Predicting Network Behavior Using Machine Learning/AI

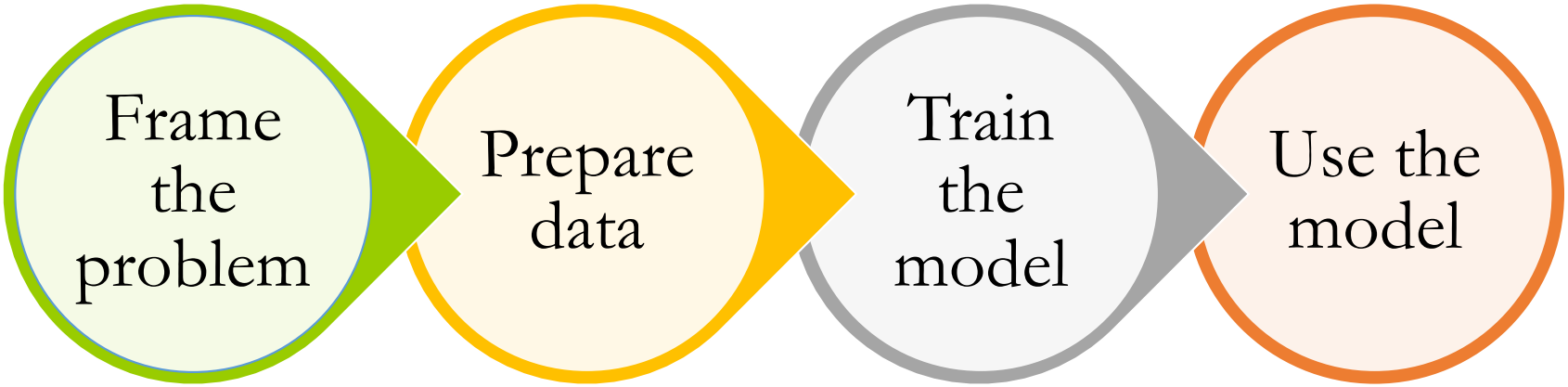
Srividya Iyer

Co-founder and CEO

Why Machine Learning in Networking ?

- Complexity of application and infrastructure requires new approaches in network design, operations and management (encrypted data, cloud infrastructure, IoT to name a few)
- Current solutions - time-series analysis and rule-based heuristic algorithms - have difficulty scaling.
- Varied sources of data and knowledge is difficult to process by humans.
- ML-based methods can achieve higher prediction accuracy by extracting patterns from data across different vantages.

Building a Machine Learning application



Frame the Problem

- ML problems are designed to identify and exploit hidden patterns in data and are categorized as follows:
  - Describe the outcome as a grouping of data (Clustering)
  - Predict the outcome of future events (Classification and Regression)
- The problem you are trying to solve determines
  - Types of data and the amount of data to collect
  - Features to extract
  - ML algorithm to select to train the model
  - Performance metrics used to test the model

Prepare data - Collection

- Offline - Historical data used for analysis and model training.
- Online - Real-time network data to adapt the model for changing network state.
- Types of data for network applications
  - Flow data (Netflow, sFlow, IPFIX)
  - Packet captures
  - Logs (syslog, firewall)
  - Telemetry
  - Device configurations
  - Network Topology

Prepare data – Feature Extraction

- Feature extraction is used to reduce dimensionality and to remove features that are irrelevant or redundant.
  - Packet-level features - statistics of packet size, including mean, root mean square (RMS) and variance, and time series information.
  - Flow level features - flow duration, number of packets per flow, number of bytes per flow
  - Transport level features - Throughput and TCP window size
  - Application specific features (for e.g., DNS, DHCP packet level features)

Train the Model

- Data is decomposed into training and test datasets.
  - Common split is 70/30
  - Need spatial and temporal diversity
- Once an ML model has been built, gauge the performance
  - Reliability, Robustness, Accuracy, and Complexity.
  - Accuracy is the critical metric in networking applications
  - Computes the difference between the actual and predicted values.
- There is no way to distinguish a learning algorithm as the "best"

