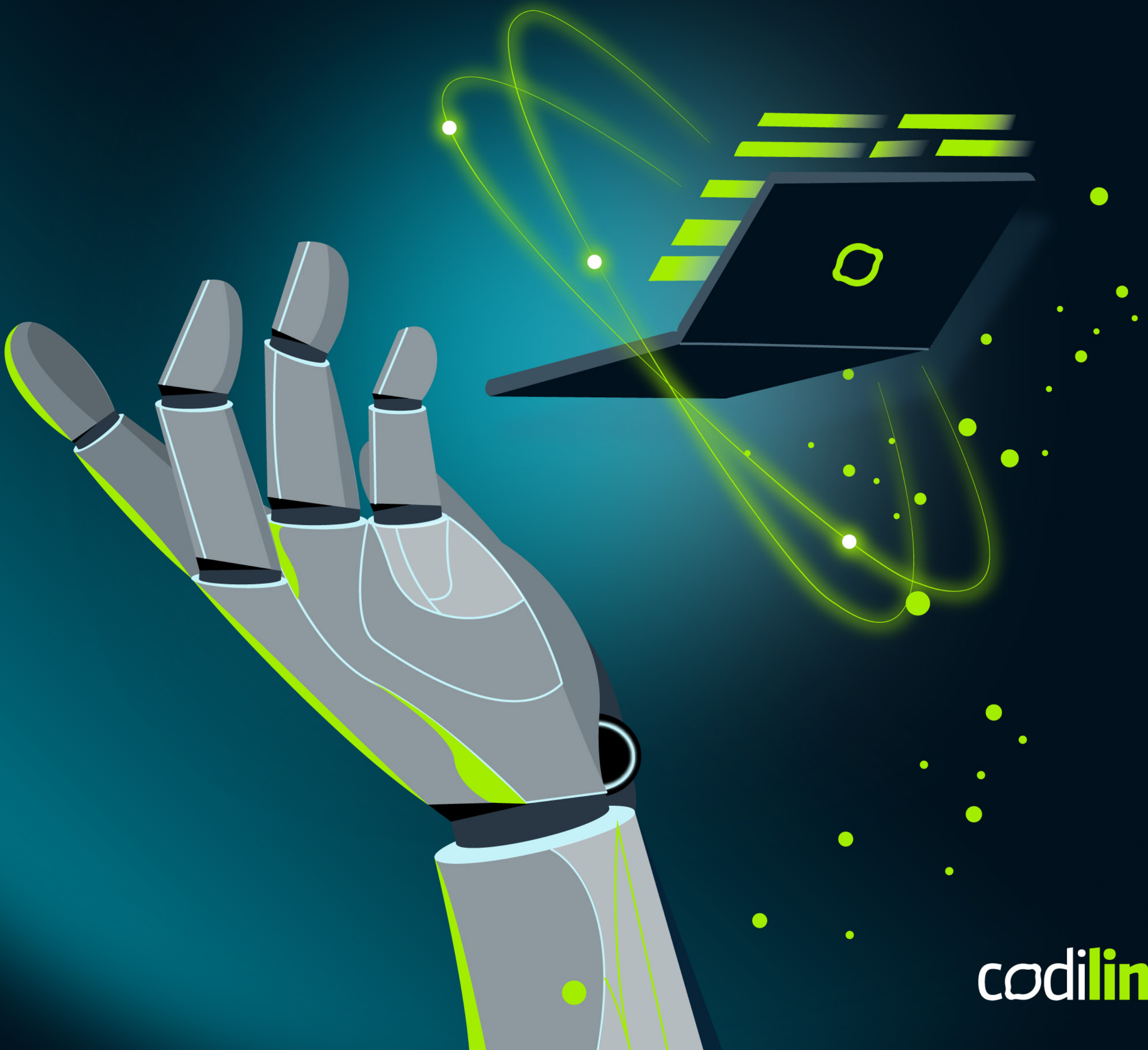
AI and Machine Learning for Networks

Methods and algorithms suited for network issues

- Generative AI-driven solutions to enhance network operations
- Time series forecasting for data measured in networks
- Classification, regression and cluster analysis for network-related data
- Reinforcement learning in the context of network routing optimization
- Anomaly detection for structured and unstructured data



codilime

Executive summary

Publication purpose

This publication contains an overview of the most suitable AI/ML methods and algorithms for networking, considering the network-related data types they work on as well as the specific problem types they can help to solve.

Possible applications

- **Generative AI-based assistants and chatbots** can be used to support various operations like network troubleshooting, anomaly detection, network configuration management, and reporting.
- **Time series forecasting** for network traffic and performance metrics is used for anticipation of future network states and potential problems.
- **Classification algorithms** help distinguish various traffic types, user categories, and alerts with respect to their importance.
- **Regression** models can be useful to find relationships between QoS metrics, QoE indicators, trend analysis of important networks, and data center resource utilization metrics.
- **Cluster analysis** can be used for extracting useful knowledge by grouping similar network-related data into subgroups, and for detection of anomalies.
- **Reinforcement learning** can be applied to optimize routing metrics in a network with link-state routing protocols (OSPF, IS-IS).

Conclusion

As progress in the area of AI applications and in ML methods and algorithms continues, modern companies should build a data-processing culture in this area.

Contents

AI and ML for networks - subject introduction

Artificial intelligence (AI) and machine learning (ML) are trending topics in all technological domains. They offer a rich **set of methods for data processing** that can be used to solve practical problems, including those occurring in networks.

