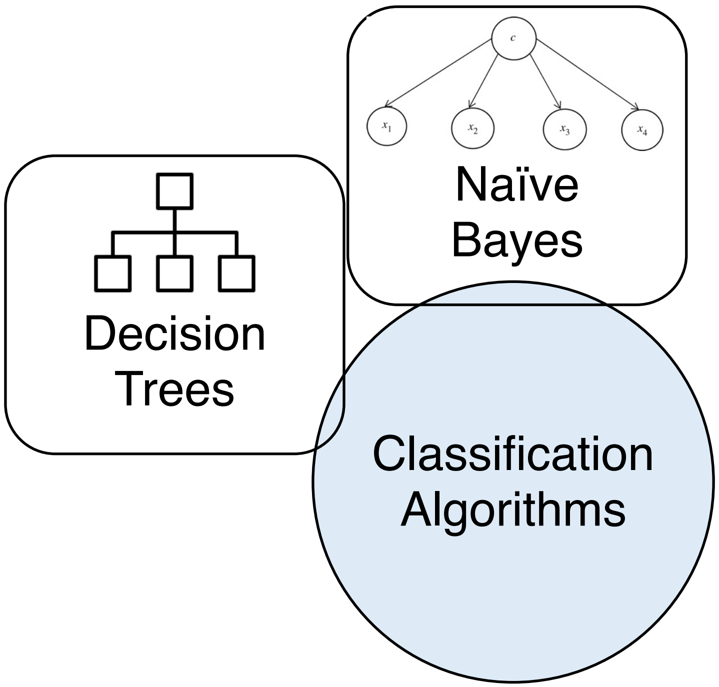
Machine Learning

Chapter 5: Nearest Neighbor

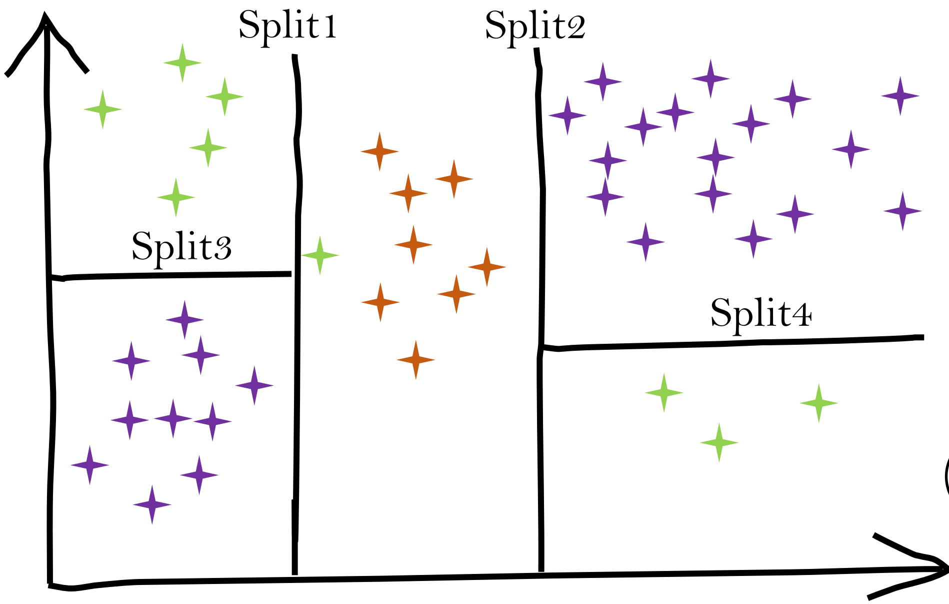
Fall 2022

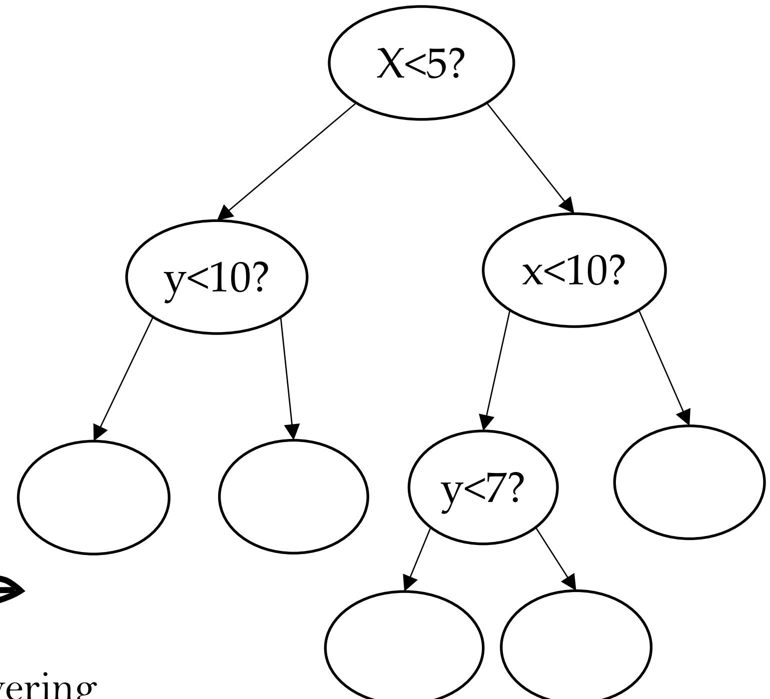
Instructor: Xiaodong Gu

