Automatic Feature Learning using Determinantal Point Process

Chapman Siu

***Abstract*—We develop a novel approach to learning optimal feature embeddings using Determinantal Point Process. Rather than having to know all possible feature embeddings during learning time, we instead consider it as a formulation in our learning framework, thereby allowing it to scale in both the size of the feature set and the number of instances. This approach works in an incremental fashion, gradually building up a feature set in a group-wise fashion, relying on regularization methods to determine the ideal group of text embeddings that is to be selected.**

Introduction

In recent years there have been an explosion of feature learning techniques used to generalize models and their representation to multiple contexts[1], e.g., text embedding models [2], [3], image embedding [4]. At the same time, there has been an increasingly popular direction in hyper-parameter optimization for Machine Learning (ML) models.

In this paper we will consider the task of feature learning to determine the optimal feature representation of a target problem. As features get transformed into many different representations, we must be comfortable in both searching over a large (possibly infinite) parameter space of possible representations, and selecting the feature representation which is ideal for the problem at hand.

Existing group feature selection techniques often do not search over the parameter space, and instead assume that they arrive in batches in the online feature selection sense[5]. While existing hyper-parameter optimization frameworks generally aim to determine the optimal machine learning model, rather than the optimal feature representation[6]. To this end, we propose using Determinental Point Process for sampling over a large parameter space, and group Lasso methods to select optimal feature representations of the data in a group-wise nature.

This framework consists of two parts. First, hyper-parameter sampling on infinite parameter scale using DPP. Second, online group feature selection for determining optimal embedding.

This work makes the following contributions.

- Extending online feature selection methods to ensemble like models
- Infinite Hyperparameter optimization using DPP

The paper is organized into the following sections. Section II we will lay the preliminary foundations and review related approaches to the online group feature learning problem. In section III, we will introduce our framework online group feature learning. In section IV we will provide experimental results to demonstrate the effectiveness. We will conclude this work in section V.

Preliminaries and Related Work

In this section we will first review related work for group feature selection and online feature selection. Afterwards we will provide a review of determinantal point process and the hyperparameter sampling problem.

Feature Selection

Traditionally, feature selection has been performed in an offline setting. The feature selection problem can be framed as follows. We are given a matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$ which has n instances and d-dimension feature space $F = [f_1, f_2, \ldots, f_d] \in \mathbb{R}^d$. The goal of feature selection is to selection a subset of the feature space such that $U \in \mathbb{R}^l$ where l is the number of desired features, where in most cases $l < d$[7]. Offline feature selection is a widely studied topic with many different reviews[8]. Rather than provide a comprehensive review, we will instead focus on several selected techniques and their online feature selection counterparts. We will cover feature selection from two perspectives as a filter and wrapper method. From the filter approach, we will consider batch approaches using statistical significance and spectral feature selection as well as their online variants being Online Streaming Feature Selection and Online Group Feature Selection respectively. We will also consider the wrapper methods in the batch setting using regularization and information criterion approaches, as well as the online variants being grafting and alpha-investing respectively.

For completeness, the third approach for feature selection is the embedded method which perform feature selection in the process of training as they are specific to models. Approaches here could include

decision tress such as CART, which have built-in mechanism to perform feature selection[8]. To the best of our knowledge, there are not any embedded methods present from an online feature selection perspective.

Correlation Criteria and OSFS: The first approach uses the filter method, which evaluates features by certain criterion and select features by ranking their evaluation values or by some chosen threshold.

One common approach is to consider the correlation related criteria[8] such as mutual information, maximum margin, or independence criterion. Of particular interest is conditional independence criterion which is constructed through consideration of relevance and redundancy of features in terms of condition independence[9]. In this setting the process of labelling a feature to be relevant or redundant is performed using statistical tests based on conditional independence.

