The Experiment Report of Machine Learning



Student ID：

201721045886, 201721045909, 201721045893

**SUBJECT:**SOFTWARE ENGINEERING

December 18, 2017

Grade:

Postgraduate

Supervisor:

Qingyao Wu

Author:

Siyuan Xiao, Li Zhang, Shengyan Wen

**SCHOOL:** SCHOOL OF SOFTWARE ENGINEERING

[[1]](#footnote-0)

Face Classification Based on AdaBoost Algorithm

Abstract—AdaBoost is a popular machine learning framework that combines multiple weak learners, and finally gain more accuracy than the original single weak method.

# INTRODUCTION

This report will talk about the whole experiment I have made on a face classifier based on AdaBoost. Its content is organized as follow:

1. Section II contains the experiment steps.
2. Section III contains the code for the two experiments.
3. Section IV makes conclusion for the experiment result.

# METHODS AND THEORY

Here we use 1,000 images which is placed in the requirement. Half of them contain faces and the rest do not.

Then the experiment will be performed by the following steps:

1. If it is not the first time executing this program, load the preprocessed data into memory and go to step 6; else, go to step 2.
2. Covert the images into grayscale mode.
3. Resize the images to 24 x 24.
4. Extract NPD feature vectors from them.
5. Split the feature data into training set and validation set, then save the preprocessed data.
6. Set parameters for the AdaBoost classifier, then start to train it.
7. In each iteration, train a base classifier by the given weights, and calculate weights for the next classifier.
8. Do prediction by combining all the trained classifiers.
9. Print the prediction result, compare with the single weak classifier.

# Experiment

Here I placed the critical code for the experiment:

1. ensemble.py:

|  |
| --- |
| 1. import pickle 2. import math 3. import numpy as np 4. import copy 5. class AdaBoostClassifier: 6. '''A simple AdaBoost Classifier.''' 7. \_\_base\_classifier\_\_ = None 8. \_\_classifiers\_\_ = None 9. \_\_max\_base\_\_ = 0 10. \_\_n\_base\_\_ = 0 11. \_\_alpha\_\_ = None 12. def \_\_init\_\_(self, weak\_classifier, n\_weakers\_limit): 13. self.\_\_base\_classifier\_\_ = weak\_classifier 14. self.\_\_max\_base\_\_ = n\_weakers\_limit 15. def fit(self,X,y): 16. self.\_\_alpha\_\_ = [] 17. self.\_\_classifiers\_\_ = [] 18. W = np.zeros([self.\_\_max\_base\_\_, X.shape[0]]) 19. W[0, :] = 1 / X.shape[0] 20. for m in range(0, self.\_\_max\_base\_\_): 21. #train m-th classifier 22. print ("train the " + str(m + 1) + " base classifier") 23. base = copy.deepcopy(self.\_\_base\_classifier\_\_) 24. base = base.fit(X, y, W[m]) 25. self.\_\_classifiers\_\_.append(base) 26. #predict through the m-th classifier 27. y\_predict = base.predict(X) 28. #calculate error 29. h = np.zeros(y\_predict.shape) 30. for i in range(0, y.shape[0]): 31. if y\_predict[i] != y[i]: 32. h[i] = 1 33. else: 34. h[i] = 0 35. h[i] = h[i] \* W[m, i] 36. epsilon = np.sum(h) 37. #calculate alpha value 38. self.\_\_alpha\_\_.append(0.5 \* math.log(1 / epsilon - 1)) 39. #reach max number of classifiers or good enough 40. if m >= self.\_\_max\_base\_\_ - 1 or epsilon < 0.1: 41. self.\_\_n\_base\_\_ = m + 1 42. break 43. #calculate weights for the next classifier 44. w = np.zeros([X.shape[0]]) 45. for i in range(0, X.shape[0]): 46. w[i] = W[m, i] \* math.exp(-self.\_\_alpha\_\_[m] \* y[i] \* y\_predict[i]) 47. z = np.sum(w) 48. for i in range(0, X.shape[0]): 49. W[m+1, i] = w[i] / z 50. def predict(self, X, threshold=0): 51. #sum prediction of all classifiers by their alpha 52. alpha = np.array(self.\_\_alpha\_\_) 53. h = np.zeros([self.\_\_n\_base\_\_, X.shape[0]]) 54. for m in range(0, self.\_\_n\_base\_\_): 55. h[m] = alpha[m] \* self.\_\_classifiers\_\_[m].predict(X) 56. return np.sum(h, axis=0) |

