Machine Learning

Instructors: Mohammadreza A. Dehaqani, Babak n. Arabi, Mostafa Tavassolipour

Mohammad Javad Mousavi, Mehrnaz Hosseini

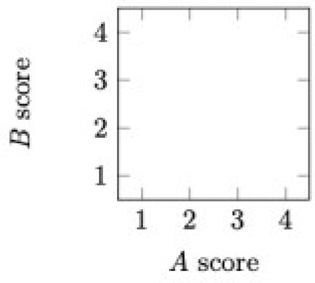
Fall 2025

Homework 3

Q1. Perceptron

You want to predict if movies will be profitable based on their screenplays. You hire two critics A and B to read a script you have and rate it on a scale of 1 to 4. The critics are not perfect; here are five data points including the critics' scores and the performance of the movie:

#	Movie Name	A	B	Profit?
1	Pellet Power	1	1	−
2	Ghosts!	3	2	+
3	Pac is Bac	2	4	+
4	Not a Pizza	3	4	+
5	Endless Maze	2	3	−



1. First, you would like to examine the linear separability of the data. Plot the data on the 2D plane above; label profitable movies with + and non-profitable movies with − and determine if the data are linearly separable.

2. Now you decide to use a perceptron to classify your data. Suppose you directly use the scores given above as features, together with a bias feature. That is $f_0 = 1$, f_1 = score given by A and f_2 = score given by B.

   Run one pass through the data with the perceptron algorithm, filling out the table below. Go through the data points in order, e.g. using data point #1 at step 1.

step	Weights	Score	Correct?
1	$[-1, 0, 0]$	$-1 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = -1$	yes
2			
3			
4			
5			

   Final weights:

3. Have weights been learned that separate the data?

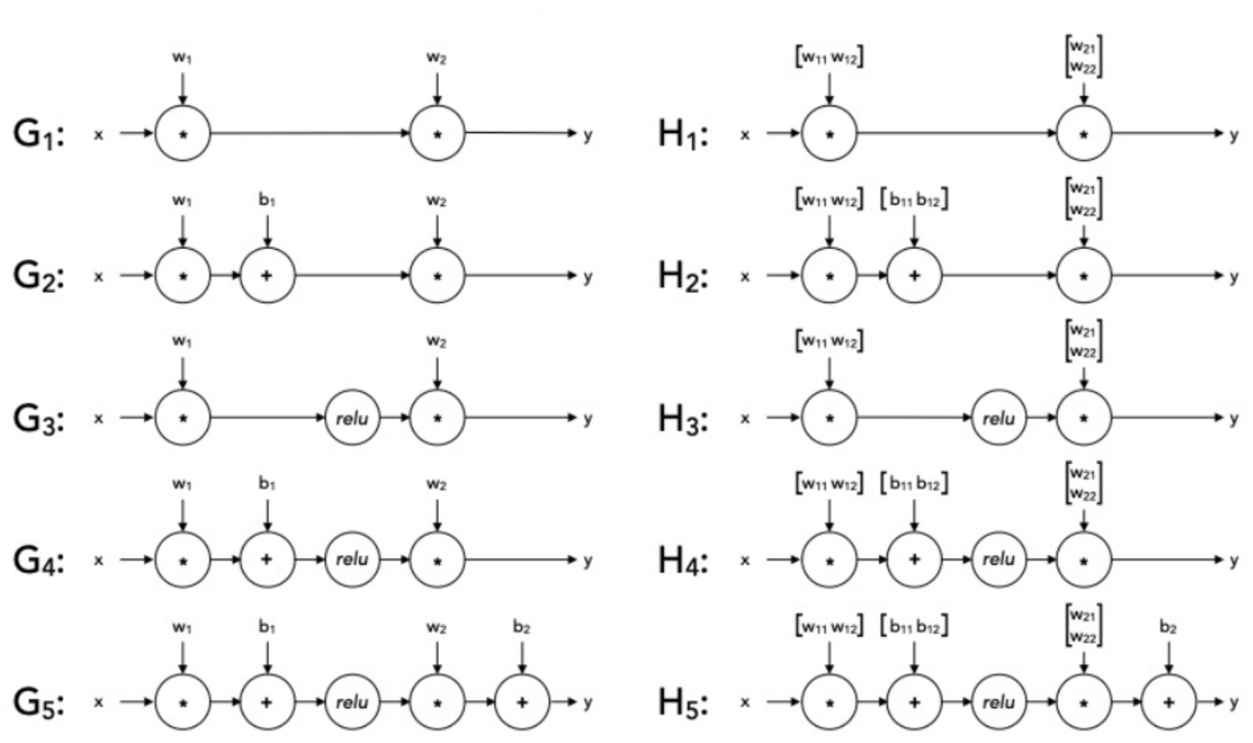
4. More generally, irrespective of the training data, you want to know if your features are powerful enough to allow you to handle a range of scenarios. Circle the scenarios for which a perceptron using the features above can indeed perfectly classify movies which are profitable according to the given rules:

(a) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be profitable, and otherwise it won't be.

