**CS 512: Advanced Machine Learning**

# Homework 2

*Prof.: Sathya N. Ravi* Assigned: 03/27/2023, Due: 04/24/2023

Few reminders:

- Homework is due at the end of day on the designated date on Blackboard.
- No homework or project is accepted in mailbox of instructor.
- You may discuss homework with classmates and work in groups of up to **two**. However, you may not share any code, carry out the assignment together, or copy solutions from any other groups. Discussions between groups should be minimal, verbal during lecture or if appropriate, on Piazza only. The submitted version must be worked out, written, and submitted by your group alone.
- **Important:** Each question (or subpart) should have a **lead** who has finalized the submitted solutions after discussions within group. The lead should be *explicitly* indicated in the submissions. There can be two leads for any question. The submitted solution is assumed to be verified by the other person, and so the grade is assigned equally.
- All solutions must be typeset (I recommend Latex, Markdown, or Word, please do not use nonstandard formats) with code attached in Python format whichever is appropriate. In addition to submitting code in Jupyter Notebook, please attach a a PDF copy of Jupyter Notebook in your PDF along with figures, comments, text, any other results, behavior of your implementation in other parameter settings, and so on.
- Your final submission will be zipped, and should contain: (i) the main PDF file containing solutions for each question including text, sample figures, calculation, analysis, and general writeup, and (ii) Jupyter notebook for coding or implementation questions. Additional/utility code, results can be included as folders with Readme file indicating the purpose of the folder. **Everyone** will make this submission.
- Submitting someone else's work (outside of the group) as your own is academic misconduct. Such cheating and plagiarism will be dealt with in accordance with University procedures (see the page on Academic Misconduct at this link).

Answer the following questions as completely as possible. We use $\mathbb{R}^d$ to denote $d$ dimensional vector spaces over the reals $\mathbb{R}$. Similarly, $\mathbb{Z}_+ =$ Set of positive integers, $\mathbb{Z}_{\geq 0} =$ set of nonnegative integers, $[k] = \{1, 2, \ldots, k\}$.

1. Exercise 19.1 in PML on Derivative of log partition function.

2. Exercise 19.2 in PML on Conditional Independence properties of Gaussian Graphical Models.

3. **Alignining Features/Embeddings By Optimizing Inverse Similarity Functions.** In this problem, we will formulate and solve a task that is often faced in merging or combining models trained with different modalities viz., text, audio, images, video, tabular, and so on. Intuitively, "modality" can be thought of data collected from number of types of sensors used ot study a phenomenon of interest. To simplify matters, we will first consider the case where there are two modalities denoted by $\mathcal{X} \subseteq \mathbb{R}^d$, and $\mathcal{Y} \subseteq \mathbb{R}^d$. The goal in the so-called alignment problem is to learn an *Alignment function* $\mathcal{M} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ such that it outputs a high value ($\approx 1$) whenever $x \in \mathcal{X}, y \in \mathcal{Y}$ are aligned. Furthermore, we will assume that the alignment function $\mathcal{M}$ is a **nontrivial** *bilinear "form"* on $\mathbb{R}^d$, and so we will use matrix $M \in \mathbb{R}^{d \times d}$ to denote its coordinate representation with the requirement that $M$ is invertible.

   **Formulation.** To figure out the function or to learn the form $M$, we are given with a dataset $\mathcal{D}$ with $n$ points of the form $\mathcal{D} = (x_i, y_i)_{i=1}^n$ where $x_i \in \mathcal{X}, y_i \in \mathcal{Y} \; \forall \; i = 1, \ldots, n$. First, note that $x_i, y_i \in \mathbb{R}^d$, that is, both $x_i, y_i$ are elements of a vector space $\mathbb{R}^d$. So, we simply maximize the likelihood associated with point $i$ given by $\exp\left[-(x_i - y_i)^T M^{-1}(x_i - y_i)\right] / Z_i(M)$ with respect to $M$, where $Z_i(M)$ is the normalizing constant as a function of $M$.

(a) Specify a valid normalizing function $Z_i(M)$. You may assume for all parts (a-d) that $M$ will always be positive definite for simplicity. Hint: Please see definition of Multivariate Gaussian Distribution.

(b) For a fixed $M$, draw the forward pass to compute the loss function or log-likelihood. What is the complexity of your algorithm?

(c) For a fixed $M$, explain how to compute the gradient of the log-likelihood with respect to $M$. What is the complexity of your algorithm? Hint: You may refresh your derivatives using differentials here in Matrix Calculus, especially the ones involving inverses might be useful.

(d) Assume that we initialize gradient descent with $M_0 = \lambda I$ where $\lambda > 0$ and $I$ is the $d \times d$ identity matrix. Using (c) (or otherwise), argue that at iteration $t$, our current iterate $M_t$ is guaranteed to be positive definite.

(e) Now say we parametrize $M$ using sum of outer product of $k$ vectors, that is, $M = \lambda I + \sum_{j=1}^{k} v_j v_j^T$ where $v_j \in \mathbb{R}^d$ are the parameters to be trained. Repeat parts (b-c) for a fixed set parameters $\{v_j\}_{j=1}^k$.

4. **Multiclass Logistic Regression for Sequential Data.** We will recall the Optical Character Recognition (OCR) dataset that we used in **Homework 1 Problem 4**. We are provided with images-label pairs $(x_j^i, y_j^i)$ of word or sequence $i \in \{1, \ldots, n\}$ of letters $j \in \{1, \ldots, L_i\}$ where $x_j^i \in \mathbb{R}^{128}$, $n$ is the total number of words or sequences in the dataset, and $L_i$ is the length of the sequence $i$. In Homework 1, we disregarded the sequential structure of $j$ – that is, we treated each $(x_j^i, y_j^i)$ as if they were i.i.d even though we know that certain letter combinations are rarer. For instance, the popular website `wordfind.com` reports that number of words containing the pairwise letter sequence "wi" (1053 words listed on the website!) is much greater than words with the sequence "iw" (only 22 words listed on the website). In this problem, we will formulate, and solve a sequential likelihood problem that computes the likelihood of individual sequences with pairwise terms involving labels $y_j^i, y_{j+1}^i \; \forall \, j \in \{1, \ldots, L_i - 1\}$.