1. train.py:

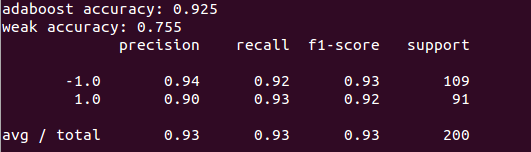
|  |
| --- |
| 1. import matplotlib.image as mpimg 2. import matplotlib.pyplot as plt 3. from sklearn.model\_selection import train\_test\_split 4. from sklearn.tree import DecisionTreeClassifier 5. import numpy as np 6. import os 7. import time 8. from scipy import misc 9. from feature import NPDFeature 10. import pickle 11. from ensemble import AdaBoostClassifier 12. from sklearn.metrics import classification\_report 13. def rgb2gray(rgb): 14. return np.dot(rgb[..., :3], [0.299, 0.587, 0.114]) 15. if \_\_name\_\_ == "\_\_main\_\_": 16. #load face data 17. datafile = 'data' 18. #data already preprocessed 19. if os.path.exists(datafile): 20. input = open(datafile, 'rb') 21. X\_train = pickle.load(input) 22. X\_vali = pickle.load(input) 23. y\_train = pickle.load(input) 24. y\_vali = pickle.load(input) 25. input.close() 26. #preprocess data 27. else: 28. facepath = 'datasets/original/face' 29. nonfacepath = 'datasets/original/nonface' 30. face = [] 31. nonface = [] 32. #for each image, convert it into grayscale presentation 33. #scale to 24x24 34. #and extract its NPD feature 35. facedir = os.listdir(facepath) 36. for i in range(0, len(facedir)): 37. if facedir[i].endswith('jpg'): 38. path = os.path.join(facepath, facedir[i]) 39. img = mpimg.imread(path) 40. img = rgb2gray(img) 41. img = misc.imresize(img, [24, 24]) 42. face.append(NPDFeature(img).extract()) 43. nonfacedir = os.listdir(nonfacepath) 44. for i in range(0, len(nonfacedir)): 45. if nonfacedir[i].endswith('jpg'): 46. path = os.path.join(nonfacepath, nonfacedir[i]) 47. img = mpimg.imread(path) 48. img = rgb2gray(img) 49. img = misc.imresize(img, [24, 24]) 50. nonface.append(NPDFeature(img).extract()) 51. X = np.array(face + nonface) 52. y = np.ones([1000]) 53. y[500:999] = -1 54. X\_train, X\_vali, y\_train, y\_vali = train\_test\_split(X, y, test\_size=0.2, random\_state=24) 55. output = open(datafile, 'wb') 56. pickle.dump(X\_train, output) 57. pickle.dump(X\_vali, output) 58. pickle.dump(y\_train, output) 59. pickle.dump(y\_vali, output) 60. output.close() 61. #create adaboost/weak classifier 62. dtc = DecisionTreeClassifier(random\_state=0, max\_depth=3, max\_features="sqrt") 63. classifier = AdaBoostClassifier(dtc, 15) 64. #train classifiers 65. classifier.fit(X\_train, y\_train) 66. dtc.fit(X\_train, y\_train) 67. #do prediction 68. result = classifier.predict(X\_vali) 69. weakresult = dtc.predict(X\_vali) 70. #calculate predicting accuracy for both 71. adacount = 0 72. weakcount = 0 73. for i in range(0, result.shape[0]): 74. if (np.abs(result[i]-1) < np.abs(result[i] + 1)): 75. result[i] = 1 76. else: 77. result[i] = -1 78. if result[i] == y\_vali[i]: 79. adacount = adacount + 1 80. if weakresult[i] == y\_vali[i]: 81. weakcount = weakcount + 1 82. print ("adaboost accuracy: " + str(adacount / result.shape[0])) 83. print ("weak accuracy: " + str(weakcount / result.shape[0])) 84. print(classification\_report(y\_vali, result)) |

# conclusion

We combine 15 weak classifiers in this experiment.

The weak classifier is a decision tree with max depth assigned to 3, uses log2(#features) features to split.

Here is the experiment result gained:



Then we can draw a conclusion according to the experiment:

1. AdaBoost classifier gain much improvement against the original weak classifier.

1. [↑](#footnote-ref-0)