Online Feature Selection for Dyanmic Models and Dependent Data Streams

Chapman Siu

Contents

Introduction

Every day more and more domains are increasing the breadth and depth of their data every year. There already exists algorithms and platforms which can learn terascale datasets with billions of features and observations (Agarwal et al. 2014). These approaches generally presume fixed set of features which can learn on potentially infinite stream of data using online learning techniques (Zhou et al. 2006). It is important to review the consequences when the restriction of a fixed set of features is removed, which leads online feature selection algorithms.

As companies begin to grow and develop new products such as internet of things devices or mobile applications, streams of data growing both in training examples and features will only continue to increase. In these situations where new features are procured, there becomes the additional constraint where historical data for the new training examples is no longer recoverable. Thus the ability to provide inference on these new products and apps without expensive re-development of machine learning models becomes more important than ever. This report will focus on *online feature selection* in conjunction with *online learning* to handle new incoming streams of data which may be arbitrarily large, both in training examples and features, with a strong bias on approaches which make fewest assumptions on the inputs of our future data.

In this review we will examine firstly, gradient descent algorithms for both feature learning and model learning; then we will provide a broad overview of selected feature selection techniques and their online variants. Finally we will assess related approaches which can handle arbitrarily large training observations and features which are not in a stream-wise fashion.

Gradient Descent Optimization

Gradient descent is an optimization method for unconstrained optimization problems. The algorithm would iterate over the training examples and for each example and update the model parameters according to the update rule given by

$$\theta_{\mathbf{t+1}} \leftarrow \theta_{\mathbf{t}} - \eta \left(\frac{\partial f(\theta_t)}{\partial \theta_{\mathbf{t}}} \right)$$

for some function f where η is the learning rate which controls the step-size in the parameter space.

All approaches considered in this review will be using stochastic gradient descent (SGD) and its variants. The goal of SGD is to approximates the true gradient of the cost function (or minimization function), $f(\theta)$ given above, by considering single training example at a time. In the formulation above, if η is considered to be fixed, then SGD may run into challenges around being too large and never converging to the global minima, or too small and being

"stuck" in a local minima. In recent times there have been attempts to remediate this by extending the learning rate η to be an adaptive function(Karampatziakis & Langford 2011), (Duchi, Hazan & Singer 2011), (Kingma & Ba 2014)

Gradient Descent for Supervised Learning

Gradient descent methods are used in many supervised and unsupervised learning approaches. In this section we will cover several of these approaches and later discuss their relationship with feature selection.

In the context of supervised learning, gradient descent methods can be framed as a set of training instances $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$ where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$. In particular, for linear models, we wish to learn a linear function $f(\mathbf{x}, \theta) = \theta^T \mathbf{x}$ where the model parameter is $\theta \in \mathbb{R}^m$.

This is often framed as a minimization problem, where we aim to minimize the regularized training error by learning the appropriate parameters (Perkins, Lacker & Theiler 2003)

$$C(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f_\theta(\mathbf{x}_i)) + \Omega(\theta)$$

where we have provided the loss function $\mathcal{L}$ and a regularization function Ω, which we will introduce in the next section.

Regularization

The regularization term, $\Omega(\theta)$ is a function of the parameters of the model that returns a high value for "unlikely" instances, or alternatively complex models may generalize badly(Perkins, Lacker & Theiler 2003). By optimizing sum of loss and regularizer, we achieve a trade-off between the two function. The form of the regularizer (or penalty) depends on the form of the model, and there are many choices which can be made. Here we will consider one family of commonly used regularizers(Perkins & Theiler 2003)

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

where the regularization coefficient is λ, p is chosen to be a non-negative real number, and the length of the parameter vector θ is denoted by m. The regularizer family correspond to their respective weighted Minkowski l_p norm raised to the p power.

