Exercise 1:

My final submission was an aggregation/majority vote out of my top 6 scoring individual classifier models.

My top 6 scoring models, in decreasing order of accuracy were:

1. kNN, k=6, using the L3 norm as the distance function, with neighbors weighted inversely proportional to their distance (on sklearn, this is `neighbors.KNeighborsClassifier(6, weights='distance', metric="minkowski", metric_params={'p':3})`), on the dataset preprocessed with a Gaussian filter, with a lambda of 1.

2. kNN, k=6, using the L3 norm as the distance function, with neighbors weighted inversely proportional to their distance, on the regular dataset, without any preprocessing.

3. kNN, k=6, using the default L2 norm as the distance function, with neighbors weighted inversely proportional to their distance (on sklearn, this is `neighbors.KNeighborsClassifier(6, weights='distance')`), on the regular dataset, without any preprocessing.

4. kNN, k=6, using the default L2 norm as the distance function, with neighbors weighted inversely proportional to their distance, on the dataset, preprocessed with a de-skewing algorithm (which I think I implemented incorrectly because the training, testing, and Kaggle scores were all lower than the unprocessed dataset, whereas studies have shown that de-skewing should improve the accuracy)

5. Bagging, with 50 base estimators, and the base estimators being KNN classifiers, with k=6 (on sklearn, this is
```
ensemble.BaggingClassifier(
    base_estimator=neighbors.KNeighborsClassifier(6, weights='distance'),
    n_estimators=50,)
```
), on the regular dataset, without any preprocessing

6. kNN, k=6, preprocessing the dataset by de-skewing it AND normalizing it to be [0,1] after de-skewing.

The code for all 6 of these models can be found in `Exercise1.py`

After training these models, and generating the test submissions on the test data (`MNIST_Xtestp.csv`, or `MNIST_Xtestp.csv` with the appropriate preprocessing applied), I aggregated their predictions by taking the majority vote for each sample, or the prediction from the most accurate base model (in this case, kNN, k=6, L3 distance, on Gaussian filtered dataset), if there was no clear majority (`CompareSubmissions.py`)

To run through this process for yourself, first run `python Exercise1.py`, followed by `python CompareSubmissions.py`. The final dataset will be in `submission_combined.csv`. (Warning, this will take a loooooooooooooooooooooooooooooooooooooooong time)

The script I used for all preprocessing (de-skewing, Gaussian filter) is in `deskew.py`.

Other notes:

Because of the large size of the dataset, and the expensive, time consuming process of training a model on the full dataset, I often trained and tested a model/idea/method I had on a subset of the full dataset (smallX.csv, smallX_{preprocessing method}.csv, smally.csv), and using that performance to decide whether or not to spend the time & energy to pursue this method and train the model on the full dataset.

I didn't use cross-validation to tune my hyper-parameters (k=6 for kNN, and lambda=1 for Gaussian filter preprocessing), because ~~I thought it would be too time consuming~~ I was too dumb to think of it at the time. Instead, I just tested all the values I wanted to try on the same dataset. Results, as well as results of other things I tried, can be found in gaussian_results.txt and results.txt.

Other things I tried (that didn't work out) include:

Preprocessing data to be "black and white", i.e. ceiling all nonzero values to 1.

AdaBoosting 100k decision trees, each with 17 leaves.

Preprocessing the data & downsampling it to a 16x16 pixel "image".

Implementing tangent distance and shape context matching as distance functions for kNN (Inspired by http://yann.lecun.com/exdb/mnist/) before reading the papers and realizing the math was way too dense to implement, and giving up.

Exercise 2.

$$\epsilon_{t+1}(h_t) = \sum_{i=1}^{n} p_i^{t+1} \left| h_t(x_i) - y_i \right|$$

When $h_t(x_i)$ agrees w/ y_i, $|h_t(x_i) - y_i| = 0$

$\therefore$ we only need to consider is s.t. $h_t(x_i) \neq y_i$

Let this set be W.

$$\therefore \epsilon_{t+1}(h_t) = \sum_{i \in W} p_i^{t+1}$$

$$= \sum_{i \in W} \frac{w_i^{t+1}}{\sum_{j=1}^{n} w_j^{t+1}} \qquad (1)$$

$$= \sum_{i \in W} \frac{w_i^t \beta_t^{1-|h_t(x_i)-y_i|}}{\sum_{j=1}^{n} w_j^t \beta_t^{1-|h_t(x_j)-y_j|}} \qquad (4)$$

