
Net Id: rpk9907
Name: Rutuja Prakash Kajave

To answer some of these questions, you will have to write extra code (that is not covered by the test cases). The extra code should import your implementation and run experiments on the various datasets (e.g., choosing ivy-league.csv for a dataset and doing experiment.run with an 80/20 train/test split). You may use the visualization code provided in src/visualize.py.
You do not need to submit extra code you write to answer these questions.

---

1. In the coding section of this assignment, you trained Decision Tree and Perceptron models on several datasets. For each dataset in the data directory, do the following:
   - Generate 5 random train/test splits of the dataset using fraction = 0.8
   - For each split, train Decision Tree and Perceptron models on the training portion of the data
   - For each split, compute the accuracy of each model on the testing portion of the data
   - Report the average test accuracy of each model type across the 5 splits
   - For Decision Tree models, report the average number of nodes in the tree, and the average maximum depth (number of levels) of the tree
   - For Perceptron models, report if (and after how many iterations, on average) they converged

   You may find the code in tests/test_experiment.py helpful for getting started.

Solution: We have trained our datasets five times each on the perceptron as well as the decision tree respectively.

There are multiple datasets that have been used in this assignment. These datasets are mentioned below:

- Blobs.csv
- Candy-data.csv
- Circles.csv
- Crossing.csv
- Ivy-league.csv
- Majority-rule.csv
- Parallel_lines.csv
- Play_tennis.csv
- Transform_me.csv

- Xor.csv

Perceptron Implementation:

Name of the Data Set	Number of Iterations	Accuracy
Blobs.csv	2	1
Candy-data.csv	195	0.58
Circles.csv	198	0.65
Crossing.csv	199	0.51
Ivy-league.csv	198	0.81
Majority-rule.csv	18	1
Parallel_lines.csv	52	0.45
Play tennis.csv	200	0.27
Transform_me.csv	195	0.30
Xor.csv	200	0.55

Decision Tree:

Name of the Data Set	Number of nodes	Maximum Depth	Accuracy
Blobs.csv	5	2	0.93
Candy-data.csv	43	9	0.58
Circles.csv	7	2	0.85
Crossing.csv	5	2	0.79
Ivy-league.csv	21	4	0.91
Majority-rule.csv	41	6	0.7
Parallel_lines.csv	7	2	0.95
Play tennis.csv	9	4	0.32
Transform_me.csv	7	2	0.41
Xor.csv	7	2	1

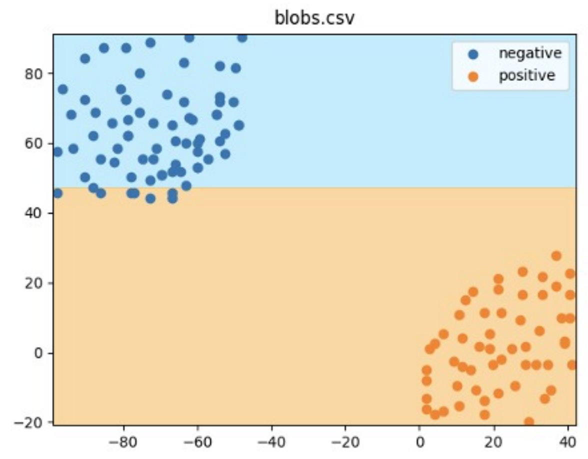
---

2. For these five datasets (blobs.csv, circles.csv, crossing.csv, parallel_lines.csv, and transform_me.csv):
   - Train a Perceptron model on the entire dataset (fraction=1.0) and plot the resulting decision regions using the plot_decision_regions function provided in visualize.py
   - Train a Decision Tree model on the entire dataset (fraction=1.0) and plot the resulting decision regions using the plot_decision_regions function provided in visualize.py

