K-Nearest Neighbors Implented From Scratch

Eric Becerril-Blas   
Department of Computer Science  
University of Nevada, Las Vegasbecere1@unlv.nevada.edu  
  
Mingon Kang  
Department of Computer Science  
University of Nevada, Las Vegasmingon.kang@unlv.edu

*Abstract*— There is a famous saying that goes: “You are who you surround yourself with”. The K-Nearest Neighbors (KNN) Machine Learning Algorithm is based on the same saying. In this paper, we go through the implementation from scratch without using any libraries or built in functions of KNN.

Keywords— KNN, machine learning, data visualization, computer science education, mnist

# Introduction

In this paper, we will discuss how we implemented a K-Nearest Neighbors (KNN) algorithm from scratch. We use 2 versions of the MNIST dataset for this implementation: MNIST\_trainng.csv and MNIST\_test.csv. MNIST\_training.csv contains training data that you will find the K-nearest neighbors, whereas MNIST\_test.csv consists of the test data we needed to predict the labels.

# The Code

## How to run it

First, let’s go over how to run my code. The name of the script is KNNscatch.py. I used python 2 for this implementation and ran it via command line through the terminal. I created a virtual environment and used pip to install the following libraries: pandas, numpy, matplotlib, sklearn, and scipy. A visual example of how to run it can be seen in Fig. 1.

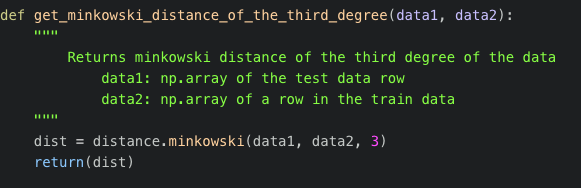


1. Command to run the code (Notice the virtual environment is on)

## Computing the Distances

Funding Agency: Eric Becerril-Blas’ Tuition Payment Funded This Paper

I computed the distances using a scipy function for the Minkowski distance. Documentation for the function is distance.minkowski(vals1,vals2,method), where vals1 and vals2 are the input value lists and method is denoted as a number to represent which distance method you use. If method=1 you calculate the Manhattan distance, if method = 2 it is the Eucladian distance, and if method = 3 it is just the Minkowski of the third degree. Through trial and error, I found that method =3 gave me the highest accuracies.



1. Code for Minkowski of Third Degree

I then return the distance and append it to a distance array to sort and get the K neighbors.

## Finding the Most Optimal K

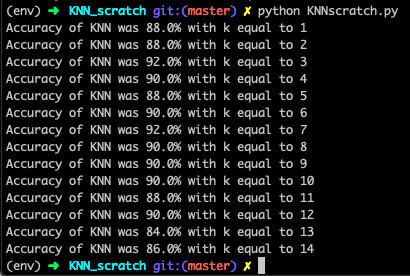
I didn’t exactly do anything cool for finding my most optimal K, I just created a for loop and iterated through many values of K until I found a good one. The longest loop I ran was for K equal to values from 1 to 95. A graph of this can be seen in Fig. 3

A close up of text on a white background

Description automatically generated

1. Accuracy vs. K

I found that the higher my K value, the lower my accuracy. But when my most optimal K values were that of K=3 and K=7. This can be seen in my command line output in Fig. 4.



1. Accuracies with different K values up to 14

# Conclusion

All in all, KNN was implemented correctly (hopefully), Accuracy was shown with different K’s, and accuracy was correctly measured.

##### Acknowledgment

We thank Eric Becerril-Blas for funding his own tuition to learn this topic. Without his generous contribution, this paper would not exist.