CSCC11 Winter 2025 Assignment 2 P2

Classification

Due: March 11th, 2025, 22:00 EDT

This is the second theory part and programming part of assignment 2, making use of lectures and tutorials for the first eight weeks. You are asked to derive solutions to problems to demonstrate your understanding of the concepts learned and then implement classification algorithms in the programming part. Submission instructions can be found at the end of the handout. We remind you that the work you hand in must be your own.

Please see the Generative AI Policy section regarding using AI chatbots such (ChatGPT, Copilot, Gemini et. al.) to help complete assignments in this course.

Student Name: Your Name Here

Student ID: Your student ID number

Email: Your UofT Email

Theory Part

Q4. Naive Bayes (6/70 points)

In the lecture, we discussed Naive Bayes model for classification problems. The key assumption is that we assume that features are independently distributed given class labels. You have a dataset with 10 training examples (see Table 1), and the goal is to classify whether an email is Spam (1) or Not Spam (0) based on three features:

- F1: The email contains the word "Discount" with possible values of {Y, N}.
- F2: The email contains a suspicious URL with possible values of {T, F}.
- F3: The length of the email with possible values of {S, M, L}.

let C_1 = spam

C_2 = not spam

Data Index	F1(Discount)	F2(URL)	F3(Length)	Target(Spam)
1	Y	F	S	1
2	Y	T	M	1
3	N	F	L	0
4	N	T	M	1
5	Y	F	L	0
6	N	F	S	0
7	N	T	L	0
8	Y	F	M	1
9	N	T	S	0
10	Y	T	L	0

Table 1: Email Dataset

$P(C=j \mid F_1F_2F_3) = \frac{P(F_1F_2F_3 \mid C=j)P(C=j)}{P(F_1F_2F_3)}$

(a) [1 pt] Write the prior probabilities of the target class label.

$P(C=1) = P(\text{spam}) = \frac{4}{10}$ $P(C=2) = \frac{6}{10}$

(b) [3 pts] Write the conditional probabilities of Features F1, F2, and F3.

$P(F_1F_2F_3 \mid C=1)$ $P(F_1F_2F_3 \mid C=0)$

(c) [2 pts] Given a new email with F1=Y, F2= F, and F3= M, using the Naive Bayes model to predict if it is a Spam or not. You should show your work by

- writing the formulas for the probabilities that you need to calculate using the Naive Bayes model you created in the previous steps.
- describing how you will predict the class label.

$\Rightarrow P(F_1 \mid C=1)\, P(F_2 \mid C=1)\, P(F_3 \mid C=1) =$

Q5. Logistic Regression (4/70 points)

Assume we want to build a logistic regression model to determine whether a loan application will be approved based on the applicant's annual income and credit score. The first few examples of the training data may look like as follows:

Data Index	Annual Income ($1000)	Credit Score	Loan Approved
1	69.99	627	0
2	50.951	504	1
3	75.478	516	0

Table 2: Loan Application Dataset

The annual income is in $1,000, the credit score is an integer and for the loan approval, 0 means rejected and 1 means approved. Write the corresponding optimization problem in terms of the data provided above and specify the parameters to be estimated.

Programming Part

Question 1. Classification using KNN and Random Forest (21/70 points)

The MNIST [1] dataset has been widely used in machine learning. It comprises images of handwritten digits, centered and resized to 28 × 28 pixels. Each image has been saved as a vector with the dimension of 784 = 28×28, and is associated with a label, namely, digits 0 through 9. MNIST has long been used to help evaluate machine learning techniques and pattern recognition methods as the data are authentic data, yet require minimal preprocessing. A small sample of MNIST images is shown in Figure 1.

`scikit-learn`, often referred to as `sklearn`, is a popular machine learning library for the Python programming language. It provides simple and efficient tools for data mining and data analysis, built on other popular scientific computing libraries such as `NumPy`, `SciPy`, and `Matplotlib`. For this section of the assignment you need to install `Numpy`, `matplotlib`, and `sklearn`. MNIST has already been imported into `sklearn`. Please complete the code based on the text provided for each section in the Python file. The blank sections that need completion are indicated by '`# TODO`'.

In this question, you will use the `sklearn` random forest classifier with entropy criterion and the KNN classifier to classify MNIST images. The model will be trained using the training set, and its performance will be evaluated on previously unseen data. The primary aim

[1] short for Modified National Institute of Standards and Technology

Figure 1: MNIST Dataset

for splitting the data into a training set and a test set is to quantify the generalization to data from the same distribution as the training data, but not seen during training. This assumption is crucial to avoid over-fitting and to ensure the model's predictive accuracy on unseen data. Use the following exercises to familiarize yourself with the `sklearn` API, and to gain proficiency in the implementation of training/validation splits, and tuning hyper-parameters for robustness to image noise. The exercises will enhance your understanding of machine learning processes, and gain practical skills in training models. In the scenario presented in the code section below, respond to the following questions. Please keep answers to discussion questions brief, using just two or three sentences.