Traffic Classification

Networking problem	Purpose	Data	Algorithm(s)
Accurately identify applications on the network Current methods – Port based or Payload based	Identify P2P applications Identify unknown applications	Flow data or packet level traces with labeled application classes	Supervised Support Vector Machines Decision Trees Random Forest

Traffic Prediction

Networking application	Purpose	Data	Algorithm(s)
Accurately estimate traffic volume Current methods – Time Series Analysis and Network Tomography	Congestion control Resource allocation	Flow data or packet traces with derived statistics (flow bytes, number of packets)	Supervised Neural Networks

Anomaly Detection

Networking problem	Purpose	Data	Algorithm(s)
Identify anomalous behavior Current methods – Rule and Threshold based, Signature based	Security events (DDoS, Malware)	Flow data (bytes, packets) and Packet headers	Unsupervised K-means clustering

Fault Management

Networking application	Purpose	Data	Algorithm(s)
Accurately predict faults Current methods – Manual detection and mitigation, Rule and Threshold based	Network Troubleshooting (avoid future network failures and performance degradation)	Packet traces Telemetry data Logs	Supervised Decision Trees Random Forest Hybrid approaches of supervised/unsupervised

ML for Networking System Components

Device setup (TAPs, port mirrors, flow monitors etc.)

Open source tools (tshark/tcpdump, nfdump, Elastic/Logstash/Kibana)

Commercial tools

Domain expertise

Python Pandas/Numpy

Algorithm selection

Python scikit-learn

Apache Spark

Deep Learning (Keras/Tensorflow/pytorch)

Features (Flatfiles, Keystore)

Machine Learning models (document/object storage)

Reporting (SQL)

Case Study – Traffic Classification

- Type of Network: Enterprise networks with about 250 to 500 devices.
- Problem: Port based identification of NetFlow traffic identified about 70% of the applications accurately
- Traffic collected – Netflow v9 From 3 different networks for 7 days in 30 minute intervals using nfdump.
- Data was sampled from different time frame to about 300,000 samples for training at 70/30 split.
- Algorithms tested: Support Vector Machine, Decision Trees and Random Forest using python scikit-learn

Case Study – Traffic Classification - Features

protocol	src_port	dst_port	packets	bytes	bpp	flows	Application
TCP	52948	443	10	2208	220	2	https
TCP	443	61299	1	87	87	1	https
TCP	443	54706	43	44969	1045	1	https
TCP	57144	443	2	80	40	1	https
TCP	443	55116	19	10187	536	1	https
TCP	57465	49156	11	3380	307	2	
UDP	51897	12222	4	676	169	1	
TCP	10123	52882	1	40	40	1	
TCP	10123	60189	1	40	40	1	
UDP	12222	39338	19	2167	114	2	

Case Study – Traffic Classification - Results

Application labels		precision	recall	F-1 score
https	95647	0.95	0.97	0.96
domain	36969	0.96	1.00	0.98
http	28112	1.00	1.00	1.00
microsoft-ds	7554	1.00	0.67	0.80

Total Sample: 300000
Labeled set: 200000 (Port based)

Recall = True positive/(True positive + False negative)
Precision = True Positive/(True positive + False negative)
F-1 score = 2 * (Precision*recall)/(Precision + recall)

Case Study – Traffic Classification - Conclusion

Algorithm Accuracy	Train set	Test set
Decision Tree	1.00	0.97
Random Forest	1.00	0.98
Support Vector Machines	1.00	0.72

- Training set sample size matters. There is an optimal sample size beyond which there is no improvement. The quality of data decides the accuracy.
- Different algorithms provide different results
- Additional data from augmented flows might provide additional insight, but they are not standardized.

