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.

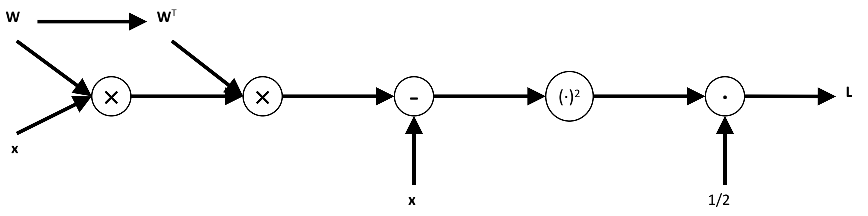
   For example, consider $\mathbf{x} \in \mathbb{R}^n$. Further, consider $\mathbf{W} \in \mathbb{R}^{m \times n}$ where $m < n$. Then $\mathbf{Wx}$ is of lower dimensionality than $\mathbf{x}$. One way to design $\mathbf{W}$ so that it still contains key features of $\mathbf{x}$ is to minimize the following 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 $\mathbf{x}$.
   Solution: With the loss to be minimized, the difference between output $\mathbf{W}^T\mathbf{Wx}$ and input $\mathbf{x}$ will be minimized. So the hidden representation $\mathbf{Wx}$ will preserve the information about $\mathbf{x}$.

   (b) (3 points) Draw the computational graph for $\mathcal{L}$.
   Solution:



   (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.
   Solution: Take a simple example. If the computational graph is composed of two path: $a \to b \to d \to e$ and $a \to c \to d \to e$. In this case, a contributes to e along two paths. Hence, the total derivative of e with respect to as is given by:

$$\frac{\partial e}{\partial a} = \frac{\partial e}{\partial d}\cdot\frac{\partial d}{\partial b}\cdot\frac{\partial b}{\partial a} + \frac{\partial e}{\partial d}\cdot\frac{\partial d}{\partial c}\cdot\frac{\partial c}{\partial a}$$

(d) (6 points) Calculate the gradient: $\nabla_{\mathbf{W}}\mathcal{L}$.
Solution: Let $\mathbf{K} = \mathbf{W}^T\mathbf{W}\mathbf{x} - \mathbf{x} \in \mathbb{R}^n$ and $\mathbf{P} = \mathbf{W}\mathbf{x} \in \mathbb{R}^m$, so

$$\frac{\partial \mathcal{L}}{\partial \mathbf{K}} = \mathbf{K}$$

Backpropagate to $\mathbf{W}^T$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^T} = \mathbf{K}(\mathbf{W}\mathbf{x})^T$$

Backpropagate to $\mathbf{P}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{P}} = \mathbf{W}\mathbf{K}$$

Backpropagate to $\mathbf{W}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{W}\mathbf{K}\mathbf{x}^T$$

$\nabla_{\mathbf{W}}\mathcal{L} =$ backprop to $\mathbf{W}$ + backprop to $\mathbf{W}^T$

$$\nabla_{\mathbf{W}}\mathcal{L} = \mathbf{W}\mathbf{K}\mathbf{x}^T + \mathbf{W}\mathbf{x}\mathbf{K}^T$$

2. (20 points) **Backpropagation for Gaussian-process latent variable model. Backpropagation for Gaussian-process latent variable model. (Optional for students in C147: Please write 'I am a C147 student' in the solution** 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 use apply the chain rule for multivariate derivatives via backpropagation. The log-likelihood is:

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

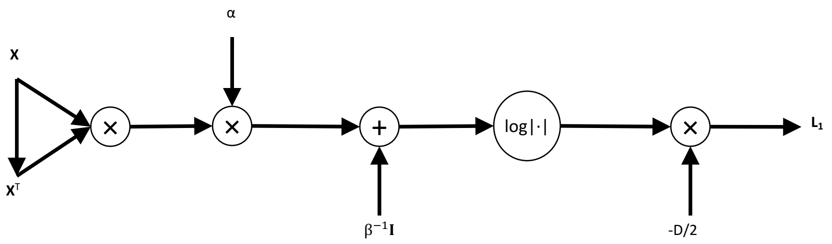
where $\mathbf{K} = \alpha\mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}$.The $|.|$ symbol in this context refers to the determinant of a matrix. To solve this, we'll take the derivatives with respect to the two terms with dependencies on $\mathbf{X}$:

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

Hint: Lawrence states the gradients in his paper, so you can check your answer. To receive full credit, you will be required to show all work. You may find following equation useful:

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

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



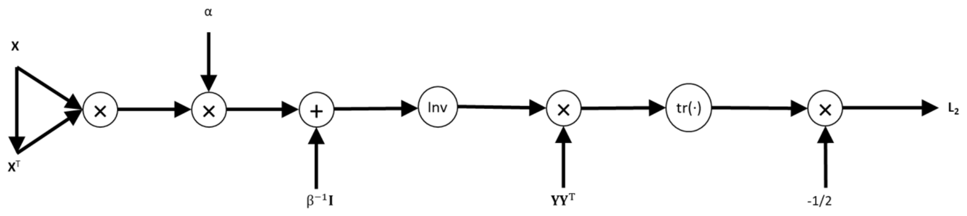
(b) (6 points) Compute $\frac{\partial \mathcal{L}_1}{\partial \mathbf{X}}$.
Solution: Calculate the derivative by backpropagation.

