Understanding Network Traffic
An Introduction to Machine Learning in Networking

Pedro CASAS

FTW - Communication Networks Group
Vienna, Austria

IIE - FING - ARTES
Department of Telecommunications

Montevideo, Uruguay

01 – 05 September 2014

Creating Communication Technologies

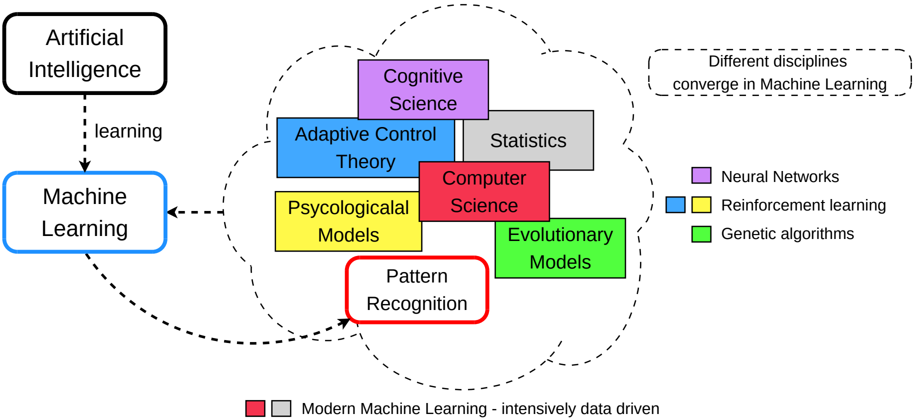
Outline

1. What is Machine Learning (ML) and why ML in Networking?
2. General overview on Machine Learning techniques:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Ensemble Learning
3. Features Extraction and Features Selection
   - Feature Extraction
   - Feature Selection
4. Final Remarks: Overfitting and Learning Evaluation
5. Machine Learning in Networking:
   - PSQA: Neural Networks for QoE Assessment
   - Sub-Space Clustering for Self Network Defense

Outline

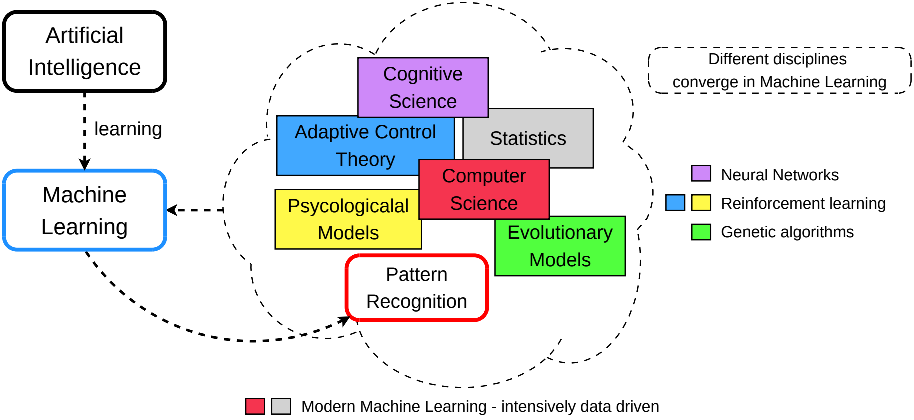
1. What is Machine Learning (ML) and why ML in Networking?
2. General overview on Machine Learning techniques:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Ensemble Learning
3. Features Extraction and Features Selection
   - Feature Extraction
   - Feature Selection
4. Final Remarks: Overfitting and Learning Evaluation
5. Machine Learning in Networking:
   - PSQA: Neural Networks for QoE Assessment
   - Sub-Space Clustering for Self Network Defense

A little bit of history...



1956 John McCarthy (Stanford): "**Artificial Intelligence** is the science and engineering of making intelligent machines, which can perceive their environment and take actions to maximize their chances of success".

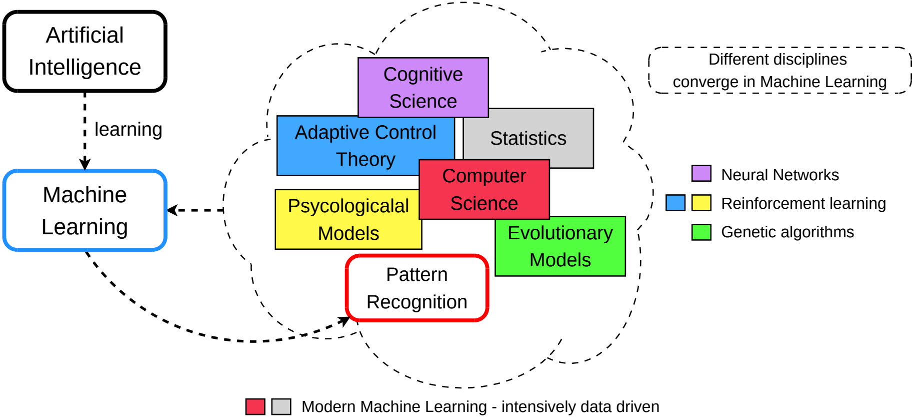
A little bit of history...



1956 Ray Solomonoff first mentioning the term "Learning Machines"...

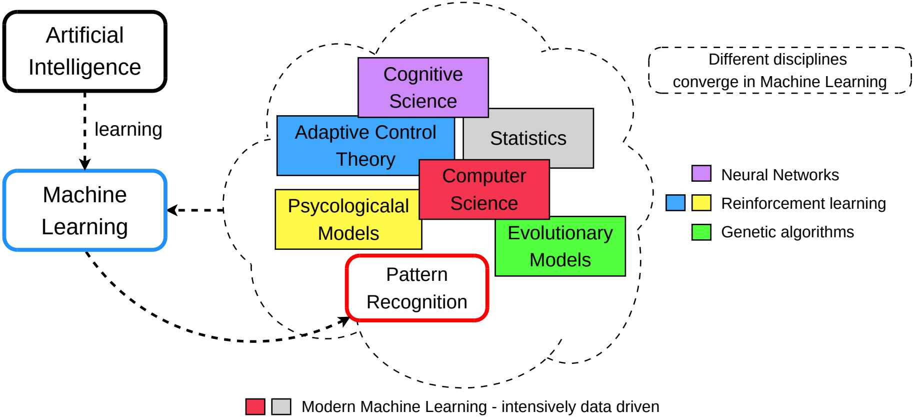
1980 ...but the first International Workshop on Machine Learning, in Pittsburgh (currently ICML) appears almost 25 years later.