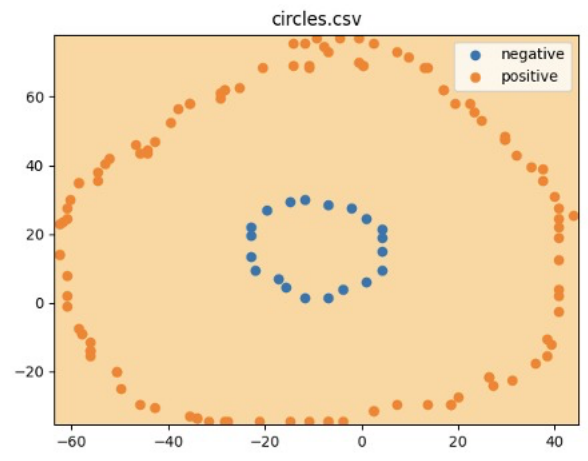
For this question, do not apply any transformation to the transform_me dataset. You should label your plots clearly using the titleargument of the plotting function.

Solution: After training our models on the above mentioned databases and plotting the resulting decision regions, we get the following graphs after implementing the plot_decision_regions function provided in the visualize.py file.

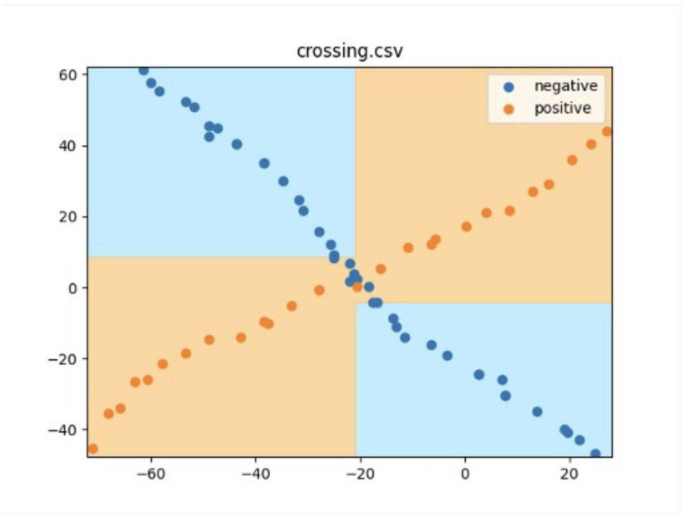
Graph for database Blobs.csv



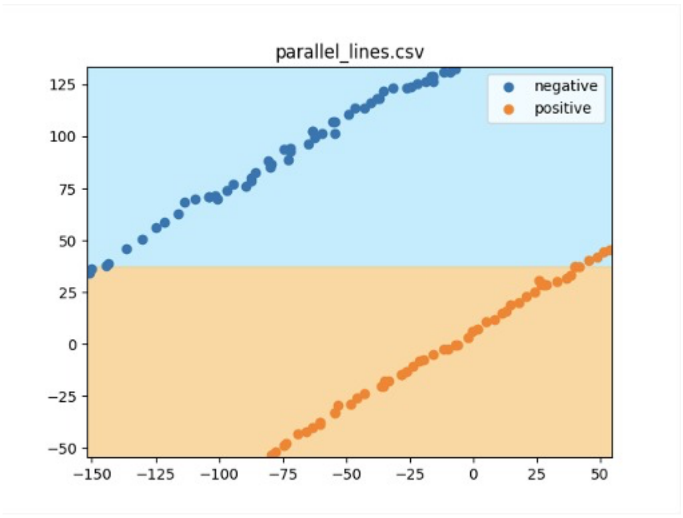
Graph for database Cricles.csv



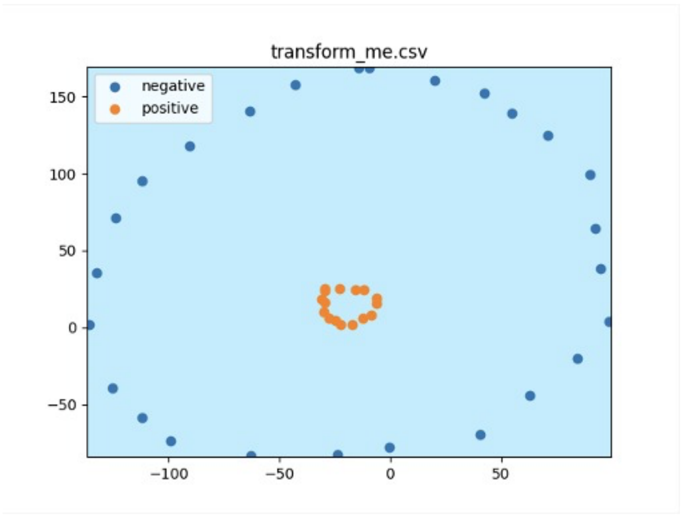
Graph for database Crossing.csv



Graph for database Parallel_lines.csv



Graph for database Transform.csv



3. For the Perceptron models you trained on the circles and (untransformed) transform_me datasets in Question 2, compute the precision, recall, and F1 scores over each dataset. While these datasets should look very similar when plotted, the performance of your Perceptron model as measured by these metrics should differ noticeably between the two. Explain why you think this is the case.

   Solution:

   The calculated values for precision, recall and F1 score for the circles database are as follows:

   Precision = 0.6
   Recall = 0.5
   F1 score = 0.545

   The calculated values for precision, recall and F1 score for the transform_me database are as follows:

   Precision = 0
   Recall = 0
   F1 score = not applicable.

