UNIVERSITY OF NORTH CAROLINA AT CHARLOTTE

ITCS-6156-00

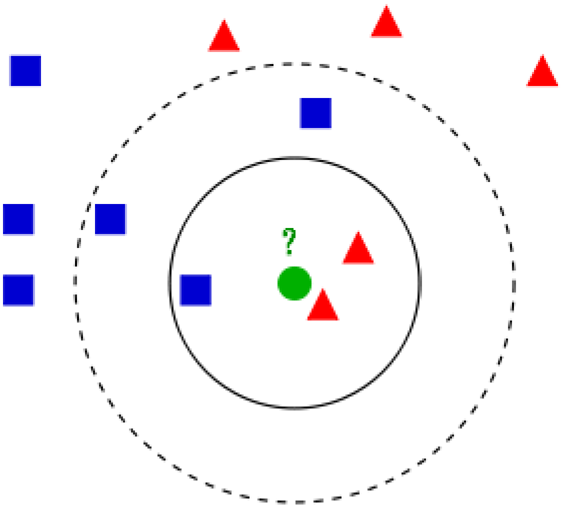
K-Nearest Neighbor & Boosting

Assignment 3 - Report

Archit Parnami

3/12/2017

Designing a K-Nearest Neighbour Classifier



Source: https://en.wikipedia.org/wiki/File:KnnClassification.svg

The KNNClassifier class

- The input to the KNNClassifier are:
    - Data points labelled with a class
    - The value K – Number of neighbours to look at
    - The distance metric to be used, which can be:
        - Euclidean
        - Manhattan
        - Hamming
    - Weighted (True/False) – Should the algorithm compute the weight of each data point and consider it for classification?

KNNClassifier
+Input: int[][] +K: int +distance_f: function +weighted: bool
-eulidean_distance(int[], int[]): float -manhattan_distance(int[], int[]):float -hamming_distance(int[], int[]):float -get_distance_function(string):function -get_nearest_neighbours(int[]):[(int[], float)] -find_majority([(int[], float)]): int +query(int[]):int

Dataset 1 - Optical Recognition of Handwritten Digits

Implementation & Analysis

Using KNNClassifier Class

- Number of input units = 64
- Number of output units = 10
- Number of Training examples = 3823
- Number of Test examples = 1000

K	Distance Function	Weighted	Accuracy
1	Euclidean	False	**97.8**
3	Euclidean	False	97.7
5	Euclidean	False	97.8
1	Euclidean	True	97.8
3	Euclidean	True	97.7
5	Euclidean	True	97.8
1	Manhattan	False	97.1
3	Manhattan	False	97.1
5	Manhattan	False	97.2
1	Manhattan	True	97.1
3	Manhattan	True	97.2
5	Manhattan	True	**97.6**
1	Hamming	False	85.6
3	Hamming	False	87.2
5	Hamming	False	84.3
1	Hamming	True	85.6
3	Hamming	True	**87.4**
5	Hamming	True	84.7

Accuracy score observed using KNNClassifier with different parameters

(Accuracy as measured on Test Set)

Dataset 2 – Amazon reviews sentiment Analysis

1. Features
    - Product name and review
    - Number of features = 2
2. Output
    - Rating from 0 to 5
3. Number of Observations = **146824**

Original Problem: Given a review of a product predict the rating.
Modified Problem: Given a review of a product rate the product as **good** or **bad**.

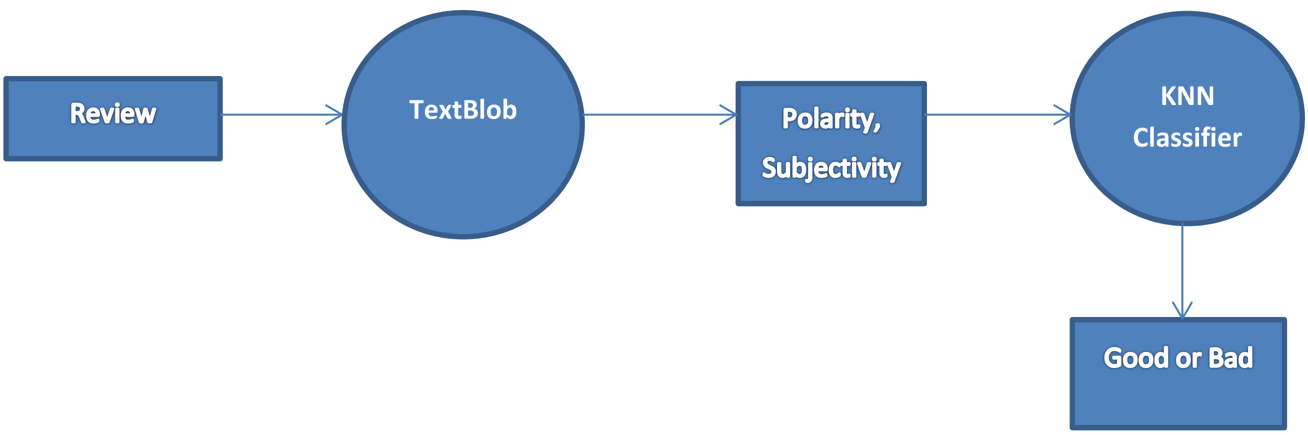
Problem Solving Approach:

- The polarity and subjectivity of a review is obtained by performing sentiment analysis with a 3rd party library called **TextBlob**.
- If a review has a rating < 3 then it is considered bad(-1).
- If a review has a rating >=3 then it is considered good(1).
- Polarity & Subjectivity are then fed to a **KNN Classifier** as input features.
- While rating of -1 & 1 is used to represent negative & positive output respectively.

Implementation Steps

1. Sentiment Analysis & Input Generation

- Input File: amazon_baby_train.csv, amazon_baby_test.csv
- Output File: Train-SentimentAnalysis.csv, Test-SentimentAnalysis.csv
- Library used for finding the Sentiment Analysis: TextBlob



Sample Input

Name	Review	Rating
Moby Wrap Original 100% Cotton Baby Carrier, Red	Bought this for my daughter....	5
Child to Cherish Handprints Tower Of Time Kit in Pink	It is very cute, and I got a lot of compliments....	4
JJ Cole Lite Embroidered Bundleme, Pink, Infant	This product is very pretty but does not fit the Graco Safe Seat	1

Sample Output

Polarity	Subjectivity	Rating
0.347	0.688	1
0.235	0.56	1
0.091	0.46	-1

2. Model Generation

- Input Files: Train-SentimentAnalysis.csv, Test-SentimentAnalysis.csv
- **Problem Statement**
  - Given the polarity, subjectivity and the rating of a review feed the data to KNNClassifier.
  - Use this KNNClassifier to predict the rating of new reviews.

3. Implementation & Analysis

- **Using KNNClassifier Class**

  - Number of input units = 2 [Polarity, Subjectivity]
  - Number of output units = 2 [-1,1]
  - Number of Training examples = 146824
  - Number of Test examples = 100

K	Distance Function	Weighted	Accuracy
1	Euclidean	False	81
3	Euclidean	False	82
10	Euclidean	False	**84**
1	Euclidean	True	81
3	Euclidean	True	81
10	Euclidean	True	83
1	Manhattan	False	**83**
3	Manhattan	False	81
10	Manhattan	False	83
1	Manhattan	True	83
3	Manhattan	True	82
10	Manhattan	True	83
1	Hamming	False	83
3	Hamming	False	**85**
10	Hamming	False	85
1	Hamming	True	83
3	Hamming	True	85
10	Hamming	True	85

Accuracy score observed using KNNClassifier with different parameters

(Accuracy as measured on Test Set)