In this publication, we present classes of **AI/ML methods and algorithms** that should play a key role in networking, considering the network-related data types they work on as well as the specific problem types they can help to solve. The emerging concept of **Generative AI** (GenAI), which is attracting increasing attention, can be seamlessly integrated into networks through the deployment of specialized AI assistants. These AI-driven tools can automate, simplify, and expedite various tasks performed by network engineers, enhancing efficiency and accuracy. Fundamental ML techniques such as **regression, classification and time series forecasting** are crucial for processing various network data. Their ability to learn patterns in data and their predictive capabilities allow for a proactive approach in the context of network management and maintenance tasks. **Cluster analysis**, in turn, allows for the extraction of useful patterns and knowledge in network-related data, and grouping similar data into subgroups. We include **anomaly detection** to highlight its importance for processing network-related data, where many types of ML methods can be used. For completeness, we briefly introduce **reinforcement learning**, explaining it in the context of network routing optimization.

Generative AI

Generative AI, or GenAI, is an innovative field within artificial intelligence that enables machines to create new content based on various prompts. GenAI uses complex algorithms and neural networks to generate novel text, images, videos, sounds, code, and 3D designs. This technology learns from vast amounts of data, identifying patterns and using them to produce original content that mimics human creativity.

GenAI is situated at the intersection of deep learning and Natural Language Processing (NLP). Deep learning, a subset of machine learning, involves training neural networks with multiple layers to recognize patterns and make decisions. NLP, on the other hand, focuses on enabling machines to understand, interpret, and generate human language. By combining these fields, GenAI can process large, unlabeled datasets through unsupervised and semi-supervised learning approaches, allowing it to produce human-like text, images, and more.

At the core of GenAI's functionality are AI models, like Large Language Models (LLMs) and multimodal models. They form the foundation of tools like ChatGPT and DALL-E. LLMs are a type of GenAI designed to understand and generate human language. These models are trained on vast amounts of text data, allowing them to understand context, grammar, and meaning to produce text that makes sense and fits the situation. LLMs like GPT-4 can carry out a variety of tasks, from completing sentences to answering questions, to writing detailed articles and having meaningful conversations

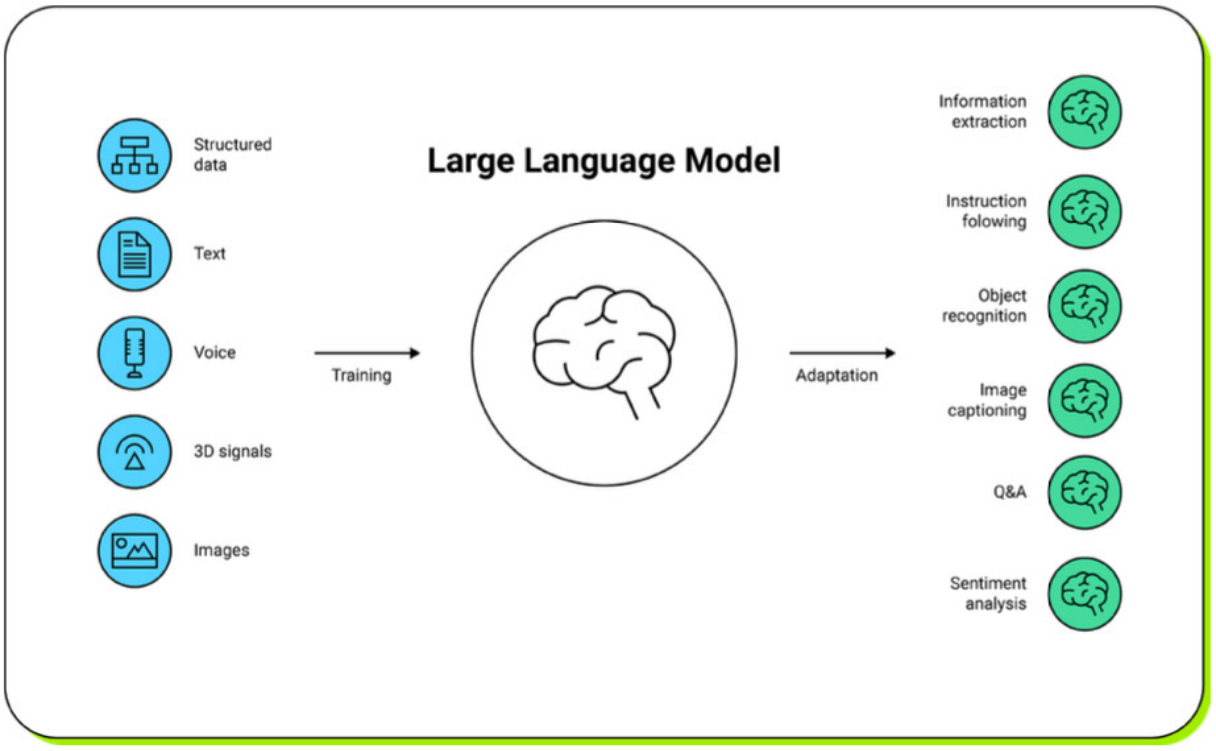


Figure 1. LLM overview

A key technology behind LLMs and many other GenAI models is the transformer. Transformers are a type of neural network architecture that is especially good at understanding and generating sequences of data, like sentences or paragraphs. Unlike previous models, transformers can look at the entire context of a sentence at once, rather than one word at a time. This helps them understand the meaning better and create more accurate and relevant text. Transformers use a mechanism called attention, which lets them focus on different parts of the input data as needed, making them very effective at tasks involving language and other sequential data. This innovation has significantly improved the performance and capabilities of generative AI systems, enabling them to produce high-quality content quickly and efficiently.