A little bit of history...



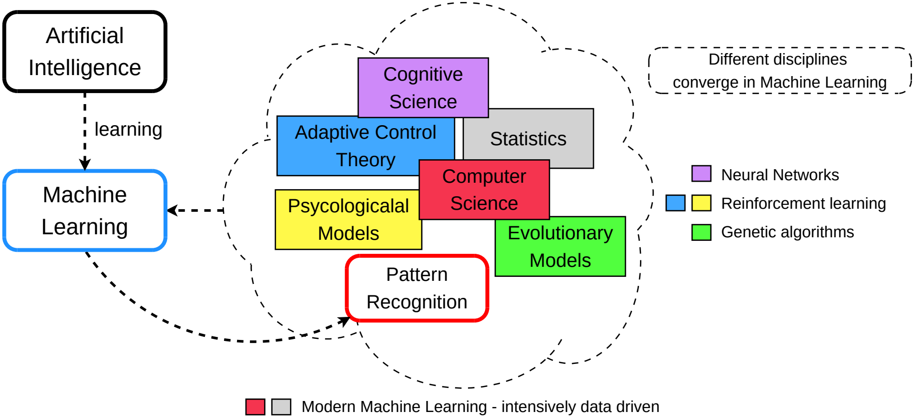
Machine Learning (ML) is about computational approaches to learning: ML aims to understand **computational mechanisms** by which **experience can lead to improved performance**, traducing these into computer algorithms.

A little bit of history...



Tom Mitchell (Chair ML Dept. in Carnegie Mellon): "ML consists in computer algorithms that improve their performance $\mathbf{P}$ on some task $\mathbf{T}$ through the experience $\mathbf{E}$...a well-defined learning task is given by $<\mathbf{P},\mathbf{T},\mathbf{E}>$".

A little bit of history...



ML in Traffic Analysis is mainly about **Pattern Recognition (PR)***: learn to automatically recognize complex patterns in data.

∗ C. Bishop, "Pattern Recognition and Machine Learning", Springer, 2006.

Machine Learning and Pattern Recognition in TMA

The **ever increasing amount of networking data** is a good reason to believe that **smart data analysis** will become even more pervasive as a necessary ingredient for **technological progress**:

Some good reasons for ML and PR in TMA:

- Proliferation of network traffic (social networking, web 2.0, video).

Machine Learning and Pattern Recognition in TMA

The **ever increasing amount of networking data** is a good reason to believe that **smart data analysis** will become even more pervasive as a necessary ingredient for **technological progress**:

Some good reasons for ML and PR in TMA:

- Proliferation of network traffic (social networking, web 2.0, video).
- Increased complexity of network and traffic modeling and analysis.

Machine Learning and Pattern Recognition in TMA

The **ever increasing amount of networking data** is a good reason to believe that **smart data analysis** will become even more pervasive as a necessary ingredient for **technological progress**:

Some good reasons for ML and PR in TMA:

- Proliferation of network traffic (social networking, web 2.0, video).
- Increased complexity of network and traffic modeling and analysis.
- Difficult to predict and to generalize applications' behavior.

Machine Learning and Pattern Recognition in TMA

The **ever increasing amount of networking data** is a good reason to believe that **smart data analysis** will become even more pervasive as a necessary ingredient for **technological progress**:

Some good reasons for ML and PR in TMA:

- Proliferation of network traffic (social networking, web 2.0, video).
- Increased complexity of network and traffic modeling and analysis.
- Difficult to predict and to generalize applications' behavior.
- Too many sources of knowledge to process by humans.

Machine Learning and Pattern Recognition in TMA

The **ever increasing amount of networking data** is a good reason to believe that **smart data analysis** will become even more pervasive as a necessary ingredient for **technological progress**:

Some good reasons for ML and PR in TMA:

- Proliferation of network traffic (social networking, web 2.0, video).
- Increased complexity of network and traffic modeling and analysis.
- Difficult to predict and to generalize applications' behavior.
- Too many sources of knowledge to process by humans.
- Black-boxes: some tasks cannot be well defined except by input/output examples.

Machine Learning and Pattern Recognition in TMA

The **ever increasing amount of networking data** is a good reason to believe that **smart data analysis** will become even more pervasive as a necessary ingredient for **technological progress**:

Some good reasons for ML and PR in TMA:

- Proliferation of network traffic (social networking, web 2.0, video).
- Increased complexity of network and traffic modeling and analysis.
- Difficult to predict and to generalize applications' behavior.
- Too many sources of knowledge to process by humans.
- Black-boxes: some tasks cannot be well defined except by input/output examples.
- Need for aggregated value solutions: *get the most out of data*.

So what is Pattern Recognition about?

- **Automatically assigning** a label to a given pattern (classification).

So what is Pattern Recognition about?

- **Automatically assigning** a label to a given pattern (classification).
- **Extracting** a model from example data (generalization).
- **Predicting** output values for unseen input cases (regression).

Machine Learning and Pattern Recognition in TMA

So what is Pattern Recognition about?

- **Automatically assigning** a label to a given pattern (classification).
- **Extracting** a model from example data (generalization).
- **Predicting** output values for unseen input cases (regression).
- **Recognizing** representative groups and outliers (clustering).

Machine Learning and Pattern Recognition in TMA

So what is Pattern Recognition about?

- **Automatically assigning** a label to a given pattern (classification).
- **Extracting** a model from example data (generalization).
- **Predicting** output values for unseen input cases (regression).
- **Recognizing** representative groups and outliers (clustering).

Some relevant examples in Traffic Monitoring and Analysis:

T $\longrightarrow$ Traffic-Flow Classification

P $\longrightarrow$ Percentage of flows correctly classified

E $\longrightarrow$ Set of labeled traffic flows: {flow descriptors, application}

Machine Learning and Pattern Recognition in TMA

So what is Pattern Recognition about?

