MACHINE LEARNING FOR DATA MINING
LECTURE 6: kNN CLASSIFICATION

K NEAREST NEIGHBORS

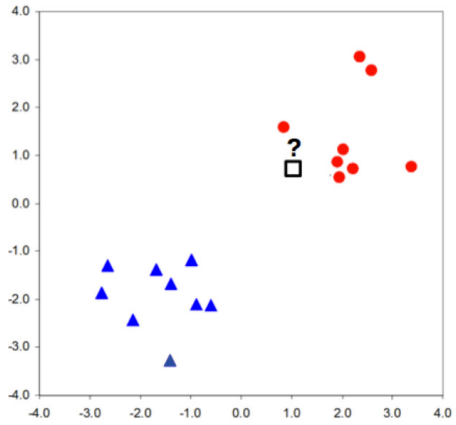
Siamak Sarmady (UUT, UU)
Viktor Lavrenko (Uni. Of Edinburgh)
Alexandre Ihler (UCI)

Nearest Neighbor

i.e. kNN, k=1

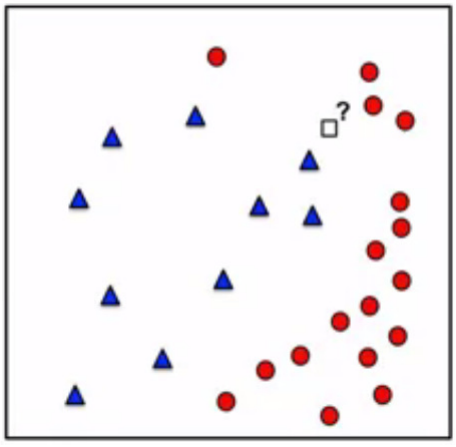
Nearest Neighbor (single neighbor!)

- We have a 2 dimensional training data set and the items are labeled with either red or blue.
- Does the box belong to red class or the blue class?
- How did you guess that?
  - Did you count the prior or the blue and red items (like you do in Naïve Bays)?
  - Did you fit a hyper plane?
- We just saw that the nearest points are red!
- And this concept is behind nearest neighbor(s) algorithm.



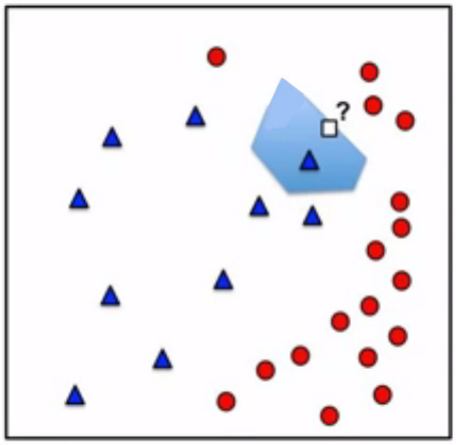
Nearest Neighbor (single neighbor!)

- How about this case?
- Nearest neighbor algorithm will calculate the distance of the data to near ones.
- Then the distances are ranked and the data is assigned to the same class as the nearest data point.
- If the nearest data point is blue, then the blue label will be assigned to the new data item.



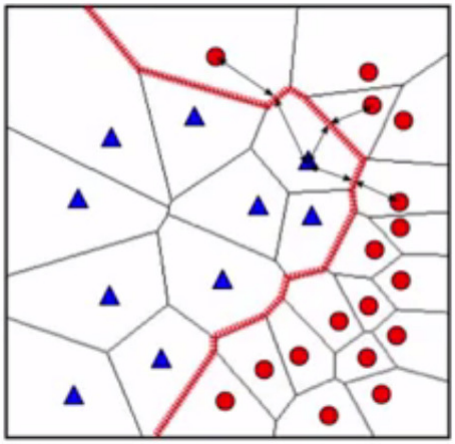
Nearest Neighbor – Voroni Cell

- In fact a region around each data point of training set can be determined that is called the Voronoi cell or region of the data point.
- Every point in the Voronoi cell is nearer to this specific blue point than any other training data point.
- The cell is a polygon (the lines are perpendicular to the line connecting neighboring points)