Online Streaming Feature Selection (OSFS) uses this framework of relevance and redundancy to determine whether incoming features are added. When a feature arrives, OSFS first analysis correlation with the label and determines whether the feature is relevant[10]. Once a feature is successfully chosen, then OSFS performs redundancy test to determine if both previous and current features are redundant and can be removed. In this setting the redundancy is a key component of OSFS approach. This approach has been extended to use mutual information with pairwise approximations called Scalable and Accurate On Line Approach (SAOLA)[11].

Regularization Methods: The wrapper method, which uses the machine learning algorithm of interest as a black box to score subsets of features.

Regularization is typically labelled as a wrapper method in the feature selection framework, meaning that it uses a model algorithm to jointly build a model and select features. This is typically employed through both minimizing empirical error and a penalty. In the context of regularization, the goal is to encourage sparsity on the feature subset. Regularizer penalties are typically framed as [12]

$$\Omega_p(\theta) = \lambda \sum_{i=1}^{m} \alpha_{\mathbf{i}} |\theta_{\mathbf{i}}|^p$$

where a choice of $p = 1$ is typically chosen to promote sparsity, commonly referred to as the Lasso penalty.

To alter this framework to an online setting, the grafting algorithm is used. Grafting is performed on any model which can be subjected to Lasso regularizer. The idea behind grafting is to determine whether the addition of a new feature would cause the incoming feature or alternatively, any existing feature to have a non-zero weight. With a chosen parameter λ, the regularizer penalty is then $\lambda|w_j|$. Thus gradient descent will accept a new incoming feature w_j if:

$$\left| \frac{\partial \bar{\mathcal{L}}}{\partial w_j} \right| > \lambda$$

where $\bar{\mathcal{L}}$ is the mean loss. In other words, if the reduction in $\bar{\mathcal{L}}$ outweighs the regularizer penalty $\lambda|w_j|$, then the new incoming feature w_j will be chosen. If this test is not passed, then the feature is discarded. As Grafting makes no assumption on the underlying model, it can be used in both linear and non-linear models.

Alternatively to alter the regularization framework for selecting groups of features. We can consider the construction of disjoint groups of features. This approach is referred to as Group Lasso method and defined as [13]

$$\Omega_p(\theta) = \lambda \sum_{j=1}^{m'} \alpha_{\mathbf{i}} |\theta_{\mathbf{j}}|^p : \theta \in \mathbb{R}^m$$

Where m' represents the index for each group of features, where if each feature was its own group, we would have $m' = m$.

Spectral Feature Selection and OGFS:

Another approach which falls under the filter method for feature selection is the use of the spectral feature selection. In spectral feature selection a graph is constructed. From this graph where the ith vertex corresponds to $\mathbf{x_i} \in X$, with an edge between all vertex pairs. In this graph construct its adjacency matrix W, and degree matrix D. The adjacency matrix is constructed differently depending on the supervised or unsupervised context. In the spectral analysis setting the adjacency matrix can be the similarity metric of choice [14], [15]. For example in the unsupervised context this is can be the RBF kernel function [14], [15], or a weighted sum of correlation metric and rank coefficient metric [16]. Once the appropriate metric is chosen then, a feature ranking function is used for filtering the features. This function can change depending on context, and can be constructed. The choice of this function can be used to determine the statistical significance of each individual feature using trace ratio criterion approach[17].

To extend Spectral feature selection to the online setting, the Online Group Feature Selection (OGFS) has been proposed which considers incoming *groups* of features and applying spectral feature selection on a group-wise level using Group Lasso as defined previously. This is used to determine the relevancy

over the particular group of features which has been shown to extend into the online setting.

Information Criterion and Alpha-investing: Another approach to feature selection in the wrapping sense is the usage of penalized likelihoods. In the context of single pass feature selection techniques, penalized likelihoods are preferred [18]. This set of approaches can be framed as:

$$-2\log(\text{likelihood}) + F$$

where the parameter F indicates how a criterion is to penalize model complexity directly.

