Active Learning for High-Dimensional Binary Features

Ali Vahdat
Noah's Ark Lab
Huawei Technologies
Toronto, Canada
ali.vahdat@huawei.com

Mouloud Belbahri
Dept. of Mathematics and Statistics
University of Montreal
Montreal, Canada
mouloud.belbahri@umontreal.ca

Vahid Partovi Nia
Noah's Ark Lab
Huawei Technologies
Montreal, Canada
vahid.partovinia@huawei.com

***Abstract*—Erbium-doped fiber amplifier (EDFA) is an optical amplifier/repeater device used to boost the intensity of optical signals being carried through fiber optic communication networks. A highly accurate EDFA model – to predict the signal gain for each channel – is required because of its crucial role in optical network management and optimization. EDFA channel inputs (i.e. features) either carry signal or are idle, therefore they can be treated as binary features. However, channel outputs (and the corresponding signal gains) are continuous values. Labeled training data is very expensive to collect for EDFA devices, therefore we devise an active learning strategy suitable for binary features to overcome this issue. We propose to take advantage of sparse linear models to simplify the predictive model. This approach improves signal gain prediction and accelerates active learning query generation. We show the performance of our proposed active learning strategies on simulated data and real EDFA data.**

***Index Terms*—Active learning, binary features, optical networks.**

I. Introduction

We start by introducing the Erbium-doped fiber amplifier (EDFA) device, and subsequently review some of the works in the literature on active learning.

A. EDFA

With the rising demand for data storage and computation, there is an increasing adoption of cloud computing networks, which are heavily dependent on the deployment of fiber-optic networks and the components that facilitate faster interconnection in these networks. Optical amplifiers made it possible to amplify together all the wavelength-division multiplexing (WDM) channels, their bit rate, modulation format and protocols, and eliminate the need to regenerate the optical signals every 80-100 *kms*. They are used as optical repeaters over the long distance optical fiber cables that carry much of the world's telecommunication links.

A highly accurate EDFA model is crucial for a number of different reasons, such as: i) to predict the light path performance, ii) to calculate optical signal to noise ratio (OSNR) and iii) to improve performance of light path setup, which in turn facilitate improved performance and better resource allocation on optical networks. However, collecting labeled data from EDFA devices – which is required to train an accurate model – is expensive. It involves an expert technician in lab environment to set the device inputs and collect and record the input-output signal levels. This is where active learning (AL) strategies are utilized to collect data that is more promising to improve the accuracy of the EDFA model.

An EDFA device receives the input signal at a channel's input and the amplified signal leaves the same channel's output. A typical EDFA device supports between 40 and 128 channels depending on the manufacturer and type. Each channel can carry the optical signal for a different service, but not all channels carry service signal at all times. Channels carrying valid service signals are interpreted as *on* and others are interpreted as dummy or *off* channels. Therefore, input signals can be deemed as binary or $x \in \{-1, 1\}$. Rather than the actual strength of the channel output, we are interested in the channel signal strength change which we refer to as gain:

$$y_c = \text{gain}_c = 10 \log_{10} (\text{out}_c / \text{in}_c)$$

where in_c and out_c are channel c's input and output signal powers, and c is the channel index, $c \in \{1, \ldots, C\}$, (C is the number of EDFA channels). Therefore, y is a continuous random variable.

Here, our objective is to use active learning to improve performance of a predictive model for a single EDFA channel. Channel outputs are independent of each other given channel inputs, so the generalization towards multivariate output is straightforward.

B. Active Learning

State-of-the-art machine learning (ML) algorithms require significant amount of data to learn an accurate model. In most applications, there is either not enough data, or most of the available data are unlabeled, and labeling them is often time-consuming and/or expensive. This gives rise to a category of algorithms called active learning (AL). AL methods start by training a model on an initial set of labeled data. Then they utilize the model uncertainty to identify the most promising (unobserved) subset of data to improve model performance. The algorithm then requests the labels for this subset, adds them to the initial labeled set, and re-trains the model. ML algorithms are capable of achieving better performance if the learning algorithm is integrated into the data collection process. This is the main objective of AL methods.

978-3-903176-24-9

There is a large body of literature on active learning [5, 2]. Methods using *uncertainty sampling* [9, 7] query data points with the highest uncertainty. After observing more points in the uncertain region, the learning algorithm becomes more confident about the neighboring subspace of the queried data point. In [6] authors train a regressor that predicts the expected error reduction for a candidate data point in a given learning state. The experience from previous AL outcomes is utilized to learn strategies for query selection.

