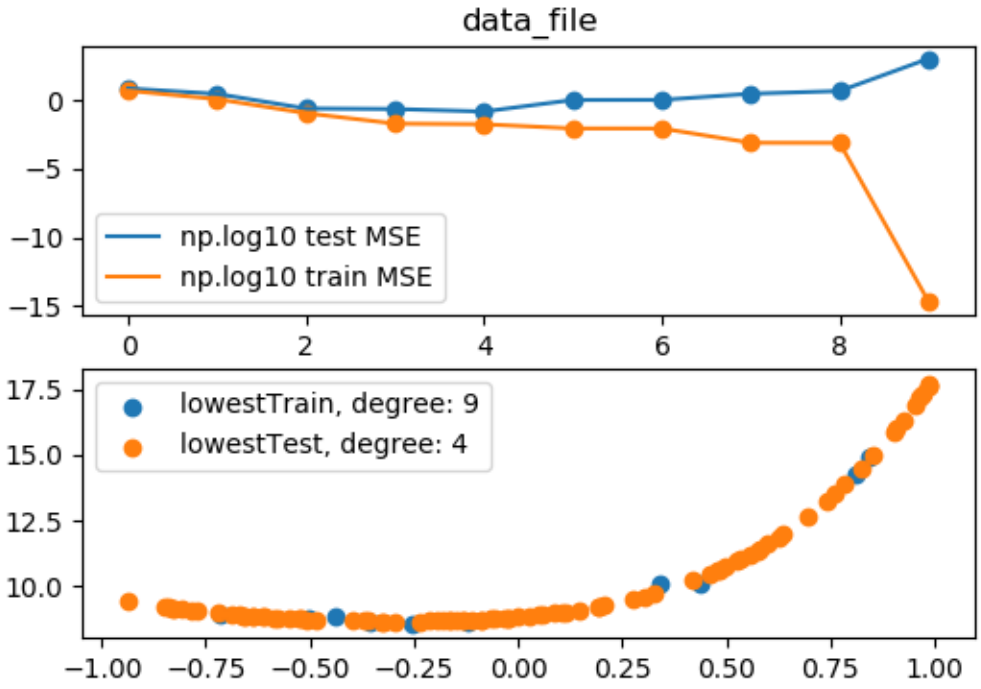
1. (0.5 point) Use the function you implemented `generate_regression_data` to generate 100 points of data (`amount_of_noise` = 0.1) from a polynomial function of degree 4. Randomly select 10 points as your training data. The rest will be your testing data. Run your implementation of `PolynomialRegression` 10 times, starting with degree 0 and increasing the degree of the polynomial each time, until you reach degree 9. Now create the following graphs.

- A. (0.25 points) A graph that shows mean squared error of your regression on the data as a function of degree. Make the horizontal dimension be degree of the polynomial used to fit the data. Make the vertical dimension the sum of squared residual error. Label your axes. Put two lines on this plot, one that shows TRAINING error as a function of polynomial degree, and one that shows TESTING error as a function of polynomial degree. Make sure to label which is testing and which is training error. *Hint: Sometimes you don't see the error for some polynomial degree because another one has huge error, making the scale too large to see the error for the other (also bad) polynomial. If you might have this situation, try taking the log of the error before plotting it.*
- B. (0.25 points) Create a 2nd plot that shows a scatterplot of your training data. Overlay this scatterplot with plots of the following polynomial curves.
    - The polynomial with lowest testing error (Be sure to label this curve with its degree and "lowest testing error".)
    - The polynomial with lowest training error (Be sure to label this curve with its degree and "lowest training error".)

