Deep Learning for Multi-label Classification[a]

Swagarika Giri (1711CS16),[1, b] Sunny Yadav (1711CS15),[1, c] and Abhijeet Kharat (1711CS02)[d]
IIT Patna

(Dated: 6 April 2018)

In multi-label classication, the main focus has been to develop ways of learning the underlying dependencies between labels, and to take advantage of this at classication time. Developing better feature-space representations has been pre-dominantly employed to reduce complexity, e.g., by eliminating non-helpful feature attributes from the input space prior to (or during) training. This is an important task, since many multi-label methods typically create many different copies or views of the same input data as they transform it, and considerable memory can be saved by taking advantage of redundancy. In this paper, we show that a proper development of the feature space can make labels less interdependent and easier to model and predict at inference time. For this task we use a deep learning approach. We present a deep network that, in an empirical evaluation, outperforms a number of competitive methods from the literature.

Keywords: Suggested keywords

Assigning a movie genre is kind of hectic process, in the sense that first people have to watch movie, then their feedback is collected and then sent to website which classifies the genre. IMDB is kind of public database which provides genre information of movie besides plot, cast and rating. It is not necessary that genre classified based on people reviews are always correct. There must be something automated system to do this impartially.

An Automatic genre classification based on plot of movie will bring revolution in genre classification by providing suggestion and the outcome may be potentially be more accurate than those humans conscience. Now the task of watching movie, asking for opinion and the allotting genre is reduced to just running an algorithm and genre will be classified within moment of time.

I. CREATION OF DATASET

We used web scrapping tool named beautifulSoup to get data from IMDB website. Our data includes imdb_id, movie name, genre, and plot of movie. Increasing imdb_id value by unit, we get url for next movie and so details and using a loop we get data set of 45572 movies stored in csv file. Our file contain set of 45572 movie details and 47 unique genre information of movies only. For this project, the entire file is randomly split into 80% and 20% sets for training and testing respectively. Some of movies has more than one genres.

In our project there are 47 genres in which 20 genres have frequency more than 1000. We have taken only those genres which has occurred in more than 1000 movies and as a result we got 20 genres only. The genre labels of movies in them are:fiction, fiction, romance, animation, music, comedy, war, horror, western, thriller, adventure, mystery, science, foreign, drama, action, crime, documentary, history, family, fantasy respectively.

A. PREPROCESSING

The preprocessing starts with removing stopwords like articles, helping verbs and all other unnecessary word which bring out no meaning in the synopsis. This is done using NLTK package. All the numerical words were also mapped to the same index as they usually do not provide much genre related information. We further proceed to do stemming. In stemming process, words with the same base but different forms mapped to the same index. Python provide very powerful library to do stemming using Python PorterStemmer.

We have only considered those movies data which have two genres only. Hence we end up having 10893 movies. A dictionary with 25372 words is generated. Out of all words in dictionary generated, only those words which has occurred more than 10 times in all training samples were used in Bag of Words representation. So only 4732 words were left ultimately. We now try to find term frequency and inverse document frequency (tf-idf) value of words using the formula:

$$\text{tf-idf}_{Wk,di} = W_k i \times \log \frac{m}{\sum_{d=1}^{m} \{W_{kd} > 0\}}$$

where Wki is the frequncy of the k-th word in the i-th movie's synopsis (di), and m is the number of training/test set.

After the preprocessing is done, each movie plots is represented by its tf-idf vector x belong to 4732. Now this tf-idf value will be input to our model. The vector is also normalized to make it a unit vector. This was done to account for the variation in length between the

[a]Dr. Arijit Mondal Asst. Professor Ph.D.
[b]CSE Department, IIT Patna.
[c]Electronic mail: sunny.mtcs17@iitp.ac.in
[d]https://github.com/SwagarikaGiri/Multi-label-classification

plots of different movies. Similar treatment of tf-idf and normalization were also done for the test set using the same 4732 words. There is also a bit vector y, associated with each movie whether y = {0, 1} to indicate whether it belong to genre r. So we have an output vector of size 20 which has {0,1} that indicates whether a particular movie belongs to that genre or not.

B. CLASSIFICATION METHOD

There can be more than one genre associated with any movie. So the task in this project is a multi-label classification problem, in our case we have considered only two labels. We evaluate one approach and will try to solve our problem using Neural Network.

C. NEURAL NETWORK

A neural network uses error back propagation algorithm. In this there is an input layer and output layer. Amid these two layer we have hidden layers and the number of hidden layers can be more than one depends upon the complexity of the problem. But as the number of layers increases the cost of our network also increases. So there is trade off between cost and layers. Each hidden layer can have many neurons depends upon complexity and again trade off exists.

1. Tensorflow

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such i.e neural networks in our case. It is used for both research and production at Google.

2. Keras

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or MXNet. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) and its primary author and maintainer is François Chollet, a Google engineer.

$In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library.$

II. METHODOLOGY

In our project we are using predefined function to get neural network provided by keras.

A. Step 1

We have used pandas dataframe for importing the file as dataframe provide flexible reading of datset from csv or text file.

