Fiber Optic Core Image Detection: Comparison of classifiers

Michael Leighton
112 Middle Street
Manchester, NH 03031
Mikejleighton@gmail.com

Abstract

Visual inspection of fiber ends is often required during installation or maintenance of fiber optic cabling. Automated analysis first requires accurately determining the location of the fiber core. In this paper, we compare the accuracy and reliability of several different classifiers in finding the fiber core. Classifiers such as naive bayes, perception, and three layer feed forward neural networks have proven to be a reliable way of recognizing items in images. These three classifiers as well as a none learning contrast-based algorithm are applied to the problem of finding the fiber core and results are compared. Our results show that using a three layer neural network is the most accurate of the detection algorithms.

Introduction

Fiber optic cores are extremely small in diameter; a typical single mode fiber is about 65 microns. To put this into proportion a human hair can range from 50 microns to 180 microns. Because of their small diameter fiber optic cores can easily become dirty or damaged, hence visual analysis of the core is required before maintenance or installation. In many cases manual analysis can become tedious and automated analysis is preferred. Automated analysis is not possible until the location of the fiber core can be accurately determined within a given image.

Having an accurate and reliable way of determining the location of the fiber core would be beneficial, and would provide several useful applications. If analysis could be done completely autonomously it would be possible to take several images while in the field and batch them for latter processing. It would also provide a convince for technicians in the field, automatically detecting and capturing images of the core could save time and money.

Images of fiber optic end faces have several properties that lead to the uses of general-purpose methods in determining the location of the fiber optic core. First, the core of a fiber optic cable is circular in nature. The cladding, which is the area surrounding the core tends to be much lighter then the core itself. Finally, the diameter of the core is known under given magnifications.

Given these properties and the nature of the problem artificial intelligence techniques, specifically supervised learning, can be used to determine the location of the fiber optic core. In this paper, I will compare the accuracy of several different classifiers in solving this problem. I will also be comparing these classifiers to a contrast based solution that does not require any form of supervised learning.

I propose that a three layer neural network trained with the back propagation algorithm is the most reliable and accurate method for finding the fiber core. The results show that it detects the fiber core constantly with greater precision then any of the other approaches.

Approaches

The following are four approaches that can be used in determining the location of a fiber optic core in an image. Three of these approaches require supervised learning and a number of training examples. The fourth requires no training examples; it works completely off of the contrast of a given image.

Naive Bayes Classifier

The Naive Bayes Classifier is a purely probabilistic classifier based on Bayes theorem. The classifier must first be given a number of training examples and the class that each training example belongs to. Probabilities are then calculated and stored for use in classification. When classification of an observed data set begins, the probability that the given data set belongs to a class is calculated. The class with the highest probability is then chosen. The following equation describes the Naive Bayes model.

$$P(C \mid \chi_1, \ldots\ldots, \chi_i) = \alpha P(c) \prod_i P(\chi_i \mid C)$$

Like the name suggests the model is naive in the fact that it makes the assumption that each feature in the set is independent of every other feature. Since Bayes theorem

requires that futures be independent the true hypothesis can never be achieved with this model, however in practice the model is fairly accurate. The Naive Classifier scales well and is fast to train. Another good quality of the classifier is that it is not affected by noise.

Perceptron

A Perceptron is a single layer neural network trained with linear regression. The number of neurons is equal to the number of input features. The following equations describe the output, and weight updates of the network.

$$\hat{y} = \theta_1 x_1 + \theta_2 x_2 + ,....,+ \theta_n x_n$$
$$\theta_n = \theta_n - \alpha(\hat{y} - y) x_n$$

Note alpha is equal to the learning rate. This needs to be determined during construction of the network. A learning rate of 0.001 was used during the training process for data collected in the analysis portion of this paper.

Single layer neural networks are good at the classification of linear functions. However, they fail when the function that is being represented is not linear. They are also sensitive to noise data. Noisy training data can throw off the weights of individual neurons in the network causing errors during the classification process.

