Lab 3: Decision Boundaries

CS 3400 Machine Learning

Learning Outcomes

- Sketch or plot the decision boundaries for fitted models with 1 or 2 features
- Use a decision boundary plot to determine whether a given decision boundary separates classes
- Use a decision boundary plot to determine whether the features for the model have enough predictive power to separate the classes
- Describe why experimental setup is important for model evaluation
- Interpret a metric to assess the quality of a model's predictions and choose between two models (model selection)
- Relate a given classification metric to a decision boundary plot

Overview

In this lab, you will explore how decision boundaries are used by machine learning methods to classify data. We primarily draw on a popular machine learning data set called the Iris data set. The author of the data set measured the sepal and petal widths and lengths of 150 flowers from three species. The data set has four numerical independent variables and a single categorical dependent variable (species).

Instructions

1. Decision boundaries and Evaluation of Model Predictions with Metrics

a. Load the "setosa_data.csv" using the Pandas read_csv() function.

b. Display the first few rows of the DataFrame using the head() function.

c. Create a scatter plot of the data set points with the **sepal length** on the horizontal axis and the **sepal width** on the vertical axis. Color each point by its class (orange for setosa, blue for not setosa).

d. Plot in the specified color and label the decision boundaries defined by the following equations:

Decision Boundary 1 (black): $y = 2x - 8$

Decision Boundary 2 (red): y = 0.75x – 0.9

Decision Boundary 3 (purple): y = -0.2x + 12

Note: in this experimental setup, the x axis corresponds to sepal length and the y axis corresponds to the vertical axis.

e. Rewrite each decision boundary equation in a modified standard form:

Ax + By + c = 0

f. Import the linear_decision_boundary_classifier() function from the provided decision_boundaries.py file. Call the function for each decision boundary like so:

```
dec_bound_vec = np.array([0.75, -1.0, -0.9])
features = setosa_df[["sepal_length (cm)", "sepal_width
(cm)"]].values
true_labels = setosa_df["label"].values
pred_labels = linear_decision_boundary_classifier(dec_bound_vec,
features, true_labels, features)
print(pred_labels)
```

g. Use the scikit-learn sklearn.metrics.accuracy_score() function to calculate and print the accuracy of the predictions using each decision boundary using the true and predicted labels.

2. Predictive Power of Features

a. Load the "versicolor_virginica_data.csv" using the Pandas read_csv() function.

b. Display the first few rows of the DataFrame using the head() function.

c. Create a scatter plot of the data set points with the **sepal length** on the horizontal axis and the **sepal width** on the vertical axis. Color each point by its class (orange for versicolor, blue for virginica)

d. Create a second scatter plot of the data set points with the **petal length** on the horizontal axis and the **petal width** on the vertical axis. Color each point by its class (orange for versicolor, blue for virginica)

e. For each pair of features, and plot and label the decision boundaries defined by the following equations:

Decision Boundary 1 (black, sepal length and width): y = -0.7x + 7

Decision Boundary 2 (black, petal length and width): y = -0.7x + 5

f. Rewrite each decision boundary equation in a modified standard form:

Ax + By + c = 0

g. Import the linear_decision_boundary_classifier() function from the provided decision_boundaries.py file. Call the function with each pair of features and its associated decision boundary like so:

```
dec_bound_vec = np.array([0.7, -1.0, 7])
features = setosa_df[["sepal_length (cm)", "sepal_width
(cm)"]].values
true_labels = setosa_df["label"].values
pred_labels = linear_decision_boundary_classifier(dec_bound_vec,
features, true_labels, features)
print(pred_labels)
```

h. Use the scikit-learn sklearn.metrics.accuracy_score() function to calculate and print the accuracy of the predictions using each decision boundary using the true and predicted labels.

3. Experimental Setup with Train-Test Splitting

When preparing models for use and deployment in real-world situations, we want the models to make accurate predictions for data which we didn't have access to during training. For example, let's look at the problem of the spam filter in your email client. Your spam filter will be trained on all emails you have received so far. But your goal is for the spam filter to accurately classify emails you will receive in the future. The spam filter will attempt to learn which words indicate spam. If the spam filter only learns words that are specific to emails you received previously (e.g., "Eppendorf tubes"), then it will not work well on future emails. We would say this model is "overfit"; it basically "memorized" the training data instead of finding patterns that "generalize" to all spam. A more general spam model would use words like "sell", "transfer", and "bank account".