B. Step 2

Create a 2D list of input data and label data from the csv file. Here in our case we have used dataset.iloc[ ].values to convert the string data of the csv file into float value of the list.

C. Step 3

We have used ScikitLearn's "train_test_split" function to divide our data, in our case, we have used 80:20 ratio where 80% data is for training and 20% data is for testing.

D. Step 4

We have used StandardScaler for fitting and transforming the training data. We have standardize our scaling so we will use the same fitting method to transform / scale the test data.

E. Step 5

Finally we start the building of the Neural Network. The modules that we need to include our Sequential module for initializing the Neural Network and Dense module to add the hidden layers.

Initializing Neural Network

$$Classifier = sequential()$$

F. Step 6

Adding multiple hidden layer will take a bit effort. We will add hidden layers one by one using Dense function.

1. Output Dimension

It is simply the number of node we want to add in this layer. In our case we have used 50 neurons in the first layer and 150 neurons in the second layer.

2. Kernel initializer

At the time of initialization, weights should be closed to 0 and will randomly initialize weights using Uniform function. In our case it is uniform.

3. Activation

There are various activation function as relu, sigmoid, softmax. Softmax and sigmoid is popularly used for multi label classification. In case of softmax we assign label to top n vectors where n is number of genre for that particular movie. In our case we have used sigmoid where we have assigned label 1 to top two values.

4. Input dimension

As the actual size of Bag of Words was 4732, which is a very large value, hence we have used Principal Component Analysis (PCA) with a variance of 95%.

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of distinct principal components is equal to the smaller of the number of original variables or the number of observations minus one. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

Because of that, the size of the input is reduced to 3146. Hence the input dimension is 3146.

G. Step 7

Till now we have added multiple layer to our classifier. Now we will compile it using compile method. The first argument is optimizer. There are different type of optimizer like SGD, Adam, in our case we have used adam. Adam is an update to the RMSProp optimizer. In this optimization algorithm, running averages of both the gradients and the second moments of the gradients are used. The second argument is loss function. Since our dependent variable is binary, we have used logarithmic loss function called binary_crossentropy. If our dependent variable has more than two categories in the output then use categorial_crossentropy. The third argument is metric. Since we wanted to improve the performance of our neural network based on accuracy, so we have added accuracy as argument of metrics.

H. Step 8

We have now trained our model with training data. We used fit method to fit our model in previous step, a batch size of 10 is used i.e. we update the weight after every ten iterations and epoch of 100 is used i.e. the total number of iterations.

I. Step 9

We have used classifier.predict() to predict the result on testing data.

III. RESULT

The evalution is based on following matrix (precision recall F1 score —-) Which is based on TP , TN, FP, FN.

A. Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \frac{TP}{TP + FP}$$

B. Recall

Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

$$Precision = \frac{TP}{TP + FN}$$

C. F1 Score

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

$$F1Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

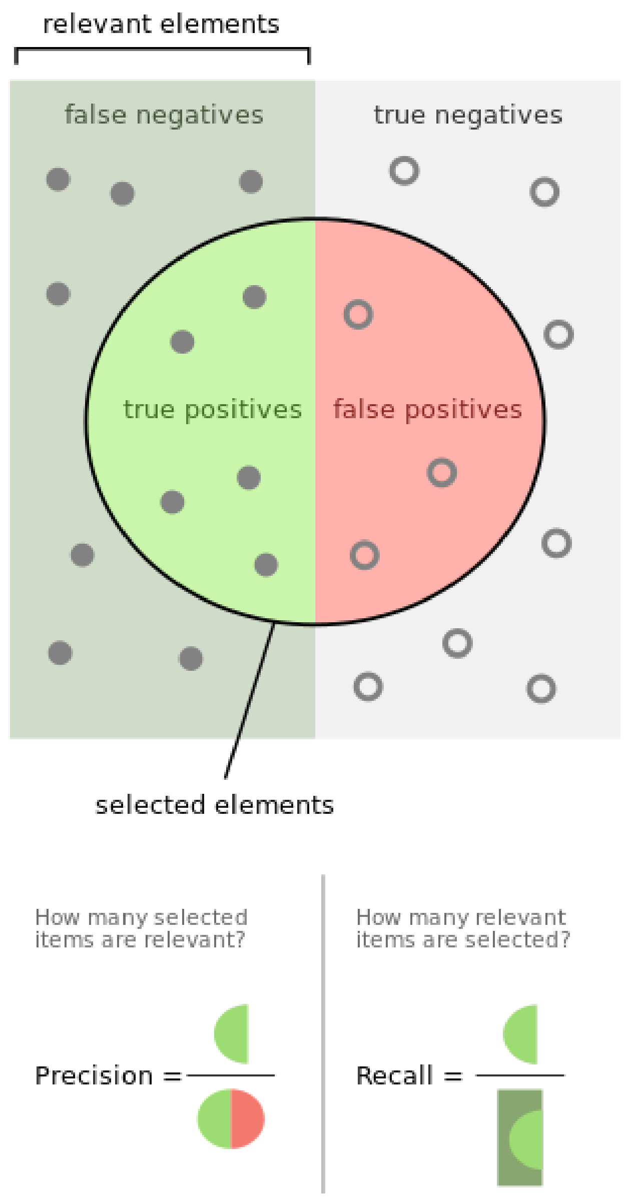

FIG. 1. Matrices

Unnormalised data							
Genre	TP	TN	FP	FN	Precision	Recall	F1-Score
Fiction	7	2142	6	24	0.54	0.23	0.32
Romance	250	1445	251	233	0.5	0.52	0.51
Animation	20	2032	68	59	0.23	0.25	0.24
Music	46	2014	57	62	0.45	0.43	0.44
Comedy	357	1148	348	326	0.51	0.52	0.51
War	22	2065	45	47	0.33	0.32	0.32
Horror	101	1859	111	108	0.48	0.48	0.48
Western	20	2087	37	35	0.35	0.36	0.36
Thriller	149	1630	176	224	0.46	0.4	0.43
Adventure	19	2036	52	72	0.27	0.21	0.23
Mystery	14	2035	69	61	0.17	0.19	0.18
Science	9	2142	6	22	0.6	0.29	0.39
Foreign	10	2005	75	89	0.12	0.1	0.11
Drama	766	634	417	362	0.65	0.68	0.66
Action	82	1816	137	144	0.37	0.36	0.37
Crime	61	1858	139	121	0.3	0.34	0.32
Documentary	73	1950	83	73	0.47	0.5	0.48
History	19	2027	50	83	0.28	0.19	0.22
Family	39	1960	88	92	0.31	0.3	0.36
Fantasy	10	2053	69	47	0.13	0.18	0.15

Normalised data							
Genre	TP	TN	FP	FN	Precision	Recall	F1-Score
Fiction	11	2142	6	20	0.64	0.35	0.44
Romance	334	1445	200	233	0.62	0.62	0.62
Animation	57	2032	50	40	0.53	0.58	0.55
Music	46	2014	57	62	0.45	0.43	0.44
Comedy	357	1148	348	326	0.51	0.52	0.51
War	34	2065	40	40	0.45	0.45	0.45
Horror	101	1859	111	108	0.48	0.48	0.48
Western	35	2087	30	27	0.53	0.56	0.54
Thriller	149	1630	176	224	0.46	0.4	0.43
Adventure	19	2036	52	72	0.27	0.21	0.23
Mystery	29	2035	60	55	0.32	0.34	0.34
Science	21	2142	6	10	0.77	0.67	0.71
Foreign	10	2005	75	89	0.12	0.1	0.11
Drama	766	634	417	362	0.65	0.68	0.66
Action	82	1816	137	144	0.37	0.36	0.37
Crime	91	1858	120	110	0.43	0.45	0.44
Documentary	73	1950	83	73	0.47	0.5	0.48
History	52	2027	50	50	0.5	0.5	0.5
Family	59	1960	80	80	0.38	0.38	0.38
Fantasy	36	2053	50	40	0.41	0.47	0.43

Unnormalised data		Normalised data				
Genre	Prec	Re	F1.	Prec	Re	F1
Fiction	0.54	0.23	0.32	**0.64**	**0.35**	**0.44**
Romance	0.5	0.52	0.51	**0.62**	**0.62**	**0.62**
Animation	0.23	0.25	0.24	**0.53**	**0.58**	**0.55**
Music	0.45	0.43	0.44	0.45	0.43	0.44
Comedy	0.51	0.52	0.51	0.51	0.52	0.51
War	0.33	0.32	0.32	**0.45**	**0.45**	**0.45**
Horror	0.48	0.48	0.48	0.48	0.48	0.48
Western	0.35	0.36	0.36	**0.53**	**0.56**	**0.54**
Thriller	0.46	0.4	0.43	0.46	0.4	0.43
Adventure	0.27	0.21	0.23	0.27	0.21	0.23
Mystery	0.17	0.19	0.18	**0.32**	**0.34**	**0.34**
Science	0.6	0.29	0.39	**0.77**	**0.67**	**0.71**
Foreign	0.12	0.1	0.11	0.12	0.1	0.11
Drama	0.65	0.68	0.66	0.65	0.68	0.66
Action	0.37	0.36	0.37	0.37	0.36	0.37
Crime	0.3	0.34	0.32	**0.43**	**0.45**	**0.44**
Documentary	0.47	0.5	0.48	0.47	0.5	0.48
History	0.28	0.19	0.22	**0.5**	**0.5**	**0.5**
Family	0.31	0.3	0.36	**0.38**	**0.38**	**0.38**
Fantasy	0.13	0.18	0.15	**0.41**	**0.47**	**0.43**

IV. REFERENCES

[1] Ka-Wing Ho Movies' Genres Classification by Synopsis http://cs229.stanford.edu/proj2011/Ho-MoviesGenresClassificationBySynopsis.pdf

[2] Pushkar Mandot Build your First Deep Learning Neural Network Model using Keras in Python https://medium.com/@pushkarmandot/build-your-first-deep-learning-neural-network-model-using-keras-in-python-a90b5864116d

[3] Jesse Read, Fernando Perez-Cruz Deep Learning for Multi-label Classification https://arxiv.org/pdf/1502.05988.pdf