Effective GenAI solutions tailored to specific network needs should make use of different sorts of device and system documentation, verified device configuration templates, network policies definitions, troubleshooting scripts, and trouble ticket data as additional knowledge. This allows for rapid extraction and analysis of crucial information related to the given network for which those solutions will operate. In that perspective several use cases can be pointed out:

• **Querying the Current State of the Network**

One key application is querying the current state of the network, such as the status of devices, interfaces, and configurations, based on specific criteria. GenAI can process natural language queries from network administrators, translating them into precise commands that quickly retrieve relevant data. This makes it easier and faster to access network information, without requiring deep expertise in device-specific CLI syntax or the database schemas and API specifications of network management systems.

• **Network Configuration Management**

Another powerful use case is the generation and validation of network configurations. GenAI can automatically generate new configurations or modify existing ones according to network policies and configuration patterns applied for a given network segment. Moreover, GenAI can validate human-made configurations by detecting semantic or syntax errors before deployment, reducing the risk of misconfigurations and ensuring consistency across the network.

• **Network Troubleshooting**

In troubleshooting, GenAI can detect anomalies (errors and mismatches) in device configurations deployed on the network. It can also analyze fresh trouble tickets to make a preliminary indication of possible root cases based on the previously collected trouble ticket data history. Another scenario would be when an AI assistant mimics a network engineer by reproducing a typical sequence of steps of a standard human-made troubleshooting using dedicated commands, scripts and tools when trying to find an issue. This enables faster diagnosis and resolution of network problems, reducing downtime.

• **Generation of Reports and Documentation**

GenAI also excels in generating detailed reports, documentation, and visualizations such as tables and charts with appropriately aggregated data. This capability allows network administrators to automate the creation of comprehensive documentation, saving time and ensuring that records are accurate and up-to-date. The AI can aggregate and summarize large volumes of data, making it easier to track network performance and compliance over time.

GenAI tools and frameworks worth your attention

Below we list the most popular NLP and GenAI tools/frameworks:

- NLTK
- TextBlob
- SpaCy
- GenSim
- Google Cloud Natural Language
- GPT

- LangChain

- LlamaIndex

Time Series Forecasting

We see time series forecasting as the primary set of ML algorithms to consider when addressing various types of network-related problems. Why? Because network and data center monitoring is mainly done by collecting time series data. We are measuring a plethora of important values in time, such as flow bit rate, packet rate, packet delays, packet jitter, packet loss, round trip time, requests rate, throughput, CPU/memory consumption, etc. to name the most basic ones. All those values are measured in consecutive timestamps forming a data structure called a time series. The measured metrics allow us to assess the current state of the working environment, to detect or explain some existing problems and then to take action if needed.

Time series usually have different characteristics in time, of which the most important ones are trend and seasonality. Specific seasonal patterns and trend shapes are subject to automatic recognition by ML models. Based on historical values, such a model can forecast the future values of a specific metric in consecutive timestamps.

Figure 2 shows an ML model trained on historical data that predicts the bit rate of a traffic flow, appropriately reflecting its trend and daily seasonality. You can also notice outliers, i.e. unusual data points that can be reported as anomalies and are usually corrected (before you start training the model) in order to achieve a higher level of prediction accuracy. You can learn more about anomalies and outliers in the **anomaly detection** section.

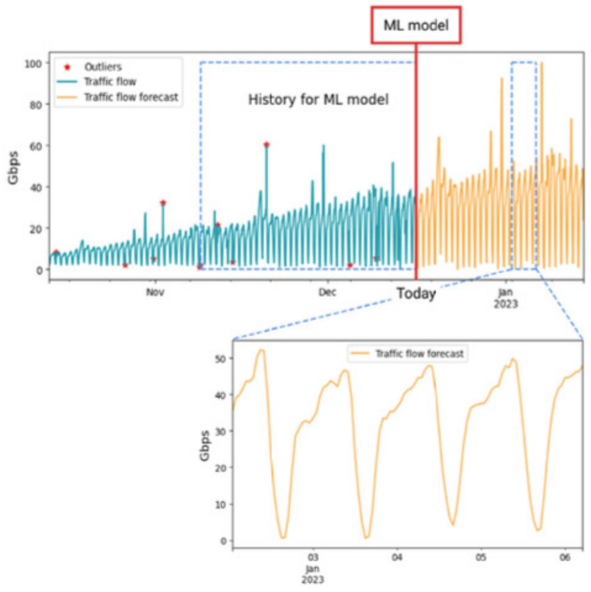


Figure 2. Time series forecasting

ML-based forecasting allows us to utilize the predicted metrics to build proactive decision support systems. For instance, long-term forecasting of traffic or resource utilization can help us better plan necessary spendings on new equipment. Such cyclical predictions can proactively provide useful information on future "lack of capacity" problems on particular devices.

In turn, medium-term traffic flow bit rate forecasting between all endpoints in the network can be used to build a proactive system to warn about future congestion points or even proactively suggest routing changes leading to a load-balanced network state. The key word here is proactivity - you can solve a problem before it really occurs.

