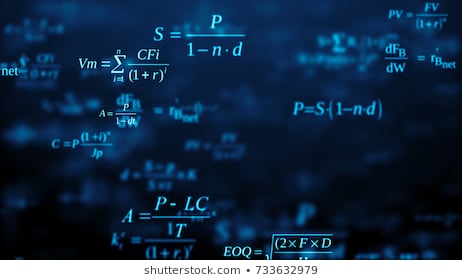
**Image Identification Using… Probability?**

**Ben Harwood**

**Introduction**

Consider an image. Any image at all. Perhaps this one:



Or maybe this one:



Computers don’t just know what pictures and images should look like. Computers break each image down into code and then interpret that code to reproduce the image. Indeed, even the images in subsequent sections have to be decoded by the computer to be rendered. So what?

The human eye receives light waves through the retina which are transmitted to the brain and interpreted. The field of computer vision crosses many disciplines, and its aim is to teach computers how to learn from images and gain high-level understanding from them. In other words, computer vision is a form of artificial intelligence that seeks to have computers do the things the human visual system is capable of doing. Again, so what?

Consider medical imaging. Specifically, an MRI of the brain and spine of a patient with multiple sclerosis. For the uninitiated, multiple sclerosis is an auto-immune disease where the body attacks the layer of fat surrounding the nerves (called myelin). A lesion (or sclerosis) is when the myelin in a certain spot is eaten away completely. For comparison, imagine taking a knife to a power cable and cutting away the rubber. Perhaps logically, multiple sclerosis is when multiple of these lesions are present. MRI scans reveal these lesions[[1]](#footnote-1), however computer vision can take this a step further and actually model where in the brain exactly these lesions occur.

Another example of the usefulness of computer vision is handwriting recognition. Suppose that a collection of handwritten documents was available. It could even be a sample of an individual’s handwriting. Computer vision techniques could be applied using these samples as a basis to determine if another document was written by this person. In the following sections, this idea will be explored.

**Section 1: Analysis and Models**

Normally at this stage a context discussion is order, however the context in this case will present itself as the paper progresses.

**Section 1.1: Data**

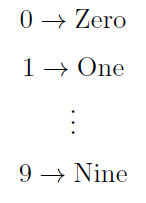
In order to demonstrate how computer vision algorithms fundamentally work (at least one way), two data sets are used. The first is a collection of 1400 images. However, as was mentioned in the introduction, each image is broken down into a code. But how does a computer encode an image? Pixel by pixel.

Each image is made up of a certain number of pixels. Generally speaking, a *pixel* is the smallest single component of any digital image. The more pixels present in an image the better the detail, clarity, color, etc. of the said image. In computer code, an image is turned into a vector of values that are, for each pixel, the color code. Without getting into an unnecessarily technical discussion about color codes, the primary of concern for the purpose of this study is the *value*, the intensity of the pixel, or the relative lightness/darkness. Values range from 0 for not present to 255 for the highest intensity. Each image was a handwritten digit from 0 to 9 and was comprised of 784 pixels. In the dataset each image is a vector 785 units long, with the corresponding values for each pixel intensity as well as a label indicating what number the image was.

Digit counts


Figure 1: Digit counts

There were no missing values present. From, a data cleaning perspective, all that was needed for the initial experiment was to convert the labels from digits into the corresponding words for use as category labels as seen below.



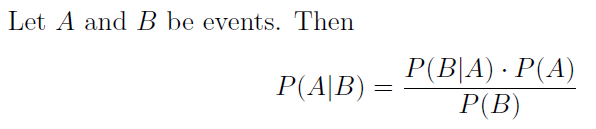
The second dataset consisted of information for an additional 1000 images, encoded the same way, however there were no labels.

**Section 1.2: Models**

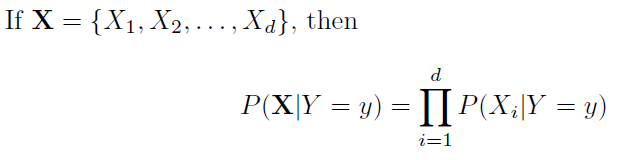
The plan of attack is to study the levels of intensity for each pixel and see what implication is present regarding which digit they represented. This will be done two ways. First with what is called Naïve Bayes.

**Section 1.2.1: Naïve Bayes**

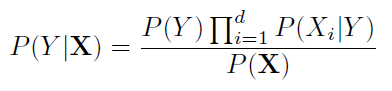
Naïve Bayes is based on Bayes’ Theorem from probability.



