Predicting Popularity of Pornography Videos

Estefania Ortiz (CS221), Jessie Duan (CS229)

I. Introduction

Popularity of online pornography videos is often seemingly random. In this project, we examine a dataset of pornography videos, and attempt to predict the success of a video based on its metadata, including title, description, categories, and uploader. Popularity is defined as a combination of views, comments, and votes, with views most heavily weighted.

Determining the expected popularity of a video has applications in both industry and research. Professionals in the pornography industry will better understand how to produce a video (ex. length), present it (by adjusting title, description, and selecting uploader), and analyze the success of a video. In addition, researchers in a variety of fields including Feminist Studies, Sexuality Studies, Sociology, Anthropology, and other fields that study pornography will have a tool to track trends in the industry, identify words, categories, and other factors that increase a video's popularity; and draw insights from the data.

In this project, we will explore algorithms to predict a pornography video's success; specifically, unsupervised learning (using K-means), and linear regression.

II. Problem Definition

We are examining an exhaustive dataset of metadata for all 811,975 videos published on the xHamster pornography website from its creation in 2007 until February 2013. The data was acquired from http://sexualitics.org/, and provides the following metadata for each video is:

id, upload date, title, channels (i.e. categories), description, number of views, number of votes, number of comments, runtime, uploader ID

We choose to use k-means clustering and linear regression, as two very different algorithms, to explore this data.

I. Assumptions

We must keep in mind the assumptions that we are making about the data, namely that:

1. Viewers are randomly presented with videos. In other words, videos have an equal opportunity to be chosen for viewing. This is the most important assumption, as it ensures randomness. In practice, this is most likely not true, as the xHamster site has a "Promoted videos" section; however, some element of randomness is provided by the "Recently uploaded" category on the homepage.

2. Viewers' choice of videos to watch is mostly based on the available metadata. The ideal predictor would take in every factor of a viewer's choice; however, we can only work with the data that we have.

3. Viewers are able to see all video metadata when making a decision on what to view. In other words, viewers can see runtime, uploader, title, description, etc.; and/or be able to filter by category, uploader, or other features. If one element of video metadata is not visible, it cannot affect the viewer's decision.

II. Data Model

A visual examination of all data shows that number of views follows an approximately exponential distribution, as seen in Figure 1.

Because it is hard to see the specific shape, we transform the number of views into a log scale to get an approximate bell curve in Figure 2.

Figure 1: *Distribution of all views*

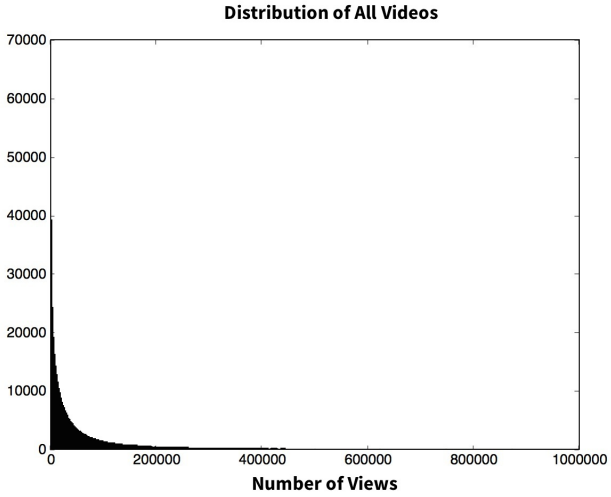
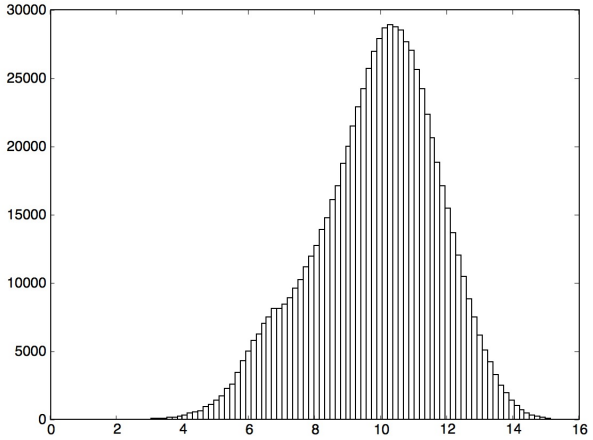


Figure 2: *Distribution of views on a log scale*



III. Training

To analyze our algorithms, we split the data into a train and test set, we randomly assign each video to either "test" or "train" with a 5% probability of being "test". This gives us 746,975 train videos and 39,146 test videos. We then train each of our algorithms on the train set, and evaluate on the test set.

