CPSC 440/540 Machine Learning (January-April, 2022)
Assignment 1 (due Friday January 21st at midnight)

IMPORTANT!!!!! Please carefully read the submission instructions that will be posted on Piazza. We will deduct 50% on assignments that do not follow the instructions.

Most of the questions below are related to topics covered in CPSC 340, or other courses listed on the prerequisite form. There are several "notes" available on the webpage which can help with some relevant background.

If you find this assignment to be difficult overall, that is an early warning sign that you may not be prepared to take CPSC 440 at this time. Future assignments may be longer and more difficult than this one.

We use blue to highlight the deliverables that you must answer/do/submit with the assignment.

Basic Information

1. Name:
2. Student ID:

1 Very-Short Answer Questions

Give a short and concise 1-2 sentence answer to the below questions. All acronyms, symbols, and methods are as used in CPSC 340.

1. Suppose that a famous machine learing personality is advertising their "extremely-deep convolutional fuzzy-genetic Hilbert-long-short adversarial-recurrent neural network" classifier, which has 500 hyper-parameters. This person claims that if you take 10 different famous datasets, and tune the 500 hyper-parameters based on each dataset's validation set, that you can beat the current best-known validation set error on all 10 datasets. Explain whether or not this amazing claim is likely to be meaningful.

2. Consider fitting a neural network with one hidden layer. Would we call this a parameteric or a non-parametric model if the hidden layer had n units (where n is the number of training examples)? Would it be parametric or non-parametric if it had d^2 units (where d is the number of input features).

3. Suppose we want to solve the binary classification problem of determining whether 26 by 26 images were images of digits or images of letters. What is an easy way to make our classifier invariant to small translations of the training images?

4. Ensemble methods are models that combine the predictions of multiple individual models. Give a reason why an ensemble method could do better than the best individual model in the ensemble.

5. A common method for outlier detection is to collect a large number of outliers, and build a binary classifier that distinguishes these outliers from standard "inlier" datapoints. What is an advantage and a drawback of this approach to outlier detection, compared to unsupervised outlier detection methods?

6. Why do we typically add a column of 1 values to X when we do linear regression?

7. If a function is convex, what does that say about stationary points of the function? Does convexity imply that a stationary points exists?

8. What is the cost in $O()$ notation for fitting a linear regression model with n examples and 1 feature using least squares? Explain under what conditions it would make sense to use gradient descent instead of the normal equations for fitting a linear regression model with 1 feature.

9. Suppose you are hired by company to look at a dataset, and they want to "find which features are relevant" for their prediction task. What are 2 warnings you would give them regarding the features found by feature selection methods?

10. If we fit a linear regression model with L1-regularization of the parameters, what is the effect of the regularization parameter λ on the sparsity level of the solution? What is the effect of λ on the two parts of the fundamental trade-off (the training error and the amount of overfitting)?

11. Suppose we fit a one-feature linear regression model with a polynomial basis, and this leads to non-zero regression weights. What would the prediction of this model be as the feature value x^i goes to ∞? How would this change if you used Gaussian RBFs as features?

12. When searching for a good w for a linear classifier, why do we use the logistic loss instead of just minimizing the number of classification errors?

13. With stochastic gradient descent, the loss might go up or down each time the parameters are updated. However, we don't actually know which of these cases occurred. Explain why it doesn't make sense to check whether the loss went up/down after each update.

14. Suppose we are given two vectors of integers, x=$(x_1, x_2, \ldots, x_n)$ and y=$(y_1, y_2, \ldots, y_n)$, and we want to find the longest sequence of numbers that appears consecutively in both x and y. Describe an $O(n^2)$ algorithm to do this using dynamic programming (you can assume the numbers have a fixed-sized representation).

15. Given a vector of n positive integers $(x_1, x_2, \ldots, x_n)$, give an $O(n)$ time algorithm for computing, for each position i, the sum of all numbers except x_i. Next, describe how you could solve this problem in $O(n)$ even if you were not allowed to use subtraction/negation (hint: you can start at the beginning or the end of the list).

16. Consider using a fully-connected neural network for 3-class classification on a problem with $d = 10$. If the network has one hidden layer of size $k = 100$, how many parameters (including biases), does the network have?