- **Automatically assigning** a label to a given pattern (classification).
- **Extracting** a model from example data (generalization).
- **Predicting** output values for unseen input cases (regression).
- **Recognizing** representative groups and outliers (clustering).

Some relevant examples in Traffic Monitoring and Analysis:

T $\longrightarrow$ 0-day Attacks Detection

P $\longrightarrow$ Detection and false alarm rates

E $\longrightarrow$ Set of traffic flows free of attacks: {flow descriptors for normal activity}

Machine Learning and Pattern Recognition in TMA

So what is Pattern Recognition about?

- **Automatically assigning** a label to a given pattern (classification).
- **Extracting** a model from example data (generalization).
- **Predicting** output values for unseen input cases (regression).
- **Recognizing** representative groups and outliers (clustering).

Some relevant examples in Traffic Monitoring and Analysis:

T $\longrightarrow$ QoE Modeling and Prediction

P $\longrightarrow$ Percentage of correctly predicted QoE levels

E $\longrightarrow$ Set of subjective tests: {QoS/app. descriptors, QoE level}

ML: discipline vs tool to solve complex problems

ML in TMA IS NOT about trying different algorithms to obtain better results. To build a solid house on your own, you need to know about architecture, as well as about the intrinsic characteristics of the construction toolbox...

ML: discipline vs tool to solve complex problems

ML in TMA IS NOT about trying different algorithms to obtain better results. To build a solid house on your own, you need to know about architecture, as well as about the intrinsic characteristics of the construction toolbox. . .

Two commonly arising problems when coupling ML and Networking:

You have to understand the problem:

- Even a ML expert fails to achieve a good networking solution if he neither knows the good descriptors nor understands the problem (e.g., try to classify flows using only port numbers).
- Keep the scope narrow, to better understand the overall process (i.e., from selecting features to evaluation and conclusions).
- The solution must be meaningful in practical terms (e.g., predicting QoE from descriptors that can't be controlled is pretty useless for QoE management).

ML: discipline vs tool to solve complex problems

ML in TMA IS NOT about trying different algorithms to obtain better results. To build a solid house on your own, you need to know about architecture, as well as about the intrinsic characteristics of the construction toolbox...

Two commonly arising problems when coupling ML and Networking:

You have to understand the tool:

- The broader overview you have about the particularities of each ML approach, the better chances to apply the correct one (e.g., avoid killing mosquitos with a hammer).
- The research community does not benefit any further from yet another untried ML approach (e.g., IDS based on KDD'99 dataset).
- A good grasp of calculus, linear algebra, and probability is essential for a clear understanding of ML and PR in TMA and Networking.

Outline

1. What is Machine Learning (ML) and why ML in Networking?
2. General overview on Machine Learning techniques:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Ensemble Learning
3. Features Extraction and Features Selection
   - Feature Extraction
   - Feature Selection
4. Final Remarks: Overfitting and Learning Evaluation
5. Machine Learning in Networking:
   - PSQA: Neural Networks for QoE Assessment
   - Sub-Space Clustering for Self Network Defense

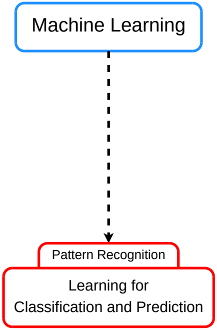
A General Machine Learning Taxonomy

This general taxonomy discriminates Machine Learning approaches by the **objectives of the learning task**.

Machine Learning

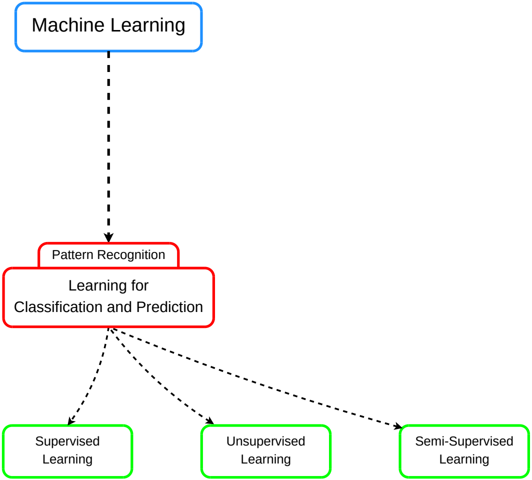
A General Machine Learning Taxonomy

This general taxonomy discriminates Machine Learning approaches by the **objectives of the learning task**.



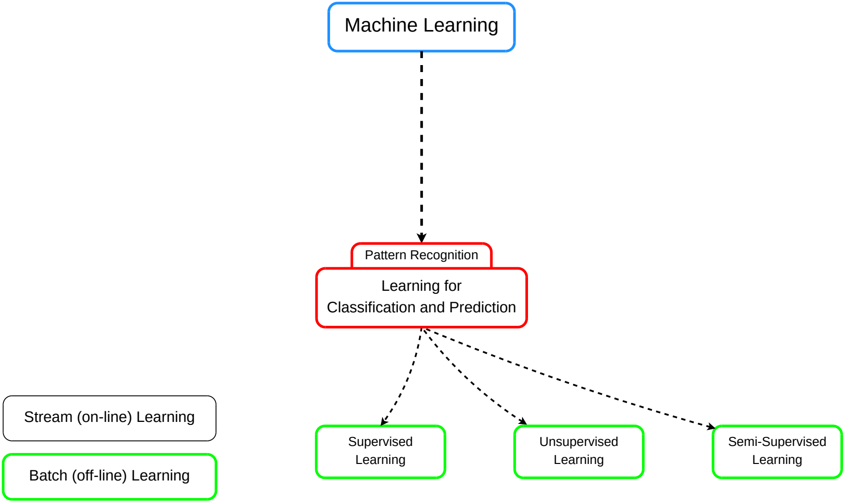
A General Machine Learning Taxonomy

This general taxonomy discriminates Machine Learning approaches by the **objectives of the learning task**.



