Assignment 1

Deadline: 26 Jan 2020, 9pm, to be submitted on brightspace.

Introduction

In this assignment you will use python notebooks to learn about kNN, the MNIST dataset, explore the curse of dimensionality, and try out the decision tree classifer in the sklearn library.

Questions

[8pts total] Q1. k-Nearest Neighbours Classifier on Synthetic Data

One of the important ways of both understanding and also debugging machine learning algorithms is to create your own, synthetic data sets. In this question you will implement a k-nearest neighbours (kNN) classifier and evaluate it on synthetic data.

a) **[2pts] Create the data set.** You will "train" a kNN classifier on the following training data:

- The data is 2-dimensional points in a grid, such that the x_1-coordinates and x_2-coordinates both range from $-1.5 \ldots 1.5$, with a spacing of 0.1 between points.
- A training point, $\boldsymbol{x} = (x_1, x_2)$, will be classified as follows:
  - Class 1 if $\|\boldsymbol{x}\|_2 \leq 1$
  - Class 2 otherwise

Write code that generates this dataset and displays it using a scatterplot, using different colours for each class.

Note $\|\boldsymbol{x}\|_p = (x_1^p + x_2^p)^{\frac{1}{p}}$ *can be implemented by:* `numpy.linalg.norm([x[0], x[1]], p)`

b) **[3pts] Implement the kNN algorithm.** This may be best done using a Python class, with a `train` method that takes the training data and k as input, and a `predict` method that takes a test example as input and returns a class prediction. You should find the nearest neighbours by using the Euclidean distance, $\|\boldsymbol{x} - \boldsymbol{y}\|_2$.

c) **[1pt] Visualize the predictions.** Use the `matplotlib.pyplot.contourf` function to display how your kNN classifier performs on new data. To do this, evaluate your model's predictions across a fairly fine `numpy.meshgrid`, with a space of about 10^{-2} between points. The meshgrid can have boundaries from -1.5 to 1.5 in both coordinates. By examining multiple plots, can you say how the performance of the classifier depends on k?

d) **[1pt] Try another data set.** Repeat part a), replacing the condition $\|\boldsymbol{x}\|_2 \leq 1$ with $\|\boldsymbol{x}\|_1 \leq 1$, and repeat part c) on the new training data. How does the performance of the classifier change with k? (Try some extreme values!)

e) **[1pt] Continue exploring data sets.** Repeat part a), replacing the condition $\|\boldsymbol{x}\|_2 \leq 1$ with $\|\boldsymbol{x}\|_{0.4} \leq 1$, repeat part c) with the data. Now how does the performance of the classifier change with k? Based on your results, do you think using Euclidean distance for kNN is always the best choice?

[5pts] Q2. k-Nearest Neighbours Classifier on MNIST Data

In this question you will use the K-nearest neighbours (KNN) classifier, implemented in Q1, to classify the MNIST images.
a) **[2pts] Explore k values.** Plot the classification accuracy on train and test sets by considering different values of k in the range of 1 to 100.

b) **[1pt] Explore data set size.** Plot the classification accuracy on train and test sets by considering different amounts of training data (randomly select the train samples ranging from 100 to 60,000) by considering the best value of k as obtained in part (a).
c) **[2pts] Explore image resolutions.** What happens if you repeat part (a) but reduce the resolution of the data sets?

MNIST dataset: The MNIST database (Modified National Institute of Standards and Technology database) of handwritten digits consists of a training set of 60,000 examples, and a test set of 10,000 examples. Additionally, the black and white images from NIST were size-normalized and centered to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

You can download the MNIST database from the following links: train set and test set

Every line of these files consists of an image, i.e., 785 numbers between 0 and 255. The first number of each line is the label, i.e., the digit which is depicted in the image. The following 784 numbers are the pixels of the 28 x 28 image.

[4pts total] Q3. The Curse of Dimensionality

Taken from an assignment by Roger Grosse

This question will explore what happens to the distances between randomly sampled points as the dimension is increased.

a) **[2pts] Computing expectation in 2D.** Take two continuous random variables, X and Y, sampled from a uniform distribution over the interval $[0, 1]$. Let Z be the squared Euclidean distance, defined by $Z = (X - Y)^2$. Compute $\mathbb{E}[Z]$ and $\mathrm{Var}[Z]$ (this will require integration, which you can compute numerically using `scipy.integrate.dblquad`). Explain the integrals that you are computing.

b) **[2pts] Computing expected distances in n-d.** Take two continuous random variables, X and Y, sampled from a uniform distribution over the unit cube in d dimensions. That is, if $X = (X_1, X_2, \ldots, X_d)$ then each X_i is uniformly sampled from $[0, 1]$, independently from each other. The squared Euclidean distance can be written as $R = Z_1 + Z_2 + \cdots + Z_d$, where $Z_i = (X_i - Y_i)^2$. Compute $\mathbb{E}[R]$ and $\mathrm{Var}[R]$. You can write your answers in terms of d, $\mathbb{E}[Z]$, and $\mathrm{Var}[Z]$. Again, you may explore this question empirically/numerically.

c) **[Non-marked]** Compare the mean and standard deviation of R to the maximum possible distance (i.e. the distance between opposite corners of the hypercube). How do the answers from a) and b) relate to the claim that "in higher dimensions, most points are far away, and approximately the same distance?"

[3pts] Q4. Decision Trees

Begin by watching this online video resource about decision trees:

`https://www.youtube.com/watch?v=-dCtJjlEEgM`

Use the sklearn library to run a decision tree classifier on each of the main datasets you used in the rest of the assignment:
(a) Let $\mathcal{D}_p$ refer to the dataset from Question 1, for different values of p, e.g. $\mathcal{D}_2$ refers to the dataset generation using the Euclidean norm, etc. Run a decision tree classifier on $\mathcal{D}_1$, $\mathcal{D}_2$, and at least 2 other values of p.
(b) Write code (not using built-in `sklearn` functionality) to split MNIST the training set to get a validation set. Then, *using* `sklearn`'s built-in `DecisionTreeClassifier`, write a function which tries 4-6 different values for `max_depth` and both information gain and Gini coefficient split criteria, and outputs the accuracies on the train, on the validation, and on the test sets[1]. What would be the meta-parameter settings that you would normally choose? Note that usually you would not run the classifier on the test set! This is simply a chance to experiment with (i.e. take a peek at) how the performance on the test set would change. Briefly

[1]That is, you can use the built-in classifier, but the validation code is your own.

summarize your observations, making sure to include clear description of any other meta-parameter choices that you made.

General Marking Notes and Tips

- You will be marked for accuracy, correctness and also clarity of presentation where relevant.
- Number each markdown cell in your notebook, so that the marker can reference their feedback to individual cells.
- Be selective in the experimental results you present. You don't need to present every single experiment you carried out. It is best to summarize some of your work and present only the interesting results, e.g. where the behaviour of the ML model behaves differently.
- In your answers, you want to demonstrate skill in using any required libraries to experiment with ML models, and good understanding / interpretation of the results. *Make it easy for the marker to see your skill and your understanding.*
- Justify your meta-parameter choices where appropriate.
- Except where you are explicitly asked to implement a particular algorithm/step yourself, if there is a procedure in sklearn that does the task, you are free to find it, read its documentation and use it. If you are not sure, it is best to on the shared notes or in class!
- You should submit one python notebook with your answers.
- Liberally add markdown cells to explain your code and experimental results. Make good use of the formatting capabilities of markdown to make your notebook highly readable.
- Add links to all online resources you use (markdown syntax is: [URL](anchor_text)). Avoid explicit long URLs in your markdown cells.