EE511 Advanced Intro. to Machine Learning **University of Washington**
Winter 2021 **Dept. of Electrical & Computer Engineering**

Homework 3. Due **Monday Feb, 22nd, 2020, 11:59pm** Electronically

Prof: J. Bilmes <bilmes@uw.edu>
TA: Lilly Kumari <lkumari@uw.edu>

Saturday Feb 13th, 2021

Prof. Bilmes will hold an extra office hour Sunday night, 2/21 at 10:00pm.

General Instructions.

This homework consists of two parts, a write-up part that needs to be turned in using a single pdf file, and also a programming part that needs to be turned in using a zipped Jupyter notebook file (or files). A Jupyter notebook file has extenion `.ipynb` and a zip file has extension `.zip` and should contain one or more `.ipynb` files.

Doing your homework by hand and then converting to a PDF file (by say taking high quality photos using a digital camera and then converting that to a PDF file) is fine, as there are many jpg to pdf converters on the web. Alternatively, you are welcome to use Latex (great for math and equations), Microsoft Word, and Google Docs, or hand-written paper, as long as the final submitted format is a single pdf.

For the plots requested in the programming session, you can either save them as pictures and insert them manually into the writeup, or directly export the completed jupyter notebook to a pdf file (in jupyter notebook, "File→Download as→PDF via LaTex") and copy it in to your writeup.

Some of the problems below might require that you look at some of the lecture slides at our web page (https://canvas.uw.edu/courses/1431528).

Note that the due dates and times are often in the evenings.

As mentioned above, for the programming problems, you need to submit your code (written in python as a Jupyter notebook) and the answers to the non-coding questions should also be included in the pdf write-up. Your code answers must to be in python, no other language is accepted.

Neatness and clarity count! : Answers to your questions must be clearly indicated in all cases. Not only correctness, but clarity and completeness is necessary to receive full credit. Justify you answers. A correct answer does not guarantee full credit and a wrong answer does not guarantee poor credit, hence show all work and justify each step, thinking "clarity" and "neatness" along the way. If we can't understand your answer, or if your answers are not well and neatly organized, you will not receive full credit.

All homework is due electronically via the link https://canvas.uw.edu/courses/1431528/assignments. This means that on canvas you turn in two files: (1) **a pdf file** with answers to the writeup questions, and (2) **a zip file with python code (in jupyter notebook files)**. **Please do not submit any fewer or any more than these two files.**

Problem 1. Math Warm-up [10 points]

In lecture slides, on the topic of binary naive Bayes with multivariate Gaussian class-conditional distributions $p(x|y)$, we claimed that if there were equal class variances the log ratio has the form $b_i x + c_i$, and if not it has the form $a_i x_i^2 + b_i x_i + c_i$.

Problem 1(a). [4 points] Derive a simple expression for b_i and c_i in the equal class variance case (5 points).

Problem 1(b). [4 points] Derive a simple expression for a_i, b_i, c_i for the unequal class variance case (5 points).

Problem 1(c). [2 points] Relate this back to logistic regression. In other words, what is the relationship between logistic regression and Gaussian class conditional distributions in the naive Bayes case?

Problem 2. Gaussian Distribution and the Curse of Dimensionality [40 points]

In this problem, we will investigate the Gaussian distribution in high dimensional space, and develop intuitions and awareness about the **curse of dimensionality**, a critical concept that everyone who wishes to pursue study in machine learning should understand. Note that, as we mentioned in class, the curse is not insurmountable, in fact people use AI/ML in high dimensional spaces every day — however, we should be aware of how strange geometry can act in high dimensions and always question ones own intuitions when one uses our intuition about 3D spaces to generalize about what we think might be happening in higher dimensionality spaces (this is an essential component to helping to debug ML systems on real data).

