CPSC 540 Machine Learning (January-April, 2020)
Assignment 1 (due January 17th at midnight)

IMPORTANT!!!!! Please carefully read the submission instructions 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 540 at this time. Future assignments will 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:
3. Have you submitted the prereq form?

1 Very-Short Answer Questions

Give a short and concise 1-2 sentence answer to the below questions. All acronyms and methods are as used in CPSC 340/532M.

1. A common pre-processing operation is to *standardize* our features. This operation replaces each x_j^i with $(x_j^i - \mu_j)/\sigma_j$, where μ_j is the mean of feature j on the training data and σ_j is the standard deviation of feature j on the training data. When doing predictions with the model, some codes standardize the test features $\tilde{x}_j^i$ with a mean $\tilde{\mu}_j$ and standard deviation $\tilde{\sigma}_j$ computing based on the test data. Describe why this usually is a mistake, and describe a setting where it can lead to terrible performance even if the training and test data come IID from the same distribution.
2. What is the difference between a test set error and the test error?
3. In the deep learning world, *neural architecture search* describes the process of searching for the hyper-parameters of a neural network model (like the number of layers, the step size, whether to use convolutions, the regularization parameters, and so on) in order to optimize the performance on a fixed validation set. What is a potential problem with this approach?
4. In a parametric model, what is the effect of the number of features d that our model uses on the training error and on the approximation error (the difference between the training error and test error)?
5. Give a way to set the depth and width of a neural network that makes the model parametric, and a choice that makes the model non-parametric.

6. You can fit a decision stump by finding the stump that maximizes the number of times $\hat{y}^i = y^i$ in the training data. Why do we not try to maximize this quantity when we fit the regression weights w in linear regression models?

7. How does λ in an L1-regularizer affect the sparsity pattern of the solution (number of w_j set to exactly 0), the training error, and the approximation error?

8. Minimizing the squared error used by in k-means clustering is NP-hard. What would be the significance of an algorithm that minimizes this error that runs in time $O(nd^2 + d^3)$.

9. Consider the regression objective given by $f(w) = \sum_{i=1}^{n} \max\{0, |y^i - w^T x^i| - \epsilon\} + \frac{\lambda}{2}\|w\|^2$ for some number ϵ. Describe a method to minimize this objective function (or closely approximate the minimum).

10. Consider an ensemble clustering method that generates m different bootstraps of the data. It then fits a k-means model (with a random initialization) to each of the boostraps. To form the final clustering for example x_i, it chooses the y_i that is most common across the m clusterings. Would this be an effective or an ineffective ensemble method? (Briefly explain.)

11. What does minimizing $f(w) = ||Xw - y||_1 + \lambda||w||^2$ correspond to in the MLE/MAP view?

12. If you perform MLE with a Gaussian likelihood, would the coefficient $\hat{w}$ change if you standardized the features? Would the predictions $\hat{y}^i$ change? Which of these would change if you perform MAP estimation with a Gaussian likelihood and Gaussian prior.

13. Do you expect the resulting loss from running NMF to be lower, higher, or the same as the loss from running PCA on the same data? Briefly justify your answer.

14. Suppose we need to multiply a huge number of matrices to compute a product like $A_1A_2A_3 \cdots A_k$. The matrices have wildly-different sizes so the order of multiplication will affect the runtime. For example, $A_1(A_2A_3)$ may be faster to compute than $(A_1A_2)A_3$. Describe (at a high level) an $O(k^3)$-time algorithm that finds the lowest-cost order to multiply the matrices.

15. You have a supervised learning dataset $\{X, y\}$. You fit a 1-hidden-layer neural network using stochastic gradient descent to minimize the squared error, that makes predictions of the form $\hat{y}^i = v^\top W x^i$ where W and v are the parameters. Explain why or why not this neural network can achieve a better training accuracy than the basic linear regression model $\hat{y}^i = w^\top x^i$.

2 Coding Questions

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.

2.1 Regularization and Hyper-Parameter Tuning

Download *a1.zip* from the course webpage, and start Julia (latest version) in a directory containing the extracted files. 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. Report the test set error.
4. Draw a figure showing the training/testing data and what the model looks like.