IV. Scoring

Our predicted result uses number of views to approximate popularity. We attempted three different scoring methods:

1. The first measures the percentage of videos where our predicted result is within N percent of the actual number of views. In other words, given m test videos,

$$S_1 = \frac{1}{m}\sum_{i=1}^{m} 1\{\frac{|\hat{y^{(i)}} - y_{actual}^{(i)}|}{y_{actual}^{(i)}} <= N\}$$

We chose to examine $N = 10\%$ and $N = 25\%$.

2. Mean sum of squared errors:

$$S_2 = \frac{|\hat{y^{(i)}} - y^{(i)}_{actual}|^2}{m}$$

This scoring mechanism is only valuable for comparison between methods, rather than serving as an absolute measure of error.

3. K-means bucketing:

$$S_3 = \frac{1}{m}\sum_{i=1}^{m} 1\{s^{(j)}\{\sum_{j=1}^{K} min(|s^{(j)} - y^{(i)}_{actual}|)\} == y^{(i)}\}$$

This mechanism was used to get an intuitive sense of the effectiveness of k-means. The "buckets" in this case represent the number of views assigned to each centroid. The score is the percentage of test videos that were given the value that minimizes the difference between the assigned value and the actual value. Let s be the solution set of the K buckets defined by the training step.

III. Methods

I. Baseline

As a baseline, we calculate the mean of all views in the training set and assign this mean to every video in our test set. In other words, for all test videos i,

$$\hat{y^{(i)}} = \frac{1}{n}\sum_{j=1}^{n} y^{(j)}$$

This mean turns out to be 75,521 views.

II. Unsupervised learning - K-means

The training set will be clustered into a number of groups based on a distance function that determines the similarities between the videos.

i. Feature Vector

We attempted to use the following features; those with the asterisk (*) are the ones that were ultimately used:

```
individual words in the title in the format of 'title_{word}'
individual words in the description in the format of 'description_{word}'
individual categories in the format of 'category_{category_name}'
individual words in the title
individual words in the description
*categories
*tokenized title in the format of 'tokenized_title_{token_id}'
tokenized title in the format of 'tokenized_description_{token_id}'
uploader id
log(runtime)
*runtime
upload date
```

where tokenid is the index of stemmed non-stop-words in either the title or description.

Many combinations of these features and their respective weights were tested with parameters were $K = 15$, training size = 1000, stopping point when 90% of cluster are not changing by more than 20%, and the threshold for changing

a video from one cluster to another is 10% improvement in distance. The values were selected based on previous exploration. More details about these values are below.

The smallest combination of features yielding the best result is: categories, tokenized title words, and runtime. This is to be expected given that the tokenized title takes out the most important words from the title which is clearly visible unlike description or uploader, the categories inform decisions based on preference and filtering ability, and runtime contributes to what the viewer expects from the video. Thus, we chose to use these three features only, and weighted title tokens, categories, and runtime with 10, 1000, and 1 respectively. These weights were arrived at through experimentation as well.

ii. Distance Function

The distance function consists of taking the squared distance between the values of the features present in either of the two videos compared. The distance if only one of the videos contains the feature is the feature's value squared.

Different distance functions were explored before arriving at this one. We tried quantifying and weighting similarities between videos and maximizing rather than minimizing the distance in different ways. These attempts yield similar but inferior results as the method explained above.

iii. Initialization Strategies

The initial k centroids are set to be random videos from the training set. These training videos are used again for the algorithm.
Another exploration was setting the initial centroid values to videos in succession, where each newly selected video has at least twice as many views as the previous one, in order to guarantee a variety of values. The results were very similar for both strategies.

iv. Centroids

The centroids are calculated by combining all of the videos in the cluster. Text based features are accumulated, date and runtime are averaged, and the cluster average is inherited and only updated after assignment.

v. Updating Cluster Assignment

In order to minimize the algorithm time, and avoid bouncing equilibriums the centroid to which a video belongs to is updated if the difference between the distance to the closest cluster and the current cluster is greater than 10% of the smaller distance. The algorithm stops when at least 90% of the clusters have reached an equilibrium, and this is determined by the fact that after updating the centroids and allocating the new videos the last average and new average are no more than 20% of the old average apart. More about this below.

vi. Prediction

After training, the prediction for each cluster is given by either the mean or median number of views across all training videos in that cluster. The number of views for a test video is then predicted by assigning the video to one of k clusters, and selecting the prediction for that cluster.