Case Study – Fault detection

- Type of Network: Enterprise networks with about 250 to 500 devices.
- Problem: Accurately identify the types of faults and performance issues flying under the radar. Traditional fault management is reactive and involves detection, localization and mitigation after it has happened.
- Traffic collected – Packet captures from 3 different networks for 7 days in 30 minute intervals using tshark.
- Data was sampled from different time frames to about 800,000 samples for training at 70/30 split.
- Algorithms tested: Decision Trees and Random Forest using python scikit-learn

Case Study – Fault detection- Features

Line Load (%)	TCP Retransmission Rate	TCP Duplicated Ack Rate	TCP RTT Avg	TCP Session Duration Avg	SYN Packet Rate
0.931813	0	0	0.019979	0.38221192	4.576789
0.093415	3.55227	3.55227023	0.079306	0.28151011	5.556439
0.061109	0	0	0.114893	0.22980118	6.576712
0.269735	3.530648	3.53064808	0.10618	0.28323412	6.129875
0.173596	0	0	0.070752	0.14327288	3.570634
0.048848	0	0	0.080547	0.30491614	6.570694
0.038728	0	0	0.000809	0.00082397	6.190834
0.071423	0	0	0.206471	0.20647097	3.418954

These values were derived for multiple TCP streams
This list of features are not exhaustive just illustrative

Case Study – Fault detection- Results

Dataset Size	Categories Identified	GeneralizationError	f1 score
100	4	0.140762	0.798643
3789	10	0.00368	0.996077
7477	10	0.003193	0.996755
36987	11	0.000812	0.999168
40676	11	0.00092	0.999058
44365	11	0.001082	0.998903
48054	11	0.000974	0.999009
51742	11	0.000379	0.999622
55431	11	0.000379	0.999626

Generalization error is a measure of how accurately an algorithm is able to predict the labels for unseen data
Two important factors determine generalization ability:

- Model complexity
- Training data size

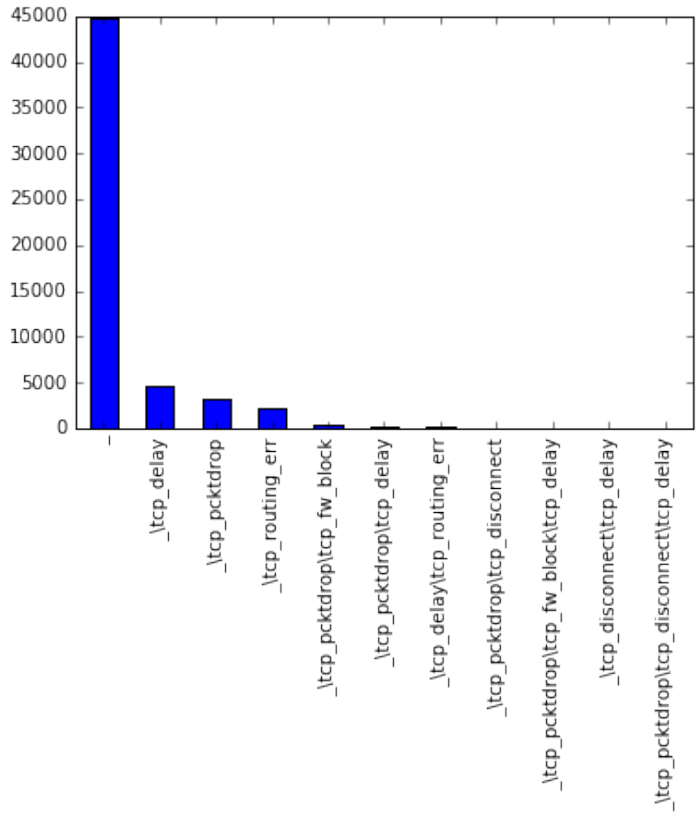
Case Study – Fault Detection - Conclusion

For training, the number of "normal" vs "fault" samples have to be balanced

Packet traces only gives you network faults. Log data needed to correctly identify server faults.

Control plane telemetry might be useful to provide additional context but could not be collected efficiently to test it.

Both Decision Tree and Random forest provided similar results.



