Fastfood Elastic Net: Combining Variable Selection with Kernel Expansion Approximations

An essay submitted in partial fulfillment of

the requirements for graduation from the

Honors College at the College of Charleston

with a Bachelor of Science in

Mathematics

Sonia Kopel

May 2017

Advisor: Paul Anderson

Fastfood Elastic Net: Combining Variable Selection with Kernel Expansion Approximations

Sonia Kopel
College of Charleston
Charleston, South Carolina
Email: kopels@g.cofc.edu

Kellan Fluette
College of Charleston
Charleston, South Carolina
Email: fluetteka@g.cofc.edu

Geena Glen
University of Washington
Seattle, Washington
Email: gmglen@gmail.com

Paul Anderson
College of Charleston
Charleston, South Carolina
Email: andersonpe2@cofc.edu

***Abstract*—As the complexity of a prediction problem grows, simple linear approaches are more likely to fail. This has led to the development of algorithms to make complicated, nonlinear problems solvable both quickly and inexpensively. Fastfood is a tool that is used to expand the feature space of a given data set quickly and can be used in conjunction with algorithms to generate reliable predictive models. However, Fastfood's usefulness has not been studied in its relation to feature selection which is useful in solving a wide array of complex real-world problems that spans from cancer prediction to financial analysis. The aim of this research is to extend Fastfood to include variable importance and feature selection by integrating it with various feature selection methods.**

Methods like Elastic net offer feature selection but can only produce linear models. Meanwhile, methods like sequential feature selection partnered with a modeling algorithm are not guaranteed to find optimal feature subsets and may stabilize at only locally efficient solutions. We show that in combining Fastfood with various feature selection methods, it is possible to retain feature selection and the nonlinearity produced by expanding to higher feature space using Fastfood. Models constructed in this manner are relatively quick and inexpensive to compute and are also quite powerful in their ability to make accurate predictions and effectively select features.

***Keywords*—Kernel Methods, Data Mining, Algorithms and Programming Techniques for Big Data Processing**

I. Introduction

The value of effective prediction methods is self-evident: being able to predict future outcomes can be applied to nearly any field. Some of the more common domains that make use of predictive technology include oncology, stock market analysis, and weather prediction. The methods and algorithms that are used to generate these predictive models continue to evolve to handle complicated real-world data sets that have high dimensionality and large sample sizes. As data collection becomes more popular in many professional fields, more emphasis is placed on developing algorithms to make sense of the influx of data.

For very large data sets, many of the machine learning techniques that are commonly used to generate models either fail or are too expensive in either their required storage space or their runtime, rendering them inefficient. More data offers the ability to train better, more realistic models, but the cost of generating these models is often computationally intractable because of the scope of the mathematical operations that a computer must perform during training. As a result, these more sophisticated models often cannot be computed in real-time, rendering them useless in many disciplines. Storage space is another concern. While the computer running the algorithms may be able to hold terabytes of data, it cannot necessarily hold a square matrix of corresponding dimensions. Thus, even in the rare cases when time is plentiful, the required computing resources to generate models from large data sets is unavailable to many researchers.

Linear techniques, such as Elastic net [7], tend not to succumb to these pitfalls because of their relative simplicity. However, complex data sets with hundreds, if not thousands, of features are unlikely to have linear decision boundaries. As a result, the linear models produced by these techniques are quite often unreliable or inaccurate.

One of the simpler methods for finding nonlinear decision boundaries is the kernel trick. The kernel trick transforms the data by implicitly mapping the features into a higher, possibly infinite, dimension, and from there it is often possible to calculate a linear decision boundary [7]. For instance, if a data set is not linearly separable in two dimensions, it can be transformed into a higher dimensional space. Depending on the function used to transform the data, it may be possible to find a linear decision boundary in this new feature space. The more dimensions the data is mapped into, the more likely it will be that a linear decision boundary can be found. The trick to this technique lies in the fact that the mapping function (ϕ) need never be explicitly defined. Rather, the individual points can be transformed by taking their dot product with a known kernel function (k), like the sigmoid function or the radial basis function [3]. The relationship between ϕ and k is as follows:

$$\langle \phi(x), \phi(y) \rangle = k(x, y) \tag{1}$$

$$f(x) = \langle w, \phi(x) \rangle = \sum_{i=1}^{N} a_i k(x_i, x) \tag{2}$$

In this project, we use Fastfood rather than the kernel trick to expand the feature space of a given data set because of its relative speed and storage efficiency. Once the data is transformed, we apply a feature selection method and then generate a model. Finally, using a correlation matrix, the model is scaled back down to the original feature space. The desired effect of this is to produce an effective predictive model that incorporates variable importance.