2 Coding Questions

The questions below use code contained in *a1.zip*, which you can download from the course webpage. The scripts mentioned in the question can be run in Julia's REPL in the directory containing the extracted files. If you have not previously used Julia, there is a list of useful Julia commands (and syntax) among the list of notes on the course webpage.

Several scripts use packages. For example, the *JLD* package is used to load data and the *PyPlot* package to make plots. If you have not previously used these packages, they can be installed using:

```
using Pkg
Pkg.add("JLD")
Pkg.add("PyPlot")
```

2.1 K-Means Clustering

If you run the function *example_Kmeans*, it will load a dataset with two features and a very obvious clustering structure. It will then apply the k-means algorithm with a random initialization. The result of applying the algorithm will thus depend on the randomization, but a typical run might look like this:
(Note that the colours are arbitrary due to the label switching problem.) But the "correct" clustering (that was used to make the data) is something more like this:

If you run the demo several times, it will find different clusterings. To select among clusterings for a *fixed* value of k, one strategy is to minimize the sum of squared distances between examples x^i and their means w_{y^i},

$$f(w_1, w_2, \dots, w_k, y^1, y^2, \dots, y^n) = \sum_{i=1}^{n} \|x^i - w_{y^i}\|_2^2 = \sum_{i=1}^{n}\sum_{j=1}^{d}(x_j^i - w_{y^i j})^2.$$

where y^i is the index of the closest mean to x^i. This is a natural criterion because the steps of k-means alternately optimize this objective function in terms of the w_c and the y^i values.

1. Write a new function called *kMeansError* that takes in a dataset X, a set of cluster assignments y, and a set of cluster means W, and computes this objective function. Hand in your code.
2. Instead of printing the number of labels that change on each iteration, what trend do you observe if you print the value of *kMeansError* after each iteration of the k-means algorithm?
3. Using the *clustering2Dplot* file, output the clustering obtained by running k-means 50 times (with $k = 4$) and taking the one with the lowest error. Note that the k-means training function will run much faster if you set `doPlot = false` or just remove this argument.
4. Explain why the *kMeansError* function should not be used to choose k.
5. Explain why even evaluating the *kMeansError* function on test data still wouldn't be a suitable approach to choosing k.

6. The data in *clusterData2.jld* is the exact same as *clusterData.jld*, except it has 4 outliers that are far away from the data. Using the *clustering2Dplot* function, output the clustering obtained by running k-means 50 times (with $k = 4$) on *clusterData2.jld* and taking the one with the lowest error. Are you satisfied with the result?

7. Implement the *k-medians* algorithm, which assigns examples to the nearest w_c in the L1-norm and then updates the w_c by setting them to the "median" of the points assigned to the cluster (we define the d-dimensional median as the concatenation of the medians along each dimension). For this algorithm it makes sense to use the L1-norm version of the error (where y_i now represents the closest median in the L1-norm),

$$f(w_1, w_2, \ldots, w_k, y_1, y_2, \ldots, y_n) = \sum_{i=1}^{n} \|x_i - w_{y_i}\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{d} |x_{ij} - w_{y_i j}|,$$

   Hand in your code and plot obtained by taking the clustering with the lowest L1-norm after using 50 random initializations for $k = 4$.

2.2 Regularization and Hyper-Parameter Tuning

If you run the script *example_nonLinear* (from Julia's REPL), it will:

1. Load a one-dimensional regression dataset.
2. Fit a least-squares linear regression model.
3. Draw a figure showing the training/testing data and what the model looks like.

Unfortunately, this is not a great model of the data, and the figure shows that a linear model with a y-intercept of 0 is probably not suitable.

1. Update the *leastSquares* function so that it includes a *y-intercept* variable w_0, and makes predictions using the model
$$y^i = w^T x^i + w_0.$$
Hand in your new function and the updated plot.

