CS109B Final Project: Movie Genre Prediction

Project Milestone 5: Final Submission, Report and Screencast

Group 40: Sahbi Ben Gdaiem, Isabelle Mieling, James Zatsiorsky

May 3, 2017

Screencast link: https://www.youtube.com/watch?v=GKgEMhrcYDg&feature=youtu.be
Github link: https://github.com/gsahbi/cs109b

Overview

More than 250,000 professional films have been made since 1920. With this many movies, there is an enormous amount of data available for analysis, discussion and exploration of various questions and challenges. In this final project, we aimed to address the challenge of assigning genres to a movie, using both a trained and untrained convolutional neural network. Using a pre-trained CNN VGG16 model resulted in a 82.3% test accuracy.

Introduction

The earliest moving pictures came about in the 1850s, with the first short motion pictures arriving in the 1890s. These motion pictures captured and emphasized movement with no sound, usually no plot and no story. One of the earliest movie shorts was created by the Lumiere Brothers in France. It featured a collection of 15-30 scenes showing a train arriving at a station, a man watering his garden, and people getting off a ferry: every-day actions. This first phase of motion pictures was very primitive, compared to today's standards. As time passed however, film makers and their films advanced and started telling stories.

It was around the 1920's, the so-called 'Golden Age of Motion Pictures', that we began attributing genres to films. Movie attendance was increasing steadily and movies were primarily being made for entertainment. It was during this time that movies began to develop into a series of 'formulas'- what we would call 'genres'- which included Westerns, Adventure, Serials, Horror Films, Gangster and Musicals. Since then, the movie industry, like many aspects of American and international culture, has evolved and expanded drastically.[1]

Since 1920, about 250,000 movies have been collected and make up part of the IMDb and TMDb databases for movies, TV and celebrities, covering 20 genres.[2,3] IMDb is an online database of information related to films, television programs, video games and more. It contains a variety of information on films, including year, popularity score, revenue, movie poster, release country, genres and more. Similarly, TMDb is a movie database containing information on movies and TV shows. With the vast amount of data contained in these databases and the wide expanse of machine learning algorithms available, we aimed to address the challenge of assigning genres to movies using predictive data from these databases.

We gathered data from the ~250,000 movies in IMDb and TMDb into one database and used both a pre-trained and untrained CNN machine learning algorithm to predict move genres for a testing set of movie. We ran the algorithms using Amazon Web Services (AWS).

Data Exploration

We gathered two primary datasets: the first containing metadata collected from IMDb and TMDb for each year from 1920-2017 and the second containing movie posters for each movie that has one (not all posters in the TMDb dataset have posters). This amounted to ~250,000 movies and about 12 Gb of data. These databases contained a variety of features for each movie as well as the genres a movie falls into. The lists of possible genres and features for each move are given in **Table 1** below.

Movie Genres	Action, Adventure, Animation, Comedy, Crime, Documentary, Drama, Family, Fantasy, History, Horror, Music, Mystery, Romance, Science Fiction, TV Movie, Thriller, War, Western
Movie Features	tmbd id, imdb id, title, tmdb genres, imbd genres, poste url, release date, popularity, vote count, vote average, original language, budget, runtime, revenue, production companies, production countries, imdb rating, year
Table 1:	Movie Genres and Movie Features from dataset.

Initial exploration of the data revealed that there were some inconsistencies; IMDb and TMDb did not always put movies into the same genres. As this was only true for a small amount of the data, we decided to use TMDb genres. Also, both sites did not always use the same terminology for equivalent genres. For example, TMDb uses the term 'Science Fiction' whereas IMDb uses the term 'Sci-Fi'. Therefore, we had to adjust for these inconsistencies and ensure the genres were labeled the same way in our datasets. We also encountered a lot of missing data. Most critically, some movies were missing genres (genre = NaN) and some movies were missing IMDb rating, which we hypothesized might be a key predictor in our model. Lastly, the movie genres in the datasets were imbalanced. These challenges, as well as others surrounding the data and genre prediction, will be discussed in the following sections.

