Chris Medlock
Scott Deans
Charles Sedgwick

Introduction

We were presented with the *problem of how to determine if a picture contains an Alberta License Plate and, if it does, further determine if the last digit is the number four, five or something else.* This was to be done by training machine learning algorithms to automate the classification as accurately as we can get with the underlying motivation of achieving a better understanding of machine learning algorithms, what they can be used for, what their limits are and what the advantages they hold. We also had to do some research into different programs for pre-processing as well as pre-processing methods themselves to help improve this problem.

Proposed Method

We do not think we can achieve the accuracy seen in commercial products that do this classification but we believe we can achieve acceptable results by selecting areas which hold features unique to Alberta license plates despite the inherent difficulty of this task. The vast variability in pictures for things like light levels, angle, color of car, dirt on plate, etc. are what make classifying very difficult. It is also a good starting position for improving and developing a usable license plate reading system as a possible end-of-degree project.

Originally we had a java program that would read in the pixels of the picture file and would create the test data and with 80% of the examples and then we would allow the user to input one file and determine its classification while this was a nice program we could not test the overall accuracy of the data we collected. Instead we converted the java program for use as an ARFF builder only and decided to use the Weka GUI directly so we could easily start using more options and do more tests.

The approach we ended up using was used Weka's implementation of a neural network with 15

hidden nodes in one hidden layer for both classification problems. The input data that we used for the ALP and non-ALP classification is a set of selected pixels from two boxes, the first measuring 26 pixels by 14 pixels starting at location 16,4 and the second box measuring 6 pixels by 20 pixels starting at location 20,18. Obtaining the RGB values for each pixel results in 1452 input attributes and 2 possible output classifications(ALP, non-ALP). These areas were chosen to approximately select the blue 'Alberta', a part of the red license number and a part of the blue 'Wild Rose Country' on most of the pictures of license plates. The input data that we used for classifying the last digit is an area measuring 20 pixels by 25 pixels starting at position 47,9. Again taking the RGB values for each pixel results in 1500 input attributes with 3 possible output classifications(4,5,other). This area was chosen to approximately select the last digit in most images of license plates.

Questions

While performing experiments to determine the best efficiency a series of questions were determined and answered by these experiments. The first question was what is the optimal position of input for pixels and this was a fundamental question that needed to be answered as the position of our pixels were our input for our neural network and would be the attributes for the ARFF file, but what was the optimal amount for determining the license plate would not be the same case for determining the last digit of alberta license plates so we then have to determine what's the best input for that feature. The next big question that would help us solve this would be what pre-processing should be done to the picture if any. This was something that was essential for improving the tests but it would have to be applied to every picture. Another question was should we use all color information, a subset of it, or grayscale? The color could help with the red rose of alberta and blue lettering of

alberta and those are big determining features of alberta license plates, however using grayscale would be more efficient as it uses less input data. The next big question that took much of our experimentation time was based around what was the optimal amount of hidden nodes to be used. Should the same amount be used for both 4,5, and other as well as ALP, and non-ALP plates? This then expanded greatly when you look at how you can add multiple layers with different nodes on each layer. The next question to be asked is how are we going to classify the pictures and then how many should be used for testing and how much for training? Many small questions came out of this, such as should the training or test data have more of the pictures? or how much should be dedicated to each class because there were more non-ALP than ALP and more 'other' plates than 'fours' or 'fives', should this show in the data we use to train with or should it be an even split regardless. And after you determine all of that out then you can decide perhaps to use cross-validation instead and if so how many folds. The final question that needed to be answered was what about all of the extra options to be added should we change the values of other options such as learning rate and how does this affect results? Does momentum affect the results at all? Validation set options will speed up efficiency but will it greatly decrease accuracy and maybe that decrease is it worth it in order to do longer running options?

