FACE AND DIGIT CLASSIFICATION

Erica Cai, Boning Ding, and Shreya Jahagirdar

November 29, 2019

1 File Structure

Our project folder contains five files and two folders with images and labels. The five files are:

a. util.py

b. prepareFeatures.py

c. NaiveBayes.py

d. Perceptron.py

e. kNN.py

We wrote prepareFeatures.py, NaiveBayes.py, Perceptron.py and kNN.py. We used util.py to store all of the code from the Berkeley website that helps us parse each image and find black and white pixels in each image.

In NaiveBayes.py, Perceptron.py, and kNN.py, we implement the Naïve Bayes, Perceptron, and kNN algorithms respectively.

In prepareFeatures.py, we have several feature functions and we have a main program to run each machine learning algorithm on the dataset and to calculate the accuracy of the machine learning predictions.

For this project, there are six machine learning algorithms:

a. Naïve Bayes for predicting if an image is a face

b. Naïve Bayes for predicting which digit an image is

c. Perceptron for predicting if an image is a face

d. Perceptron for predicting which digit an image is

e. kNN for predicting if an image is a face

f. kNN for predicting which digit an image is

2 How to Test the Algorithms

We perform the same set of steps to test each algorithm.

a. 1. Store the train images, test images, train labels, and test labels in respective arrays.

b. 2. Apply the feature function on the images to get a train features and test features array.

c. 3. Train the machine learning algorithm on the train features and train labels.

d. 4. Predict the labels for the test features.

e. 5. Calculate the percentage of labels that the machine learning algorithm guessed correctly.

Only the Naïve Bayes and Perceptron algorithms do step 3; the kNN algorithm does not have a train function and is discussed more in section x.

3 How to Store Images

The images that we use in our train set and test set belong to the folders

a. digitdata, in the files

   digitdatatest

   digitdatatrain

b. facedata, in the files

   facedatatest

   facedatatrain

We extract each image and convert it into a "datum" object, which stores information about whether pixels in the image are black, white, or gray.

We store an array of "datum" objects and create features based on each "datum" object.

4 How to generate and store features

We convert each datum into a list of features.

Our feature function for digits splits the datum into xxx parts and counts the number of black pixels in each part. Looks like PICTURE

Our feature function for faces splits the datum into xxx parts and counts the number of black pixels in each part. Looks like PICTURE

To count the number of black pixels in each part, we traverse through each pixel of a part and update the number of black pixels in that part.

5 Naive Bayes Algorithm

5.1 Overview for Face Detection

We want to calculate

$L(x) = p(y = true|x)/p(y = false|x) = p(x|y = true)p(y = true)/p(x|y = false)p(y = false)$

. We return true if L(x) ¿=1 and false otherwise.

Our train function prepares all of the values necessary for calculating p(x—y = true),p(y = true),p(x—y = f alse),p(y = f alse) based on information in the train set.

Our predict function calculates L(x) for each item in test set assigns a label of 0 or 1 to that item; a 0 represents that the item is not a face and 1 represents that the item is a face.

5.2 Data Structures for Face Detection

We use dictionaries because searching is O(1) time while searching in an array takes O(trainingsize) time.

$dictionary containing (value, probability) pairs for feature 1, dictionary containing (value, probability) pairs for$

5.3 Method Definitions for Face Detection

TRAIN

Input: train features, train labels

Output: P(face=true), P(face=not true), array1, array2

Where Array 1 is an array of dictionaries; each dictionary represents a feature and contains (value, probability) pairs for the likelihood that the feature has that value GIVEN THAT THE IMAGE IS A FACE

Where Array 2 is an array of dictionaries; each dictionary represents a feature and contains (value, probability) pairs for the likelihood that the feature has that value GIVEN THAT THE IMAGE IS NOT A FACE

5.4 Train Algorithm for Face Detection

Countface=0
countNotFace=0
Array1=[, , . . .]
Array2=[, , . . .]
THIS PART COUNTS EVERYTHING
For each array of features in the training set
If the label shows that the image IS A FACE
countFace++
for feature i in the array of features
in array1, add it to the dictionary that corresponds to feature i and update the count
If the label shows the image IS NOT A FACE
countNotFace++
for feature i in the array of features
in array2, add it to the dictionary that corresponds to feature i and update the count
THIS PART TURNS COUNTS INTO PROBABILITIES
For every count in array1, divide the count by countFace
For every count in array2, divide the count by countNotFace
pFace=countFace/total
pNotFace=countNotFace/total
return pFace, pNotFace, array1, array2