For a random variable x of m dimensions (i.e., $x \in \mathbb{R}^m$) drawn from a multivariate Gaussian distribution, recall that the Gaussian density function takes the form:

$$p(x) = \frac{1}{|2\pi C|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^{\mathsf{T}} C^{-1} (x-\mu)\right) \tag{1}$$

where $\mu \in \mathbb{R}^m$ is an m-dimensional mean vector and $C \in \mathbb{R}^{m\times m}$ is an order $m \times m$ symmetric positive definite covariance matrix. When C is a diagonal matrix, then the covariance between different dimensions is zero, what is known as an axis aligned Gaussian, and when $C = \sigma^2 I$, I being the $m \times m$ identity matrix, we already saw this in Homework 1 where the $m = 2$ (2D) Gaussian in this case has spherical (circle) shaped contours. A spherical Gaussian in m dimensions thus has the following equation form:

$$p(x) = \frac{1}{(2\pi\sigma^2)^{m/2}} \exp\left(-\frac{\|x\|_2^2}{2\sigma^2}\right) \tag{2}$$

We start with examining some basic geometric properties about the sphere in m dimensional space. A sphere is generally a collection of points such that the distance of any point to the center of the sphere (we always center the sphere on origin $\vec{0}$ for simplicity) is equal to r, the radius. In other words, we define an m-dimensional sphere as $\mathcal{S}_{m-1}(r) = \{x \in \mathbb{R}^m : \|x\|_2 = r\}$, the set of points in m-dimensional space that are distance r from the origin (note that $\mathcal{S}_{m-1}(r)$ is the equation for the **surface** of the sphere, although in some fields, such as physics, they define the sphere as all points on both the surface and the interior, as in $\{x \in \mathbb{R}^m : \|x\|_2 \le r\}$). We also use $V_m(r)$ for the volume of an m-dimensional sphere.

$S_{m-1}(r)$ represents the surface area of the m-dimensional sphere (meaning, e.g., that $m = 2$ dimensional sphere of radius r has surface area $S_1(r)$, this convention is used since the surface area is a curved $m-1$ dimensional manifold embedded in m-dimensional ambient space, and thus we say that $S_{m-1}(r)$ (using $m-1$ as the subscript to S) is the surface of an m-dimensional sphere).

Please make sure you answer every question:

Problem 2(a). [1 points] Before we move to m dimensions, let's talk about 2D and 3D cases. Write down and then think about the equations, in terms of the radius r, of $S_{m-1}(r)$ for $m \in \{2,3\}$ and $V_m(r)$ for $m \in \{2,3\}$.

Problem 2(b). [5 points] Intuitively explain the following equation:

$$S_{m-1}(r) = \frac{d}{dr} V_m(r). \tag{3}$$

Why does this equation make sense and why should it be true? You may help to convince yourself, and improve intuition, by verifying the equations of $S_1(r)$, $V_2(r)$, $S_2(r)$ and $V_3(r)$ from the previous question.

Problem 2(c). [4 points] As you may have guessed, $V_m(r)$'s only dependence on r and m is via m's power of r, or specifically r^m. Suppose for a unit sphere ($r = 1$) in m dimensional space, the surface area is given by the quantity $\bar{S}_{m-1}$ (note $\bar{S}_{m-1}$ does not involve the variable r since $r = 1$ here). Write $S_{m-1}(r)$ for any r in terms of both r and $\bar{S}_{m-1}$.

Problem 2(d). [5 points] Now consider all the points on the sphere $\mathcal{S}_{m-1}(r)$ (which, because of our definition of $\mathcal{S}_{m-1}(r)$ really means the surface). We wish to integrate over all of those points weighted by the Gaussian probability density $p(x)$ of each point, where $p(x)$ is defined as given in Equation (2). That is, we integrate over all points $x \in \mathbb{S}_{m-1}(r)$ weighted by $p(x)$, meaning $\rho_m(r) = \int_{x \in \mathbb{S}_{m-1}(r)} p(x)dx$. Indeed, this is an integration, but you can avoid doing the mathematical integration by utilizing the results from previous questions. Write the equation $\rho_m(r)$ for the integrated density of sampled points from the Gaussian distribution lying on the surface of $\mathbb{S}_{m-1}(r)$.

Problem 2(e). [5 points] For large m, show that $\rho_m(r)$ has a single maximum value at $\hat{r}$ such that $\hat{r} \approx \sqrt{m}\sigma$. Hint: recall the techniques we used to find the minimum value of a function when you did so for the case of linear regression (and in your calculus classes).

