CS534 — Implementation Assignment 2 — Due 11:59PM Oct 27, 2018

## General instructions.

1. The following languages are acceptable: Java, C/C++, Python, Matlab

2. You can work in a team of up to 3 people. Each team will only need to submit one copy of the source code and report. You need to explicitly state each member's contribution in percentages (a rough estimate).

3. Your source code and report will be submitted through TEACH

4. You need to submit a readme file that contains the programming language version you use (e.g. python 2.7 ) and the command to run your code (e.g. python main.py).

5. Please make sure that you can be run code remotely on the server (i.e. babylon01 ) especially if you develop your code using c/c++ under visual studio.

6. Be sure to answer all the questions in your report. You will be graded based on your code as well as the report. In particular, **the clarity and quality of the report will be worth 10 pts**. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables.

7. In your report, the results should always be accompanied by discussions of the results. Do the results follow your expectation? Any surprises? What kind of explanation can you provide?

# Perceptron algorithm for Optical Character Recognition (total points: 80 pts + 10 report pts + 10 result pts)

**Task description.** In the Optical Character Recognition (OCR) we seek to predict a number (between 0 to 9) for a given image of handwritten digit. In this assignment we simplify the OCR into a binary classification task. Specifically, we consider predictions for only two numbers **3 and 5**. The goal in this assignment is to develop variations of the **perceptron algorithm** to classify handwritten digits of numbers 3 or 5.

**Data.** All the handwritten digits are originally taken from http://www.kaggle.com/c/digit-recognizer/data The original dataset contains the sample digits suitable for OCR. We extract samples with only labels 3 and 5. Following a little pre-processings we produce three datasets for this assignment as follows:

(a) **Train Set (pa2_train.csv):** Includes 4888 rows (samples). Each sample is in fact a list of 785 values. The first number is the digit's label which is 3 or 5. The other 784 floating values are the the flattened gray-scale values from a 2d digital handwritten image with shape $28 \times 28$.

(b) **Validation Set (pa2_valid.csv):** Includes 1629 rows. Each row obeys the same format given for the train set. This set will be used to select your best trained model.

(c) **Test Set (pa2_test.csv):** Includes 1629 rows. Each row contains only 784 numbers. The label column is omitted from each row.

**Important Guidelines.** For all parts of this assignment:

(a) Please assign labels +1 to number 3 and -1 to label 5. In your produced predictions, please use only +1 and -1 as labels not 3 and 5.

(b) Please do not shuffle the given data. While in practice shuffling should be used to improve training convergence, for this assignment we ask you not to shuffle the data to ensure determinstic results for assessment purpose.

(c) To simplify the notation in this assignment, your load function which loads train, validation and test dataset should add a bias feature to the dataset. The bias feature is a feature with value of 1.0 for all of the samples. Therefore the feature size for each samples will become 785.

---

**Part 1 (20 pts) : Online Perceptron.** In the online perceptron algorithm we train a linear classifier with parameter $\boldsymbol{w}$ to predict the label of a sample with equation:

$$\hat{y} = \text{sign}(\boldsymbol{w}^T \boldsymbol{x}) \tag{1}$$

Where $\hat{y} \in \{-1, 1\}$. Algorithm 1 describes the online perceptron.

**Algorithm 1** Online Perceptron

**procedure** ONLINEPERCEPTRON
    ~~$\boldsymbol{w_0} \leftarrow \mathbf{0}$~~
    ~~$t \leftarrow 0$~~
    ~~while iter < iters:~~
        ~~for all sample $\boldsymbol{x_t}$ in the train set: // no shuffling~~
            ~~$u_t \leftarrow sign(\boldsymbol{w_t}^T \boldsymbol{x_t})$~~
            ~~if $y_t u_t \leq 0$:~~
                ~~$\boldsymbol{w_{t+1}} \leftarrow \boldsymbol{w_t} + y_t \boldsymbol{x_t}$~~
                ~~$t \leftarrow t + 1$~~
    $\boldsymbol{w} \leftarrow \mathbf{0}$
    while iter < iters:
        for all sample $\boldsymbol{x_i}$ in the train set: // no shuffling
            $u_i \leftarrow \boldsymbol{w}^T \boldsymbol{x_i}$
            if $y_i u_i \leq 0$:
                $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i \boldsymbol{x_i}$
        iter $\leftarrow$ iter + 1