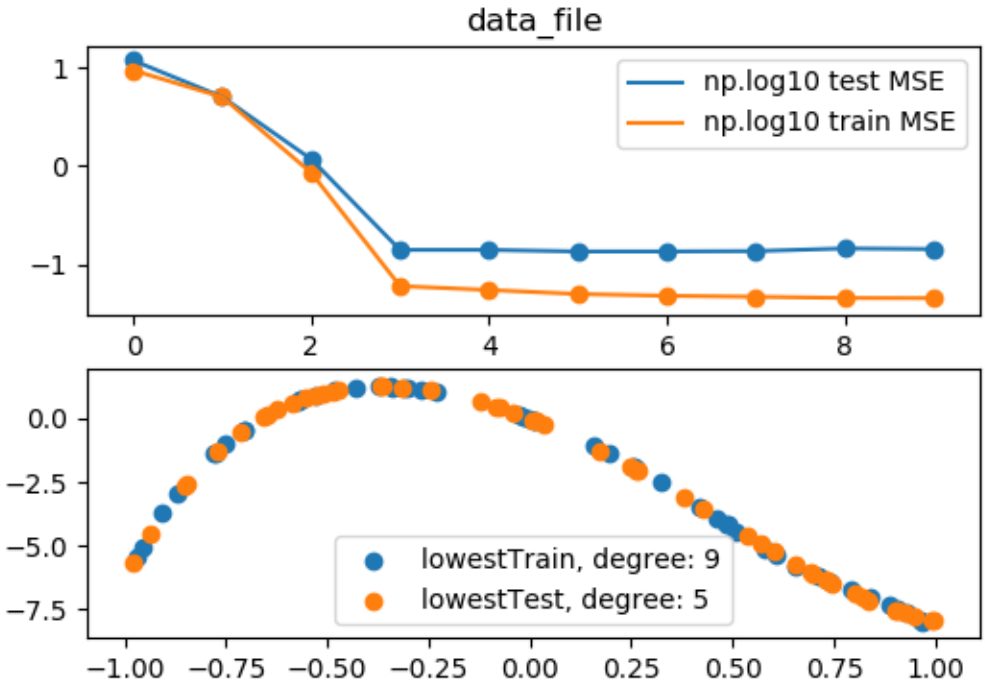


2. (0.5 points) Given your plots from question 1, describe the relationship between the degree of the polynomial used to fit the data and the training/testing error. At what degree of polynomial do you think you started overfitting the data?

If our degree is close to real data, then our test mse will decrease. If we are using the correct degree, then the test mse will reach its minimum. After that, we will start to overfit the training data and the test mse will increase.

From degree 4, the mse of training becomes smaller, but what more important is the mse of test from decreasing changing to increasing. That means, we are overfitting.

3. (0.5 points) Repeat everything you did in question 1 with one difference: randomly select 50 points as your training data.



4. (0.5 points) Compare your results from question 1 and question 3. How did increasing the number of training examples affect things?

Increasing the number of training example will reduce the affect of overfitting. Even overfitting, the difference of test mse and train mse will become smaller.

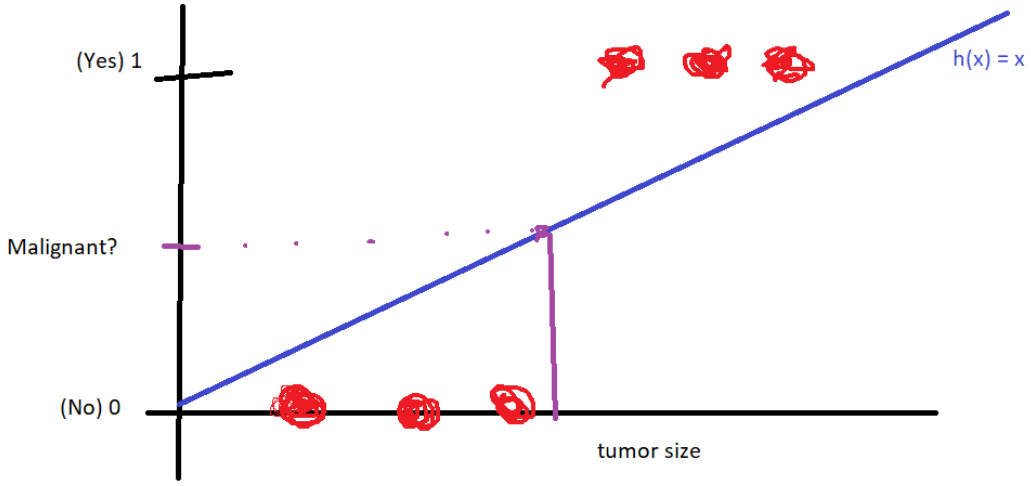
5. (0.5 points) Polynomial regression works by creating "new" dimensions from an input variable by raising to different integer powers and then running regression on this new multidimensional data. Raising the input variable to an integer power isn't the only way to create new dimensions that allow a non-linear regression fit to the data. Give a general rule for the kind of transformation that can be done to create a new dimension that will allow the math of linear regression to work.

We can feed more input data, and then we can let the math of linear regression work.

6. (0.5 points) Explain how to do linear classification via linear regression. Be clear. Use a graph to illustrate how it works. This graph doesn't need to be generated by your code, but it must be clearly labeled and made by you (not cut and pasted from somewhere). Explain one key weakness of classification via regression. Be clear. Perhaps illustrate this point, as well, with a graph.

