Applications of Machine Learning in Cable Access Networks

Karthik Sundaresan, Nicolas Metts, Greg White; Albert Cabellos-Aparicio
CableLabs; UPC BarcelonaTech.

Abstract

Recent advances in Machine Learning algorithms have resulted in an explosion of real-world applications, including self-driving cars, face and emotion recognition, automatic image captioning, real-time speech translation and drug discovery.

Potential applications for Machine Learning abound in the areas of network technology, and there are various problem areas in the cable access network which can benefit from Machine Learning techniques. This paper provides an overview of Machine Learning algorithms, and how they could be applied to the applications of interest to cable operators and equipment suppliers.

INTRODUCTION

Recent advances in Machine Learning, and in particular, deep learning, have resulted in an explosion of real-world applications, including self-driving cars, face and emotion recognition, automatic image captioning, real-time speech translation and drug discovery. Powerful new learning algorithms, harnessed via open-source toolkits and running on general-purpose GPUs (even cloud instances) are the key enablers for this explosion.

Machine Learning technologies can learn from historical data, and make predictions or decisions, rather than following strictly static program instructions. They can dynamically adapt to a changing situation and enhance their own intelligence by learning from new data. Statistical pattern recognition, data mining, clustering and deep learning have proven to be powerful components in the Machine Learning space.

Potential applications for Machine Learning abound in the areas of network technology and applications. It can be used to intelligently learn the various environments of networks and react to dynamic situations better than a fixed algorithm. When it becomes mature, it would greatly accelerate the development of autonomic networking.

There are various problem areas in the cable access network which can benefit from Machine Learning techniques. For example, the Proactive Network Maintenance data which represents network conditions obtained from CMTS and CM can be processed through Machine Learning algorithms to enable automatic recognition of issues in the cable plant and initiating the needed corrective actions. Patterns in network traffic, equipment failures, device reboots or link failures can be used to identify the root cause of various network problems. Video channel usage data could be used to optimize the multicast channel lineups in a network, and be used to predict which programs to record automatically in a DVR.

This paper provides an overview of Machine Learning algorithms with focus on the applications of interest to cable operators and equipment suppliers. The paper discusses several learning toolkits and computational platforms, and hypothesizes some of the most compelling near-term uses. It will explore how to apply learning algorithms to various challenges faced by operators in managing network devices and services. In addition, the paper covers best practices for achieving optimal results with Machine Learning.

OVERVIEW OF MACHINE LEARNING

This section provides a very brief introduction to Machine Learning. The reader familiar with such techniques may safely skip this section. Machine Learning (ML) is a field of study of Artificial Intelligence that provides computers with learning techniques without being explicitly programmed. Traditional programming requires manually coding an algorithm and combining it with input data to produce the resulting output data. This model is fundamentally different in ML: a learning algorithm (e.g., a Neural Network) is trained with the input data and the corresponding output data, which is known as training set, to automatically produce the algorithm, which will be used to predict future results. But how do computers actually learn such an algorithm?

Systems can be always described by a mathematical function and in many cases such a function is multi-dimensional (i.e., has many input and output variables, maybe hundreds) and is non-linear. If we have such a function, we have a complete model of the system. For instance, the function that describes a network may have the form: function (topology, load, configuration) = performance. That is, the function links the topology of the network, the traffic (load) and configuration (e.g., routing) with the resulting performance (e.g., delay). Analytical modeling provides tools to obtain such a function (e.g., Queueing theory in networking). However, such techniques often use simplifications and cannot be applied when the network is too complex.

Machine Learning provides a different way to obtain such a function. As shown in Figure 1 below, a learning algorithm fed with a representative training set will fit the function, even if such function has hundreds of variables or it is complex and non-linear. The algorithm produced by a Machine Learning approach is the function itself and can also produce accurate answers for input points for which it has not been trained explicitly, provided that the initial training set was representative enough. This is accomplished by interpolation, that is estimating an unseen prediction point between two already seen test points and extrapolation, estimation of the point beyond the observed training set.

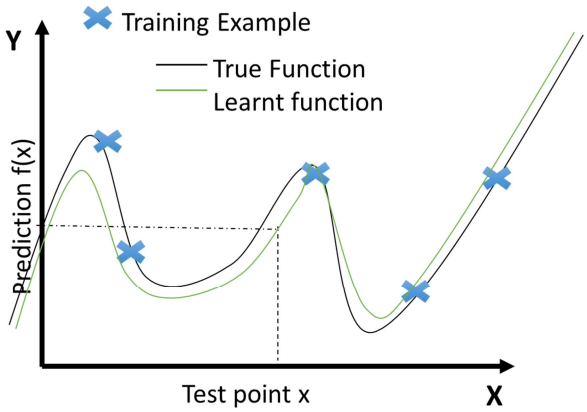

Figure 1. An overview of Machine learning

In what follows, we provide a summary of the two most relevant techniques in Machine Learning: supervised and unsupervised learning.

SUPERVISED LEARNING

In supervised learning, there is a given data set and the knowledge on what the correct output looks like. The idea is that there is a relationship between the input and the output.

The input data set is called training data and has a known label or result, such as cat/not-cat (for identifying cat photos) or a stock price at a time (for predicting stock prices). Network examples could include classifying traffic patterns for intrusion detection as harmful-traffic vs. normal-traffic, or predicting the usage of a certain network link based on various parameters. A model is prepared through a training process where it is required to make predictions and is corrected when those predictions are wrong. The

training process continues until the model achieves a desired level of accuracy on the training data.

Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, the goal is to predict results within a continuous output, essentially trying to map the input variables to some continuous function. In a classification problem, the goal is to predict results in a discrete output, trying to map the input variables into discrete output categories.