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{K}} = \frac{-D}{2}(\mathbf{K}^T)^{-1}$$

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{X}\mathbf{X}^T} = \frac{-\alpha D}{2}(\mathbf{K}^T)^{-1}$$

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{X}} = -\alpha D(\mathbf{K}^T)^{-1}\mathbf{X} = -\alpha D\mathbf{K}^{-1}\mathbf{X}$$

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



(d) (6 points) Compute $\frac{\partial \mathcal{L}_2}{\partial \mathbf{X}}$.
Solution: Calculate the derivative by backpropagation.

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T} = -\frac{1}{2}\mathbf{I}$$

(Using derivatives for product of matrices, we can write)

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{K}^{-1}} = -\frac{1}{2}\mathbf{Y}\mathbf{Y}^T$$

Substituting , $\frac{\partial L_2}{\partial \mathbf{K}^{-1}} = -\frac{1}{2}\mathbf{Y}\mathbf{Y}^T$ in the equation given in hint

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{K}} = \frac{1}{2}\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\mathbf{K}^{-1}$$

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{X}\mathbf{X}^T} = \frac{1}{2}\alpha\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\mathbf{K}^{-1}$$

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{X}} = \alpha\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\mathbf{K}^{-1}\mathbf{X}$$

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

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \alpha \mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T\mathbf{K}^{-1}\mathbf{X} - \alpha D \mathbf{K}^{-1}\mathbf{X}$$

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:

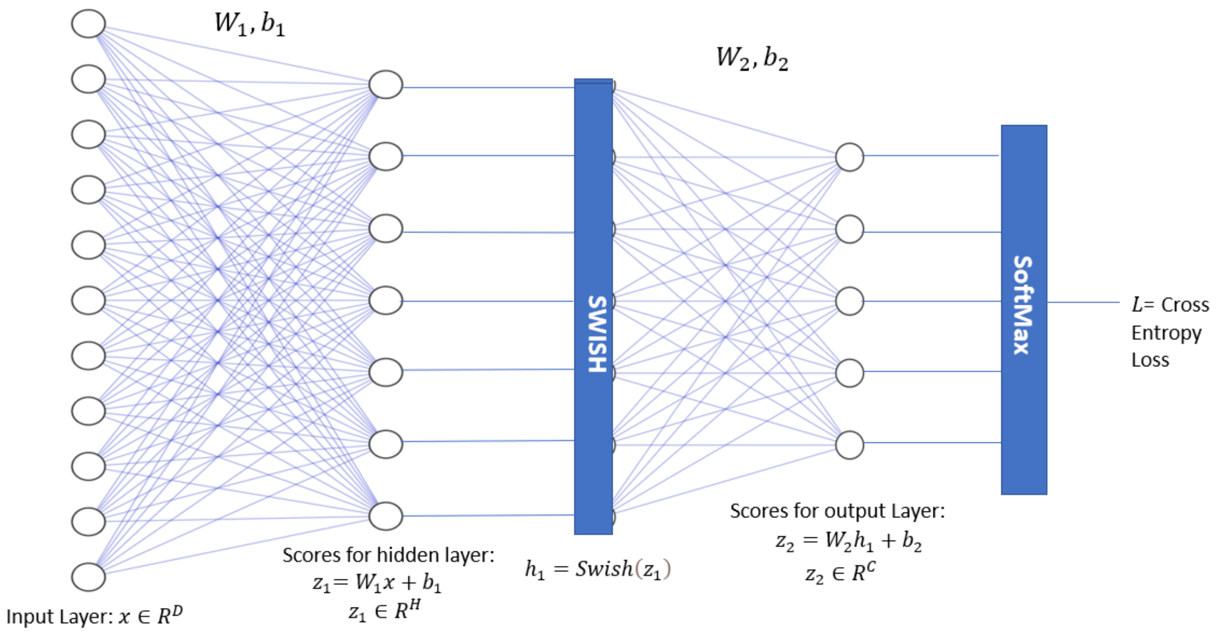
In the architecture shown, D represents the number of neurons in input layer, H represents the number of neurons in hidden layer , C represents the number of neurons in the output layer (in our design $C = 7$). The output is then passed through a softmax classifier. Although we learned about the ReLu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

$$\text{swish}(k) = \frac{k}{1+e^{-k}} = k\sigma(k),$$
$$\text{where, } \sigma(k), \text{is the sigmoid activation function you have seen in lectures}$$

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 $\nabla_{W_2} L$, $\nabla_{b_2} L$.