The three more common regularizers are

- Ω_1 which is the "lasso" penalty
- Ω_2 which is the regularizer commonly seen in ridge regression.
- $\alpha_1\Omega_1 + \alpha_2\Omega_2$ which is elasticnet, typically chosen where $\alpha_1, \alpha_2 > 0$ and $\alpha_1 + \alpha_2 = 1$

When $p < 2$ the regularizer tends to force subset of θ to be exactly zero at the optimum of the cost C(Tibshirani 1996). This is what is required for models to perform feature selection. Based on this knowledge, regularizers which perform feature selection are typically lasso or elasticnet variants.

Generalized Linear Models

SGD methods can be used to learn generalized linear models(Toulis, Rennie & Airoldi 2014). More specifically if we consider any distribution from the exponential family,

$$y|\mathbf{x} \sim \exp\left(\frac{(\theta \cdot \mathbf{x})y - b(\theta \cdot \mathbf{x})}{\phi} + c(y, \phi)\right)$$

Where ϕ is the dispersion parameter, b, c are appropriate real-value functions.

Then based on this formulation, for all models in the generalized linear model setting can update $\theta_{\mathbf{t+1}}$

$$\theta_{\mathbf{t+1}} \leftarrow \theta_{\mathbf{t}} - \eta\left(y - h(\theta_{\mathbf{t}} \cdot \mathbf{x})\right)$$

Where function h is the inverse of the link function function of the GLM, which is $\mathbb{E}(y|\mathbf{x}) = h(\hat{\theta} \cdot \mathbf{x})$.

This class of models is extremely flexible and has been used in distributed machine learning frameworks, and is the backbone of many online feature selection algorithms which we will examine later in this review.

However it is worth noting that SGD for machine learning today go beyond simple linear models, and employ techniques such as gradient boosted machines, or neural networks - both approaches which which remove the assumption of linear relationship between the predictors and response[Hastie2009].

Gradient Boosted Machines

As expressed previously, gradient boosted machines (GBM) allows for non-linear relationships (such as interactions or splines) to exist between predictors and response. GBMs are a class of additive models, where we have parameters θ, and weak learners f_θ. Online variants have been constructed in various libraries including XGBoost (Chen & Guestrin 2016) and SparkML (Zaharia et al. 2016). Formally, let $\hat{y}_i^{(i)}$ be the prediction of the ith instance at the yth iteration, we will need to add $f_\theta^{(t)}$ to minimize the cost function

$$C^{(t)}(\theta) = \sum_{i=1}^{n} \mathcal{L}(y_i, \hat{y}_i{}^{(t-1)} + f_\theta^{(t)}(\mathbf{x_i})) + \Omega(f_\theta^{(t)})$$

This means we can greedily add the $f_\theta^{(t)}$ that most improves our model. Using second-order approximation can quickly optimize the object in the general setting (Chen & Guestrin 2016)

$$C^{(t)}(\theta) \approx \sum_{i=1}^{n} \left(\mathcal{L}(y_i, \hat{y_i}^{(t-1)}) + g_i f_\theta^{(t)}(\mathbf{x_i} + \frac{1}{2} {f_\theta^{(t)}}^2(\mathbf{x_i})) \right) + \Omega(f_\theta^{(t)})$$

where $g_i = \frac{\partial l(y_i, \hat{y_i}^{(t-1)})}{\partial \hat{y}^{(t-1)}}$ and $h_i = \frac{\partial^2 l(y_i, \hat{y_i}^{(t-1)}}{\partial (\hat{y}^{(t-1)})^2}$ are the first and second order gradient statistics on the cost function. This could be further simplified to

$$C^{(t)}(\theta) \approx \sum_{i=1}^{n} \left(g_i f_\theta^{(t)}(\mathbf{x_i}) + \frac{1}{2} h_i {f_\theta^{(t)}}^2(\mathbf{x_i}) \right) + \Omega(f_\theta^{(t)})$$

GBM can yield out-of-core or online learning through approximate algorithm for split finding where the gradients can be approximated at (fixed) percentiles over all features which can be done in an online fashion (Chen & Guestrin 2016).

