ECE C147/C247, Winter 2023 **Homework #3**
Neural Networks & Deep Learning Prof. J.C. Kao
UCLA ECE TAs: T.M, P.L, R.G, K.K, N.V, S.R, S.P, M.E

Due Wednesday, 8 Feb 2023, by 11:59pm to Gradescope.
100 points total.

1. (15 points) **Backpropagation for autoencoders.** In an autoencoder, we seek to reconstruct the original data after some operation that reduces the data's dimensionality. We may be interested in reducing the data's dimensionality to gain a more compact representation of the data.

   For example, consider $\mathbf{x} \in \mathbb{R}^n$. Further, consider $\mathbf{W} \in \mathbb{R}^{m \times n}$ where $m < n$. Then $\mathbf{Wx}$ is of lower dimensionality than $\mathbf{x}$. One way to design $\mathbf{W}$ so that $\mathbf{Wx}$ still contains key features of $\mathbf{x}$ is to minimize the following expression

   $$\mathcal{L} = \frac{1}{2} \left\| \mathbf{W}^T \mathbf{W} \mathbf{x} - \mathbf{x} \right\|^2$$

   with respect to $\mathbf{W}$. (To be complete, autoencoders also have a nonlinearity in each layer, i.e., the loss is $\frac{1}{2} \left\| f(\mathbf{W}^T f(\mathbf{Wx})) - x \right\|^2$. However, we'll work with the linear example.)

   (a) (3 points) In words, describe why this minimization finds a $\mathbf{W}$ that ought to preserve information about $\mathbf{x}$.

   (b) (3 points) Draw the computational graph for $\mathcal{L}$. **Hint:** You can set up the computational graph to this problem in a way that will allow you to solve for part (d) without taking 4D tensor derivative.

   (c) (3 points) In the computational graph, there should be two paths to $\mathbf{W}$. How do we account for these two paths when calculating $\nabla_{\mathbf{W}} \mathcal{L}$? Your answer should include a mathematical argument.

   (d) (6 points) Calculate the gradient: $\nabla_{\mathbf{W}} \mathcal{L}$.

2. (20 points) **Backpropagation for Gaussian-process latent variable model. (Optional for students in C147: Please write 'I am a C147 student' in the solution and you will get full credit for this problem).** An important component of unsupervised learning is visualizing high-dimensional data in low-dimensional spaces. One such nonlinear algorithm to do so is from Lawrence, NIPS 2004, called GP-LVM. GP-LVM optimizes the maximum-likelihood of a probabilistic model. We won't get into the details here, but rather to the bottom line: in this paper, a log-likelihood has to be differentiated with respect to a matrix to derive the optimal parameters.

   To do so, we will apply the chain rule for multivariate derivatives via backpropagation. The log-likelihood is:

   $$\mathcal{L} = -c - \frac{D}{2} \log |\mathbf{K}| - \frac{1}{2} \mathrm{tr}(\mathbf{K}^{-1} \mathbf{Y} \mathbf{Y}^T)$$

where $\mathbf{K} = \alpha\mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}$ and c is a constant. The $|.|$ symbol in this context refers to the determinant of a matrix. To solve this, we'll take the derivatives with respect to the two terms with dependencies on $\mathbf{X}$:

$$\begin{aligned}\mathcal{L}_1 &= -\frac{D}{2}\log|\alpha\mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}| \\ \mathcal{L}_2 &= -\frac{1}{2}\text{tr}\left((\alpha\mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I})^{-1}\mathbf{Y}\mathbf{Y}^T\right)\end{aligned}$$

Hint: To receive full credit, you will be required to show all work. You may use the following matrix derivative without proof:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{K}} = -\mathbf{K}^{-T}\frac{\partial\mathcal{L}}{\partial\mathbf{K}^{-1}}\mathbf{K}^{-T}.$$

Also, consider the matrix operation, $\mathbf{Z} = \mathbf{X}\mathbf{Y}$. If we have an upstream derivative, $\partial\mathcal{L}/\partial\mathbf{Z}$, then backpropagate the derivatives in the following way:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{X}} &= \frac{\partial\mathcal{L}}{\partial\mathbf{Z}}\mathbf{Y}^T \\ \frac{\partial\mathcal{L}}{\partial\mathbf{Y}} &= \mathbf{X}^T\frac{\partial\mathcal{L}}{\partial\mathbf{Z}}\end{aligned}$$

(a) (3 points) Draw a computational graph for $\mathcal{L}_1$.

(b) (6 points) Compute $\frac{\partial\mathcal{L}_1}{\partial\mathbf{X}}$.

(c) (3 points) Draw a computational graph for $\mathcal{L}_2$.

(d) (6 points) Compute $\frac{\partial\mathcal{L}_2}{\partial\mathbf{X}}$.

(e) (2 points) Compute $\frac{\partial\mathcal{L}}{\partial\mathbf{X}}$.