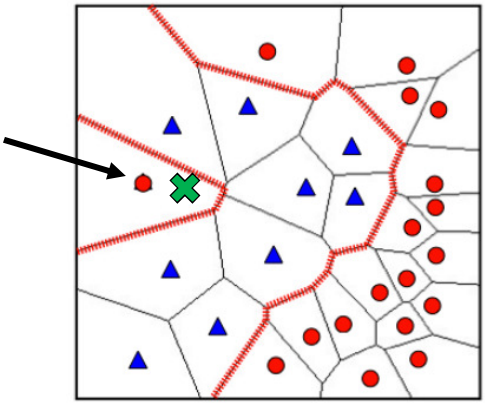
Nearest Neighbor – Voroni Cell

- If you draw the Voronoi cells of each training data point, the space will be divided into separate regions.
- These lines will eventually determine the class boundaries. The boundary follows the boundary of the Voronoi cells of similar class.
- Boundary is non-linear and could be very complex!
- This is impressive for a simple method like this.



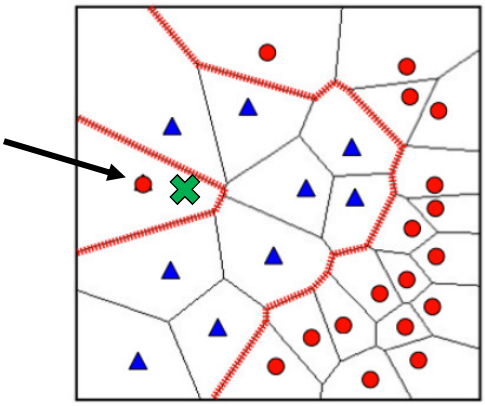
Nearest Neighbor – Sensitivity

- The simplicity of this classifier comes at a cost. The classifier is very sensitive to outliers.
- This means the classifier may over-fit to training data set and in fact memorize it.
- Assume one of the data points has been wrongly labeled (or is an outlier).
- As far as the classifier is concerned, any data that falls in the Voronoi cell of the red training data will be classified as red.
- So just one data item can hugely change the class boundaries.
- We don't want small changes to have big affect on our classifier.



Nearest Neighbor – Soloution

- One way to solve the problem is to look into more than one neighbor to determine the class of a new item.
- In this way the affect of just one training item on the specification of the class of the new item will be reduced.
- If we have 4 other training points with label "blue", they will reduce the affect of the "red" outlier
- We'll take the majority vote from the neighbors…
- and we get a blue classification for the regions around the "red" outlier.



kNN: k Nearest Neighbors Classification

k Nearest Neighbors

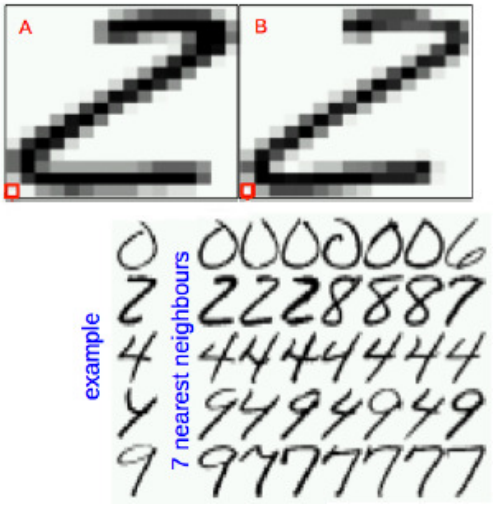
- **Training Data:** is a set of n data points in the form of (X_i, y_i)
  - Each data point Xi has a set of m attributes $\{x_{i1}, x_{i2}, ..., x_{ij}\}$
  - y_i is the label of the item. Labels are from a set of either 2 or more class names
    - {spam, not-spam}, digits{1, …, 9} …
- **Test Data:** is one or a set of points that we want to classify. Let's show the point with X_t
- **Algorithm:**
  - Compute the distance of X_t to all of the points in the training data set $D(X_t, X_i)$
  - Rank the training points based on the distance (in ascending order)
  - Now select k points from them (with the least distance to the point)
  - The selected label is the one that is more frequent among the k lists (gets higher votes)

kNN Applications – Character Recognition

- MNIST handwritten digit database (Yann LeCun)
- Each digit is properly segmented (and oriented)
- 16 pixels x 16 pixels with 8 bit gray scale (i.e. 256 possible values for each of 256 attributes.
- The distance is calculated using Eucleadian distance.