   As it is observed, the dataset look very much similar to each other when plotted on a graph but the perceptron works much better with circles.csv data. This is because the perceptron was able to get true positives in circles. The number of true positives in transform was zero and that is the reason why perceptron performed poorly for the transform.csv dataset.

---

4. In src/perceptron.py, you passed a test case by implementing the function transform data. Describe the transformation you made to the input data to pass the test case. Explain how it allowed you to pass the test case (i.e. how did the change made to the data enable the system to do what it could not previously do?).

   Solution: When we look into the csv file, it is observed that the data present in the transform_me.csv file is not linearly separable. Therefore the plot it forms has two classes which are into one another. Due to this reason, perceptron does not work aptly on this database.
   To resolve this issue and to make the data linearly separable, we add a new dimension to the data 'y^2'. By adding this extra feature to the data, we add another dimension which makes it easier for us to rotate the plot which then makes it linearly separable by a plane.

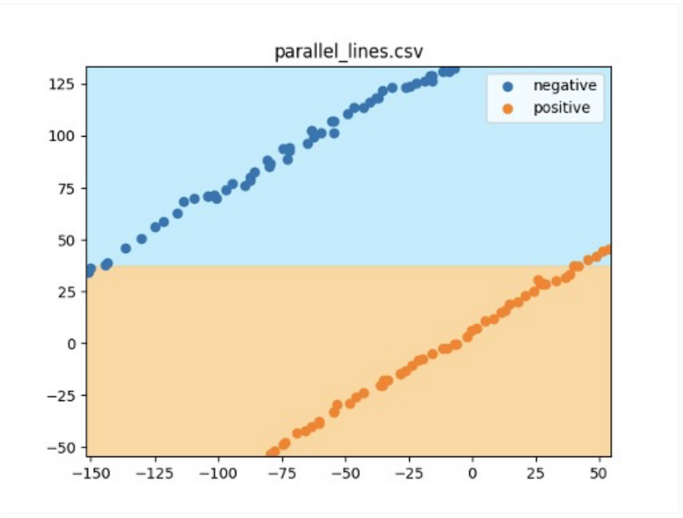
---

5. Choose two datasets D1 and D2 such that Decision Tree has better accuracy than Perceptron on D1, but Perceptron has better accuracy than Decision Tree on D2. Give the names of your chosen datasets and compute accuracy with fraction=1.0 in train_test_split(). For each dataset, visualize the data and explain why one model outperformed the other. You may reuse your code and/or scatter plots from Question 2. Your explanations should discuss the inductive biases of both models.

