CMPUT 366, Winter 2019
Assignment #3

Due: Monday, Mar. 25, 2019, 2:59pm
Total points: 120

Name:
Student number:
Collaborators:

1. **(Monte Carlo)**

   (a) **[30 points]** Implement rejection sampling in Python 3 by editing the `rejection_sampled_expectation` function in the provided `rejectionSampling.py`. The function should compute and return the expectation of the provided function `f` on the provided target distribution using rejection sampling. The `rejectionSampling.py` file contains an example `monte_carlo_expectation` function, and `main` function that will automatically test your implementation. You may edit the main file to perform any additional tests that you like.

   We will run your code by importing the `rejectionSampling` module and calling `rejection_sampled_expectation`, so it is important that your code follow these naming conventions.

   Submit all of your code for this question and question 2 (including provided boilerplate files) in a single zip file.

   (b) **[3 points]** Estimate the expected value of a Gaussian distribution with mean 1.75 and standard deviation 0.5, truncated below 0.75 and above 2.75. (This distribution is provided as `trunc`). Use your rejection sampling implementation to draw 2,000 samples from the target distribution. Use a Uniform[-3,3] distribution (the `U33` distribution) as a proposal distribution.

      i. What was the estimated expectation?

      ii. How many samples were rejected?

   (c) **[3 points]** Estimate the expected value of a Gaussian distribution with mean 1.75 and standard deviation 0.5, truncated below 0.75 and above 2.75. (This distribution is provided as `trunc`). Use your rejection sampling implementation to draw 2,000 samples from the target distribution. Use an untruncated Gaussian distribution with mean 2.75 and standard deviation 0.5 (the `untrunc` distribution) as a proposal distribution.

      i. What was the estimated expectation?

      ii. How many samples were rejected?

(d) **[10 points]** Were substantially more samples rejected using one proposal distribution than the other? If so, explain why. If not, explain why not.

(e) **[6 points]** Suppose you are evaluating a trained classifier using 0/1 loss on a test set of 1,000 examples. Your classifier's accuracy on the test set is 0.85 (i.e., 85%). Give an upper bound on the probability that the classifier's true accuracy (i.e., on the population from which the test set was drawn) is more than 5% different from 85%.

2. **(Neural Networks)** The MNIST dataset is a set of images of handwritten digits labelled by their actual digit. We will operate on two versions of this dataset: one with the images shifted 2 pixels to the upper-left, and one with the images shifted 2 pixels to the bottom-right.

   This question requires the use of TensorFlow. You may need to install Tensorflow using the following command:

   ```
   pip3 install tensorflow
   ```

   (a) **[10 points]** Implement a fully-connected feed-forward neural network for classifying MNIST images according to the digit that they represent by editing the `mlp2` function in the provided `cnn.py` file.

   The network should have two hidden layers: one with 128 rectified linear ('relu') units, and one with 64 rectified linear units. The output should be a fully-connected layer of 10 units with the softmax activation. The `cnn.py` file contains an example implementation of a network with a single hidden layer in the `mlp1` function that you may template from. The `main` function will test your program for you; you may add any tests that you like. It will also create a file called `examples.png` that contains examples images from the two test sets.

   The function should train the network using the training features `train_x` and labels `train_y`, and then evaluate the accuracy of the trained network on two different test sets: `test1_x`, `test1_y`, and `test2_x`, `test2_y`. This is demonstrated in `mlp1`.

   We will run your code by importing the `cnn` module and calling `mlp2`, so it is important that your code follow these naming conventions.

   Submit all of your code for this question and question 1 (including provided boilerplate files) in a single zip file.

   (b) **[30 points]** Implement a convolutional neural network for classifying MNIST images by editing the `cnn` function in the provided `cnn.py` file.

   The network should have the following architecture:

   - A layer of 32 convolutional units with a kernel size of 5×5 and a stride of $1, 1$.
   - A max-pooling layer with a pool size of 2×2 and a stride of $2, 2$.
   - A layer of 64 convolutional units with a kernel size of 5×5 and the default stride.
   - A max-pooling layer with a pool size of 2×2 and the default stride.
   - A `Flatten` layer (to reshape the image from a 2D matrix into a single long vector)
   - A layer of 512 fully-connected relu units
   - A layer of 10 fully-connected softmax units (the output layer)

   Submit all of your code for this question and question 1 (including provided boilerplate files) in a single zip file.

   (c) **[2 points]** What was the accuracy of your trained 2-hidden-layer feedforward network on the two test sets?

   (d) **[2 points]** What was the accuracy of your trained convolutional neural network on the two test sets?

   (e) **[10 points]** Did one of your implementations perform substantially better on one of the test sets than the other implementation did? If so, why? If not, why not?

3. **(Bayesian Learning)**

   Suppose that you have three models $(\theta_1, \theta_2, \theta_3)$ of changes in the price of a single stock. You know that one of these models is the true model. Each model gives the probability that tomorrow's price will be higher than today's price (y_{t+1}), based on whether today's price was higher than the price the day before (y_t). So you can make money on average by buying the stock when $p(y_{t+1} \mid y_t, \theta^*) > .5$ and selling when $p(y_{t+1} \mid y_t, \theta^*) < .5$, where θ^* is the true model.

   Your prior belief is that θ_1 and θ_2 are equally likely to be true, and θ_3 is three times more likely than θ_1 to be true.

   You have a dataset D of past observations, and you have computed that

$$p(D \mid \theta_1) = .00084$$
$$p(D \mid \theta_2) = .00105$$
$$p(D \mid \theta_3) = .00007.$$

4. **[6 points]** What are the posterior probabilities of each model being the true model?

5. **[4 points]** Now suppose that you have run each model, and they make the following predictions:

$$p(y_{t+1} \mid y_t, \theta_1) = .75$$
$$p(y_{t+1} \mid y_t, \theta_2) = .4$$
$$p(y_{t+1} \mid y_t, \theta_3) = .6.$$

   What is the maximum a posterior estimate for $p(y_{t+1} \mid y_t)$? Based on the MAP estimate, would you be better off buying or selling?

6. **[4 points]**

   What is the estimate according to the posterior predictive distribution for $p(y_{t+1} \mid y_t)$? (I.e., using model averaging.) Based on the PPD, would you be better off buying or selling?

Submission

Each assignment consists of a *problem set* file in PDF format containing the list of questions to answer, and a zipfile of *support files* containing stub code, support modules, and a LaTeX template for generating a PDF for your answers.

The intention is that you should be able to unzip the support files into an empty directory, work on the problems, and then zip the directory into a new file for submission.

Each assignment is to be submitted electronically via GradeScope before the due date. The submission should consist of a PDF file containing the written answers to the problem set, and a single zipfile containing all of your code. The zipfile includes a LaTeX file that you can edit to produce a PDF of your answers if you wish. Otherwise, you can type your answers into your favourite word processor and print to PDF, or you can write your answers (legibly!) by hand and upload a scan.