The family of classification



Review: Decision Trees

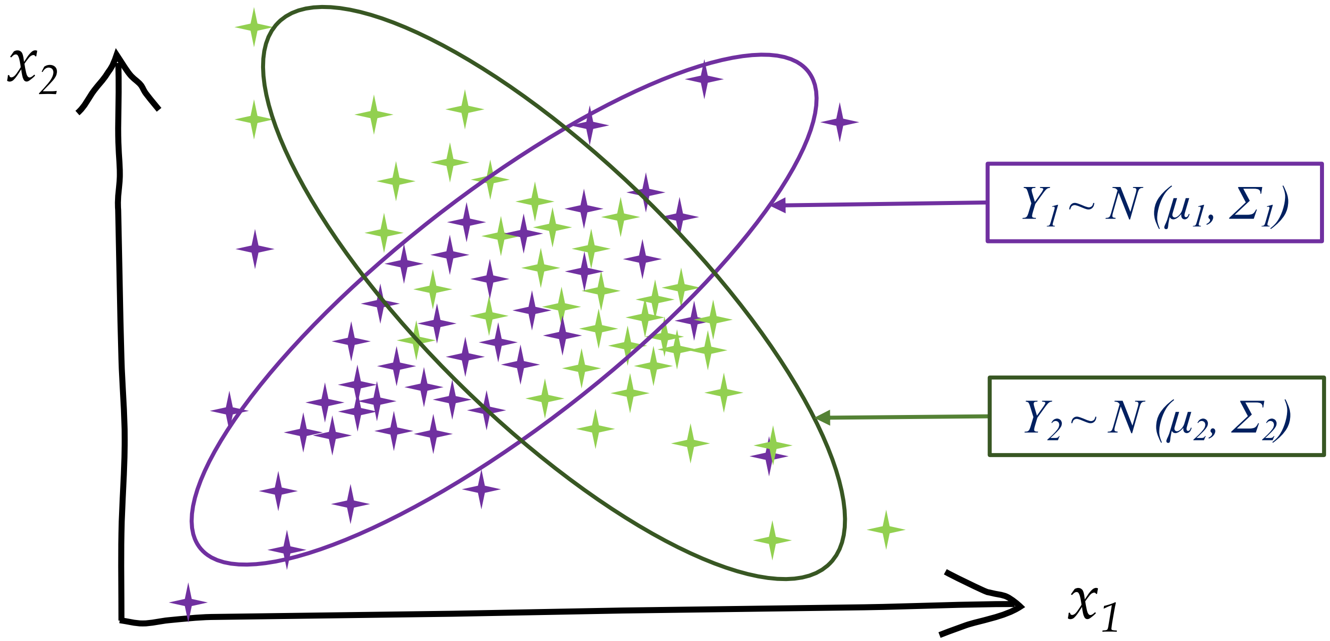




Combines a set of different linear models, each covering a region of the input space disjoint from others.

