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 data 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}

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 & Analysis

1.3.1 Digit

Training Data Used	Labels out of 5000	Validation Accuracy	Test Accuracy
10%	500	81%	76%
20%	1000	77%	79%
30%	1500	80%	78%
40%	2000	85%	77%
50%	2500	84%	82%
60%	3000	80%	82%
70%	3500	81%	84%
80%	4000	83%	86%
90%	4500	84%	87%
100%	5000	82%	86%

Table 1: Percentage of training data with iteration 3

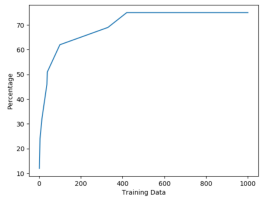

Figure 1: Digit training data

1.3.2 Face

Training Data Used	Labels out of 450	Validation Accuracy	Test Accuracy
10%	45	74%	62%
20%	90	91%	76%
30%	135	91%	72%
40%	180	94%	81%
50%	225	95%	79%
60%	270	99%	87%
70%	315	95%	80%
80%	360	93%	80%
90%	405	98%	82%
100%	450	100%	85%

Table 2: Percentage of face training data with iteration 3

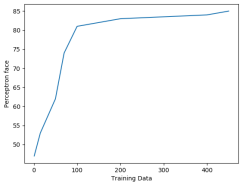

Figure 2: Face training data

2 Naive Bayes

2.1 Prior Probability

Our naive Bayes model has several parameters to estimate. One parameter is the prior probability over labels (digits, or face/not-face), $\Pr[Y]$.
To estimate $\Pr[Y]$, we extract data from training data labels.

$$\Pr[y] = \frac{c(y)}{n}$$

To implemet this formula, we use incrementAll, and normalize function from Counter() class in util.py. First initialize dictionary using Counter class and using incrementAll function, all the label keys to set 0. And then, count all of the label in the training data and save to dictionary, which is total_cnt_label in our code. For example, using 100 of labels, it looks {0: 13, 1: 14, 2: 6, 3: 11, 4: 11, 5: 5, 6: 11, 7: 10, 8: 8, 9: 11}. Next step is that using normalize function, we divide each value by the sum of all values. For example, it looks {0: 0.13, 1: 0.14, 2: 0.06, 3: 0.11, 4: 0.11, 5: 0.05, 6: 0.11, 7: 0.1, 8: 0.08, 9: 0.11}. Final step is that declare self.prior_probabilty variable and put it in this value so that we can use the other function to calculate base on Naive Bayes.

2.2 Conditional Probabilities

The next estimation is the conditional probabilities of our features given each label y: $\Pr[F_i|Y = y]$.

$$\Pr[F_i = f_i|Y = y] \quad = \quad \frac{c(f_i, y)}{\sum_{f_i' \in \{0,1\}} c(f_i, y)}$$

First, to calculate the formula in the denominator ($\sum_{f_i' \in \{0,1\}} c(f_i, y)$), we declare total_features_labels with util.Counter(). And to get a cost of feature and label set, we extract data from training data. When we use division, to avoid zero error occurs, we set all value as 1 that include keys of 0 and 1.
Second, to calculate the formula in the numerator ($c(f_i, y)$), we declare conditional_feature_label with util.Counter(). In this case, we only need to add pixel,which called feature that value is greater than 1.