Problem 2(f). [10 points] For large m, consider a small value $\epsilon \ll \hat{r}$ (the symbol "$\ll$" means "much less than"), and show that

$$\rho(\hat{r} + \epsilon) \approx \rho(\hat{r})e^{-\frac{\epsilon^2}{\sigma^2}}. \tag{4}$$

Hint: during your derivation, first get the expression simplified and close to the desired form. Then use a Taylor expansion to get the approximation.

Problem 2(g). [3 points] The previous problem shows that $\hat{r}$ is the radius where most of the probability mass is in a high dimension Gaussian, and moreover, as we move away from this radius, say going from $\hat{r}$ to $\hat{r}+\epsilon$, then the total mass becomes smaller exponentially quickly in ϵ. Also, note that since $\hat{r} \approx \sqrt{m}\sigma$, for large m we have $\sigma \ll \hat{r}$. Note that σ (the standard deviation) is in low dimensions usually used to indicate where much of the probability mass is, and this mass is concentrated at the origin since $p(x) > p(x')$ whenever $\|x\|_2 < \|x'\|_2$ with the highest density point being at the origin $x = 0$ with $p(0)$ being the largest value. In low-dimensions we think of most of the mass being centered at 0 with a window roughly of σ.

When we get to high dimensions, however, most of the mass is far away from the σ neighborhood around the origin. Taken together, this means that most of the probability mass, in a high dimensional Gaussian, is concentration in a thin skin (e.g., think of the skin of an m-dimensional apple or synthetic leather layer of an m-dimensional soccer ball) of large radius.

If we only get finite number of samples from a high dimensional Gaussian distribution, therefore, where do most of points reside? At what radius do they reside? For a low dimensional Gaussian distribution, where do most points reside?

Problem 2(h). [2 points] The conclusion from the previous questions may seem highly counter intuitive, but so it goes with the curse of dimensionality. Calculate and compare the probability density at the origin and at one point on sphere $\mathbb{S}_{m-1}(\hat{r})$. The curse of dimensionality comes from the extremely high growth rate of volume as dimensionality of the space increases (there's just a lot of room available in high dimensions).

Problem 2(i). [5 points] The

Write a python script that samples from an m-dimensional Gaussian. For each $m \in \{1, 2, \ldots, 40\}$, produce 100 samples and compute the mean and standard deviation of the radii of the samples, and plot this as a function of m. Is your plot consistent with the above? Why or why not? Fully understand what you see, and clearly explain to us that you understand it and how you justify it.

Problem 3. Programming Problem: Lasso [50 points]

We provide an ipython notebook "lasso.ipynb" for you to complete. In your terminal, please go to the directory where this ipynb file is located, and run command "jupyter notebook". A local webpage will be automatically opened in your web browser. Click the above file to open the notebook.

You need to complete everything below each of the "TODO"s that you find (please search for every "TODO"). Once you have done that, please submit the completed ipynb file as part of your included .zip file.

In your writeup, you also need to include the plots and answers to the questions required in this session. Please include the plots and answers in the pdf file in your solution.

Recall that for lasso, we aim to solve:

$$\text{argmin}_{\theta,\theta_0} F(\theta,\theta_0) \quad \text{where} \quad F(\theta,\theta_0) = \frac{1}{2}\sum_{i=1}^{n}(\langle x^{(i)},\theta\rangle + \theta_0 - y^{(i)})^2 + \lambda\sum_{j=1}^{m}|\theta_j|. \tag{5}$$

where λ is the hyperparameter to control the regularization. Here, $x^{(i)}$ is an m-dimensional data point with the corresponding label $y^{(i)}$, θ is the weight vector of m dimensions and θ_0 is a scalar offset. Note that in class, we've been saying that $x_0^{(i)} = 1$ for all i (so this would be $m+1$ dimensions if we were to wrap it up in one vector) but here we are breaking it out so that θ_0 is not in the L1 regularization term $\lambda\sum_{j=1}^{m}|\theta_j|$.

Remarks: Read each subproblem carefully before proceeding to work on them. Do not use the extended representation in Homework 2, because we do not want to regularize/penalize θ_0. Do not include θ_0 in the computation of precision/recall/sparsity. However, do not forget to include it when you compute the prediction produced by lasso model, because it is one part of the model.