Example classification algorithms include Logistic Regression and the Back Propagation Neural Network, SVMs, kNN, etc.

Support Vector Machine (SVM)

SVM is type of supervised Machine Learning algorithm. Given a set of training examples, each marked as belonging to one of two classes, an SVM training algorithm builds a model that assigns new examples into one class or the other. A hyperplane splits the input variable space. In SVM, a hyperplane is selected to best separate the points in the input variable space by the class to which they belong. The SVM learning algorithm finds the coefficients that results in the best separation of the classes by the hyperplane. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

The distance between the hyperplane and the closest data points is referred to as the margin. The best or optimal hyperplane that can separate the two classes is the line that has the largest margin. Only these points are relevant in defining the hyperplane and in the construction of the classifier. These points are called the support vectors as they support or define the hyperplane.

K-Nearest Neighbors Algorithm (kNN)

kNN makes predictions for a new data point by assigning it the value/label of its K nearest neighbors. For regression this might be the mean output variable, in classification this might be the mode (or most common) class value.

In kNN classification, an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. In kNN regression, the output value is the average of the values of its k nearest neighbors.

The simplest technique is to use the Euclidean distance between neighbors. kNN performs a calculation (or learns) when a prediction is needed, just in time. The training instances can be updated or curated over time to keep predictions accurate.

UNSUPERVISED LEARNING

Unsupervised learning attacks problems with little or no idea what the results should look like. The idea is to derive structure from data where there is no apriori knowledge of the effect of the variables. This structure is derived by clustering the data based on relationships among the variables in the data. With unsupervised learning there is no feedback based on the prediction results. The input data set is not labelled and does not have a known result.

A model is prepared by deducing structures present in the input data. This may be to extract general rules. It may be through a mathematical process to systematically reduce redundancy, or it may be to organize data by similarity.

Typical unsupervised algorithms are Clustering (K-Means, Fuzzy and Hierarchical

clustering, etc.), Principal Component Analysis and the Apriori Algorithm.

Clustering Algorithms (K-means, Fuzzy and Hierarchical Clustering)

Give a set of data points, clustering is the problem of grouping the points in such a way that points in the same cluster are more similar to each other than to those in other clusters. Usually, the points are in a high-dimensional space, and similarity is defined using a distance measure. As an example, let's assume that we want to understand content consumption among users, movies for instance. If we have access to a large dataset of movie consumption per user, K-Means will cluster and group users with similar movie preferences. With this we can analyze the profiles of such users and be able to predict which types of movies they will consume.

The K-means algorithm takes a set of points and attempts to identify the set of K clusters for which the average distance of each point to the centroid (mean) of its cluster is minimized. In an algorithm like K-means, data is divided into distinct clusters, where each data element belongs to exactly one cluster. In Fuzzy clustering, data elements can belong to more than one cluster, and each element belongs to each cluster to a certain degree (e.g., a likelihood of belonging to the cluster). Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters.

Principal Component Analysis (PCA)

Sometimes data is collected on a large number of variables from a single population. With a large number of variables, the data will be too large to study and interpret properly. There would be too many pairwise correlations between the variables to consider. To interpret the data in a more meaningful form, it is necessary to reduce the number of variables to a few, interpretable linear combinations of the data. Each linear combination will correspond to a principal component. Principal components are the underlying structure in the data. They are the directions where there is the most variance, the directions where the data is most spread out. PCA is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. With PCA once these patterns in the data are found, one can compress the data, i.e., by reducing the number of dimensions, without much loss of information.

Apriori Algorithm

Mining for associations among items in a large database of transactions is an important database mining function. Apriori algorithm is a technique to find the sets of frequent items and learning the association rules between items in a dataset. The algorithm works by identifying frequent individual items in a dataset and extending them to larger item sets as long as those sets appear sufficiently often in the dataset. These frequent item sets can be used to determine association rules which highlight general trends in that data set.

The interested reader can find further information on Machine Learning in [4], [6].

APPLYING MACHINE LEARNING TO NETWORKING PROBLEMS

With the exponential growth in network traffic (over-the-top video, social networking, cloud services, etc.), there is just too much data to manually learn and understand the many patterns of traffic flow. Network equipment is getting more complex, e.g., bringing up a router or a CMTS can take thousands of lines of configuration. Along with this comes the increasing complexity of the network itself, e.g., various overlays, underlays and connection of services across disparate networks. Traffic modeling and analysis are becoming increasingly

challenging. With the proliferation of applications, it is difficult to predict and to generalize application behavior and the impact it has on network traffic. Simply put, there are too many sources of information for manual processing by humans. ML techniques are ideal solutions for these kind of problems.

One example in the access network is how an operator can effectively and quickly process all the PNM data which comes up from the access network devices. As another example, the problem of dynamically fitting optimal DOCSIS 3.1 OFDM Profiles to hundreds of modems on an OFDM channel with thousands of subcarriers with changing conditions is a complex challenge.

Also, to understand and predict network behavior, some problems in networks are so complex that one cannot express them in a simple fashion, in those cases it is easier to represent the problem as a black box with a set of inputs and outputs. That can now help frame the problem as a Machine Learning problem and start making progress towards solving it in a timely fashion.

Applying Machine Learning to networking has been proposed several times in the past, a notable example is D. Clark and the Knowledge Plane [5]. However and for many use-cases it is very challenging to apply learning techniques to an inherently distributed system. The rise of the Software-Defined Networking (SDN) paradigm and Network Telemetry are enabling technologies for applying Machine Learning. This provides all the data needed from the network elements at a centralized server; ML algorithms can learn from this centralized repository of data. Also since the SDN controller can configure and manage the network elements, it provides an avenue for the ML algorithm to take corrective action on the network based on the patterns the algorithm sees.