As we can see, the weight $\boldsymbol{w}$ which is used to predict a sample at time step $t+1$ is equivalent to:

$$\boldsymbol{w} = \sum_{\boldsymbol{x_i} \in S_t} y_i \boldsymbol{x_i} \tag{2}$$

where $S_t$ is a list containing all the previous samples that have been incorrectly classified by the model (some example may appear multiple times). Therefore the prediction at time step $t+1$ can also be given by:

$$\hat{y}_{t+1} = \text{sign}((\sum_{\boldsymbol{x_i} \in S_t} y_i \boldsymbol{x_i})^T \boldsymbol{x_{t+1}}) \tag{3}$$

In this part we are interested in the following experiments for the online perceptron algorithm:

(a) Implement the online perceptron model with algorithm described in Algorithm 1. Set the iters = 15. During the training, at the end of each iteration use the current **w** to make prediction on the validation samples. Record the accuracies for the train and validation at the end of each iteration. Plot the recorded train and validation accuracies versus the iteration number.

(b) Does the train accuracy reach to 100%? Why?

(c) Use the validation accuracy to decide the test number for *iters*. Apply the resulting model to make predictions for the samples in the test set. Generate the prediction file oplabel.csv. Please note that your file should only contain +1 (for 3) and -1 (for 5) and the number of rows should be the same as pa2_test.csv.

**Part 2 (20 pts) : Average Perceptron.** In this part we are interested to utilize average perceptron to deal with some issues regarding the online perceptron. Algorithm 2 describes the average perceptron.

As shown in the Algorithm 2, we compute a running average $\bar{\boldsymbol{w}}$ which is used to predict the label of any sample $\boldsymbol{x_i}$ as follows:

$$\hat{y}(x) = \text{sign}(\bar{\boldsymbol{w}}^T \boldsymbol{x_i}) \tag{4}$$

We are interested in below experiments for average perceptron:

(a) Please implement the average perceptron described in Algorithm 2.

(b) Plot the train and validation accuracies versus the iteration number for iters $= 1, ..., 15$.

(c) How average model has affected the validation accuracy comparing to the online perceptron?

**Algorithm 2** Average Perceptron

**procedure** AVERAGEPERCEPTRON
  $\boldsymbol{w} \leftarrow \mathbf{0}$
  $c \leftarrow 0$
  $\bar{\boldsymbol{w}} \leftarrow \mathbf{0}$ // keeps running average weight
  $s \leftarrow 0$ // keeps sum of cs
  while iter < iters:
    for all sample $\boldsymbol{x_t}$ in the train set: // no shuffling
      $u_t \leftarrow \text{sign}(\boldsymbol{w}^T \boldsymbol{x_t})$
      if $y_t u_t \leq 0$:
        if $s + c > 0$ :
          $\bar{\boldsymbol{w}} \leftarrow \frac{s\bar{\boldsymbol{w}} + c\boldsymbol{w}}{s+c}$
        $s \leftarrow s + c$
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_t \boldsymbol{x_t}$
        $c \leftarrow 0$
      else: $c \leftarrow c + 1$
  if c > 0:
    $\bar{\boldsymbol{w}} \leftarrow \frac{s\bar{\boldsymbol{w}} + c\boldsymbol{w}}{s+c}$

**Part 3 (40 pts). Polynomial Kernel Perceptron.** The online/average perceptron in Algorithm( 1 and 2) are linear models. In order to increase the model's complexity, one can project the feature vectors into a high (or even infinite) dimensions by applying a projection function $\Phi$. In this case the prediction at time (t+1) is given by:

$$\hat{y}_{t+1} = \text{sign}(\Phi(w_{t+1})^T \Phi(x_{t+1})) \tag{5}$$

Or equivalently by:

$$\hat{y}_{t+1} = \text{sign}((\sum_{x_i \in S_t} y_i \Phi(x_i))^T \Phi(x_{t+1})) \tag{6}$$

