Linear regression and PLA

The inputs for the tasks are in Numpy's npz format. They contain 4 arrays: `X`, `Y` for the training inputs and outputs, and `X_test` and `Y_test` for the test inputs and outputs, respectively. The samples are the rows of the `X` matrices.

Use only the training data for calculating the parameters (weights). Some input files contain reference solutions, you can use these to check the correctness of your answer (but only for that).

1. Using linear regression, find the weights and bias that generated the dataset in the following files. Measure the Mean Squared Error for the training and test data.

   Please implement it using basic matrix operations (ie. using the equation $(X^T X)^{-1} X^T y$), don't use the built in linear regression commands.

   a. `linear.npz`: no bias term
   b. `affine1.npz`: bias term is nonzero
   c. `affine2.npz`: there is noise in the dataset. Add some small (one-dimensional) perturbations to the weight vector (and bias) you found, and compare the new MSE.
   d. `rank.npz`: `X` may be a little bit weird
   e. `zero.npz`: which coefficients are neglibible? Why? Why is the error so large? What do you think, is there output noise in the dataset? Plot the outputs![1].

   Try exercis d. with `numpy.linalg.pinv` or `numpy.linalg.lstsq`. What algorithms do they use?

2. *Hoeffding equation.* You are testing a hypothesis on 43 samples, sampled from the data distribution. After measurement, you find your error to be 0.1. Your boss told you that she will tolerate at most an error of 0.2 in production (averaged for the long run). Can you bound the probability that the error will be larger (and she will be unhappy with you)?[2]

3. Implement the Perceptron Learning Algorithm.

   a. Find a linear separator that separates the two classes (-1 and 1) in the `pla.npz` dataset.

   b. Plot the training data and the separator in each iteration. (You can use the code in `visualise.py`.)[3]

      A simple way to do nonblocking plot in Matplotlib is:

      ```
      import matlotlib.pyplot as plt
      from visualise import visualise
      fig, ax = None, None
      while not done:
          w = calculate()
          fig, ax = visualise(w, X, Y, block=False, fig=fig, ax=ax)
          plt.pause(0.5)
      ```

   c. Repeat a) and b) on `pocket.npz` as well. How did it perform? Can you do better with a small modification?

[1] https://matplotlib.org/stable/gallery/mplot3d/scatter3d.html
[2] The answer is "Yes, you can bound it"; the correct solution to this exercise includes your bound and its derivation.
[3] To plot an animation, you can use the guides here: https://matplotlib.org/stable/api/animation_api.html