Intro to Machine Learning (CS771A, Autumn 2020)
Homework 1
Due Date: October 30, 2020 (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).
- The PDF writeup containing your solution has to be submitted via Gradescope `https://www.gradescope.com/`.
- 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 `hw1sol.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 $n+1$.

Problem 1 (15 marks)

(Absolute Loss Regression with Sparsity) The absolute loss regression problem with ℓ_1 regularization is

$$\boldsymbol{w}_{opt} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{n=1}^{N} |y_n - \boldsymbol{w}^\top \boldsymbol{x}_n| + \lambda ||\boldsymbol{w}||_1$$

where $||\boldsymbol{w}||_1 = \sum_{d=1}^{D} |w_d|$, $|.|$ is the absolute value function, and $\lambda > 0$ is the regularization hyperparameter.

Is the above objective function convex? You don't need to prove this formally; just a brief reasoning based on properties of other functions that are known to be convex/non-convex would be fine.

Derivate the expression for the (sub)gradient vector for this model.

Problem 2 (15 marks)

(Feature Masking as Regularization) Consider linear regression model by minimizing the squared loss function $\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$. Suppose we decide to mask out or "drop" each feature x_{nd} of each input $\boldsymbol{x}_n \in \mathbb{R}^D$, independently, with probability $1 - p$ (equivalently, retaining the feature with probability p). Masking or dropping out basically means that we will set the feature x_{nd} to 0 with probability $1 - p$. Essentially, it would be equivalent to replacing each input $\boldsymbol{x}_n$ by $\tilde{\boldsymbol{x}}_n = \boldsymbol{x}_n \circ \boldsymbol{m}_n$, where $\circ$ denotes elementwise product and $\boldsymbol{m}_n$ denotes the $D \times 1$ binary mask vector with $m_{nd} \sim \text{Bernoulli}(p)$ ($m_{nd} = 1$ means the feature x_{nd} was retained; $m_{nd} = 0$ means the feature x_{nd} was masked/zeroed).

Let us now define a new loss function using these masked inputs as follows: $\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \tilde{\boldsymbol{x}}_n)^2$. Show that minimizing the *expected* value of this new loss function (where the expectation is used since the mask vectors $\boldsymbol{m}_n$ are random) is equivalent to minimizing a **regularized** loss function. Clearly write down the expression of this regularized loss function.

Problem 3 (40 marks)

(Multi-output Regression with Reduced Number of Parameters) Consider the multi-output regression in which each output $\boldsymbol{y}_n \in \mathbb{R}^M$ in a real-valued vector, rather than a scalar. Assuming a linear model, we can model the outputs as $\mathbf{Y} \approx \mathbf{XW}$, where $\mathbf{X}$ is the $N \times D$ feature matrix and $\mathbf{Y}$ is $N \times M$ response *matrix* with row n being $\boldsymbol{y}_n^\top$ (note that each column of $\mathbf{Y}$ denotes one of the M responses), and $\mathbf{W}$ is the $D \times M$ **weight matrix**, with its M columns containing the M weight vectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M$. Let's define a squared error loss function $\sum_{n=1}^{N}\sum_{m=1}^{M}(y_{nm} - \boldsymbol{w}_m^\top \boldsymbol{x}_n)^2$, which is just the usual squared error but summed over all the M outputs. Firstly, verify that this can also be written in a more compact notation as $\text{TRACE}[(\mathbf{Y} - \mathbf{XW})^\top(\mathbf{Y} - \mathbf{XW})]$.

Further, we will assume that the weight matrix $\mathbf{W}$ can be written as a product of two matrices, i.e., $\mathbf{W} = \mathbf{BS}$ where $\mathbf{B}$ is $D \times K$ and $\mathbf{S}$ is $K \times M$ (assume $K < \min\{D, M\}$). Note that there is a benefit of modeling $\mathbf{W}$ this way, since now we need to learn only $K \times (D + M)$ parameters as opposed to $D \times M$ parameters and, if K is small, this can significantly reduce the number of parameters (in fact, reducing the *effective* number of parameters to be learned is another way of regularizing a machine learning model). Note (you can verify) that in this formulation, each $\boldsymbol{w}_m$ can be written as a linear combination of K columns of $\mathbf{B}$.

With the proposed representation of $\mathbf{W}$, the new objective will be $\text{TRACE}[(\mathbf{Y} - \mathbf{XBS})^\top(\mathbf{Y} - \mathbf{XBS})]$ and you need to learn both $\mathbf{B}$ and $\mathbf{S}$ by solving the following problem:

$$\{\hat{\mathbf{B}}, \hat{\mathbf{S}}\} = \arg\min_{\mathbf{B},\mathbf{S}} \text{TRACE}[(\mathbf{Y} - \mathbf{XBS})^\top(\mathbf{Y} - \mathbf{XBS})]$$

We will ignore regularization on $\mathbf{B}$ and $\mathbf{S}$ for brevity/simplicity.

Derive an alternating optimization (ALT-OPT) algorithm to learn $\mathbf{B}$ and $\mathbf{S}$, clearly writing down the expressions for the updates of $\mathbf{B}$ and $\mathbf{S}$. Are both subproblems (solving for $\mathbf{B}$ and solving for $\mathbf{S}$) equally easy/difficult in this ALT-OPT algorithm? If yes, why? If no, why not?

Note: Since $\mathbf{B}$ and $\mathbf{S}$ are matrices, if you want, please feel free to use results for matrix derivatives (results you will need can be found in Sec. 2.5 of the Matrix Cookbook). However, the problem can be solved even without using matrix derivative results with some rearrangement of terms and using vector derivatives.

Problem 4 (10 marks)

Ridge Regression using Newton's Method Consider the ridge regression problem:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \frac{1}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\lambda}{2}\boldsymbol{w}^\top \boldsymbol{w} = \arg\min_{\boldsymbol{w}} \frac{1}{2}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^\top \boldsymbol{w}$$

where $\mathbf{X}$ is the $N \times D$ feature matrix and $\boldsymbol{y}$ is the $N \times 1$ vector of labels of the N training examples. Note that the factor of $\frac{1}{2}$ has been used in the above expression just for convenience of derivations required for this problem and does not change the solution to the problem.

Derive the Newton's method's update equations for each iteration. For this model, how many iterations would the Newton's method will take to converge?

Problem 5 (20 marks)

(Dice Roll) You have a six-faced dice which you roll N times and record the number of times each of its six faces are observed. Suppose these numbers are $N_1, N_2, \ldots, N_6$, respectively. Assume that the probability of a random roll of the dice showning the k^{th} face ($k = 1, 2, \ldots, 6$) to be equal to $\pi_k \in (0, 1)$.

Assuming an appropriate conjugate prior for the probability vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_6]$, derive its MAP estimate. In which situation(s), you would expect the MAP solution to be better than the MLE solution?

Also derive the full posterior distribution over $\boldsymbol{\pi}$ using the same prior that you used for MAP estimate. Given this posterior, can you get the MLE and MAP estimate without solving the MLE and MAP optimization problems? If yes, how? If no, why not?