HarvardX: PH125.9x Data Science

Athanasios Chrysanthakopoulos

June 15 2019

Overview

This project is related HervardX: PH125.9x Data Science: Capstone course. The present report start with a general idea of the project and by representing its objectif.

Then the given dataset will be prepared and setup. An exploratory data analysis is carried out in order to develop a machine learning algorithm that could predict movie ratings until a final model. Results will be explained. Finally the report ends with some concluding remarks.

Introduction

The data set contains information about users in the Social Network. The Social Network has a business client which is a car company that puts ads on the social network and the social network gathered not only not only informations like the age and the estimated salary of those users but also took the response to these users to the ad that is zero if the user didn't buy the product the car and one if the user bought the product. It's a luxury SUV launched at a low price. So a lot of people when they saw the ad said let's do this let's get the car and we can see there a lot of buyers.

Aim of the project

The aim in this project is to train a machine learning algorithm that predicts if our users is going to buy this luxury cheap SUV. Here we 've got two independent variables (Age, EstimateSalary) and a dependent variable (Purchased).

To do this we are going to use the K-NN function.

Theory of K-NN

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

Both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/d, where d is the distance to the neighbor.[2]

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A peculiarity of the k-NN algorithm is that it is sensitive to the local structure of the data.

Algorithm: The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). In the context of gene expression microarray data, for example, k-NN has been employed with correlation coefficients, such as Pearson and Spearman, as a metric. Often, the classification accuracy of k-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

A drawback of the basic “majority voting” classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example, in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. K-NN can then be applied to the SOM.

Dataset

The dataset is on github file -> https://github.com/thanosclab/HarvardX-PH125.5x-Capstone

First we are going to import the dataset:

```
# Importing the dataset

dataset = read.csv('Social_Network_Ads.csv')

dataset = dataset[3:5]
```

Then we have to encode the target feauture as factor:

```
# Encoding the target feature as factor

dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
```

The next step is to split the dataset into the training set and the test set:

```
# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
## Warning: package 'caTools' was built under R version 3.5.3
set.seed(123)
split = sample.split(dataset$Purchased, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
```

The next thing we have to do is the feature scaling:

```
# Feature Scaling
training_set[-3] = scale(training_set[-3])
test_set[-3] = scale(test_set[-3])
```

Then we fitting the K-NN function to the training set and and predicting the test set results.It predicts if the users of our test set by yes or no to buy the SUV. The K-NN arguments are: Train: to specify what the training set is. The training set contain the indepedent variables and the dependent and we only nedd to take the independent variables. Test: to specify the test set. As the training set we need only the independent variables. Cl:To train a classifier the class needs to have both independent and dependent variables. k: the number of neighbors.

```
# Fitting K-NN to the Training set and Predicting the Test set result
s
library(class)
y_pred = knn(train = training_set[, -3],
             test = test_set[, -3],
             cl = training_set[, 3],
             k = 5,
             prob = TRUE)
```

Next thing is to make the confusion matrix:

```
# Making the Confusion Matrix
cm = table(test_set[, 3], y_pred)
```

We want to imagine if those imaginary pixel point as imaginary users of the social network will buy or not the SUV according to their coordinates and the graph that is their age and their estimated salary. Once the prediction is done it will be colorized in the proper color green if this imaginary user pixel point is predicted to buy the SUV and red if it's predicted not to buy the SUV. Visualising the Training set results:

```
# Visualising the Training set results
library(ElemStatLearn)
```

```
## Warning: package 'ElemStatLearn' was built under R version 3.5.3
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = knn(train = training_set[, -3], test = grid_set, cl = traini
ng_set[, 3], k = 5)
plot(set[, -3],
     main = 'K-NN (Training set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), a
dd = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3',
'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```

We can see there are a lot of irregularities but that's because of the way the K-NN model works, because it's taking each time the five nearest neighbors that's all the prediction boundary.These are two prediction regions, the red one and the green one. All these points are real observation points, real users if the dataset and their results. We can see that most of our predictions are correct because all the red points are in the red region and most of the green points are in the green region. We also see that the K-NN classify it makes a much better job at classifying those uses here that their logistic regression can classify correctly because it was a straight line and some green points were falling on the red region. So here is a much better job. And all those users with an age about the average and in Estimate salary below the average are well classified in a green region.

Visualising the Test set results:

```
library(ElemStatLearn)
set = test_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Age', 'EstimatedSalary')
y_grid = knn(train = training_set[, -3], test = grid_set, cl = traini
ng_set[, 3], k = 5)
plot(set[, -3],
     main = 'K-NN (Test set)',
     xlab = 'Age', ylab = 'Estimated Salary',
     xlim = range(X1), ylim = range(X2))
```

```
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), a
dd = TRUE)

points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3',
'tomato'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
```

Most of the red points are in the red region most of the green points are in the green region and of course we had a few incorrect predictions.

Conclusion

We can affirm to have built a machine learning algorithm to predict if someone is going to buy the new luxury SUV car compared with his salary and his age.

Appendix - Enviroment

```
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          5.2
## year           2018
## month          12
## day            20
## svn rev        75870
## language       R
## version.string R version 3.5.2 (2018-12-20)
## nickname       Eggshell Igloo
```