Boosting Implementation

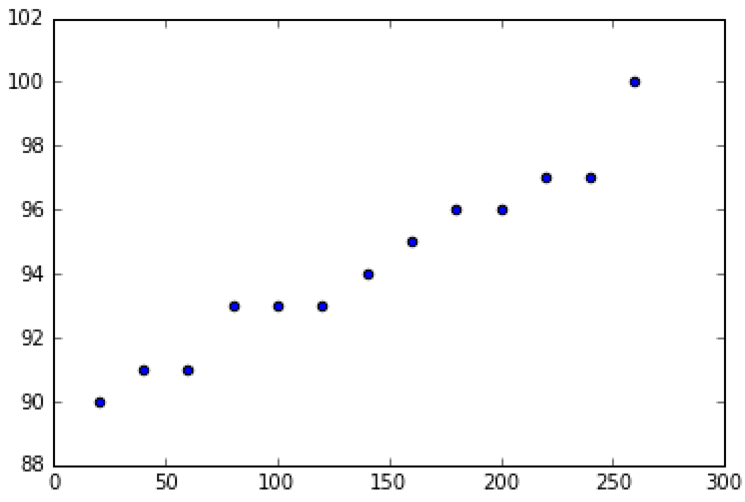
Algorithm

- Initialize the following:
  - X - Input Data
  - Y – Target output
  - n_items - Number of input Items
  - nIters – Number of Iterations / Number of hypothesis
  - alphas = []
  - weights = [1/n_items, 1/n_items, 1/n_items……., 1/n_items,] n_items times
  - weak_learners = [] (Set of hypothesis generated in each iteration)
- For each iteration
  - Generate a Decision Tree Classifier (H) with Max Depth = 1
  - Use H to fit X, Y with initial weights
  - Predict the output values of X on H
  - Calculate the weighted error on number of mismatches as follows:
    - For each item in Y
      - If prediction(Yi) != Yi
        - Error = Error + Weight(i)
  - Normalize the error as:
    - Error = Error / Sum of Weights
  - Calculate Alpha as:
    - Alpha = Log((1-Error)/Error) / 2
  - Update Weights as:
    - For each item i
      - If prediction(Yi) == Yi
        - Weight(i) = weight(i) / (2 * (1 – error))
      - Otherwise
        - Weight(i) = weight(i) / (2 * error)
  - Add H to the list of weak Learners
  - Add Alpha to the list of alphas
- Combine all the hypothesis with Alphas as :
  - H_Final = Sum [H(i) * Alpha(i)]   {i from 1 to number of iterations(nIters)}
- Predict Xt using H_Final as:
  - If H_Final(Xt) > 0
    - Return 1
  - Otherwise
    - Return -1

Applying Boosting on Amazon Data Set

- Number of input units = 2 [Polarity, Subjectivity]
- Number of output units = 2 [-1, 1]
- Number of examples used for classification = 100

Number of Classifiers	Accuracy
20	90.0
40	91.0
60	93.0
80	93.0
100	93.0
120	94.0
140	95.0
160	96.0
180	96.0
200	97.0
240	97.0
260	100.0



Accuracy Vs Number of Classifiers

SAMME Boosting for Multi Class Classification Implementation

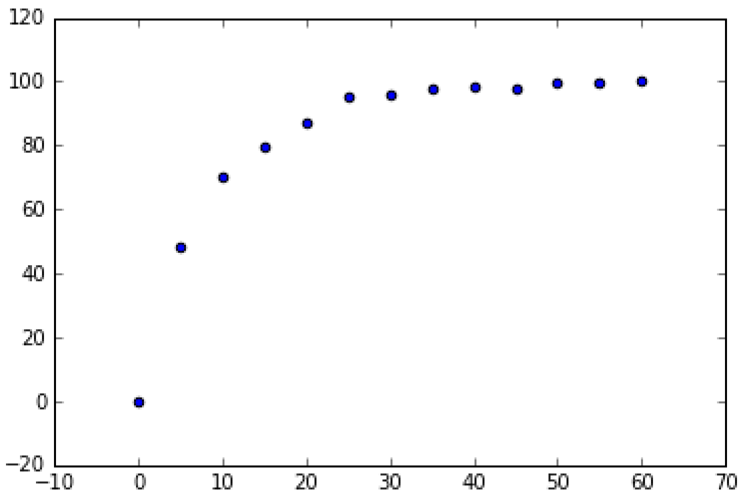
Algorithm

- Initialize the following:
    - X - Input Data
    - Y – Target output
    - n_items - Number of input Items
    - nIters – Number of Iterations / Number of hypothesis
    - alphas = []
    - weights = [1/n_items, 1/n_items, 1/n_items……., 1/n_items,] n_items times
    - weak_learners = [] (Set of hypothesis generated in each iteration)
    - K – Number of Output Classes
- For each iteration
    - Generate a Decision Tree Classifier (H) with Max Depth = 2
    - Use H to fit X, Y with initial weights
    - Predict the output values of X on H
    - Calculate the weighted error on number of mismatches as follows:
        - For each item in Y
            - If prediction(Yi) != Yi
                - Error = Error + Weight(i)
    - Calculate Alpha as:
        - Alpha = Log((1-Error)/Error) + Log(K-1)
    - Update Weights as:
        - For each item i
            - If prediction(Yi) == Yi
                - Weight(i) = weight(i) / (K * (1 – error))
            - Otherwise
                - Weight(i) = weight(i) / (K * error)
    - Add H to the list of weak Learners
    - Add Alpha to the list of alphas
- Predict Xt as:
    - Predict the output using each of the hypothesis h(i)
    - Sum the weights corresponding to the hypothesis which predict the same values
    - Return the predicted value against the hypothesis which has maximum sum of weights.

Applying SAMME Boosting on Digit Recognition Data Set

- Number of input units = 64
- Number of output units = 10
- Number of examples used for classification =200

Number of Classifiers	Accuracy
0	0.0
5	48.5
10	70.5
15	79.5
20	87.0
25	95.5
30	96.0
35	98.0
40	98.5
45	98.0
50	99.5
55	99.5
60	100.0



Accuracy Vs Number of Classifiers