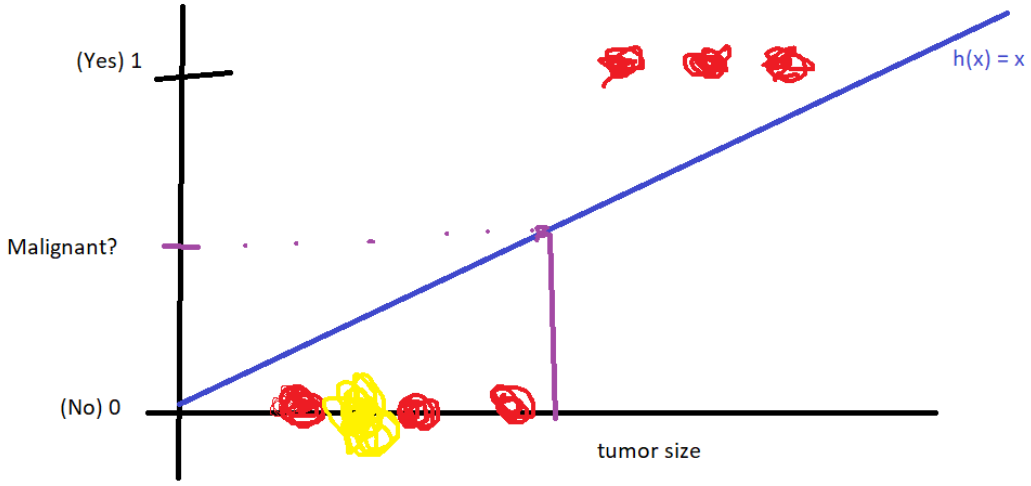
You can apply linear regression for classification by assigning a threshold, given below is an example.

Threshold classifier output h(x) at 0.5:



If h(x) >= 0.5, predict = 1
if h(x) < 0.5, predict = 0

A weakness is that sometimes it is difficult to define a proper threshold to classify them.

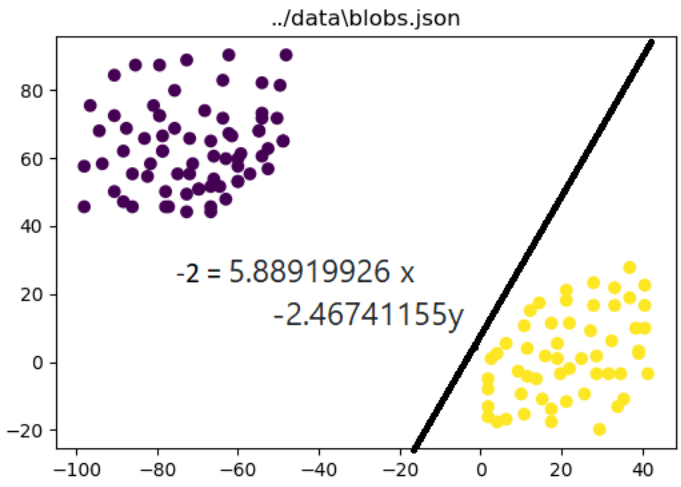


If h(x) >= 0.5, predict = 1
if h(x) < 0.5, predict = 0

For example, the yellow one should be yes. Because of the threshold, it is predicted as No. Therefore, it is not always the case that we can classify them by a threshold.

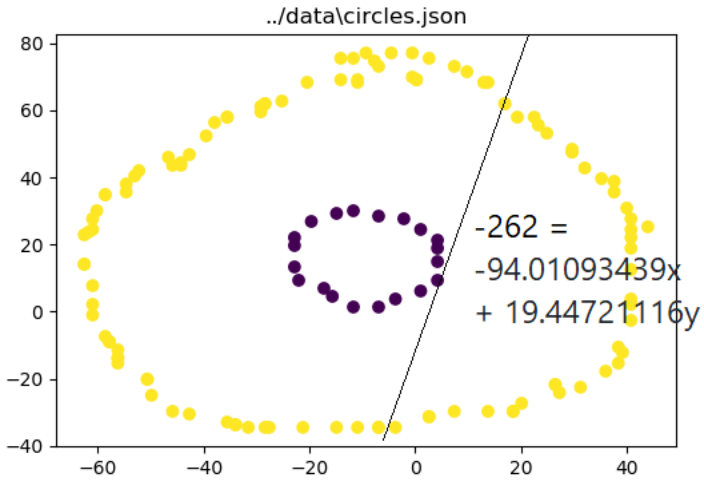
7.(0.5 point) In the data directory, we provide 5 datasets as json files: blobs, circles, crossing, parallell_lines and transform_me. For each of these 5 provided datasets (mentioned above), generate a scatter plot of the data with your own code. Color each point according to its true class. Plot the linear separator learned by your perceptron on top of the scatter plot, for each data set. If the perceptron never converged on a final line, pick the line it had generated by the time it reached max_iterations. See the source for `load_json_data` for an example of how to plot a scatterplot. Using http://https://matplotlib.org is encouraged.
Aside: you can upload the json files containing the datasets to http://ml-playground.com/ to see how different learning algorithms work on each dataset. This is not required for points, and you're definitely not allowed to submit scatter plots built by ml-playground, but it might be illuminating and useful for checking your work.

