**CSE 415 Artificial Intelligence**

**Final Project**

**Perceptron Image Recognition**

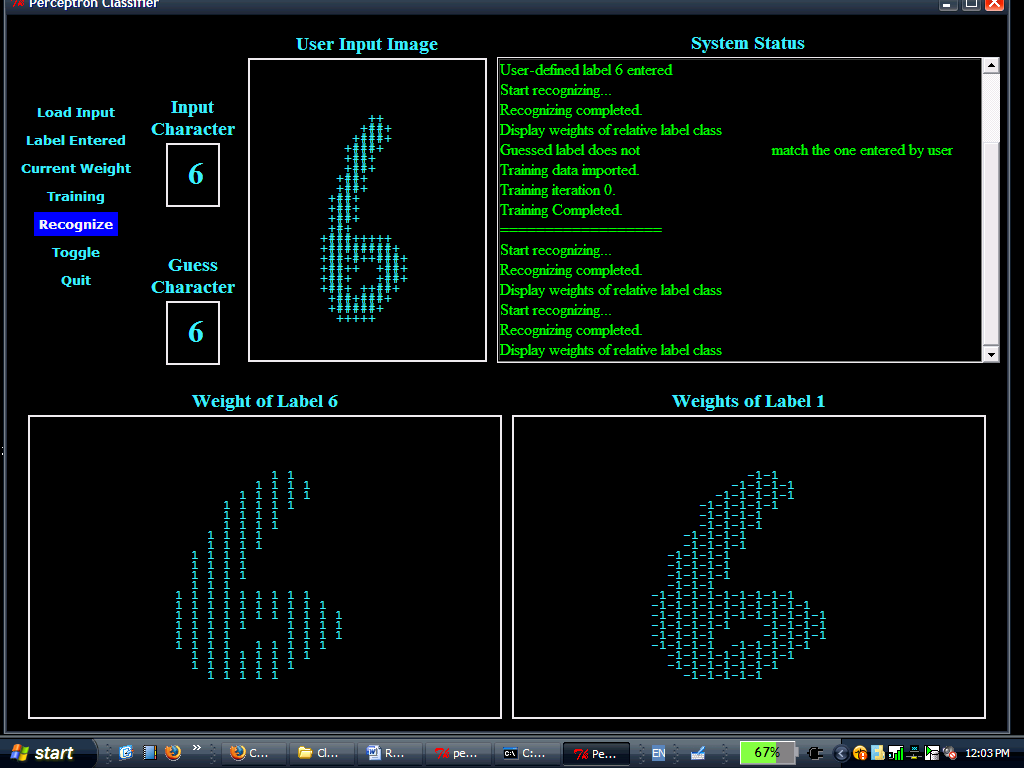
Jann-Chyi Chen

March 19, 08

**Technique and Description**

This project is to demonstrate the application of AI technique using Python and TKinter. Perceptron learning technique is used to recognize a set of pre-defined images. It means that the label associates with each image must be defined in the program prior to recognition process. In this project the AI is used for hand-written characters recognition. The user can “teach” the perceptron to recognize his or her unique hand-writing style. The change of prototype weights of each class is showed in during the program execution. After multiplying the input vector by the prototype weights of a defined label and sum them up, we can acquire a score for that class. After computing the score for every defined label class, the perceptron AI pick the class with the highest score and output its label. However, if the output label is not correct, the perceptron classifier must be trained to recognize the label for that given image. Each label on the record of the perceptron AI program has its own class of prototype weights. For the class of a label which was mis-selected by the perceptron AI, its prototype weights are decremented by the input vector; for the class of a label which was not selected but supposed to, its prototype weights are incremented by the input vector. Such change in prototype weights of both labels ensures that next time, the class of right label will be more likely to get selected by the AI perceptron.