Review: Bayes

Modeling the generation of data using probabilities?



have clear patterns of probabilistic distributions and dependences

Do we really need a model ?

Data

Model

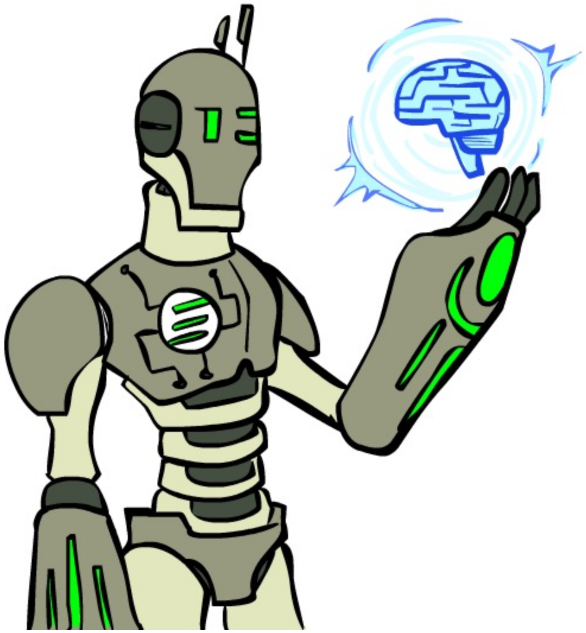
?

What if we directly use the data ?

Today

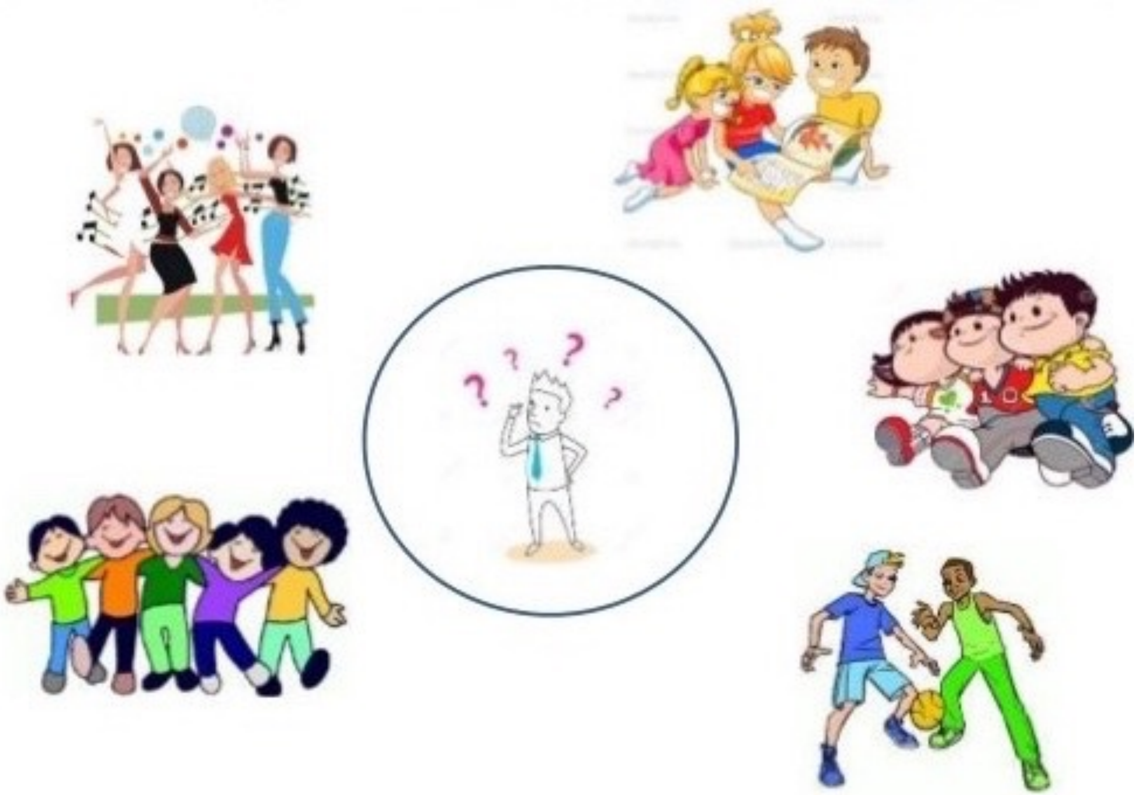
Let's learn a super easy (naïve) method for classification.

