Laboratory 1 Report
K-NN Classifier

14-06-2019

Diogo Correia s264324
diogo.valacorreia@studenti.polito.it

1 Theoretical Cheatsheet

1.1 Introduction to Machine Learning

Machine learning: a set of methods to automatically detect patterns in data and use them to predict future data to make inference and decisions about new data. [1]

Descriptive or **Unsupervised** Learning:

$$D = (x_i)_{i=1}^{N} \tag{1}$$

Predictive or **Supervised** Learning:

$$D = (x_i, y_i)_{i=1}^{N} \tag{2}$$

- **D** → training set [2]
  - **N** → number of training examples
  - $\mathbf{x_i}$ → features, attributes, i.e height and weight or even a complex structure object such as an image
  - $\mathbf{y_i}$ → categorical or nominal variable from some finite set $y_i \epsilon 1, ..., C$ - classification ; i.e male or female, can be a real-valued scalar (such as income level) - regression

Function 3 outputs the most probable class label. This is called **MAP** estimate.

$\hat{y} = \hat{f}(x)$ gives an estimate of unknown $y = f(x)$.

$$\hat{y} = \hat{f}(x) = arg \max_{C=1}^{C} p(y = c|x, D) \tag{3}$$

1.2 K-NN Classifier

k nearest neighbor classifier [1]:

- Look at the k points in the training set that are nearest to test input x
- Count how many members of each class are in this set
- Return the fraction as the estimate

$$p(y = c|x, D, k) = \frac{1}{k} \sum_{i \epsilon N_k(x,D)} \mathbb{1}(y_i = c) \tag{4}$$

In function 4, $N_k(x, D)$ → indices of the k nearest points to x in D. $\mathbb{1}(e)$ is the **indicator function** where returns 1 if e is *true* and is 0 if e is *false*.

1.3 Misclassification rate

$$err(f, D) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(f(x_i) \neq y_i) \tag{5}$$

2 Implementation & Results

2.1 Exercise 1

2.1.1 Algorithm

The task to implement a k-NN classifier in Matlab was implemented following the proposed points in the professor's slides [1]. They can be seen in section 1.2.

The implementation code can be seen in the Appendix (section 3.1).

The code checks the number of features and calculates the difference matrix of each feature between them from the test data and the training data. The algorithm follows by calculating the euclidean distance with the computed matrices and returns the k shortest distances for each test data entry.

The function them computes and returns the misclassified rate and misclassified test data indexes.

2.1.2 Results

In figure 1 can be seen the 2D representation of the training data, test data and misclassified test data for a $k = 12$.

The relation of the misclassification rate and k can be seen in figure 2. As expected, for $k = 1$ for the training data, the misclassification rate is zero.

Can be observed that with the increase of k starts to happen an overfitting of the data.

2.2 Exercise 2

2.2.1 Algorithm

Following the implementation and study case in Exercise 1 in section 2.1, it extends to a study of the misclassification rate dependency with the amount of training data provided to the k-nn algorithm and k.

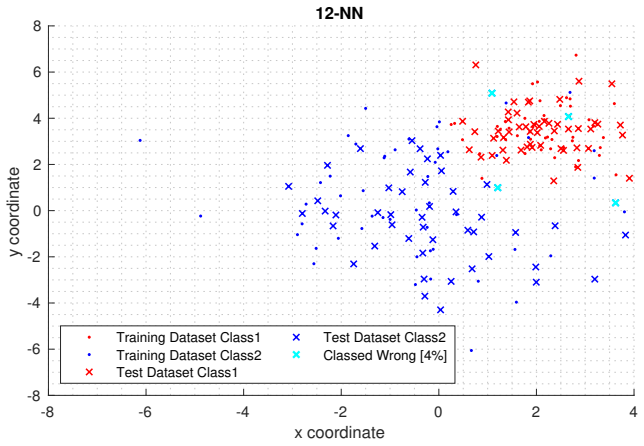

Figure 1: 2D Data Representation with missclassified points

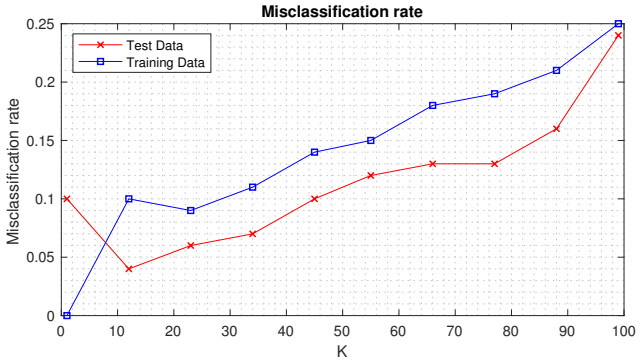

Figure 2: Misclassification rate relation with K

2.2.2 Results

The obtained result can be seen in figure 3. We can observe that with the increase in training data provided to the algorithm the misclassification rate decreases.