Imbalanced Data

One large potential challenge to address was the imbalance of movie genres in the dataset. As there were some genres with more movies than others, this posed a challenge in terms of future genre prediction, as an imbalance of classes in our training set could cause overfitting on some genres. A bar graph is given below in **Figure 1** showing the movie counts for each genre in the dataset. We can see from this visualization that the data was quite imbalanced; a small number of classes had a disproportionately large number of counts. In particular, there were far more dramas and comedies than any other genre. As mentioned, this imbalance could cause a problem with prediction, as we would have likely overfit to the more prevalent classes and underfit the classes with a smaller prior probability.

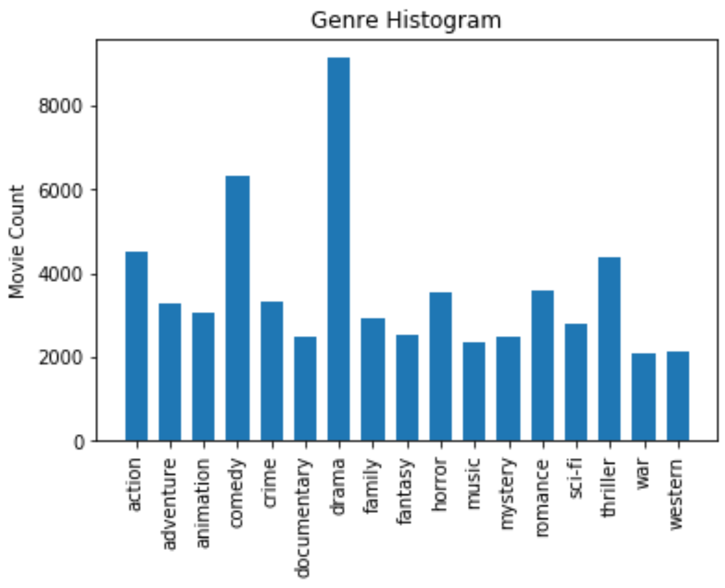

Figure 1: Histogram of movie count per genre

In order to combat the issue of the imbalanced nature of the data as mentioned above, we took a stratified sampling approach. We decided on stratified sampling as it is beneficial when subpopulations within an overall population vary, as was the case in the dataset. We had over 250,000 movies in our dataset and 20 genres into which they fell.

We first took an undersampling approach: keeping all the movies in underrepresented classes but taking only a subset of the movies in overrepresented classes (i.e. dramas, comedies) in order to balance the data. Through stratified sampling, we divided the data into homogeneous subgroups before sampling, therefore ensuring equal representation for each genre in our training set.

As is shown above in **Figure 1**, 'tv movie' was the least common genre, with 3,532 movies falling into this genre. Therefore, we started with ~70,000 samples, ~3500 samples drawn from each of the 20 classes (genres). We then took a subset of 20,000 samples from this for training. Since we had many movies to choose from for the overrepresented classes, we only selected movies in which the genres reported from IMDB and TMDB matched. We believed that these movies might be more helpful than others in serving as part of the training data if both sources give the same genre(s).

Missing Data/Values

A second challenge to address was the issue of missing data. Some movies were missing genre data all together, while others were missing IMDb rankings, revenue, etc. We had to decide how to handle this missing data so as not to bias our results. If we removed rows with missing data, we would risk biasing our results around only more well-known movies. If we removed columns with missing data, we would no longer be able to use them as predictors.

One solution for the columns with many missing values was to remove these columns all together. For example, revenue and budget were 0 for many movies and thus we removed these as predictors. Production countries also had many missing values, though far less than the number of missing values for revenue and budget, and therefore we replaced missing values with 0 and included these in the model. There were also some IMDb ratings of 0 and we replaced these with the average rating of all the movies with ratings, 6.5. Lastly, we decided to drop the ~55,000 rows out of 242,000 rows with NaN genres as there was nothing more we could do with movies without a genre.

Modeling

Input - Model Features

As mentioned above, we used subject-based knowledge as well as number of missing values to determine which variables to include in our models. From the metadata dataset, our features included title, release date, popularity, runtime, original language and production country. We thought title would be an interesting feature to explore through some text analysis. We used as predictors the number of words in the title and the average length of a word from the title. We thought year would be informative since it is likely that different periods saw varying trends in popular movie genres. Additionally, we believed popularity might be indicative of genre, especially when paired with other features for interaction effects. For instance, it is intuitive that western movies would be more popular in the U.S. than in Europe. We also believed runtime would be intuitively indicative of genre. Comedies tend to be shorter than long-winded fantasies. Also, we included original language as it might be indicative of genre as different regions of the world may have different distributions of movie genre popularities.