- K-Nearest Neighbor



A Simple Analogy…

- Tell me about your friends (who your neighbors are). Then I will tell you who you are.



K-Nearest Neighbors

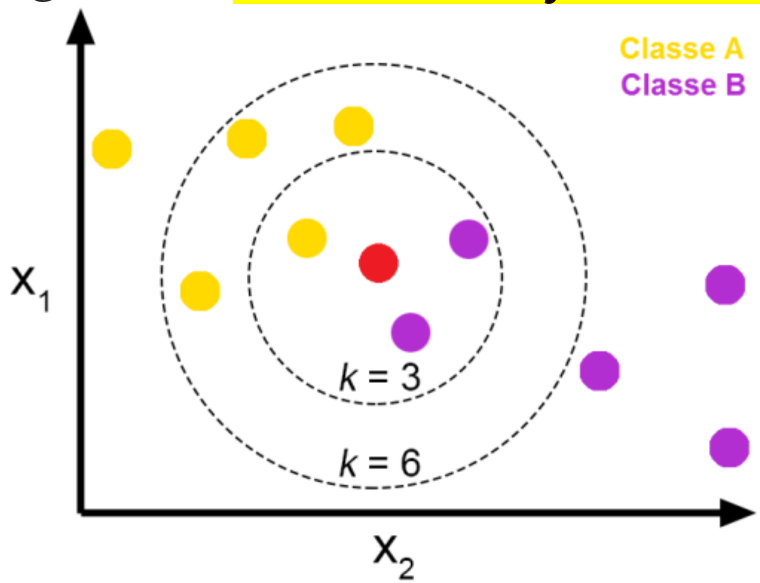
Instance-based learning, also called lazy learning.

- simply stores the training instances without learning a model.
- whenever we have new data to classify, we find its K-nearest neighbors from the training data.

K-Nearest Neighbor (KNN) is a simple algorithm that stores all the available cases and classifies the new data or case based on a similarity measure.

K-NN Classification

- Classified by "MAJORITY VOTES" from neighbor classes.
- An object is classified to the most common class amongst its k nearest neighbors **measured by a "distance" function**



How to determine whether an object falls into the k-nearest neighbors?

Distance Measures

- **Euclidean Distance**

欧几里得距离

$$D(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

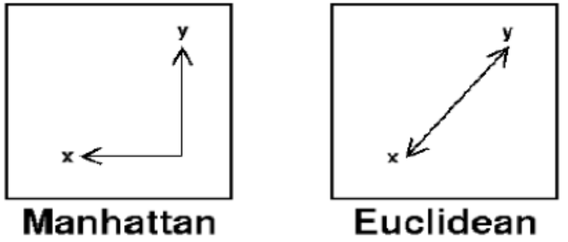
where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ represent the n attribute values of two records.

doesn't work well in high dimensions and for categorical variables because it ignores the similarity between attributes.

Distance Measures

- **Manhattan Distance** (a.k.a. city block distance)

  $D(x, y) = \sum_{i=1}^{n} |x_i - y_i|$



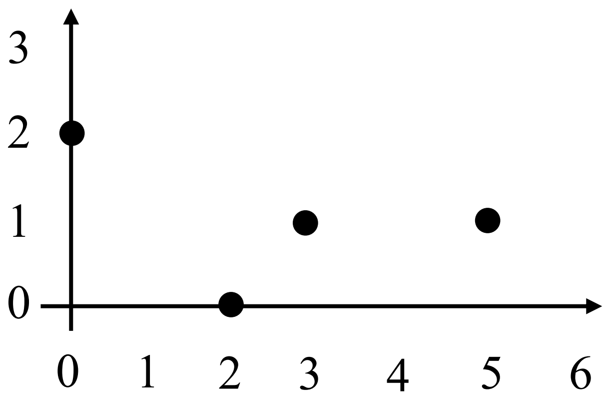
$|x_1 - x_2| + |y_1 - y_2|$

- **Minkowski Distance**

$$D(x, y) = \left(\sum_{u=1}^{n} |x_u - y_u|^p \right)^{\frac{1}{p}}$$

Distances

Example



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

Euclidean Distance Matrix

	p1	**p2**	**p3**	**p4**
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Normalization