Where $S_t$ is a list containing all the previous sample vectors for which the model has a wrong predictions (there can be repetitions for a sample). Simplifying above equation yields:

$$\hat{y}_{t+1} = \text{sign}(\sum_{x_i \in S_t} y_i \Phi(x_{t+1})^T \Phi(x_i))) \tag{7}$$

As we see in the equation 7, the prediction at each time steps utilizes a similarity terms i.e. $\Phi(x_{t+1})^T \Phi(x_i)$ between the current projected sample $\Phi(x_{t+1})$ and all the previous samples which we have incorrect predictions.
Let's define a kernel function with parameter vectors x and y to be $k(x, y) = \Phi(x)^T \Phi(y)$. Therefore the equation is simplified to:

$$\hat{y}_{t+1} = \text{sign}(\sum_{x_i \in S_t} y_i k(x_{t+1}, x_i)) \tag{8}$$

There are different kernel functions. For example to compute the similarity between vectors x and y in the polynomial space (with polynomial degree p) we utilize below kernel:

$$k_p(x, y) = (1 + x^T y)^p \tag{9}$$

For $p = 1$ we have a linear kernel. In this part we are interested in polynomial kernel perceptron as described in Algorithm 3:
Please consider below experiments:

(a) Implement the polynomial kernel function $k_p$ in the Algorithm 3. This function takes two vectors $x_1$ and $x_2$ and an integer $p$ for the polynomial degree, and returns a real value.

**Algorithm 3** Kernel (polynomial) Perceptron

**procedure** KERNELPERCEPTRON
  $N$ : number of train samples, $F$ : number of features
  $X_{N\times F} \leftarrow$ train set
  $\boldsymbol{\alpha}_{N\times 1} \leftarrow \mathbf{0}$
  $\boldsymbol{y}_{N\times 1} \leftarrow$ train labels
  $k_p(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x}^T \boldsymbol{y})^p$
  $\mathbf{K}_{N\times N}(i, j) = k_p(x_i, x_j) \qquad \forall x_i, x_j \in X$

  while iter $<$ iters:
    for all sample $\boldsymbol{x_i} \in X$: // no shuffling
      $u = \text{sign}(\sum_j \mathbf{K}[j, i]\alpha[j]\boldsymbol{y}[j])$
      if $y_i u \leq 0$:
        $\boldsymbol{\alpha}[i] \leftarrow \boldsymbol{\alpha}[i] + 1$

(b) Define a Gram matrix $\mathbf{K}$ with size $N \times N$ where N is the number of training samples. Fill matrix $K(i, j) = k_p(x_i, x_j)$ for all of the pairs in the training set.

(c) Implement the rest of the kernel perceptron in Algorithm 3. For each $p$ in [1, 2, 3, 7, 15]:
1) Run the algorithm to compute $\alpha$.
2) At the end of each iteration use the current $\alpha$ to predict validation set.
3) Record the train and validation accuracy for each iteration and plot the train and validation accuracies versus the iteration number.
4) Record the best validation accuracy achieved for each $p$ over all iterations.

(d) Plot the recorded best validation accuracies versus degrees. Please explain how $p$ is affecting the train and validation performance.

(e) Use your best $\alpha$ (the best you found over all $d$ and iterations above) to predict the test data-set. Please name the predicted file as kplabel.csv.

**Submission.** Your submission should include the following:
1) Your source code with a short instruction on how to run the code in a readme.txt.
2) Your report only in pdf, which begins with a general introduction section, followed by one section for each part of the assignment.
3) Two prediction files oplabel.csv and kplabel.csv. These prediction file will be scored against the ground truth $y$ values and 10% of the grade will be based on this score.
4) Please note that all the files should be in one folder and compressed only by .zip.