2. Allowing a non-zero y-intercept improves the prediction substantially, but the model still seems sub-optimal because there are obvious non-linear effects. Write a function called *leastSquaresRBFL2* that implements *least squares using Gaussian radial basis functions (RBFs) and L2-regularization.*
You should start from the *leastSquares* function and use the same conventions: n refers to the number of training examples, d refers to the number of features, X refers to the data matrix, and y refers to the targets. Use Z as the data matrix after the change of basis, σ as the standard deviation in the Gaussian, and λ as the regularization parameter. Note that you'll have to add two additional input arguments (λ for the regularization parameter and σ for the Gaussian RBF variance) compared to the *leastSquares* function. To make your code easier to understand/debug, you may want to define a new function *rbfBasis* which computes the Gaussian RBFs for a given training set, testing set, and σ value.
Hand in your function and the plot generated with $\lambda = 1$ and $\sigma = 1$.
Note: the *distancesSquared* function in *misc.jl* is a vectorized way to quickly compute the squared Euclidean distance between all pairs of rows in two matrices.

3. Modify the script to split the training data into a "train" and "validation" set (you can use half the examples for training and half for validation), and use these to select λ and σ. Hand in your modified script and the plot you obtain with the best values of λ and σ.

4. Consider a model combining the first two parts of this question,

$$y^i = w^T x^i + w_0 + v^T z^i,$$

where z^i is the Gaussian RBFs and v is a vector of regression weights for those features. Suppose that we first fit w and w_0 assuming that $v = 0$ (Part 1 of this question), and then we fit v with w and w_0 fixed (you could use your code for Part 2 to do this by modifying the y^i values). Why would someone want to do this?
Hint: think about how this model would behave with $x^i = 10$.

2.3 Multi-Class Logistic Regression

The script *example_multiClass.jl* loads a multi-class classification dataset and fits a "one-vs-all" logistic regression classifier using the *findMin* gradient descent implementation, then reports the validation error and shows a plot of the data/classifier. The performance on the validation set is ok, but could be much better. For example, this classifier never predicts some of the classes.

Using a one-vs-all classifier hurts performance because the classifiers are fit independently, so there is no attempt to calibrate the columns of the matrix W. An alternative to this independent model is to use the softmax probability,

$$p(y^i|W, x^i) = \frac{\exp(w_{y^i}^\top x^i)}{\sum_{c=1}^k \exp(w_c^\top x^i)}.$$

Here c is a possible label and w_c is column c of W. Similarly, y^i is the training label, w_{y^i} is column y^i of W. The loss function corresponding to the negative logarithm of the softmax probability is given by

$$f(W) = \sum_{i=1}^n \left[-w_{y^i}^\top x^i + \log\left(\sum_{c'=1}^k \exp(w_{c'}^\top x^i) \right) \right].$$

Make a new function, *softmaxClassifier*, which fits W using the softmax loss from the previous section instead of fitting k independent classifiers. Hand in the code and report the validation error.

Hint: you can use the *derivativeCheck* option when calling *findMin* to help you debug the gradient of the softmax loss. Also, note that the *findMin* function treats the parameters as a vector (you may want to use *reshape* when writing the softmax objective).

3 Calculation Questions

3.1 Matrix Notation, Quadratics, Convexity, and Gradients

This question reviews several comptutional tools covered in CPSC 340. You may also find some of the notes on the course webpage useful.

1. Consider the function
$$f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j + \sum_{i=1}^{n} b_ix_i + c,$$
where x is a vector of length n with elements x_i, b is a vector of length n with elements b_i, and A is an $n \times n$ matrix with elements a_{ij} (not necessarilyn symmetric). Write this function in matrix notation (so it uses A and b, and does not have summations or references to indices i.

2. Write the gradient of f from the previous question in matrix notation.

3. Give a linear system whose solution gives a minimum value of f in terms of x, in the case where A is symmetric and positive semi-definite (which implies that f is convex).

4. Consider weighted linear regression with an L2-regularizer with regularization strength $1/\sigma^2$,

$$f(w) = \frac{1}{2}\sum_{i=1}^{n} v^i(y^i - w^T x^i)^2 + \frac{1}{2\sigma^2}\sum_{j=1}^{d} w_j^2,$$

where $\sigma > 0$ and we have a vector v of length n containing the elements v^i. The other sybmbols are our standard regression notation. Write this function in matrix notation.
Note: you and use X as the matrix containing x^i as the rows, y as the vector containing the elements y^i, and V as a diagonal matrix containing the vector v along the diagonal.

5. Assuming that we have $v^i \geq 0$ for all i, show that f from the previous part is convex.

6. Assuming that we have $v^i \geq 0$ for all i, give a linear system whose solution gives a minimum value of f in terms of w.

