[redacted] In this programming exercise you will implement a Multilayer Perceptron (MLP) for optical-digit classification. You will train your MLP using the `optdigits_train.txt` data, tune the number of hidden units using the `optdigits_valid.txt` data, and test the prediction performance using the `optdigits_test.txt` data. For each file, the first 64 columns correspond to the features for different samples while the last one stores the labels. Features in each matrix should be normalized as $X_{norm} = \frac{(X-\mu_{trn})}{\sigma_{trn}}$. Notice that μ and σ, the mean and the standard deviation, are always calculated from the training set (even when normalizing the validation and test set).

(a) Implement a MLP with 1 hidden layer for classifying the 10 digits (read the algorithm in Figure 11.11 and section 11.7.3 in the textbook), use the tanh activation function $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ for the hidden layer, and the softmax activation function $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum\limits_k e^{x_k}}$ for the output layer. The error function is the cross-entropy loss:

$$E(r^t, y^t) = -\sum_i^C r_i^t \log y_i^t \tag{1}$$

where y^t is the predicted probabilities for different classes, C is the number of candidate labels. r^t is the one-hot vector for ground truth label, $r_i^t = 1$ if the current sample belongs to the i_{th} class and 0 otherwise.

Try MLPs with $H = 4, 8, 12, 16, 20$ and 24 hidden units. **Report the validation accuracy by the number of hidden units. How many hidden units should we use? Report the accuracy on the test set using this number of hidden units.** Hint: Given $y_i = \text{softmax}(\alpha_i) = \frac{e^{\alpha_i}}{\sum\limits_k e^{\alpha_k}}$, the derivative $\frac{\partial y_i}{\partial \alpha_j} = y_i(\delta_{ij} - y_j)$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

(b) Train your MLP with 2 hidden units and visualize the data by the values of the hidden units (similar to Figure 11.18). **Make a plot for the training, validation, and test sets separately**. Use different colors for different digits.

(c) Repeat previous question by training 3 hidden units and do the visualization using 3-D plot. **Compare the 2-D and 3-D plots and explain the results in the report.**

We have provided the skeleton code `MyMLP.py` and `visualization.py` for implementing the algorithm. `MyMLP.py` is written in a *scikit-learn* convention, where you have a *fit* function for model training and a *predict* function for generating predictions on given samples. To verify your implementation, call the main function `hw3.py`.