Lab 5: Feedforward Neural Network

CS 3400 Machine Learning

Overview

In your previous courses, you were exposed to basic neural networks (NNs). Early forms of NNs were developed in the 1950's but have seen a resurgence of popularity in the last decade. The popularity is fueled by massive increases in data and compute power as well as new activation functions and optimization methods that solve the so-called "vanishing gradient" problem to enable the construction of large networks that can outperform classical machine learning models on a variety of tasks.

Neurons are organized into layers. Each neuron implements a linear model whose output is processed through an "activation" function:

$y = f(B_0 + B_1x_1 + B_2x_2 + \ldots B_nx_n)$

The hidden layers of a network often use a rectifier activation function:

$f(x) = \max\{0, x\}$

while the output layer often uses a sigmoid function and essentially consists of logistic regression models. In this lab, you're going to apply your knowledge about decision boundaries of linear classifiers to explore how neural networks perform classification. We'll use a multilayer perceptron model, which is a type of dense, feedforward neural network.

Instructions

1. Train Multilayer Perceptron (MPL) Models on Petal and Sepal Features

a. Load the Iris data set using:

```
from sklearn import datasets
iris = datasets.load_iris()
```

b. Train a MLP on the petal heights and widths:

```
scaled_X = StandardScaler().fit_transform(iris.data)
```

```
mlp_petals = MLPClassifier(hidden_layer_sizes=(4,),
                           max_iter=1000, solver="lbfgs")
mlp_petals.fit(scaled_X[:, 2:], iris.target)
```

c. Create a second MLP model for the sepal features by repeating for features 0 and 1 (the sepals height and widths).

2. Visualize Planes Learned by Individual Neurons

a. Extract the weight vectors for the hidden layers:

```
mlp_petals_models = np.vstack([mlp_petals.intercepts_[0],
                               mlp_petals.coefs_[0]]).T
```

The columns of this matrix correspond to B_0, B_1, and B_2. Each row corresponds to the model from a separate neuron.

b. Re-arrange the following equation to solve for x_2

$0 = B_0 + B_1 x_1 + B_2 x_2$

c. Plot features 0 and 1 from scaled_X along with the planes for the 4 neurons in the hidden layer. Use 100 points in the range [-2, 2] for x_1 to calculate the corresponding x_2.

d. Repeat for features 2 and 3 (the sepal height and width).

3. Visualize Decision Boundaries Resulting from Planes and ReLU Activation Function

a. Create a mesh grid in the range of [-2, 2] along each dimension.

b. Plot the grid as a scatter plot.

c. Use the Input and Neuron classes in the provided neurons.py file to calculate the value for the first hidden layer neuron at each grid point. (Pass the grid points into the predict() method as X).

```
input = Input()
p_layer = Neuron([input], mlp_petals_models[0, :])
pred = p_layer.predict(X)
```

d. Plot the model outputs as a heatmap or contourf plot.

e. Repeat for the remaining 3 neurons in the hidden layer.

f. Repeat this process for the hidden layer of the sepals model.

4. Train Logistic Regression models on Transformed and Original Features

a. Use the Input, Neuron, and HStack classes with the weights from the MLP model to recreate the hidden layer.

```
input = Input()
p_layer_1 = Neuron([input], mlp_petals_models[0, :])
p_layer_2 = Neuron([input], mlp_petals_models[1, :])
p_layer_3 = Neuron([input], mlp_petals_models[2, :])
p_layer_4 = Neuron([input], mlp_petals_models[3, :])
stacked = HStack([p_layer_1, p_layer_2, p_layer_3, p_layer_4])
```

b. Predict the transformed values to create a transformed feature matrix

```
transformed_petals_X = stacked.predict(scaled_X[:, 2:])
```

c. Repeat for the sepal widths heights MLP model.

d. Combine the two transformed feature matrix into a new feature matrix with 8 columns using np.hstack.

e. Train two LR models using SGDClassifier(loss="log") – one on the original 4 features and one on the new transformed feature matrix with 8 columns.

f. Evaluate the two models using accuracy and confusion matrices.

Reflection Questions

Put answers to the following reflection questions at the top of your notebook (after your title and name).

Problem 1:

1. What do the parameters to the MLPClassifier class mean?

2. Draw the network.

3. What activation functions are used for each node?

Problem 2:

1. What are the dimensions of mlp_petals.coefs_[0] and mlp_petals.intercepts_[0]? Where do those dimensions come from?

2. What are the dimensions of mlp_petals_models? What do the dimensions correspond to?

3. Comment on the abilities of the lines to separate setosa vs the rest, versicolor vs the rest, and virginica vs the rest with petal features.

Problem 3:

1. How does a ReLU function differ from a logistic function? What would the heatmaps/contour plots look like if we used logistic function as an activation layer instead?

2. A neural network consists of different layers and a final classification layer. Which activation function (ReLU or logistic) is more suitable to use for a classification layer? Which activation function is more suitable to use for an inner layer? – You may need to look up this information.

Problem 4:

1. How do the confusion matrices and accuracies of the two models compare? Did the transformed features produce a more accurate model?

Submission Instructions and Grading Criteria

Save the notebook as a PDF named lastname_lab05.pdf. Upload the PDF through Canvas.

I will be looking for the following:

- A title, your name, an introduction (including your own summary of the lab), and your answers to the reflection questions at the top of the notebook in Markdown.
- That your plots look reasonable. I will be checking for proper axis labels.
- That your accuracy values are reasonable.
- Obvious effort went into answering the reflection questions.

Followed submission instructions	5%
Presentation: axes are properly labeled, used correct axes for variables, points were colored as required, lines were coloring as required, used a legend, chose appropriate axes limits to make plot readable and do not cause misleading interpretations, etc.	5%
Trained Models	
Trained 2 MLP models	5%
Visualize Model Planes	
8 plots of planes and data	10%
Decision Boundaries	
Recreation of NN hidden layers	5%
8 decision boundary plots	10%
Model Comparison	
Created the transformed feature matrix correctly	5%
Trained the models correctly	5%
Calculated the metrics correctly	5%
Reflection Questions	
Problem 1	10%
Problem 2	10%
Problem 3	10%
Problem 4	10%
Exceeded Expectations	5%