Three Layer Feed Forward Neural Network

The three layer neural network is the most powerful of the approaches and, unfortunately, the hardest to train. It is similar to the single layer network but because of its hidden layer cannot be trained with linear regression. The standard method of training the network is known as back propagation. The following equation describes the back propagation algorithm.

$$\Delta_i = Err_i \bullet g'(in_i)$$
$$W_{j,i} \leftarrow W_{j,i} + \alpha \bullet \alpha_j \bullet \Delta_i$$
$$\Delta_j = g'(in_j) \sum W_{j,i} \bullet \Delta_i$$
$$W_{j,i} \leftarrow W_{k,j} + \alpha \bullet \alpha_k \bullet \Delta_j$$

Like the name suggest the back propagation algorithm propagates the error of the network backwards through all of its neurons. To use back propagation an activation function with a derivative is required. My implementation of the back propagation algorithm uses the sigmoid function and its derivative.

$$g(x) = \frac{1}{1 + e^{-x}}$$
$$g'(x) = g(x) \bullet (1 - g(x))$$

There are several design decisions that need to be made while creating a three layer neural network. First, how many neurons should be in the hidden layer of the network? There are many rules of thumb on how to determine this number. I settled on 110 neurons by using trial and error. Next, what will the learning rate be? Similar to the single layer network a learning rate must be determined. I tried several rates starting at 0.001 but I found that the network learned slowly. When I increased this number to 0.05 learning was much faster and accuracy remained the same or better.

Finally, when to stop training of the network? The back propagation algorithm requires a number of training epochs. A training epoch is defined as one iteration through all of the training examples. During an epoch, error is tracked by summing the error for each individual training example. It may be possible to max or average over error for each example, however this approach was not taken. When this number drops below a given threshold training is complete. This is another number that can be modified to change the performance and accuracy of the network. The implementation analyzed here used a threshold of 0.05.

One important thing to note when designing a three layer network is that all weights should not be set to the same value initially. Error is propagated backwards based on the current weight of a neuron and the derivative of the activation function. If all weights are the same each neuron in the network will always contain the same weight and will never learn. This implementation uses a random number generator with a known seed to initialize weights.

Contrast-Based Algorithm

Unlike the previous three approaches the contrast-based algorithm requires no training data. The algorithm works as follows. First, the mean brightness of the image is found by summing the brightness of each pixel, then dividing by the total number of pixels in the image. Next, pixels below the mean are removed from the image. The remaining pixels are then clustered together and analyzed. A center point of each cluster is picked and distances from the outer most pixels and the center pixel are compared. If the cluster appears to be circular it is picked as the core. The algorithm is fast but can be easily thrown off by noise. For example if the image is dirty the fiber core may not be detected.

Data Collection

Three of the four approaches required training the classifier to recognize a fiber core. In order to train these classifiers training data needed to be obtained. I wanted all of the images to be of the same size and roughly the same brightness. I used a video fiberscope to capture 270 individual fiber end face images. I then randomly selected

50 of those images for later testing. The remaining 230 images were used as training data.

The next step was to manually select the location of the fiber core in each of these images. I created a small application that allowed me to draw a rectangle around the fiber core, and then save that location to a file. I then did this for all 270 images, both training and test.

The last step in collecting data to train the classifiers was to determine what the feature set should be. There needed to be a feature for each pixel in the image of the fiber core. Pixel brightness was used. An integer was generated for each pixel (0 being the darkest and 10 being the brightest). The final training file was then generated by scanning over each of the 270 images with a rectangle that was the same size as the fiber core. If the rectangle contained the core it was labeled with a class of 1, if it did not contain the core it was labeled with a class of zero.

Initially full size images were used to generate the test data and to classify the images. The initial image size was 720 by 576 and had a fiber core size of 125 by 125. The feature count for these images was over 10,000. This made training and classification very slow. The solution was to shrink each image to 45 by 36 giving a fiber core rectangle a size of 17 by 17 and a feature count of 287. It is important to note that some data is lost by scaling the images but training and classification becomes much faster.

Classifying an Image

Searching for the fiber optic core with the contrast-based algorithm was straightforward. The algorithm was given an image to search and it returned the location of the fiber core. The three classifiers however required a different procedure.

The classifiers were trained to give an output of one if the observed data resemble a fiber optic core and zero if it did not. Searching for the core required scanning over the image with the classifier. The classifier ran at each pixel location in the image, and the core was located by maxing over the output of the classifier.

