University of Toronto at Scarborough
Department of Computer and Mathematical Sciences

CSCC11: Introduction to Machine Learning

Winter 2021

Take-Home Open Book Final Exam

April 17, 03:00 PM – April 19, 03:00 PM

Preamble: The goal in this exam is to apply what you have learned. You are encouraged to use the online notes, your class notes, and you are encouraged to use your assignment code in solving the problems below. Your work should be your own. Do not post problems online or ask others for help.

Instructions: Download the starter file called `exam.tgz` on Quercus. Once downloaded and unpacked, a directory called `Exam` will exist. Please do not change the directory structure or the headers of functions in python files. All modified python files and others you generate (e.g., plots) should be left in the Exam directory (which you'll submit). Remember to read all instructions and comments in the python files very carefully.

Written work: For written answers you can either write on paper or use an editor that supports equations (e.g. Word or LaTeX). If writing on paper, please scan or photograph and save as a png file or perhaps pdf (and make sure it is easily legible). If formatting your answers in Word or LaTeX etc, please save as pdf. All such png or pdf files should be left in the Exam directory for submission.

Programming work: You'll use the same environment you've been using on assignments. You will need Python3, Numpy, Matplotlib, and Pickle; you **may not** import or use any other python packages. Please pre-install if working on your own computer. If you'd rather work on a UTSC machine, remote login to one of these lab machines:

1. IC lab (ranging from 01 to 50): i406-[01-50]
2. BV lab (ranging from 01 to 40): b473-[01-40]

with addresses `iits-<machine_to_use>.utsc-labs.utoronto.ca`. E.g. to access IC406 machine 1 from MACOS or linux, use `ssh -X <utorid>@iits-i406-01.utsc-labs.utoronto.ca`. Alternatively, you may access the Mathlab machine using `mathlab.utsc.utoronto.ca`.

NOTE: You are not required to vectorize your programs to earn full marks.

Submission: Submit your exam on the Quercus (like assignments). Do not modify the Exam directory tree or method/function headers therein.

For submission, all code, pdf files etc should be in a single directory called `Exam`. Tar up that directory into a file named `ExamSol_<utorid>.tar`; the command should be

```
tar -cf <name>.tar Exam/**/*.py Exam/**/*.txt Exam/**/*.png Exam/**/*.pdf
```

Before the actual submission, make sure that you did not tar up the datasets, can untar the file and see all the necessary files it should contain. To submit the compressed file onto Quercus, click on the `Assignments`, and then on `Final Exam`, or alternatively, you can find it on the `Final Exam` module located at the bottom of the home page.

Important Note: Submission must be received by 3 PM on April 19, 2020. We strongly suggest you upload well before 3 PM (especially if you require VPN connection to upload), and then, if you have modifications to your code or written solutions, you can resubmit as many times as you like on Quercus.

Questions during the exam? You may send questions **(privately)** through Piazza. We will respond to individuals or the entire class as necessary. We aim to log in at the following times: 8pm and 11pm Saturday; 9am, noon, 4pm, 8pm, and 11pm on Sunday; and 9am and noon on Monday.

Q1. Multivariate Polynomial Regression [9 marks]

Suppose you are given exchange rate data for different currencies that specify the relative prices of oil in different countries. And your task is to predict the price of oil in Canada relative to other countries. Specifically, you are given prices of oil for 21 countries in local currency relative to the U.S. dollar (oil exchange rates), along with the relative cost in Canada. You are given one data point about every two weeks over a 20 year period.

Your job is to learn the parameters of a polynomial regression model that makes prediction given the input vector. You will consider linear and quadratic polynomials and are asked to use cross-validation to determine whether to use a linear or quadratic function as the best model. Starter code is provided, but you are free to incorporate ideas or code fragments from any of your assignments in order to quickly find a good solution to the problem.

Data: In a directory called `Q1/data` you will find a file called `oil_500.pkl` which contains the data. This includes the input feature vectors, `X`, the oil prices of 21 countries taken at various random times, and `y`, the corresponding oil prices in Canada.

Tasks:

(a) Regression. Complete the code in `polynomial_regression.py` that one can use to fit a linear or a quadratic model of the input data `X` to predict the oil price in Canada.