Many of these network problems naturally lend itself to be solved by Machine Learning algorithms, which can obtain the best possible knowledge from the large sets of data available on hand.

LEARNING TOOLKITS AND COMPUTATIONAL PLATFORMS

This section gives a brief overview of the tools and software platforms available which a practitioner of Machine Learning can use to solve various problems.

Scikit-learn [21] is a Python library for Machine Learning. It is an open-source product (BSD License) and is based on NumPy and SciPy, two mathematical and data analysis libraries known for their speed and efficiency. The scikit-learn library is somewhat of a Swiss Army Knife amongst Machine Learning packages. The functionality implemented in scikit-learn includes classification, regression, clustering, dimensionality reduction, model selection, preprocessing, and many other features such as performance metrics.

The scikit-learn library includes almost every well-known classification and regression technique (Naive Bayes, logistic regression, Support Vector Machines (SVMs), lasso regression, etc.), along with some more cutting edge techniques such as neural networks. Users considering scikit-learn should be familiar with the Python programming language and would benefit from some Machine Learning background as well as familiarity with NumPy and SciPy.

Since scikit-learn has been around for nearly a decade, there is a mature user community along with excellent API documentation and many helpful tutorials. If you have an issue with scikit-learn, you can likely find an answer on StackOverflow.

Matlab incorporates a Neural Network Toolbox and a Machine Learning Toolbox[7] that provides the most common features for automatic learning: classification, regression, clustering, etc. The tool is capable of programming neural networks and large networks with deep learning autoencoders. The tool can be accelerated using GPUs and parallel environments and, more importantly, provides out-of-the-box neural networks that are auto-configured by the toolbox. This represents an important advantage for people that are starting to with Machine Learning. Matlab also provides implementations of various ML algorithms, so it is a great starting point to apply many different techniques to a give data set, to quickly analyze the feasibility of various algorithms.

Theano [8] is an open-source deep learning framework in the form of a Python library that lets you define, optimize and evaluate mathematical expressions. Theano has its roots at the MILA lab at University of Montreal and allows you to define mathematical expressions (typically multi-dimensional matrix calculations) that are then compiled to run efficiently on either CPU or GPU architectures [15],[17]. Theano is deeply optimized and has many applications in Machine Learning; however, as opposed to Matlab, it does not provide any sort of auto-configuration.

Pylearn2 [9] is an open-source Machine Learning library toolbox for easy scientific experimentation. Pylearn2 is built on top of Theano. With Pylearn2, one can easily define learning algorithms (e.g., neural networks) that will be then computed and optimized by Theano. As such, Pylearn2 may be seen as a Machine Learning front-end for Theano that eases experimentation.

TensorFlow is a data flow based programming model/interface for expressing and implementing Machine Learning algorithms, recently open sourced by Google. A computation expressed in TensorFlow can be executed on a wide variety of platforms ranging from handheld devices to GPU cards to large distributed systems. The flexible system can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models. TensorFlow provides a Python API. It has been used for conducting research and for deploying Machine Learning systems into production, mainly Google's products. [18]

Torch is an open source scientific computing framework that supports a wide variety of Machine Learning algorithms, with a focus on optimizing for GPU. Torch uses the Lua programming language in a scriptable form (by way of LuaJIT) and an underlying C/CUDA implementation. Torch is used by Google DeepMind and Facebook (FAIR) among others, and is supported on Linux, MacOS, Android and iOS. A large number of extensions, referred to as packages, have been developed and contributed to the community, providing support for solving a wide variety of Machine Learning problems. [16], [29]

Microsoft has created a toolkit to enable development of Machine Learning algorithms for big data, tasks that are too large for a single machine, and require coordinated processing across a cluster of machines. Their Distributed Machine Learning Toolkit is open source, and precompiled binaries are available for Windows and Linux. The goal of DMTK is to handle the complexities of interprocess communication, distributed storage, thread management, etc. so that the researcher can focus on building their model. [30]

The interested reader can find a performance comparison among several tools in [16], [17], [18].

APPLICATIONS OF INTEREST TO CABLE

This section describes various problems in the cable network space, each with large sets of data, to which various Machine Learning techniques can be used to solve various problems.

Proactive Network Maintenance

Proactive Network Maintenance (PNM) involves processing vast amounts of fine-grained information about the state of the network to look for patterns that indicate impending problems. Current PNM solutions have focused on leveraging the physical-layer channel measurements performed by DOCSIS cable modems and CMTSs, along with knowledge of the HFC plant topology and well-known digital signal processing algorithms in order to locate physical degradations of the plant. ML algorithms could open up this space to identify a much wider variety of network impairments, particularly when those impairments aren't predictable using a priori knowledge.

CMTS and cable modem features and capabilities can be leveraged to enable measurement and reporting of network conditions so that undesired impacts like plant equipment and cable faults, interference from other systems, and ingress can be detected and measured. With this information, cable network operations personnel can make modifications necessary to improve conditions and monitor network trends to detect when network improvements are needed [3]. DOCSIS Downstream PNM Measurements and Data include: Symbol Capture, Wideband Spectrum Analysis, Noise Power Ratio (NPR) Measurement, Channel Estimate Coefficients, Constellation Display, Receive Modulation Error Ratio (RxMER) Per Subcarrier, FEC Statistics, Histogram, and Received Power. DOCSIS Upstream PNM Measurements and Data include: Capture for Active and Quiet Probe, Triggered Spectrum Analysis, Impulse Noise Statistics, Equalizer Coefficients, FEC Statistics, Histogram, Channel Power, and Receive Modulation Error Ratio (RxMER) Per Subcarrier.