This script uses the *JLD* package to load the data and the *PyPlot* package to make the plot. If you have not previously used these packages, they can be installed using:

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

Unfortunately, this is not a great model of the data, and the figure shows that a linear model is probably not suitable.

1. 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, y refers to the targets, Z refers to the data matrix after the change of basis, and so on. 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.

2. 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.

3. There are reasons why this dataset is particularly well-suited to Gaussian RBFs are that (i) the period of the oscillations stays constant and (ii) we have evenly sampled the training data across its domain. If either of these assumptions are violated, the performance with our Gaussian RBFs might be much worse. Consider a scenario where either (i) or (ii) is violated, and describe a way that you could address this problem.

2.2 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 even 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).

2.3 Principal Component Analysis

The script *example_PCA* will load a dataset containing 50 examples, each representing an animal. The 85 features are traits of these animals. The script gives two unsatisfying visualizations of it. First it shows a plot of the matrix entries, which has too much information and thus gives little insight into the relationships between the animals. Next it shows a scatterplot based on two random features and displays the name of 10 randomly-chosen animals. Because of the binary features even a scatterplot matrix shows us almost nothing about the data.

The function *PCA* applies the classic PCA method (orthogonal bases via SVD) for a given k. Using this function, modify the demo so that the scatterplot uses the latent features z_i from the PCA model with $k = 2$. Make a scatterplot of the two columns in Z, and use the *annotate* function to label a bunch of the points in the scatterplot.

1. Hand in your modified demo and the scatterplot.
2. Which trait of the animals has the largest influence (absolute value) on the first principal component? (Make sure not to forget the "+1" when looking for the name of the trait in the *dataTable*).
3. Which trait of the animals has the largest influence (absolute value) on the second principal component?
4. How much of the variance in X (after centering) is explained by our two-dimensional representation from the previous question?
   Note: you can compute the Frobenius norm of a matrix using the function *norm*. Also, note that the "variance explained" formula from CPSC 340 assumes that X is already centered.
5. How many PCs are required to explain 50% of the variance in the data?

3 Calculation Questions

3.1 Minimizing Strictly-Convex Quadratic Functions

Solve for the minimizer w of the below strictly-convex quadratic functions:

1. $f(w) = \frac{1}{2}w^\top \Lambda w + u^\top w + \lambda$ (general quadratic).
2. $f(w) = \frac{1}{2}(Xw - y)^\top \Sigma^{-1}(Xw - y) + \frac{\lambda}{2}\|w\|^2$ (L2-regularized least squares with weight covariance Σ).
3. $f(w) = \frac{1}{2}\sum_{i=1}^n v_i(w^\top x^i - y^i)^2 + \frac{\lambda}{2}\|w - u\|^2$ (weighted least squares shrunk towards u).

Above we use our usual supervised learning notation. In addition, we assume that u is $d \times 1$ and v is $n \times 1$, λ is a positive scalar, and Σ and Λ are symmetric positive-definite matrices. You can use V as a diagonal matrix with v along the diagonal (with the v_i non-negative). You can use I as an identity matrix. Hint: positive-definite matrices are invertible.

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. Laplace likelihood (with a scale of 1) for each training example and Laplace prior centered at u with scale $1/\lambda$.
$$y^i \sim \mathcal{L}(w^\top x^i, 1), \quad w_j \sim \mathcal{L}(u, 1/\lambda),$$
where u is $d \times 1$.

2. Gaussian likelihood with a separate variance σ_i^2 for each training example, and Gaussian prior with a separate variance $1/\lambda_j$ for each variable,
$$y^i \sim \mathcal{N}(w^T x^i, \sigma_i^2), \quad w_j \sim \mathcal{N}\left(0, \frac{1}{\lambda_j}\right).$$

3. Time-independent censored survival analysis likelihood with a uniform prior,
$$p(y^i, v^i | x^i, w) = \exp(v^i w^T x^i) \exp(-y^i \exp(w^T x^i)), \quad p(w_j) \propto \kappa$$
Here, y^i is a positive number giving the latest time that we observed patient i, and $v^i = 1$ if patient i has quit the study while $v^i = 0$ if they are still in it.[1]

For this question, you do not need to convert to matrix notation.

