# CMPE 49T Homework 1 - Fall 2024

Instructor: Mehmet Turan

October 21, 2024
Due: November 6, 2024

## Problem 1: Image Classification Using Convolutional Neural Networks

You are working with an image classification system that uses grayscale images. The system is designed to predict whether an image belongs to one of two classes: **Class 0** (representing a defect-free product) or **Class 1** (representing a defective product). Each image is represented as a 5x5 matrix, where the values correspond to pixel intensities. These pixel intensities can range from negative to positive values, reflecting the variation in lighting conditions or noise in the image capture process.

You are tasked with designing and evaluating a Convolutional Neural Network (CNN) that can classify these images. The network includes convolution, activation functions, pooling, and a fully connected layer. You will work through each step by hand, using a specific image as input.

## Given:

You are provided with the following data, Input image I:

$$I = \begin{bmatrix} -1 & 2 & -2 & 0 & 1 \\ 3 & -3 & 1 & 2 & 0 \\ -1 & 0 & 2 & -1 & 3 \\ 2 & 1 & -2 & -3 & 0 \\ -3 & 2 & 0 & 1 & -1 \end{bmatrix}$$

Two convolutional filters, $F_1$ and $F_2$, are given as:

$$F_1 = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad F_2 = \begin{bmatrix} -1 & 0 \\ 1 & 1 \end{bmatrix}$$

## Step 1: Convolution Operation

In this step, you will apply the two filters $F_1$ and $F_2$ to the input image $I$ using **valid padding** (no padding) and a stride of 1. For each filter, slide it across the image, multiply the overlapping values, and sum them to produce a 4x4 feature map. Show your work.

## Step 2: Activation Function

After performing the convolution, apply the following activation functions to the feature maps to introduce non-linearity: **ReLU**, **Sigmoid**, and **Tanh**.

### ReLU (Rectified Linear Unit)

The **ReLU** activation sets negative values to zero and leaves positive values unchanged:

$$\text{ReLU}(x) = \max(0, x)$$

### Sigmoid

The **Sigmoid** activation maps values to the range (0, 1):

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

### Tanh (Hyperbolic Tangent)

The **Tanh** activation maps values to the range (-1, 1):

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

## Step 3: Pooling Operation

In this step, you will apply three different pooling operations—**max pooling**, **min pooling**, and **average pooling**—to the feature maps obtained after applying the activation functions. Pooling helps reduce the size of the feature maps while retaining important information.

For each operation, the pooling window will be 2x2, and you will apply the operation to non-overlapping 2x2 blocks.

Here's an example for a 2x2 block:

$$\begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix}$$

For **max pooling**, the result is: $\max(4, 2, 1, 3) = 4$
For **min pooling**, the result is: $\min(4, 2, 1, 3) = 1$
For **average pooling**, the result is: $\frac{4+2+1+3}{4} = 2.5$

You will now apply these three pooling methods to the feature maps produced by the activation functions. Overall 9 matrices needs to be reported here. You can skip calculations and write the final results

**What to Do:**
1. Divide each 4x4 feature map into non-overlapping 2x2 blocks.
2. For **max pooling**, select the maximum value from each block.
3. For **min pooling**, select the minimum value from each block.
4. For **average pooling**, calculate the average of all values in each block.
5. Show your calculations and results for each pooling method on the feature maps from ReLU, Sigmoid, and Tanh

## Step 4: Fully Connected Layer

In this step, you will use the **ReLU-activated** pooled outputs from max pooling (Task 3.1) as input to a fully connected layer. The pooled 2x2 matrices will first be flattened into 1D vectors, and then used to compute the output for two neurons. Each neuron has a set of weights and a bias term.

The output of each neuron is computed as:

$$y = W \cdot X + b$$

Where:
- $W$ is the weight vector for the neuron.
- $X$ is the flattened input vector from the ReLU-activated max-pooled output.
- $b$ is the bias term for the neuron.

The weights and bias values for the neurons are as follows:
Neuron 1: $W_1 = [1, -1, 0, 1]$, $b_1 = 1$
Neuron 2: $W_2 = [0, 1, -1, -1]$, $b_2 = -1$

**Tasks:**
Flatten the 2x2 ReLU-activated max-pooled matrix from Task 3.1 into a 1D vector. Using the weight vector and bias, compute the output for **Neuron 1**. and **Neuron 2**. Show your work.

### Step 5: Classification Decision

In this step, you will use the outputs from Neuron 1 and Neuron 2 to classify the image by applying the **softmax** function. The softmax function converts the raw outputs (also called logits) into probabilities that sum to 1, which can be interpreted as the likelihood of the image belonging to each class.

The softmax function is defined as:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Where $z_1$ and $z_2$ are the outputs from Neuron 1 and Neuron 2, respectively.

- $z_1$ corresponds to the score for Class0.
- $z_2$ corresponds to the score for Class1.

Using the outputs $z_1$ and $z_2$ from Neuron 1 and Neuron 2 (computed in Step 4), apply the softmax function to compute the probabilities for **Class0** and **Class1**. Show your calculations step by step, ensuring that the probabilities sum to 1. Based on the softmax probabilities:

If the probability of Class0 is higher, predict that the image belongs to Class0. If the probability of Class1 is higher, predict that the image belongs to Class1. Clearly state the predicted class based on your calculations.

## Problem2: Gradient Descent with Ridge and Lasso for Nonlinear Regression

In this problem, you will manually perform two steps of gradient descent for a regression model with **nonlinear interaction terms** between two features. You will apply both **Ridge (L2)** and **Lasso (L1)** regularization and compute metrics to assess the performance of the model.