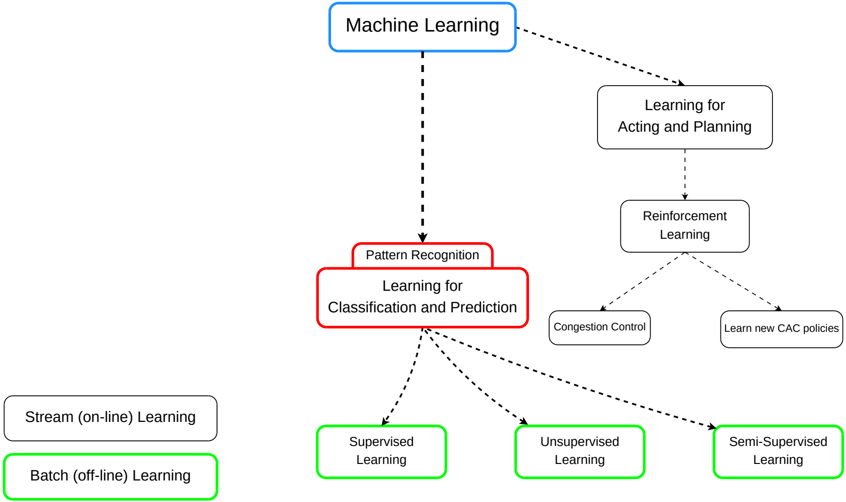
A General Machine Learning Taxonomy

This general taxonomy discriminates Machine Learning approaches by the **objectives of the learning task**.



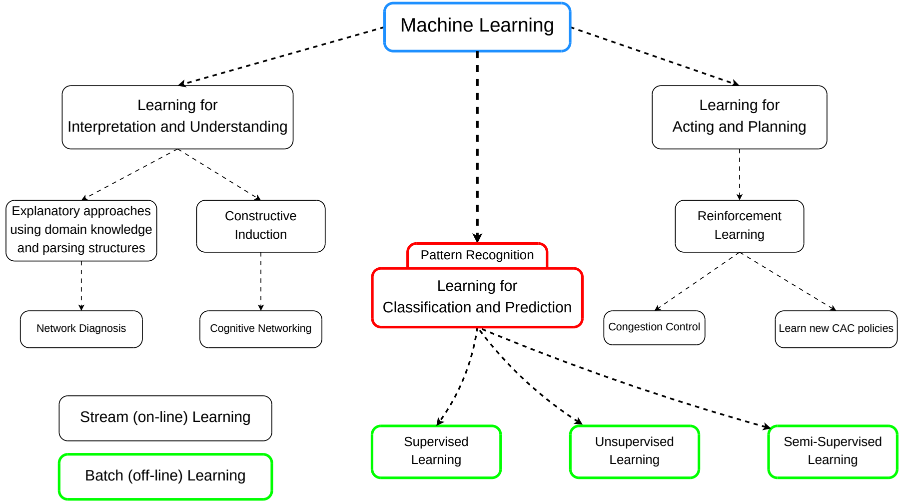
A General Machine Learning Taxonomy

This general taxonomy discriminates Machine Learning approaches by the **objectives of the learning task**.



A General Machine Learning Taxonomy

This general taxonomy discriminates Machine Learning approaches by the **objectives of the learning task**.



Patterns and Features

A **pattern** p represents the object under analysis, for which we want to draw some conclusion or **answer some particular question**:

In Traffic Classification, p could be a 5-tuple IP flow $\{IP_{src/dst}, port_{src/dst}, proto\}$

- Which of these applications generated flow p: Skype, YouTube, or µTorrent?

Patterns and Features

A **pattern** p represents the object under analysis, for which we want to draw some conclusion or **answer some particular question**:

In Traffic Classification, p could be a 5-tuple IP flow $\{IP_{src/dst}, port_{src/dst}, proto\}$

- Which of these applications generated flow p: Skype, YouTube, or μTorrent?

In Network Security, p could represent the aggregated traffic directed to the same IP_{dst} in a time slot of t seconds.

- Given some well defined notion of similarity, is p similar to other patterns or markedly different?

Patterns and Features

A **pattern** p represents the object under analysis, for which we want to draw some conclusion or **answer some particular question**:

In Traffic Classification, p could be a 5-tuple IP flow $\{IP_{src/dst}, port_{src/dst}, proto\}$

- Which of these applications generated flow p: Skype, YouTube, or µTorrent?

In Network Security, p could represent the aggregated traffic directed to the same IP_{dst} in a time slot of t seconds.

- Given some well defined notion of similarity, is p similar to other patterns or markedly different?

In QoE, p could represent a Skype call performed in some known network conditions.

- Which is the MOS value that an end-user would assign to this call?

Patterns and Features

Each **pattern** p is represented by a set of d descriptors or **features**, thus it can be interpreted as a point in a d-dimensional **feature space**:

$$p \rightarrow \mathbf{x} = \{x_1, x_2, x_3, \ldots, x_d\}$$

- Features represent the most critical part of the overall analysis; their accurate definition requires extensive domain knowledge.
- Quantitative: discrete or continuous, and qualitative: ordinal or nominal.

Patterns and Features

Each **pattern** p is represented by a set of d descriptors or **features**, thus it can be interpreted as a point in a d-dimensional **feature space**:

$$p \rightarrow \mathbf{x} = \{x_1, x_2, x_3, \ldots, x_d\}$$

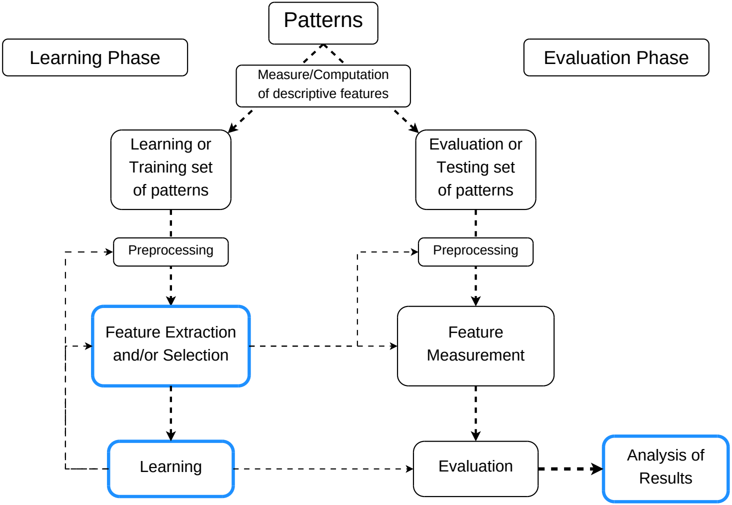
- Features represent the most critical part of the overall analysis; their accurate definition requires extensive domain knowledge.
- Quantitative: discrete or continuous, and qualitative: ordinal or nominal.