The alpha-investing algorithm [18], makes use the information in order to determine whether a new incoming stream of features is considered to be relevant or not. It makes use of the *change* in log-likelihood and is equivalent to a t-statistic, which means a feature is added to the model if its p-value is greater than some α. Alpha-investing works through adaptively controlling the threshold for adding features. This works through increasing the wealth α when a feature is chosen to reduce the change of incorrect inclusion of features. Similarly when a feature is assessed wealth is "spent", which reduces the threshold, in order to avoid adding additional spurious features.

Determinantal Point Process

We begin by reviewing determinantal point processes (DPPs) and conditional DPP.

A point process $\mathcal{P}$ on a discrete set $\mathcal{Y} = \{1, 2, \ldots, N\}$ is a probability measure over all $2^{\mathcal{Y}}$ subsets. $\mathcal{P}$ is a determinantal point process (DPP) if $\boldsymbol{Y}$ range over finite subsets of $\mathcal{Y}$, we have for every $A \subseteq \mathcal{Y}$

$$P(\mathcal{A} \subseteq \boldsymbol{Y}) = \det(\mathbf{K}_{\mathcal{A}})$$

where $K \in \mathbb{R}^{M \times M}$ is a positive semidefinite kernel matrix, where all eigenvalues of K are less than or equal to 1. An alternative construction of DPP is defined by L-ensembles where L_{ij} is a measurement of similarity between elements i and j, then DPP assigns higher probability to subsets that are diverse. The relationship between K and L has been shown to be [19]

$$K = (L + I)^{-1}L$$

Where I is the identity matrix. Then the choice of a specific subset Y is shown to be [19]

$$\mathcal{P}_L(\boldsymbol{Y} = Y) = \frac{\det(L_Y)}{\det(L + I)}$$

In terms of computational complexity, DPP has shown to be of order $O(Nk^2)$[20], though recently proposed MCMC methods have demonstrated to be much faster[21][22].

Determinantal Point Process for Hyperparameter Sampling: In order to sample a large set of hyperparameters, the feature representation and hyperparameter is simply discretized using a one-hot encoding scheme, as such then choices of embedding and model can be treated as hyperparameters [23].

In order to prepare the hyperparameter values to be sampled using DPP, the resulting hyperparameter values need to be encoded in a way which can be processed. Here we present two approaches of how this can be done.

DPP-One-hot: The simplest approach to preprocess the hyperparameter set is to use a one-hot encoding scheme, where every attribute and value is one-hot encoded. This would ensure that every value would be encoded and projected in the range $[0, 1]$. This is synonymous to the hamming variant for categorical variables as described by Hutter[24].

DPP-Scaled: Under the simple one-hot encoding scheme, information around the hyperparameter choice would be lost. For example, when choosing between two different feature embedding sizes for a word2vec model, when the choice of an embedding size of 10 vs 15 or 10 vs 100 would be equally (dis)similar. Here we proposed an additional adjustment based on function $\lambda(n)$, where n is the scaling factor. For a hyperparameter that takes a numerical value in range $[x_{\min}, x_{\max}]$, the hyperparameter value x would be encoded and projected in the range $[0, 1]$:

$$\lambda(n) = \left(\frac{x - x_{\min}}{x_{\max} - x_{\min}}\right)^n$$

The projection to $[0, 1]$ prevents hyperparameters from dominating the similarity calculations, whilst at the same time the parameter n allows one to control the similarity measure. To provide more weight to different range of values, one could increase the weight of n, or decrease the influence, or decrease the weight of n.

Framework

The setup for the proposed framework uses a sequential model-based optimization (SMBO) strategy[25], in which it decides the optimal feature representation to keep. More specifically it uses the Random Online Aggressive Racing (ROAR) variant, except instances of promising configurations are instead sampled using DPP, rather than uniform normal. Like ROAR, this means that the $FitModel$ procedure returns constant model and never used.

Algorithm 1 DPP for Hyperparameter search as SMBO/ROAR

while not terminated **do**
  Intensify using DPP conditioned on previous evaluations to select new set of hyperparameters.
  Evaluate the resulting feature set, discarding any feature representation that are not performant.
end while

In this paper we will examine how to extend this framework to particle filtering and the application to DPP.