(b) Your reviewers are art critics. Your movie will be profitable if and only if each reviewer gives either a score of 2 or a score of 3.

(c) Your reviewers have weird but different tastes. Your movie will be profitable if and only if both reviewers agree.

Q2. Neural Networks: Representation

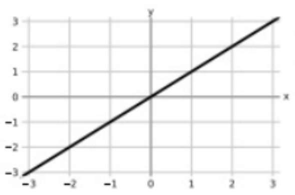


For each of the piecewise-linear functions below, mark all networks from the list above that can represent the function exactly on the range $x \in (-\infty, \infty)$. In the networks above, *relu* denotes the element-wise ReLU nonlinearity:

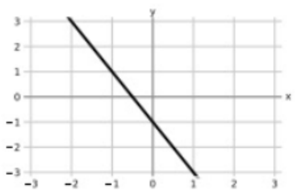
$$\text{relu}(z) = \max(0, z).$$

The networks G_k have 1-dimensional layers, while the networks H_k have some 2-dimensional intermediate layers.

(a)

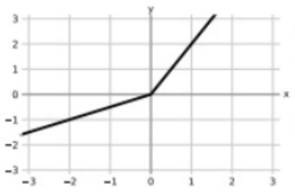


☐ G_1 ☐ H_1
☐ G_2 ☐ H_2
☐ G_3 ☐ H_3
☐ G_4 ☐ H_4
☐ G_5 ☐ H_5
○ None of the above

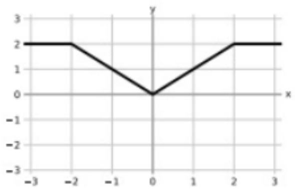


☐ G_1 ☐ H_1
☐ G_2 ☐ H_2
☐ G_3 ☐ H_3
☐ G_4 ☐ H_4
☐ G_5 ☐ H_5
○ None of the above

(b)

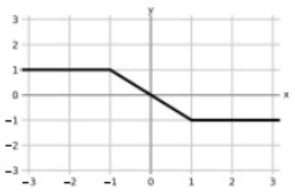


☐ G_1 ☐ H_1
☐ G_2 ☐ H_2
☐ G_3 ☐ H_3
☐ G_4 ☐ H_4
☐ G_5 ☐ H_5
○ None of the above

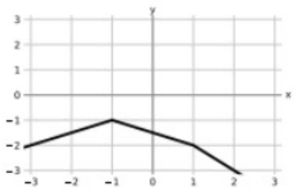


☐ G_1 ☐ H_1
☐ G_2 ☐ H_2
☐ G_3 ☐ H_3
☐ G_4 ☐ H_4
☐ G_5 ☐ H_5
○ None of the above

(c)



☐ G_1 ☐ H_1
☐ G_2 ☐ H_2
☐ G_3 ☐ H_3
☐ G_4 ☐ H_4
☐ G_5 ☐ H_5
○ None of the above



☐ G_1 ☐ H_1
☐ G_2 ☐ H_2
☐ G_3 ☐ H_3
☐ G_4 ☐ H_4
☐ G_5 ☐ H_5
○ None of the above

Q3. Neural Networks: Forward and Backward Propagation

Consider a two-layer neural network defined as follows:

$$\begin{aligned}
z_1 &= W_1 x^{(i)} + b_1, \\
a_1 &= \mathrm{ReLU}(z_1), \\
z_2 &= W_2 a_1 + b_2, \\
\hat{y}^{(i)} &= \sigma(z_2), \\
L^{(i)} &= y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right), \\
J &= -\frac{1}{m} \sum_{i=1}^{m} L^{(i)}.
\end{aligned}$$

Note that $x^{(i)}$ is a single input example with shape $D_x \times 1$, and $y^{(i)}$ is a scalar label. The dataset contains m examples. We use D_{a_1} hidden units in the hidden layer, so z_1 has shape $D_{a_1} \times 1$.

1. What are the dimensions of W_1, b_1, W_2, and b_2? If we want to *vectorize* the network over multiple examples, how do the shapes of the weights and biases change? If we vectorize over multiple input examples, what will be the shapes of X and Y?
2. What is $\frac{\partial J}{\partial \hat{y}^{(i)}}$? Denote this quantity by $\delta_1^{(i)}$. Using this result, what is $\frac{\partial J}{\partial \hat{y}}$?
3. What is $\frac{\partial \hat{y}^{(i)}}{\partial z_2^{(i)}}$? Denote this quantity by $\delta_2^{(i)}$.
4. What is $\frac{\partial z_2}{\partial a_1}$? Denote this quantity by $\delta_3^{(i)}$.
5. What is $\frac{\partial a_1}{\partial z_1}$? Denote this quantity by $\delta_4^{(i)}$.
6. What is $\frac{\partial z_1}{\partial W_1}$? Denote this quantity by $\delta_5^{(i)}$.
7. What is $\frac{\partial J}{\partial W_1}$? You may use the previous results.