Challenges - Machine Learning in Networking

- How to collect the right data and extract features ?
  - Data is inconsistent and messy
- How to choose the right machine learning method for a specific networking problem?
  - There are many ways to approach the traffic prediction, classification and detection problems.
- How to make the solution scalable to large and diverse networks?
- How to make the ML models learn uniformly across non-uniformly designed networks?

Future Trends ?

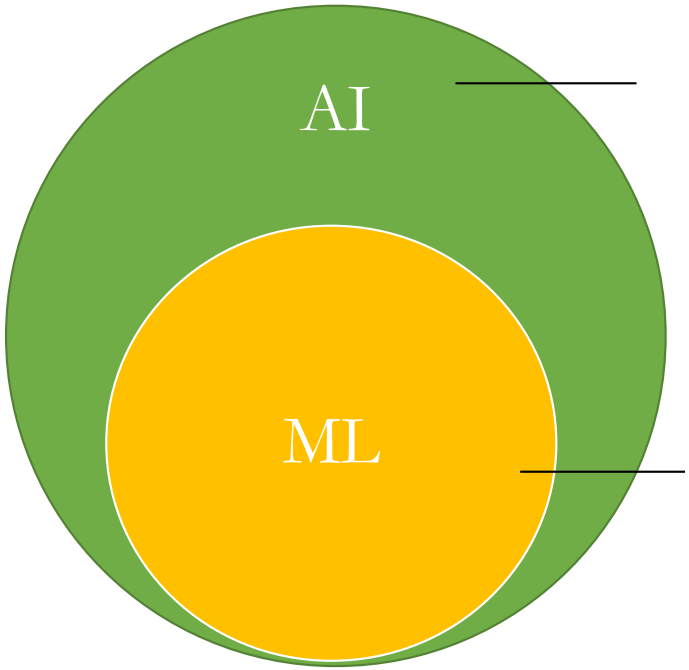
- Data collection
  - SDN/NFV for centralized data collection
  - Open datasets
- Feature extraction
  - Standardization of training datasets and features for common networking applications
- Model
  - The robustness of machine learning algorithms
  - Automatic selection of the right algorithm for the right problem.
  - Combining ground truths/models from several vantage points

Questions ?

Srividya Iyer
siyer@caniv-tech.com
@siyeratwork

Background Slides

Artificial Intelligence vs Machine Learning



AI is the ability to make intelligent machines, that can perceive and act to satisfy some objective, often without being explicitly programmed how to do so.

ML is a subset of AI where computers can learn from real-world examples of data. They apply what they've learnt to new situations – just like humans do.

Machine Learning Terminology

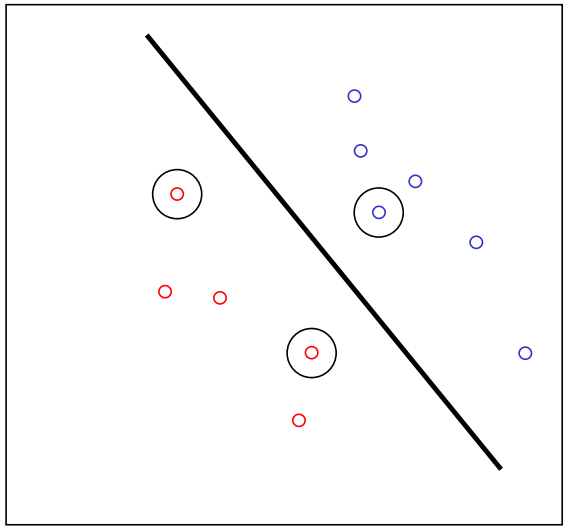
- Training data – Used by the learning algorithm to "learn" the patterns in data
- Learning Algorithm – Used to train the model to learn from data.
  - Supervised - A method where training data includes both the input (features) and the target result (label). There is a label associated to each class and the model is trained using the label.
  - Unsupervised – A method where training data only includes the input(features). The target result is inferred.
- ML Model - Trained using one or more Machine Learning algorithms with known data to predict unknown events.