Due to the nature of the base learning algorithm, GBM will demand that the feature size be fixed upon training. Based on this observation it might appear that boosting is incompatible with learning new streams of features. However it has been demonstrated that this deficit may be overcome in the context of online feature selection when GBM are considered in a more general setting. This idea will be explored more under the *Grafting* algorithm (Perkins, Lacker & Theiler 2003).

Gradient Descent Streamwise Feature Selection

Grafting

One approach to gradient descent for streaming feature selection is the use of grafting(Perkins & Theiler 2003). The idea behind grafting is to use gradient descent in conjunction to regularization, which allows sparse models to be built from a large, finite set of features.

Grafting can work on any models which is subject to l_1 regularization. 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:

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

where $\bar{\mathcal{L}}$ is the mean loss. In other words, if the reduction in $\bar{\mathcal{L}}$ outweights 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. If multiple weights pass the test, then the weight with the highest magnitude is added to the mode.

Connections to Boosting

Grafting can be viewed as a special variant of boosting algorithm where our weak learner f is a linear model, $f(\mathbf{x}) = w_0 + w_1 x_i$, where x_i. Then the weak learner choses the new, incoming feature (which can arrived streamed or otherwise), and w_0, w_1 are learned weights for this model. (Perkins, Lacker & Theiler 2003).

This formulation has many advantages, provided certain assumptions are sensible. Given the nature of the weak learner above, a feature can be grafted if it can be assumed to be independent from the other predictors. Furthermore the weaker learner is of a simplistic form which can allow a large number candidate features be evaluated in an embarassingly parallel fashion. Further work could be undertaken in the future to assess grafting groups of features which can not be assumed to be independent, that is if we learn a model in a form $f(\mathbf{x}) = w_0 + w_i x_i + w_j x_j$ which considers the global redundancy of variables or possibly with the inclusion of interaction terms.

Online Substitution Algorithm

Online Substitution algorithm, is based on grafting with the large difference that it only performs a partial gradient step based on the incoming feature (Yang et al. 2016). The substitution part of the algorithm is based on the notion that the learner can only have at most s features, and will be required to *substitute* an old feature in order to assess the efficacy of a new incoming feature. The procedure is (Yang et al. 2016):

1. Update coefficients of retained features. $\theta_{\mathbf{s}} \leftarrow \theta_{\mathbf{s}} - \frac{\eta}{m} \nabla_s C(\theta)$ and update coefficient of the new feature with index j using $\theta_{\mathbf{j}} = \eta \nabla_j C(\theta)$
2. Retain top k largest magnitude elements of θ and set the rest to 0.

Where $m \geqslant 1$ is a parameter.

The advantages presented in this approach include asynchronous update through the use of partial gradient steps for an incoming feature vector. This presents this as a parallel algorithm which can be scaled across large number of workers which many other gradient descents suffer from due to their additive nature.

One of the challenges for a substitution algorithm is that it demands a pre-known fixed feature set size, and is unable to reassess previous accepted features. In the context of assessing an infinite number features, this may or may not be a reasonable assumption. Also as the algorithm depends on gradient descent, this may suffer from the assumption that data ought to be normalise, otherwise the coefficients of abnormally large coefficients may overpower the regularizer (Shamir 2015).

Learning to Select Features

The previous section addressed when the machine learning model is known to be solvable using gradient descent approaches with the intent of solving online learning and online feature selection. In this section, we will explore approaches beyond gradient descent which are focused solely on feature selection. In comparison with the section above, none of these approaches assume any shape or form related to the underlying machine learning algorithm that has been chosen.

In the framework provided previously, we are given a set of training examples $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)$ where $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$. The goal is to select an optimal feature subset that will provide an improved predictive model, as the best subset problem is known to be NP-complete(Zhou et al. 2006).

This will consider stepwise methods, which will either add features which are deemed helpful (relevancy), and remove features that are not deemed helpful (redundancy). We will divide our feature selection algorithms into usage of regularization, statistical and finally similarity approaches.