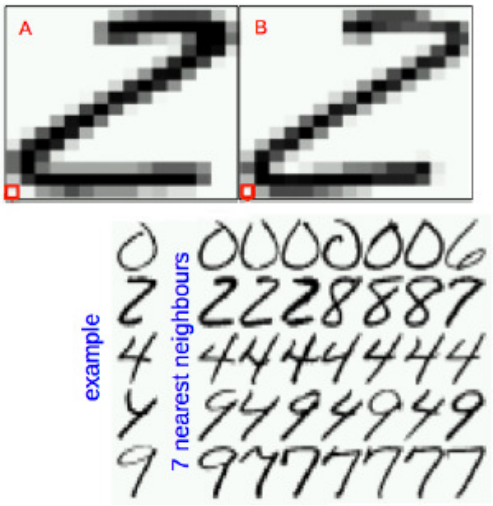
$$\mathrm{D}(\mathrm{A},\mathrm{B}) = \sqrt{\sum_r \sum_c (A_{rc} - B_{rc})^2}$$

- We calculate the distance of the new character (test item) with all the characters in the training dataset. We select the class which is more common among the k most near points in the training data set to the character in question.



kNN Applications – Character Recognition

- The accuracy (performance) is quite good for such a simple algorithm in comparison to powerful classifiers like SVM.
  - 7-NN: 95.2%
  - SVM: 95.8%
  - Humans: 97.5%



kNN Regression

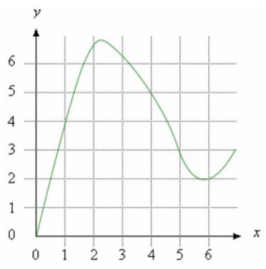
kNN Regression

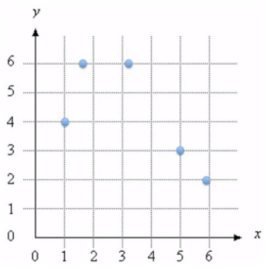
- **Regression:** kNN can be used for regression analysis too.
- **Training Data:** is a set of n data points in the form of (X_i, y_i)
  - Each data point Xi has a set of m attributes $\{x_{i1}, x_{i2}, \ldots, x_{ij}\}$
  - y_i is a single real value: $y_i \in \mathbb{R}$.
- **Test Data:** is one or a set of points (with n dimensions) whose y value should be calculated. We show the point with X_t and the output is shown as y_t
- **Algorithm:**
  - Compute the distance of X_t to all of the points in the training data set $D(X_t, X_i)$
  - Rank the training points based on the distance (in ascending order)
  - Now select k points from them (with the least distance to the point)
  - Calculate an average of the y value of those k training points. This will be the predicted output for the new point (i.e. y_t).

$$\hat{y} = \frac{1}{k}\sum_{j=1}^{k} y_j$$

kNN Regression

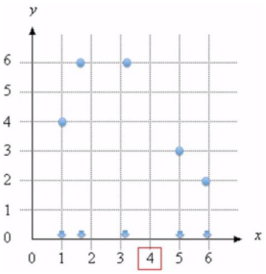
- Let's assume we have a data set which in fact matches the above graph.
  - Note that we don't have this graph or function, otherwise we wouldn't need regression.
- We just have the lower set of training data.
- In contrast to other regression models, we do not build a model beforehand.

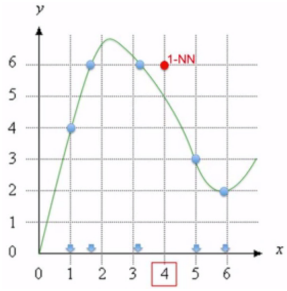


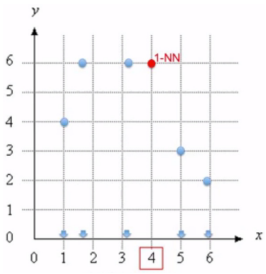


kNN Regression

- We want to predict the value of y for an x=4
- We first find N nearest points to x=4, and calculate the average of their y
- For N=1 this will look like the figure at the bottom
- Does it look good?

