Intro to Machine Learning (CS771A, Autumn 2018)
Homework 2
Due Date: September 28, 2018 (11:59pm)

Instructions:

- Only electronic submissions will be accepted. Your main PDF writeup must be typeset in LaTeX (please also refer to the "Additional Instructions" below).
- Your submission will have two parts: The main PDF writeup (to be submitted via Gradescope `https://www.gradescope.com/`) and the code for the programming part (to be submitted via this Dropbox link: `https://tinyurl.com/cs771-a18-hw2-a`). Both parts must be submitted by the deadline. We will be accepting late submissions upto 72 hours after the deadline (with every 24 hours delay incurring a 10% late penalty). We won't be able to accept submissions after that.
- We have created your Gradescope account (you should have received the notification). Please use your IITK CC ID (not any other email ID) to login. Use the "Forgot Password" option to set your password.

Additional Instructions

- We have provided a LaTeX template file `hw2sol.tex` to help typeset your PDF writeup. There is also a style file `ml.sty` that contain shortcuts to many of the useful LaTeX commends for doing things such as boldfaced/calligraphic fonts for letters, various mathematical/greek symbols, etc., and others. Use of these shortcuts is recommended (but not necessary).
- Your answer to every question should begin on a new page. The provided template is designed to do this automatically. However, if it fails to do so, use the `\clearpage` option in LaTeX before starting the answer to a new question, to *enforce* this.
- While submitting your assignment on the Gradescope website, you will have to specify on which page(s) is question 1 answered, on which page(s) is question 2 answered etc. To do this properly, first ensure that the answer to each question starts on a different page.
- Be careful to flush all your floats (figures, tables) corresponding to question n before starting the answer to question $n+1$ otherwise, while grading, we might miss your important parts of your answers.
- Your solutions must appear in proper order in the PDF file i.e. solution to question n must be complete in the PDF file (including all plots, tables, proofs etc) before you present a solution to question $n+1$.
- For the programming part, all the code and README should be zipped together and submitted as a single file named `yourrollnumber.zip`. Please DO NOT submit the data provided.

Problem 1 (10 marks)

(A Circular Definition) Consider a logistic regression model $p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \frac{1}{1+\exp(-y_n \boldsymbol{w}^\top \boldsymbol{x}_n)}$, with a zero-mean Gaussian prior $p(\boldsymbol{w}) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$. Note that this loss function for logistic regression assumes $y_n \in \{-1, +1\}$ instead of $\{0, 1\}$. Show that the MAP estimate for $\boldsymbol{w}$ can be written as $\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$ where each α_n itself is a function of $\boldsymbol{w}$. Based on the expression of α_n, you would see that it has a precise meaning. Briefly state (in 50 words of less, may include equations) what α_n means, and also briefly explain (in words or less) why the result $\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$ makes sense for this model.

Problem 2 (10 marks)

(Generative meets Discriminative) Consider a generative classification model for binary classification. Assume the class-marginal distribution to be defined as $p(y = 1) = \pi$ and assume each class-conditional distribution to be defined as a product of D Bernoulli distributions, i.e., $p(\boldsymbol{x}|y = 1) = \prod_{d=1}^{D} p(x_d|y = 1)$ where $p(x_d|y = 1) = \text{Bernoulli}(x_d|\mu_{d,1})$, and $p(\boldsymbol{x}|y = 0) = \prod_{d=1}^{D} p(x_d|y = 0)$ where $p(x_d|y = 0) = \text{Bernoulli}(x_d|\mu_{d,0})$. Note that this makes uses of the naïve Bayes assumption.

Show that this model is equivalent (in its mathematical form) to a probabilistic discriminative classifier. In particular, derive the expression for $p(y = 1|\boldsymbol{x})$, and state what type of decision boundary will this model learn - linear, quadratic, or something else (looking at the expression of $p(y = 1|\boldsymbol{x})$ should reveal that)? Clearly write down the expressions for the parameters of the equivalent probabilistic discrimininative model in terms of the generative model parameters $(\pi, \mu_{d,0}, \mu_{d,1})$. Note that you do not have to estimate the parameters $\pi, \mu_{d,0}, \mu_{d,1}$ (but you may try that for practice if you want).

Problem 3 (5 marks)

(The Equivalence) Consider a constrained version of least squares linear regression where we constrain the ℓ_2 norm of $\boldsymbol{w}$ to be less than on equal to some $c > 0$: $\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2, \quad \text{s.t.} \quad ||\boldsymbol{w}|| \leq c$

Show, formally, that it is possible to have an ℓ_2 regularized least squares linear regression model that will give the *exact* same solution as the solution to the above constrained problem.

Problem 4 (20 marks)

(Softmax and Variants) Consider N training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$ where $\boldsymbol{x}_n \in \mathbb{R}^D$, and $y_n \in \{1, \ldots, K\}$. Suppose we wish to use this data to learn the multiclass logistic (or "softmax") regression model which defines $p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \mu_{nk} = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^{K} \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)}$, where μ_{nk} is the predicted probability of $y_n = k$.