Motivated with the EDFA application, we develop an AL algorithm for data with discrete features and a continuous response. A data point selected to be inquired about its label is usually referred to as a *query*, and the entity providing the label for the queried data point is usually called an *oracle*. Oracle can be a human, a database, or a software providing the label for the query.

II. Methodology

A typical pool-based AL algorithm has access to a small pool of labeled data$D_L = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i \in \mathbb{R}^p$ is the predictor and $y_i \in \mathbb{R}$ is the response. Also, there is a potentially larger pool of unlabeled data $D_U = \{\mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \ldots, \mathbf{x}_m\}$. In the EDFA application each $\mathbf{x}$ is a set of p input channels, and y is one of the output channels selected for modeling. Output channels are conditionally independent, which allows us to model each output channel independently.

A typical AL algorithm starts by training a model using the labeled pool D_L. Then, at each iteration an AL strategy selects a promising data point $\mathbf{x}_i$ from the unlabeled pool D_U and queries its label y_i. Once label is retrieved for $\mathbf{x}_i$, this data point $(\mathbf{x}_i, y_i)$ is removed from D_U and is added to D_L. The classifier is then trained on the new pool D_L, including the recently added $(\mathbf{x}_i, y_i)$. This process is repeated until a termination criteria – usually a sampling budget T – is reached. With a small sampling budget T, the goal of AL is to find the best sequence of data points to be queried in order to maximize the test accuracy of the model.

Suppose the response variable is measured with an additive statistical error ε and the relationship between the response and the predictors is fully-determined by a linear function

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{1}$$

where $\mathbf{y}_{n\times 1}$ is the vector of observed response, $\mathbf{X}_{n\times p}$ is row-wise stacked matrix of predictors, $\boldsymbol{\beta}_{p\times 1}$ is the p-dimensional vector of coefficients, and $\boldsymbol{\varepsilon}_{n\times 1}$ is white noise with zero mean and a constant variance σ^2. There is a strong reason to start with a linear model. A linear model with interactions fully describes any complicated model built over discrete features, suitable for the EDFA data setting. The coefficients $\boldsymbol{\beta}$ are unknown in practice, and are estimated using least squares, $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$.

In ultra high-dimensional settings ($p \gg n$) where most feature selection methods fail computationally, it is suggested to order the features with a simple measure of dependence like Pearson correlation and select some of relevant features. This simplifies the ultra high-dimensional setting to a high-dimensional setting [4] where $p \sim n$ and feature selection methods are computationally feasible. In AL, ultimately, a query is generated with an estimated model dimension $m \ll p$.

A. Feature ordering

In active learning for EDFA, model building starts with small number of observations n, say $n \approx 20$. If the feature dimension $p \gg 40$, least squares estimate of coefficients $\hat{\boldsymbol{\beta}}$ are ill conditioned, because $\mathbf{X}^\top\mathbf{X}$ is rank-deficient. Regularization, feature selection, dimension reduction, are common solutions to this. Here we focus on sparse estimation of the coefficients often implemented by L_1 regularization or *lasso* [11].

However, L_1 regularization is still computationally expensive for large $p \gg n$. [4] recommends *sure* screening to pre-select a subset of features with large absolute correlation (with the response), and then to run L_1 regularization on this subset. This dimensionality reduction is fast and requires only $\mathcal{O}(np)$ operations to compute the correlations, and $\mathcal{O}(n \log n)$ to order them. The total computation complexity of *sure* is $\mathcal{O}(pn \log n)$.

B. Feature selection

In a linear model with p covariates, there are 2^p candidate models. The *lar* (least angle regression) algorithm [3] computes the *lasso* with minor modifications, however its implementation is significantly faster. Choosing the model dimension and choosing the L_1 regularization constant λ are inter-related. The *lar* algorithm efficiently computes the path of $\hat{\boldsymbol{\beta}}(\lambda_j)$ over a sequence of λ_j that the parameter dimension changes. The *lar* algorithm finds the path of λ_j and individual estimates $\hat{\boldsymbol{\beta}} \mid \lambda_j, j = 1, \ldots, p$, with the same computational complexity of a single least square. So one can choose a value λ_j, and evaluate the model for the effective dimension imposed by it. Repeating the same process for all model dimensions and picking the best model dimension m from the p candidate models is efficient since it allows to avoid evaluating 2^p candidate models and reduce the search space to only p model evaluations.

For each λ_j its corresponding nonzero $\hat{\boldsymbol{\beta}}(\lambda_j)$ are selected to create a new design matrix $\mathbf{X}_j$ with dimension $n \times m$. The best model is chosen by maximizing the predictive log likelihood ℓ_j. Theorem 1 derives the predictive log likelihood for small sample sizes inline with the BIC of [8]. It is not difficult to see this predictive model is asymptotically equivalent to the BIC. However, in small samples they behave differently.

