Massachusetts Institute of Technology

6.038 Artificial Intelligence in Practice

Problem Set 1 Solutions

1 Part A — Face Recognition

In part A, we asked you to think about how to tackle some related problems in face recognition:

- Recognizing whether a given image belongs to a specific person, say Joe, in any orientation, with glasses or not, etc.
- Recognizing all of the 20 people in the training set (in any orientation, etc).
- Recognizing that an image is of someone wearing sunglasses, or someone looking to the left, etc.

using some of the basic Machine Learning (ML) approaches you learnt about in 6.034:

- Nearest Neighbors (NearN),
- Identification Trees (IDT), and
- Neural Nets (NN).

Many of you assumed that since there were three problems and three approaches, this was a one-to-one match. But, that was not the intent. In fact, one can (in principle) use any of the methods to attack any of the problems. All the problems are instances of *classification problems*, that is, problems where one is trying to decide on a label (drawn from a small class) for a given family of inputs. All of the ML methods above are suitable for that type of problem. Had we been trying to do *regression problems*, that is, problems where one predicts a continuous output based on the input, then only NN would have been a reasonable choice. The other two methods need some extension to attack regression problems.

There are a couple of key issues that determine the suitability of a particular method for attacking a particular problem: (a) the type of mathematical function that is learnt by each of the methods and (b) the "feature space" used to describe the input. These two issues are tightly coupled. In particular, the choice of features determines whether a particular approach is likely to work. The simplest way of approaching the face recognition problem is to use all the pixel values in the image as "features". Note, however, that it is not practical to use every pixel in a 128×120 image, which is why in Part B, the images are subsampled to one-quarter size along each linear dimension. Using all these pixels has a number of implications.

- It is not reasonable to expect the intensity at individual pixels to convey much information about a whole face. When used on raw pixel values, an ID tree would reach whatever classification decision it reached based on comparing a relatively few individual pixel values against some threshold values. One should not expect such a very localized function to be a good choice for any of these problems (except see the results on identification trees below).

- One would expect that the relative importance of the different pixels to making decisions would vary a lot (a) among pixels and (b) across tasks. For example, the background pixels should not really contribute to any decision and, for example, the pixels around the eyes are the only ones that matter in detecting sunglasses. If one considers using a Nearest Neighbor method, one would really have to get a handle on these relative weightings. Figuring out where to get these weights is not at all obvious. In some sense, NearN is too global, as opposed to the ID trees that are too local. The majority of people picked Nearest Neighbors as the method of choice and in most cases neglected this issue.

- The NN would have a weight associated with each pixel for each (first level) hidden unit in the net. So, if we think of each unit specializing on some output, the net has the ability to find a weight that reflects each pixels importance. These weights provide a way of balancing local and global influences.

So, given that one is to use pixel values, a NN is probably the best choice for all these tasks.

One could, however, use different features and thereby make some of the other methods more competitive. For example, we discussed using Principal Component Analysis (PCA) as a way of recoding the input set into a smaller number of feature values. Basically, PCA finds a set of "basis pictures" such that one can (approximately) reconstruct the pictures in the input set as linear combinations (wighted sum) of these basis pictures. We can then use the coefficients of the weighted combinations as the features. In this case, it would make much more sense to use methods such as Nearest Neighbor and Identification trees, since these principal components have the right kind of global support but generally focusing on the interesting stuff – if a suitable input set is used.

Many of you assumed that one could compute "high level" features, such as location of eyes, angle of jaw, etc., and use these to construct feature vectors for the ML methods. Conceivably, these features would be more powerful, if one could compute them reliably. But, at the moment we can't. Instead, we can do a much better job of recognizing faces (using the whole images) than we can at finding reliable individual features to describe the faces reliably. That's because the individual features are much more *ambiguous* than the whole face is. You can convince yourself of this by looking at pictures through a soda straw and trying to categorize what you are looking at.

There were a number of significant *misconceptions* that surfaced in the Part A papers, among these were:

- Misconception: NNs can only have one output.