Problem 3(a). [15 points] Implement the coordinate descent algorithm to solve the lasso problem in the notebook.

We provide a function `DataGenerator` to generate synthetic data in the notebook. Please read the details of the function to understand how the data are generated. In this problem, you need to use $n = 50, m = 75, \sigma = 1.0, \theta_0 = 0.0$ as input arguments to the data generator. Do not change the random seed for all the problems afterwards.

Stopping criteria of the outer loop: stop the algorithm when either of the following is fulfilled: 1) the number of steps exceed 100; or 2) no element in θ changes more than ϵ between two successive iterations of the outer loop, i.e., $\max_j |\theta_j(t) - \theta_j(t-1)| \le \epsilon$, where the recommended value for $\epsilon = 0.01$, where $\theta_j(t)$ is the value of θ_j after t iterations.

At the beginning of lasso, use the given initialization function θ, θ_0 `= Initialw(X, y)` to initialize θ and θ_0 by the least square regression or ridge regression.

You can try different values of λ to make sure that your solution makes sense to you (Hint: `DataGenerator` gives the true θ and θ_0).

Solve lasso on the generated synthetic data using the given parameters and report indices of non-zero weight entries. Plot the objective value $F(\theta,\theta_0)$ v.s. coordinate descent step. The objective value should always be non-increasing.

Problem 3(b). [5 points] Implement an evaluation function in the notebook to calculate the precision and recall of the non-zero indices of the lasso solution with respect to the non-zero indices of the true vector that generates the synthetic data. Precision and recall are useful metrics for many machine learning tasks. For this problem in specific,

$$\text{precision} = \frac{|\{\text{non-zero indices in } \hat{\theta}\} \cap \{\text{non-zero indices in } \theta^*\}|}{|\{\text{non-zero indices in } \hat{\theta}\}|}; \tag{6}$$

$$\text{recall} = \frac{|\{\text{non-zero indices in } \hat{\theta}\} \cap \{\text{non-zero indices in } \theta^*\}|}{|\{\text{non-zero indices in } \theta^*\}|}, \tag{7}$$

where θ^* is the θ in true model weight, while $\hat{\theta}$ is the θ in lasso solution.

You also need to report the sparsity (the number of nonzero entries) of $\hat{\theta}$, and the RMSE of the training data (RMSE was given in Problem 2 of Homework 2). Note that a solution can have high precision with low recall (e.g., the solution contains only one correct non-zero index while the true vector contains many) and vice versa. Report the precision and recall of your lasso solution from the previous problem.

Problem 3(c). [10 points] Vary λ and solve the lasso problem multiple times. Choose 50 evenly spaced λ values starting with $\lambda_{max} = \|(y-\bar{y})X\|_\infty$ ($\bar{y}$ is the average of elements in y, and $\|a\|_\infty = \max_j |a_j|$), and ending with $\lambda_{min} = 0$. Plot the precision v.s. λ and recall v.s. λ curves on a single 2D plot. Briefly explain the plotted pattern and curves. On top of this, try to have fun with λ and play with this hyperparameter, explore, discover, and tell us what you have discovered.

Draw a "lasso solution path" for each entry of θ in a 2D plot. In particular, use λ as the x-axis, for each entry θ_i in θ achieved by lasso, plot the curve of θ_i vs. λ for all the values of λ you tried similar to the plot we showed in class from the Murphy text (in your case, there are 50 points on the curve). Draw such curves for all the m entries in θ within a 2D plot, use the same color for the 5 features in `DataGenerator` used to generate the data, and use another very noticeably distinct color for other features. If necessary, set up proper ranges for x-axis and y-axis, so you can see sufficient detail.

Now change the noise's standard deviation $\sigma = 10.0$ when using `DataGenerator` to generate synthetic data, draw the lasso solution path again. Compare the two solution path plots with different σ, and explain their difference. Be complete, and clear.

Problem 3(d). [5 points] Use the synthetic data generation code with different parameters: $(n = 50, m = 75), (n = 50, m = 150), (n = 50, m = 1000), (n = 100, m = 75), (n = 100, m = 150), (n = 100, m = 1000)$ (keeping other parameters the same as in Problem 3(a)). Vary λ in the same way as in the previous question (Problem 3(c)), and find the λ value that can generate both good precision and recall for each set of synthetic data points.

