MSOE UNIVERSITY

Build a feedforward Neural Net to Classify Flowers

1 Background

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. Within a layer, each neuron implements a linear model whose output is processed through an "activation" function:

$$f(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_m x_m).$$

The hidden layers of a network of often use a rectifier or rectified linear unit (ReLU) activation function:

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

The output layer often uses a sigmoid function that takes in the outputs of the final hidden layer and operates similarly to a set of logistic regressions. In this project, you will explore how neural networks perform classification. We'll use a multilayer perceptron model, which is a type of dense, feedforward neural network, to classify the flowers in the famous iris dataset.

2 Instructions

1. Train Multilayer Perceptron (MLP) models using the petal and sepal information as features and the flower type (referred to as "target" in the dataset) as the label:

   - Load the Iris data set:

     from sklearn import datasets
     iris = datasets.load_iris()

   - Center and scale each feature by applying the StandardScaler function from the sklearn.preprocessing library to create features with a mean of 0 and unit variance.
   - Train an MLP using the rescaled petal heights and widths as features using the MLPClassifier function from the sklearn.neural_network library. Use one hidden layer with 4 neurons, set the maximum number of iterations to 1000, and use the "lbfgs" solver.

- Repeat to create a second MLP model for the sepal information (use the 1st and 2nd columns of the scaled data – this is the sepal height and width).

2. Visualize separating planes learned by individual neurons in your petals model:

   - Extract the weight vectors for the hidden layers – there are a number of ways to do this; one possibility is to use the np.vstack command. For example if you call the fitted MLP using the petal data as features model mlp_petals:

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

     The columns of this matrix correspond to β_0, β_1, and β_2. Each row corresponds to the bias and weights used in a single neuron from the hidden layer.
   - The separating plane is described by all the feature values that would lead to the feature weight linear combination for that neuron, $\beta_0 + \beta_1 x_1 + \beta_2 x_2$, having a value of 0. To find these feature values set the feature weight linear combination equal to 0 and solve the resulting equation for x_2 in terms of $x1$,
   - Plot the petal features (the 3rd and 4th columns from the scaled data) along with the planes for the 4 hidden layers. Use 100 points in the interval [-2, 2] for x_1 to calculate the corresponding x_2.
   - Repeat for features 1 and 2 (the sepal height and width).

3. Visualize decision boundaries resulting from planes and the ReLU activation function for each neuron.

   - Create a mesh grid in the interval [-2, 2] along each dimension.
   - Calculate the value for the first hidden layer neuron at each grid point.
   - Plot the model outputs as a heatmap or contourf plot.
   - Repeat for the remaining 3 neurons in the hidden layer.
   - Repeat this process for the hidden layers of the sepals model.

4. Train logistic regression models on transformed and original features.

   - For each observation on the petal features in the scaled data compute the output of each neuron in the petals MLP model you created above.
   - Repeat for the sepal MLP model.
   - Combine the 8 resulting neuron outputs into a single transformed feature matrix.
   - Train two LR models using SGDClassifier(loss=”log”) – one on the original 4 features from the scaled data and one on the new transformed feature matrix with 8 columns.
   - Evaluate the two models using accuracy and confusion matrices.

3 Submission Instructions

Write up your analysis in a separate 2-page report. Your report should address:

1. Background on the data set – what is the iris data set? What did the creator of the data hope to achieve in collecting and analyzing it? What have others used it for since?

2. Include a summary of the architecture of your MLP models and how fit them. How wide and deep is your network? What activation functions are used for each node? What do the parameters to the MLPClassifier class mean? How do these MLP models compare with the logistic regression models trained in part 4?

3. The plane equation tells us which side of the plane a point is on (0 on the plane, $<$ 0 one side, and $>$ 0 other side). How does ReLU function modify the output and impact the interpretation? How does the ReLU activation layer make it easier or more difficult to use the decision boundaries?

4. Which combination of planes can you use to separate setosa vs the rest, versicolor vs the rest, and viriginica vs the rest with the petal features? Create a table of planes versus the three class comparisons. Indicate in the table which planes are useful as decision boundaries for each particular comparison. (Planes may be used across multiple comparisons.)

5. Interpret the confusion matrices and accuracy. Were the errors balanced across the classes for both models? Were some classes harder to correctly predict than others? If not, why do you think this is? Did the transformed features produce a more accurate model?

Submit your lab report as a PDF and your code as a notebook file in canvas.