The overfitting can be also observed as the k increases.

2.3 Exercise 3

2.3.1 Algorithm

The implementation in exercise 3 followed the same path but with some matrices modifications and operations in order to be possible to work with the datasets.

The algorithm starts by computing the euclidean distance of each of the test vectors with each one of the training vectors. This results in $M * N_c$ cell vector, with each cell containing a three dimensional matrix $M * D * N_c$. With this results the sorter K index values are found and is computed the probability for each class.

The algorithm implementation can be seen in appendix section 3.2.

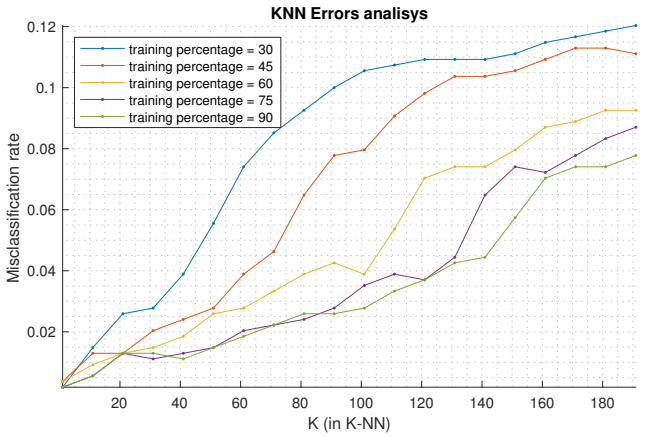

Figure 3: Misclassification rate dependency with k and the amount of training data provided to k-nn algorithm

2.3.2 Results

The results obtained can be seen in figure 4. We can see the higher accuracy for lower values of K, which is expected since the training dataset

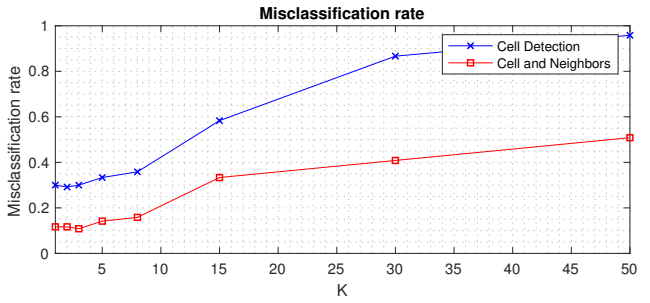

Figure 4: Misclassification rate for ex.3

3 Appendix

3.1 K-NN Classifier

K-NN Classifier in Ex.1

```
function [error, errorIndex] = knn_classifier(train_data,
↪ test_data, k)
  % 1. Look at the k points in the training set that are
↪ nearest to the test input

  % Get sizes of data
  n_features = size(train_data, 2) - 1;
  n_train_data = length(train_data);
  n_test_data = length(test_data);

  % Calculating euclidean distances
  dist_features = cell(1, n_features);
  dist = zeros(n_test_data, n_train_data);

  for f = 1 : n_features
    dist_features{f} = repmat(test_data(:,f), 1,
    ↪ n_train_data) - train_data(:,f)';
  end

  for f = 1 : n_features
    dist = dist + dist_features{f}.^2;
  end

  dist = sqrt(dist);

  % Calculating the k minimun distances
  [ ~, idx ] = sort(dist, 2);
  idx = idx(:, 1:k);

  % 2. Count how many members of each class are in
  ↪ this set
  test_data_classes = test_data(:, end);
  idx_class = test_data_classes(idx);

  % Classifier (choose the most common class)
  class = mode(idx_class, 2);

  % Return Misclassification error
  error = sum(class ~= test_data_classes) / n_test_data;

  % Return indexes of misclassificated data points
  errorIndex = find(class ~= test_data_classes);
end
```

3.2 K-NN Classification approach in Ex.3

K-NN Classification approach

```
[ D, M, Nc ] = size(traindata);
    % D - Number of measured features
    % M - Number of times measurements were taken
    % Nc - Number of classes
res = cell(M*Nc, 1);

% K-NN classifier: Returns an array for each testdata
↪ vector of size Nc with the probability for each
↪ class
for i = 1 : M*Nc
  % Calculate eucleadian distance between each test
  ↪ vector and each training vector
  r = traindata - testdata(:, i);
  r = sqrt(sum(r.^2, 1));

  % Find minimum K distance values
  [~, r] = mink(r(:), k);

  % Find classes for K minimum distances
  r = ceil(r./M);

  % Calculate probability for each class
  res{i} = sum(repmat(r, 1, Nc) == 1:Nc, 1)./k;
end
```

References

[1] Enrico Magli. Slides: Statistical learning and neural, ict for smart societies.

[2] K. Murphy. *Machine learning*. Cambridge, Mass.: MIT Press, 2012.