II. Related Work

A. Random Kitchen Sinks

The kernel trick can easily become intractable to compute as the computation and storage requirements for the kernel matrix are exponentially proportional to the number of samples in the data set [3].

However, the Random Kitchen Sinks (RKS) algorithm chooses to approximate the kernel function more effectively than the kernel trick by randomizing features instead of optimizing them [6]. RKS does this by randomly selecting these features from a known distribution. The authors show that with this method shallow neural nets achieve comparable accuracy to AdaBoost, a popular ensemble method which adjusts weak learners in favor of instances misclassified by previous classifiers [2]. Overall, RKS has comparable predictive ability to the commonly used AdaBoost, but does not require the same level of rigor on the part of the computer during training. However, RKS makes use of dense Gaussian random matrix multiplications which are computationally expensive. Le and Smola mitigate this problem by replacing these multiplications with the multiplication of several diagonal matrices in their algorithm: Fastfood [3].

B. Fastfood

To reduce the already quick run-time complexity of RKS, Fastfood combines Hadamard matrices with Gaussian scaling matrices [3]. The algorithm replaces the Gaussian random matrices in RKS with this combination. Because of the relative ease of these computations in contrast to multiplying Gaussian random matrices, the complexity of the runtime is reduced from $O(nd)$ to $O(n \log d)$ [3]. This allows Fastfood to approximate the kernel feature map quickly. The main takeaway from this is that unlike in the kernel trick where ϕ is never defined, in Fastfood (and RKS) it is approximated.

Another appeal of Fastfood is that it has previously been combined with other machine learning techniques, most notably neural nets. By implementing the algorithm at each layer of the neural net, additional nonlinearity is added to the training of the neural net and the training process is also sped up [1].

C. Elastic Net

Despite the successes of Fastfood in making accurate predictions in loglinear time, it is unable to measure variable importance and perform variable selection because it relies on kernel approximation. The premise of feature selection is to extract a subset of features from the total number of features in a data set [5]. Feature selection is important because it helps to remove useless features which can slow down calculations or lead to overfitting during the modeling process. Many modelling techniques require having more observations than features, which can make feature selection crucial for problems that have a lot of features, but relatively few observations [8].

Elastic net has variable selection built into the modelling process, but is a linear technique characterized by the below equation [8].

$$\hat{\beta} = \arg\min_{\beta}(\|y - \mathbf{X}\beta\|^2 + \lambda_2\|\beta\|^2 + \lambda_1\|\beta\|_1) \quad (3)$$

Elastic net is a combination of lasso and ridge regression [8]. The parameter α ranges from $[0, 1]$ and determines the elastic net penalty. $\alpha = 1$ would give a model equivalent to the one found by ridge regression and $\alpha = 0$ would be equivalent to lasso [8]. Unfortunately, elastic net is limited in its ability to in creating a predictive model for complicated data sets because it is still a linear technique.

Promisingly, Elastic net has been reduced to support vector machines (SVMs) [7]. SVMs are often used in combination with the kernel trick because of the linearity of the models produces with the method. Since elastic net has been shown to be essentially equivalent to SVMs, then the idea of combining it with a comparable feature expansion technique has a solid theoretical basis.

D. Other Feature Selection Algorithms

Sequential feature selection methods are often very basic in theory. They have been known to perform well at the cost of speed [5]. These algorithms work by sequentially changing the features used in modelling until they optimize performance. Often, there is a penalty term that increases with the number of features. This helps protect against overfitting.

Minimal-redundancy-maximal-relevance considers all of the features simultaneously. It attempts to pick out the features that most give the most information about the outcome variable while discarding the features that give similar information about the outcome [5]. When used in combination with other feature selection techniques, it shown to be accurate and fast [5]. This makes techniques like this favorable for computing feature selection in a higher dimensional space.