(a) Using ideas similar to the previous question, we will introduce a edge weight matrix $T \in \mathbb{R}^{26 \times 26}$ such that $T_{y,y'}$ gives us the energy of pairwise label susequence $y = y_j^i, y' = y_{j+1}^i$ in the word $i$. We will use $x^i$ to denote the $i-$th sequence of images, and similarly $y^i$. Using this, we can write the probabilistic model for word/label pair as,

$$p(y^i | x^i, W, T) := \frac{E(y^i, x^i, W, T)}{Z_i} := \frac{\exp\left(\sum_{j=1}^{L_i} w_{y_j^i}^\top x_j^i + \sum_{j=1}^{L_i - 1} T_{y_j^i, y_{j+1}^i}\right)}{\sum_{j=1}^{L_i} \sum_{y_j^i} E(y_j^i, x^i, W, T)}, \quad (1)$$

where $W \times \mathbb{R}^{128 \times 26}$ denote the node weights of all labels or letters, and $w_y \in \mathbb{R}^{128}$ correspond to the weights used for label or letter $y$ as in standard logistic regression (or $y-$th column of $W$ if $y$ is stored as integer). Draw a Probabilistic Graphical Model (PGM) representation of the ikelihood given in (1), clearly indicating where the parameters $W, T$, and data $x_j^i, y_j^i$ are used, and how the likelihood can be computed.

(b) **Inference.** As usual, the first step is to implement a procedure to assign label $y$ for a new sequence sample $x$ of length $L$. For this, we will use the Maximum A-Posteriori estimate of $y$, that is, we will assign $y^* \in \{0, 1\}^{26 \times L}$ such that,

$$y^* \in \arg \max_{\{y_1, \ldots, y_L\}} E(y_1, \ldots, y_L, x_1, \ldots, x_L, W, T). \quad (2)$$

To implement MAP or compute $p(y^* | x, W, T)$ using equation (2), you may consider the exponentiated gradient descent given in Homework 1 that solves the variational formulation

of log-sum-exp if you can afford the memory. However, I strongly suggest using either max-sum or max-product algorithms with the order from left to right (increasing $j$) as it is natural. Moreover, they can be easily modified to calculate the normalizing constant $Z$ as sum-product since $Z$ is just the marginal distribution over all nodes $y_j$. What is the complexity of your inference algorithm in terms of $L_i$ and number of labels $|\mathcal{Y}|$ ($|\mathcal{Y}|$ is equal to 26 in our case)? What is the complexity of computing $Z$ using your algorithm?

(c) **Training.** To train parameters $W$, and $T$, we have to use gradient descent type iterative algorithms on the regularized total log-likelihood given by

$$\frac{1}{n}\sum_{i=1}^{n}\log p(y^i|x^i,W,T)+\lambda_1\|W\|_{Fro}^2+\lambda_2\|T\|_{Fro}^2, \tag{3}$$

where $\|\cdot\|_{Fro}^2$ is the squared Frobenius norm, essentially sum of squares of entries in the matrix input, and $\lambda_1>0,\lambda_2>0$ are regularization parameters chosen using cross-validation. Computing the gradient of all the terms in objective function in (3) is straightforward except for $\log(Z_i)$. Note that naively using the definition in equation (1) is inefficient.

(d) **Gradient of $\log Z$ wrt $W$, and $T$.** Using Chain rule on the definition in (1), and that $y_j^i$ is "one-hot" (as in Problem 1 in this Homework), show that gradient $\nabla_{w_y}\log Z$ is given by,

$$\nabla_{w_y}\log Z_i=\sum_{j=1}^{L_i}p(y_j^i=y|x^i,W,T)x_j^i. \tag{4}$$

This means that, if we can do inference or equivalently compute $p(y^i|x^i,W,T)$ in part (b), we can reuse the computation to obtain the gradient of $\nabla_{w_y}\log Z_i$. Derive a similar result for the gradient with respect to $T_{yy'}$ – it will obviously require joint distributions $p(y_j^i=y,y_{j+1}^i=y'|x^i,W,T)$ which can be computed using part (b) inference results also. What is the complexity of computing gradients $\nabla_{w_y}\log Z_i,\nabla_T\log Z_i$? Hint for implementation: This also means that if you implement the inference procedure in a differentiable manner or for which chain rule can be used, then the gradients can be obtained using any off-the-self automatic differentiation package.

(e) What is the best accuracy after performing 10 fold cross-validation on $\lambda_1,\lambda_2$ under the two settings viz., (i) with and without using gradients from the $\log Z_i$ terms in part (d), and (ii) using L-BFGS vs Gradient Descent, so total four choices possible? Hint for implementation: You do not need to implement L-BFGS from scratch, please use the scipy implementation provided here and here, requires you to provide gradient information.

(f) **Analyzing Optimal Solutions** Report the best accuracies as a $2\times 2$ table. Visualize the optimal matrices $W^*,T^*$ of the dataset under the above two settings (four choices). What is the condition number of $W^*,T^*$? Plot the eigen/singular values of $W^*,T^*$ and recommend the appropriate parameters for prediction or deployment purposes.

**Further Instructions on Calculation of $Z$.**

---