Analysis

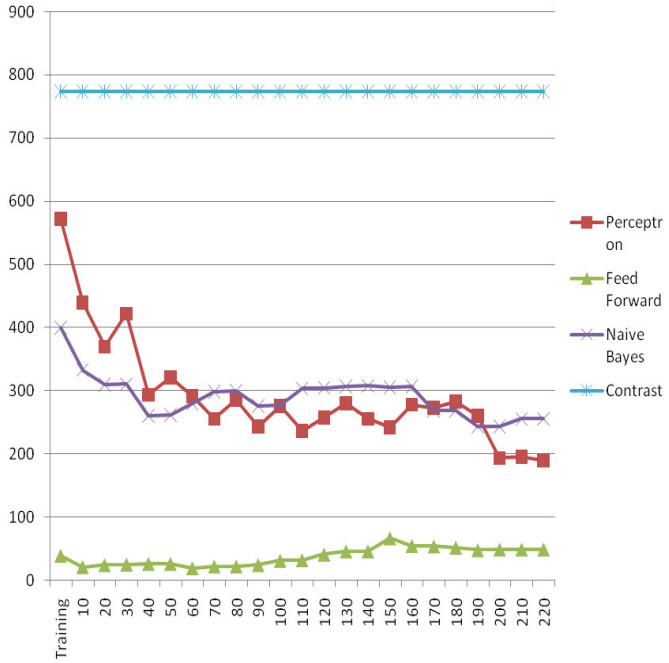
The easiest way to test the accuracy of a classifier is to run it on a set of data with known output and tabulate the number of times it is incorrect. This form of testing did not give me a good estimate of how well the classifier would perform when searching an image; it also prevented me from comparing the contrast-based algorithm with the other classifiers.

To solve this problem I manually selected the location of each fiber optic core in all of the test images. I then searched the test images using each of the classifiers and the contrasted-based algorithm and calculated error by using the Manhattan distance of the returned location and the location that I manually selected.

The contrasted-based algorithm does not learn but the 3 other classifiers have the ability to perform with better accuracy given more training data. To analyze this I trained each of the classifiers with a small amount of the training data and calculated error by running the classifier over all of the 50 images in the test set. Then incremented the training counts and recalculated, and continued this until I had used up all of the training data. The following graph was generated. The y-axis represents the total number of pixels away from the location that I manually selected over all 50 images. The x-axis is the amount of training data that was used to train the classifier. I also placed the contrasted-based algorithm on the graph for comparison; however it is important to note that it does not use training data and therefore remains constant as training data increases.

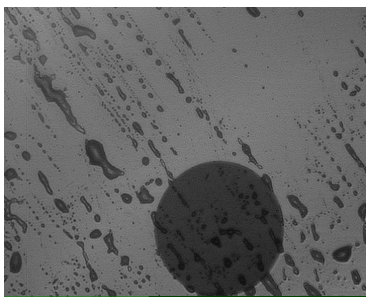
I was also curious to see if there was a pattern in the kind of examples that each classifier got incorrect. To do this I summed the error for each image and each classifier. This allowed me to see which images each classifier did the worst on. The results are displayed along with their average error over all training counts.

The final two comparisons are time based. Benchmarking was done with a Centrino Duo machine running at 2 GHZ with 2 GB of ram. The Classifiers were constructed in a similar manner and share the same un-optimized design. Time comparisons are meant to be relative and not final times.

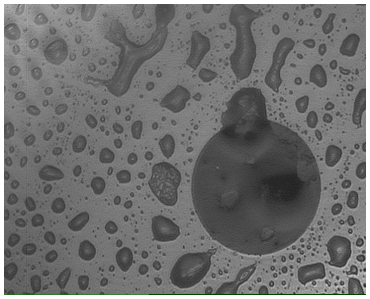


Training Count vs. Error (Pixels)

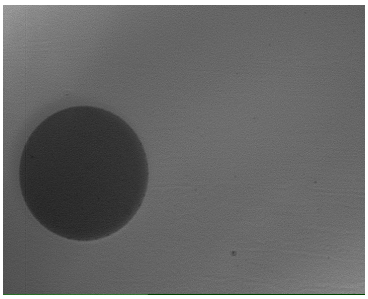
Naive Bayes Classifier



Average Error: 36 Pixels.

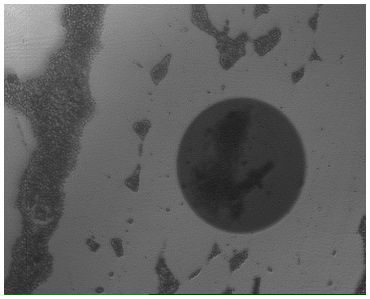


Average Error: 36 Pixels.

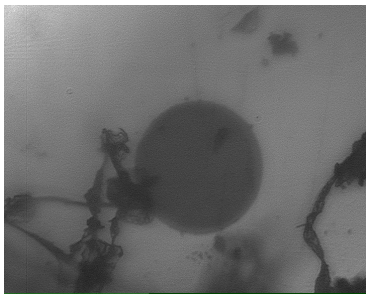


Average Error: 3 Pixels.