Theorem 1: Suppose $\mathbf{X}^\top\mathbf{X}$ is positive definite and

$$\begin{aligned} \mathbf{y} &\sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}), \\ \boldsymbol{\beta} &\sim \mathcal{N}(\hat{\boldsymbol{\beta}}, n\sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}). \end{aligned}$$

The predictive log likelihood

$$\ell_j = \log \int_\infty^\infty \cdots \int_\infty^\infty f(\mathbf{y} \mid \boldsymbol{\beta}, \mathbf{X}) dF(\boldsymbol{\beta} \mid \mathbf{X})$$

simplifies to $\ell_j = \ell(\hat{\boldsymbol{\beta}}) - 0.5\log(n+1)$, where $\hat{\boldsymbol{\beta}}$ is the maximum likelihood estimate of $\boldsymbol{\beta}$ and $\ell(.)$ is the log likelihood function. Refer to the arXiv version of the paper at https://arxiv.org/abs/1902.01923 for the proof.

C. Linear query generation

Linear models are attractive because the class of linear models including main effects with interactions cover any complex function on discrete features. We start with a linear model with main effects only (and no interaction) to create an extremely fast query generation method, called *query-by-sign*. Then generalize it to a linear model with main effects and pairwise (i.e. 2nd order) interactions to trade off some computation for improved accuracy. We call this method *query-by-variance*. To allow for models with higher order interactions we utilize bagged trees to produce *query-by-bagging*.

In AL context, the objective is to request a new observation that most improves the model performance. There are two major paradigms to interpret model performance; i) smaller variance of prediction $\hat{\mathbf{y}}$, and ii) smaller variance of estimators $\hat{\boldsymbol{\beta}}$. Here we choose the former approach, and focus on improving prediction accuracy as the objective.

In an AL setting, at each iteration, a new data point $\underline{\mathbf{x}}_{1\times p}$ is queried, and after observing its response variable $y(\underline{\mathbf{x}})$ the training set is updated. Therefore, we use the notation $\hat{\boldsymbol{\beta}}(\underline{\mathbf{x}})$ to emphasize that this new $\hat{\boldsymbol{\beta}}$ is estimated after adding this new observation to the previously observed design matrix $\mathbf{X}_{n\times p}$.

The new design matrix, after adding the new observation is

$$\underline{\mathbf{X}}_{(n+1)\times p} = \begin{bmatrix} \mathbf{X}_{n\times p} \\ \underline{\mathbf{x}}_{1\times p} \end{bmatrix},$$

Note that the prediction variance is a function of the new observation $\underline{\mathbf{x}}$ only. To improve prediction accuracy, we query a new observation $\underline{\mathbf{x}}$ under which the model prediction has the largest uncertainty $\mathbb{V}\{\hat{y}(\underline{\mathbf{x}})\}$. From *conditional variance theorem* [10]

$$\mathbb{V}\{\hat{y}(\underline{\mathbf{x}})\} = \mathbb{E}_{\underline{\mathbf{x}}}[\mathbb{V}\{\hat{y}(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}}\}] + \mathbb{V}_{\underline{\mathbf{x}}}[\mathbb{E}\{\hat{y}(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}}\}].$$

Since $\underline{\mathbf{x}}$ is a query and under our control, this simplifies to

$$\mathbb{V}\{\hat{y}(\underline{\mathbf{x}})\} = \mathbb{V}\{\hat{y}(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}}\}.$$

From linear model assumption the response variance $\mathbb{V}\{y(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}}\}$ is constant σ^2.

To maximize the prediction variance $\mathbb{V}\{\hat{y}(\mathbf{x}) \mid \underline{\mathbf{x}}\}$ one needs to keep the maximizer scale-invariant, otherwise any direction $\mathbf{x}$ with a large scale c is a solution because $\mathbb{V}(c\mathbf{x}) = c^2\mathbb{V}(\mathbf{x})$. Suppose $\underline{\mathbf{x}}$ is of a fixed norm to avoid scaling, therefore

$$\underset{\underline{\mathbf{x}}}{\operatorname{argmax}}\ \mathbb{V}\{\underline{\mathbf{x}}^\top\hat{\boldsymbol{\beta}}\} = \underset{\underline{\mathbf{x}}}{\operatorname{argmax}}\ \sigma^2\underline{\mathbf{x}}^\top(\mathbf{X}^\top\mathbf{X})^{-1}\underline{\mathbf{x}}, \quad (2)$$

where σ^2 is a constant and can be ignored in maximization.