Statistical Approaches

In the context of single pass feature selection techniques, penalized likelihoods are preferred (Zhou et al. 2006).

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

where the parameters F indicate how a criterion is to penalize model complexity either directly.

Common criterions which fall under this model include:

- Akaike information criterion, with $F = 2m$
- Bayesian information criterion, with $F = m\log(n)$
- risk inflation criterion, with $F = 2m\log(m)$

Where m is the number of features in the model.A feature is then added/removed, if the *change* in log-likelihood from adding the feature is greater/less than the penalty.

Likelihood Streamwise Feature Selection

The α-investing algorithm (Zhou et al. 2006), 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. α-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.

Algorithm *α-investing (Zhou et al. 2006)*

1. **Input**: A vector of y values (response), a features x (which can be streamed or otherwise), W_0, and α_Δ
2. Initialize:

$$G' = \{\}, \text{ the set of all selected features.}$$
$$i = 1$$
$$w_1 = W_0$$

3. **While** `not_terminate_condition`:
   1. $x_i \leftarrow$ `get_next_feature`.
   2. $\alpha_i \leftarrow \frac{w_i}{2i}$.
   3. **if** pvalue$(x_i, G') < \alpha_i$:

$$G' \leftarrow G' \cup x_i$$
$$w_{i+1} \leftarrow w_i + \alpha_\Delta - \alpha_i$$

   4. else

$$w_{i+1} \leftarrow w_i - \alpha_i$$

   5. $i \leftarrow i + 1$
4. **End**

Using this algorithm allows features to be selected adaptive, thus fulfulling the relevancy portion of feature selection. However this algorithm does not consider the future redundancy of features. That is once a feature is accepted, α-investing has no possible way to subsequently remove that feature if it does turn out to be uninformative.

Although no extensions to the algorithm have been made to consider redundancy, one could naively have a two step procedure; first step to consider relevancy using α-investing, and the second step to consider and remove redundant features using any relevant algorithm.

Similarity Based

The notion behind similarity based feature selection is to remove redundancy of features based on similiarity measures. This can be done in both unsupervised or supervised contexts using

techniques such as Spectral analysis (Zhao & Liu 2007) or energy based approaches(Roffo, Melzi & Cristani 2015).

In both spectral analysis and energy approaches we construct a graph From this graph where the ith vertex correspondes to $\mathbf{x_i} \in X$, with an edge between all vertex paris. In this graph construct its adjacency matrix W, and degree matrix D. The adjancency matrix is constructed differently depending on the supervised or unsupervised context.

In the spectral analysis setting the adjancy matrix can be the similarity metric of choice (Zhao & Liu 2007), (Wang et al. 2015). For example in the unsupervised context this is can be the RBF kernel function (Zhao & Liu 2007), (Wang et al. 2015), or a weighted sum of correlation metric and rank coefficient metric (Roffo, Melzi & Cristani 2015).

$$W(i,j) = \exp(-\frac{||\mathbf{x_i} - \mathbf{x_j}||^2}{2\delta^2})$$

For classification case, it could be constructed as

$$W(i,j) = \begin{cases} \frac{1}{n_l} & y_i = y_j = l \\ 0 & \text{otherwise} \end{cases}$$

where l is the class label and n_l is the number of observations with label being l.

In the regression senario, the weights are based on a chosen kernel k and constructed by k-nearest neighbors (Hou et al. 2011)

$$W(i,j) = \begin{cases} k(x_i, x_j) & y_i \in \text{KNN}(y_j) \\ 0 & \text{otherwise} \end{cases}$$

For some choice of a kernel k and $y_i \in \text{KNN}(y_j)$ indicates y_i is one of the k nearest neighbors of y_j.