(b) Final Analysis. In at most four sentences each, answer the following three questions in the file called `Questions.txt` in the `Q1` directory:

1. Between linear and quadratic regression models, which one best fits to the model?
2. How did you decide this was a good choice and why?
3. Would you try more complex models with higher degree of polynomial? Why?
4. How many parameters do we need to consider if we were to create a K degree polynomial regression model for D dimensional inputs?
5. If you know some of the features are highly correlated, what would you do to reduce the number of parameters?

Q2. Written Problems [23 marks]

In Q3, you will be using Gaussian Class Conditional (GCC) and Naive Bayes (NB) to classify images. To that end, we want to understand the effects of dimensionality and explore alternative algorithms/models. The written solutions should be saved as `ExamSol_<utorid>.pdf` in the `Exam` directory.

Curse of Dimensionality. Before you considered GCC and NB, you thought about using K-Nearest Neighbours. You recall that your amazing instructors and teaching assistants mentioned the "curse of dimensionality" multiple times. To figure out why, you came up with the following toy example. A hypersphere is a generalization of the concept of a sphere (not just 3 dimensional). Consider a d dimensional hypersphere with radius r. Its (hyper)volume is

$$V_d(r) = \begin{cases} \frac{\pi^k}{k!} r^d, & d = 2k \\ \frac{2(k!)(4\pi^k)}{(2k+1)!} r^d, & d = 2k+1 \end{cases}. \tag{1}$$

The fraction of its hypervolume lying between values $r - c$ and r, where $0 < c < r$ is given by

$$f = 1 - \left(1 - \frac{c}{r}\right)^d. \tag{2}$$

1. For any fixed c, f tends to 1 as $d \to \infty$. Show this numerically with $\frac{c}{r} = 0.01$, for $d = 2, 10, 500$.
2. For any fixed r, evaluate the fraction of the hypervolume which lies inside the radius $\frac{r}{2}$ for $d = 2, 10, 500$.

3. Suppose we sample points according to a uniform distribution inside a very high-dimensional hypersphere centered at the origin, where in the hypersphere would we most likely find the sampled points? Why is this a problem for some algorithms such as K-Means and K-Nearest Neighbours?

Naive Bayes. In Chapter 9 of the online course lecture notes, we introduced GCC and NB for classification. The generative model for a data point $\mathbf{x}$, in the two class case, is formulated as

$$p(\mathbf{x}) = \sum_{j=0}^{1} p(\mathbf{x} \,|\, c = j)\, p(c = j). \tag{3}$$

where c is the class random variable; it takes on the value 0 for one class, and 1 for the other. The decision function for the CC model is the log posterior ratio,

$$a(\mathbf{x}) = \log p(c = 1 \,|\, \mathbf{x}) - \log p(c = 0 \,|\, \mathbf{x}) \,. \tag{4}$$

To learn the model, given N data points $\{\mathbf{x}_i, c_i\}_{i=1}^N$, we estimate the class priors, $p(c = 0)$ and $p(c = 1)$, and likelihoods, $p(\mathbf{x}|c = 0)$ and $p(\mathbf{x}|c = 1)$. For GCC models the likelihoods are Gaussian,

$$p(\mathbf{x}|c = j) = G(\mathbf{x}; \mu_j, \Sigma_j) \,, \tag{5}$$

for which we estimate the means and covariances. In NB, we further assume that the features are conditionally independent of the class (i.e. $p(\mathbf{x}|c) = \prod_{i=1}^d P(x_i|c)$).

Consider the following problems in the context of binary classification:

1. Compute the prior class probability $\pi = p(c = 1)$ for the NB model using Maximum Likelihood (ML) estimate (i.e. $\pi_{ML} = \arg_\pi \max P(\{\mathbf{x}_i, c_i\}_{i=1}^N|\pi)$).
2. We had seen that ML is prone to overfitting. As a result we would want to use the Bayes' estimate instead. Let the prior of the prior class probability be distributed following Beta distribution (i.e. $\pi \sim \text{Beta}(\alpha = 1, \beta = 1)$). Compute the prior class probability π for the NB model using Bayes' estimate. How do α and β affect the prior class probability and reduce overfitting?
3. Compute the parameters of each Gaussian for the NB model using ML estimate.
4. Naive Bayes is useful when features are conditionally independent and it can also reduce the number of parameters. Suppose you know that only some features are conditionally independent, what can you do instead to model some correlations while reducing number of parameters? Explain how your approach reduces number of parameters.