Some of the above measurements are very valuable in that they can reveal problems in the network. We illustrate a couple of examples below to which we can effectively apply Machine Learning techniques. This will help in quickly and automatically identifying problems, instead of a human operator looking through the data and flagging issues manually.

PNM : Upstream equalization

The upstream pre-equalization mechanism relies on the interactions of the DOCSIS ranging process in order to determine and adjust the CM pre-equalization coefficients. These coefficients can be read to reveal problems, such as echo tunnels and excessive group delay. Echo tunnels make reflections that create ripples in the frequency domain. Group delay is typically caused by filters.

The set of coefficients on each modem in the plant can be monitored over time, and a Machine Learning algorithms can spot patterns in the changes, for a modem or a group of modems. Support Vector Machine or other classification algorithms such as K-nearest neighbors can help classify in trouble CMs immediately and help flag the operator long before a customer actually sees and reports an issue.

PNM : Downstream spectral analysis

A CM with full-band tuner capability acts like a spectrum analyzer. The spectrum analysis reveals suck-outs, excessive tilt, frequency peaking, unwanted filters, FM (or other) radio ingress, early roll-off, etc.

Each of these issues (signal patterns) can be learnt by a Machine Learning application which can then monitor live signals to flag problems in the network.

Machine Learning algorithms can look for abnormalities from the data obtained from each CM and across all CMs in the network. Grouping of CMs with common problems can be used to identify problems affecting multiple subscribers and isolate the cause of an issue in the cable plant.

Predictive Video Channel Lineups/ DVR Recording

Applications around the video delivery product could enable more efficient network utilization as well as better engagement from customers. Using Machine Learning in order to understand user habits, and to provide recommendations, is almost a cliché in the ML world. Any industry that has both a large catalog of assets and a large user base is in possession of a huge resource that can be automatically mined by a ML algorithm in order to provide recommendations.

In the case of video distribution, the ability to accurately identify video assets that might appeal to a viewer or set of viewers could have multiple applications. Some of the more obvious applications are VOD suggestions to customers. This is similar to the Netflix recommendation engine, see the Netflix Prize [28]. This is essentially a clustering and classification Machine Learning problem.

One could take this a step further with predictive DVR recordings for a customer based on content which is trending across the user base and intersecting that with the individual custromers preferences. Another idea is to pre-position the most popular content in CDN caches or in the local DVR storage. The operator could push trending content ahead of time, during off peak hours to reduce congestion during peak times.

One could also use such a recommendation system to improve the network traffic load and utilization. This problem can be described as the selection of video programs/channels for IP multicasting for a linear TV lineup. In different service groups on a cable plant, the viewership of the most popular channels will differ based on the demographics. So instead of making canned decisions on the channel lineup, and statically deciding which programs/channels make the cut for IP multicasting, an operator could use Machine Learning algorithms to understand the usage patterns over time and make recommendations to the operators on which channels to add to the multicast lineup dynamically.

ML algorithms which learn from video viewer data can do a much better job at analyzing which are the top programs in an automatic and real-time fashion. This learning could also apply in a weekly/daily timeframe or even in real-time, where say certain programs or channels are popular at certain days or times and the ML engine could automatically push those programs to be part of the multicast lineup at the appropriate times (e.g., ‘Saturday Night Live’ on Saturday nights). The benefit of this is the conservation of access network bandwidth in a IPTV deployment by intelligently planning the content on the network.

Profile Management

DOCSIS 3.1 introduces the concept of modulation profiles for OFDM channels. A modulation profile defines the modulation order to be used for each subcarrier within a channel. The CMTS can define multiple modulation profiles for use on a channel, where the profiles differ in the modulation orders assigned to each subcarrier. The CMTS can then assign each CM to the modulation profile that best suits it, taking into account the characteristics of the channel between the CMTS and that CM. Determining the best set

of modulation profiles to use on a channel is complex, given the number of CMs (and subcarriers) and the differences in signal quality that they experience on the plant. A Profile Management Application can help operators determine what the best modulation profiles are for each channel, given the channel characteristics seen by each CM on the network. The goal of optimizing profiles is to increase throughput per CM, maximize network capacity, and minimize errors.

Given a set of CMs using a DOCSIS 3.1 OFDM channel, the problem is to find a specific set of profiles which maximize the network capacity by assigning each CM to the best profile it can handle with a limitation on the number of profiles to what a system can support.

The input data set is the MER data per subcarrier for all the CMs. This can be directly translated into a bit-loading number for each subcarrier. So for example, an OFDM downstream channel with a 4K FFT has 3800 active subcarriers, and 200 CMs are using that channel. That means the input data set is a 200x3800 matrix of bit loading capabilities. Machine Learning clustering algorithms (like K-means) could help identify clusters of CMs which have similar bit loading capabilities across the whole channel. For each of these clusters a profile definition can be easily built by using the minimum bit loading in all the subcarrier dimensions, for that cluster of CMs. The set of profiles thus produced can be optimized further, balancing the number of profiles with the cost of having too many profiles.

Network Health Patterns

The access networks deployed by cable operators often experience abnormal behavior. Machine Learning techniques help create anomaly detection algorithms that are adaptive to changes in the characteristics of normal behavior in the network. Patterns in network traffic, equipment failures, device reboots or link failures can be used to identify the root cause of various network problems. Some examples are:

- CM/STB reboots after certain uptime.
- CM performance degradation compared to neighbors.
- IPv6 session failures on a certain SW version of the CM.
- CM performance as it relates to the time of day.
- Denial-of-service attacks on the plant.
- Correlation of customer WiFi issues to channel settings, etc.