Perceptron

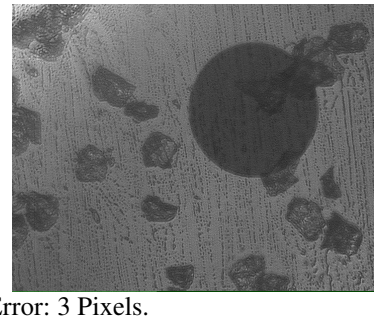


Average Error: 24 Pixels.



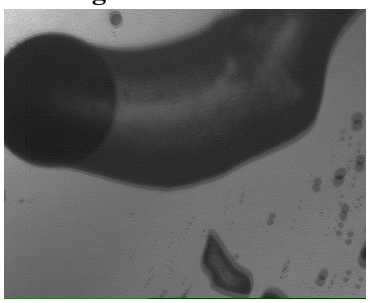
Average Error: 17 Pixels.

Three Layer Feed Forward Neural Network

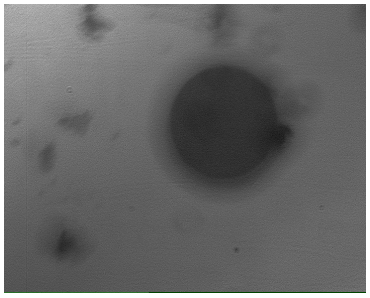


Average Error: 3 Pixels.

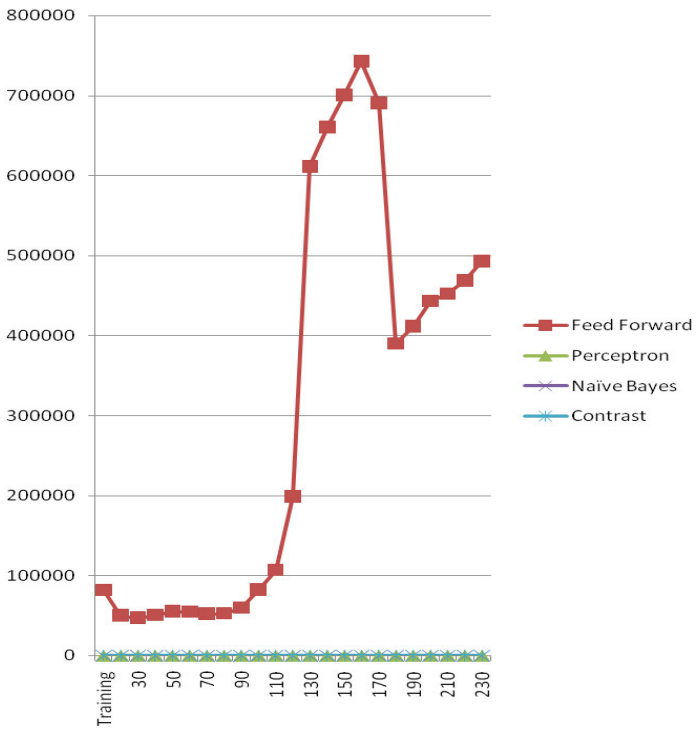
Contrast-Based Algorithm



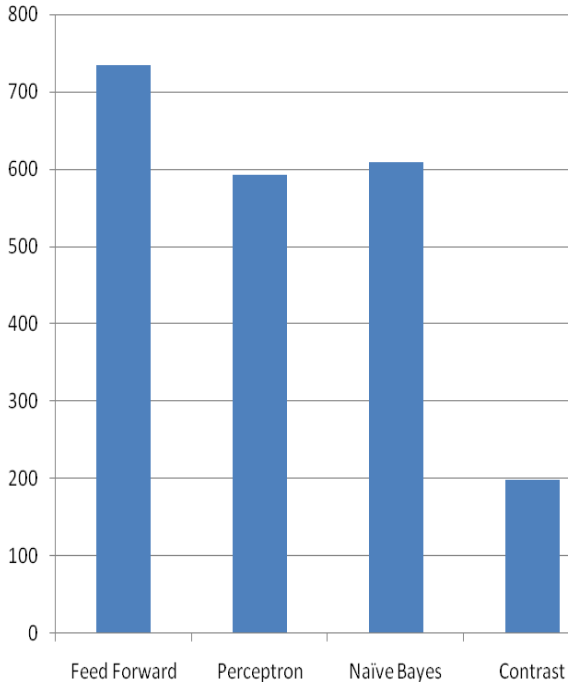
Average Error: 64 Pixels.



Average Error: 52 Pixels.