Some examples:

- Flow descriptors: # pkts, average pkt size, flow size and duration, average inter-pkts time, first 10 Fourier coefficients of pkt size, etc.
- Traffic descriptors: # IP flows, # IP srcs and dsts, # dsts ports, in time-slot t, etc.
- Video Streaming descriptors: codec, video bit-rate, video content nature, link bandwidth, loss rate, loss pattern, etc.

Design of a Learning Classification/Prediction System

Steps in the design of a **batch learning classifier/predictor**:



Outline

Supervised Learning

In **supervised learning**, there is a **label** associated to each pattern, which is supposed to **answer a particular question** about it:

- If the label is discrete, we talk about **Classification**
- If the label is continue, we talk about **Regression**
- We shall refer to these labels as the **Ground Truth** for our problem.

Supervised Learning

In Classification, we consider c classes $w_1, w_2, \ldots, w_c$, and assume:

- Classes are complete: $\cup_{i=1}^{c} w_i$ defines the problem space.
- Classes are mutually exclusive: $w_i \cap w_j = \emptyset$.
- Then, each label l_i corresponds to one single class w_i.

Supervised Learning

In Classification, we consider c classes $w_1, w_2, \ldots, w_c$, and assume:

- Classes are complete: $\cup_{i=1}^{c} w_i$ defines the problem space.
- Classes are mutually exclusive: $w_i \cap w_j = \emptyset$.
- Then, each label l_i corresponds to one single class w_i.

The Classification Problem:

Given a pattern p described by $\mathbf{x} = \{x_1, \ldots, x_d\}$, decide which of the c classes the pattern belongs to, i.e., decide which is its label l.

The Supervised Classification Problem:

Take a better decision by relying on a **training** ground truth set of patterns correctly classified:

$$S = \{p_i, l_i\}$$

Classification: a Probabilistic Approach

We assume that $\mathbf{x}$ belonging to class w_i is an observation drawn randomly from the class-conditional probability density function $p(x|w_i)$.

Imagine we know the *prior* probabilities of the classes $P(w_i)$ ($\sum_{i=1}^{c} P(w_i) = 1$).

Based only on $P(w_i)$, one would $\boxed{\text{decide label } l_i \text{ if } P(w_i) > P(w_j), \forall j \neq i.}$

Classification: a Probabilistic Approach

We assume that $\mathbf{x}$ belonging to class w_i is an observation drawn randomly from the class-conditional probability density function $p(x|w_i)$.

Imagine we know the *prior* probabilities of the classes $P(w_i)$ ($\sum_{i=1}^{c} P(w_i) = 1$).

Based only on $P(w_i)$, one would $\boxed{\text{decide label } l_i \text{ if } P(w_i) > P(w_j), \forall j \neq i.}$

If we now consider the conditional densities $p(x|w_i)$, we can refine our decision, using a **Bayesian approach** to get the *posterior* class probability:

$$P(w_i|\mathbf{x}) = \frac{p(\mathbf{x}|w_i)\, P(w_i)}{p(\mathbf{x})}$$

$$p(\mathbf{x}) = \sum_{i=1}^{c} p(\mathbf{x}|w_i)\, P(w_i)$$

Classification: Optimal Bayes Decision Rule

A decision problem has a *loss function* associating a cost to the decision.

$L(w_i|w_j)$ is the loss incurred in deciding w_i when the correct class is w_j.

The *expected loss* of deciding w_i, known as the *risk* of deciding w_i, is:

$$R(w_i|\mathbf{x}) = \sum_{i=1}^{c} L(w_i|w_j)\, P(w_j|\mathbf{x})$$

The optimal Bayes decision rule is the one that minimizes the risk:

$$\boxed{\text{decide } w_i \text{ if } R(w_i|\mathbf{x}) < R(w_j|\mathbf{x}), \forall j \neq i}$$

In classification, we use a binary loss function (0 correct, 1 otherwise).

The optimal decision becomes then a Maximum A Posteriori (MAP) rule:

$$\boxed{\text{decide } w_i \text{ if } P(w_i|\mathbf{x}) > P(w_j|\mathbf{x}), \forall j \neq i}$$

The Naïve Bayes Classifier

Using Bayes decision rule we can build a simple classifier.

$$P(w_i|\mathbf{x}) \propto p(\mathbf{x}|w_i)\, P(w_i)$$

$P(w_i)$ can be estimated from the training data set S ($P(w_i) = \#w_i/\#S$).
Regarding $p(\mathbf{x}|w_i)$, we can take the *naïve* approach (independent features):

$$\boxed{P(w_i|\mathbf{x}) \propto P(w_i) \prod_{j=1}^{d} p(x_j|w_i)}$$

The Naïve Bayes Classifier

Using Bayes decision rule we can build a simple classifier.

$$P(w_i|\mathbf{x}) \propto p(\mathbf{x}|w_i)\, P(w_i)$$

$P(w_i)$ can be estimated from the training data set S ($P(w_i) = \#w_i/\#S$).
Regarding $p(\mathbf{x}|w_i)$, we can take the *naïve* approach (independent features):

$$\boxed{P(w_i|\mathbf{x}) \propto P(w_i) \prod_{j=1}^{d} p(x_j|w_i)}$$

The class-conditional probabilities $p(x_j|w_i)$ can be estimated in multiple ways:

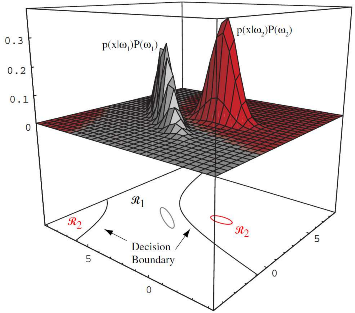
- Discretizing the values of x_j (e.g. histogram).
- Parametric estimation (maximum-likelihood estimation, using for example Gaussian distributions - Central Limit Theorem).
- Non-parametric estimation (e.g. kernel density estimation).