3.3 Machine Learning Model Memory and Time Complexities

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

1. What is the storage space required for a naive Bayes classifier with binary features and k class labels?
2. What is the training time required for k-means clustering? You can use t as the number of iterations it takes to converge.
3. What is the training time for linear regression with Gaussian RBF features? You can use σ^2 as the variance of the Gaussian RBFs.
4. What is the storage space required for an L2-regularized linear regression model with a polynomial basis, where we have used the kernel trick? You can use λ as the regularization parameter and p as the degree of the polynomial.
5. What is the cost of trying to minimize the logistic regression loss by running t iterations of stochastic gradient descent?
6. What is the cost of forward propagation (computing one value $\hat{y}_i$) in a neural network (for regression) with 3 fully-connected hidden layers? Use k_1 as the number of hidden units in layer 1, k_2 as the number of hidden units in layer 2, and k_3 as the number of hidden units in layer 3.

3.4 Gradients and Hessians in Matrix Notation

Express the gradient $\nabla f(w)$ and Hessian $\nabla^2 f(w)$ of the following functions in matrix notation, simplifying as much as possible:

[1] This likelihood can be used in regression settings to estimate survival times when some patients are still alive.

[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.

1. The quadratic function
$$f(w) = w^T u + u^T A w + \frac{\lambda}{2} w^T w + w^T A w,$$
wher u is $d \times 1$ and A is $d \times d$ (not necessarily symmetric).

2. L2-regularized weighted least squares with non-Euclidean quadratic regularizaiton,
$$f(w) = \frac{1}{2} \sum_{i=1}^{n} v_i (w^\top x^i - y^i)^2 + \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} w_i w_j \lambda_{ij}$$
where you can use V as a matrix with the v_i along the diagonal and Λ as a positive-definite $d \times d$ (symmetric) matrix with λ_{ij} in position (i, j).

3. Weighted L2-regularized probit regression,
$$f(w) = -\sum_{i=1}^{n} \log p(y^i | x^i w) + \frac{1}{2} \sum_{j=1}^{d} u_j w_j^2.$$
where u is $d \times 1$, $y^i \in \{-1, +1\}$, and the likelihood of a single example i is given by
$$p(y^i | x^i, w) = \Phi(y^i w^T x^i).$$
where Φ is the cumulative distribution function (CDF) of the standard normal distribution.

Hint: You can use the results from the linear and quadratic gradients and Hessians notes to simplify the derivations. You can use 0 to represent the zero vector or a matrix of zeroes and I to denote the identity matrix. It will help to convert the second question to matrix notation first. For the last question, it is useful to define a vector c containing the CDF $\Phi(y^i w^T x^i)$ as element c_i and a vector p containing the corresponding PDF as element p_i. For the probit question you'll need to define new vectors to express the gradient and Hessian in matrix notation (and remember the relationship between the PDF and CDF). As a sanity check, make sure that your results have the right dimension.

3.5 Norm Inequalities

Show that the following inequalities hold for vectors $w \in \mathbb{R}^d$, $u \in \mathbb{R}^d$, and $X \in \mathbb{R}^{n \times d}$:

1. $\|w\|_\infty \leq \|w\|_2 \leq \|w\|_1$ (relationship between decreasing p-norms)
2. $\|w\|_1 \leq \sqrt{d}\|w\|_2 \leq d\|w\|_\infty$ (relationship between increasing p-norms)
3. $\sqrt{m}\|w\|_2 \leq \|w\|_H \leq \sqrt{M}\|w\|_2$ (relationship between quadratic norm and Euclidean norm).

You should use the definitions of the norms, but should not use the known equivalences between these norms (since these are the things you are trying to prove). Hint: for many of these it's easier if you work with squared values (and you may need to "complete the square"). Beyond non-negativity of norms, it may also help to use the Cauchy-Schwartz inequality, and/or to use that $\|x\|_1 = x^\top \text{sign}(x)$. We've used M as the largest eigenvalue of the (positive-definite) matrix H and m as the smallest eigenvalue, so $mI \preceq H \preceq MI$.