- Standardize the range of independent variables (features of data)

Z-score normalization: rescale the data so that the mean is zero and the standard deviation from the mean (standard scores) is one.

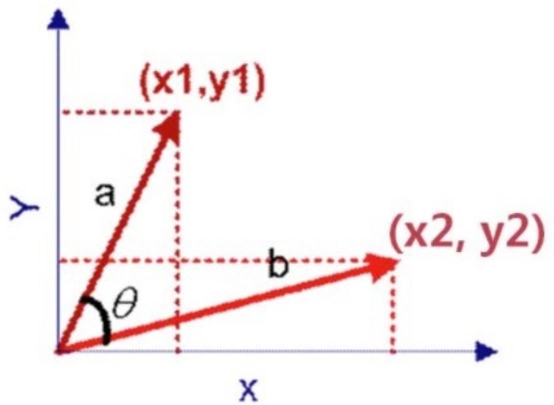
$$x_{norm} = \frac{x - \mu}{\sigma}$$

min-max normalization: scale the data to a fixed range between 0 and 1.

$$x_{morm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Similarity vs. Distance

- **Similarity:** numerical measure of how alike two data objects are.
  - Is higher when objects are more alike.
  - Often falls in the range [0, 1].
- **Cosine Similarity**



$$\cos(\theta) = \frac{a \bullet b}{||a|| \times ||b||}$$

$$= \frac{(x_1, y_1) \bullet (x_2, y_2)}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

$$= \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

$$\cos(d_1, d_2) = \begin{cases} 1: \text{exactly the same} \\ 0: \text{orthogonal} \\ -1: \text{exactly opposite} \end{cases}$$

Quiz

$\cos(d_1, d_2) = (d_1 \cdot d_2)/||d_1||\ ||d_2||$

$d_1 = [3\ 2\ 0\ 5\ 0\ 0\ 0\ 2\ 0\ 0]$

$d_2 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 2]$

$d_1 \cdot d_2$= *3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5*

$||d_1||$ = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)**0.5**= (42)^**0.5**= 6.481

$||d_1||$ = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)**0.5**= (6)^**0.5**= 2.245