Finally, short-term forecasting of time series data can also be used for anomaly detection. The forecast indicates the "expected value" for the next metric measurements. If there is a large difference between predicted and measured values, an alert can be raised. It is common practice to define thresholds for a specific metric to raise alerts when the metric value is over or under the predefined thresholds. Detecting anomalies based on comparison of actual values with expected values allows for raising alerts for anomalies also when the values do not exceed the defined thresholds, as the forecast is used as a reference.

Popular methods for time series forecasting

Below we list several of the most popular methods for time series forecasting. Other models can be found in such libraries as **darts**, **GluonTS** or **sktime.**

- **ARIMA** (Autoregressive Integrated Moving Average), SARIMAX (Seasonal ARIMA with eXogenous factor), VARIMA (Vectorized ARIMA)
- **Exponential smoothing**
- **Prophet**
- **Theta model**
- **TBATS** (Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components)
- **LSTM** (Long Short Term Memory)
- **DeepAR** (probabilistic forecasting with Autoregressive Recurrent networks)
- **N-BEATS** (Neural Basis Expansion Analysis for Interpretable Time Series forecasting)
- **Temporal Fusion Transformer**

There is no single method that is best for particular time series data, as much depends on data resolution, available historical data, required forecasting horizon and the statistical characteristics of the data. A multi-model approach can also be used, as a good practice that may lead to improved forecasting accuracy. It consists of decomposition of a time series into trends, seasonal and residual components, and applying various algorithms for forecasting them separately. Finally, such forecasted components are merged to retrieve a final forecast.

In the field of network and data center infrastructure, it is necessary to **simultaneously forecast multiple time series** from similar sources. One approach is to train a single, global model for the entire base of such correlated time series, and then use it to forecast each time series. In terms of forecast accuracy, it may be advantageous to train a global model, as it can take into account correlations between different time series from the database. VARIMA, DeepAR and Temporal Fusion Transformer are examples of such a global approach. An alternative approach is to apply a forecasting model separately to each time series, the so-called local approach. The other mentioned approaches usually create models for a single time series. We have to remember that taking into account the freshest data when training a model, usually provides better predictions. This means that models have to be updated (retrained) over time to keep a desired level of forecasting precision. This requires additional computing infrastructure.

Classification

In machine learning, **classification** is a supervised learning problem of identifying to which category an observation (or observations) belongs (see Figure 3). A category is often called a class label. Supervised learning means that both input variables (independent variables) and the output category (dependent variable) must be collected for the training process. The simplest example is classifying an email as spam/non-spam or a tweet as having positive or negative sentiment. These are examples of binary classification problems, as the class label can have only two values. When the number of classes is greater than two we are talking about a multi-class classification problem.

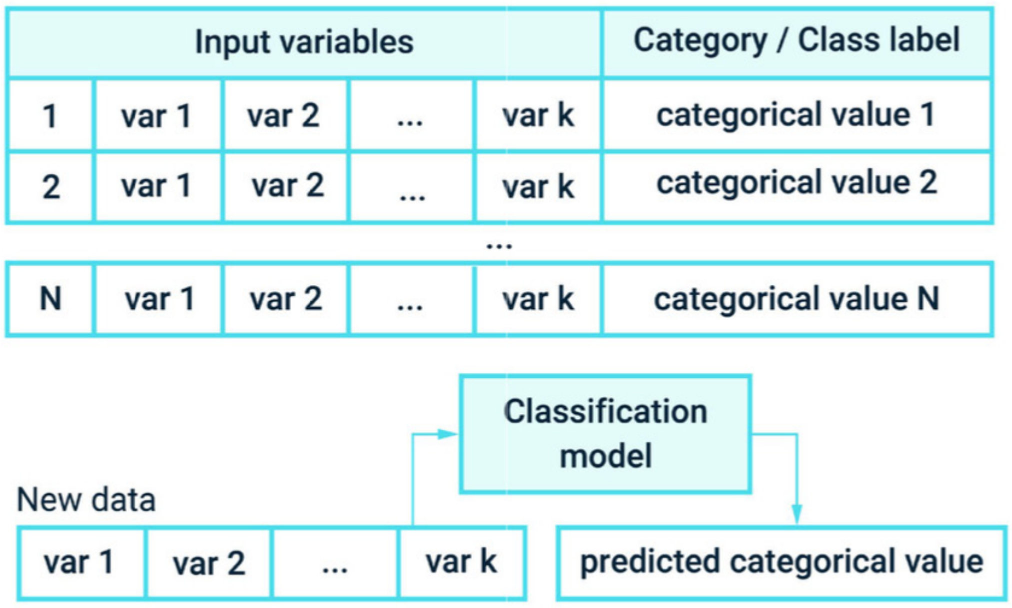


Figure 3. Classification problem

In a networking context, classification algorithms are often considered to distinguish different types of traffic such as video, voice or data and apply different routing or forwarding policies to them, for example. Also, classification methods can be used to distinguish various traffic categories such as "normal" traffic, bots, web crawlers and even attacks. Different traffic categories can then be handled in different ways or with different priorities. Automatic classification of users in terms of their behavior or traffic patterns is considered an important element of grouping them into classes and more efficient allocation of network resources, differentiating QoS and service offer, or even targeted advertising.

The classification of alerts and trouble ticket data, in turn, can be used to prioritize tasks for administrators to identify the most critical in a given moment. Classification algorithms can also be helpful in analyzing logs of different device types and their performance indicators, classifying them as healthy or unhealthy.

