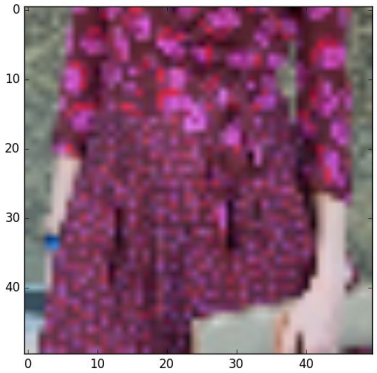
Machine Learning Week 6-8 Project
Nicolas Johnson
19-Oct-2016

For this project, a data set of 15,702 images of dresses was used to train a machine learning model to be able to classify which of 17 dress patterns was shown in the image. The data were labeled ahead of this project into the 17 classes, such as floral, stripes, polka-dot, plain, or animal. The images were full-color RGB format as png files stored in a publicly accessible AWS S3 storage location. Also, the images were marked with a red rectangle around the actual dress in the image, so as to enable filtering out of much of the irrelevant background in the images.

Data Preparation

To begin the process, the data needed to be cropped and validated. Using the source data file from the CrowdFlower data set Image Categorization: Dress Patterns (https://www.crowdflower.com/wp-content/uploads/2016/07/dress_patterns.csv), which contains labels and URLs for each of the 15702 images, a python script was created to download the images. These were saved to local storage in case of a need to reprocess the data, without having to re-download them again, as this data transfer took several hours to complete.

Using the OpenCV package in python, which gives us the ability to alter graphical images and detect shapes, the red rectangles were found in each of the images, and then cropped out. Because the rectangles were sometimes horizontal and sometimes vertical, they were then cropped to a square the size of the narrowest side of the rectangle. Then, in order to make the data modeling activities work more efficiently given the CPU capabilities available on a desktop computer, the images were reduced down to 50x50 pixels. The figures below show an example of an original image, and the cropped image scaled down to 50x50 pixels.



With the image formatted to this consistent size, focused on the center of the dress, I still had to deal with 7500 data elements per picture as follows: 50 pixels across, 50 pixels down, and 3 values for Red, Green and Blue to get the colors. This means that 50 x 50 x 3 = 7500, so I had a lot of data for this imaging problem.

Another problem I had to deal with were bad images. Of the 15702 images, 87 of them had incomplete red rectangles or had been delivered in black and white. These images were simply excluded from the dataset, leaving 15615 images. With these, the samples were split into a data set for training our machine learning model - 70% for the training data to build the model, and 30% to test it for accuracy.

Preparation proved to be incredibly important, as I had an error in my code that I didn't find until after generating the models. I had left out a couple of zeros in the code, which caused the data being modelled to be just 5x5 pixels of the 50x50 pixels. The remaining ones were garbage values, causing terrible results in the models. Unfortunately, the data had to be re-processed and the models regenerated.

Model Generation

Several kinds of models were built in an effort to find the best way to build a tool to classify the dress patterns. I started with a Convolutional Neural Network, which is considered one of the best models to dealing with image data. The parameters for this model were largely thanks to the example from the fantastic class resources from Machine Learning 54000 at Lewis University. This model is a complex web of six layers of various functions on the 7500 input data elements, feeding subsequent layers in different ways. After the process refined the model over 10 iterations, I had the best model of all I generated that predicted the test data at a precision of 54% and a recall of 63%. That means that for all dress patterns classified in some category, 54% were classified correctly, and for all the dress patterns in some category, 63% were chosen correctly.

This model was by no means great, so I tried with a simpler model based on an example in the Keras documentation. It is still a convolutional neural network like before, but a bit more simple, with basically one convolutional layer, which allows the functions to feed back to itself as it refines the model. There was also a layer to shape the final output data to our 17 possible dress patterns. This model was quite a bit worse than the one before, with 29% precision and 54% recall.

The next model was a non-convolutional neural network, based on a previous homework assignment where I had found a simpler 3-layer neural network that starts with 100 nodes connected to our 7500 inputs, then connected to a 500 node layer, and finally to a 17-node layer mapped to our 17 dress patterns. This model achieved the exact same accuracy as the previous neural network, with only 29% precision and 54% recall.

With neural networks performing poorly, I tried logistic regression with the Keras library, and a stochastic gradient descent to attempt to minimize the error as the model was refined over 100 iterations. However, this model again returned the infamous 29% precision and 54% recall.

To try something different, I implemented another logistic regression model, this time using the ScikitLearn module in python. This approach took about a half hour to complete, compared to less than a minute for the Keras procedure. However, the results were a bit different with precision at 41% and recall at 50%. This is still a bad model, as it's getting it right less than half the time out of the 17 dress patterns.