### Scenario:

You are given a dataset with 5 points, each with two features $x_1$ and $x_2$, and the corresponding outputs $y$:

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| $-1$ | 2 | 1.5 |
| 2 | $-2$ | 0.5 |
| $-2$ | 1 | $-1.0$ |
| 1 | 1 | 1.0 |
| $-1$ | $-1$ | $-1.5$ |

The hypothesis function (model) is:

$$h(x_1, x_2) = w_1 x_1 + w_2 x_2 + w_3 (x_1 \cdot x_2) + b$$

Where:

- $w_1$, $w_2$, and $w_3$ are the weights (slopes) for the features and interaction term.
- $b$ is the bias (intercept).

The loss function is the **regularized mean squared error (MSE)**, defined as:

$$L(w_1, w_2, w_3, b) = \frac{1}{N} \sum_{i=1}^{N} \left( h(x_1^{(i)}, x_2^{(i)}) - y_i \right)^2 + \lambda R(w_1, w_2, w_3)$$

Where:

- $N = 5$ is the number of data points.
- $\lambda = 0.1$ is the regularization parameter.
- $R(w_1, w_2, w_3)$ is the regularization term, which differs for Ridge and Lasso regression.

## Ridge Regression (L2 Regularization):

The regularization term for Ridge regression is:

$$R_{\text{Ridge}}(w_1, w_2, w_3) = w_1^2 + w_2^2 + w_3^2$$

## Lasso Regression (L1 Regularization):

The regularization term for Lasso regression is:

$$R_{\text{Lasso}}(w_1, w_2, w_3) = |w_1| + |w_2| + |w_3|$$

You will use both Ridge and Lasso regularization for this problem.

# Part 1: Ridge Regression (L2 Regularization)

## Task 1.1: Forward Pass (Compute Predictions)

Using the initial values $w_1 = 0$, $w_2 = 0$, $w_3 = 0$, and $b = 0$, calculate the predicted values $h(x_1, x_2)$ for each data point using the hypothesis function. Show your work.

## Task 1.2: Compute the Regularized MSE (Ridge)

Using the predictions from Task 1.1, calculate the regularized MSE loss for Ridge regression:

$$L_{\text{Ridge}}(w_1, w_2, w_3, b) = \frac{1}{5} \sum_{i=1}^{5} \left( h(x_1^{(i)}, x_2^{(i)}) - y_i \right)^2 + 0.1(w_1^2 + w_2^2 + w_3^2)$$

Show your detailed calculations for each data point.

## Task 1.3: Compute the Gradients (Ridge)

To minimize the regularized loss, compute the gradients of the Ridge loss function with respect to $w_1$, $w_2$, $w_3$, and $b$.

$$\frac{\partial L_{\text{Ridge}}}{\partial w_1} = \frac{2}{N} \sum_{i=1}^{N} \left( h(x_1^{(i)}, x_2^{(i)}) - y_i \right) x_1^{(i)} + 2\lambda w_1$$

$$\frac{\partial L_{\text{Ridge}}}{\partial w_2} = \frac{2}{N} \sum_{i=1}^{N} \left( h(x_1^{(i)}, x_2^{(i)}) - y_i \right) x_2^{(i)} + 2\lambda w_2$$

$$\frac{\partial L_{\text{Ridge}}}{\partial w_3} = \frac{2}{N} \sum_{i=1}^{N} \left( h(x_1^{(i)}, x_2^{(i)}) - y_i \right) (x_1^{(i)} \cdot x_2^{(i)}) + 2\lambda w_3$$

$$\frac{\partial L_{\text{Ridge}}}{\partial b} = \frac{2}{N} \sum_{i=1}^{N} \left( h(x_1^{(i)}, x_2^{(i)}) - y_i \right)$$

## Task 1.4: Perform Two Iterations of Gradient Descent (Ridge)

Perform two iterations of gradient descent using a learning rate $\alpha = 0.1$. Show your calculations for each update.

# Part 2: Lasso Regression (L1 Regularization)

## Task 2.1: Compute the Regularized MSE (Lasso)

Using the same predictions from Task 1.1, calculate the regularized MSE loss for Lasso regression:

$$L_{\text{Lasso}}(w_1, w_2, w_3, b) = \frac{1}{5}\sum_{i=1}^{5}\left(h(x_1^{(i)}, x_2^{(i)}) - y_i\right)^2 + 0.1(|w_1| + |w_2| + |w_3|)$$

## Task 2.2: Compute the Gradients (Lasso)

Compute the gradients of the Lasso loss function with respect to $w_1$, $w_2$, $w_3$, and $b$. Use the sign function for Lasso regularization:

$$\frac{\partial L_{\text{Lasso}}}{\partial w_1} = \frac{2}{N}\sum_{i=1}^{N}\left(h(x_1^{(i)}, x_2^{(i)}) - y_i\right)x_1^{(i)} + \lambda \text{sign}(w_1)$$

Repeat for $w_2$ and $w_3$.

# Part 3: Evaluate Model Performance

## Task 3.1: Mean Squared Error (MSE)

After completing the iterations, calculate the final MSE for both Ridge and Lasso regression.

## Task 3.2: Mean Absolute Error (MAE)

In addition to MSE, compute the Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|h(x_1^{(i)}, x_2^{(i)}) - y_i|$$

## Task 3.3: Convergence

Continue iterating until the model parameters converge. Report the final values of $w_1$, $w_2$, $w_3$, and $b$, and the final MSE and MAE for both Ridge and Lasso regression.

## Submission Guidelines

Show all your calculations, including intermediate steps for the forward pass, loss calculation, and gradient updates. Submit your final results after the model has converged. You can do this part with a simple python function. Submit the final values found with your convergence criteria used. Add your code at the end of your report.