CS109B/STAT121B Project Final Report

Emre Erdem, Gerard Nignon, Xian Pu, Hanan Zalkinder

# Introduction and Data Exploration

The aim of this project was to predict movies genres from a number of attributes of movies. The data we used were randomly extracted from IMDB and TMDB movie databases through APIs. We merged the two data sets using IMDB IDs and obtained 4444 movies including the following features, classified in the following groups:

* Meta features: budget, original language, popularity, production company, production country, release data, revenue, runtime, spoken language, vote average, vote count,
* Director, actors, writer
* Text features: overview, title, tagline
* Image features: poster images (pixels)

In the beginning, we computed the correlation between the genres as well as part of meta features and the results showed that there is very little correlation between genres. Also even though some metadata features might be useful in our prediction such as production company, production country and voting average, overall there is weak correlation with genre. Therefore our prediction accuracy would be low if we only use metadata from TMDB and IMDB. In addition to the metadata that have correlation over 3% with genre, we incorporated image recognition of posters and natural language processing of plot summaries / actors, writer and directors to help us improve our accuracy.

## Challenges

#### Initial showed that we will encounter the following challenges for predicting genre:

#### **Multiple dependent variables:** The genre which we are trying to predict, often has multiple values. We chose a genre randomly in order to deal with a multi-class problem instead of a multi-label one.

#### **Large number of genres:** There were many different genres and some had very small sample. In order to have enough data for each genre, we eliminated movies with genres which had less than 50 observations and ended up with 4250 movies.

#### **Imbalanced sample: S**ome genres were much more frequent than others, this made the sample unbalanced. In order to deal with this problem, we incorporated balancing weights to our models.

#### **Low correlation of metadata with genre:** Correlation analysis showed that there are not many variables currently in the database that are correlated with genre. Therefore we decided to use image recognition of posters as well as natural language processing of plot summaries to predict genre.

# Prediction using Traditional Machine Learning Methods

In this part, we predicted movie genre using traditional machine learning methods. We came up with two schemes (see the figure below for illustration the flowchart):

* In Scheme 1 (proposed in milestone 3), we extracted bag-of-word term frequencies from plot summary and movie titles and use principal component analysis (PCA) to reduce the dimensionality. The PC scores were used as new features and combined with the features from meta data (including directors, writers, actors, production company, production country, budget etc).
* In Scheme 2 (proposed in milestone 4), we extracted text features using TF-IDF from movie overview, title and tagline. We prescreened the TF-IDF features using Chi-square score and filtered out those extremely non-discriminable terms.

|  |  |
| --- | --- |
|  |  |

The 4250 movies were split into a training set (70% observations) and a test set (30% observations). Three traditional machine learning methods; logistic regression, linear support vector machine and random forest were applied to train on the features generated in scheme 1 and scheme 2, and the hyper parameters of these models were tuned. The prediction accuracy on test set are shown in the table below.

|  |  |  |
| --- | --- | --- |
| Model | Scheme 1 | Scheme 2 |
| Logistic Regression | 18% | 45.6% |
| Support Vector Machine | 16% | 43.0% |
| Random Forest | 20% | 34.9% |

We saw that logistic regression using features in scheme 2 had the highest prediction accuracy of 45.6%.

# Prediction using Deep Learning

## a. Deep Learning model from Scratch

In this part, we trained a convolutional neural network (CNN) model from scratch. We trained a baseline CNN model from scratch and extended it to fit from augmented data. The baseline CNN model included 5 layers. Our baseline network structure can be summarized as follows:

==> First Layer: Convolutional input layer, 32 feature maps with a size of 3×3, a rectifier activation function and a weight constraint of max norm set to 3. Max Pool layer with size 2×2.

==> Second Layer: Convolutional layer, 32 feature maps with a size of 3×3, a rectifier activation function and a weight constraint of max norm set to 3. Max Pool layer with size 2×2.

==> Thrid Layer: Convolutional layer, 64 feature maps with a size of 3×3, a rectifier activation function and a weight constraint of max norm set to 3. Max Pool layer with size 2×2.

==> Fourth Layer: Convolutional layer, 64 feature maps with a size of 3×3, a rectifier activation function and a weight constraint of max norm set to 3. Max Pool layer with size 2×2.

==> Convolutional layer: Flatten layer. Fully connected layer with 64 units and a rectifier activation function. Dropout set to 50%. Fully connected output layer with 14 units and a softmax activation function.

A categorical crossentropy loss function was used with the stochastic gradient descent optimization algorithm configured with a large momentum and weight decay start with a learning rate of 0.01. Since we are dealing with a classification problem, our metric was accuracy. We also added a couple of extra features to our training:

* Learning rate scheduler : Decaying learning rate over the epochs usually helps model learn better.
* Early stopping : Number of epochs with no improvement after which training will be stopped.
* Model checkpoint : We will save the model with best validation accuracy. This is useful because our network might start overfitting after a certain number of epochs, but we want the best model.