Training Count vs. Training Time (ms)



Time to classify image (ms) vs. Classifiers

Discussion

It is clear from the error graph that the three layer feed forward neural network trained with back propagation learns with fewer examples and is more accurate in identifying the fiber optic core in an image than the other three approaches. The three-layer network starts off at about 41 pixels of error total over the 50 images. This means that with 10 training examples it was able it pick a location, on average, that was 1 pixel away from the location that I had picked manually while creating the test examples. It continued to perform well as training examples were added, but there were some fluctuations. At 150 examples it jumped up to 67 pixels of error. It is not clear why this happened, but one possible cause could be that the additional training examples did not match the test examples well. Since the number of hidden neurons was picked by trial and error it is also possible that the network was over fitting at different training sets. Over fitting is when there are too many hidden nodes in the network to match the true hypothesis well.

The Naive Bayes classifier started off at about 400 pixels of error with 10 training examples and finished off at 250 pixels of error with 230 examples. The classifier was clearly learning, though error did occasionally go up with more training examples.

The single layer network did the most learning out of the three classifiers. It started at about 600 pixels of error with 10 training examples and ended up with just fewer than 200 pixels of error with 230 training examples. Like the other two classifiers error would occasionally go up with training data but in general it would go down.

The contrast-based algorithm had the least amount of accuracy. It is however important to note that the evaluation was not completely fair to this algorithm. When I created the test data and selected the location of the fiber optic core I gave it a slight boarder. Since the classifiers were trained with this boarded they would pick a location that also supplied this boarder. The contrasted-based algorithm picked a location containing the fiber but did not account for this boarder giving it a natural error of a couple of pixels per image. This could very quickly add up over 50 images. If I had more time I would remove this boarder from both the test and training data to get a more accurate comparison.

When comparing the images that each classifier did the worst on, it's easy to see that dirty fiber appears to be the culprit. There is an exception; one of the fiber images that the three layer network got wrong most frequently is a clean fiber image. I went back to my test data and checked the point on the image that I manually selected. It appears that I picked the point without giving it any boarder. This is most likely the reason why the classifier misclassified it more then any other image.

The training times graph clearly shows that the accuracy of the 3 layer network comes at a cost. It is much slower to train then the other three. It is also important to note that there is a drop in training time. This could be caused by variations in the training set. The contrast based algorithm

is placed on the graph as a reference but it will always be zero since it does not train.

The last comparison is running time (time to classify an image). The 3 layer network is the slowest of the three. Perception and naive bayes run at similar speeds, about 100ms faster then the 3 layer network. The contrast based algorithm is the fastest. The graph is meant to be used as a relative comparison. The classifiers have not been optimized and have the potential to run faster. In general however if the same optimizations were made to each classifier relative speeds would remain the same.

Extensions

In the future I would like to run the same comparison but with more test and training data. It is clear to see that both the naive bayes classifier and the perceptron were learning at a decent rate. I am curious if they would have matched or exceeded the three layer networks accuracy given more examples.

Also there are several techniques that I would like to try to speed up the classification process and increase the accuracy of the classifiers. One of these techniques involves training several classifiers on different size images, starting from very small images and moving to the full size image. The small classifiers would then be used to find the general location of the fiber core. The larger classifier would then use that location so it does not have to scan the entire image. This could possibly speed up the classification process.

A genetic algorithm or simulated annealing could also be used during training of the three layer network. As mentioned earlier many decisions need to be made before the network can be trained using back propagation. Variables such as the number of nodes in the hidden layer, the learning rate, and the termination error rate can all be modified to affect the performance of the network. An algorithm could determine combinations of these variables that would lead to a more accurate classifier. Finding a true optimal solution could be impossible since the number of possible combinations is infinite. Using a local optimal approach such as a genetic algorithm or simulated annealing would likely lead to good results.

It is also interesting to see that each classifier had it own weakness. While most of the images that each classifier had trouble with had dirt in them, they where still different dirty images. This leads me to believe that a boosting technique such as ADA boosting could help combine the classifiers to make a very strong classifier that is more accurate then any one of than alone.

Acknowledgments

I would like to thank Wheeler Ruml for several tips given during the implementation process of the project. I would also like to thank the engineering staff at Noyes Fiber for lending me the equipment required to get training and test images.

References

Russell, Stuart and Norvig, Petter 2003. Artificial Intelligence, A Modern Approach 2nd edition. 20:716-738

Davies, E. R. 2005. Machine Vision : Theory, Algorithms, Practicalities 24:691-700

P. Langley,W. Iba, and K. Thompson. An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 399–406, San Jose, CA, 1992.AAAI Press.