For the example of a cable modem or a TV Set-top box rebooting in the network, the input learning parameters/features could be device uptime, software/hardware version, number of packets processed through the device, or the type of services running on a device. Some useful indicators available in the network are CM status, Upstream Transmit Level, Upstream Receive Level, Upstream SNR (MER), Upstream Codeword Error Rate, Downstream Receive Level, Downstream SNR (MER), DS Codeword Error Rate, etc. Each of these data points for a CM could reveal health issues at a given time. Over time an application could gather all the relevant data and supply it to an ML algorithm to figure out any patterns. This would be an unsupervised learning problem, where the idea would be to identify which of the features contribute towards a catastrophic reboot of a device. Principal component analysis could help in identifying the factors which contribute most towards a failure.

A similar approach would work for another class of problems debugging connectivity issues, this example is around IPv6 connectivity in an operator's network. A large population of devices in the network were experiencing IPv6 connectivity issues, which the operator was not aware of and only came to realize based on some IPv6 testing by a

third party. After manually tracing the broken IPv6 prefixes to a set of CMs, they realized that most of the failing modems were from one manufacturer and a specific software version. This type of debugging would be an ideal unsupervised learning problem, where a Machine Learning application could be running various network health tests and tracking issues or problems over time to identify problems and bubble them up the operator, before they become a major issue with the consumers or with the press.

Internet Traffic Classification

Traffic classification is the ability to understand the types of traffic flows which pass through the network and being able to identify the normal and abnormal patterns in those flows. It enables network operators to better manage their network resources, e.g., apply QoS or route traffic appropriately. It also helps with network security for an IPS/IDS (intrusion prevention/detection system) by finding abnormal patterns of traffic, and flagging a denial of service (DoS) or other attacks.

For every set of flows in the network, an operator wants to understand the patterns represented, as it will help make informed decisions or answer a particular question. For example, observing and learning from 5-tuple IPs flow (IP source/destination, Port Source Destination, Protocol) will help answer the question of which applications are generating a specific flow: Skype vs YouTube vs bitTorrent? For security applications, another pattern to learn from could be the aggregated traffic directed to the same IP destination in a certain timeframe. Given some well-defined notion of normal patterns of traffic, the goal would be to predict if the patterns from a new flow are markedly different.

Various supervised and unsupervised Machine Learning techniques can be applied to classify internet traffic. Traditionally port-based techniques are used to classify traffic, this is based on knowing the port numbers for the applications, but many new applications use random port numbers which makes this method of classification hard. Payload-based techniques are used by Deep Packet inspection engines which match the payload of the traffic based on well-known signatures. This method becomes hard with encrypted traffic. Machine Learning techniques such as classification and clustering can help solve this in an elegant fashion.

See [25], [26], [27] for examples of how to use ML techniques to the classification problem.

Network Traffic Engineering

Reacting to changes in network load is an important task for a network operator, since it is not cost-effective to deploy network equipment in excess of demand, while on the other hand, it is unacceptable from a user-experience perspective to run networks well into saturation. Properly done, tracking and responding to network utilization both for long-term capacity planning and for short-term traffic engineering purposes can result in well-functioning networks that operate cost efficiently.

ML techniques can bring some new tools to bear on this problem, with the result being quicker reaction to sudden changes in traffic flows, and the possibility – longer term – of these reactions being put into place automatically to ensure that networking resources are put to their best use.

A ML application could track utilization of various links, the byte counts of data traversing an interface, dropped packets at an interface etc., to get an understanding of the normal operation of the network elements. This by itself can reveal patterns in network utilization within a day-to-day time frame and also expose patterns for the longer term. Any

deviations from the norm can be automatically flagged for the operator. In the case where the application is tied into a SDN controller or orchestrator which can effect changes in the network, the application could actually create new paths or bring up new virtual routers etc., to handle the abnormal situation, for example, a temporary overload in data traffic.

Customer Churn Prediction

Given that acquiring new customers is generally three times the cost of keeping existing customers, an analysis of churners and an ability to identify potential churners can be very valuable. The task of churn prediction can be thought of as either a classification problem (classify a customer as a churner or non-churner) or a regression problem (calculate the likelihood that a customer will churn). Usually churn prediction is a classification task, and in particular, a binary classification task (i.e., customers are predicted to either churn or not churn in the next billing cycle). Multi-class classification is another possible approach. In this setting, customers could, for example, be predicted as belonging to either a low-risk, medium-risk, or high-risk class.

In the context of cable, a cable operator might want to identify cord shavers: subscribers with both broadband and video subscriptions who drop their video for over-the-top services such as Netflix, Hulu Plus, Amazon Instant Video, etc. For such a task, data would be required from several different sources. Linear viewing data would be useful as this would identify how much video a customer watches, what channels they generally watch, and what times they tend to watch. These features might be predictive of a cord shaver, as a customer that has a high-degree of interest in live sports may be less likely to drop his or her video service. Deep packet inspection (DPI) could also yield some useful features such as the percentage of bandwidth that streaming services like Netflix consume. Customer billing data would also be useful. For example, a customer who had subscribed to a promotional package for video and broadband might be more likely to drop video service once the promotional period ends. Combining the linear video, DPI, and customer billing record features would likely provide the most value. Some classification techniques that have proven successful in churn prediction include ensemble methods, SVMs, logistic regression, and random forests [19], [22]. There are also some indications that deep learning can yield some promising results in churn prediction [24].

One challenge with churn prediction is that churners tend to make up a small percentage of customers in any given billing period. For example, suppose that for a certain month, 2% of customers drop their video service. A classifier trained on this data would see 98% of the examples as non-churners. Some classifiers, such as Naive Bayes, use the class prior probability in predictions that are made. This would bias the classifier toward the majority class. Some techniques for addressing such class imbalances include undersampling the majority class (using fewer examples), oversampling the minority class (possibly by creating synthetic examples), and weighting the minority class as more important in the classifier being used [20], [22].