Particle Filtering

This problem can also be framed as a particle filtering problem[26].

Algorithm 2 Particle Filtering

Input: $\tilde{S}_t = S_t = \emptyset$, T iterations, M samples per iterations
for $t = 1, ..., T$ **do**
  for $m = 1, ..., M$ **do**
    Sample $x_t^{(m)} \sim P(x_t|u_t, x_{t-1}^{(m)})$
    Assign weights $\tilde{w}_t^{(m)} = P(z_t|x_t^{(m)})$
    $\tilde{S}_t = \tilde{S}_t + \langle x_t^{(m)}, \tilde{w}_t^{(m)} \rangle$
  end for
  for $m = 1, ..., M$ **do**
    Resample by drawing i with probability $\propto w_t^{(i)}$
    add x_t^m to S_t
  end for
end for
Output: S_t

In our formulation at iteration t, u_t refers to the *Intensify* action as described by SMBO framework, and the signal z_t, being the metric of choice for our machine learning algorithm to maximize. The usage of DPP can be applied in the *intensify* action to provide suitable diverse samples to be shown and perturbed. This can be used in conjunction with Bayesian Optimization approaches such as Random Forest Regressors to ensure suitable samples are produced.

Experiments

Here, we run several experiments, involving text and image data to demonstrate ability to learn appropriate representations. We compare our model-free approach with ROAR which is also a model free approach to model based optimization.

Classification Pipeline

This can be applied to determine the optimal classification pipeline in a similar way as TPOT[27]. In this setting there are two stages; preprocessing, postprocessing. Optionally we may choose to optimize pipelines-of-pipelines, making using of stacked generalization[28].

Preprocessing: Preprocessing stage consists of one of the following operations:

- Identity
- Polynomial Features

Optionally, we will perform univariate feature selection techniques including:

- Variance filter
- Mutual Information Score
- F score

Postprocessing: Post processing stage consists of one of the following operations:

- PCA
- ICA
- SVD

Optionally, we will perform univariate feature selection techniques including:

- Variance filter
- Mutual Information Score
- F score

Stacking: In contrast to the approach taken by [27], where pipelines are combined into one single modeling dataset, we instead combine outputs through the use of stacked generalization[28]. This allows us to reuse the outputs of previously evaluated pipelines, without recomputing the results all over again.

Text Representation

To determine the best embedding, we used benchmark datasets being the 20 newsgroup dataset to determine the best text embedding. We will compare our variant using DPP and against ROAR[25]. For either model optimization problem, text embedding can be composed of three stages; preprocessing, model choice, postprocessing.

Preprocessing: Preprocessing stage consists of one of following operations:

- Identity
- Stemming
- Lemmatization

and in additional, option to remove predefined stop words. This leads to a total number of possible combinations to be 6.

Model Choice: Regardless of the choices made in the preprocessing stage, there are several models which are available at our disposal from the scikit-learn library and gensim [29][30]. The following models are used in this experiment

- Bag of words/Word count - one hot encoded transformations
- tfidf/tfidf variants - including options to filter based on tfidf scores, or n-grams.
- LDA/HDP - hyperparameters are the number of topics to be modelled
- Word2Vec - hyperparameters are the embedding size, and also whether skip-gram or continuous bag-of-words models are used.
- Doc2Vec - hyperparameters are the embedding size, and also whether skip-gram or continuous bag-of-words models are used.
- LSI - hyperparameters are based on embedding size.

In addition, each model may have hyperparameters specific to each model which would also have to be turned. Most importantly is the embedding size that is used for the various models.

Postprocessing: In the postprocessing step, we can apply further transformations to control embedding size (especially in situations where no embedding is provided like in BOW models). Here the following operations can be performed:

- PCA
- SVD
- Identity

Through these options, assuming that each option only has one choice, there are 6 choices in preprocessing, 6 in model choice, 3 in postprocessing, which leads to just over 100 possible options. As we consider hyperparameters, the number of possible combinations to evaluate will easily exceed 10^5. If deeper neural network representations are used this could easily increase to a much larger space[31].