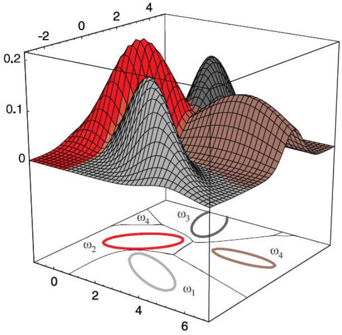
Discriminant Analysis

One common way to classify patterns is by defining a set of **discriminant functions** $g_i(\mathbf{x})$, $i = 1, \ldots, c$.

$$\boxed{l(\mathbf{x}) = \arg \max_{i=1,\ldots,c} g_i(\mathbf{x})}$$

The set of c discriminant functions divides the feature space into c *decision regions* $\mathcal{R}_i$, separated by *decision boundaries*:





Discriminant Analysis

A $0/1$-loss Bayes classifier (MAP classifier) is easily represented in this way, taking $g_i(\mathbf{x}) \propto P(w_i|\mathbf{x}) \propto p(\mathbf{x}|w_i)\,P(w_i)$.

For practical reasons, we usually take a logarithmic transformation of the discriminant functions:

$$g_i(\mathbf{x}) = \ln(p(\mathbf{x}|w_i)) + \ln(P(w_i))$$

Discriminant Analysis

A $0/1$-loss Bayes classifier (MAP classifier) is easily represented in this way, taking $g_i(\mathbf{x}) \propto P(w_i|\mathbf{x}) \propto p(\mathbf{x}|w_i)\, P(w_i)$.

For practical reasons, we usually take a logarithmic transformation of the discriminant functions:

$$g_i(\mathbf{x}) = \ln(p(\mathbf{x}|w_i)) + \ln(P(w_i))$$

Let us assume that class-conditional probabilities are multivariate normal: $p(\mathbf{x}|w_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. In this case, we can write $g_i(\mathbf{x})$ as:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^{\mathsf{T}}\, \boldsymbol{\Sigma}_i^{-1}\,(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(w_i) + \text{cte}$$

$$g_i(\mathbf{x}) = \mathbf{x}^{\mathsf{T}}\, \mathbf{W}_i^{-1}\, \mathbf{x} + \mathbf{w}_i^{\mathsf{T}}\, \mathbf{x} + \lambda_i \longrightarrow \text{a } \textbf{hyperquadric}$$

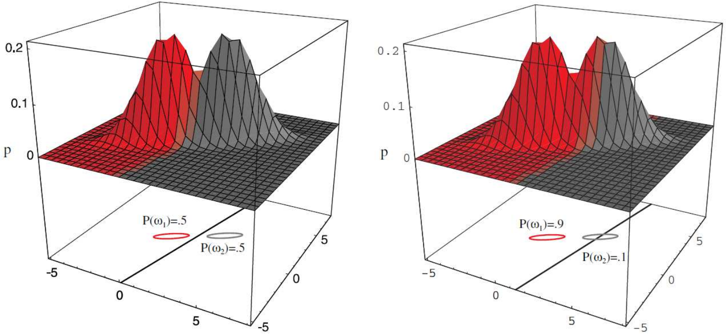
$$\mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1}, \quad \mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\,\boldsymbol{\mu}_i, \quad \lambda_i = -\frac{1}{2}\boldsymbol{\mu}_i^{\mathsf{T}}\, \boldsymbol{\Sigma}_i^{-1}\, \boldsymbol{\mu}_i - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(w_i)$$

Linear Discriminant Analysis

A particularly interesting case arises when the covariance matrices are identical, $\mathbf{\Sigma}_i = \mathbf{\Sigma}, \forall i = 1, \ldots, c$.

In this case, the hyperquadric becomes an **hyperplane** (i.e. the term $\mathbf{W}_i$ is the same $\forall g_i(\mathbf{x})$):

$$\boxed{g_i(\mathbf{x}) = \left(\mathbf{\Sigma}^{-1}\,\boldsymbol{\mu}_i\right)^{\mathsf{T}}\mathbf{x} - \left(\frac{1}{2}\boldsymbol{\mu}_i^{\mathsf{T}}\,\mathbf{\Sigma}^{-1}\,\boldsymbol{\mu}_i - \ln P(w_i)\right)}$$

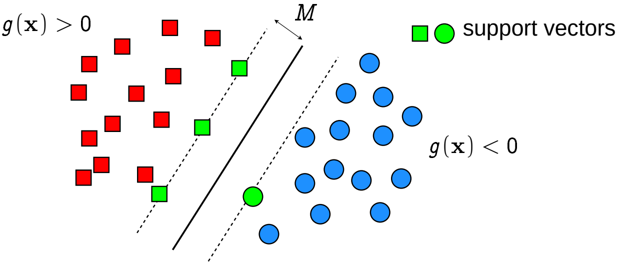


A Non-Probabilistic Approach: Support Vector Machines

Let us return to a two-class classification problem with labels $l_1 = 1$ and $l_2 = -1$, using a linear discriminant function:

$$\begin{aligned} g(\mathbf{x}) &= \mathbf{w}^{\mathsf{T}}\mathbf{x} + \lambda \\ \text{if } g(\mathbf{x}) > 0 &\rightarrow \text{decide } l = 1 \\ \text{if } g(\mathbf{x}) < 0 &\rightarrow \text{decide } l = -1 \end{aligned}$$

Let us assume that the training patterns are linearly separable in the feature space. We want to find the hyperplane $\{\mathbf{w}_0^{\mathsf{T}}, \lambda_0\}$ that maximizes the *margin* M:



Support Vector Machines

In this case, the n training patterns verify $l_i \, g(\mathbf{x}_i) > 0$, $i = 1, \ldots, n$. The margin M is the minimum distance from $g(\mathbf{x}_i)$ to a training pattern.