We shall define our feature ranking function to be ϕ which can change depending on context, for example if we wish evaluate features via normalized cut, an appropriate ϕ would be (Zhao & Liu 2007)

$$\phi(F_i) = \frac{\mathbf{f_i}^T(D - W)\mathbf{f_i}}{\mathbf{f_i}^T D\mathbf{f_i}}$$

Where F_i is the current feature index being evaluated, $\mathbf{f_i}$ representing a feature vector, and D is the degree matrix, where $D(i,j) = \begin{cases} \sum_{k=1}^{n} w_{ik} & i = j \\ 0 & \text{otherwise} \end{cases}$. In this setting, $\phi(F_i) < \phi(F_j)$ would imply that the ith feature is more relevant than the j feature.

In constrast in the energy setting the ranking function is defined to be a matrix $R = \sum_{l=1}^{\infty} W^l$ where $R(i,j)$ is the energy between two vertices of *all* paths of length $l = 1, 2, ...$ This can be be computed algebraically (Roffo, Melzi & Cristani 2015).

$$R = \sum_{l=1}^{\infty} W^l = (\mathbf{I} - rW)^{-1} - \mathbf{I}$$

Where r is a real-valued regularization factor. In this setting the feature ranking function is defined to be $\phi(F_j) = \sum_{i=1}^{n} R(i,j)$, where more relevant features have higher values (or energies), that is $\phi(F_i) > \phi(F_j)$ would imply the ith feature is more relevant that jth.

Algorithm *Similarity Feature Selection (Zhao & Liu 2007)*
Input: X, W, D

1. **Initialise**

$$G' = \{\}$$

2.For *each feature vector* f_i
   1.1. $G' = G' \cup \phi(F_i)$
2. End
3. Rank G' according to choice of ϕ
4. return G'

In this setting G' would represent an ordered list of features from most relevant to list relevant which can be used for feature selection.

Similarity based Streamwise Feature Selection

There are several methods of using similarity based measure for online feature learning. One way is the calculate similarity directly another is to use spectral feature selection(Li et al. 2013). The advantages of this approach is that it aims to determine relavancy and redundancy through relationships between features rather than evaluating them individually.

One proposed method of using streamwise selection is called *Online Group Feature Selection* algorithm (Li et al. 2013). It works leveraging a two-stage process:

1. Intra-Group Selection: this approach is to evaluate the underlying group structure in the same way expressed in the similarity based feature selection

2. Inter-Group Selection: the following to is to determine the optimal set of features taking into account features already in the model. Inter-group selection combines the features in the incumbent model and the intra-group step, which is used to obtain an optimal subset based on all features. This is done using regularization techniques like Lasso as expressed in the regularization section.

These two steps together form the basis for *Online Group Feature Selection* feature selection approach.

One of the strengths of this approach is the ability to understand the underlying group structure of the dataset. This is particularly important as it is a reasonable assumption to suppose batches of features which arrive are inter-related rather than running an underlying assumption that they are independent and ought to be treated one at a time. As this algorithm makes use of Spectral analysis for its feature selection, this can furthermore offer flexible choice between unsupervised or supervised techniques as well as custom kernel construction.

Despite being lauded as as an online feature selection approach, "Online Feature Selection with Group Structure Analysis" suffers from computational complexities related to similarity calculation. Furthermore the redundancy step is suggested using L_1 regulariser with a linear relationship between all features. This may not be a suitable assumptions when non-linear relationships are known to exist between predictors and response variables.

Related Approaches

There have been several approaches in combining disparate datasets together in past literature. Of interest are additive ensemble approaches and approaches rooted in Markov Chain Monte Carlo.

Ensemble Methods

A simple approach to consider an ensemble approach to the problem. In this setting we would simply treat the new data *independent* of the old data and construct a separate learner. Then we can combine it together using a super learner approach. There are several definitions of super learners, they are also known as stack generalises and other classes of models. The basis for these models is to build a "model of models". Let $g^k(x)$ be a generalizer g, where a generalizer is an appropriate output for an hypothesis learnt from the training set (Wolpert 1992), based on the training data x, and the subscript k denote the k model built where $k = 1, 2, ..., K$. For example $g_k(x)$ might represent a gradient boosted tree model, or a logistic regression, or a neural network model.