Neural Networks. You recall again from your amazing instructors that neural networks are good for image data. Since you have never used deep learning libraries PyTorch, TensorFlow, Jax, etc., you decided to write your own implementation.

For simplicity, consider a binary classification task where you are given inputs with one feature (i.e. inputs $x \in \mathbb{R}$ with labels $t \in \{0, 1\}$). You decided to apply a fully connected network (MLP) with two hidden layers, each with two hidden units. The first hidden layer is followed by a non-linear activation function and the second hidden layer is followed by a sigmoid function (See Figure 1 (**Left**)). The prediction y is then fed into the loss function $\mathcal{L}$ along with the label t.

When we stack more and more layers, we run into a problem known as **vanishing gradient** where the gradient in earlier layers becomes very small. This may happen when we have activation function with small gradient in general (e.g. sigmoid, tanh, etc.). To account for this, we can sum the output of second hidden layer with input x right before applying the sigmoid. This addition is known as a **skip connection** that would preserve the gradient (See Figure 1 (**Right**)). Networks with these skip connections are known as residual networks.

For the following questions, we define the fully connected network as f and define the residual network as g:

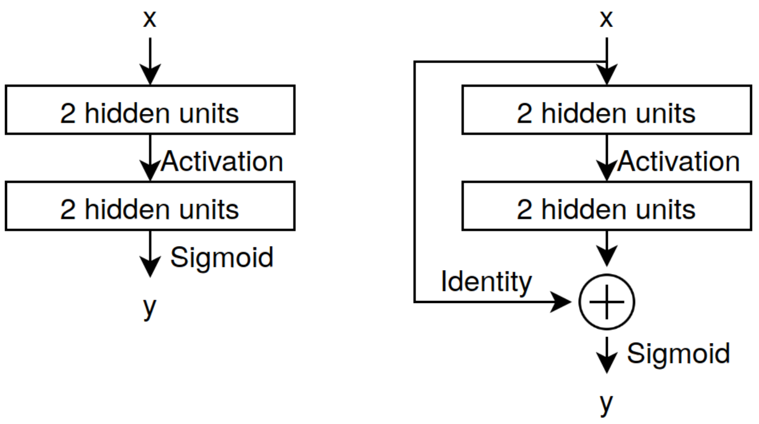

Figure 1: **Left:** Input x is fed into a fully connection network with two hidden layers, each with two hidden units. The first hidden layer's activation is any arbitrary non-linear function and the second hidden layer's activation is a sigmoid. The output of the sigmoid is the prediction y. **Right:** Input x is fed into a residual network. The network has two hidden layers, each with two hidden units. The first hidden layer's activation is any arbitrary non-linear function. The output of the second layer is summed with input x (i.e. a **skip connection**) right before being fed into a sigmoid. The output of the sigmoid is the prediction y.

1. Draw the computational graphs with networks f and g. For simplicity, ignore the bias term. You may also find it helpful to denote the j'th activation in the i'th layer as $\sigma_j^{(i)}$, the linear combination between i'th layer inputs and parameters of the j'th hidden unit as $h_j^{(i)}$, and the k'th parameter in the j'th hidden unit in the i'th as $w_{jk}^{(i)}$.
2. Let t be the label, y be the prediction, and $\mathcal{L}(y,t)$ be a loss function. Compute the derivative of $\mathcal{L}$ with respect to x using f and g. For simplicity, you may write the gradient in terms of partial derivatives.
3. Using the derivatives above, provide an example where f experiences **vanishing gradient** (i.e. $\frac{\partial \mathcal{L}}{\partial x} = 0$) whereas g does not suffer from this problem. **Hint:** Define the activation in the first hidden layer.