vii. Hyper-parameters

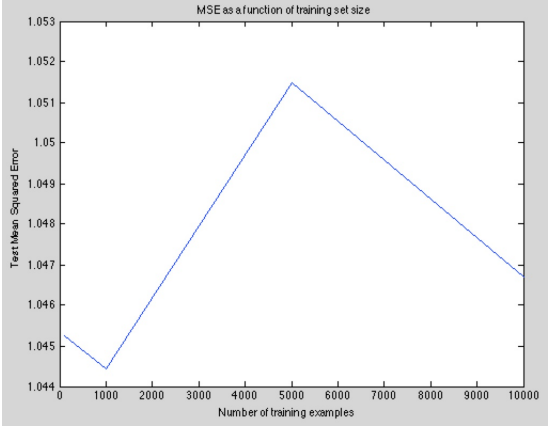
As shown in Figure 3, increasing the training set size above 1000 did not significantly improve performance.

Based on measured runtime of the algorithm in comparison with the results achieved, the values determining the stopping point of the learning part of the algorithm were determined. There are 3 values contributing to this point:

A. How many clusters have been updated in the last iteration? If the number is less than 10% of the total number of clusters K, there are no more iterations.

B. How do we decide when to update a cluster? Once assignment has been made, the average of all the videos in the cluster is calculated, if the average diverges by more than 20% of the old value from the old value, the cluster is updated to the new average.

Figure 3: *K-means MSE using optimal hyper parameters, features, and* $K = 20$

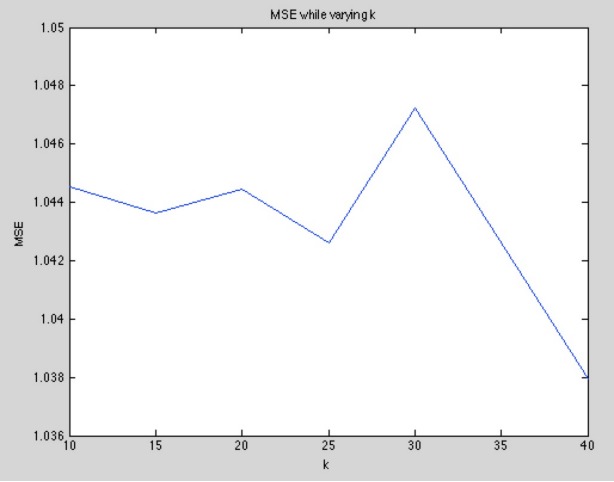


C. When do we change a video from one cluster to another? If the cluster that minimizes the distance and the current cluster for the video have a distance difference of more than 10% the video is changed to the cluster that minimizes the distance.

viii. Choice of k

To determine the optimal value of k, we ran the k-means algorithm with several different values of k and observed mean squared error. Although $k = 40$ provided the lowest mean squared error, we chose to use $k = 15$ as a local minimum that still ran in a reasonable amount of time.

Figure 4: *Mean squared error as a function of k*



III. Supervised learning - Linear Regression

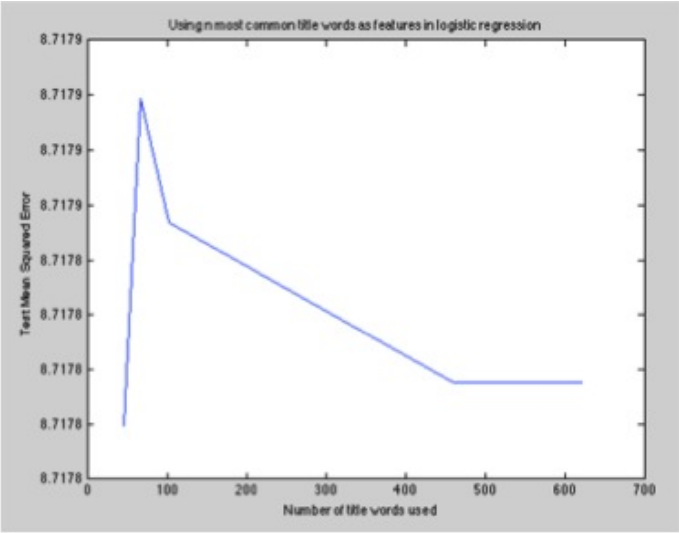
Linear regression used a subset of the available features:

```
Runtime
Days uploaded before Feb 28, 2013 (the last date of a video upload)
Categories, as a series of vector entries marked 1 if the video was in the given category and 0 otherwise
Title
```

To incorporate the title, we took all words in training video titles, removed stop words, stemmed them using the Python nltk library, and counted their frequency. Because the range of frequencies was from 1 to 11,000 (out of almost

750,000 training videos), we chose to only examine the n most frequent stems. For each video, we then created a vector representing the title as a series of binary entries indicating presence of each of the top n title stems. To select the optimal n, minimizing noise while providing enough data, we ran linear regression using several values of n, ultimately finding that $n = 46$ provided the lowest squared error.