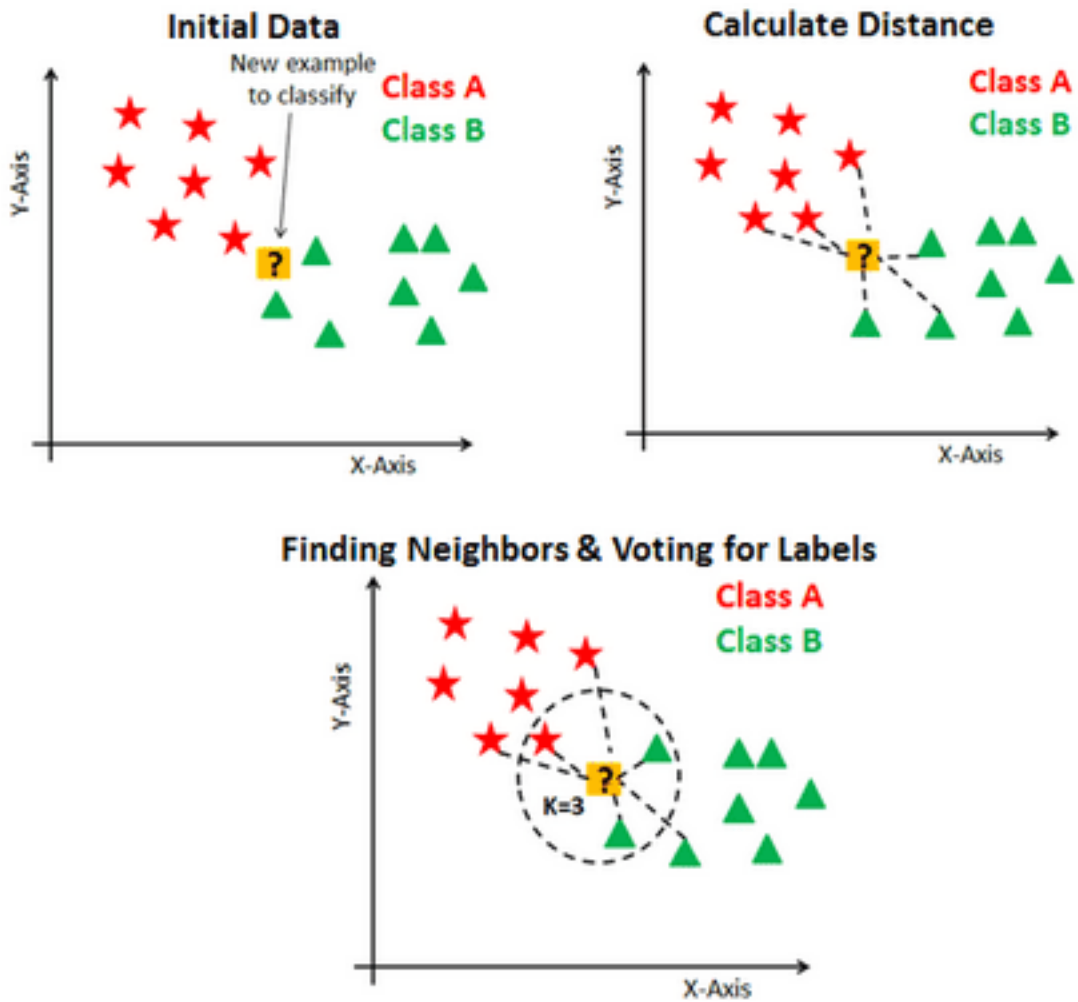
cos(*d1*, *d2*) = .3150

The Algorithm

Algorithm

1. Determine parameter K
2. Choose a sample from the test data that needs to be classified and compute its distance to all the training examples.
3. Sort the distances obtained and take the k-nearest data samples.
4. Assign the test class to the class based on the majority vote of its k neighbors. (根据选出来的k个neighbor中占多数的那个class作为当前这个data sample的预测类型结果)

The Algorithm



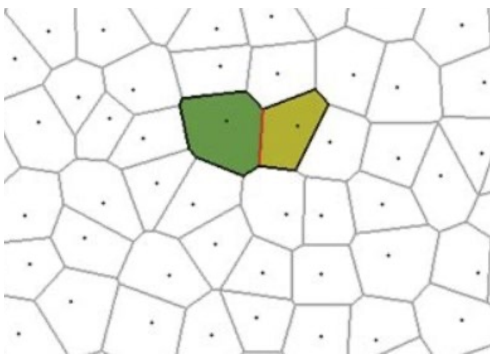
What is the best value of K to use?

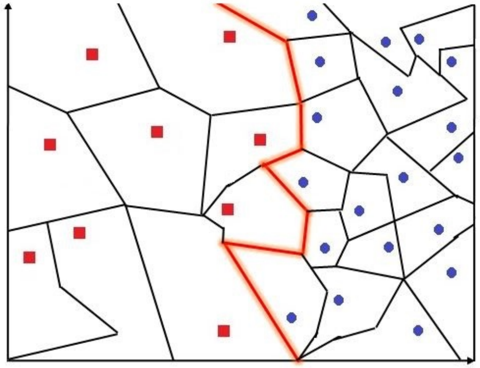
Decision Boundary

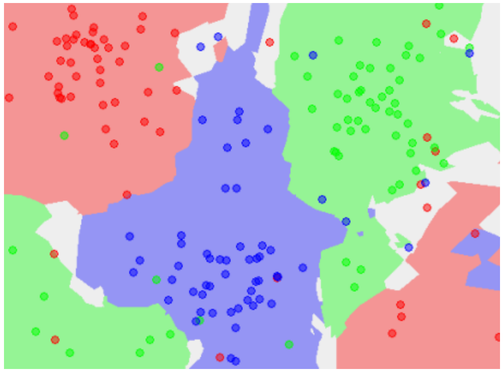
Voronoi Tessellation (沃罗诺伊分割)

- Partition the space into areas that are nearest to any given point
- Boundary: points at the same distance from two different training examples.