- Misconception: Because faces come from the same person they will produce feature vectors (in pixel value space) that are close to each other. But, think about the impact of illumination or change in pose on the individual pixel values.

- Misconception: We can pre-process images to find, e.g. intensity edges, and then "feed" them to a NN (or other ML method). But, note that there are (a) a variable number of edges that would be produced and (b) located in different places. So, to use such pre-processing one would have to find a fixed length feature vector that would encode the output so that individual features could be meaningfully compared across images.

2 Part B — Neural Nets (NN)

You can find the code for adapting the NN code to each of the problems, as well as the results of running the code on the data in `/mit/6.038/ps1/sample-result`.

As most of you saw, adapting the code was very easy, primarily the issue was adjusting the input files and picking the particular target values. Also, you saw that the NN (a) converged very quickly and (b) had very good performance on the test sets (90% or above in all the cases).

It is safe to say that you are not likely to find another NN application that will work this easily. Generally, NN are slow to converge, finicky about the learning rate and momentum parameters, and require much more data to get reliable behavior. This data set is artificial and has a few unusual characteristics which are highlighted by the performance of the identification tree (see below).

However, some people have a hard time believing that a neural net can actually "discover" interesting functions that are hard for humans to characterize. This example is a simple demonstration of that. On the other hand, other people are willing to believe that a neural net can do anything with almost no data. Unfortunately, this example, reinforces that misconception. As a sort of antidote to that, let's look at the performance of ID trees on this problem.

3 Part B — Identification Trees

You can find the results of running the C4.5 code on the data at `/mit/6.038/ps1/sample-result`.

As we discussed above, one would expect ID trees to be very bad at doing any face recognition task from raw pixel values. However, when you actually try them on this data set, you find that, in the name-that-person task, the performance is comparable to that of the NN. In fact, the rather small tree below gets 95% correct on the test set.

```
row_21_col_31 <= 0 :
|   row_0_col_0 <= 19 : mitchell (9.0)
|   row_0_col_0 > 19 :
|   |   row_0_col_0 <= 40 : night (14.0)
|   |   row_0_col_0 > 40 : sz24 (15.0)
row_21_col_31 > 0 :
|   row_23_col_29 <= 38 :
|   |   row_4_col_0 <= 11 :
|   |   |   row_0_col_17 <= 23 : kawamura (16.0)
|   |   |   row_0_col_17 > 23 : saavik (14.0)
|   |   row_4_col_0 > 11 :
|   |   |   row_7_col_0 <= 32 :
|   |   |   |   row_16_col_26 <= 30 :
|   |   |   |   |   row_3_col_0 <= 20 :
|   |   |   |   |   |   row_1_col_0 <= 10 : karyadi (16.0)
|   |   |   |   |   |   row_1_col_0 > 10 : ch4f (21.0)
|   |   |   |   |   row_3_col_0 > 20 :
|   |   |   |   |   |   row_8_col_29 <= 18 : glickman (11.0)
|   |   |   |   |   |   row_8_col_29 > 18 :
|   |   |   |   |   |   |   row_16_col_31 <= 18 : bpm (10.0)
|   |   |   |   |   |   |   row_16_col_31 > 18 :
|   |   |   |   |   |   |   |   row_1_col_7 <= 30 : at33 (13.0)
|   |   |   |   |   |   |   |   row_1_col_7 > 30 : an2i (12.0)
|   |   |   |   row_16_col_26 > 30 :
|   |   |   |   |   row_0_col_0 <= 26 : cheyer (10.0)
|   |   |   |   |   row_0_col_0 > 26 : kk49 (14.0)
|   |   |   row_7_col_0 > 32 :
|   |   |   |   row_0_col_12 > 24 : steffi (11.0)
|   |   |   |   row_0_col_12 <= 24 :
|   |   |   |   |   row_8_col_0 <= 31 : phoebe (17.0)
|   |   |   |   |   row_8_col_0 > 31 :
|   |   |   |   |   |   row_0_col_9 <= 12 : megak (14.0)
|   |   |   |   |   |   row_0_col_9 > 12 : choon (12.0)
```

```
|   row_23_col_29 > 38 :
|   |   row_7_col_4 <= 19 : boland (14.0)
|   |   row_7_col_4 > 19 :
|   |   |   row_0_col_0 <= 27 : tammo (17.0)
|   |   |   row_0_col_0 > 27 : danieln (17.0)
```