An important usage of ML classification algorithms also takes place in the area of security. Detection and prevention of various types of attacks, such as malware infection, SQL injection and DDoS attacks are examples where trained classifiers can play a key role. Also, classification algorithms can help distinguish legitimate traffic from suspicious traffic, detect intrusion attempts of known attacks, and identify emerging threats.

Popular classification algorithms

Below, algorithms that can be helpful in solving the above-mentioned problems are listed.

- Logistic Regression
- Linear Discriminant Analysis
- Classification and Regression Trees
- Naive Bayes
- K-Nearest Neighbors (KNN)
- Learning Vector Quantization (LVQ)
- Support Vector Machine (SVM)
- XGBoost

The algorithms apply different statistical approaches, require different computation power to be trained, and are more or less resistant to the occurrence of outliers in the data. The selection of appropriate algorithms is usually performed with an iterative experimental process. The final model accuracy is however strongly determined by the quality of properly labeled data. The process of data labeling can be expensive, especially when an automatic or semiautomatic approach is impossible in a specific use case.

Regression

Regression is another type of **supervised learning method**. Similarly to classification algorithms, regression algorithms try to predict an output variable based on input variables (also known as features), but in this case the output is a continuous numerical value (see Figure 4). As it is a supervised learning approach, both input variables (independent variables) and the output variable (dependent variable) must be collected for the training process. ML regression models can learn complex relationships between input features and the output variable, trying to find the function best mapping input variables to output variable. Thanks to such a function, the output variable can be easily evaluated for a new record of input variables.

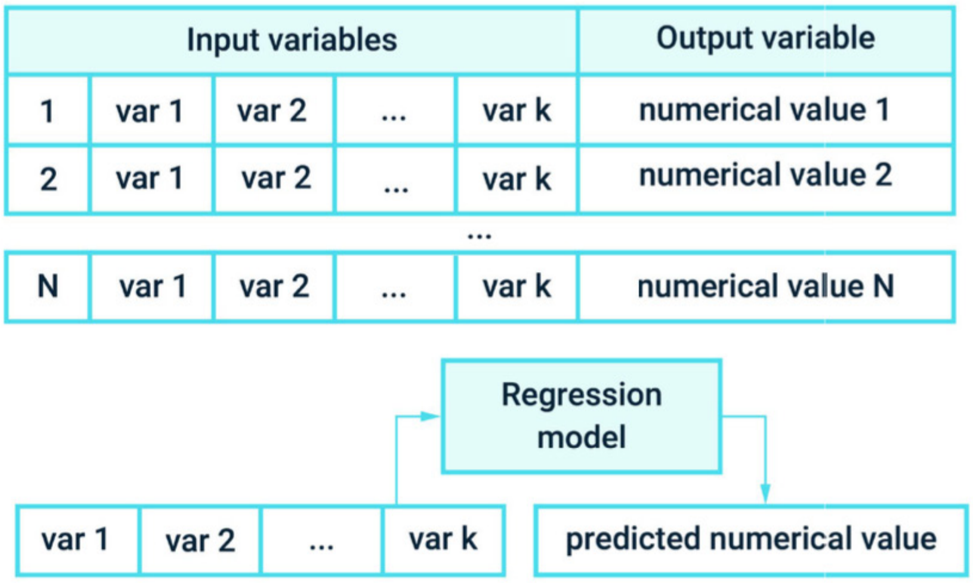


Figure 4. Regression Problem

In a network and data center management context, regression models can be useful to find relationships between QoS (Quality of Service) metrics and QoE (Quality of Experience) indicators. QoS metrics such as delay, packet loss, jitter, throughput, etc. refer to objective, system-related characteristics that provide insight into the performance (at a network level) of the service you want to deliver. On the other hand, QoE indicators measure performance of a particular service from the perspective of users. Consider a video streaming service - typical QoE metrics are the video startup time, video resolution, playback failure rate, rebuffering rate or an overall user satisfaction (for instance using a 0-to-10 rating scale). The QoE metrics can be objective and subjective and should properly reflect the true user satisfaction and willingness to continue using the service.

Finding correlations between QoS and QoE metrics allows us to build regression models describing the strength of influence of particular QoS metrics on QoE metrics. Such regression models can help identify those QoS metrics that have to be improved in order to optimize QoE metrics and thus increase the level of user satisfaction. From a practical point of view, QoE monitoring can be expensive and problematic as the measurements should be collected on user devices or via user surveys. Therefore, such measurements are usually only made periodically. However, having the computed regression models one can assess QoE metrics based on QoS metrics (the latter are a little easier to measure because they do not require user involvement).

Regression algorithms are often associated with time series data. By this, we mean scenarios where the input feature is time and the output variable is a measured metric. Almost everyone knows the linear regression approach used for quick trend evaluation of a measured metric. Also, other functions like polynomial, logistic, exponential, power, etc. can be used to reflect future trends in measured data. So, the natural question is when to use a regression algorithm and when to use a more sophisticated time series forecasting algorithm. The answer depends on the objectives, but there are some hints.

Regression algorithms in the context of time series data are more often used for evaluating trends. For instance, having monitored resource utilization (CPU, memory, link capacity), using simple linear regression you can assess when you will suffer from a lack of resources. When you are interested in trends but also seasonality (daily, weekly), you will instead use algorithms for time series forecasting, more specialized in seasonal pattern recognition.