7. If we fit a linear classifer with the exponential loss (used in older boosting algorithms), it would take the form
$$f(w) = \sum_{i=1}^{n} \exp(-y^i w^T x^i).$$
Derive the gradient of this loss function.

8. The support vector regression objective function is
$$f(w) = \sum_{i=1}^{n} \max\{0, |w^T x^i - y^i| - \epsilon\} + \frac{\lambda}{2}\|w\|^2.$$
where ϵ is a non-negative hyper-parameter. It is similar to the L1 robust regression loss, but where there is no penalty for being within ϵ of the prediction (which can reduce overfitting).Show that this non-differentiable function is convex.

3.2 MAP Estimation

In 340, we showed that under the assumptions of a Gaussian likelihood and Gaussian prior,

$$y^i \sim \mathcal{N}(w^\top x^i, 1), \quad w_j \sim \mathcal{N}\left(0, \frac{1}{\lambda}\right),$$

that the MAP estimate is equivalent to solving the L2-regularized least squares problem

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w^\top x^i - y^i)^2 + \frac{\lambda}{2}\sum_{j=1}^{d} w_j^2,$$

in the "loss plus regularizer" framework. For each of the alternate assumptions below, write it in the "loss plus regularizer" framework (simplifying as much as possible):

1. Gaussian likelihood with a separate variance σ_i^2 for each training example, and Laplace prior with a separate variance $1/\lambda_j$ for each variable,

$$y^i \sim \mathcal{N}(w^T x^i, \sigma_i^2), \quad w_j \sim \mathcal{L}\left(0, \frac{1}{\lambda_j}\right).$$

2. Robust student-t likelihood and Gaussian prior centered at u.

$$p(y^i | x^i, w) = \frac{1}{\sqrt{\nu} B\left(\frac{1}{2}, \frac{\nu}{2}\right)}\left(1 + \frac{(w^T x^i - y^i)^2}{\nu}\right)^{-\frac{\nu+1}{2}}, \quad w_j \sim \mathcal{N}\left(u_j, \frac{1}{\lambda}\right),$$

   where u is $d \times 1$, B is the "Beta" function, and the parameter ν is called the "degrees of freedom".[1]

3. We use a Poisson-distributed likelihood (for the case where y_i represents counts), and we use a uniform prior for some constant κ,

$$p(y^i | x^i, w) = \frac{\exp(y^i w^\top x^i)\exp(-\exp(w^\top x^i))}{y^i!}, \quad p(w_j) \propto \kappa.$$

   (This prior is "improper" since $w \in \mathbb{R}^d$ but it doesn't integrate to 1 over this domain.)

[1]This likelihood is more robust than the Laplace likelihood, but leads to a non-convex objective.

3.3 Machine Learning Model Memory and Time Complexities

Answer the following questions using big-O notation, and a brief explanation. Your answers may involve n, d, and perhaps additional quantities defined in the question. As an example, the (linear) least squares model has $O(d)$ parameters, requires $O(d)$ time for prediction, and requires $O(nd^2 + d^3)$ time to train.[2]

1. What is the storage space required for the k-means clustering algorithm?
2. What is the cost of clustering t examples using an already-trained k-means model?
3. What is the training time for linear regression with L2-regularization?
4. What is the prediction cost for linear regression with L2-regularization on t test examples?

[2]In this course, we assume matrix operations have the "textbook" cost where the operations are implemented in a straightforward way with "for" loops. For example, we'll assume that multiplying two $n \times n$ matrices or computing a matrix inverse simply costs $O(n^3)$, rather than the $O(n^\omega)$ where ω is closer to 2 as discussed in CS algorithm courses.

5. What is the storage space required for the model, after training a linear regression model with Gaussian RBFs as features?
6. What is the prediction time for linear regression with Gaussian RBFs as features on t test examples? You can use σ^2 as the variance of the Gaussian RBFs.
7. What is the cost of evaluating the support vector regression objective function?
8. What is the cost of trying to minimize the exponential loss regression loss by running t iterations of gradient descent?
9. What is the cost of trying to minimize the exponential loss by running t iterations of stochastic gradient descent?

Hint: many people got very-low grades on this question last year. If you are not sure how to answer questions like this, get help!