Unlike our treatment of boosting, we will not be updating our generalisers, rather we are seeking an ideal way to combine (already built) models together.

Bagging (Voting Algorithm)

The simplest approach is simply to take a vote or average of the most common classified result (Breiman 1996). This is most famously seen in bagging approaches in random forest algorithm (Bauer et al. 1999). The algorithm for Bagging is as follows

Algorithm *Bagging Predictors (Breiman 1996)*
Given base classifier f.
1. The data set is randomly divided into a test set $\mathcal{V}$ and a training set $\mathcal{T}$.
2. For $t = 1$ to T:
   1. A bootstrap sample $\mathcal{T}_B$ is selected from $\mathcal{T}$
   2. Construct a new base classifier f_t using $\mathcal{T}_B$.
3. Output the final classifier

$$F_T = \arg\max_{y \in Y} \sum_{i: f_i(x) = y} 1$$

i.e. the most often predicted label y.

In the construction of the base classifier (Breiman 1996) suggests using 10-fold cross-validation, and select any hyperparameters associated with f_t by using the original learning set $\mathcal{T}$. In other literature (Bauer et al. 1999) this does not appear to be necessarily required.

Stacking

If β_k are our weights and g_k are our final models which are not necessarily of the same base model, then the most simple linear super learners could be frames as a simple linear combination:

$$\sum_{k=1}^{K} \beta_k g_k(x)$$

Stack generalisers by (Wolpert 1992) was one of the first instances which demonstrated how one could combine estimates for neural networks, which was subsequently expanded by (Breiman 1996) under "stacked regression".

In both cases stacking method would aim to minimize

$$\sum_{i=1}^{N} l(\sum_{k=1}^{K} \beta_k g_k(x))$$

for some loss function l. For example, if we wish to use mean squared error then we would aim to minimize

$$\sum_{i=1}^{N} \left(y_i - \sum_{k=1}^{K} \beta_k g_k(x_i) \right)^2$$

In examples which (Leblanc & Tibshirani 1996) tried; the linear combinations did not have good predictive performance. However, when an additional constraint that the coefficients β_k

were to be non-negative, the final model $\sum_{k=1}^{K} \beta_k g_k(x)$ demonstrated better prediction error than any of the individual $g_k(\mathbf{x})$.

(Breiman 1996) also suggested adding an additional restriction $\sum_{k=1}^{K} \beta_k = 1$, where the ensemble is a convex combination of base learners. In his results he noted that this additional restriction did not have much empirical support, and made little difference.

Historically, in stacking implementations, the metalearning algorithm is often some sort of regularized linear model, however, a variety of parametric and non-parametric methods can used as a metalearner to combine the output from the base fits.

If we extend this model to use any arbitary learner instead of a linear learner as above, then the model is called a "Super Learner"(Laan, Polley & Hubbard 2007).

Markov Chain Approaches

The idea behind markov chain approaches is to dynamically select features can be seen in a variety of transdimensional models which focus on the "birth" and "death" states which can be framed in as model determination problem using Reversible Jump Markov Chain Monte Carlo framework (Denison & Smith 1998), (Hastie & Green 2012) or using an Indian Buffet Process in the Bayesian non-parametric sense (Palla, Knowles & Ghahramani 2017).

Reversible Jump Markov Chain Monte Carlo (RJMCMC)

In this setting the problem we are aiming to solve is around model determination problem, where the parameters of the model can be framed as the birth, death state of new features in a similar fashion as expressed (Denison & Smith 1998).

In order to approach this problem, we will be using the formulation by (Green 1995) where the metropolis hastings acceptance probability can be written in the form:

$$\min\{1, (\text{Bayes Factor}) \times (\text{prior ratio}) \times (\text{proposal ratio}) \times (\text{Jacobian})\}$$

In our example, we will ignore the Jacobian term as we will purposely draw new or changing components of new dimension parameters $\theta^{(k)}$ independently of their current values. This approach was suggested in (Mallick, Denison & Smith 1999)(Denison & Smith 1998).