Equation (2) is key to active learning for linear models. Suppose the model dimension is estimated properly $m < p$, and $\underline{\mathbf{x}}$ is continuous, $\underline{\mathbf{x}} \in \mathbb{R}^{\mathrm{m}}$. The scale-invariant solution query generation requires maximizing (2) subject to a bounded norm $\underline{\mathbf{x}}^\top\underline{\mathbf{x}} = c^2$ as in Theorem 2.

Theorem 2:

$$\begin{aligned} \hat{\underline{\mathbf{x}}} \quad = \quad & \operatorname{argmax}\ \underline{\mathbf{x}}^\top(\mathbf{X}^\top\mathbf{X})^{-1}\underline{\mathbf{x}}, . \\ & \text{s.t.}\ \ \underline{\mathbf{x}}^\top\underline{\mathbf{x}} = c^2, \end{aligned} \quad (3)$$

It is easy to show that $\hat{\underline{\mathbf{x}}} = c\ \mathbf{e}_{\min}$ where $\mathbf{e}_{\min}$ is the eigenvector associated with the smallest eigenvalue of $\mathbf{X}^\top\mathbf{X}$. The computational cost of this solution is $O(m^2)$, which is negligible for small m. Refer to the arXiv version for the proof.

The application of (3) is not restricted to continuous feature space. If feature space is binary, $\underline{\mathbf{x}} \in \{-1, +1\}^m$, then $\underline{\mathbf{x}}^\top\underline{\mathbf{x}} = m$ and a relaxed approximate solution is

$$\hat{\underline{\mathbf{x}}} = \operatorname{sign}(\mathbf{e}_{\min}). \quad (4)$$

D. Ensemble-based query generation

In many applications the prediction function is a nonlinear function. While a linear model helps to identify important features, they are usually inaccurate for these applications. As a consequence, an inaccurate prediction model leads to generating sub-optimal queries for AL. We address both issues by fitting a flexible ensemble of trees on the sparse features, and relax the constant variance assumption by computing the empirical variance of the prediction.

Bagging [1] is a method for fitting an ensemble of learning algorithms trained on bootstrap replicates of the data in order to get an aggregated predictor. Suppose that B bootstrap replicates are sampled from the observed n independent data and for $b = 1, \ldots, B$, a regression tree T_b is fitted. Therefore the response prediction is $\hat{\mathbf{y}} = B^{-1}\sum_b \hat{\mathbf{y}}_b$, where $\hat{\mathbf{y}}_b = \hat{T}_b(\mathbf{x})$ is the prediction output of a single tree. Hence, the prediction variance $\mathbb{V}\{\hat{y}(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}}\}$ is estimated by the empirical variance

$$\hat{\mathbb{V}}\{\hat{y}(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}}\} = \frac{1}{B-1}\sum_b \{\hat{y}_b(\underline{\mathbf{x}}) - \hat{y}(\underline{\mathbf{x}})\}^2.$$

In AL context, the query-by-bagging suggests $\hat{\underline{\mathbf{x}}}$ that maximizes the empirical variance such as

$$\hat{\underline{\mathbf{x}}} = \underset{\underline{\mathbf{x}}}{\operatorname{argmax}}\ \hat{\mathbb{V}}\{\hat{y}(\underline{\mathbf{x}}) \mid \underline{\mathbf{x}}\}. \quad (5)$$

III. Experimental Analysis

We evaluate our methods on two sets of data; synthetic simulated data, and real-world EDFA data.

A. Simulations

We conduct a simulation study to assess the performance of the three proposed active learning methods: *query-by-sign*, *query-by-variance* and *query-by-bagging*. Each method has its associated fitted model; a linear model using main effects only for *query-by-sign*, a linear model using main effects and 2nd order interaction terms for *query-by-variance*, and an ensemble of bagged trees for *query-by-bagging*. We compare the three different query generation strategies against random sampling. We evaluate the performance of these methods by varying the complexity of the simulated data.

We synthetically generate two simulated data sets; in the first scenario data is simulated by a linear model using main effects and 2nd order interactions, and the second scenario uses main effects, 2nd and 3rd order interactions. In the first scenario query-by-sign fails because the fitted model only incorporates main effects, and therefore is not an accurate approximation of the data. Query-by-variance and query-by-bagging AL strategies

outperform the random sampling strategy and eventually find the "true" model as the sampling budget increases, however, query-by-bagging finds the "true" model more smoothly.