5.5 Predict Algorithm for Face Detection

Input: test features, pFace, pNotFace, array1, array2
Output: predict labels
Need to calculate p(x—y = true),p(y = true), p(x—y = f alse),p(y = f alse)

Algorithm
For each item in test features
Use the following formula to compute p(x—y = true) from array1 and p(x—y = f alse) from array2
PICTURE
Then compute L(x)
Then add label based on into the predict array
Need to calculate p(x—y = true),p(y = true), p(x—y = f alse),p(y = f alse)

5.6 Overview for Digit Identification

Instead of calculating L(x) as for face, calculate p(x—y = true)*p(y = true) for each digit and predict the digit with the maximum result in this calculation.

The train and predict algorithm are very similar to that for Face.

5.7 Method Definitions for Digit Identification

Train
Input: train features, train labels
Output: P(digit 0=true), P(digit 1=true), P(digit 2=true), P(digit 3=true), P(digit 4=true), P(digit 5=true), P(digit 6=true), P(digit 7=true), P(digit 8=true), P(digit 9=true), array0, array1, array2, array3, array4, array5, array6, array7, array8, array9
Predict
Input: test features, test labels, P(digit 0=true), P(digit 1=true), P(digit 2=true), P(digit 3=true), P(digit 4=true), P(digit 5=true), P(digit 6=true), P(digit 7=true), P(digit 8=true), P(digit 9=true), array0, array1, array2, array3, array4, array5, array6, array7, array8, array9
Output: predict

6 Perceptron

6.1 Overview for Face Detection

FACE
We want to calculate f(xi , w) = w0 + w1PHI1(xi) + w2PHI2(xi) + w3PHI3(xi) + $\cdots$ + wlPHIl(xi),. If ¿=0, we predict that it is a face; otherwise, we predict that it is not a face.

Our train function prepares all of the weights necessary for calculating f(xi , w) = w0 + w1PHI1(xi) + w2PHI2(xi) + w3PHI3(xi) + $\cdots$ + wlPHIl(xi) based on information in the train set.

Our predict function calculates f(xi,w) for each item in test set assigns a label of 0 or 1 to that item; a 0 represents that the item is not a face and 1 represents that the item is a face.

6.2 Method Definitions for Face Detection

TRAIN
Input: train features, train labels
Output: array of weights
Algorithm
Initialize weights as [0.0001]
Iterate through the train features and labels until you don't have to adjust the weights anymore, or until a specific time

Return weights
PREDICT
Input: test features, array of weights
Output: predict labels

6.3 Algorithm for Face Detection

For each test feature array, calculate f(x)
Based on f(x), append 0 or 1 to the predict array, where 0 represents that the item is not a face and 1 represents that the item is a face.

6.4 Overview for Digit Identification

Instead of calculating one f(xi,w) function, calculate 10 of the them and choose the digit that corresponds to the maximum f value as the value that the algorithm predicts for an image

6.5 Method Definitions for Digit Identification

TRAIN
Input: train features, train labels
Output: array of weights for digit 0, array of weights for digit 1, array of weights for digit 2, array of weights for digit 3, array of weights for digit 4, array of weights for digit 5, array of weights for digit 6, array of weights for digit 7, array of weights for digit 8, array of weights for digit 9
PREDICT
Input: test features, array of weights for digit 0, array of weights for digit 1, array of weights for digit 2, array of weights for digit 3, array of weights for digit 4, array of weights for digit 5, array of weights for digit 6, array of weights for digit 7, array of weights for digit 8, array of weights for digit 9
Output: predict labels

7 kNN

Unlike the Naïve Bayes and Perceptron algorithms, kNN only has a predict function. The kNN algorithm takes in training features, training labels, test features, test labels, and a k value and returns an array of predicted labels for the test features.

For each array of features in the test set, the kNN algorithm looks through all of the arrays of features in the training set and chooses the k images that are most similar to the image in the test set. Then it uses the most frequent label of the k images to predict whether the current image in the test set is a face or not.