Using a proper change of variables, it can be shown that maximizing M is equal to the following quadratic optimization problem:

$$\boxed{\begin{array}{ll} & \min \ \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} & l_i \, g(\mathbf{x}_i) > 1, \ \forall i = 1, \ldots, n \end{array}}$$

Support Vector Machines

In this case, the n training patterns verify $l_i\, g(\mathbf{x}_i) > 0$, $i = 1, \ldots, n$. The margin M is the minimum distance from $g(\mathbf{x}_i)$ to a training pattern.

Using a proper change of variables, it can be shown that maximizing M is equal to the following quadratic optimization problem:

$$\boxed{\begin{array}{rl} & \min\ \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} & l_i\, g(\mathbf{x}_i) > 1,\ \forall i = 1, \ldots, n \end{array}}$$

Using Lagrange multipliers α_i, we compute the Lagrangian function:

$$L(\mathbf{w}, \lambda, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i \left(l_i\, (\mathbf{w}^{\mathsf{T}}\, \mathbf{x}_i + \lambda) - 1 \right)$$

The solution to $\left(\min_{\mathbf{w},\lambda} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \lambda, \boldsymbol{\alpha}) \right)$ gives $\mathbf{w}_0 = \sum_{i=1}^{n} \alpha_i\, l_i\, \mathbf{x}_i$ and λ_0.

Support Vector Machines

In the sum $\mathbf{w}_0 = \sum_{i=1}^{n} \alpha_i \, l_i \, \mathbf{x}_i$, it can be shown that $\alpha_i > 0$ only for the Support Vectors (SV): the patterns at the max M hyperplanes, i.e., $l_i \, (\mathbf{w}_0^{\mathsf{T}} \mathbf{x}_i + \lambda_0) = 1$.

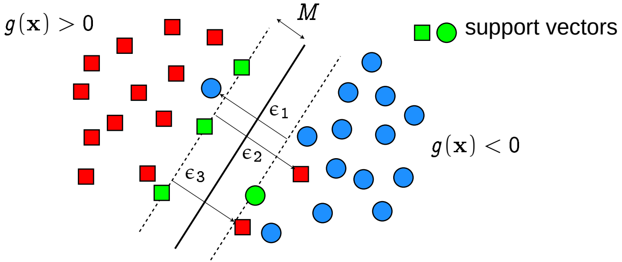
The only important patterns for the classification are the SV. The final classifier is given by $\boxed{g(\mathbf{x}) = (\sum_{i \in \mathrm{SV}} \alpha_i \, l_i \, \mathbf{x}_i)^T \mathbf{x} + \lambda_0.}$

Support Vector Machines

In the sum $\mathbf{w}_0 = \sum_{i=1}^{n} \alpha_i \, l_i \, \mathbf{x}_i$, it can be shown that $\alpha_i > 0$ only for the Support Vectors (SV): the patterns at the max M hyperplanes, i.e., $l_i \left(\mathbf{w}_0^{\mathsf{T}} \mathbf{x}_i + \lambda_0 \right) = 1$.

The only important patterns for the classification are the SV. The final classifier is given by $\boxed{g(\mathbf{x}) = \left(\sum_{i \in \text{SV}} \alpha_i \, l_i \, \mathbf{x}_i \right)^T \mathbf{x} + \lambda_0.}$

SVM can be slightly modified to consider misclassifications, adding some penalization ϵ_i for a misclassified pattern i:



Support Vector Machines

In this case, the optimization problem is the following:

$$\boxed{\begin{array}{rl} & \min C \sum_{i=1}^{n} \epsilon_i + \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} & l_i\, g(\mathbf{x}_i) \geqslant 1 - \epsilon_i,\ \epsilon_i > 0,\ \forall i = 1, \ldots, n \end{array}}$$

where $C > 0$ is a controls the tradeoff between penalty and the margin.

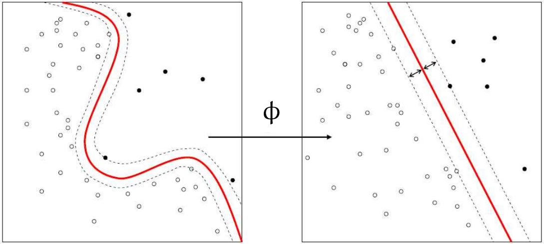
Support Vector Machines

In this case, the optimization problem is the following:

$$\boxed{\begin{array}{ll} & \min\ C\sum_{i=1}^{n}\epsilon_i + \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} & l_i\ g(\mathbf{x}_i) \geqslant 1 - \epsilon_i,\ \epsilon_i > 0,\ \forall i = 1,\ldots,n \end{array}}$$

where $C > 0$ is a controls the tradeoff between penalty and the margin.

So far we have considered a linear SVM classifier, but what about this case:



Non-linear SVM and the Kernel Trick

In a general case, the linear classifier can be rewritten as:

$$g(\mathbf{x}) = \mathbf{w}^{\mathsf{T}} \, \phi(\mathbf{x}) + \lambda$$

where $\phi(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathcal{F}$ is a feature space transformation. The corresponding SVM solution is exactly the same as before:

$$g(\mathbf{x}) = \left(\sum_{i \in \mathsf{SV}} \alpha_i \, l_i \, \phi(\mathbf{x}_i) \right)^{\mathsf{T}} \phi(\mathbf{x}_i) + \lambda_0$$

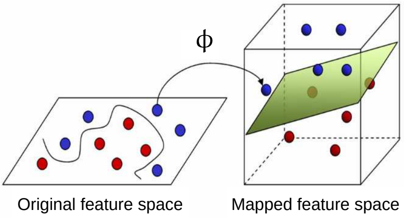
To apply the SVM solution in any general mapped feature space $\mathcal{F}$, it is only neccesary to know the inner product $\phi(\mathbf{x}_i)^{\mathsf{T}} \, \phi(\mathbf{x})$.