Note that it is using less than 10 pixel values on any image to classify the userid of the person quite accurately, even when the pose is changing. This should be surprising! In fact, if you look at the tree carefully you note that it is using mostly *background* (note the preponderance of row 0 and col 0 pixels) to do the classification. In fact, below is a map of the pixels used in the tree (x's):

```
x--------x--x----x--------------
x------x------------------------
--------------------------------
x-------------------------------
x-------------------------------
--------------------------------
--------------------------------
x---x---------------------------
x----------------------------x--
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------x----x
--------------------------------
--------------------------------
--------------------------------
--------------------------------
-------------------------------x
--------------------------------
-----------------------------x--
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
```

What seems to have happened is that the data set was acquired at a given time with a fixed background for each person. However, there are small changes in the background across the people. These pixel values are sufficiently informative to identify the users.

This effect highlights a **very** important aspect of any the theory and practice of ML. It is absolutely imperative that the training data be representative of the data one expects to get in the actual application.

Note that the ID trees do substantially worse on identifying the pose (across users). This reflects the fact that the spurious information in the background is not helpful. The following tree (obtained with the option -m 6 instead of the default -m 2 – i.e. at least 6 samples are necessary to justify a branch) gets about 75.5% of the test set "all_pose" cases correct. Note that, this time, it is actually looking at the (boundary of th) faces, see the map of pixels in the tree below.

```
row_19_col_9 > 61 : right (46.0/3.0)
row_19_col_9 <= 61 :
|   row_9_col_10 > 95 : right (9.0)
|   row_9_col_10 <= 95 :
|   |   row_16_col_9 > 92 : right (6.0/1.0)
|   |   row_16_col_9 <= 92 :
|   |   |   row_19_col_20 <= 53 :
|   |   |   |   row_9_col_15 <= 5 : right (6.0/2.0)
|   |   |   |   row_9_col_15 > 5 :
|   |   |   |   |   row_12_col_9 > 62 : left (6.0/2.0)
|   |   |   |   |   row_12_col_9 <= 62 :
|   |   |   |   |   |   row_8_col_21 <= 14 : straight (6.0/3.0)
|   |   |   |   |   |   row_8_col_21 > 14 :
|   |   |   |   |   |   |   row_10_col_15 <= 113 :
|   |   |   |   |   |   |   |   row_22_col_16 <= 90 : straight (12.0)
|   |   |   |   |   |   |   |   row_22_col_16 > 90 :
|   |   |   |   |   |   |   |   |   row_11_col_19 <= 7 : straight (8.0/2.0)
|   |   |   |   |   |   |   |   |   row_11_col_19 > 7 : up (52.0/1.0)
|   |   |   |   |   |   |   row_10_col_15 > 113 :
|   |   |   |   |   |   |   |   row_6_col_19 <= 35 : straight (40.0/1.0)
|   |   |   |   |   |   |   |   row_6_col_19 > 35 : up (7.0/3.0)
|   |   |   row_19_col_20 > 53 :
|   |   |   |   row_22_col_19 > 114 : up (10.0/3.0)
|   |   |   |   row_22_col_19 <= 114 :
|   |   |   |   |   row_17_col_26 <= 1 : straight (7.0/2.0)
|   |   |   |   |   row_17_col_26 > 1 : left (62.0/3.0)

--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
-------------------x------------
--------------------------------
---------------------x----------
----------x----x----------------
---------------x----------------
-------------------x------------
---------x----------------------
--------------------------------
--------------------------------
--------------------------------
---------x----------------------
--------------------------x-----
--------------------------------
---------x----------x-----------
--------------------------------
--------------------------------
----------------x--x------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
--------------------------------
```