# HW: Deep Multilayer Perceptrons

1. Explain, using a formal derivation of the delta rule, why using the cross-entropy error function instead of the sum-of-squares error function for classification problems leads to:
   (1) Faster Training: Derive the gradients of both the cross-entropy and sum-of-squares loss functions with respect to the weights, and explain how the nature of these gradients contributes to quicker convergence during the training of a neural network.
   (2) Improved Generalization: Based on the properties of the derived gradients, discuss why the cross-entropy loss function might lead to better generalization in a classification model compared to the sum-of-squares error function.

2. In multi-class classification, why is the softmax function typically used to produce a probability distribution over classes instead of applying separate sigmoid functions to each output and normalizing them afterward? Explain how softmax differs from this approach and why it's generally preferred.

3. Batch Normalization
   (1) Explain how batch normalization can alleviate the problem of covariate shift in neural networks. Discuss the mechanism by which batch normalization normalizes layer inputs and its effect on the training process.
   (2) What would be the effect of removing the scale ($\gamma$) and shift ($\beta$) parameters from a batch normalization layer? Discuss how this alteration might impact the model's ability to learn complex functions.
   (3) Describe how batch normalization can act as a form of regularization during training. What does this imply about its impact on model generalization and overfitting?
   (4) How does the choice of batch size influence the effectiveness of batch normalization? Discuss potential issues that may arise with very small or large batch sizes, and their implications on the normalization process.

4. Skip connection
   (1) How do skip connections in neural networks mitigate the problem of

vanishing gradients?

(2) Discuss the advantages and disadvantages of using addition and concatenation for combining features in residual connections of neural networks. Provide insights into scenarios where each method would be preferable.

(3) Explain why the BERT (Bidirectional Encoder Representations from Transformers) architecture adopts addition for its skip connections rather than concatenation. Discuss the implications of this design choice on the model's performance and training efficiency.

(4) Hypothesize how the BERT model's architecture and performance might be affected if it were to use concatenation in its skip connections instead of addition. Consider aspects such as parameter count, computational complexity, and potential impacts on learning.

(5) Propose and justify an alternative method (other than addition and concatenation) for combining features in the skip connections of BERT. Discuss how this method could potentially benefit or pose challenges to the BERT architecture.

5. Explain why minimizing the cross-entropy loss in multi-class classification is equivalent to maximizing the likelihood of the observed labels under a multinomial distribution assumption. Illustrate how this statistical interpretation aligns with the learning process in a multi-layer perceptron (MLP) model.

6. Imagine a machine learning classifier that predicts four categories (A, B, C, D) for image data. After training, the classifier's predicted probabilities for a specific image are as follows: P(A) = 0.4, P(B) = 0.3, P(C) = 0.2, P(D) = 0.1 (assuming a softmax output layer used).

(1) Calculate Self-Information: Compute the self-information (in bits) for the classifier predicting each category for this image.

(2) Calculate the entropy of the classifier’s predictions. This represents the average uncertainty in the classifier's predictions for this image.

(3) Assume the true label distribution for this image is Q(A) = 0.3, Q(B) = 0.3, Q(C) = 0.3, Q(D) = 0.1. Calculate the cross-entropy between the classifier's predicted distribution (P) and the true distribution (Q).

(4) KL Divergence: Determine the Kullback-Leibler divergence from the classifier's predicted distribution (P) to the true distribution (Q). Discuss what this value represents in terms of the classifier's performance.

7. Given the following target (ground truth) classes and predicted probabilities for Class 1:

   Target (Ground Truth) Classes: [1, 0, 1, 0, 1, 0, 0, 1, 1, 0]

   Predicted Probabilities (Sigmoid Output) for Class 1:
   [0.8, 0.6, 0.75, 0.4, 0.9, 0.3, 0.2, 0.7, 0.85, 0.5]

   (1) Calculate the True Positive Rate (TPR) and False Positive Rate (FPR) for different threshold values ranging from 0.1 to 1.0 with increments of 0.1. Arrange your results in the table below:

| Threshold | True Positive | False Positive | TPR | FPR |
|---|---|---|---|---|
| 0.1 | | | | |
| 0.2 | | | | |
| 0.3 | | | | |
| 0.4 | | | | |
| 0.5 | | | | |
| 0.6 | | | | |
| 0.7 | | | | |
| 0.8 | | | | |
| 0.9 | | | | |
| 1.0 | | | | |

   (2) Plot the Receiver Operating Characteristic (ROC) curve by connecting the calculated points.

   (3) Roughly estimate the Area Under the ROC Curve (AUCROC).

8. Use the deep Multi-Layer Perceptron (MLP) code in Keras provided in this Colab notebook to perform a classification task on the Iris dataset: (https://colab.research.google.com/drive/1qO6YgenqYLiPqr9OGabUQwBMzid7rBAX?usp=drive_link).

   You can download the Iris dataset from (https://www.kaggle.com/datasets/saurabh00007/iriscsv?resource=download).

Adapt the code to perform a different classification task using your own dataset. Choose a dataset that's relevant to your job or of personal interest. You may need to make adjustments based on the structure of your new dataset.
Note: The code covers all topics introduced in the lecture, such as data preprocessing, model definition, and evaluation. Feel free to make improvements to enhance classification performance. You can experiment with different architectures, parameters, or preprocessing techniques as discussed.

(1) List your program and its Colab URL and output snapshots.
(2) Make performance comparisons of sigmoid and ReLU as the activation functions for the hidden nodes and describe your findings.
(3) Perform a comprehensive experiment by considering different configurations of the deep MLP based on the above minimal requirements. Discuss and analyze the experimental results to highlight whatever insights or knowledge you identified. You should copy and paste necessary the screen snapshots of outputs to support your arguments.
(4) Describe under what kind situation the overfit could happen and how you solve it.

9. [Extra points] Challenging exercise.
Problem Title: Designing an Exponential Weight MLP for Iris Classification

Problem Description:

You are tasked with designing a Multi-Layer Perceptron (MLP) neural network for the classification of the Iris dataset. However, there's a unique twist to this challenge. Your goal is to design an MLP architecture and derive a custom learning rule that ensures all weights in the network remain positive throughout the training process.

Problem Components:

(1) MLP Architecture Design:

Design a feedforward neural network architecture for the classification of the Iris dataset. The architecture should consist of an input layer, one or more hidden layers, and an output layer. However, all weights in the network (including those in hidden layers) should be exponentialized during training.

(2) Derive the Learning Rule: Develop a custom learning rule that allows the network to learn and adapt its weights while ensuring they remain positive. You should provide a mathematical derivation or description of this learning rule, explaining how it guarantees positive weights.

(3) Implementation: Implement your designed MLP architecture along with the custom learning rule in a programming language of your choice. Use this implementation to train the network on the Iris dataset for classification.

(4) Evaluation: Evaluate the performance of your trained network on the Iris dataset using appropriate evaluation metrics such as accuracy, precision, recall, and F1-score.

(5) Documentation: Provide clear documentation of your MLP architecture, the derivation of the learning rule, and the implementation code. Explain any assumptions or considerations made during the design.

(6) Discussion: Discuss the advantages and potential limitations of your approach compared to conventional MLP training methods. Reflect on the challenges faced and the insights gained from this unique problem.

(7) Deliverables:

- A report or documentation containing detailed explanations, mathematical derivations, and code.
- The code implementing the designed MLP and learning rule.
- Performance evaluation results on the Iris dataset.

Note: This problem is challenging and requires creative thinking to

design a neural network and learning rule that meet the specified criteria. It encourages students to explore unconventional techniques in neural network training.