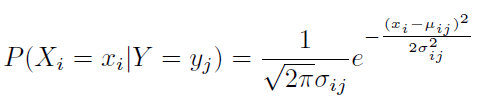
Bayes’ Theorem provides a way to calculate a conditional probability if the individual probabilities and the opposite conditional probability are known (or easily computable). In the case of a vector (like a dataset), a conditional probability is calculated by determining the conditional probability is done component-wise and then multiplying them all together:



Technically this only works if the components are independent, but independence is just assumed when using this technique[[2]](#footnote-2), and Bayes’ Theorem becomes

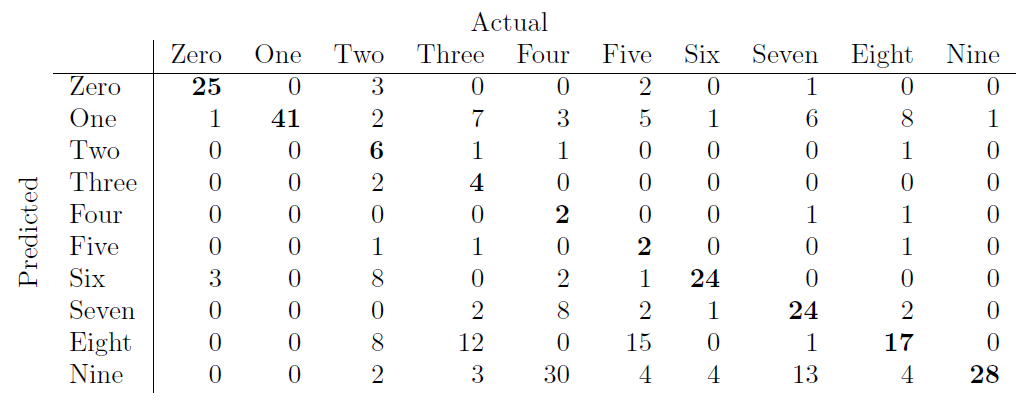


and because *P*(**X**) does not depend on *Y* the case (*Y*) that maximizes the numerator is likely “winner”. For categorical variables this all works very easily, however for continuous variables a distribution but be used to calculate the probabilities. Generally, the Gaussian distribution is used:



Now, with all of that groundwork laid, to determine the correct Naïve Bayes model to use, five separate models were developed. In each case, a random sample 75% of the original data was used for testing. Each model was then applied to the remaining portion of the dataset to see how well the model could predict which digit was represented.

As an example, the following table shows predicted labels vs actual labels. The bold face entries show how many were correctly predicted for each.

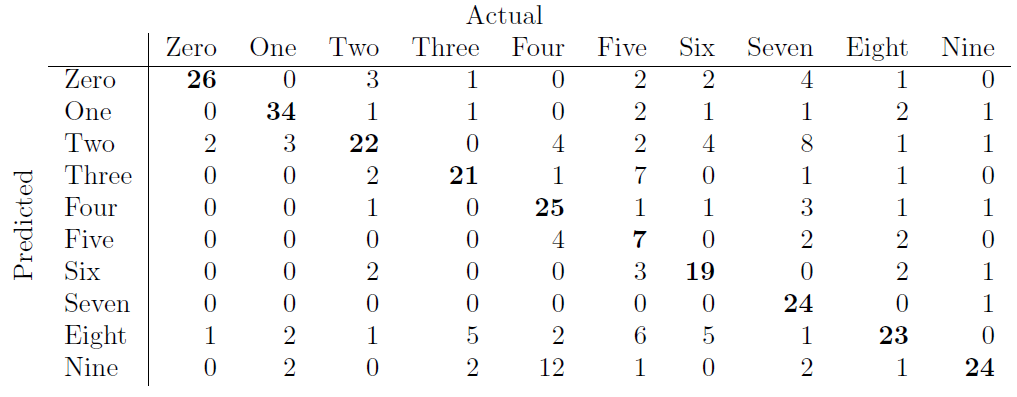


A fair level of success, however the following plots show how very different the predicted labels and actual labels were.

|  |  |
| --- | --- |
| A picture containing screenshot  Description automatically generated  Figure 2:Naïve Bayes predicated labels | A screenshot of a cell phone  Description automatically generated  Figure 3: Actual labels |

**Section 1.2.2: Decision Tree**

Having performed Naïve Bayes, decision trees were developed to see if they might be more accurate. The same subsets were used for model training as were used in Naïve Bayes. In this case, however, the values themselves needed to be discretized (categorized) for the decision trees to work. To this end, each value was divided by 255 (the maximum value it could be) to determine the “intensity percentage”. With this done, each pixel’s values were binned into 10% increments, so a value of 37 would be in the 20% range (37/255=14.5% < 20%), a value of 81 would be in the 40% range (81/255=31.8% < 40%), etc. With the pixel values all binned, five decision tree models were developed. Here are the results of the first (to compare to the Naïve Bayes model).