Derive the MLE solution for $\mathbf{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$. You would notice that, just like logistic regression, there is no closed form solution for $\mathbf{W}$. So you will need to write down the log likelihood, take its derivative w.r.t. each column $\boldsymbol{w}_k$ of $\mathbf{W}$ to compute the gradient, and derive the gradient descent (GD) update rule for each $\boldsymbol{w}_k$. Show the basic steps of your derivation and write down the final expression for the GD update of each $\boldsymbol{w}_k$. Assume a fixed learning rate $\eta = 1$ for simplicity.

Next, consider the stochastic gradient descent (SGD) update for the same model where each iteration takes a randomly chosen example $(\boldsymbol{x}_n, y_n)$ to update $\mathbf{W}$. Write down the expressions for these SGD updates and the overall sketch of the corresponding SGD algorithm.

Finally consider a special case of the above SGD algorithm for this model, where the predicted "soft" class probabilities μ_{nk} are replaced by hard class assignments, i.e., $\mu_{nk} = 1$ for $k = \arg\max_\ell \{\mu_{n\ell}\}_{\ell=1}^{K}$, and $\mu_{nk'} = 0, \forall k' \neq k$. Write down the expressions for the SGD update in this case and the overall sketch of the SGD algorithm. How are these updates different from the previous case where you used soft probabilities μ_{nk}?

Problem 5 (10 marks)

(Separating Convex Hulls) Given a set of data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, we define the convex hull to be the set of all points $\boldsymbol{x}$ given by $\boldsymbol{x} = \sum_n \alpha_n \boldsymbol{x}_n$ where $\alpha_n \geq 0$ and $\sum_n \alpha_n = 1$ (Intuitively, the convex hull of a set of points in the solid region that they enclose.) Consider a second set of points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M$ together with their corresponding convex hull. Show that the set of $\boldsymbol{x}$s and the set of $\boldsymbol{y}$s are linearly separable *if and only if* the convex hulls do not intersect.

Problem 6 (5 marks)

(Arbitrary Choice?) Formally, show that changing the condition $y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq 1$ in SVM to a different condition $y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + b) \geq m$ does not change the effective separating hyperplane that is learned by the SVM. Assume the hard-margin SVM for simplicity.

Problem 7 (40 marks)

(Programming Problem)

Part 1: You are provided a dataset in the file `binclass.txt`. In this file, the first two numbers on each line denote the two features of the input $\boldsymbol{x}_n$, and the third number is the binary label $y_n \in \{-1, +1\}$.

Implement a generative classification model for this data assuming Gaussian class-conditional distributions of the positive and negative class examples to be $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_+, \sigma_+^2 \mathbf{I}_2)$ and $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_-, \sigma_-^2 \mathbf{I}_2)$, respectively. Note that here $\mathbf{I}_2$ denotes a 2×2 identity matrix. Assume the class-marginal to be $p(y_n = 1) = 0.5$, and use MLE estimates for the unknown parameters. Your implementation need not be specific to two-dimensional inputs and it should be almost equally easy to implement it such that it works for any number of features (but it is okay if your implementation is specific to two-dimensional inputs only).

On a two-dimensional plane, plot the examples from both the classes (use red color for positives and blue color for negatives) and the **learned decision boundary** for this model. Note that we are not providing any separate test data. Your task is only to learn the decision boundary using the provided training data and visualize it.

Next, repeat the same exercise but assuming the Gaussian class-conditional distributions of the positive and negative class examples to be $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_+, \sigma^2 \mathbf{I}_2)$ and $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_-, \sigma^2 \mathbf{I}_2)$, respectively.

Finally, try out a Support Vector Machine (SVM) classifier (with **linear kernel**) on this data and show the learn decision boundary. For this part, you do not need to implement SVM. There are many nice implementations of SVM available, such as the one in scikit-learn and the very popular libSVM toolkit. Assume the "C" (or λ) hyperparameter of SVM in these implementations to be 1.

Note: One of the goals of this problem is to familiarize you with SVM implementations which, despite the popularly of deep learning methods nowadays, still remain a very useful tool for classification problems. SVM also allows learning nonlinear models too using kernels but, for now, let's limit ourselves with linear SVMs (we will soon see kernels in the class, after which you can try nonlinear SVMs). Also, while in this homework, you would just be playing with existing SVM implementations, in the next homework, you will implement SVM on your own (which isn't all that hard if you have understood the basics of SVM and SVM optimization). :-)

Part 2: Repeat the same experiments as you did for part 1 but now using a different dataset `binclassv2.txt`. Looking at the results of both the parts, which of the two models (generative classification with Gaussian class-conditional and SVM) do you think seems to work better for each of these datasets, and in general?

Deliverables: Include your plots (use a separate, appropriately labeled plot, for each case) and experimental findings in the main writeup PDF. Submit your codes in a separate zip file on the provided Dropbox link. Please comment the code so that it is easy to read and also provide a README that briefly explains how to run the code. For the SVM part, you do not have to submit any code but do include the plots in the PDF (and mention the software used).