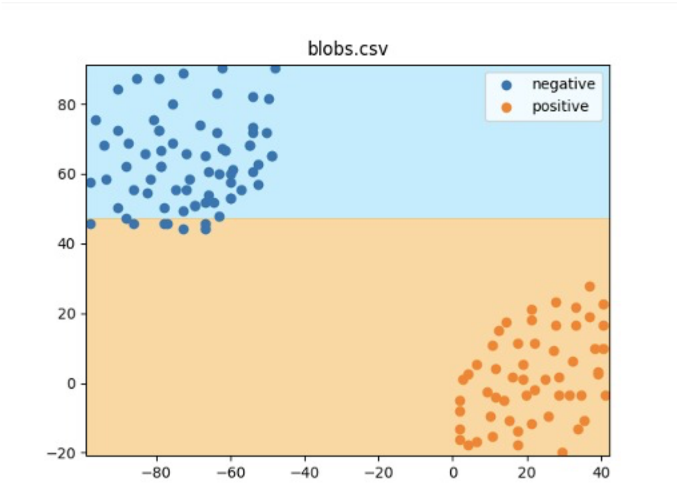
   Solution: The two databases which we have chosen for this are parallel_lines.csv and Blobs.csv

   Graph plotting after visualization is as follows:

   D1 = parallel_lines.csv



   D2 = Blobs.csv



   Accuracy of models of the respective functions is as follows:

   Decision tree on D1 = 0.89

   Perceptron on D2 = 1

6. For each of the datasets you considered in the previous question, describe a learning algorithm adjustment or data transformation that would allow the worst-performing model to perfectly classify the data. Explain your decisions.

   Solution:

---

7. Assume you have a deterministic function that takes a fixed, finite number of Boolean inputs and returns a Boolean output. Can a Decision Tree be built to represent any such function? Give a simple proof or explanation for your answer. If you choose to give a proof, don't worry about coming up with a very formal or mathematical proof. It is up to you what you want to present as a proof.

   Solution: Decision Tree can represent any such function since the function is a deterministic one. Assuming the no of features to be k, the decision tree can be built in such a way that there is a path that connects root to a node representing the k features for a specific example. This wat we have a decision tree that can give us the exact path of every data entry and deterministically gives the Boolean output. In worst case we can have a decision tree with 2n nodes depicting all possible values for a dataset of n features.

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

A

False True

B B

False True False True

Predictions - False True True False

8. There is a difference between a data structure (e.g. a decision tree) and the algorithm used to build that data structure (e.g. ID3). Different algorithms have different assumptions and biases built into them. At every decision point, the ID3 algorithm asks, "Which feature is most correlated with the output of the function we seek to model?" It then chooses to split on that feature. What is the inherent assumption built into using that question to choose splits? Describe a function with a Boolean output that violates this assumption but that could still be represented as a decision tree. Explain why it violates the assumption. Hint: keep track of the number of nodes in each decision tree you fit; which datasets require the largest trees? Why?

   Solution: The ID3 algorithm is supposed to function on the basis of reducing randomness at every split. It searches through all the attributes and then chooses the one that gives us the most amount of information gain or reduces the randomness of the model at the most. Therefore, the algorithm looks at only one node at a time and assumes that the attributes independently affect the outcome. This means that the algorithm takes greedy decisions which means that it focusses on maximising the result for a certain number of steps. This results in making the algorithm short sighted.

   The case where the algorithm fails is majority rule. Here the Boolean output is decided by majority of the features. The relationship between the features is very important in this case. The ID3 algorithm splits considering only one attribute at a time to reduce randomness but does not check them as a whole unit. Due to the following reason, the accuracy of the model in majority rule dataset above is also low.