Machine Learning Algorithms

- Supervised Learning
  - Support Vector Machines
  - Artificial Neural Networks
  - Decision Trees
  - Random Forest
- Unsupervised
  - K-Means Clustering

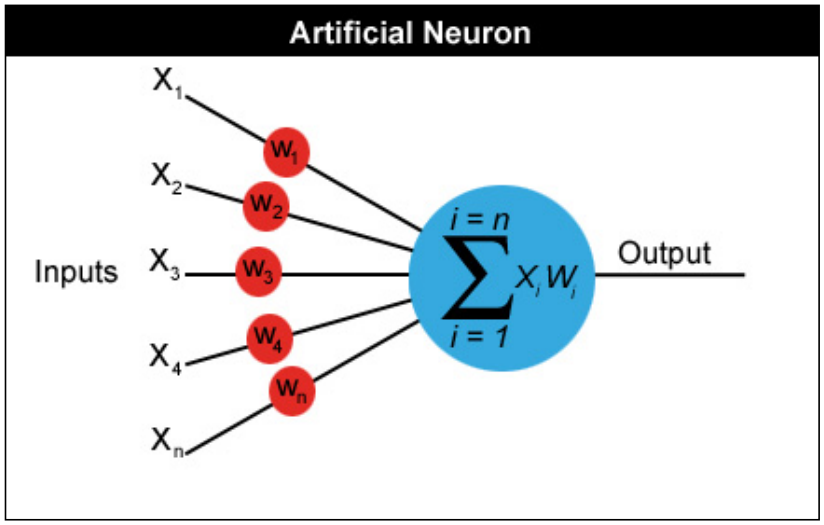
ML Algorithms – Support Vector Machines

- In this example, we are classifying red and blue, based on a given set of features *(x,y)*.
- SVM takes these data points and outputs the hyperplane that best separates red and blue. This line is the decision boundary: anything that falls to one side of it will be blue and anything that falls to the other as red.
- Support vectors (indicated by circles) are the data points that lie closest to the decision boundary (or hyperplane)



ML Algorithms – Neural Networks

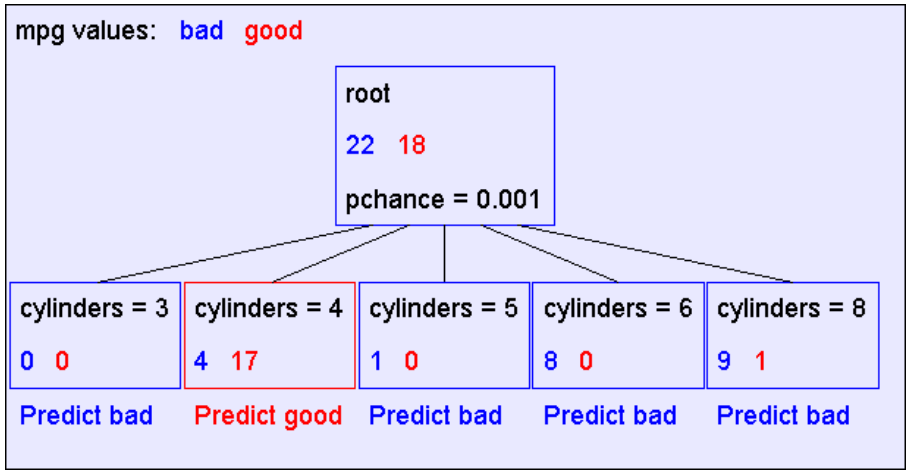
- Neural networks are made up of many artificial neurons.
- Each input into the neuron has its own weight associated with it
- The inputs may be represented as *x1, x2, x3… xn.*
- And the corresponding weights for the inputs as *w1, w2, w3… wn.*
- Output $a = x1w1+x2w2+x3w3... +xnwn$



Source:
web2.utc.edu/~djy471/documents/b2.2.MLP.ppt

ML Algorithms – Decision Trees

- A decision tree maps the input to the decision value by performing a sequence of tests on the different feature values.
- For a given data instance, each internal node of the tree tests a single feature value to select one of its child nodes.
- This process continues till leaf node is reached which assigns the final decision value to the instance.