In the second scenario the 3rd order interaction terms are added to the data simulation model. As shown in Figure 1, query-by-variance (left panel) fails compared to the random sampling strategy. This suggests that query-by-variance needs to be adjusted if significant 3rd order interactions are present in the model. However, query-by-bagging (right panel), outperforms the random sampling.

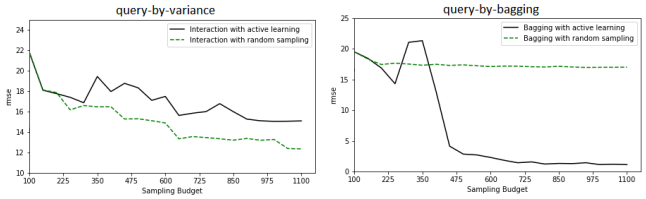

Fig. 1. Validation RMSE on data simulated by a linear model with main effects, 2nd and 3rd order interactions using two different AL strategies; query-by-variance (left) and query-by-bagging (right).

For a comprehensive experimental analysis on simulated data please refer to the arXiv version of the paper.

B. Application

We apply our query generation methods to the data collected from the optical amplifier equipment (EDFA). Our data set contains about 9000 observations for an EDFA device with 40 channels. We divide the data set into training, validation, and test sets with 50%, 25% and 25% splits. We further divide the training set into a labeled pool of 100 observations and an unlabeled pool with the rest. Sampling budget is 1000.

Allowing a large number of features in the model renders query generation computationally expensive, and forces a less frequent feature selection updates. However, keeping the maximum number of features in the model small, allows faster query generation, and updates the features selection more frequently. Therefore a fine balance should be maintained between maximum number of features allowed in the bagging model and the update frequency of feature selection.

Allowing up to 18 features in the ensemble trees reduces the final validation RMSE to 0.085 (Figure 2) which is enough to save multiple hours of engineers' time for collecting labeled data. On the right side of this figure we can further observe the increasing model size as more and more observations are queried by AL. Although the model can add more useful features or drop less useful ones at each model update step (as can be seen from the oscillating model size graph), the model using the AL strategy takes better advantage of this freedom compared to the random sampling strategy, and reaches the maximum number of features allowed to index for modeling and query generation (i.e. 18.) The performance of AL strategy increases as model size upper bound increases to 20 or higher, but this comes with a computational cost.

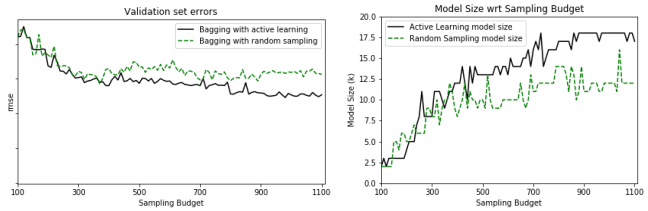

Fig. 2. Validation RMSE on EDFA data using query-by-bagging (left). The estimated model size as the number of samples increases (right).

IV. Conclusion

Active learning helps make better use of limited labeling budget by integrating data collection process into the learning algorithm. We proposed three different active learning strategies with different computational costs and running times. The simplest strategy, query-by-sign, only considers main effects of a linear model for query generation. Query-by-variance takes advantage of 2nd order interactions, and query-by-bagging considers high-order interactions by using an ensemble of trees to model data and generate the queries. We simulated data using models with 2nd and 3rd order interactions, and compared the three different active learning strategies. We then applied our findings to EDFA data, a very small and highly complex data set. We observed that query-by-bagging, when tuned properly, improves the model prediction performance and saves engineers' data collection time. Also, the simpler sampling strategy, query-by-variance, displays interesting results, but on data sets with lower-order main effect interactions.

References

[1] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[2] H.-M. Chu and H.-T. Lin. Can active learning experience be transferred? In *ICDM*, pages 841–846. IEEE, 2016.

[3] B. Efron, T. Hastie, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[4] J. Fan and J. Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B*, 70(5):849–911, 2008.

[5] W.-N. Hsu and H.-T. Lin. Active learning by learning. In *AAAI*, pages 2659–2665, 2015.

[6] K. Konyushkova, R. Sznitman, et al. Learning active learning from data. In *NeurIPS*, pages 4228–4238, 2017.

[7] M. E. Ramirez-Loaiza, A. Culotta, and M. Bilgic. Anytime active learning. In *AAAI*, pages 2048–2054, 2014.

[8] G. Schwarz et al. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[9] M. Sharma and M. Bilgic. Most-surely vs. least-surely uncertain. In *ICDM*, pages 667–676. IEEE, 2013.

[10] R. Sheldon et al. *A first course in probability*. Pearson Education India, 2002.

[11] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.