AI Project

Xiaoming Hu
Yukang Shu
May 7, 2018

We devised three classifiers for detecting faces and handwritten numerical digits. They are Naive Bayes, Perceptron and MIRA.

Naive Bayes classifier

This classifier is based on Bayesian Model. At first, we calculate the PRIOR which is the probability of a joint distribution over label Y. In order to do it, we count that how many pictures are there for each label in the training data and divide it by the total number of training data.

$$P(y_i) = \frac{c(y_i)}{n}$$

Second, we calculate the LIKELIHOOD, which is the probability of the training data given each label. This is represented as *P(X|Y)*. To get this, we count the probability of a set of features given each label and multiply them together.

$$\begin{aligned} P(x_i|y_i) &= P(\phi_1(x_i), \phi_2(x_i), \phi_3(x_i), ...\phi_n(x_i)|y_i) \\ &= \prod_{k=1}^{n} P(\phi_k(x_i)|y_i) \end{aligned}$$

In order to calculate the probability of a set of features given each label, we count that how many pictures are there for each label and how many each specific feature in the training data. Then divide them by the number of pictures for each label, no matter what value it has for each features.

$$P(\phi_i|y_i) = \frac{c(\phi_i, y)}{\sum c(\phi_i, y)}$$

During the training phase, we use the training data to get the probability of each possible label and the possibility of each possible value of each feature given each label. During the testing phase, we identify what the value of each feature is in this picture at first. Then use the possibility we already have to calculate the likelihood of this picture given each label.

Then, we use Naive Bayesian model to calculate the POSTERIOR probability of each label given this picture, which is

$$P(y_i|x_i) = \frac{P(y_i|x_i) \cdot P(y_i)}{P(x_i)}$$

Finally, we choose the label with the highest posterior as our result.

We also smooth our parameter estimates by using Laplace smoothing. That is,

$$P(\phi_i|y_i) = \frac{c(\phi_i,y)+k}{\sum(c(\phi_i,y)+k)}$$

Perceptron classifier

This classifier uses a weight vector w_y for each label. This vector contains the weights of all the features we have for each label. Given one picture, we count the value of each feature it has for each label at first. Second, we multiply the value vector and the weight vector to get a score of each label. Then, we choose the label with the highest score as our predicted label for this picture.

During the training phase, we train the values of each weight vector. At first, we calculate all the scores for all the labels, and then we choose the label with the highest scores at our predicted label. Second, we compare this label with the correct label. If the predicted label and the observed label are the same, we did correct, so we do nothing. If the two labels are not the same, we predicted this picture wrong. It means that we somehow overestimate the score of the wrong label and underestimate the score of the correct label. Thus we have to increase the weight vector for the correct label and reduce the weight vector for the wrong label.

$$\begin{aligned} w_{yobserved} &= w_{yobserved} + f \\ w_{ypredicted} &= w_{ypredicted} + f \end{aligned}$$

MIRA classifier

MIRA classifier is similar to the perceptron classifier. The only difference is that, for Perceptron classifier, we increase or reduce the weight vector in the training period by controlling the value of features for the training pictures. But for MIRA classifier, we increase or reduce it by dealing with a different value τf. f refers to the value of features for one training picture. We calculate τ by doing

$$\tau = min(C, \frac{(w_{ypredicted} - w_{yobserved})f+1}{2||f||_2^2})$$

C is a positive constant that we have already defined as 0.001. f is the values of features for one training picture.
All the rest steps are the same for Perceptron classifier and MIRA classifier. It means that we update our weight vectors for the correct label and the wrong label by

$$w_{yobserved} = w_{yobserved} + \tau f$$
$$w_{ypredicted} = w_{ypredicted} + \tau f$$

Feature Extraction

The handwritten numerical digits feature was extracted by finding the boundary between white areas and grey or black areas at first. Then we found neighbors of each pixel and the continuous area with the white color. Finally we counted numbers of white continuous regions and return features of edge detection and numbers of continuous regions. Usually, number 1,3,4,5,7 contain only 1 continuous region, whereas number 6,0 contain 2 and number 8 has 3.

The face feature was extracted by first separating the image into 5×5 grids. Then numbers of black pixel the input 5×5 grid has were counted. Finally, each grid with at least 1 black pixel was assigned to 1 and each grid with no black pixel was assigned to 0. Return the assigned value of each grid.

Conclusion

Table below shows comparisons of their trained classifiers for handwritten numerical digits and face recognitions, respectively. For face recognition, generally Naive Bayes classifier used least time with highest accuracy as compared to Perceptron classifier and Mira classifier. For digits recognition, Naive Bayes classifier consumed least time as compared to perceptron classifier and Mira classifier. However, Perception classifier instead gave highest accuracy and MIRA classifier ranked as the second. Naive Bayes classifier gave the lowest accuracy. By comparing digits recognition to face recognition, all three classifiers took longer time to be trained and to make predictions of digits and the accuracy level of all three classifiers for digits is higher than that for faces. This might result from different characteristics of classifiers. Naive Bayes builds a probabilistic model to train by data. The extracted feature usually is independent from the probabilistic inference. In the training phase, Naive Bayes classifier was trained based on the average performance of prediction by combining all features across all images. Therefore it overlooked the image-by-image variability. Therefore, it cannot give higher accuracy as the data points increase. Rather, Perceptron and Mira take each image into account by evaluating and enhancing their performance on every single image. Therefore, the high accuracy can be maintained until the accuracy maxes out. Perceptron classifier and Mira classifier are often more accurate than Naive Bayes classifier.

Traingin Data%	Classifiers	5000 In Total	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
		Correct%	52%	81%	84%	84%	85%	86%	87%	85%	86%	85%
	NaiveBayes	Time(s)	4.22	3.88	4.81	5.12	5.46	5.88	6.23	6.73	7.07	7.39
		Correct%	80%	80%	82%	83%	69%	70%	69%	84%	87%	71%
	Perceptron	Time(s)	4.15	5.12	5.67	6.13	6.66	7.43	8.46	9.18	9.80	9.99
		Correct%	78%	86%	71%	85%	83%	82%	73%	87%	70%	78%
Face Recognition	Mira	Time(s)	4.56	5.34	6.08	6.70	7.40	8.55	9.21	10.00	10.89	10.97
		451 In Total										
		Correct%	72%	79%	79%	81%	82%	83%	82%	83%	83%	83%
	NaiveBayes	Time(s)	24	22.9	26.71	30.98	34.82	39.33	43.57	49	52.17	56.78
		Correct%	78%	79%	82%	87%	89%	86%	89%	86%	91%	92%
	Perceptron	Time(s)	56.79	107.35	164.72	214.02	263.78	294.52	359.19	376.46	454.59	479.17
		Correct%	78%	80%	79%	87%	82%	87%	90%	86%	88%	87%
Digit Recogntion	Mira	Time(s)	62.18	133.20	190.01	248.12	306.36	375.21	430.67	481.15	542.96	597.14