Experiments

We performed lots of experiments to determine what we thought was the most desirable setting to determine these license plates. Alot of experiments reduced their accuracy and some increased it. Some decreased the accuracy by a little but it was a small enough percentage that its increase in something else such as speed so the decrease was justified. All of the experiments done were linked to

our questions to try to solve them, a lot of questions arose by surprise from doing experiments we had not planned but everything we did increased furthering of our goal of learning about machine learning algorithms. Every experiment we did was done for both sets of data(ALP and last digit) with varying tests and varying results.

The first experiment for the the last digit 4,5 and other we tried applying some pre-processing to our image sets in which we used auto sharpen and auto contrast together on the pictures and then tried just auto contrast with these tests but we found that they both drastically decreased the accuracy so we decided to instead just use no pre-processing. We also changed the number of hidden nodes between 10,15,20,25 and concluded for this test the peak was about 15 hidden nodes was the best with the others varying about 38-40% accuracy with base option settings. We tried numerous multiples of hidden layers with different number of nodes on each layer, again ranging values from about 5-25. We tried a layer with 3 nodes as well because there are 3 possible outputs but the single layer always gave best results at about 45-47% depending on our other options. We changed the learning rates by adding 0.1 and decreasing by 0.1 being 0.2,0.4, and concluded on 0.3 as it usually had better rates by about 0.5%. We then changed the momentum again by 0.1 to get 0.1 and 0.3, and decided on 0.2 as it had better rates by a couple percent. Uur validation set sizes that we tried ranged from 5 to 20 to improve time and it helped cut the time down but it kept accuracy relatively the same varying only a partial of a percentage so it was a good implementation as decreasing the time allowed us to try cross-validation. We later discovered cross-validation and validation sets together lowered our accuracy.We tried a few different fold sizes but 5 was a decent speed to finish .Before we started doing cross-validation we originally were using a percentage-split of 80% training data which gave a higher percentage of accuracy from about 50-53% which we also varied with different percentages as well. However we thought that

while the cross-validation had a lower percentage it was a lot more consistent for each test and provides a closer approximation of actual accuracy for new untested images. We repeated all these tests with gray scaled images and there was only a slight decrease in accuracy so we decided to just keep the colour images. We also tried adding 'edge' information by using ImageJ's find edges function as an additional data point per pixel but its accuracy was severely lowered. In all of our experiments for tests for getting the last digit we had varying results from flipping a coin at 33.3% up to about 53.6%.

All of the above experiments were also done on the data set for ALP/non-ALP as well with mostly the same results. We did the same process with hidden networks but with the single layer we found that they were all almost indisguisable but the single layer was still better than the multi layer. The biggest difference was with the selection of pixels to use. We first tried using an area that selected the whole plate on average but then selected every second pixel from that area as it was a large area that included too many pixels to input. We then tried selecting more specific areas which ended up with the final selection of an area that included the blue 'Alberta' on most pictures and a second area that included a sliver of the red licence number and blue 'Wild Rose Country'. This proved to be the most effective.

After all the experimentation was done we settled on using default settings except setting the hidden layers to 15 for both problems and after running a 5-fold cross-validation test on it we got the following accuracies and confusion matrices:

Chris Medlock
Scott Deans
Charles Sedgwick

ALP/non-ALP

Correctly Classified Instances 477 81.5385 %
=== Confusion Matrix ===

```
  a   b   <-- classified as
260  19 |   a = ALP
 89 217 |   b = NONALP
```

Last Digit

Correctly Classified Instances 75 45.4545 %
=== Confusion Matrix ===

```
 a  b  c   <-- classified as
20 21 14 |  a = 4
17 25 13 |  b = 5
11 14 30 |  c = other
```

Conclusions

This project helped improve learning of the additional options available to neural networks including momentum learning rate validation sizes, and hidden networks and how they affect different sets of data. We learned while there may be good pre-processing tools out there that can help improve it, with the minimum knowledge in that field it is very hard to use it right to improve accuracy. We learned what kind of data works well with neural networks and furthered our knowledge on how to work with this data. We also learned that Weka requires a lot of memory to use and may possibly have a memory leak issue. There is a lot of options you can do to help improve neural networks such as speeding up the time by setting validation size. Overall this task is a very difficult task to accomplish at a high degree of accuracy.