Movie Popularity Prediction

Ieremias Viorel

June 2, 2018

Contents

1 Overview

1.1 Proposed problem

The entertainment industry, and the movie industry consequently, are dynamic fields, that are strongly influenced by social trends of the times. Movie producers have to a appeal to an audience that continuously evolves, and constantly changes its habits, preferences and stlye, so it is harder for everybody to estimate whether the product they are working on will be a big success or a soon-to-be-forgotten creation.
Thus, the need for an application that can accuratelly give a prediction of the popularity of a movie, based on a quantity of information about a movie, taking into consideration revires of other movies is obvious.

1.2 Specification

The application considers important features of a movie, like the most important actors, the director, the genre and the year of lauch and provides an estimation of the popularity of movie as a mark, ranged between 0 and 10.

1.3 Tools

The tool to implement the application is **scikit-learn**, an open-source Python-based solution for machine learning problems, started in 2007 and regularly maintained and updated.

1.3.1 Installation requirements

Scikit-learn requires:

- Python (at least 2.7 or 3.3)
- NumPy (at least 1.8.2)
- SciPy (at least 0.13.3)

1.3.2 Installation instructions

After meeting the above requirements, the easiest way to install scikit-learn is using pip:

```
pip install -U scikit-learn
```

or conda:

```
conda install scikit-learn
```

1.3.3 Running examples

The examples available on the **scikit-learn** website in the examples section (examples) can be downloaded as a Python file that can be run from the command line using:

```
python file_name
```

1.3.4 ExistingExample

- Diabetes Diagnosis - link

- Polynomial Interpolation - link

2 Theoretical aspects

2.1 Data representation

There are many features that can be taken in consideration when estimating the popularity of a movie, but a sensible selection, in an approximate order of their importance, is:

- most relevant 4-5 actors
- director(s)
- writer(s)
- genre(s)
- year of launch
- duration
- awards and prizes of the actors

Train data for the algorithm comes from movie datasets provided by:

- IMDb
- kaggle
- MovieLens

2.2 Algorithm

2.2.1 Support vector machine

Support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification or regression, SVMs can efficiently perform a non-linear classification/regression using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

SVMs are provide an espcially efficient tool when there is a large number of features, out of which none are singnificantly more important then the others due to the way they place examples in the multidimensional space equal in number of dimensions to the number of features. Then, the target of a new example is calculated with regard to the distance to the examples in the training set.

2.2.2 Linear regression

Linear regression is one of the simplest and most basic tools in statistics and machine learning. It is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

The advantage of using linear regression is that in the case of movie popularity too, it is expected that there is a linear relationship between some of the features at least (like presence of some actors, year of launch, etc.) and the popularity of the movie.

3 Implementation

After considering the approaches of the two algorithms and the advantages and disadvantages that each of them provide, I decided to work with an SVM model, and apply regression to predict the popularity of a movie as a real number of a scale of one to ten, taking into consideration the features previously mentioned:

- most relevant 4-5 actors
- director(s)
- writer(s)
- genre(s)
- year of launch
- duration

3.1 Data Preprocessing

There are a multitude of datasets about or related to movies, three sources being presented in the Data Representation section. After analyzing the structure and the features of the mentioned dataset, the one from IMDb was chosen, because it has clear identification rules for movies and actors, is well structured, provides a large number of examples, from all genres and types of video productions, and besides data about movies, it also contains some basic information about actors, directors and other cinematography figures.

Data is accessible through a public API, as a set of .csv files, separated on the scope, ranging from basic information about movies and actors, to ratings from users and alternative titles.

The data filtering steps consisted in:

- selecting only the movies, ignoring the short movies, the TV shows, the documentaries and other media productions
- selecting only movies produced after 1998, as an interval of 20 years is considered relevant and sufficient for making inferences about the movies that appear now or in the near future

- eliminating movies with a duration of less than 60 minutes or more than 210 minutes; most commercial movies qualify in the range with a large margin and the outliers are not relevant in this case
- considering a movie only it has at least 1000 ratings; movies with less ratings tend not to be representative for the general viewer and often higher ratings than expected

After applying the filters, the dataset consists of approx. complete 11.000 examples of movies produced between 1998 and 2018, with a duration between one and three and a half hours, and which have been rated by at least 1000 users.

The structure of an example is the following:

```
Submarine 2010 97 Comedy,Drama [1064292, 1020089, 175916, 852965] [1547964]
7.3 76008
```

which contains the name, the year of launch, the genres, a list of actors id's, a list of producers id's and finally, the average rating and the number of toal user ratings.

3.2 Feature selection

For this version, the most common and simplest features were selected:

- most relevant 4-5 actors
- director(s)
- writer(s)
- genre(s)
- year of launch
- duration

As further improvement, a set of calculated features, including the awards of the leading actors and of the director, up to the moment of the production of the movie and amount of resources spent in advertising the movie will be taken in consideration in a future version.

3.3 Feature scaling and normalization

Both the actors, directors and genres were encoded using one-hot encoding, result for each example an array of about 30000 bits, were 1 on a certain position signifies the presence of the actor, director or genre with the specific id in the movie, while 0 signifies the opposite. Since the duration and the year are both positive integers with a value significantly larger than 1, normalization and scaling was applied to both of them, as initial test have shown that without it, the algorithm doesn't even converge.

3.4 Parameter fitting

SVM only have 2 parameters to set:

- **kernel type** - sets the metric by which the distance between 2 examples is calculated; options available in scikit-learn are linear, polynomial and rbf (radial basis function)
- **C** - a regularization parameter that sets the tradeoff between fitting the examples really well and obtain a large margin hyperplane. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job at fitting the training points correctly. Conversely, a very small value of C will cause the optimizer to introduce more regularization, fitting less well the training points, but generalizing well to new points that were not part of the trainng set

After performing 700 iterations with each of the kernels, the results indicate the linear kernel as the best option, which is as expected, since the linear kernel is best suited for datasets where the number of features is comparable to the number of examples.

```
kernel: linear
train error:  0.53
test error:  0.79

kernel: polynomial
train error:  0.98
test error:  0.97

kernel: rbf
train error:  0.62
test error:  0.74
```

Regarding the C parameter, tests were performed in the range of 10 to 2000, again with 700 iterations of the algorithm, and the best value was found to be 30.

```
value: 10
test error:  0.76
value: 30
test error:  0.75
value: 60
test error:  0.77
value: 100
test error:  0.79
value: 300
test error:  0.87
value: 600
test error:  2.03
value: 1000
test error:  2.03
value: 1500
```

```
test error:  2.03
value: 2000
test error:  2.03
```

4 Results

Using the linear kernel and 30 as the value for C, the model was trained with 7000 iterations on the dataset of 10578 examples, distributed as 8000 training examples and 2578 test examples. The errors obtained are as follows:

```
train error:  0.22
test error:  0.76
```

In previous smaller scale test, on the same dataset and with the same parameters, but after only 4000 iterations, it was observed that the train error suffered a significant improvement (decreased with 0.5), but the test set error suffered little to no change, so it can be induced that a larger number of iterations causes the model to overfit the train examples, but doesn't improve at all the test set performance.