Multilabel Extreme Classification

Andrew Yeh ay1626 * **Alec Hon abh466** * **Eelis Virtanen ev933** * **Mrinal Jain mj2377[Submitter]** *

Abstract

This project explores the use of different models in performing classification for an extreme amount of classes. We begin our study implementing various models such as Linear Support Vector Machines and Random Forests in a binary relevance (One Versus Rest) setting, and then explore the use of neural networks and finally ensemble methods. Our final model is a polynomial ensemble with validation LRAP score of 0.6450 and test LRAP Score of 0.6528. Link to code repository

1. Introduction

It is estimated that 40 million lawsuits are filed every year just in the United States, leading to 40 million potentially useful cases and 40 million precedents related to a large variety of topics.[1]

Therefore, it would be very desirable to have a machine learning algorithm label these legal documents for us based on the content. Rather than mutually exclusive classes (multi-class problems), this is a multi-label problem: legal documents can fall into many ancillary categories, e.g. "civil" and "small claims" and "property damage," all at once. [2]

We develop and compare several algorithms to handle multi-label classification. Our strongest individual models are the Random Forest binary relevance classifier as well as a regularized Neural Network with corresponding label ranking average precision (LRAP) scores of approximately 0.56 and 0.63, respectively. Our final predictive model is based on a polynomial ensemble of all of our models, including information gain from some of the weaker individual classifiers.

In Section 2, we formally define the problem including the specific dataset inputs and label outputs. We then discuss our approach to this problem as well as the final prediction model. In Section 3, we discuss the design of our experiments including data cleaning, splits, evaluation metrics, and our model selection process. In Section 4, we discuss baseline model results and the results of the approach discussed in detail in Section 2. In Section 5, we conclude our findings and discuss future work.

[1] https://www.onelegal.com/blog/top-court-filing-statistics-from-around-the-country/

[2] https://www.courts.ca.gov/1000.htm?rdeLocaleAttr=en

2. Methodology

2.1. Problem Statement

The problem that we address in this paper is extreme multi-label classification. In a multi-class setting, an observation can belong to, at max, one class which gives rise to One vs One and One vs Rest classifier solutions. However, in multi-label classification, each observation can belong to any combination of multiple classes at the same time–this necessitates a different modelling perspective. The objective is, given a feature matrix, to predict all possible classes that a test data point belongs to.

2.2. Chosen Algorithm

Our final algorithm is a polynomial ensemble of the KNN, SVM, Random Forest, and Neural Network models. We find through stepwise addition of features that each individual model provides value to the final ensemble. The KNN model requires no training time and simply finds the 3 closest neighbors using a distance metric and averages their labels. A SVM attempts to draw a separating hyperplane between various classes. A Random Forest ensembles multiple decision trees in order to have a low bias, high variance estimator. The SVM and Random Forest are run according to binary relevance with a separate classifier for each label. The Random Forest has its bootstrap sample weights adjusted according to class frequency to deal with class imbalance. Finally, a neural network with a single hidden layer is trained, and techniques like dropout and batch normalization are used to reduce over-fitting. An ensemble is created by finding an optimal 2-degree polynomial combination of models based on the validation set: KNN, SVM, Random Forest, and Neural Network with interaction terms created between Neural Networks and the other models. Specific implementations of each individual model can be found in Section 4.

3. Experimental Design

3.1. Datasets and Cleaning Process

The training set is legal text documents that had been pre-processed into a sparse dictionary such that only the features and labels present in each document are included. There are a total of 5,000 features and 3,993 labels to classify into. There were an average of 5.32 tags per document in the training data with a standard deviation of 1.35, suggesting a fairly narrow spread. However, since tags per document ranged from 1 to 24, we concluded it was not optimal to predict a fixed number of tags per document.

The main hurdles posed by the data were high dimensionality, sparsity of features, and severe class imbalance. High dimensionality and sparsity of features meant extended training times and restriction on what models we could use to train. For example, we found that boosting algorithms had infeasible train times. There was also severe class imbalance for each label. On average, each class in the training data had 20.67 positively labelled instances out of 15,511 total observations. We dealt with this problem on a model-by-model basis: in binary relevance models, we up-weighted the samples with positive labels in order to penalize misclassification of them by more.

To process the data, we changed the sparse format of the features and labels into two dense matrices. For the feature matrix, we filled up the features that an example does not contain with 0's. The label matrix had 0's when the label is not present and a value of 1 when it is present. Moreover, for neural networks to converge faster and to a better minima, we applied a log transformation, followed by standardization to the features. The data was given to us in the form of a training set, validation set, and testing set, and we applied the same transformations on all three datasets to create feature and label matrices.

We also experimented with Principal Components Analysis (PCA) on the data in order to reduce the dimensionality of the problem and allow us to train more complex models. However, we did not proceed with this transformation step in our final model for two reasons. Firstly, we decided that the standardization of the data required to perform a PCA would cause us to lose a meaningful part of the data: since a TF-IDF process was followed to generate each feature, they are on the same scale, and so standardization would cause us to lose information about the variance of each feature. Second, we validated this hypothesis by trying to train different components of our final model using PCA and got significantly lower evaluation metrics.

