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).

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. Compare the test MSE to one with random weights.
   d. `rank.npz`: `X` may be a little bit weird
   e. `zero.npz`: which coefficients are neglibible? Why? Plot the outputs against the first and third components of the inputs[1].

2. 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`.)

      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