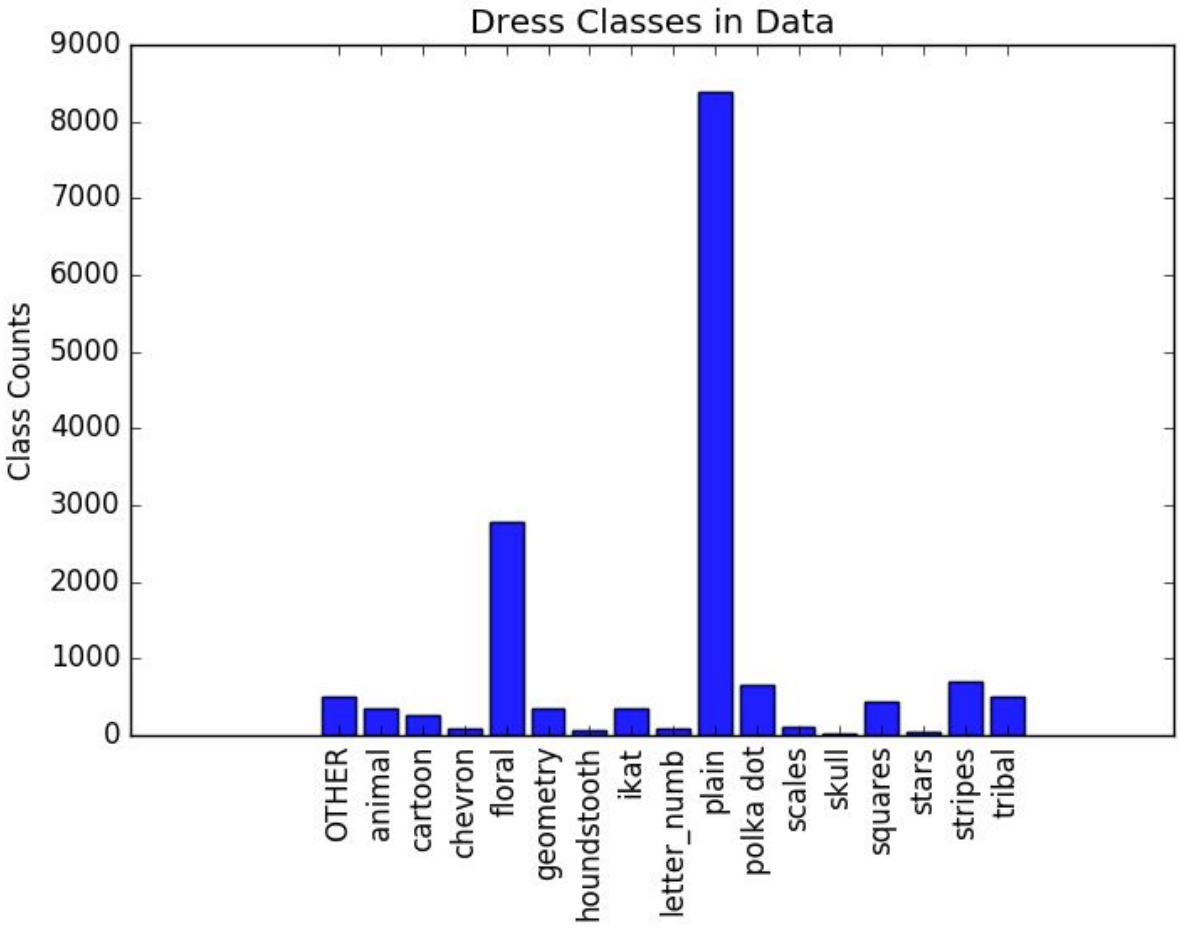
With those results, I tried a random forest model, which are generally not considered great for image data. This model generated 100 decision trees to attempt to tease out the various dress categories, and then took the best of them. This model resulted in a precision of 59% and a recall of 58%. The random forest is actually working much better than the neural networks and logistic regression models, and better precision than even the convolutional neural network. The

convolutional neural network still has the best recall, however. It's still not a great model, but it has precision and recall better than 50%.

Conclusions

The models generated are not great, but they serve as a good starting point to enhancing the image classification problem. The models would not be useful for any real classification system. Image data is notoriously challenging, as the best algorithms have many parameters to tweak, and require plenty of computer processing power.

As we run the machine learning predictive models against the test data, most test records are being predicted as a "plain" dress pattern in all of the models. Apparently, our data is unbalanced among the classes - over half of the original data set is labeled as "plain". This is illustrated in the figure below. If a deeper process was to be undertaken for this dataset, I'd recommend balancing out the data to more even levels.



There are enhancements to this exploration that can be made, should the pursuit of further models over this data be warranted. Other recommendations I'd consider are increasing the image size from 50 x 50 pixels to something much larger, say 500 x 500 pixels. This would allow more of the pattern in each piece of clothing to be seen by the algorithm. I'd also recommend exploring larger convolutional neural networks, which may require the use of larger computing resources or computers utilizing GPUs. Without these enhancements, the training of the models would be extremely long and CPU intensive.