SDN Routing

Optimal or quasi-optimal routing has been a well-known challenge in computer networks. In routing, typically the objectives are to configure the routing policies in such a way that fulfill the requirements of the flows or that maximize the minimum link utilization. This area has been strongly limited by the fact that networks are inherently distributed systems where each node has a partial view and control over the network. In addition, the traditional destination-based

routing limits routing granularity and hence, the performance of the solution itself.

With the rise of the Software-Defined Networking (SDN) paradigm, routing becomes a centralized problem where the logically centralized controller receives all the information from routers and switches, computes an optimal solution and provisions the network equipment with the appropriate routing policies. In addition SDN also provides flow-based routing granularity with technologies such as OpenFlow [10] or LISP [11]. A notable example of this is B4, Google's SDN network [12].

In order to compute such algorithm, the SDN controller requires a model of the network, such model can be either an analytical or a computational model. In both cases the model may be hard to obtain or incorporate inaccuracies that usually arise when modeling real systems. In this context Machine Learning techniques can represent a solution for such inaccuracies.

Indeed, Machine Learning (ML) algorithms using a supervised approach can be trained with the monitoring data from the network. Specifically, the ML algorithm (see Figure 2 below) can be trained with the routing configuration, the load of the network (e.g., traffic matrix) and the resulting performance (e.g., delay or link utilization). With this, the ML algorithm is learning the function that relates routing, load with the resulting performance: function(routing configuration, load) = performance. Please note that this function is a model of the network and, if the dataset is large enough, an accurate one that can take into account not just common network behavior such as queuing but also complex ones such as the delay introduced by hardware, etc.

This model can be then explored online by the SDN controller using a traditional optimization algorithm to compute the optimal solution. For instance, the SDN controller may search which is the optimal routing configuration for a particular objective (e.g., delay) taking into account that the network is loaded with a particular set of traffic.

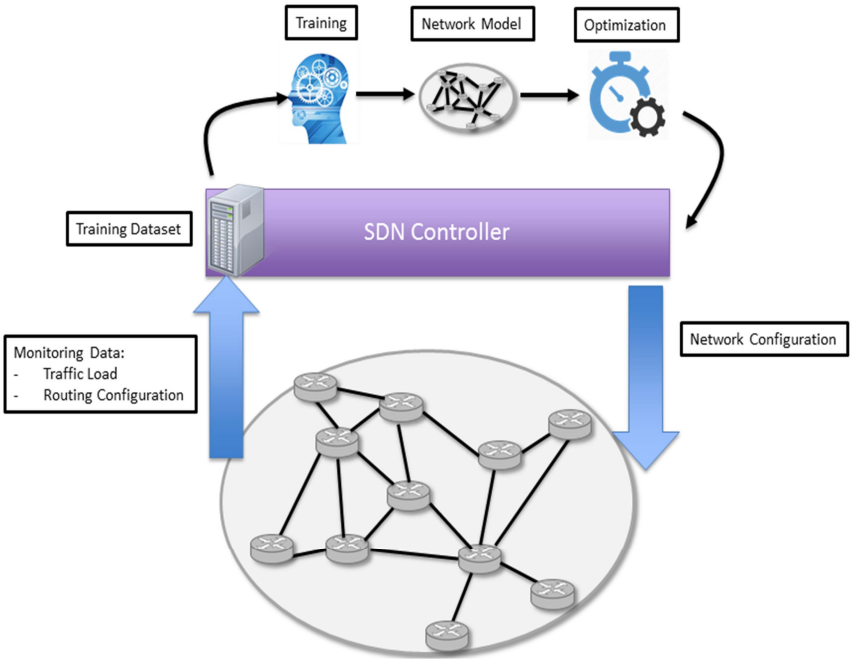

Figure 2. A Machine Learning-enabled SDN routing algorithm

NFV/SFC: Allocating VNFs to Appropriate VMs

Network Function Virtualization (NFV) [13] is a networking paradigm where network functions (e.g., firewalls, load-balancers, etc.) no longer require specific hardware appliances and instead are implemented in the form of Virtual Network Functions (VNFs) that run on top of general purpose hardware. Service function chaining (SFC) defines and instantiates an ordered list of instances of such functions, and the steering of traffic flows through those functions.

The resource management in NFV/SFC scenarios is a complex problem since VNF placement may have an important impact on the overall system performance. The problem of optimal Virtual Machine (VM) placement has been widely studied for Data Centers (DC) scenarios (see [14] and the references therein). However, in DC scenarios the network topology is mostly static, while in NFV scenarios the placement of a VNF modifies the performance of the virtualized network. This increases the complexity of the optimal placement of VNFs in NFV deployments.

In the VNF placement problem all the information is available, e.g., virtual network topology, CPU/memory usage, energy consumption, VNF implementation, traffic characteristics, current configuration, etc. However, in this case the challenge is not the lack of information but rather its complexity. The behavior of VNFs depends on many different factors and thus it is challenging developing accurate models.

In this context, some challenges that NFV resource allocation presents can be addressed by Machine Learning algorithms. Indeed, the SDN controller can characterize, via ML techniques, the behavior of a VNF as a function of the analytics data collected, such as the traffic processed by the VNF or the configuration pushed by the controller. With this model, the resource requirements of a VNF can be modeled by machine learning without having to modify the network. This is helpful to optimize the placement of this VNF, and therefore, to optimize the performance of the overall network.