Since $\forall\, i \in W$, $h_i(x_i) \neq y_i$, and $h_t, y_i \in \{0,1\}$, $|h_t(x_i) - y_i| = 1$, so $1 - |h_t(x_i) - y_i| = 0$

So $\beta_t^{1-|h_t(x_i)-y_i|} = \beta_t^0 = 1$

$$\therefore \epsilon_{t+1}(h_t) = \sum_{i \in W} \frac{w_i^t}{\sum_{j=1}^{n} w_j^t \beta_t^{1-|h_t(x_j)-y_j|}}$$

Similarly, we can seperate j into 2 groups, where $h_t(x_j) = y_j$, and $h_t(x_j) \neq y_j$.

Let these groups be R, W.

$$\epsilon_{t+1}(h_t) = \sum_{i \in W} \frac{w_i^t}{\sum_{j \in W} w_j^t \beta_t^{1-|h_t(x_j)-y_j|} + \sum_{j \in R} w_j^t \beta_t^{1-|h_t(x_j)-y_j|}}$$

so now, $\forall j \in W$, $|h_t(x_j) - y_j| = 1$, so $1 - |h_t(x_j) - y_j| = 0$

$\forall j \in R$ $|h_t(x_j) - y_j| = 0$, so $1 - |h_t(x_j) - y_j| = 1$

$$\epsilon_{t+1}(h_t) = \sum_{i \in W} \frac{w_i^t}{\sum_{j \in W} w_j^t \beta_t^0 + \sum_{j \in R} w_j^t \beta_t^1}$$

$$= \sum_{i \in W} \frac{w_i^t}{\sum_{j \in W} w_j^t + \sum_{j \in R} w_j^t \beta_t}$$

What is β_t?

$$\beta_t = \frac{\epsilon_t}{1 - \epsilon_t} \quad (3)$$

$$= \frac{\sum_{i=1}^{n} p_i^t |h_t(x_i) - y_i|}{1 - \sum_{i=1}^{n} p_i^t |h_t(x_i) - y_i|} \quad (2)$$

Once again, we can apply the fact that when $i \in R$, $|h_t(x_i) - y_i| = 0$, so we only need to consider when $i \in W$:

$$\beta_t = \frac{\sum_{i \in W} p_i^t}{1 - \sum_{i \in W} p_i^t}$$

$$= \frac{\sum_{i \in W} \frac{w_i^t}{\sum_{j=1}^{n} w_j^t}}{1 - \sum_{i \in W} \frac{w_i^t}{\sum_{j=1}^{n} w_j^t}} \quad (1)$$

$$= \frac{\frac{\sum_{i \in W} w_i^t}{\sum_{j=1}^{n} w_j^t}}{1 - \frac{\sum_{i \in W} w_i^t}{\sum w_j^t}}$$

(move summation to top)

$$= \frac{\sum_{i \in W} w_i^t}{\left(\sum_{j=1}^{n} w_j^t\right)\left(1 - \frac{\sum_{i \in W} w_i^t}{\sum_{j=1}^{n} w_j^t}\right)}$$

$$= \frac{\sum_{i \in W} w_i^t}{\sum_{j=1}^{n} w_j^t - \sum_{i \in W} w_i^t} = \frac{\sum_{i \in W} w_i^t}{\sum_{i \in R} w_i^t}$$

(since W, R are complementary, and make up $\{1 \ldots n\}$ combined)

So now, $\epsilon_{t+1}(h_t)$ becomes

$$\frac{\sum_{i \in W} w_i^t}{\sum_{i \in W} w_i^t + \left(\sum_{i \in R} w_i^t\right)\left(\frac{\sum_{i \in W} w_i^t}{\sum_{i \in R} w_i^t}\right)}$$

$$= \frac{\sum_{i \in W} w_i^t}{\sum_{i \in W} w_i^t + \sum_{i \in W} w_i^t} = \frac{\sum_{i \in W} w_i^t}{2 \sum_{i \in W} w_i^t}$$

$$\boxed{= \frac{1}{2}}$$