The broad idea is to use the reversible jump Monte Carlo sampler for *step functions*(Green 1995). The step functions represent a *finite* collection of possible transformation functions which are used to build a design matrix, but too large to iterate over the whole collection. As pointed out by (Green 1995) there are "no claims of optimality for the particular choices made" and we are focused entirely on the flexibility in using this process.

Using the same notation as (Denison & Smith 1998), the Metropolis-Hastings acceptance can be written as:

$$\alpha = \min\left\{1, \frac{P(D|\theta')}{P(D|\theta)}\frac{P(\theta')}{P(\theta)}\frac{S(\theta' \to \theta)}{S(\theta \to \theta')}\right\}$$

Where θ denotes the current model parameters, θ' denotes the proposed model parameters, $S(\theta \to \theta')$ the probability of proposing a move to θ' from θ and D the data. The Jacobian ignored as drawing new parameters θ' will be independent from θ.

Bayesian Non-parametrics

The *birth-death feature allocation process* (BDFP) (Palla, Knowles & Ghahramani 2017) is based on the equilibrium distribution given by the Indian buffet process (IBP). The birth-death process is defined as a markov process which jumps to different states (birth and dealth).

Suppose s is a feature allocation with K_s nonzero features and s' is another feature allocation that differs from s in having one additional feature of size $|a|$, so that $K'_s = K_s + 1$

The state transition for *birth* is defined to be the transition rate as

$$q_{ss'} = R\frac{(|a|-1)!(N-|a|)!}{N!}$$

In comparison the state transition for death is

$$q_{s's} = \frac{Rr}{\alpha}$$

Where R is the parameter governing the birth rate and $\frac{R}{\alpha}$ is the parameter governing the death rate.

(Palla, Knowles & Ghahramani 2017) has demonstrated that this markov process has IBP(α) and is reversible

Conclusion

Online learning and online feature selection are topics that are more important than ever, and is now an important research topic in machine learning. The objects are to provide scalable approaches to learn relevant features in near real-time becomes more and more important.

This survey has provided framework for many streamwise feature selections and has offered differing approaches to how we might build and update models through use of gradient descent methods and other statistical techniques.

The aim of this survey is not to claim superiority for a particular approach, but merely to provide a new framework for which real-time online learning and feature selection may be performed for particular use cases.

The future direction for online feature selection would be two fold. Firstly, to determine how best to leverage knowledge around the dependencies and relationships within incoming features to select diverse features, in both supervised and unsupervised sense. Secondly, to extend existing non-linear modelling frameworks to allow dynamically updating models when new features are streamed in.

References

Agarwal, A., Chapelle, O., Dudík, M. & Langford, J. 2014, 'A reliable effective terascale linear learning system', *Journal of Machine Learning Research*, vol. 15, pp. 1111–33.

Bauer, E., Kohavi, R., Chan, P., Stolfo, S. & Wolpert, D. 1999, 'An empirical comparison of voting classification algorithms: Bagging, boosting, and variants', *Machine Learning*, vol. 36, no. August, pp. 105–39.

Breiman, L. 1996, 'Bagging Predictors', *Machine Learning*, vol. 24, no. 421, pp. 123–40.

Chen, T. & Guestrin, C. 2016, 'XGBoost: A scalable tree boosting system', *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, NY, USA, pp. 785–94.

Denison, M., David G. T. & Smith, A.F.M. 1998, 'Bayesian MARS', *Statistics and Computing*, vol. 8, no. 4, pp. 337–46.

Duchi, J., Hazan, E. & Singer, Y. 2011, 'Adaptive subgradient methods for online learning and stochastic optimization', *Journal Machine Learning Research*, vol. 12, pp. 2121–59.

Green, P. 1995, 'Reversible jump Markov chain Monte Carlo computation and Bayesian model determination', *Biometrika*, vol. 82, no. 4, pp. 711–32.