In this case the scenario works as follows: the SDN controller receives a query by the user/owner to run a particular VNF, which is a black box from the SDN controller point of view. The controller, (e.g., via OpenStack) launches the new VNF and starts monitoring it, specifically it monitors the traffic being consumed by the VNF (e.g., by means of traffic features such as number of flows, distribution of the inter-arrival time, etc.) and a performance parameter such as delay or CPU consumption. With this dataset, the SDN controller trains a ML algorithm with the objective of learning the function(traffic features)=performance. Once the VNF has been characterized, the model can be used by optimization algorithms to optimally place the VNF and/or to provide delay guarantees of the virtual network.

BEST PRACTICES, GUIDELINES, CHALLENGES

This section documents some of the points to be aware of when framing and solving a problem using Machine Learning techniques. For a more detailed review please see references: [1], [2]

Clear Problem Definition

Before starting to solve a problem using any Machine Learning techniques, it is important to define the aim of the Machine Learning. Is the problem one of data distribution, identifying patterns, or making decisions? Is it a regression problem or a classification probem? What are the assumptions and boundaries? How would the

problem be solved manually? These questions help define a problem accurately.

Objective functions

Another important decision is to define how to evaluate the results of the Machine Learning algorithms. What is the metric or objective function which needs to be maximized or minimized. This will help compare learning algorithms and determine how much better it does as compared to simple or random predictions.

Proper Features:

Lots of thought needs to be given to the decision of how to represent the data available. There may be a set of original features which are relevant and other artificial features can be developed as well. Data and feature definition is an essential part of system design.

Suitable Methods and Algorithms

The main guideline to remember is to try out different ML techniques and algorithms. There are many effective methods and algorithms and no algorithms which work perfectly for certain applications/use cases. As the training set increases, it reduces the effect of algorithm selection.

Adequate Representative Data

The learning data needs to represent realistically the space; this is important to make sure the ML algorithm understands the whole range of inputs and outputs. If not, the predictions will be skewed. Random re-sampling of the data is preferred, and data cleaning will be necessary in many cases. Another guideline is that the training samples are expected to be much more in number than the testing samples and samples to be predicted.

Training data Testing and Cross Validation

With any given dataset of training examples, if you use the entire data set to train your model, the final model will normally overfit the training data and the error rate estimate will be overly optimistic. A better approach is to split the training data into disjoint subsets: a training set (e.g., 80%) and a test set (e.g., 20 %). The procedure is to run the algorithm on the training set, minimizing the error of the learning function. Then the idea is to compute the error on the test set to figure out how good the model is. This training vs test split has issues: if the training data is a sparse dataset, one may not afford setting aside a portion of the dataset for testing, and also if the split happens to be not representative of the data set, the model learnt will be off. These limitations can be overcome with resampling methods like cross validation.

Metrics / Evaluation Criteria

When evaluating a classification or regression task, it's important to choose the right metric for the task. Classification is commonly measured by accuracy. Accuracy is defined as the percentage of predicted labels that were correct. However, this metric is not always the best measure of success. For example, in a task with an imbalanced class distribution (such as churn prediction), a very bad classifier can obtain a high accuracy score. Suppose that 98% of the examples in our test set are of one class, *Class A*. A classifier can simply predict *Class A* for every example and obtain an accuracy of 98%. However, in tasks such as churn prediction or fraud detection, identifying the minority class is very important. For this reason, accuracy is not generally a good metric for evaluating the performance of a classifier when dealing with an imbalanced dataset. A better way to view the performance of binary classification would be to examine the number of true positives

(tp), false positives (fp), true negatives (tn), and false negatives (fn). For churn prediction, the class of "churner" would be considered the positive class. So churn prediction success might be measured by Precision, defined as $\frac{tp}{tp + fp}$, or Recall, defined as $\frac{tp}{tp + fn}$ [23]. In general, it's always useful to keep the high-level objective of the task in mind. In tasks such as fraud detection or disease identification, a false negative will likely be considered more costly than a false positive. So in these cases, the classifier would be optimized to minimize false positives.

Evaluating a Hypothesis

A hypothesis may have low error for the training examples but still be inaccurate (because of overfitting). The idea is to troubleshoot errors in predictions by some of the following techniques: increasing the number of training examples, trying a smaller or larger set of features, changing the parameter settings in an algorithm, etc.

Above all are statistically significant results from a given ML technique: It goes without saying that the predictive results of a ML algorithm should be of a much higher possibility than random choices at a very minimum.

CONCLUSIONS

There are very many good use cases within the cable access network for the application of Machine Learning Algorithms. The ultimate goal of applying Machine Learning to networks is to provide automation and remove many of the manual tasks associated to network control and operation. Problems which have large sets of data are tough to solve manually and ML algorithms provide an intelligent way to get knowledge from the data.

This paper describes nine differnet and relevant use cases to which machine learning can be applied today. The application of these will give the operators knowledge about their networks and how to most effectively run the network. ML algorithms will be assimilated into systems which can configure networks, identify issues and take the appropriate corrective action.

There are important open research challenges that need to be addressed before this vision is fully implemented. First, typically ML is applied to scenarios that are resilient to errors, such as image recognition. However, computer networks do not handle errors well. In this context there is a need to understand how such probabilistic Machine Learning techniques can be effectively incorporated to networks. And second, learning requires a representative dataset. But what does representative mean in the context of computer networks? Are observable network loads and configurations (i.e., those that do not break the system) representative enough to provide accurate estimations? Generating a training set that allows the algorithm to learn the boundary between functioning and failing networks can be challenging. At the time of this writing, there is no clear answer to this question and will require further research efforts.

