ECE/CS/ME 539 – Fall 2024 — Homework 6

1.

(4 points)

(a) The starter code `logistic_starter.ipynb` defines the two datasets shown below – one is linearly separable and the other is not. Do you expect to *correctly classify* all samples from the first dataset when using a logistic regression model? Answer the same question for the second dataset.

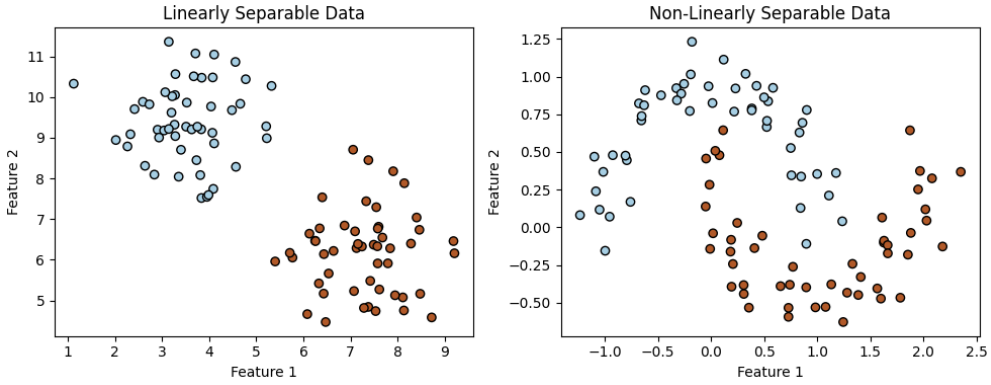

Figure 1: Linearly Separable Data and Non-Linearly Separable Data

(b) Train a logistic regression model on each dataset and report the number of training samples that were misclassified. Assume the standard probability threshold $p_{\text{thr}} = 0.5$. Refer to `sklearn.linear_model.LogisticRegression`.

(c) Using your results from part (d) of Activity 14 plot the decision boundaries on top of the scatter plots for the corresponding datasets.

(d) The provided starter code overlays the predicted probabilities over the entire feature space as a colormap, with darker blues indicating certain of negative class and darker reds indicating certainty of positive class. How would you expect the predicted probabilities to change on the first dataset, if the positive and negative clusters of samples overlapped with each other.

2.

(3 points) In this class, we will use stochastic gradient descent (SGD) to train a simple linear regression model to minimize a least squares objective.

$$f(x; \mathbf{w}, b) = \mathbf{x}^T \mathbf{w} + b$$

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} \frac{1}{2N} \sum_{i=1}^{N} \|y_i - f(\mathbf{x}_i; \mathbf{w}, b)\|_2^2$$

Data adaptive training finds the optimal parameters by gradient descent. In other words, it starts from a random guess for the model parameters, $\mathbf{w}^{(0)}, b^{(0)}$, and at each iteration it takes a small step in the direction of the negative gradient:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \alpha \nabla_{\mathbf{w}} L(\mathbf{w}^{(k)}, b^{(k)})$$

$$b^{(k+1)} = b^{(k)} - \alpha \nabla_{b} L(\mathbf{w}^{(k)}, b^{(k)})$$

where $L(\mathbf{w}^{(k)}, b^{(k)})$ is the loss function for the model at iteration k and α is the learning rate. Review the provided starter code carefully (`SGD_starter.ipynb`). This code is complete. It loads the iris dataset, splits it into training and testing data. It then defines several helper functions to compute the model predictions, compute the loss, update the parameters, and compute the evaluation metric R^2. Finally, the code uses these functions to train the model.

(a) After creating a copy of the code, modify it to compute both the training and testing R^2 at the end of each epoch. Plot the training R^2 vs epoch and test R^2 vs epoch. Discuss your observations.

(b) Try different learning rates (use `lr=0.03, 0.01, 0.003, 0.001`). Plot the test R^2 vs epoch for each run. Discuss your observations.

(c) Try different batch sizes (use `batch_size=1, 3, 10, 30, 100`). Plot R^2 vs epoch for each run. Discuss your observations.

3.

(3 points) Modify the code from problem 2 to train a logistic regression classifier. For this classification problem, use all 4 features from the iris dataset ($\mathbf{x} \in \mathbb{R}^4$) and learn a classifier that predicts if a sample belongs to class 2 or not. Beyond the dataset, you also need to modify the network function `net` and the loss function `loss`.

Recall that a logistic regression classifier predicts the probability of the positive class as

$$P(Y = 1|\mathbf{x}; \mathbf{w}, b) = \sigma(\mathbf{x}^T \mathbf{w} + b)$$

where $\sigma(\cdot)$ is the sigmoid function, and trains the model's parameters to minimize the binary cross-entropy loss between its prediction and the class labels

$$\mathbf{w}^*, b^* = \arg\min_{\mathbf{w},b} -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log P(Y = 1|\mathbf{x}; \mathbf{w}, b) + (1 - y_i) \log(1 - P(Y = 1|\mathbf{x}; \mathbf{w}, b)) \right).$$