3.2. Evaluation Metrics

Our evaluation metric to compare model performance is primarily label ranking average precision (LRAP). LRAP evaluates to between 0 and 1 where 0 is the worst possible LRAP score and 1 is the best. The formula for LRAP is given below:

$$LRAP(y, \hat{f}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \frac{1}{||y_i||_0} \sum_{j:y_{ij}=1} \frac{|\mathcal{L}_{ij}|}{\text{rank}_{ij}}$$

where $\mathcal{L}_{ij}$ determines whether the predicted label is in the true set and orders the set, $rank_{ij}$ is the rank of the label in the predicted set, $|\cdot|$ is the number of elements in the set, and $||\cdot||_0$ is the standard l_0 norm, or the number of non-zero elements in the set.

Essentially, for a given observation, LRAP follows this process for every observation-label whose true value is 1 for that label. It does not perform this calculation for observation-labels whose true value is 0:

1. For true positive x, subset to all labels for which the predicted probability is at least as high (including x).

2. Within that subset, calculate how many samples had a ground truth label of 1.

3. Divide the number of samples from Step 2 by the number of samples from Step 1. Since the labels from Step 1 have at least as high a predicted ranking as the true positive x, we are implicitly predicting all of those labels to also be 1. In this step, we calculate how many of these predicted 1s are actually 1s–an implicit precision metric.

4. Finally, we average the pseudo-precision metrics calculated in Step 3 over all users to obtain the overall LRAP score.

3.3. Model Selection

We approached our classification task by first attempting a few simple baseline models such as Linear Support Vector Machines (SVM) and K-Nearest Neighbor (K-NN) to see the overall ability to classify with a simpler model. We trained our initial models using the default hyper-parameter settings and then compared the LRAP on the validation set.

Based on the outcome of our baseline experiments, we determined that a Random Forest or Neural Network would be more suited for a multi-labelling problem. We saw significantly higher LRAP scores from these models, especially the neural network after tuning. Afterwards, we discovered that ensembling our advanced models with our baseline models helped improve our final LRAP score. Our baseline and final model had Validation LRAP scores listed in Table 1 and Test LRAP scores listed in Table 2.

Table 1. LRAP Scores on the Validation Set

MODEL	LRAP
(NAIVELY) PREDICTING THE TOP 10 LABELS	0.0147
K-NEAREST NEIGHBORS (JACCARD OVERLAP)	0.2913
BINARY RELEVANCE LINEAR SVM	0.3313
RANDOM FOREST	0.5639
NEURAL NETWORK (NN)	0.6301
ENSEMBLE (NN + RF + SVM + KNN)	0.6410
POLYNOMIAL ENSEMBLE (NN + RF + SVM + KNN)	**0.6450**

Table 2. LRAP Scores on the Test Set

MODEL	LRAP
NEURAL NETWORK (NN)	0.6283
NN ENSEMBLE	0.6414
ENSEMBLE (NN + SVM + KNN)	0.6468
ENSEMBLE (NN + SVM + KNN + RF)	0.6475
POLYNOMIAL ENSEMBLE (NN + SVM + KNN + RF)	**0.6528**

4. Results

4.1. K-NN

Our first approach was to use KNN in order find the nearest neighbours for each sample and make a prediction based upon them for unlabelled data. Jaccard overlap performs the best with a LRAP of 0.2913 compared to Euclidean Distance and Manhattan Distance.

4.2. Binary Relevance (OvR) SVM

The second approach was to create a separate binary classifier for each label individually using linear SVMs. In this approach, 3993 different binary classifiers were created for each label. This approach had high computational cost but achieved an improvement over the KNN with a LRAP of 0.3313.

4.3. Binary Relevance Random Forest Classifier

We trained one random forest for each label on all of the features for a total of 3,993 random forest classifiers. In order to counter class-imbalance, we adjusted the weights of observations to be inversely proportional to its class frequency, i.e. we increased the weights of positive labels since they were comparatively rarer. The Random Forest classifier had an LRAP of 0.5639.

Random Forests prove to be a strong classifier. However, there are a few weaknesses as well. Since we must train a new random forest on each label, this process is computationally expensive. In comparison, a baseline neural network is an order of magnitude faster to train. Random forests are also more difficult to tune, and the various hyperparameter configurations we tried did not improve LRAP scores.

4.4. Neural Network

Given that we're dealing with data in an extremely high-dimensional space, exploring how neural networks will perform in such a setting is a natural choice. We experiment with various network architectures, and reach the conclusion that the best performing network is generally a single-layer network, with potentially a large number of hidden units.