Figure 5: *Test squared error as a function of number of title stems used in linear regression*



Finally, the value that we predicted was the natural log of the number of views, using the Python sklearn package. Experimentation with predicting the log number of views, versus the number of views, showed slight improvement with the natural log.

IV. Further exploration - Human Prediction

As part of this exploration into prediction, we wanted to compare our performance against an optimum solution. We chose to use the average human's prediction as this comparison, and used Amazon Mechanical Turk. We had humans look at the available features (title, description, categories, runtime, and date uploaded), and predict the number of views to be within logarithmically increasing buckets:

```
0 - 9 views
10 - 99 views
100 - 999 views
1000 - 9999 views
10000 - 99999 views
100000 - 999999 views
1000000+ views
```

We then evaluated what percentage of these were correctly classified, with the intent of gathering an intuitive sense for how well humans could perform.

IV. Results

Method	Mean Squared Error	Correct w/ 10% margin	Correct w/ 25% margin	K-Means Bucket
Baseline	2.79e10	3.85%	9.66%	–
K-Means (w/ mean)	2.71e10	3.91%	9.85%	0.0778
K-Means (w/ median)	2.79e10	6.00%	14.0%	0.24
Linear Regression	2.23e10	4.35%	10.8%	–

Contrary to what we expected, qualitative observation of the clusters did not seem to yield any strong correlation with category, title, or any other feature.

Furthermore, coefficients from linear regression were determined below, implying that runtime and categories had the largest impact on prediction of views.

Feature	Coefficient
Days since upload	7.96e1
Runtime	4.77e4
Categories (sum)	1.33e6
Title stems	Varies; order of e2

Finally, our human prediction method showed that humans predicted the correct view bucket 21.7% of the time.

V. Analysis

Clearly, both K-means and linear regression outperformed the baseline. Interestingly, K-means with median had a similar MSE to the baseline but better correctness scores, whereas K-means with mean had a better MSE but similar correctness to the baseline. The median seems to perform better from this correctness and bucketing standpoint, which makes sense because of our data distribution (with few videos having very high number of views, skewing the mean).

Although linear regression had the lowest MSE, its correctness was between that of the two K-means methods. This is most likely because linear regression is based on minimizing squared error. However, it is particularly interesting that the features identified as being most influential by linear regression are the same as those for K-means, simply weighted differently. These different weights may be because of the way the two different methods calculate "distance". The most important features themselves, though, were not surprising since non stop title words, categories, and runtime inform the user about what to expect from the video, are more visible than description, and are less arbitrary than upload date.

One of the biggest challenges with K-means was optimizing many variables at the same time. Looking for the optimal size of training set, k, and hyper parameters to optimize for both speed and performance was very complicated. Finding out that increasing the training set size was not as effective in terms of improving performance as we would have expected made the process easier. At first it was surprising, but given the features we are using it makes sense that the training set size is not that influential after a certain point. We only have 92 categories, categories are one of the biggest predictors of popularity, and a lot of words used in the video titles are redundant.

The challenge with linear regression was extracting the most important features. Ideally, every possible token in the title and description would be included in the regression; however, the resulting vectors had over 800k entries and were not possible to fit. Choosing the top n title word stems seemed to be effective because most title word stems were used less than 100 times across the training dataset of 750k videos, and indeed, the noise from using too many title words is visible.

Our results seemed fairly poor at first. However, human prediction showed that even humans don't perform very well in predicting video popularity. Knowing that, it is not surprising that our algorithms performed poorly - ultimately, there was limited information and we had made strong assumptions about randomness.

Overall, it seems that K-means with median is most successful at predicting an almost-correct number of views for a specific video, while linear regression is most successful at minimizing error across the entire dataset.

VI. Further Work

Further work could involve additional adjustment to the k-means objective function, perhaps deriving more features.

We could also consider shaping the data by detecting and discarding anomalies. However, because the data does fit a rough bell curve, it is not clear how far from the median a data point would have to be to be considered an anomaly.

Another avenue to explore is dimension reductionality, such as with PCA. Although we attempted to reduce the number of dimensions for linear regression, we were unsuccessful in using only those features for the regression; reductionality would help identify the most important features as well.

However, the largest improvement will most likely come from obtaining visual information that users see when browsing a pornography website - specifically, thumbnails.