For the movie poster database, our choice of features for X was rather limited. For each poster, we had the (R,G,B) triplet for each pixel in the image. We did not plan to use traditional machine learning techniques that we learned at the beginning of this course on this dataset. Instead, we used deep learning on this data to find patterns in the poster images.

Output

We decided to train a binary classifier for each genre, then aggregate the results. Each movie would have 20 outputs (one for each genre), and we trained a separate model for each output. For example, the movie "The Mark of Zorro" belongs to the genres western, adventure, drama, action, and romance. So, we have the outputs Y_{Action} = 1, $Y_{Adventure}$ = 1, Y_{Drama} = 1, $Y_{Romance}$ = 1, $Y_{Western}$ = 1, and Y = 0 for all other genres. When training the model for adventure movies, we used the output $Y_{Adventure}$ = 1 to train the binary classifier for adventure. After aggregating the results, we were able to identify one or more genres that a movie belongs to. Since we were not looking for a mutually exclusive outcome, we proceeded with a Convolutional Neural Networks (CNN) model for multi-label classification.

Model 1: Random Forest Classifier

The first model we tested was a random forest classifier. In our implementation, we set the *class_weight* parameter of the model to "balanced". This helped us deal with the imbalanced nature of our dataset by assigning weights to samples in order to even out the imbalance in the class labels. We used area under the Receiver Operating Characteristic (ROC) curve, AUC score, as the performance metric for this model. We decided not to use accuracy for this model as accuracy is not always intuitive when dealing with imbalanced data. By looking at the ROC curve and associated AUC score, we were able to evaluate the performance of the classifier. We then trained random forest classifiers for each of the genres in our dataset. We produced ROC AUC plots to visualize the performance of our models, given below in **Figure 2**.

Model 2: Extra Trees Classifier

The second model we tested was the extra trees classifier. Like in the random forest model above, the extra trees classifier used a random subset of candidate features. However, the thresholds for extra trees were drawn at random for each candidate feature, and the best of the randomly generated thresholds was used when splitting the decision trees. Although we slightly increased the bias of our model by randomly drawing thresholds, it tended to decrease the variance of the model. The performance we got was quite similar to the random forest model above. We used the AUC score as a metric to evaluate the performance of this model. We trained extra tree classifiers for each of the genres present in our dataset, as we did for random forests. We again produced ROC AUC plots to visualize the performance of our models, given below in **Figure 2**.

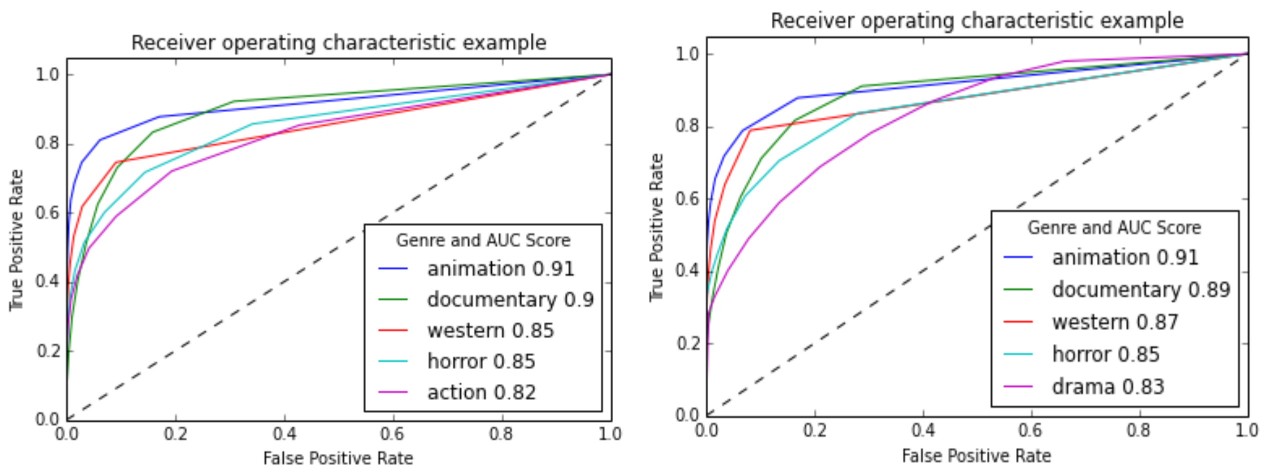

Figure 2: Receiver Operating Curve for Random Forest Classifier (left) and Extra Trees Classifier (right) for five movie genres.

