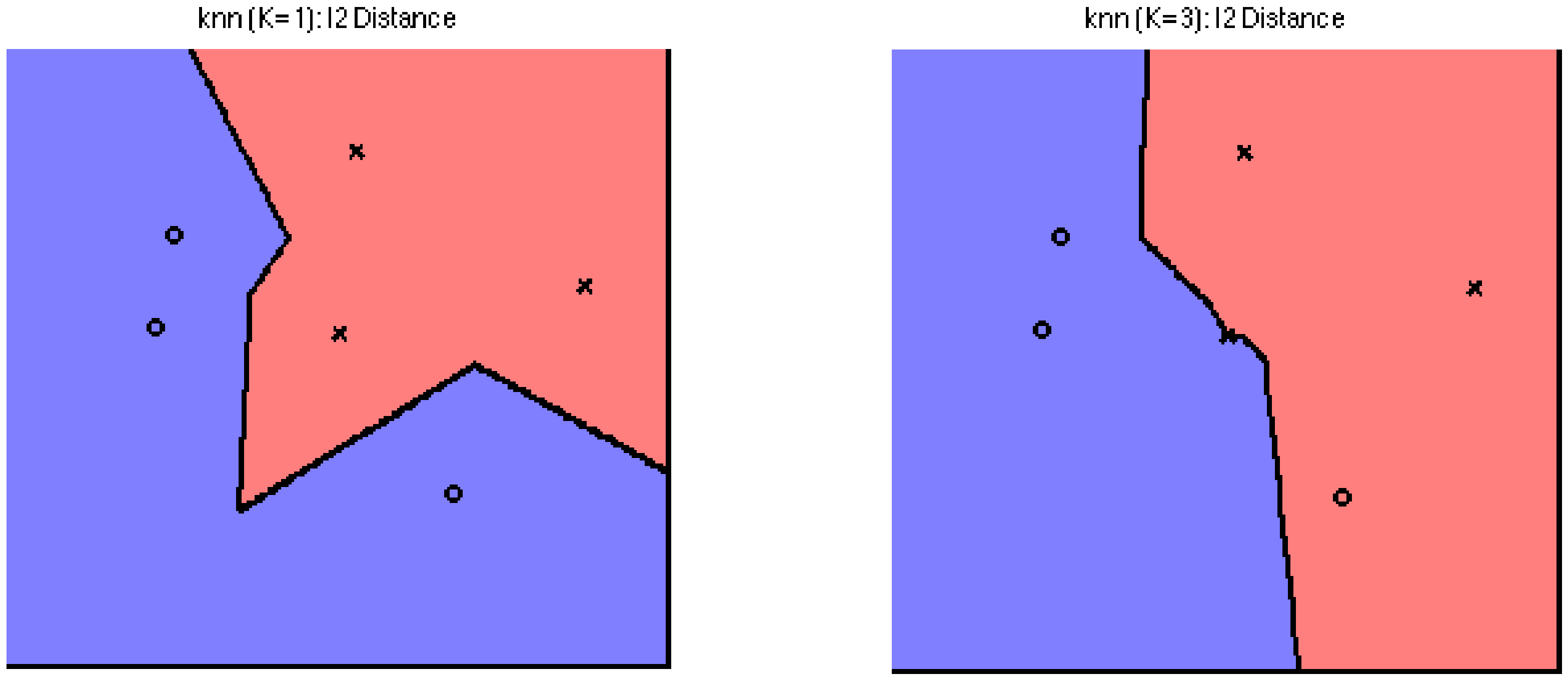
Decision Boundaries change with the distance function







Decision Boundaries change with K



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?