Here is a high-level view of the tree itself[[3]](#footnote-3):

A close up of a map

Description automatically generated

And here are the plots showing the predicted labels and actual labels once again:

|  |  |
| --- | --- |
| A screenshot of a cell phone  Description automatically generated  Figure 4: Decision tree predicted labels | A screenshot of a cell phone  Description automatically generated  Figure 5: Actual labels |

**Section 2: Results**

Having developed numerous models[[4]](#footnote-4) using both the Naïve Bayes method and decision trees, all with similar results, attention was turned to the secondary data set with no labels. Here is the Naïve Bayes result:

A picture containing screenshot

Description automatically generated

Figure 6: Naive Bayes label predictions

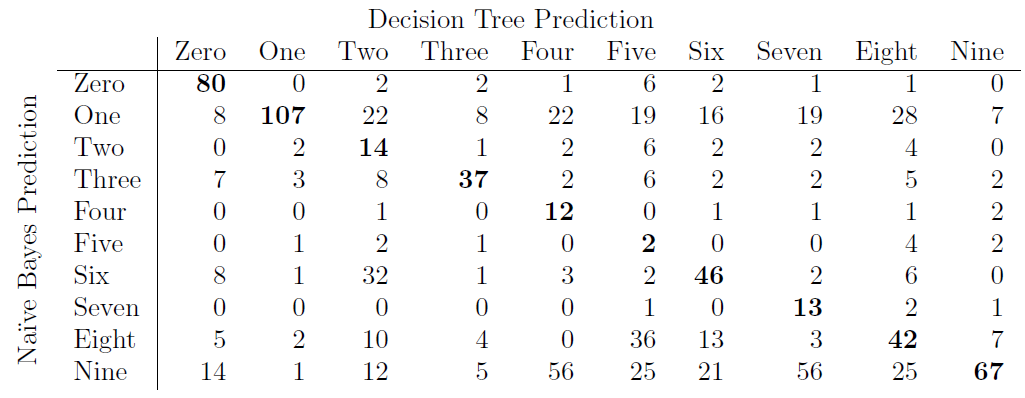
And here is the decision tree result:

A screenshot of a cell phone

Description automatically generated

Figure 7: Decision Tree label predictions

Finally, the following confusion matrix shows how well the two models matched in the predictions:



Some interesting things can be seen in this table. Notice that Naïve Bayes predicted a total of 282 9’s while the decision tree only predicted 88. Similarly, Naïve Bayes predicted 256 1’s but only 117 were predicted by the decision tree.

It is at this point that one might conjecture that Naïve Bayes in this scenario is not the best option. Indeed, in every NB model 1’s, 8’s and 9’s had vastly higher prediction rates than the other digits, but the decision tree models developed on the same training sets produced much more evenly distributed digit predictions, which also were more reflective of the overall data set and the testing set. From that perspective, it does seem that the decision tree is likely more accurate.

**Section 3: Conclusion**

It’s always a good idea to use multiple techniques to test results. As was seen in the previous sections, often times a perfectly logical choice of technique can yield less than satisfactory results. Previous studies considered the authorship of the Federalist Papers using two different techniques (one of which was used in this study). In that case as well, one method seemed to outperform the other.

Speaking of the Federalist paper authorship, could it be possible to further conclude who wrote the disputed papers using the techniques described here? Using the papers whose authorship in known, one might take each letter in the papers and treat them as described in Section 1.1, then follow the same path as outlined before. Granted, such a thing could be challenging since the actual pages themselves are most likely locked under glass at the National Archives or behind so many layers of security at the Library of Congress you’d need access to the Matrix to get in, but it would be a very interesting study. Assuming handwritten versions exist, of course.

In the end, the two avenues that were pursued here are not the only methods used in computer vision research. David Mumford, the Fields[[5]](#footnote-5) Medalist mathematician from Brown University, has pioneered research into computer vision since the 1980’s. His approaches are much, much more mathematical in nature, however he did champion the use of the technique in Section 1.2.1.

1. The present author’s brain and spine light up like a Christmas tree from these lesions. [↑](#footnote-ref-1)
2. Hence the name Naïve Bayes [↑](#footnote-ref-2)
3. Not the easiest tree to read, to be sure [↑](#footnote-ref-3)
4. Not including all of the different models for each was for brevity purposes, and to save on typing [↑](#footnote-ref-4)
5. The Fields medal is arguably the highest honor a mathematician can receive, on the order of the Nobel Price for Mathematics (since there isn’t one). It is awarded every four years. [↑](#footnote-ref-5)