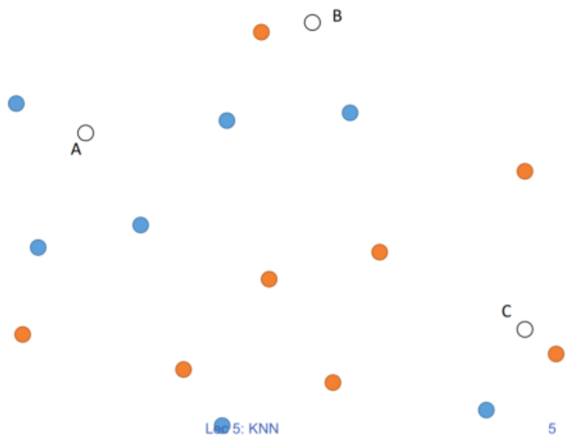
Lec Mon

Monday, October 15, 2018 3:58 PM

K Nearest Neighbors

Motivation: How would we color the blank circles?



Question: How are the boundaries different in KNN and decision tree?

Motivation: (Computer Vision) How can we classify what type of flower it is from a photo?

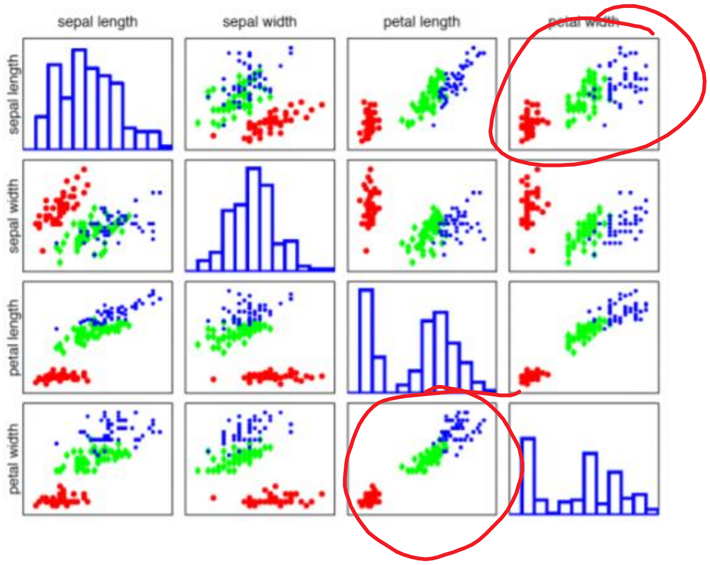
- KNN was widely used in CV before deep learning was around

Example data:

Fisher's *Iris* Data

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*
4.4	2.9	1.4	0.2	*I. setosa*
4.9	3.1	1.5	0.1	*I. setosa*

Which features make a good separator?



Since the clusters are physically more separated

Now onto the algorithm:
Nearest Neighbors: The basic version

- training examples are vectors x_i associated with label y_i
- Learning: just store training examples
- Prediction: for a new example x,
  - find the training example x_i that is closest to x
  - predict the label of x to be the label y_i associated with x_i

K-Nearest Neighbors
Same as above, but:

- find the **k closest** training examples to x
- construct the label of x using these k points

Issues in designing KNN modeling

- how to choose k?
- how to define distance?
- how to aggregate the information from the k neighbors and make the prediction?

What kind of problems can KNN solve?

- classification: every neighbor votes on the label, predict the most frequent label among the neighbors
- regression: predict the mean value

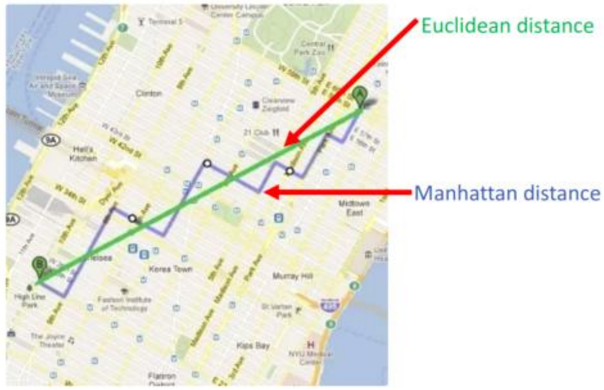
Distance - how to measure it?

- **Euclidean distance**

$$||\mathbf{x}_1 - \mathbf{x}_2||_2 = \sqrt{\sum_{i=1}^{n} (\mathbf{x}_{1,i} - \mathbf{x}_{2,i})^2}$$

- **Manhattan distance**

$$||\mathbf{x}_1 - \mathbf{x}_2||_1 = \sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|$$



- **L_p-norm**
  - Euclidean = L_2
  - Manhattan = L_1

$$||\mathbf{x}_1 - \mathbf{x}_2||_p = \left(\sum_{i=1}^{n} |\mathbf{x}_{1,i} - \mathbf{x}_{2,i}|^p\right)^{\frac{1}{p}}$$

$p \in Z^+$

- Most common distance is the **Hamming Distance**
  - number of bits that are different
  - or number of features that have different value
  - Ex:
    - x_1: {shape=triangle, *color=red*, location=left, *orientation=up*}
    - x_2: {shape=triangle, *color=blue*, location=left, *orientation=down*}
    - Hamming Distance = 2
  - we could apply weights to features if some are more important

How to find K?
Recap: Tuning parameters using a validation set

- Split data into:
  - Training Data
    - to train the model
  - Test Data
    - to evaluate the model
  - Development Data
    - to tune hyperparameters
- try a number of values for k. For each value,
  - train a model using training set
  - evaluate performance on development set
  - choose the value with best performance on dev set

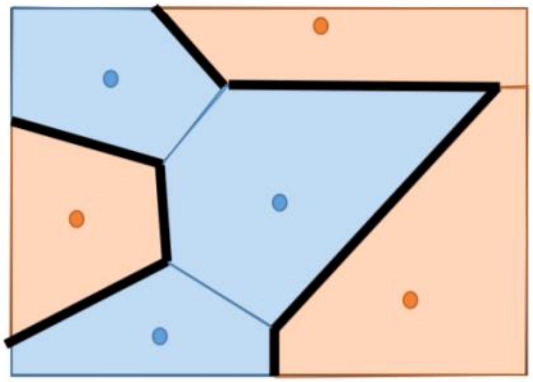
Tips/good practice

- In general, use an odd k to avoid ties
- **Feature Normalization** to prevent larger-valued features from dominating others in distance calculations
  - normalize data to have zero mean and unit variance in each dimension
  - how to deal with categorical features? usually we turn categorical into numerical features

decision boundary for KNN

- is KNN explicitly building a function? no

It can be represented as a **Voronoi Diagram**:



Instance based learning

- a class of learning methods involving
  - learning: storing examples
  - classifying based on existing, similar examples
- most computation is performed only at prediction time
  - like open vs closed book exams

Advantages

- training time ~ 0
- easy to update the "model", since we just add the new data points
- can learn complex decisions

Disadvantages

- hard to represent outliers
- stores all examples, model can be very big
- predicting new examples takes lots of time
- curse of dimensionality - things can go wrong in high-dimensional spaces
  - what's intuitive for 2-3 dimensions may not be for more dimensions
- because of the dimensionality curse, "neighborhood" becomes very large - distances become large, and certain features will dominate over less relevant ones