In addition there are also non-technical issues that need to be addressed. Machine Learning builds on top of huge datasets that, in some cases, will contain user information. As a consequence there are privacy issues since there is a potential of leakage of personal information. This will require that the appropriate safety mechanisms are put in place and, if possible, to always learn based on aggregated-information.

As a result, the use-cases that do not directly operate over the network but that rather provide recommendations to the network owner/administrator will be commercialized first. Such systems are safe since they involve human intervention and validation, and will help operators get used to this new technology and better understand its potential.

REFERENCES

[1] Ji Shufan, Case Study: Autonomic Network Configuration Using Machine Learning, https://www.ietf.org/proceedings/94/slides/slides-94-nmlrg-2.pdf

[2] Pedro Domingos, A few useful things to know about Machine Learning, Communications of the ACM, Vol. 55 No. 10, 2012. http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

[3] (DOCSIS PHYv3.1), DOCSIS 3.1, Physical Layer Specification, CM-SP-PHYv3.1-I08-151210, Cable Television Laboratories, Inc. http://www.cablelabs.com/wp-content/uploads/specdocs/CM-SP-PHYv3.1-I08-151210.pdf

[4] Stanford Professor Dr. Andrew Ng, Machine Learning Course on Coursera https://www.coursera.org/learn/machine-learning

[5] Clark, David D., et al. "A knowledge plane for the internet." Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications. ACM, 2003. http://groups.csail.mit.edu/ana/Publications/PubPDFs/A%20knowlege%20plane%20for%20the%20internet.pdf

[6] Michalski, Ryszard S., Jaime G. Carbonell, and Tom M. Mitchell, eds. Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013.

[7] Matlab's Neural Network Toolbox and Machine Learning toolbox : http://www.mathworks.com/products/neural-network/index.html, http://www.mathworks.com/products/statistics/index.html

[8] Theano:, http://deeplearning.net/software/theano/#

[9] Pylearn2, http://deeplearning.net/software/pylearn2/

[10] McKeown, Nick, et al. "OpenFlow: enabling innovation in campus networks." ACM SIGCOMM Computer Communication Review 38.2 (2008): 69-74.

[11] Farinacci, Dino, Darrel Lewis, David Meyer, and Vince Fuller. "The locator/ID separation protocol (LISP)." (2013). (RFC 6830)

[12] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., ... & Zolla, J. (2013, August). B4: Experience with a globally-deployed software defined WAN. In ACM SIGCOMM Computer Communication Re-view (Vol. 43, No. 4, pp. 3-14). ACM. Chicago

[13] Matias, J., Garay, J., Toledo, N., Unzilla, J., & Jacob, E. (2015). Toward an SDN-enabled NFV architecture. Communications Magazine, IEEE, 53(4), 187-193.

[14] Berral, J. L., Goiri, Í., Nou, R., Julià, F., Guitart, J., Gavaldà, R., & Torres, J. (2010, April). Towards energy-aware scheduling in data centers using machine learning. In Proceedings of the 1st International Conference on energy-Efficient Computing and Networking (pp. 215-224). ACM.

[15] Bergstra, James, et al. "Theano: a CPU and GPU math expression compiler." Proceedings of the Python for scientific computing conference (SciPy). Vol. 4. 2010.

[16] Collobert, Ronan, Koray Kavukcuoglu, and Clément Farabet. "Torch7: A matlab-like environment for machine learning." BigLearn, NIPS Workshop. No. EPFL-CONF-192376. 2011.

[17] Bastien, Frédéric, et al. "Theano: new features and speed improvements." arXiv preprint arXiv:1211.5590 (2012).

[18] TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, http://download.tensorflow.org/paper/whitepaper2015.pdf

[19] "Handling class imbalance in customer churn prediction", J. Burez and D. Van den Poel, Expert Systems with Applications Volume 36, (pp. 4626-4636), (2009)

[20] "SMOTE: Synthetic Minority Over-sampling Technique", Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, W. Philip Kegelmeyer, Journal of Artificial Intelligence Research Volume 16, (pp. 321-357) 2002

[21] scikit-learn- http://scikit-learn.org/stable/

[22] “Telco Churn Prediction with Big Data”, Yiqing Huang, Fangzhou Zhu, Mingxuan Yuan, Ke Deng, Yanhua Li, Bing Ni, Wenyuan Dai, Qiang Yang, Jia Zeng, SIGMOD '15 Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, (pp. 607-618), 2015

[23] Precision and recall definition: https://en.wikipedia.org/wiki/Precision_and_recall

[24] "A New Neural Network Based Customer Profiling Methodology for Churn Prediction", Ashutosh Tiwari, John Hadden, and Chris Turner, "Computational Science and Its Applications -- ICCSA 2010: International Conference, Fukuoka, Japan, March 23-26, 2010, Proceedings, Part IV", (pp. 358-369) (2010) Springer Berlin Heidelberg

[25] A Machine Learning Approach for Efficient Traffic Classification Wei Li and Andrew W. Moore, http://www.cl.cam.ac.uk/~awm22/publications/li2007machine.pdf

[26] An SVM-based machine learning method for accurate internet traffic classification, Ruixi Yuan & Zhu Li & Xiaohong Guan & Li Xu, https://pdfs.semanticscholar.org/556a/209838cf5c742d2f9a75a3436a0de6611526.pdf

[27] Realtime Traffic Classification Based on Semi-supervised Learning, Chengjie GU1, Shunyi ZHANG, Xudong CHEN, Anyuan DU, http://www.jofcis.com/publishedpapers/2011_7_7_2347_2355.pdf

[28] The BellKor Solution to the Netflix Grand Prize, http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

[29] Torch | Scientific computing for LuaJIT, http://torch.ch/

[30] Distributed Machine Learning Toolkit, http://www.dmtk.io/