CSCC11 Introduction to Machine Learning, Winter 2024

Assignment 2

This assignment makes use of material up to week 6 of the course. Like the first assignment, it comprises two parts, a written part and a coding part. The written work asks for you to derive some solutions to problems and for you to demonstrate an understanding of certain concepts. The written and programming parts will have different deadlines (see below). Submission instructions can be found at the end of the handout.

We remind you that the work you hand in for this assignment must be your own. Students should not make use of large language models for programming assistants like Copilot.

Theory Questions (Due Monday, March 4, 11:59pm)

Question 1. MAP and Bayes Estimates (10 marks)

In the class, we discussed parameter estimation for a Bernoulli random variable from N independent observations. In particular we estimated the probability θ that a coin toss will land heads-side up, given the outcomes of N coin tosses. Initially, we assumed a uniform prior distribution over potential value of θ. Now, suppose we have a non-uniform prior. For example, a commonly used prior for a probability on the unit interval $[0, 1]$ is the Beta distribution:

$$Beta(\theta \,|\, \alpha, \beta) \;=\; \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1} \;, \tag{1}$$

where α and β are the two parameters of the density function, and B is a constant which ensures the density integrates to 1. (The Beta distribution is known as a conjugate prior for the Bernoulli distribution, meaning that its comes from the same family of distributions as a posterior for θ. Conjugate priors are widely used as they have many convenient properties.)

a Setting $\alpha = \beta = 1$ in the Beta distribution, what function will we obtain? How about setting $\alpha = \beta = 2$? For both cases, find the maximum value of the probability density function and identify the location of this maximum.

b Suppose we observe N independent coin flips. Derive a mathematical expression for the posterior distribution over θ using the Beta prior.

c Derive a mathematical expression for the MAP estimate for θ.

d In class, we showed that the Bayes estimate with a uniform prior has some appealing properties. Compare two different estimators, namely, (i) θ_{MAP} with a Beta prior, and (ii) the Bayes estimate with a uniform prior. How do they differ? Is it possible to obtain θ from the Bayes estimate using a Beta prior?

e Discuss the differences in estimates of θ from (i) the MAP estimate with a uniform prior, (ii) the Bayes estimate with the uniform prior, and (iii) the MAP estimate with a Beta prior, when the number of observed coin tosses is small, versus when the number of coin tosses tends to infinity. In a couple of sentences, explain which one seems more useful?

Question 2. Decision Trees (10 marks)

In class we discussed how to used information gain to help determine which attribute (or feature) is the most useful for discriminating between two classes. In this question, we are looking to construct a decision tree using a simple example. Assume we need to suggest whether a person should to go to a Chinese restaurant or an Italian restaurant.

Based on your observations over the last two weeks you observe people with excellent taste in restaurants having gone to the Italian restaurant 12 times and the Chinese restaurant 18 times. During your observations, you have also noted several factors that appear relevant to the prediction, namely, (i) whether one goes to the reastaurant early in the week (Monday-Wednesday), later in the week (Thurs-Friday) or on the weekend, (ii) whether it is raining, and (iii) whether one is planning to go for lunch or dinner. The table below shows our observations, i.e., what conditions existed when people went to the two restaurants.

Observation	Resturant	Raining	Day	Meal
x1	Italian	Y	Mon-Wed	Lunch
x2	Italian	N	Thurs-Fri	Dinner
x3	Italian	N	Thurs-Fri	Dinner
x4	Italian	Y	Sat-Sun	Dinner
x5	Itlian	N	Mon-Wed	Dinner
x6	Chineese	Y	Mon-Wed	Dinner
x7	Chineese	N	Mon-Wed	Dinner
x8	Chineese	N	Mon-Wed	Dinner
x9	Chineese	N	Sat-Sun	Lunch
x10	Chineese	Y	Sat-Sun	Lunch
x11	Chineese	Y	Sat-Sun	Lunch
x12	Chineese	Y	Sat-Sun	Lunch
x13	Chineese	Y	Mon-Wed	Dinner
x14	Italian	Y	Sat-Sun	Dinner
x15	Italian	N	Mon-Wed	Dinner
x16	Italian	Y	Sat-Sun	Dinner
x17	Chineese	Y	Mon-Wed	Lunxh
x18	Chineese	N	Mon-Wed	Dinner
x19	Italian	N	Mon-Wed	Lunch
x20	Chineese	N	Mon-Wed	Lunch
x21	Chineese	Y	Mon-Wed	Lunch
x22	Chineese	N	Mon-Wed	Dinner
x23	Italian	N	Sat-Sun	Lucnh
x24	Italian	Y	Mon-Wed	Lunch
x25	Italian	N	Mon-Wed	Dinner
x26	Chineese	N	Thurs-Fri	Lunch
x27	Chineese	N	Thurs-Fri	Lunch
x28	Chineese	Y	Mon-Wed	Dinner
x29	Chineese	Y	Thurs-Fri	Dinner
x30	Chineese	N	Mon-Wed	Dinner