The early stopping criterion and learning rate are tuned to optimize the performance and to control overfitting. The prediction accuracy is 31.5%. Moreover, we augmented data by generating new posters for training from the existing posters, by slightly:

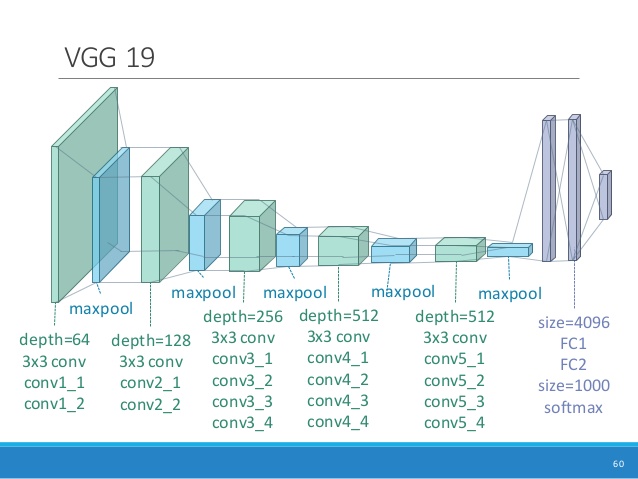
* Translating of poster image
* Rotating of poster image
* Shearing the poster image
* Zooming in/out of the poster image

We refit the baseline model on the augmented data, but the prediction accuracy was 27.8%, lower than the baseline model on the original poster data.

## b. Prediction with Pre-Trained CNN Model

To improve the performance of CNN on poster image data, we chose to proceed with VGG19 pre-trained model. The model has 19 layers and achieved very high accuracy in ImageNet 2014 Challenge. VGG19 model achieves this performance through increasing depth using an architecture with very small (3x3) convolution filters and pushing the depth to 19 weight layers. You can find the architecture of the model below from Very Deep Convolutional Networks for Large-Scale Image Recognition paper by K. Simonyan and A. Zisserman. We used this model as base to predict movie genres since the pre-trained weights can help us recognize images with high accuracy.

On top of the VGG19 model, we added a trainable max-pooling layer to down-sample and further reduce the dimensionality of the image before flattening. After flattening images to two dimensions, 2 fully connected layers with size 512 and 64 is used to gradually decrease number of channels which perform classification on the features extracted by the convolutional layers. Finally, the last layer is used predict across 14 genres using softmax as we have a multi-class problem. The structure of VGG 19 is illustrated below.



*VGG19 Architecture: During training, the input to our ConvNets is a fixed-size 224 × 224 RGB image. The only preprocessing we do is subtracting the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations we also utilize 1 × 1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 × 2 pixel window, with stride 2. A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000- way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity. (Source: Very Deep Convolutional Networks for Large-Scale Image Recognition paper by K. Simonyan and A. Zisserman)*



After modification and hyper-parameter tuning of the VGG19 CNN pre-trained model, our insight was that pre-trained and fine-tuned VGG19 model with top-layers performed better than the fine-tuned CNN model from scratch (37% accuracy vs. 32%). The most important factor in increasing VGG19 model accuracy seemed to be adding top layers to the pre-trained model. Also fine-tuning learning rate played a significant role to boost accuracy performance.

In order to improve our model for final submission, we decided to create an ensemble of models by combining the traditional machine learning model in scheme 2 and the modified VGG19 models which had 46% and 37% accuracy respectively.

# Model Ensemble

In this step, we combined the logistic regression model mining text features and meta data, we well as the CNN model mining poster images. A large amount of experiments have shown that, effective ensemble of models with mutual information can improve the prediction performance. We used a voting-on-probability approach for model ensemble. Specifically, let us consider a movie. For each class (genre), all three models output probabilities of the movie belong to this class and we compute a linear combination of these three probabilities. Repeat this step for every class and we get the strengths (linear combinations of probabilities) for every class, and we classify the movie to the class with maximal strength. The structure of model ensemble is shown below and the final prediction accuracy on test data reaches 47.2%.



# Conclusion and Discussion

In conclusion, we built a number of models to predict movie genres using different types of movie features, and the final ensemble model reached 47.2% in accuracy, significantly higher than a naïve random model. There are still many ways to improve the work in the future. For example, we can apply the word embedding technology for text analysis and we can apply temporal models such as hidden Markov model, conditional random field and recurrent neural network to mine the contextual information from movie overview and comments. We may also improve the model by integrating the information from users who watch/vote movies on IMDB and TMDB websites.