For each case, draw the "lasso solution path" defined in Problem 3(c).

To get good precision and recall values, which one of the following relationships between n and m is more probable: $n = O(m^2)$, $n = O(m)$ or $n = O(\log(m))$? You can try more choices of n and m to verify your guess empirically.

Problem 3(e). [15 points] This question is challenging, requiring major change to your previous implementation as well as significant training time. Run lasso to predict reviews' stars on Yelp by selecting important features (words) from review comments. We provide the data in hw3_data.zip. You can unzip the file and use the provided function `DataParser` in the notebook to load the data. There are three files: `star_data.mtx`, `star_labels.txt`, `star_features.txt`. The first file stores a matrix market matrix and `DataParser` reads it into a scipy csc sparse matrix, which is your data matrix X. The second file contains the labels, which are the stars of comments on Yelp, is your y. The third file contains the names of features (words). For the last two txt files, you can open them in editor and have a look at their contents.

The sparse data X has size 45000×2500, and is split into the training set (the first 30000 samples), validation set (the following 5000 samples) and the test set (the last 10000 samples) by `DataParser`. Each column corresponds to a feature, i.e., a word appearing in the comments. Your mission is to solve lasso on the training set, tune the λ value to find the best RMSE on the validation set, and evaluate the performance of the obtained lasso model on the test set.

Important to read before you start: Here we are dealing with a sparse data matrix. Most numpy operations for dense matrices you used for implementing lasso in Problem 3(a) cannot be directly applied to sparse matrices here. You can still use the framework you got in Problem 3(a), but you need to replace some dense matrix operations (multiply, dot, sum, slicing, etc.) by using sparse matrix operations from `scipy.sparse` (please refer to `https://docs.scipy.org/doc/scipy/reference/sparse.html` for details of sparse matrix operations).

The sparse matrix format here aims to help you make the algorithm more efficient to handle sparse data. Do not try to directly transform the sparse matrix X to a dense one by using `X.todense()`, since it will waste too much memory. Instead, try to explore the advantage of different sparse matrix types (csc, coo, csr) and avoid their disadvantages, which are listed under each sparse matrix type in the above link. You can change the format of a sparse matrix X to another one by using (for example) `X.tocsc()` if necessary, but do not use it too often. For some special sparse matrix operations, it might be more efficient to write it by yourself. We provide an example `cscMatInplaceEleMultEveryRow` in the notebook. You can use it, or modify it for your own purpose.

This will be a good practice for you to think about how to write an efficient ML algorithm. Try to avoid building new objects inside the loop, or computing anything from scratch. You can initialize them before the loop start, and use the lightest way to update them in the loop. Note any operation that seems "small" inside the loop could possibly lead to expensive computations, considering the total number of times it will be executed.

You can use "if" to avoid unnecessary operations on zeros. Do not loop over matrices or vectors if not necessary: use matrix or batch operations provided by numpy or scipy instead. Try to use the most efficient operation when there are many choices reaching the same result. If you write an inefficient code here, running it will take extremely longer time. During debugging, timing each step or operation in the loop will help you figure out which step takes longer time, and you can then focus on how to accelerate it.

Running this experiment will take 1-3 hours, please start early. Before you leave it running by itself, make sure that the timing results indicate a reasonable finishing time, and remember to save intermediate results to avoid possible crush or restart from very beginning. You need to finish:

- Plot the training RMSE (on the training set) v.s. λ values and validation RMSE (on the validation set) v.s. λ values on a 2D plot. Use the definition of λ_{max} in Problem 3(c) and run experiments on multiple values of λ. You can reduce the number of different λ values to 20. You are also allowed to increase the minimal λ to be slightly larger than 0 such that $0 \leq \lambda_{min} \leq 0.1\lambda_{max}$. These two changes will save you some time.

- Plot the lasso solution path defined in Problem 3(c).

- Report the best λ value achieving the smallest validation RMSE you find on the validation set, report the corresponding test RMSE (on test set).

- Report the top-10 features (words in comments) with the largest magnitude in the lasso solution w when using the best λ value, and briefly explain if/why they are meaningful.