For instance, forecasting traffic flows in terms of their peak hours and hourly bit rate distribution (seasonality) allows you to group flows that can compensate for each other (i.e. when one flow is high, the second one is low and vice versa). It can be beneficial for load balancing when we can route such traffic flows to share the same resources (links, paths in the network).

Similarly, consider two different applications, where one is extensively used during working hours and the second is extensively used in the evenings. The knowledge of such different, compensating seasonalities in terms of incoming and outgoing traffic can be used to instantiate such applications on the same infrastructure component (e.g. the same node of a Kubernetes cluster) if it does not break any other constraints, of course (e.g. related to security, etc.) The load characteristics of given links and network interfaces will then be more balanced compared to the situation when a pair of applications having traffic peaks at the same time are instantiated on the same infrastructure component.

It is also worth mentioning that a specific regression algorithm can also be used as a component of trend evaluation in time series forecasting, when a time series decomposition method is used. In such a case, the final algorithm can be composed of multiple models corresponding to different classes of ML problems.

Popular regression models

Below we list the most popular regression models that can be applied in various use cases.

- Linear Regression
- Ridge Regression
- LASSO Linear Regression (Least Absolute Shrinkage and Selection Operator)
- Elastic Net Regression
- K-Nearest Neighbors
- Classification and Regression Trees
- Support Vector Machine (with non-linear kernel)
- Neural network regression

Clustering

Clustering (also called segmentation) is a process of grouping similar entities into subgroups (clusters), where entities belonging to the same cluster are more similar to each other than to those of other clusters (see Figure 5). This is an unsupervised learning task, which means that it only interprets input data and finds natural groups taking into account feature space. Contrary to supervised learning, in unsupervised learning there is no output category or output variable.

Each entity in a data set is described by numerical and categorical variables and a key point is to use the notion of a "distance metric" able to represent similarity between entities. In most cases the number of clusters is unknown a priori and evaluation of identified clusters is subjective. Clustering is a method for knowledge extraction by grouping data records that is difficult or time consuming, even for experts in the considered domain. However, a valuable clustering of particular data often requires tight collaborative work between a domain expert and a data scientist.

In the context of network/DC management, clustering can be used for **analysis of trouble ticket data**. Retrieval of data about each handled trouble ticket, such as the problem type, the customer, related infrastructure elements, number of involved engineers, handling time, keywords describing detailed technical issues, etc. is the first step in applying cluster analysis.

Analysis of proposed clusters and trouble ticket examples in each group allows you to gain useful knowledge, e.g. which groups of troubles took too much time to be served, which involved too many engineers, and which may indicate specific environment configuration problems (e.g. network routing) that seem to occur too often, and therefore need to be further analyzed and improved.

It is worth noting that a similar analysis can be done by querying prepared tabular data using SQL queries with 'GROUP BY' statements. Such an approach is often used, but the cluster analysis algorithms use a more holistic approach, trying to find similarities taking into account features without imposing strict grouping criteria.

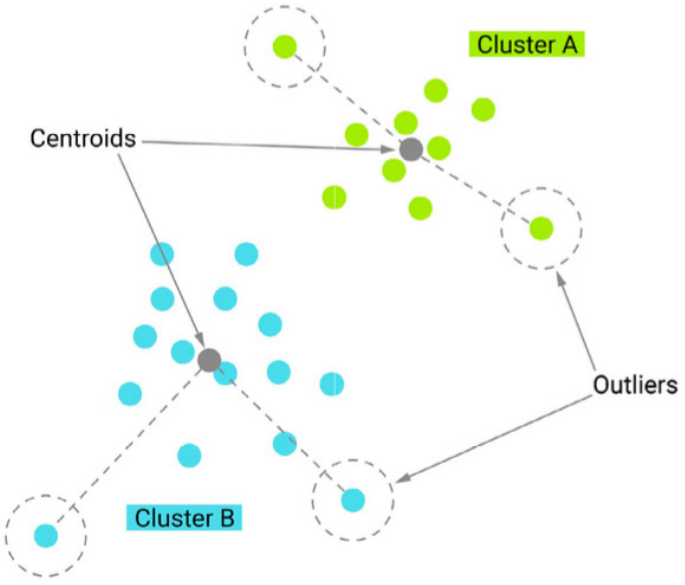


Figure 5. Cluster analysis

Another use case where cluster analysis could be used is the detection of anomalies. Many clustering algorithms use the notion of centroids, which describe the centers of the clusters. Such a centroid defines a typical or average representative of the group and the algorithms can also list outermost elements.

To be a little more practical, in the networking/DS management context, collection of configuration data for particular types of element (e.g. routers, switches, interfaces, firewalls, load balancers, applications, etc.) extended by recent operational statistics (performance counters, resource utilization, etc.) can be a good **dataset for segmentation in the context of searching for anomalous elements**. Thus, there is a chance to discover an incorrect configuration of those elements causing noticeable poor performance, potential vulnerabilities related to inconsistent settings, or applied policies. The general idea is to separate the potential anomalies and examine why they are different from the rest of the other, similar elements.

Algorithms for cluster analysis

There are many algorithms for cluster analysis available. There are also some methods for assessment if the number of clusters is optimal for a given dataset. However, there is no clear theory as to which clustering algorithm best matches particular data or business objectives. Again, the key point is collaboration between a data scientist and a domain expert, who iteratively interpret the clustering analysis results for different experiments (for various sets of features and applied algorithms). Below is a list of the most popular algorithms utilizing different approaches for determining clusters.