III. Methods

Despite the successes of Fastfood in making accurate predictions in loglinear time, it is unable to inherently measure variable importance and perform variable selection because it relies on kernel approximation and projection into a new feature space. Our work builds directly upon the Fastfood algorithm. Herein, we describe our research into developing a computationally efficient variable importance measure for Fastfood by leveraging the built-in feature selection of Elastic net [7] and other feature selection methods. Our results indicate that models generated with our improved variation of Fastfood retains the benefits of the Fastfood while also giving insight into variable importance in the original feature space.

Our base method takes the original feature matrix (X) of size $(n \times d)$ and applies Fastfood. This transforms the original features to a new feature matrix (F) of size $(n \times p)$, where $p \geq d$. Typically, p will be much greater than d. In this step, Fastfood approximates ϕ to transform X into the higher dimensional space p.

$$\underset{n\times p}{F} = \underset{n\times d}{X} \times \underset{d\times p}{\phi} \quad (4)$$

A feature selection method is applied to this new feature matrix (F) and is used to generate a model with variable selection (L). Parameters within the feature selection algorithms and modelling algorithms are also optimized throughout the process by running using vectors of possible values and selecting the best performing model.

Model performance measures vary between classification and regression tasks. For classification, we use Area under the ROC Curve as our primary metric. For regression, we look at R^2 values. In both cases, we run a large number of trials so as to estimate means for these metrics as well as standard deviations. To compare means, two tailed t-tests are performed.

To reduce back to the original dimensionality, we calculate a correlation matrix (ρ) between the new features and the original features. Finally, these correlations are aggregated to reduce and relate the Elastic net coefficient vector (L) of size $(1 \times p)$ to the original feature space of size $(1 \times d)$. We call this final model O. The overall process of FFEN is shown in Figure 1.

$$\underset{1\times d}{O} = \underset{1\times p}{L} \times \underset{p\times d}{\rho} \tag{5}$$

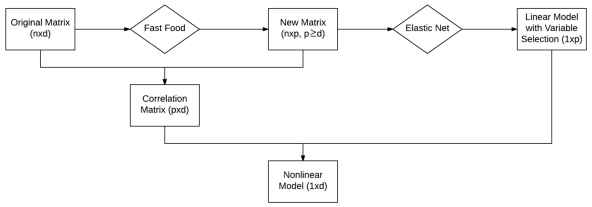

Fig. 1. Process

This method can then be run iteratively to remove extraneous variables or a single run can be used to to integrate variable importance in the original dimensional space with the kernel approximation basis space.

To compare the accuracy of the models and the insight into the original features, we compare our lower dimensional models to those generated by by the feature selection methods without any transformations.

IV. Results

For evaluation, tests have been performed on both simulated linear data sets as well as real data sets from the UCI machine learning repository [4]. The simulated data sets had random variables masked, providing a known gold standard for feature selection and variable importance. Our Fastfood enhanced Elastic net (FFEN) was compared to lasso: a common implementation of the linear technique Elastic net. To keep the comparisons fair, parameters λ and α for lasso were optimized in the same way as they were for FFEN. For the simulated linear data, the true β values for each variable were known, so the primary metric that was used to compare the FFEN model to lasso was average mean absolute error (MAE) for all of the β values. Table 1 depicts the results for simulated data sets of differing dimensions.

TABLE I. Simulated Linear Data

n	d	FFEN MAE	Lasso MAE
400	100	0.2103	0.0401
1000	80	0.1162	0.0301
4000	10	0.0471	0.0300
5000	10	0.0234	0.0400

TABLE II. Empirical Data

Dataset	n	d	FFEN R^2	FFEN σ	lasso R^2	lasso σ
Protein	45729	9	0.3366	0.0268	0.2387	0.0038
KEGG	64608	27	0.8754	0.0120	0.8992	0.0023
Music	515345	90	0.2245		0.2098	

Because this data was known to be linear, the data was not transformed into a higher feature space when using the model produced by FFEN.

The results indicate that FFEN improves as the number of samples increases and the number of dimensions decreases. To justify this, we repeated our simulation 20 times on data sets of varying size (see Table 1). The average MAE for FFEN was 0.03097 as opposed to 0.0473 for lasso. Given the small standard deviations, 0.007572 and 0.00725 respectively, we can with more than 99% certainty conclude that FFEN outperforms lasso on simulated linear data sets with those dimensions. We statistically confirmed this by performing a paired T-Test which gave a p-value of essentially 0.