If we train and evaluate models on the same data, we will not be able to identify cases where the model is overfit and will overestimate its performance on new, unseen data. We can set up our experiments to reflect the real-world use case with more fidelity by splitting data into training and testing sets. For a problem like spam classification, we may split the emails so that all emails received before a certain date are used for training and all emails received after that date are used for testing.

The best way to think of our data is that it is made of observations that are described by features and responses. The power of supervised learning comes from us knowing the answers (responses) to the questions and being able to test our models using that information. Training/testing splits allow us to divide our known data into isolated sets, so that we can use statistical approaches to approximate the performance of our trained models. The "real" data set in this case represents data that our model might see in the real world, where we don't know the observation's response value – we are trying to predict it. If care is not taken to keep our training/testing datasets separated, we may over-estimate our model's performance.

We are going to look at an example that illustrates model overfitting and how evaluating the model on the training set can be deceptive. We partitioned the setosa/non-setosa data set into three smaller data sets: training, testing, and validation.

a. Create scatter plots of the validation, training, and testing data points (3 separate plots). Put the sepal length on the horizontal axis, and the sepal width on the vertical axis. Color each point by its class (orange for setosa, blue for not setosa). Plot the with the following decision boundary on each plot.

Decision Boundary (black): $y = 2x - 8$

b. Evaluate the predictions for the given decision boundary using the pairs of data sets in the table below. Use the `linear_decision_boundary_classifier`() function to predict labels for the following data sets and the accuracy_score() function to evaluate the accuracies of the predictions.

Train On	**Evaluate On**
Training Set	Training Set
Training Set	Testing Set
Training Set	Validation Set

Reflection Questions

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

Problem 1:

1. Just by looking at your plot, which of the two decision boundaries does a better job of separating the two classes of points?

2. Which of the two decision boundaries gave the more accurate predictions?

3. How does the accuracy metric seem to relate to the ability of the decision boundary to separate the classes?

Problem 2:

1. For which pair of features are the classes more easily separated?

2. Just by looking at your plots, for which pair of features does the decision boundary do a better job of separating the classes?

3. Which decision boundary gives the most accurate predictions?

4. How does the choice of features seem to impact the ability to make accurate predictions?

Problem 3:

1. Compare the accuracies you calculated from the training and testing data sets. Predictions for which data set were evaluated as more accurate? Which accuracy score do you think is a more realistic representation of the performance of the model?

2. Look at the scatter plot of the validation data points. Will the model make errors in predicting the labels? Why?

3. Look at the scatter plots of the training and testing data points. Which data set is more representative of the validation data set? Which data set will demonstrate errors similar to the validation set?

4. Compare the accuracies you calculated from the training and testing data sets to the validation data set. Which accuracy calculation is more representative of the accuracy for the validation data set?

5. Explain why training and evaluating a model on the same data set can be deceptive.

6. Explain how dividing data into training and testing sets with no repeated points resolves some of the problems associated with training and evaluating a model on the same data set.

7. Name 3 potential ramifications for publishing a model that was trained and characterized using the same data set.

Submission Instructions and Grading Criteria

Save the notebook as a PDF named lastname_lab03.pdf. Upload the PDF through Canvas. Print off and submit a hard copy of this pdf during the following week's lab meeting.

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%
Decision Boundary Plots	
6 total plots	10%
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.	10%
Accuracy: Used correct data sets for each plot, points and lines are plotted where they are supposed to be	10%
Classification and Evaluation	
Correctly rewrote decision boundaries equations in given form (6 times)	10%
Called linear_decision_boundary_classifier() with correct arguments (6 times)	10%
Called accuracy_score() with correct arguments (6 times)	10%
Reflection Questions	
Problem 1	10%
Problem 2	10%
Problem 3	10%
Exceeded Expectations	5%