- Affinity Propagation
- K-Means
- Agglomerative Clustering
- BIRCH (Balanced Interative Reducing and Clustering using Hierarchies)
- DBSCAN (Density -Based Spatial Clustering of Applications with Noise)
- Mean Shift
- OPTICS (Ordering Points To Identify the Clustering Structure)
- Spectral Clustering

Anomaly detection

Anomaly detection has already been mentioned several times in the previous sections. This section aims to clarify the terms related to anomalies and outliers when data is processed with the use of ML methods.

Anomaly detection in machine learning focuses on identifying data points or data patterns that significantly differ from the expected norm or behavior. Anomalies can be detected for structured data organized in a specific format (tabular data, XML, JSON, CSV, spreadsheet) and unstructured data (mail, post, document, audio, video, image). The characteristics of the data greatly influence the choice of specific techniques and algorithms.

Anomalies are often called outliers and the two terms are used interchangeably. They both represent unusual data points, however there is a significant distinction between them. The difference lies in the context in which they occur. Outliers are defined based on the distribution of the data, so they might be caused by natural variations. Anomalies are defined based on an expected behavior or norm. In practice, we would usually like to detect anomalies to diagnose potential problems and we would usually like to correct or remove outliers from data when we are preparing an ML model to increase its accuracy.

In the context of networking and DC management, **detection of anomalies helps to ensure reliability and security**. Anomalies in time series data representing various performance metrics can be quickly discovered compared to forecasted/expected values. The key point here is to define what should be treated as an important anomaly. For instance, in how many consecutive data points such differences occur and how much they differ from the forecasted values. Usually, only a domain expert can give valuable input defining which anomalies are important.

Also, classification models can be used to distinguish abnormal situations in the context of resource utilization. For instance, high CPU and memory utilization when the request rate is low for some services might indicate abnormal behavior that can be detected and the appropriate engineers alerted. Similarly, cluster analysis of configuration files can find repeating patterns, group them in terms of similarity, and indicate those files that do not belong to any cluster.

Such a config file can be reviewed and modified when necessary. In the same way, trouble ticket data can be used to find tickets which were served too long, or engaged too many specialists.

Algorithms for anomaly detection

Below, we list a few ML-based algorithms that can be used for anomaly detection problems.

- Isolation forest
- One class SVM (one class Support Vector Machine)
- Local Outlier Factor
- KNN (K Nearest Neighbors)
- Gaussian mixture model

There are also other approaches, not necessarily based on ML techniques. What's more, hybrid or multi-method approaches are very often used to increase detection accuracy.

Reinforcement Learning

Reinforcement learning (RL) is another type of machine learning. The objective of reinforcement learning is to learn how to make decisions based on past experiences, on past decisions which could have been right or wrong. The key concepts related to RL are AI agent, environment, environment state space, actions set, and reward system.

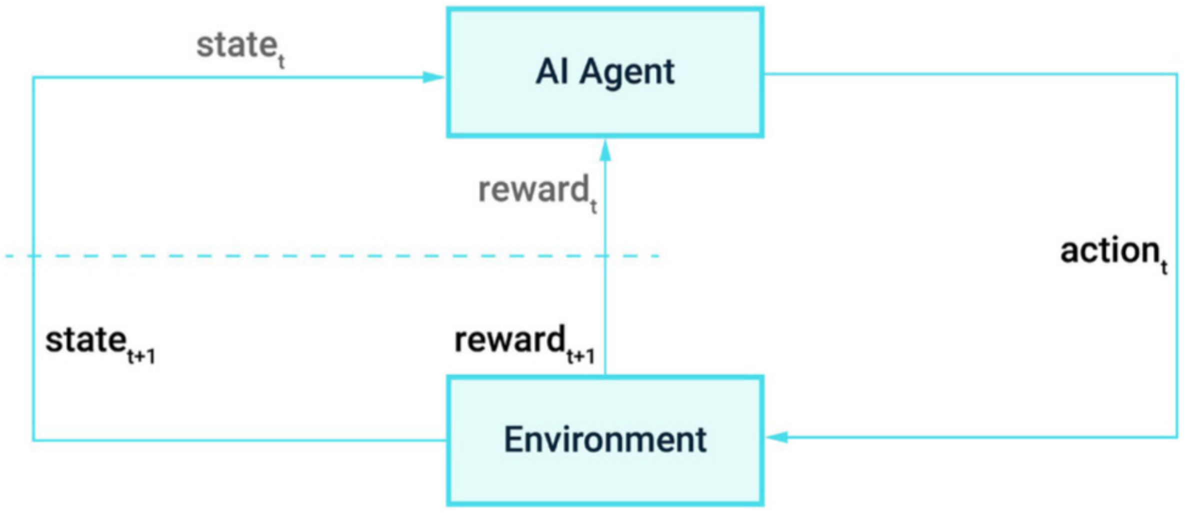


Figure 6. Reinforcement learning

The AI agent interacts with the environment by taking actions. Each action changes the state of the environment and can be evaluated in the form of a reward or penalty value. The discovered combinations of state, action, reward are used to learn a policy, which is a decision-making strategy the agent can use to select an optimal action in a given state. The goal is to maximize the cumulative reward value, obtained in finite consecutive time steps (Figure 6).

The reinforcement learning approach can be applied for network use cases, e.g. to optimize routing metrics in a network with link-state routing protocol (OSPF, IS-IS). In such a case, the network is an environment consisting of nodes and links with a given capacity. The environment state can be described by particular link weights that are used to compute the shortests paths between any pair of nodes. The AI agent is a central unit that can perform an action, i.e change those link weights. In consequence, the optimal routing paths for some origin-destination pairs of nodes may change.