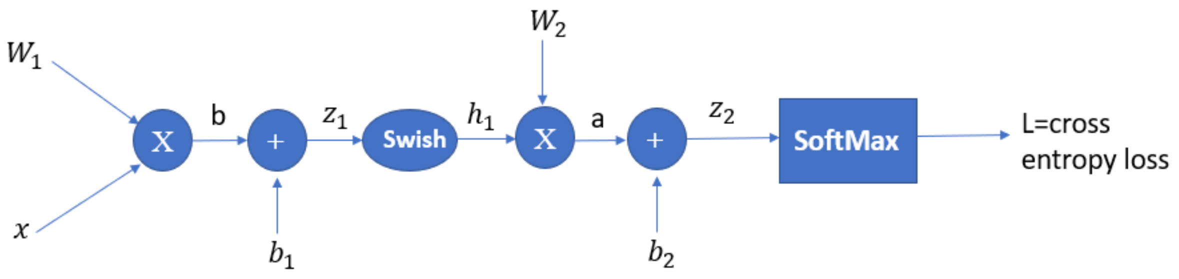


W_1, b_1
W_2, b_2
SWISH
SoftMax
L= Cross Entropy Loss
Scores for hidden layer: $z_1 = W_1x + b_1$ $z_1 \in R^H$
$h_1 = Swish(z_1)$
Scores for output Layer: $z_2 = W_2h_1 + b_2$ $z_2 \in R^C$
Input Layer: $x \in R^D$

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

Solution:

(a) (3 points) The computational graph can be found below



(b) (5 points) Lets name the intermediate edges on the computational graph as a,b. We know from HW2 that , the value ,
$\frac{\partial L}{\partial z_2} = \text{softmax}(z_2) - e_k,$
where e_k is vector of size same as $z_2 \in R^C$ and it has 1 in the k th position which represents the class of the given x. (We will not take away points if you have written your answer in terms of $\frac{\partial L}{\partial z_2}$)

- "+" sign equally distributes the gradient so $\frac{\partial L}{\partial a} = \frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_2}$, $\in R^C$

- From chain rule ,

$$\frac{\partial L}{\partial W_2} = \frac{\partial a}{\partial W_2}\frac{\partial L}{\partial a}$$

from 3D tensor derivatives learnt in class this simplifies to

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial a}h_1^T = \frac{\partial L}{\partial z_2}h_1^T, \in R^{C\times H}$$

(c) (7 points)

- Now to back propagate to z_1 , we need compute $\nabla_{h_1} L$, so again using chain rule ,

$$\frac{\partial L}{\partial h_1} = \frac{\partial a}{\partial h_1}\frac{\partial L}{\partial a}$$

based on vector to vector derivative from class

$$\frac{\partial L}{\partial h_1} = W_2^T\frac{\partial L}{\partial z_2}, \in R^H.$$

- From chain rule, $\frac{\partial L}{\partial z_1} = \frac{\partial h_1}{\partial z_1}\frac{\partial L}{\partial h_1}$.
- Lets calculate $\frac{\partial h_1}{\partial z_1}$,

  Let,$h_{11}, h_{12} \dots h_{1H}$be individual elements of vector h_1

  Let,$z_{11}, z_{12} \dots z_{1H}$be individual elements of vector z_1

  Lets compute the derivative for

  $\frac{\partial h_{11}}{\partial z_{11}} = \frac{\partial \text{swish}(z_{11})}{\partial z_{11}}$

  $= \frac{\partial z_{11}\sigma(z_{11})}{\partial z_{11}}$

  $= z_{11}\sigma(z_{11})(1-\sigma(z_{11})) + \sigma(z_{11}), \text{product rule}$

  $= z_{11}\sigma(z_{11}) + \sigma(z_{11})(1 - z_{11}\sigma(z_{11}))$

  $= \text{swish}(z_{11}) + \sigma(z_{11})(1-\text{swish}(z_{11}))$

  $= h_{11} + \sigma(z_{11})(1-h_{11})$.

  Now the above is a scalar derivative.

  The derivative$\frac{\partial h_1}{\partial z_1}$would be a HxH diagonal matrix

  $\frac{\partial h_1}{\partial z_1} = diag(h_{11} + \sigma(z_{11})(1-h_{11}), h_{12} + \sigma(z_{12})(1-h_{12}), \dots h_{1H} + \sigma(z_{1H})(1-h_{1H})) \in R^{H\mathbf{x}H}$

  $\frac{\partial L}{\partial z_1} = (h_1 + \sigma(z_1)\odot(1-h_1))\odot\frac{\partial L}{\partial h_1}, \in R^H$

  $\frac{\partial L}{\partial z_1} = (h_1 + \sigma(z_1)\odot(1-h_1))\odot(W_2^T\frac{\partial L}{\partial z_2}), \in R^H$

- "+" again equally distributes the gradient , $\frac{\partial L}{\partial z_1} = \frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial b}$, $\in R^H$
- From chain rule we can write,

  $$\frac{\partial L}{\partial W_1} = \frac{\partial b}{\partial W_1}\frac{\partial L}{\partial b}$$

  From 3D tensor derivatives learnt in class this simplifies to,

  $$\frac{\partial L}{\partial W_1} = ((h_1 + \sigma(z_1)\odot(1-h_1))\odot(W_2^T\frac{\partial L}{\partial z_2}))x^T, \in R^{H\times D}$$

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.

   Solution: As a policy we don't release solutions to coding questions.

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.
   Solution: As a policy we don't release solutions to coding questions.