Image Classification

Jaehyun Kim, Alex Vobornov

August 2019

1 Perceptron

1.1 Extracting

In the classifier.py, the raw data from training data return a set of pixel features indicating whether each pixel in the provided datum is '0' if it is empty ,or '1' if it is '+' or '#'. This works occurs at basicFeatureExtractorDigit and basicFeatureExtractorFace functions. For example, the feature will be {(0,0):0,(0,1):0,(0,2):1...(28,28):.0}.

1.2 Perceptron Classifier

PerceptronClassifier is called by classifier. Arguments of 'legalLabels' and 'option.iterations' are used. legalLabels is type of list that include 0 to 9 to compare labels. option.iterations is the option with -i that how many repeat the training. The default iteration is 3. That means hidden layers are 3.
For each layer, we scan one instance at a time and find the label with the highest score. To get a high score, we set y as real label and y' is guess label. The guess label can get from this formula y' = arg max score(feature, y") .
To get y', guessed label, we use classify function in perceptron class. Multiply weight[label] and current data as datum. It means multiply each features of value and then return the sum of the values to vector which is type of list. For example,

vectors[l] = weights[l] * datum
$= sum+ = weight[label] * datum[label]$
$= sum+ = (a, b) : z * (a', b') : z' = z * z'$
$= ((0, 0) : 0 * (0', 0') : 0) + ... + ((28, 28) : 0 * (28, 28) : 0)$
vector[l] = (0*0)+...+(0*0)
vector = {0:0, 1:144, 2:4, ... , 9:0}

14 seconds ago classificationMethod.py After calculating all of the vectors, we choose the maximum number in the vector list and using argMax function, which is in util.py, we return the highest key of the highest value.

And then, We compare y and y'. If y and y' are same then skip. If y and y' are not matching. It means that we guessed y' but we should have guessed y.

The weight of y should have scored f higher and weight of y' should have scored f lower.
In our code, for example, we used 100 of training data to see how y and y' work.

('Starting iteration ', 0, '...')
We have guessed 0 but should have guessed 5
We have guessed 5 but should have guessed 0
We have guessed 0 but should have guessed 4
We have guessed 5 but should have guessed 1
We have guessed 0 but should have guessed 9
We have guessed 0 but should have guessed 2
We have guessed 9 but should have guessed 1
We have guessed 1 but should have guessed 3
We have guessed 1 but should have guessed 4
We have guessed 1 but should have guessed 3
We have guessed 3 but should have guessed 5
We have guessed 3 but should have guessed 6
We have guessed 3 but should have guessed 1
We have guessed 3 but should have guessed 7
We have guessed 3 but should have guessed 2
We have guessed 3 but should have guessed 8
We have guessed 3 but should have guessed 6
We have guessed 1 but should have guessed 9
We have guessed 3 but should have guessed 4
We have guessed 3 but should have guessed 0
We have guessed 4 but should have guessed 9
We have guessed 9 but should have guessed 1
We have guessed 3 but should have guessed 2
We have guessed 2 but should have guessed 4
We have guessed 2 but should have guessed 3
We have guessed 3 but should have guessed 2
We have guessed 2 but should have guessed 7
We have guessed 2 but should have guessed 3
We have guessed 3 but should have guessed 8
We have guessed 3 but should have guessed 6
We have guessed 3 but should have guessed 9
We have guessed 3 but should have guessed 0
We have guessed 3 but should have guessed 5
We have guessed 3 but should have guessed 6
We have guessed 2 but should have guessed 0
We have guessed 2 but should have guessed 7
We have guessed 3 but should have guessed 1
We have guessed 0 but should have guessed 8
We have guessed 3 but should have guessed 7
We have guessed 0 but should have guessed 9

We have guessed 3 but should have guessed 8
We have guessed 0 but should have guessed 5
We have guessed 3 but should have guessed 9
We have guessed 0 but should have guessed 7
We have guessed 9 but should have guessed 4
We have guessed 9 but should have guessed 8
We have guessed 9 but should have guessed 4
We have guessed 8 but should have guessed 1
We have guessed 8 but should have guessed 4
We have guessed 4 but should have guessed 6
We have guessed 6 but should have guessed 4
We have guessed 4 but should have guessed 5
We have guessed 8 but should have guessed 1
We have guessed 2 but should have guessed 0
We have guessed 4 but should have guessed 0
We have guessed 0 but should have guessed 1
We have guessed 4 but should have guessed 7
We have guessed 4 but should have guessed 6
We have guessed 0 but should have guessed 3
We have guessed 1 but should have guessed 2
We have guessed 4 but should have guessed 7
We have guessed 0 but should have guessed 9
We have guessed 0 but should have guessed 2
We have guessed 2 but should have guessed 7
We have guessed 7 but should have guessed 9
We have guessed 9 but should have guessed 4
We have guessed 9 but should have guessed 8
We have guessed 9 but should have guessed 7
('Starting iteration ', 1, '...')
We have guessed 8 but should have guessed 5
We have guessed 4 but should have guessed 3
We have guessed 4 but should have guessed 5
We have guessed 8 but should have guessed 1
We have guessed 7 but should have guessed 1
We have guessed 7 but should have guessed 4
We have guessed 1 but should have guessed 8
We have guessed 7 but should have guessed 9
We have guessed 8 but should have guessed 0
We have guessed 1 but should have guessed 5
We have guessed 0 but should have guessed 6
We have guessed 8 but should have guessed 5
We have guessed 0 but should have guessed 3
We have guessed 9 but should have guessed 4
We have guessed 4 but should have guessed 9
We have guessed 9 but should have guessed 8
We have guessed 8 but should have guessed 1

We have guessed 9 but should have guessed 7
We have guessed 1 but should have guessed 2
We have guessed 0 but should have guessed 9
We have guessed 9 but should have guessed 6
We have guessed 9 but should have guessed 3
We have guessed 9 but should have guessed 7
('Starting iteration ', 2, '...')
We have guessed 3 but should have guessed 5
We have guessed 9 but should have guessed 2
We have guessed 9 but should have guessed 1
We have guessed 2 but should have guessed 0

In this case, we updated the weights as $w^y = w^y + f$ and $w^{y\prime} = w^{y\prime} - f$. In the code, we uses util.py methods which are __radd__ and __sub__ to increment/decrement counters.

$self.weights[y].\_\_radd\_\_(trainingData[i])$
$self.weights[yPrime].\_\_sub\_\_(trainingData[i])$

1.3 Training data

Percentage	Training Data
12%	2
24%	5
32%	14
46%	38
51%	40
62%	170
69%	330
75%	420
75%	1000

Table 1: Percentage of traing data with iteration 3

1.4 analysis

1.4.1 Perceptron

To analyze for our accuracy, we calculate percentage of validate data and test data.
For valid data with 100 trained, it shows us 69 correct out of 100 (69.0%). And for test data with 100 trained it shows us 62 correct out of 100 (62.0%).

1.4.2 Naive Bayes

2 Naive Bayes