[Intro to AI] HW 4

Download MNIST datasets by running *hw4_template.ipynb* from the piazza. We'll use both *train_dataset* and *test_dataset* this time.

0. **Data type conversion** This step has been implemented in *hw4_template.ipynb* and just execute the cell. Step 0 helps us work with the numpy array X_{train} (training data), $\mathbf{y}_{train}$ (class labels of X_{train}), X_{test} (test data), $\mathbf{y}_{test}$ (class labels of X_{test}) from now on, which is converted from torch.Tensor data type.
1. **Data preprocessing** We'll create new datasets composed of digits 0 and 1 only.
   - In python, `b = a[a==0]` is a syntax that selects all the elements of `a` where the values are equal to zero, and that assigns them to `b`.
   - Given the datasets from step 0, choose all the images of 0 and 1. In the meantime, be sure to convert class label 0 to -1: class label of images 0 is -1 from now on.
   - Create $X_{train,01}$ (training data of 0 and 1 only), $\mathbf{y}_{train,01}$ (class labels of $X_{train,01}$), $X_{test,01}$ (test data of 0 and 1 only), $\mathbf{y}_{test,01}$ (class labels of $X_{test,01}$).
   - You might need `np.logical_or` or `np.concatenate` depending on how you implement step 1.
   - Vectorize all images by using `numpy.reshape`.
   - Append ***ones*** as a feature to the vetorized images. Use `np.hstack` or `np.concatenate`.

2. **Perceptron learning** Implement the perceptron learning algorithm summarized in slide 08. If you want to code up the learning rule as a function, see below for your information:

```python
def train_perceptron(X, y, w, learning_rate = 1e-3):
    # X: data matrix
    # y: class labels of images in X
    # w: model parameters

    # nb_changes: the number of parameter changes/updates.
    # equivalent to the number of misclassified images.
    nb_changes = 0

    # -------- Fill in here -----------

    # --------------------------------

    return w, nb_changes
```

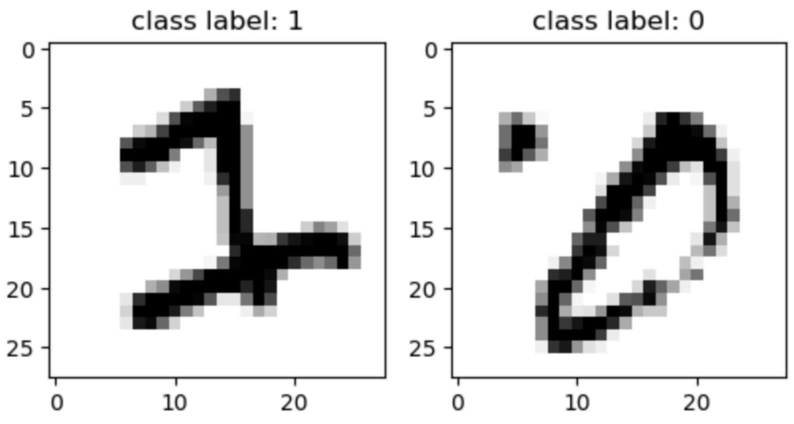
3. **Train and test the model**

- We want to stop training the model when `nb_changes=0` or the the number of epoch (the number of times the algorithm sees the *entire* data) exceeds a threshold.
- Initialize the model parameters `w` as all zeros or some random samples (e.g. gaussian, uniform, etc.)
- Refer to the code below, and try to code up so that the progress of learning is displayed.

```
nb_epochs_max = 100
w = np.random.randn(...)
w = np.zeros(...)

# -------- Fill in here -----------

# ---------------------------------

```

```
epoch 0 nb_changes 50 train_error 0.09% test_error 0.09%
epoch 1 nb_changes 27 train_error 0.14% test_error 0.19%
epoch 2 nb_changes 17 train_error 0.09% test_error 0.19%
epoch 3 nb_changes 15 train_error 0.05% test_error 0.05%
epoch 4 nb_changes 10 train_error 0.11% test_error 0.19%
epoch 5 nb_changes 11 train_error 0.04% test_error 0.19%
epoch 6 nb_changes 11 train_error 0.03% test_error 0.09%
epoch 7 nb_changes 9 train_error 0.03% test_error 0.05%
epoch 8 nb_changes 11 train_error 0.06% test_error 0.14%
epoch 9 nb_changes 8 train_error 0.00% test_error 0.09%
epoch 10 nb_changes 0 train_error 0.00% test_error 0.09%
```

4. **Visualize misclassified test images**


class label: 1
class label: 0