Generally, we were pleased with the performances of our two models. We were able to get significantly better-than-chance classifiers from some very basic meta-data on the movies, such as runtime, title, release date, and production language. If our classifiers were only as effective as a coin flip, then the curves would coincide with the dashed black line at y = x. Instead, all the ROC curves for the Random Forest Classifier model resided in the upper-left quadrant, indicating that we trained classifiers that were significantly better than chance. We computed AUC scores, as described in the sections above, and ranked the genres based on their AUC scores. We had more success classifying genres with high AUC scores. Based on this ranking, we saw that we had the most success classifying animated movies. This was followed by documentaries and western movies. Our classifier had a more difficult time classifying mystery and crime movies, on which it performed the worst.

For the Extra Trees Classifier we also saw that all of our ROC curves were in the upper-left quadrant, so we again produced better-than-chance classifiers for this model. The ranking of genres based on AUC score were similar to that from the random forest classifier, which makes sense since these two models are quite similar. Again, animation was at the top of the list, followed by documentaries and westerns. This model had the hardest time with fantasy and history movies, for which it had the lowest AUC scores.

Convolutional Neural Network

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 98, 98, 32)        896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 96, 96, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 44, 44, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 22, 22, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 30976)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               15860224
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 17)                8721
=================================================================
Total params: 15,934,513.0
Trainable params: 15,934,513.0
Non-trainable params: 0.0
```

Table 2: CNN model parameters

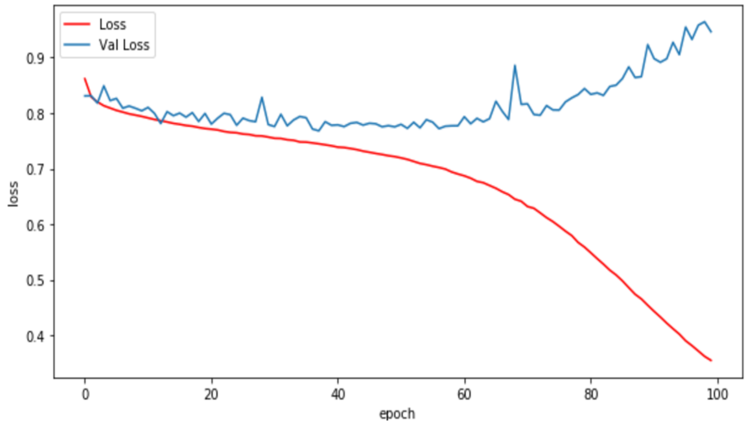

Figure 3: Accuracy vs. Epoch for CNN

CNN Trained from scratch

We trained and optimized a CNN model from scratch using 2D convolution layers. This network had one input layer, two hidden layers, and one output layer. The predictor poster has 3 RGB layers of square matrix of 100x100 pixels.