Question 4. Hessian of a Two-Layer Network

Consider a two-layer neural network (one hidden layer). Let the forward relations be

$$a_j = \sum_i w_{ji}^{(1)} x_i, \qquad z_j = h(a_j), \tag{0.1}$$

$$a_k = \sum_j w_{kj}^{(2)} z_j, \tag{0.2}$$

and let E_n denote the error for data point n.

Define

$$\delta_k \equiv \frac{\partial E_n}{\partial a_k}, \qquad M_{kk'} \equiv \frac{\partial^2 E_n}{\partial a_k \, \partial a_{k'}}. \tag{0.3}$$

Derive expressions for the elements of the Hessian (second derivatives of E_n with respect to weights) in terms of δ_k and $M_{kk'}$ for the following cases:

1. both weights are in the second layer $\left(w^{(2)}\right)$,
2. both weights are in the first layer $\left(w^{(1)}\right)$,
3. one weight is in each layer (mixed second derivative).

Question 5. Linear Discriminant Analysis (LDA)

Consider two sets of observations

$$Y_1 = \{y_i\}_{i=1}^{n_1}, \qquad Y_2 = \{y_j\}_{j=1}^{n_2}.$$

Let J denote the average squared difference between all pairs of observations from the two sets, defined as

$$J = \frac{1}{n_1 n_2} \sum_{y_i \in Y_1} \sum_{y_j \in Y_2} (y_i - y_j)^2.$$

Show that

$$J = (m_1 - m_2)^2 + \frac{1}{n_1} s_1^2 + \frac{1}{n_2} s_2^2.$$

Question 6. Feature Conditioning and PCA

In the context of feature conditioning methods, several approaches taught in class are supervised and require labeled data. There also exist unsupervised methods that perform dimensionality reduction without using labels.

One such method is Principal Component Analysis (PCA). Consider the one-dimensional case of PCA. Formulate the optimization problem for finding a single principal component and show that solving this optimization problem leads to the eigenvalue equation

$$Sw = \lambda w,$$

where S denotes the data covariance matrix. Clearly state the objective function and the constraint, and derive the above eigenvalue equation.

Question 7. Feature Conditioning (Computational Exercise)

In machine learning, feature conditioning refers to a set of preprocessing techniques applied to data in order to improve model performance, stability, and learning speed. These techniques include, but are not limited to, normalization, standardization, dimensionality reduction, feature selection, feature extraction, and handling missing values.

The classification accuracy is measured using the Correct Classification Rate (CCR), defined as

$$\text{CCR} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Samples}} \times 100.$$

Using the attached dataset, perform feature selection for a Naive Bayes classifier by applying both Forward Selection and Backward Elimination. At each step, evaluate the model performance using CCR.

Report the selected features, the corresponding CCR values, and identify the feature subset that achieves the best performance.

Dataset link

Question 8. Computational Exercise: Flower Image Classification

In this computational exercise, the goal is to classify flower images into **five classes** using neural networks and transfer learning techniques. The dataset is provided in a ready-to-use format and is already split into `train` and `validation` sets.

The dataset directory structure is as follows:

- `train/` containing five subfolders (each corresponding to one flower class)
- `validation/` containing five subfolders with the same class names

Dataset link.

General Requirements (Mandatory)

1. Use classification accuracy as the primary evaluation metric.
2. Training and validation procedures must be implemented **manually** (i.e., explicitly write training and validation loops). The use of high-level training APIs such as `model.fit()`, PyTorch Lightning, or similar frameworks is not allowed.
3. Plot and report training and validation loss and accuracy curves.

Part I: Training from Scratch

1. **Linear Model (No Activation Functions)**
   Implement and train a fully linear neural network for 5-class classification. No activation functions are allowed in intermediate layers. A `Softmax` layer must be used at the output. Report training and validation accuracy.
2. **One Hidden Layer with ReLU**
   Implement and train a neural network with one hidden layer consisting of 128 neurons and a ReLU activation function. Compare its performance with the linear model.
3. **Improved Neural Network**
   Improve the model from the previous step by incorporating the following techniques:
   - Regularization (e.g., L2)
   - Batch Normalization
   - Dropout
   - Early Stopping

   Report and analyze the effect of these techniques on model performance.

Part II: Transfer Learning (Frozen Backbones)

For both of the following models, the backbone must be **completely frozen**, and only the classifier head may be trained.

1. **AlexNet (Frozen Backbone)**
   Use a pre-trained AlexNet model, freeze all backbone parameters, and replace the final classification layer to perform 5-class classification. Train only the classifier head and report validation performance.
2. **VGG16 (Frozen Backbone)**
   Repeat the above procedure using a pre-trained VGG16 model. Compare the results with AlexNet and with the models trained from scratch.

Deliverables

- Source code (notebook or script) with manually implemented training and validation loops
- Training and validation loss and accuracy plots
- A comparison table summarizing the performance of all models
- A brief analysis discussing the observed differences between models