Imagine that you also have a forecasted time series representing traffic demands between any pair of nodes, for which you would like to optimize the link weights. Let us assume that forecasted trends in traffic flows between nodes for current routing settings will cause some congestion on some links. Then some of the traffic will start to drop. The goal of the RL algorithm would be to learn the sequence of actions (changes in link weights on particular links) so that some routing paths traversing congested links are modified and less loaded parts of the network are used for them.

Designing RL algorithms for network problems is very challenging, as it usually assumes access to the real environment. This is a very unrealistic assumption, because no one will allow the training of an RL-based decision policy in a real environment, especially one as dynamic as a network environment. That is why the example presented above is slightly different. It operates on a network model at traffic flow level and tries to learn how to optimize routing for future traffic conditions. The goal is to find a set of changes in the routing configuration and proactively inform a network admin of expected congestion with solutions to overcome it.

There are many other approaches applying RL to routing-related problems in networking. They can be found here.

Challenges of AI and ML use in networks

If you want to solve practical network problems, you can use a wide range of available AI/ML methods, depending on the nature of the problem and the specific requirements related to it.

However there are many challenges to overcome. The most important are listed below.

- **Data volume and quality:** Measuring and collecting data from the network is a complex issue (you may want to read this blog post for more details). Sometimes the possibility of obtaining the appropriate volume of data needed for further analysis may be limited for various

reasons. Lack of proper data may limit the potential of the ML methods that you want to use. Also the quality of data used to train ML models is crucial for their accuracy.
- **Training data imbalance:** Collected training data can be imbalanced. For instance data anomalies that need to be discovered may be too rare to build valuable classification models.
- **Feature extraction and model accuracy:** Extracting relevant features from raw data and selecting the right features is critical to model accuracy. The process usually requires iterative experimentation and close cooperation between a data scientist, ML engineer, and domain expert.
- **Model interpretability:** Even accurate models must be interpretable. Decisions made based on such models need to be understandable by network engineers.
- **Scalability:** Many machine learning methods can be used for network data but in many cases they were not designed for such data. That is why choosing the right ML model in terms of its scalability can be a challenge
- **Real-time processing issue:** ou need to know that many ML algorithms were not designed for real-time data processing, which is often the case in a dynamic network environment.
- **Unique characteristics of network environments:** It is quite probable that ML models that work well for a given network environment can be inappropriate for other, bigger or more complex networks. You should also know that networks are a very heterogeneous environment (in terms of technology, topology, target use, etc.) and individual network segments can differ significantly from each other. Thus the usage of pretrained models (transfer learning) may not be possible and models will have to be built from scratch.
- **Required resources:** Data collection as well as training, usage and maintenance of ML models require significant storage and computational capacity. Especially when deep learning algorithms are used.
- **Multidisciplinary teams:** Application of various ML techniques usually requires engagement of engineers skilled in many disciplines. Close collaboration between data scientists, ML engineers and domain experts is essential for successful deployments.

Final conclusions

We have presented the crucial types of AI/ML methods and algorithms that are, at least in our opinion, the most relevant and helpful in the context of network-related problems. Also, some potential use cases where particular methods can be used have been mentioned. For each class of AI/ML problems, we have listed examples of the popular algorithms and frameworks. Finding the optimal algorithm requires iterative experiments with well-prepared data for the specific use case. We have outlined the necessity of tight collaboration between data scientists, ML engineers and domain experts at the stage of preparing valuable ML models. Finally, we have indicated the major challenges that need to be considered and overcome when applying ML techniques in network-related problems.

There is a common opinion among researchers and practitioners dealing with network problems, and trying to deploy various ML techniques, **that there are many clear ideas but little transfer to industry**. Engineers can identify weak points in existing network deployments and know what needs to be improved, but very often they do not know exactly how to apply ML techniques to automate such a process. At the same time, tremendous progress in the area of AI applications and in ML methods and algorithms is taking place. It seems mandatory for modern companies to build competences and a data processing culture in this area.

About the author

Tomasz Janaszka - Solution Architect

With 20 years of telco/IT experience under his belt, Tomasz has led and developed projects in network design, capacity planning, traffic engineering, resource optimization, and automating network management processes. Currently at CodiLime's R&D department, where he puts his telecommunications doctorate to good use by developing practical AI/ML applications in networking.

Looking for more ML/AI-related content in terms of networks? Definitely check out our AI/ML for networks webinar.

Build your AI/ ML solution with us!

Since 2011, CodiLime has been the engineering partner of choice for semiconductor companies, networking vendors, telecom services, and software solution providers. Our experts help explore their data assets better by creating scalable solutions that make storing, processing, and analyzing data a breeze.

We appreciate long-term collaborations above all, as well as our partners. Below are some of the names that have already trusted us:

cisco paloalto EQUINIX JUNIPER NUTANIX altera bigleaf networks
SELECTOR NTT Security Holdings FluxNinja FREEDOMFI CLOUDIFY AT&T
NVIDIA Captor Therapeutics GIGASPACES MESH7 neptune.ai swift NAVIGATION Procyon

CodiLime aims to **link network engineering talent with business domain expertise** – we focus on five N.E.E.D.S. - Networks, Equipment, Environment, Data and Security.

Do you need any further information regarding our services? Don't hesitate to contact us!