With this setup we can define the $Intensify$ component of SMBO framework to choose new hyperparameter instances in a hierarchical manner; first, the best model, then preprocessing and finally postprocessing, based on the assumption that the class of models would be related to each other and more influential.

Results: To prepare a fair comparison with DPP approach and ROAR, we will use to the same set of code with a termination criteria based on system time.

Conclusion

In this paper, we have presented a new algorithm for group feature learning which allows us to efficiently search over a large infinite hyperparameter space in order to select the best feature embedding for the ML problem at hand. We have introduced the Group Grafting algorithm and demonstrated the usefulness of this algorithm to generate features which adapt to changing contexts.

References

[1] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks," in *Advances in neural information processing systems 27*, 2014, pp. 3320–3328.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[5] J. Wang, Z.-Q. Zhao, X. Hu, Y.-M. Cheung, M. Wang, and X. Wu, "Online group feature selection." in *IJCAI*, 2013, pp. 1757–1763.

[6] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in neural information processing systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.

[7] J. Wang, M. Wang, P. Li, L. Liu, Z. Zhao, X. Hu, and X. Wu, "Online feature selection with group structure analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 3029–3041, 2015.

[8] I. Guyon, A. Elisseeff, and A. M. De, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[9] D. Koller and M. Sahami, "Toward optimal feature selection," *International Conference on Machine Learning (1996)*, pp. 284–292, 1996.

[10] X. Wu, K. Yu, H. Wang, and W. Ding, "Online streaming feature selection," *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pp. 1159–1166, 2010.

[11] K. Yu, X. Wu, W. Ding, and J. Pei, "Towards scalable and accurate online feature selection for big

data," in *Data mining (icdm), 2014 ieee international conference on*, 2014, pp. 660–669.

[12] S. Perkins and J. Theiler, "Online feature selection using grafting," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, 2003, pp. 592–99.

[13] K. Lounici, M. Pontil, S. van de Geer, and A. B. Tsybakov, "Oracle inequalities and optimal inference under group sparsity," *Ann. Statist.*, vol. 39, no. 4, pp. 2164–2204, Aug. 2011.

[14] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," *Proceedings of the 24th international conference on Machine learning - ICML '07*, pp. 1151–1157, 2007.

[15] J. Wang, M. Wang, P. Li, L. Liu, Z. Zhao, X. Hu, and X. Wu, "Online feature selection with group structure analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 3029–41, Nov. 2015.

[16] G. Roffo, S. Melzi, and M. Cristani, "Infinite feature selection," in *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[17] E. Grave, G. Obozinski, and F. Bach, "Trace lasso: A trace norm regularization for correlated designs," *NIPS*, 2011.

[18] J. Zhou, D. P. Foster, R. a. Stine, and L. H. Ungar, "Streamwise feature selection," *Journal of Machine Learning Research*, vol. 7, pp. 1861–85, 2006.

[19] A. Kulesza and B. Taskar, "Learning determinantal point processes," in *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, 2011.

[20] A. Kulesza and B. Taskar, "k-DPPs: Fixed-size determinantal point processes," in *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[21] C. Li, S. Jegelka, and S. Sra, "Fast dpp sampling for nyström with application to kernel methods," in *Proceedings of the 33rd international conference on international conference on machine learning - volume 48*, 2016, pp. 2061–2070.

[22] N. Anari, S. O. Gharan, and A. Rezaei, "Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes," in *29th annual conference on learning theory*, 2016, vol. 49, pp. 103–115.

[23] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn," 2014.

[24] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration (extended version)," University of British Columbia, Department of Computer Science, TR-2010-10, 2010.

[25] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration." *LION*, vol. 5, pp. 507–523, 2011.

[26] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics (intelligent robotics and autonomous agents)*. The MIT Press, 2005.

[27] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," in *Proceedings of the genetic and evolutionary computation conference 2016*, 2016, pp. 485–492.

[28] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[30] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010, pp. 45–50.

[31] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR 2017*.