**Screen Shot**



**Demo**

To use the program, one first needs to create a bmp image file with dimension of 112 x 112. Write down single digit or character and save it under the same folder as the program files. Then click interface.py file will start the program. After the program is successfully started, click “Load Input” button on the graphic user interface, and the program will convert the bmp image file to a file contains a datum object and save it under the same folder. Once the datum object is created, recognition process can begin. Click on “Recognize” button and the program will display its guess of label in “Guess Character” section. If the guess is not correct, you can tell the program the correct one by click “Label Entered” button and enter the label for that datum object. The value you just entered now shows up in the “Input character” section. Now click on “Training” button, and the program starts the training session. After training is finished, click on “Recognize” button to show changes of the prototype weights of the user input label and pre-guessed label. If there are too many zero to see the weight change. Click on “Toggle” to remove all the zero from the canvas. Click again to put them back.

**Code**

Following is the source code for perceptron classifier, one interesting thing about this code is the learning rate. One can make the AI learn faster or slower by adjusting the rate. However, or this project, learning rate, which is temp.POS\_WEIGHT and temp.NEG\_WEIGHT in the code, is defaulted as one. So the program learns very fast and forget very fast when new information comes in.

The second interesting thing in this code is the iteration times of a training set. More iteration times mean better learning results. Iteration only matters when the perceptron AI has stored a large set of labels or learning rate is too low to see significant change in prototype weight after a single training session.

Both iteration and learning rate can be taken cared by algorithm which adjusts their value if necessary. For example, if the perceptron AI “forgets” or ignores a pre-learned label, the algorithm then decreases perceptron learning rate. On the other hand, if the perceptron hardly learns the symbol after a set of iteration, the algorithm then can increase AI’s training iteration to improve training result.

def train( self, trainingData, trainingLabels, n, myList):

# iterate n times for a given training data

for iteration in range(n):

myList.insertItem("Training iteration " + str(iteration) + ".")

guess = []

# iterate through different training data

for i in range(len(trainingData)):

test = 0

vectors = Counter()

# iterate through each legal label class, and

for l in temp.LEGAL\_LABELS:

# compute the score for each class of weights

vectors[l] = self.weights[l] \* trainingData[i]

# determine whether a given legal label class has been trained

# if so, set variable "test" to true

if len(self.weights[l]) == 0:

test = 0

else:

test = 1

# if a class has never been train or the label having the highest

# grade is not what the user has inputted, which mean

# mis-recognize

if test == 0 or vectors.argMax() != trainingLabels[i]:

# assign the class of weights to a temp variable

posTrainData = trainingData[i]

negTrainData = trainingData[i]

# for each weight value in a label class of input labe

for j in trainingData[i]:

if trainingData[i][j] > 0:

# Factor each weight value in the input label class of weight

posTrainData[j] = (temp.POS\_WEIGHT)\*trainingData[i][j]

negTrainData[j] = (temp.NEG\_WEIGHT)\*trainingData[i][j]

# decrease the weight value of mis-guessed class by that of

# input datum object

self.weights[vectors.argMax()] -= negTrainData

# increase the weight value of correct class by that of

# input datum object

self.weights[trainingLabels[i]] += posTrainData

**Things Learned**

This project provides an insight of how perceptron classifier function, and what its potential is. I learned a lot of python and TKinter programming through this project. I also learned how to plan for constructing a program through the use of unified modeling language (UML).

**More Features**

If more time is allowed for this program, several things can be added to the perceptron AI. The first is the learning rate. The learning rate can be any value between 0 and 1. Of course, the higher the rate, the faster the perceptron AI can learn and easily to “over-learn”.

Another feature to add is the cropping feature. Given a handwritten sentence, the AI can pick up each letter, comma, dot, and space character, and successfully recognize them.

**Citations**

<http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm>

<http://www.java2s.com/Code/Python/CatalogPython.htm>

<http://www.pythonware.com/library/pil/handbook/image.htm>

<http://effbot.org/zone/tkinter-complex-canvas.htm>