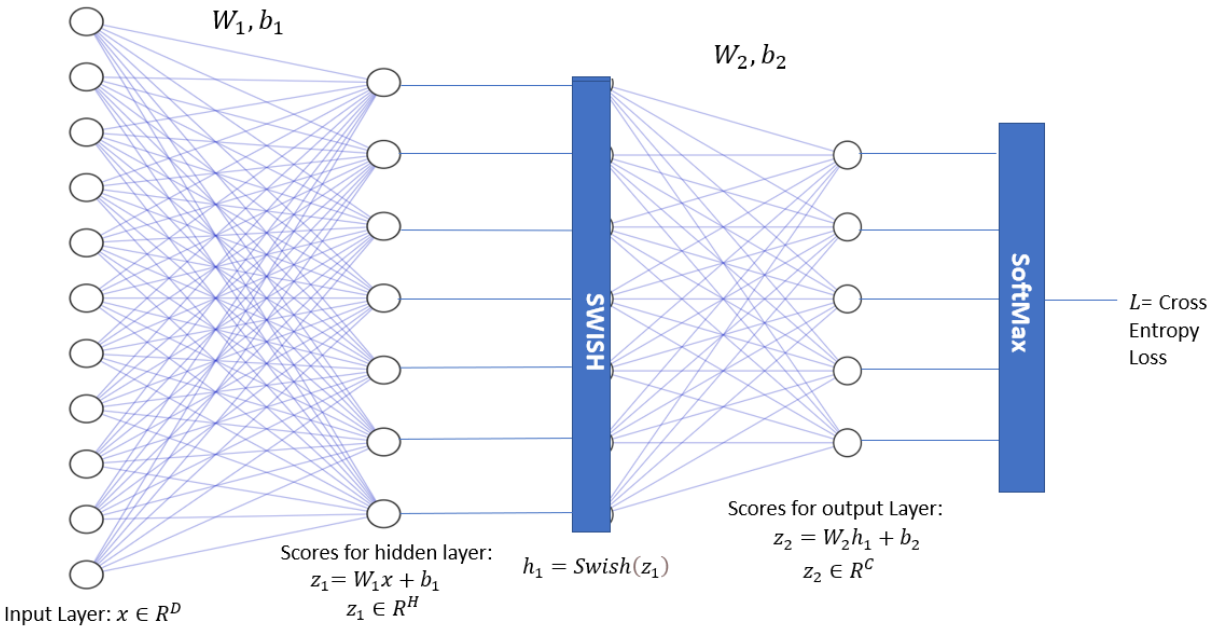
3. (15 points) **NNDL to the rescue!!**

It looks like a calm Monday morning and you are almost done with NNDL HW for the week (sigh)! But then suddenly (tring tring ...) your phone starts buzzing, you pick up the call, and the person from the other end sounds tense. The person exclaims ... there is a national emergency!!

7 different Pandora creature species (from Avatar) have been spotted in 1000's of numbers across various places in the country. They are having a hard time adjusting to the earth's climate and are causing chaos. As a result there has been a power outage in many cities. Luckily LA is an exception. UCLA's engineering division is helping out with this emergency, and you have been summoned to help.

You quickly take a bird to the secret facility and meet with director in charge of this operation. The director gives you a dataset consisting of images of these creatures along with their species type and instructs you to design a machine learning model to classify the images into species type. The only design constraint that the director has imposed is that the model should not have a very large number of parameters because some of UCLA's compute facilities are overloaded due to the power outages.

You just learned about fully connected neural networks (FC net) in class and decide to use it for accomplishing the task. To satisfy the design constraint, you decide to build a 2-layer FC net and train it using the provided dataset. The trained model will not only enable you to classify the images into species type but the hidden representations (outputs of intermediate layers) can be used to analyze the various properties of the species. A pictorial representation of the 2-layer FC net is shown above:



In the architecture shown, D represents the number of neurons in input layer, H represents the number of neurons in hidden layer, and C represents the number of neurons in the output layer (in our design $C = 7$). The output is then passed through a softmax classifier. Although we learned about the ReLu activation in class, we decided to use the Swish activation function (introduced by Google Brain) for the hidden layer. The Swish activation function for any scalar input k is defined as,

$$\text{swish}(k) = \frac{k}{1 + e^{-k}} = k\sigma(k),$$

where $\sigma(k)$is the sigmoid activation function you have seen in lecture.

You will train the 2-layer FC net using gradient descent and for that you will need to compute the gradients. For the gradient computations, you are allowed to keep your final answer in terms of $\frac{\partial L}{\partial z_2}$.

(a) (3 points) Draw the computational graph for the 2-layer FC net.

(b) (5 points) Compute $\nabla_{W_2} L$, $\nabla_{b_2} L$.

(c) (7 points) Compute $\nabla_{W_1} L$, $\nabla_{b_1} L$.

4. (30 points) **2-layer neural network.**

   Complete the two-layer neural network Jupyter notebook. Print out the entire notebook and relevant code and submit it as a pdf to gradescope. Download the CIFAR-10 dataset, as you did in HW #2.

5. (20 points) **General FC neural network.**

   Complete the FC Net Jupyter notebook. Print out the entire notebook and relevant code and submit it as a pdf to gradescope.