However, neural networks on their own are highly prone to overfit. We use two techniques to reduce over-fitting that help the model generalize well:

- Dropout[3]: At a high level, dropout basically means to set the output of a specific unit in the network to 0, with probability p. For example, choosing $p = 0.5$ would "drop" 50% of randomly chosen units while training, during each epoch. This means that the network becomes less complex, and therefore is a means of regularization. An interesting reason behind the success of dropout is the fact that it effectively makes "bagging" possible for neural networks. Randomly choosing a set of units during training can be thought of as training infinitely many neural networks, all sharing their parameters. Traditional bagging techniques are not practical for neural nets due to their complex nature, but dropout is a good approximation of the same. We use a dropout of 0.8 (randomly dropping 80% of the units from the hidden layer), in combination with large number of units in the hidden layer.

- Batch Normalization[4] [5]: Traditionally, batch normalization reduces the "covariate-shift" and reduces the variance of inputs to the hidden layers. However, batch normalization also has a regularization effect, and it makes the loss function smooth, leading to faster training times and better convergence.

We train 3 single-layer neural networks, with dropout and batchnorm, having 2048, 4096, and 6144 units in the hidden layer respectively. Due to heavy regularization, all achieved similar LRAP score, but with certain differences in the predicted values. We later took an ensemble of the predictions made by these networks, and combined them with other model types (random forest, SVM and K-NN).

[3] https://www.cs.toronto.edu/ hinton/absps/JMLRdropout.pdf
[4] https://arxiv.org/abs/1502.03167
[5] https://arxiv.org/abs/2003.00152

4.5. Linear Ensemble

The neural network performed the best out of all the individual models. However, all of the previous models can be ensembled in order to combine the distinct characteristics of each classifier into one superior classifier. The predictions of each classifier were weighted via a convex combination. Naturally, the stronger the classifier, the more weight was generally given. However, some strong classifiers such as the random forest were given less weight than might be expected since their errors seemed to correlate with the errors of neural networks such that ensembling them did not offer as strong a benefit compared to other classifiers such as KNN which had errors uncorrelated with neural network errors. The best combination achieves a LRAP of 0.6410 which is 0.01 greater than the LRAP for the neural network individually.

4.6. Polynomial Ensemble

In the section above, we take a linear ensemble of our models, capturing how different models predictions may complement each other. We hypothesized that there may be non-linear relationships and interactions between the model predictions. In linear regressions, in order to capture non-linear relationships and interactions between the covariates, you can include polynomial and interaction terms in your model. This inspired us to try the same with our ensemble method. We did not find any example in literature that has tried this method before.

We decide to include interaction terms between the neural network and all of the other models. Similarly, we include squared terms for the neural network and the random forest predictions. In order to search for the best weights for the polynomial ensemble, random grid search is used which was guided by our priors on what the weights should be. The neural network weight is searched for in a higher weight range (0.4-0.8) since it is the best individual model whereas all of the other individual models, interaction terms, and squared terms are constrained to a lower range (0-0.4) in our initial search. The best weights are found based on LRAP, and a new more narrow search is performed more closely around the found weights. The best weights found from the second search are used as the final polynomial ensemble weights. Further tuning is not attempted to avoid over-fitting on the validation set.

5. Discussion

5.1. Evaluation of Findings

One of our key findings is that neural networks are more suited to multi-label predictions than binary relevance models with both faster training time and better LRAP scores. This faster training time cannot be solely explained through usage of the GPU via Pytorch. Indeed, even the Scikit-learn implementation of the Multi-layer Perceptron (MLP) is significantly faster. While neural-networks are generally considered slower to train, in extreme multi-label classification, neural networks (when using only a few hidden layers) are in fact faster than the OvR models since only one model is needed as opposed to separate models for each label. This relationship is not exact, and neural networks can be slower with enough hidden units and layers.

Additionally, after building several baseline and more advanced models, we discovered that an ensemble of them did better than each one individually primarily by reducing the variance of the final estimator. The final LRAP we achieved on the test set is 0.6528 which is 0.0227 higher than our best individual model, suggesting that a good ensemble allows the models to slightly correct each other's mistakes. Even weak classifiers such as the SVM and the KNN help achieve minor improvements in LRAP. Similarly, an ensemble of two strong classifiers that make similar errors is often worse than an ensemble of a strong and weak classifier that make very different errors.

5.2. Areas of Future Work

One of the main shortcomings of our current ensemble method is that it takes a significant time to train each component model. One solution could be to parallelize the computation across a cluster. For example, the binary relevance computations which were used for the Random Forest and SVM could be easily parallelized instead of being computed sequentially on a single machine.

Another potential limitation of our approach is that many of our models use the binary relevance perspective. Binary relevance makes no use of correlation between labels since each label is predicted independently, potentially losing valuable predictive information. One solution could be to adjust the binary relevance predictions with the observed correlations between the labels, which is currently being researched on as a method to improve multi-label classification.[6]

[6] https://link.springer.com/chapter/10.1007/978-3-319-13560-1_8