Hastie, D.I. & Green, P.J. 2012, 'Model choice using reversible jump Markov chain Monte Carlo', *Statistica Neerlandica*, vol. 66, no. 3, pp. 309–38.

Hou, C., Nie, F., Yi, D. & Wu, Y. 2011, 'Feature selection via joint embedding learning and sparse regression', *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, AAAI Press, Barcelona, Catalonia, Spain, pp. 1324–9.

Karampatziakis, N. & Langford, J. 2011, 'Online importance weight aware updates', *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, AUAI Press, Arlington, Virginia, United States, pp. 392–99.

Kingma, D.P. & Ba, J. 2014, 'Adam: A method for stochastic optimization', *arXiv*, vol. arXiv:1412.698.

Laan, M.J. van der, Polley, E.C. & Hubbard, A.E. 2007, 'Super Learner', *Statistical Applica-*

tions in Genetics and Molecular Biology, vol. 6, no. 1, p. Article25.

Leblanc, M. & Tibshirani, R. 1996, 'Combining Estimates in Regression and Classification', *Journal of the American Statistical Association*, vol. 91, no. 436, pp. 1641–50.

Li, H., Wu, X., Li, Z. & Ding, W. 2013, 'Online group feature selection from feature streams', *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI Press, Bellevue, Washington, pp. 1627–28.

Mallick, B.K., Denison, D.G.T. & Smith, A.F.M. 1999, 'Bayesian survival analysis using a MARS model', *Biometrics*, vol. 55, no. 4, pp. 1071–7.

Palla, K., Knowles, D. & Ghahramani, Z. 2017, 'A birth-death process for feature allocation', in D. Precup & Y.W. Teh (eds),*Proceedings of the 34th international conference on machine learning*, PMLR, International Convention Centre, Sydney, Australia, pp. 2751–59.

Perkins, S. & Theiler, J. 2003, 'Online feature selection using grafting', *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, AAAI Press, Washington, DC, USA, pp. 592–99.

Perkins, S., Lacker, K. & Theiler, J. 2003, 'Grafting: Fast, incremental feature selection by gradient descent in function space', *Journal of Machine Learning Research*, vol. 3, pp. 1333–56.

Roffo, G., Melzi, S. & Cristani, M. 2015, 'Infinite feature selection', *The IEEE International Conference on Computer Vision (ICCV)*.

Shamir, G.I. 2015, 'Minimum description length (MDL) regularization for online learning', *Proceedings of the 1st International Workshop on Feature Extraction: Modern Questions and Challenges*, PMLR, Montreal, Canada, pp. 260–76.

Tibshirani, R. 1996, 'Regression selection and shrinkage via the lasso', *Journal of the Royal Statistical Society, Series B*, vol. 58, no. 1, pp. 267–88.

Toulis, P., Rennie, J. & Airoldi, E.M. 2014, 'Statistical analysis of stochastic gradient methods for generalized linear models', *Proceedings of the 31st International Conference on International Conference on Machine Learning*, JMLR, Beijing, China, pp. 667–75.

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

Wolpert, D.H. 1992, 'Stacked generalization', *Neural Networks*, vol. 5, no. 2, pp. 241–59.

Yang, H., Fujimaki, R., Kusumura, Y. & Liu, J. 2016, 'Online feature selection: A limited-memory substitution algorithm and its asynchronous parallel variation', *Proceedings of the 22Nd acm sigkdd international conference on knowledge discovery and data mining*, ACM, New York, NY, USA, pp. 1945–54.

Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S. & Stoica, I. 2016,

'Apache spark: A unified engine for big data processing', *Communications of the ACM*, vol. 59, no. 11, pp. 56–65.

Zhao, Z. & Liu, H. 2007, 'Spectral feature selection for supervised and unsupervised learning', *Proceedings of the 24th International Conference on Machine Learning*, ACM, New York, NY, USA, pp. 1151–57.

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