Decision Boundary: boundaries that separates two different classes.



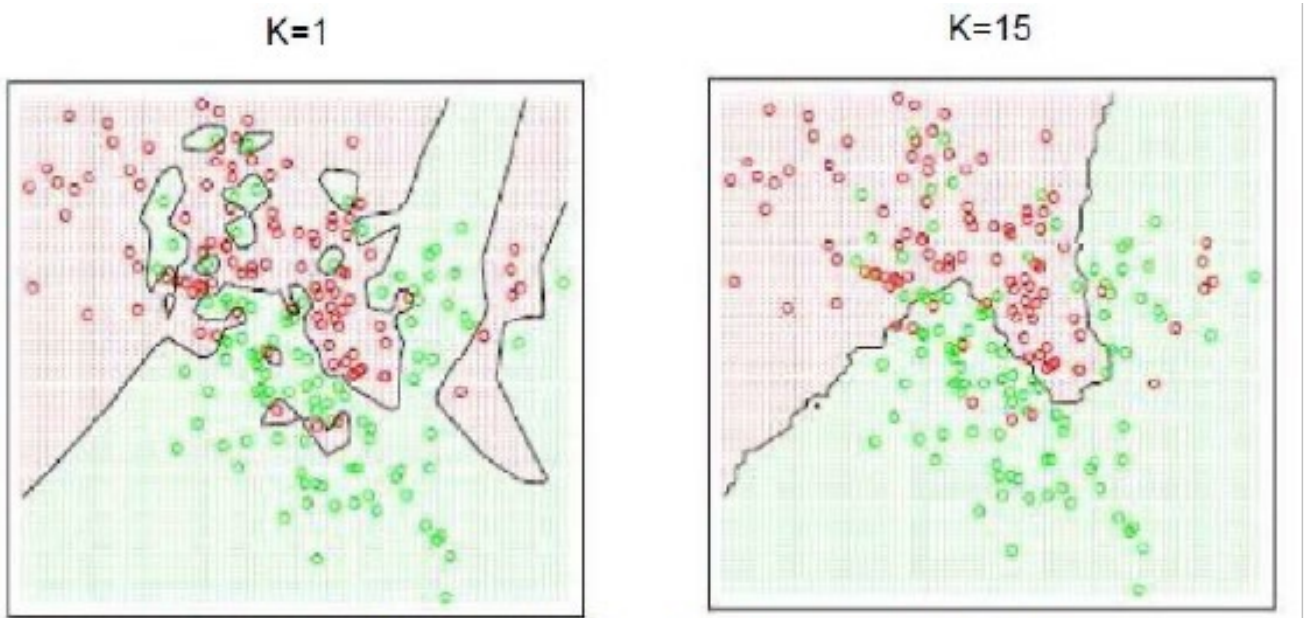




With large number of examples and possible noise in the labels, the decision boundary can become nasty!

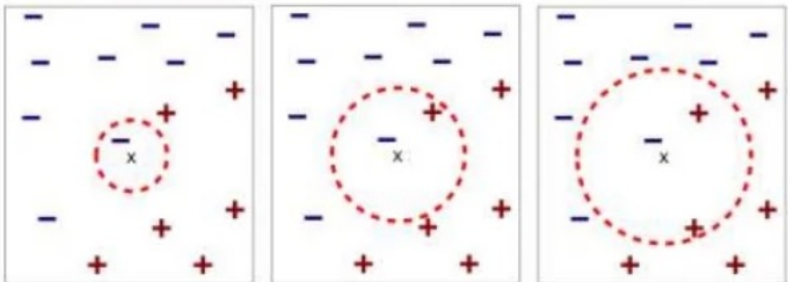
Effect of K

- Larger *K* produces smoother boundary effect
- When *K*==*N*, always predict the majority class



Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Discussion: which model is better between *K*=1 and *K*=15? Why?