- In the first scenario, we're working with noiseless input images and labels. The aim is to explore hyperparameter tuning for KNN classifiers and decision trees. For weighted KNN Classification we need to select the number of neighbors with a Euclidean distance measure. For Random Forests, our exploration involves finding the minimum number of samples required for a leaf node for up to 100 trees in the forest. During model training, we aim to find the hyper-parameters that deliver the best performance for both the training and test sets. With this context, using `MNISTclassification.py` with variables `Q2 = False` and `Q3 = False`, answer questions below.

  - For KNN with uniform weighting, plot the accuracy of training and testing with respect to the number of neighbors. Vary the number of neighbors from 1 to 100 in steps of 5 (1, 6, 11, 16 ...). What is the best number of K? Justify your selection.

  - For Random Forests, plot training and test accuracy as a function of the minimum number of samples for a leaf node (from 5 to 50 with 5 steps). Select the best hyper-parameter, and justify your choice.

- In the second scenario, our objective is to delve into the concept of generalization.

Typically, we expect a model trained on a large training set to exhibit similar performance on a smaller test set, assuming both are drawn from the same distribution. Practical experience often contradicts this assumption however. In reality, the test set distribution may not align perfectly with the distribution of training data, leading to challenges in obtaining good generalization. This underscores the importance of critically assessing model performance in real-world scenarios where data distributions may deviate from commonplace idealized assumptions. To mimic this behaviour, we add some noise to the test set. With this new data, please answer these questions, using `MNISTclassification.py` with `Q2 = True` and `Q3 = False`.

  - Perform a hyper-parameter search for KNN and Random Forest and report the best hyperparameter for each method.

  - Discuss the results based on both algorithms and how noise can increase or decrease performance.

  - Propose a method to enhance overall performance. Note that there is no single correct answer.

- In the third scenario, our objective is to assess the performance of the model in the context of dealing with noisy labels. Above we assumed noiseless labels, but practical experience shows that such noise is common in real-world datasets. This noise can stem from various sources, including human error in the annotation process, leading to less precise labeling. The introduction of such noise into the data has the potential to reduce a model's ability to generalize effectively, highlighting the need for robust strategies to handle and mitigate the impact of noise in the training and evaluation processes. To this end, we add noise to the training labels. Using the `MNISTclassification.py` with `Q2 = False` and `Q3 = True`, answer the following questions:

  - Perform a hyper-parameter search for KNN and Random Forest and report the best performance achieved by each method.

  - Discuss how the optimal parameters for KNN change between Scenario 1 above and Scenario 3 here.

Question 2. Logistic Regression with Batch Gradient Descent (9/70 points)

Using the model you build in Theory Part Q5, perform 10 iterations of the batch gradient descent algorithm to determine the parameters assuming that the initial estimate of the parameters is set to 0.1's and the learning rate is 0.1. The training set `loan_application.csv` is on Quercus. For each estimate (including the initial one), report the following

- The value of the estimated parameters

- The accuracy of the resulting logistic regression model when applied to the training

data.

You are free to write your own python program from scratch or you may use the starter file `Q2_LR_BGD.py` provided by the teaching staff. You may modify any part of the starter file.

Note you will need to do feature scaling to all the features and the standardization is a good one for this dataset. That is we transform features to have a mean 0 and a standard deviation of 1. For a feature x, standardization is done as

$$x_{scaled} = \frac{x - \mu}{\sigma},$$

where x is the original feature value, μ is the mean of the feature values, and σ is the standard deviation of the feature values.

Submission Instructions

Theory Part Q4 and Q5 This part may be done with pen and paper or a text editor (eg LaTeX). Formatted answers are easier to read but likely require more time. In any case, your answers should be handed in electronically as a single pdf file (for the 2 questions). The file should be called `a2tp2_solution.pdf`, and it should be submitted to Markus. If you do work with pen and paper you could scan or take photos of your work, then convert to pdf.

Programming Part Q1 and Q2 For this part, you will need to submit the following files to Markus.

- `MNISTclassification.py`: the completed python file for Q1
- `Q2_LR_BGD.py`: the python file to solve Q2.
- `a2_prog_soln.pdf`: Answers to Q1 (a), (b) and (c), and Q2.

The GenAI Acknowledgement part is for you to document chat transcripts you have had with Generative AI chatbot (see Generative AI Policy section for details) if there is any. The file to be submitted is

- `GenAILog.pdf`

If you did not use Generative AI chatbots, put the following statement in the `GenAILog.pdf` file.

- I hereby declare that I did not use any Generative AI chatbot to complete this assignment.

Generative AI Policy

Asking general questions about concepts relevant to the assignment questions is permitted. You are not permitted to directly ask chatbots for hints or solutions of assignment questions. You should not feed the contents of the assignment into the chatbot and then copy or paraphrase the solution provided by the chatbot. For the programming part of the assignment, you must properly attribute any code generated by the chatbot. You need to include all chat transcripts (including the prompts) that have helped you to complete this assignment in the file named `GenAILog.pdf`.