blobs.json:



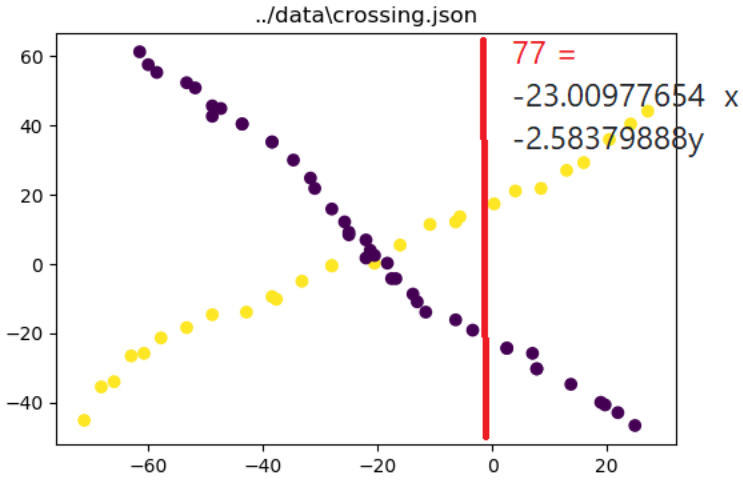
w: [2. 5.88919926 -2.46741155]

circles.json:



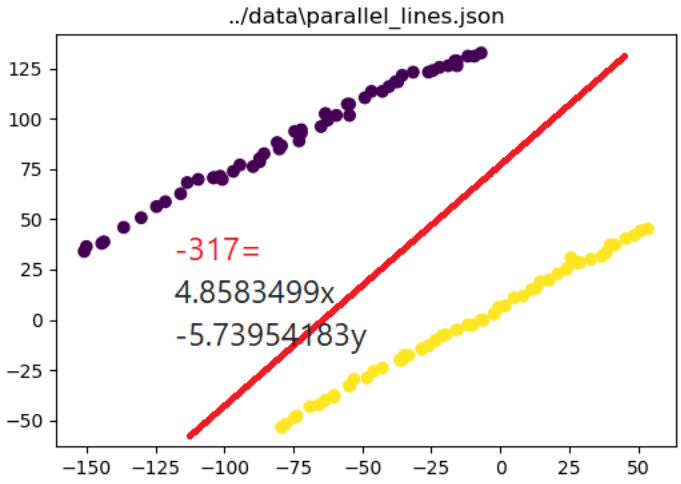
w: [262. -94.01093439 19.44721116]

crossing.json:



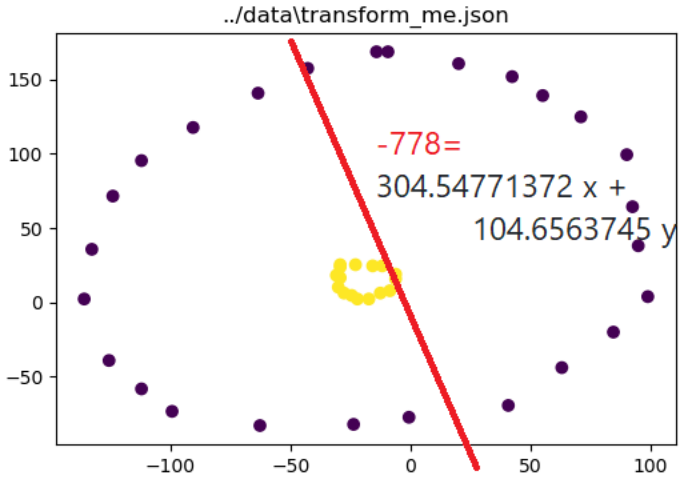
w: [-77. -23.00977654 -2.58379888]

parallel_lines.json:



w: [317. 4.8583499 -5.73954183]

transform_me.json:



w: [778. 304.54771372 104.6563745]

8. (1 point) For each of the 5 datasets (0.2 points per data set), say whether your perceptron successfuly separated the two classes on this dataset. Explain why or why not.

I didn't apply transforming the features. I use all original data and their features.

blobs.json: yes, because this one is linear separable.

circles.json: No, this one is not linear separable. There are two circles.

crossing.json: No, this one is not linear separable. There are two lines cross each other.

parallel_lines.json: yes, because this one is linear separable.

transform_me.json: No, this one is not linear separable. There are two circles.

9. (0.5 points) In `code/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?).

```
    tran_features[i, 0] = abs(features[i, 0])
      tran_features[i, 1] = abs(features[i, 1])
```

I try to change data in to absolute value.

Because one type is very close to 0, no matter x or y.

Another type has at least one very far away from 0.

Therefore, if I change value to absolute value, they will be linear separable.

One type will stay in the bottom left corner, and the other type will stay related top right.