Table 1: Observation of 30 people (with good taste) going to restaurants.

a For the root node of the tree, compute the entropy for the distribution over our target variable (i.e., Restaurant) for which we aim to learn a predict tool.

b Explain why entropy (for discrete random variables) is always non-negative?

c How many different split functions are there for this data set? What are they? For each, compute the information gain at the root node.

d What split function would you use for the root of the decision tree?

e Draw (by hand if you like) the full decision tree that you would expect to learn.

Question 3. Logistic Regression (10 marks)

The main approach to fitting a logistic regression model involves maximizing the log-likelihood of the observed data with respect to the model parameters. This is typically achieved using optimization algorithms, such as gradient descent or its variants, which iteratively update the model parameters to minimize the loss function (i.e., the negative log-likelihood given in Equation (9.46) in the course notes).
As explained in Section 10 of the course notes, gradient descent works by iteratively updating the model parameters in the direction of the steepest descent of the loss function. The Newton-Raphson algorithm is another optimization method that can be used for logistic regression. Unlike gradient descent, which updates the parameters based on the gradient of the loss, the Newton-Raphson algorithm also uses the second derivatives

(the Hessian matrix) to determine the direction and step size for parameter updates. The basic steps of the Newton-Raphson algorithm for logistic regression are as follows:

1. Initialize parameters: Start with an initial guess for the model parameters.
2. Compute the gradient and Hessian: Calculate the gradient vector and the Hessian matrix of the negative log-likelihood function with respect to the model parameters.
3. Update parameters: Use the Newton-Raphson update rule to iteratively update the model parameters. The update rule is
$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} - (\mathbf{H}^{-1} \cdot \mathbf{g})$$
where,
   - $\boldsymbol{\theta}_{\text{new}}$ is the updated parameter vector.
   - $\boldsymbol{\theta}_{\text{old}}$ is the current parameter vector.
   - $\mathbf{H}$ is the Hessian matrix.
   - $\mathbf{g}$ is the gradient vector.

With this algorithm in mind, please answer the following questions:

a Using Equation (9.46), what is the range of negative log likelihood?

b Derive a mathematical expression for the Hessian matrix using Equation (9.46).

c Does the Hessian depend on the output (i.e., y in Equation (9.46)). If not, is it correct to claim that the curvature of log likelihood directly or indirectly does not depend on the output?

d The Hessain matrix reflects the curvature of the log-likelihood function in logistric regression. Discuss how adding the regulizer discussed in Section 9.6.1 (on Regularized Logistic Regression) results in a smoother objective function (ie, lower curvature).

Programming Questions (Due Thursday, March 7, 11:59pm)

The MNIST[1] dataset has been widely used in machine learning. It comprising images of handwritten digits, centered and resized to 28×28 pixels. Each image has been saved as a vector with the dimension of $784 = 28 \times 28$, 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 below.[2]
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 `XXX`. The completed python files do not need to be submitted.

Question 1. Classification using KNN and Random Forest (20 marks)

In this question, you will use the sklearn random forest classifier 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 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.

[1] short for Modified National Institute of Standards and Technology
[2] The entire dataset can be visualized at Tensorflow dataset website.

Figure 1: MNIST dataset

a In the first scenario, we're working with noiseless input images and labels. The aim is to explore hyper-parameter tuning for KNN classifiers and decision trees. For weighted KNN Classification we need to select the number of neighbors when 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 forests. 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 `MNISTclassification1.py`, please respond to the questions below.

- For KNN with distance 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.

b 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 `MNISTclassification2.py`.

- 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.

c 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 `MNISTclassification3.py`, 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.

Submission Instructions

Theory Questions (Due Monday, March 4, 11:59pm) 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 all 3 questions). The file should be called `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 Questions (Due Thursday, March 7, 11:59pm) For this part, you need to answer the questions for each section and provide the requested plot for each part. Your answers should be submitted electronically as a single PDF file (for all sections). The file should be named `solutioncoding.pdf` and submitted to **Markus**

Note that in Markus you find **separate sections** dedicated to theory and programming parts of this assignment.