To an empty network model, we added an input later with 16 filters, a (5,5) kernel size, and a rectified linear unit activation function. We then did max pooling with the 2,2 default stride behavior. We added 2 hidden layers to the network, the first with 32 filters and the second with 128 filters, both with a (5,5) kernel size and rectified linear unit activation function. We did max pooling after both 2D convolution layers with the default 2,2 stride behavior. Lastly, we flattened the data for a fully-connected classification layer and added the output layer: a dense layer with 17 output values for the 17 classes/genres in the model, as show in **Table 2**. The loss function used to determine the accuracy of this model was a binary cross entropy loss function with a stochastic gradient with a learning rate of 0.01, decay of 10^-6, and momentum of 0.5. We used a batch size of 256 and 10 epochs. We provided the model with validation data in order to determine when the training should stop.

Optimization

We tested several SGD learning rates and decays for an optimal learning curve. We also added class weights as the probability of each genre in the data set to account for the unbalanced nature of the data. All these optimizations were run with a smaller subset of data. We also trained the model with sigmoid prediction layer and binary cross entropy loss, which is more suited for multi label classification. The performance of the model can be visualized in a plot of accuracy vs. epoch below in **Figure 3**. We used a Matthews correlation coefficient to further enhance the quality of binary classifications by calculating optimal cut-off thresholds per genres. The per genre accuracy we received can be seen in **Figure 4.**

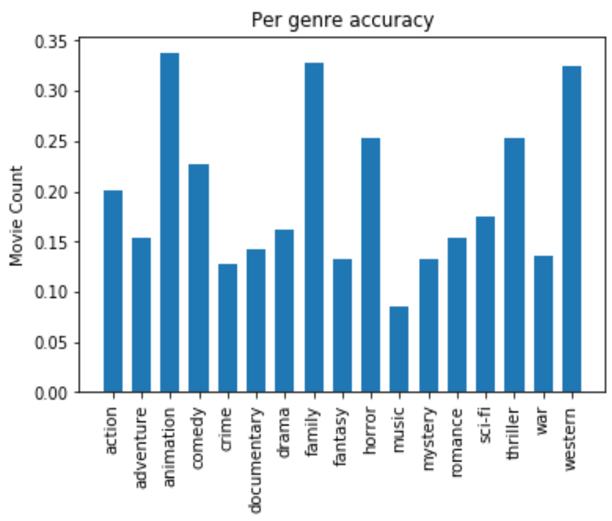

Figure 4: Per Genre Accuracy

Pre-Trained CNN: VGG16

We used the VGG16 model, which has weights that are pre-trained on ImageNet and calculated the prediction for every posted in our data set. This model was introduced in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition"[4] and uses very small convolution filters. Then, using a normalized cosine similarity matrix, we found out that VGG16 was able to pick visually similar posters and therefore could yield promising results in genre classification.

We remove the top fully connected layer using include_top: False. Then we recovered the same 3 fully connected Dense layers with ReLU activation and we replaced the last sigmoid prediction layer with 17 classes since the VGG16 pre-trained model was made for 1000 classes, but we only have 17 classes (genres). We also marked all layers not to be trained except the last prediction layer which was a sigmoid. We trained the model using a SGD with a learning rate of 0.1 and decay = 10-6.

In the example below, **Figure 5**, we see that from an input animation movie poster (Ice Age) we obtained many others which feature the same genre and visuals. This pre-trained CNN resulted in a test loss of 1.223 and test accuracy of 0.823.

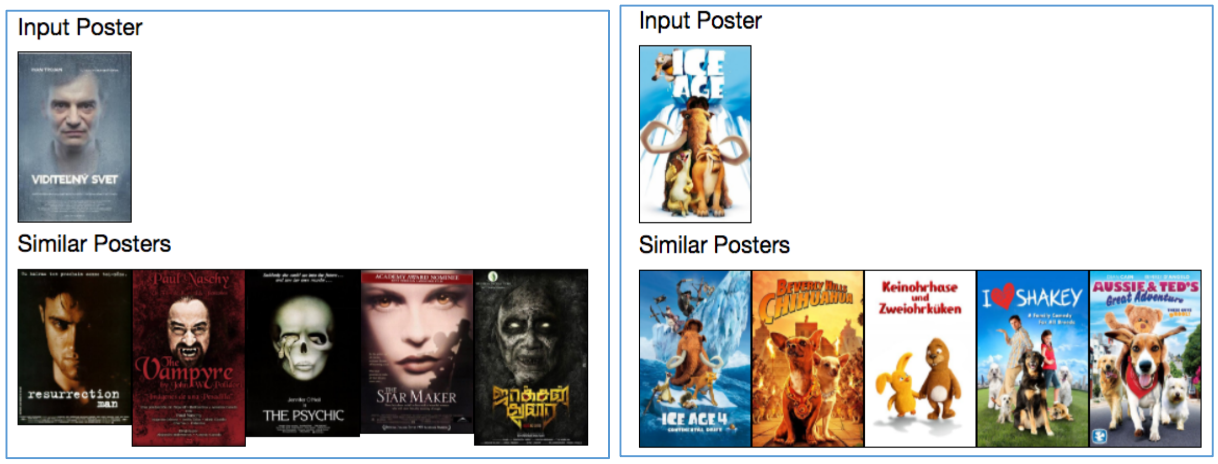

Figure 5: Example of VGG16 Predicting Similar Movie Posters

Citations

1 http://faculty.washington.edu/baldasty/JAN13.htm
2 http://www.imdb.com
3 https://www.themoviedb.org/?language=en
4 Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).