Q3. Classification with Gaussian Class Conditionals and Naive Bayes [18 marks]

Suppose you are now given 25×20 grayscale image data (See Figure 2) and your goal is to predict whether the image contains an eye. Your job is to learn the parameters of a Gaussian Class Conditional (GCC) model and a Naive Bayes (NB) model that makes a binary prediction given an image.

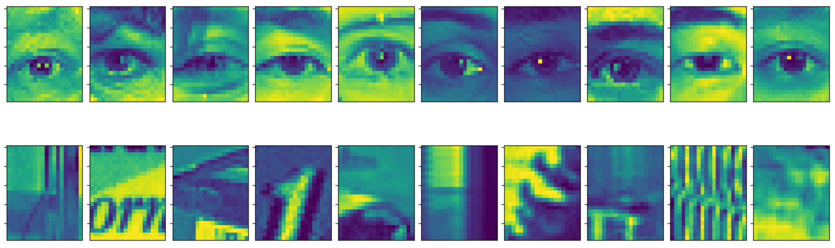

Figure 2: A subset of images from the eye dataset. The top rows are examples of eye images and the bottom rows are examples of non-eye images.

You supervisor suggests that you consider PCA for dimensionality reduction since images are high dimensional and are highly correlated between pixels. She suggests that you use the PCA subspace coefficients (a low-dimensional representation) as the inputs to the classification model. As she explains, PCA also provides a form of regularization as it limits the complexity of your model.

Your job is to learn the parameters of the classification models that make prediction given a vector of subspace coefficients. You will consider GCC and NB models with different numbers of subspace dimensions. You are asked to use cross-validation to determine a) the dimension of the subspace, and b) whether to use GCC or NB of the subspace coefficients. Starter code is again provided, but you are free to incorporate ideas or code fragments from any of your assignments in order to quickly find a good solution to the problem.

Data: In a directory called `Q3/data` you will find a file called `eye_image_data.pkl` which contains the data. This includes the input feature vectors, `X`, the images taken at various random times, and `y`, the corresponding label.

Tasks:

(a) Gaussian Class Conditionals. You should implement the GCC model by completing the methods `train` and `predict` in `gcc.py` :

1. `train` fits the GCC model using the Maximum Likelihood estimates. The means and covariances for the likelihoods should be stored in `self.means` and `self.covariances`. The learned priors are stored in `self.priors`.
2. `predict` takes inputs and computes the probability of each input being assigned to each class.

(b) Naive Bayes. You should implement the NB model by completing the method `train` in `nb.py`. Similar to GCC, `train` fits the NB model using the Maximum Likelihood estimates. The means and covariances for the likelihoods should be stored in `self.means` and `self.covariances`. The learned priors are stored in `self.priors`.

(c) Sanity Test. You can test your code by running `debug_model.py`. It provides simple test cases and feedback to help debug your code.

(d) Cross Validation. To determine the best model, we need to decide whether to use GCC or NB. And we need to choose the subspace dimension (i.e. how many principal directions to use). To this end, complete the code in `cross_validation.py`, to split the dataset into training and validation sets, and to compute cross-validation (CV) score.
Then complete the code in `model_selection.py` to find the CV score over all the models being considered. Your code should produce plots of the CV score for the GCC model as a function of the subspace dimension, and for the NB model as a function of the subspace dimension. There should also be plots that contain the reconstructed images using dimensionality chosen from the best subspace dimensionality for each model. The terminal will output the best model from the set the GCC and NB models tested. Finally, save **only** the plots for the CV scores as `gcc_scores.png` and `nb_scores.png` respectively in the `Q3` directory.

(e) Final Analysis. In at most four sentences each, answer the following three questions in the file called `Questions.txt` in the `Q3` directory:

1. How many Principal Components do you think is needed to give the best fit to each of the model?
2. Would you choose Naive Bayes or Gaussian Class Conditionals?
3. How did you decide the above were good choices and why?
4. Looking at the training and validation scores, they collapse into constant functions as we increase the dimensionality. Can you explain why this is happening?
5. Looking at the reconstructed images using the best dimensionality of each model, do you think you can predict whether a given image is an eye easily? Why?
6. Why would dimensionality reduction help produce a better model?