Source: www.cs.cmu.edu/~awm

ML Algorithms – Random Forest

- Random forest is an ensemble classifier that consists of many decision trees.
- Ensemble methods use multiple learning models to gain better predictive results.
- In Random forest, the model creates an entire forest of random uncorrelated decision trees to arrive at the best possible answer.
  - Bagging: Train learners in parallel on different samples of the data, then combine the results.
  - Boosting: Train learners on the filtered output from other learners.

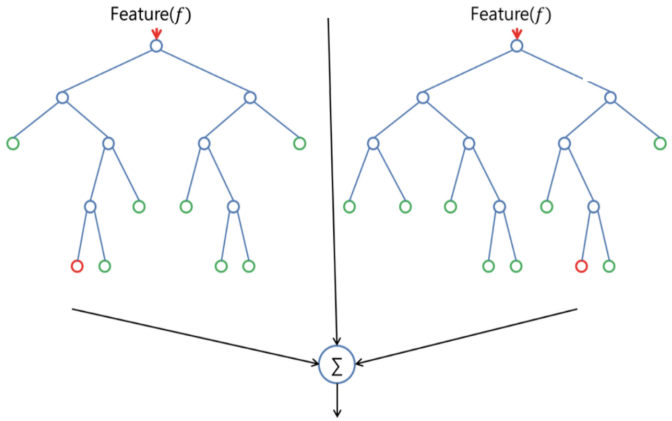
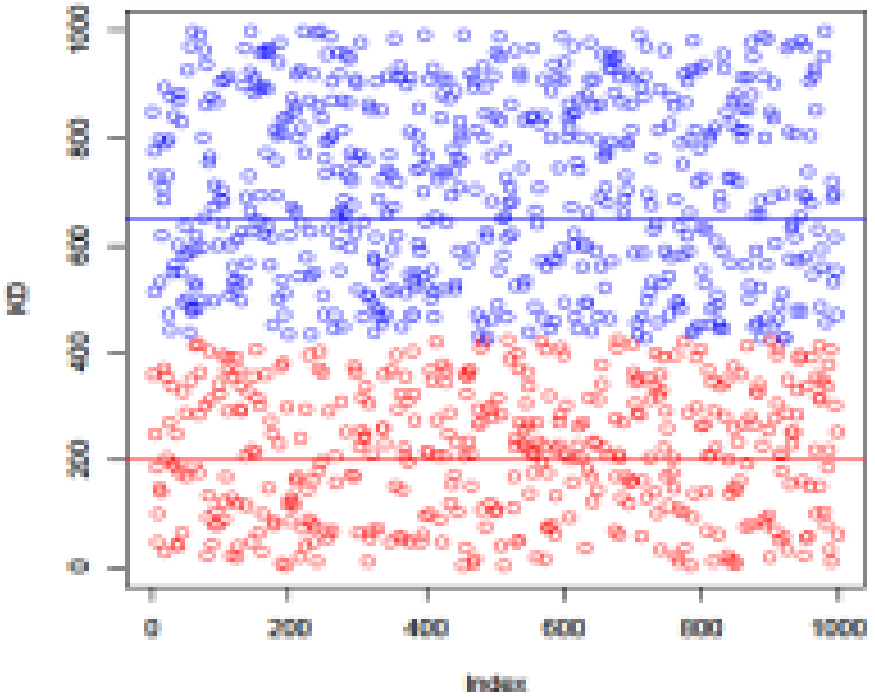


Diagram Source: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

ML Algorithms – K-Means Clustering

- One method to train a data set without labels is to find groups of data which are similar to one another called clusters.
- K-Means is one of the most popular "clustering" algorithms.
- K-means stores “k” centroids that it uses to define clusters.
- A point is considered to be in a particular cluster if it is closer to that cluster's centroid.
- Clustering results are dependent on the measure of similarity (or distance) between “points” to be clustered



Source: www.cs.cmu.edu/~awm