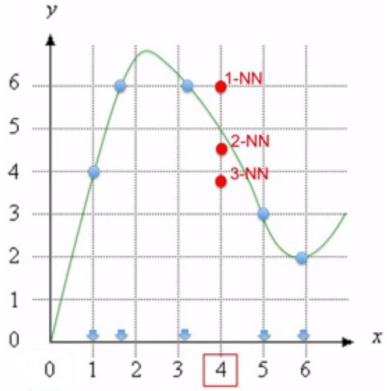






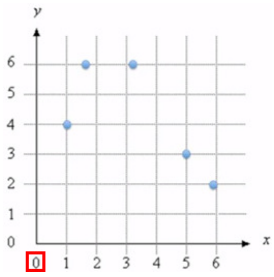
kNN Regression - Interpolation

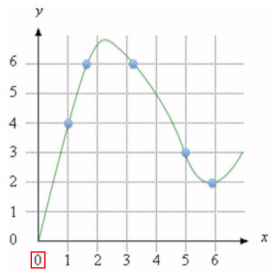
- With 2 points the accuracy increases. N=2 is in fact an interpolation
- With 3 points the accuracy is worse (in the case of this example which provides very sparse data)
  - If the points had better coverage we would get lower average errors for a set of predictions (not just one).



kNN Regression - Extrapolation

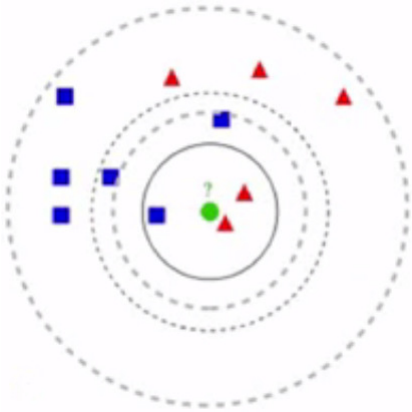
- Can we find a y value for x=0?
- This becomes and interpolation problem and kNN is not good at that.





kNN Classification – What k value?

- Least value: k=1
  - The method will become nearest neighbor (single neighbor) in this case.
  - As we saw the method was very susceptible to outliers and over-fitting…
- Largest value: k = n
  - In this case all the neighbors will be included in the voting. As a result the voting will always select the class with higher number of members (higher frequency).
  - This is somehow similar to Naïve Bays method.
  - This means severe under-fitting (i.e. disregards the structure of the data, just counts class members and selects the class with higher frequency every time)



kNN Classification – What k value?

- So k should be something between 1 and n.
  - There is no concrete method for determining a proper value of k, except cross validation (i.e. using training and validation/test datasets) to determine what k value gives higher accuracy.
- **Note:** what value of k gives the highest accuracy for the training set? Always 1.

kNN method – Distance function

- **Distance function:** is possibly the only parameter of kNN that you could modify (beside k).
  - Different Distance functions might give totally different results.
- **Euclidean distance:** is The most popular distance function

$$\mathrm{D}(\mathrm{A},\mathrm{B}) = \sqrt{\sum_d |A_d - B_d|^2}$$

  - **Advantages:** It is symmetrical, spherical and treats all dimensions equally
  - **Disadvantages:** very sensitive to deviations in a single attribute
    - If all attributes are near but one of them is off by a big value, that difference is going to be squared and affect the whole distance calculation (undermining the affects of other attributes)
- Hamming Distance (for categorical/nominal attributes):

$$\mathrm{D}(\mathrm{A},\mathrm{B}) = \sum_d 1_{A_d = B_d}$$

  - Number of attributes which are different

kNN method – Distance function

- **Minkowski Distance:** as discussed in an earlier chapter, is the generalization of Euclidian distance.

$$\mathrm{D}(\mathrm{A},\mathrm{B}) = \sqrt[p]{\sum_d |A_d - B_d|^p}$$

  - p = 2 : Euclidian distance
  - p = 0 : hamming distance (logical and)
  - p = 1 : Manhattan distance (city block distance)
  - p = ∞ : $\max_d |A_d - B_d|$ (logical or) gives the maximum distance among attributes

kNN using Scikit Learn

Using SkLearn – Program 1 (Data from Array)

```
import numpy as np

features_train = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1],[3,3]])
labels_train   = np.array([1, 1, 1, 2, 2, 2])
features_test  = np.array([[-2, -2], [-2, -3], [2, 3], [1, 2]])
labels_test    = np.array([1, 1, 1, 2])

from sklearn import neighbors

n_neighbors=3
clf = neighbors.KNeighborsClassifier(n_neighbors, weights='uniform')
clf.fit(features_train, labels_train)
pred = clf.predict(features_test)

print "Test labels:      ", labels_test
print "Predicted labels: ", pred

from sklearn.metrics import accuracy_score
print "Accuracy:         ", accuracy_score(pred, labels_test)

print "\nPredicted label for ", [-0.8, -1] ," is ", (clf.predict([[-0.8, -1]]))
```