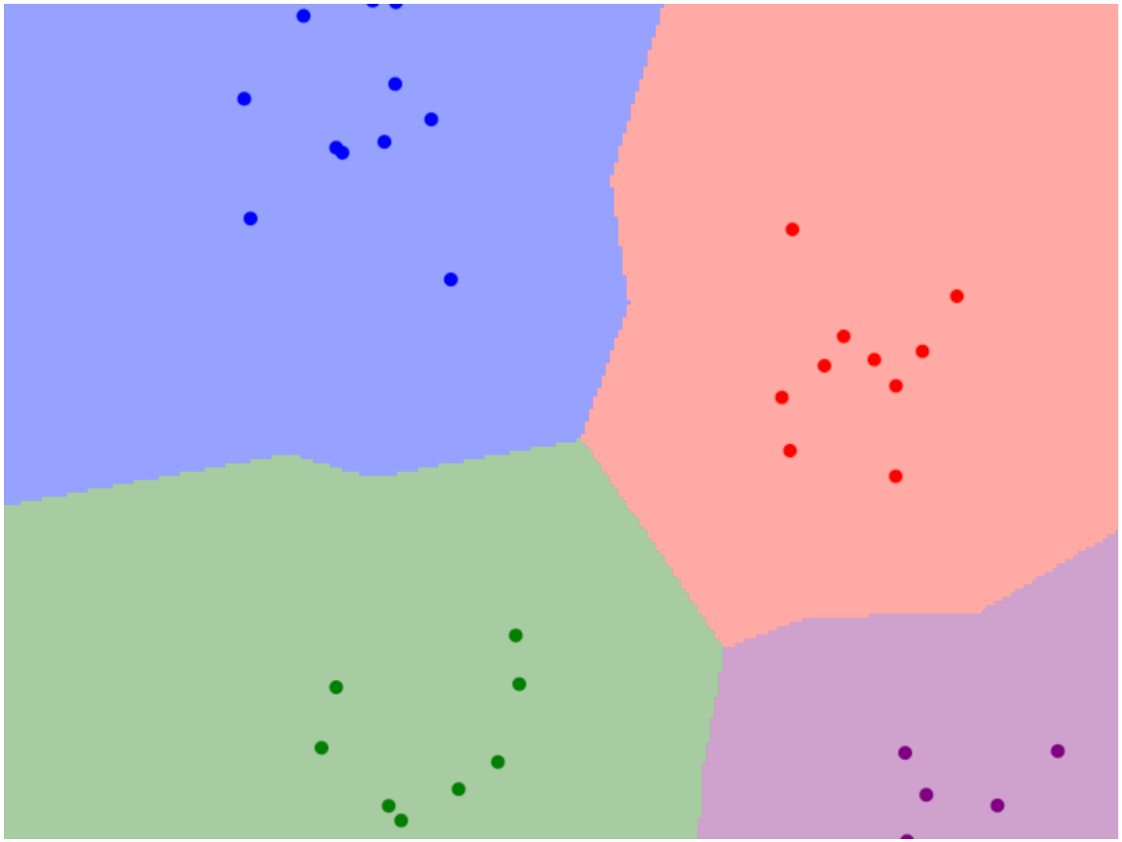


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

How to choose K?

- If K is too small, efficiency is increased but becomes susceptible to noise.
- Larger K works well. But too large K may include majority points from other classes, but risk of over-smoothing classification results

Try it yourself



http://vision.stanford.edu/teaching/cs231n-demos/knn/

Pros and Cons

Advantages:

- Simple to understand, explain, and implement,
- No effort for training,
- New data can be added seamlessly without hampering the model accuracy

Pros and Cons

Disadvantages:

- Does not scale with large data sets (calculating distance is computationally expensive)
- Highly susceptible to the curse of dimensionality
- Large storage requirements
- Data normalization is required

Tutorial: KNN with Python

What’s Next?

Logistic Regression

Classification by a discriminative function.

- Linear functions
- Differentiable!