Given the positive results with simulated, we moved onto the data sets from the UCI machine learning repository [4]. To compare performance on these real-world data sets, we generated 20 different training and test sets for each data set, partitioning all of the data into either a training or a test set each time. We then ran both FFEN and lasso on each set. After this was done 20 times, once for each training/test pair, we compared the R^2 values by running a paired T-Test to test to see which model better explained the data. Table 2 depicts the average R^2 values for the 20 runs. FFEN performed slightly worse than lasso for the KEGG Metabolic Reaction Network Data set, but the performance drop was only by 1-2%. The high value of R^2 for lasso indicates that this is a linear data set, and therefore, FFEN provides marginal benefits. However, FFEN significant outperforms lasso on the Physicochemical Properties of Protein Tertiary Structure Data set (Protein) and also outperformed lasso on the UCI million song data set. Because UCI provided a preferred training and test set partition for the UCI million song data set, only one run was needed to generate the results, so there is no standard deviation to report for this data set. The R^2 values for FFEN and lasso from this run are also depicted in Table 2.

The standard deviations for the FFEN and the lasso for the protein data set are 0.0268 and 0.0038 respectively. The standard deviation of the R^2 values for the KEGG data set were found to be even smaller, only 0.0120 and 0.0023, respectively. The difference in the standard deviations suggests that FFEN

may not be as consistent as methods such as lasso. The most likely explanation for this is the inherent randomness of Fastfood. However, this increase in variability is accounted for in our testing, and the results still indicate that FFEN consistently outperforms lasso.

To directly test against lasso, again, the data was not transformed into the higher feature space during testing. This allows us to directly compare linear approximations and see if we truly have been able to gain insight into the original feature space with our method.

V. Conclusion

In conclusion, FFEN has been shown to offer promising results. The models generated by our algorithm are comparable or better than those of Elastic net on simulated linear data sets when measuring the accuracy of the resulting coefficients. Further, FFEN shows improved accuracy on complex data sets while incorporating novel variable importance not available in standard Fastfood. The use of Fastfood allows us to generate these models both quickly and inexpensively.

Our preliminary results on simulated data indicate that it is possible to retain the accuracy and applicability to nonlinear data of FastFood, while simultaneously providing variable selection with Elastic net. The results we currently have are rather basic, but are very encouraging as they indicate the feasibility of our proposed method.

VI. Future Work

In the coming months, we hope to adjust the algorithm to better predict zeroes as well as test on more empirical data sets like the CIFAR-10 data set as well as several large biological data sets. We also hope to test further permutations of our baseline algorithm and use other feature selection methods. We also plan to use these algorithms for classification tasks.

Acknowledgment

The authors would like to thank the College of Charleston for hosting the NSF Omics REU which is funded by the National Science Foundation DBI Award 1359301 as well as the UCI machine learning repository [4].

References

[1] M. Bowick and W. Neiswanger. Learning fastfood feature transforms for scalable neural networks. *Proceedings of the International conference on...*, pages 1–15, 2013.

[2] Y. Freund and R. R. E. Schapire. Experiments with a New Boosting Algorithm. *International Conference on Machine Learning*, pages 148–156, 1996.

[3] Q. Le, T. Sarlós, and A. J. Smola. Fastfood Approximating Kernel Expansions in Loglinear Time. *International Conference on Machine Learning*, 28(1):1–29, 2013.

[4] M. Lichman. UCI machine learning repository, 2013.

[5] J. Pohjalainen, O. Räsänen, and S. Kadioglu. Feature Selection Methods and Their Combinations in High-Dimensional Classification of Speaker Likability, Intelligibility and Personality Traits. *Computer Speech and Language*, 29:145–171, 2013.

[6] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in Neural Information Processing Systems*, 1(1):1–8, 2009.

[7] Q. Zhou et al. A Reduction of the Elastic Net to Support VectorMachines with an Application to GPU Computing. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 28, 2013.

[8] H. Zou and T. Hastie. A Reduction of the Elastic Net to Support VectorMachines with an Application to GPU Computing. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 67:301–320, 2004.