Non-linear SVM and the Kernel Trick

$$g(\mathbf{x}) = \left(\sum_{i \in \mathrm{SV}} \alpha_i \, l_i \, \phi(\mathbf{x}_i)\right)^{\mathsf{T}} \phi(\mathbf{x}_i) + \lambda_0$$

To apply the SVM solution in any general mapped feature space $\mathcal{F}$, it is only neccesary to know the inner product $\phi(\mathbf{x}_i)^{\mathsf{T}} \phi(\mathbf{x})$.

Patterns in higher dimensional spaces becomes separated, thus the linear SVM solution provides proper solution if the mapping is done to a much higher feature space $\mathcal{F} \in \mathbb{R}^m$, with $m >> d$:



Non-linear SVM and the Kernel Trick

But as we saw, we don't need to explicitly do the mapping, as we only need the inner product in $\mathcal{F}$.

The **kernel trick** permits to map the feature space into a high dimensional space with better structural properties, without actually doing the mapping.

We define the inner product in terms of a **kernel function** $k(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x}_i)^\mathsf{T} \phi(\mathbf{x})$:

$$\boxed{g(\mathbf{x}) = \sum_{i \in \text{SV}} \alpha_i \, l_i \, k(\mathbf{x}, \mathbf{x}_i) + \lambda_0}$$

Some standard kernel functions:

- Linear: $k(\mathbf{x}, \mathbf{x}_i) = \mathbf{x}_i^\mathsf{T} \mathbf{x}$
- Polynomial: $k(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}_i^\mathsf{T} \mathbf{x})^p$
- Gaussian radial basis function: $k(\mathbf{x}, \mathbf{x}_i) = e^{-\gamma \, ||\mathbf{x} - \mathbf{x}_i||^2}$

Non-linear SVM and the Kernel Trick

The **kernel trick** permits to map the feature space into a high dimensional space with better structural properties, without actually doing the mapping.

$$\boxed{g(\mathbf{x}) = \sum_{i \in \mathrm{SV}} \alpha_i \, l_i \, k(\mathbf{x}, \mathbf{x}_i) + \lambda_0}$$

In a **Multiclass SVM** problem, we can take two simple procedures to generalize the above classifier:

- **one-vs-all**: c different SVMs, the classifier with the highest output assigns the class (classifiers must be scaled for comparison):

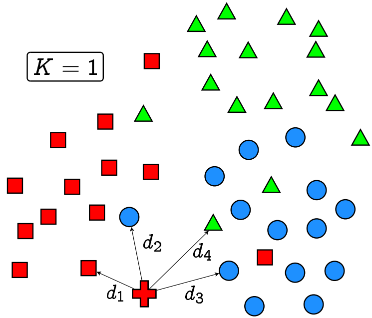
$$l(\mathbf{x}) = \arg \max_{i=1,\ldots,c} g_i(\mathbf{x})$$

- **one-vs-one**: $c(c-1)/2$ different 2-class SVMs, then every classifier assigns a class, and the class with more votes is chosen.

* **Note:** SVM can also be used for regression.

A Metric-based Classifier: K-Nearest Neighbors

The simplest and most intuitive classifier is based on the concept of **similarity**: similar patterns should be assigned to the same class:

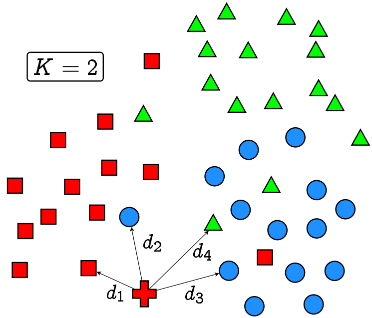


In K-NN, we decide the class of a new pattern by a majority vote of its k neighbors, **given a similarity measure** (e.g. Euclidean distance).

K-NN assumes no knowledge on the underlying classes; it is based on the training patterns alone. Note that K-NN has no training phase.

A Metric-based Classifier: K-Nearest Neighbors

The simplest and most intuitive classifier is based on the concept of **similarity**: similar patterns should be assigned to the same class:

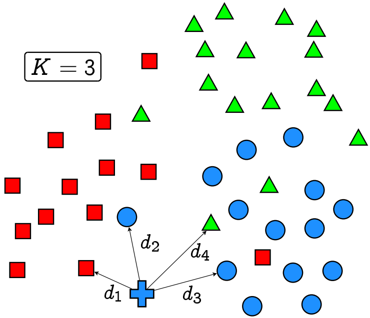


In K-NN, we decide the class of a new pattern by a majority vote of its k neighbors, **given a similarity measure** (e.g. Euclidean distance).

K-NN assumes no knowledge on the underlying classes; it is based on the training patterns alone. Note that K-NN has no training phase.

A Metric-based Classifier: K-Nearest Neighbors

The simplest and most intuitive classifier is based on the concept of **similarity**: similar patterns should be assigned to the same class:

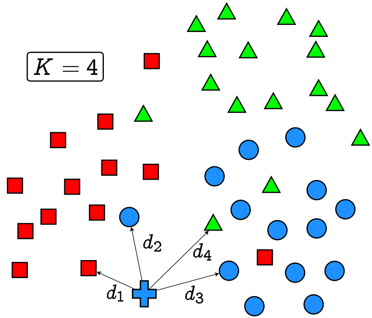


In K-NN, we decide the class of a new pattern by a majority vote of its k neighbors, **given a similarity measure** (e.g. Euclidean distance).

K-NN assumes no knowledge on the underlying classes; it is based on the training patterns alone. Note that K-NN has no training phase.

A Metric-based Classifier: K-Nearest Neighbors

The simplest and most intuitive classifier is based on the concept of **similarity**: similar patterns should be assigned to the same class:

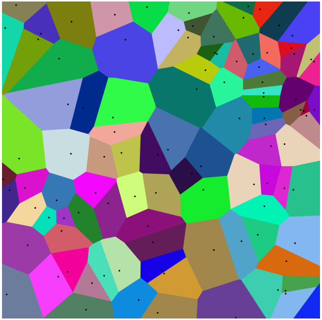


In K-NN, we decide the class of a new pattern by a majority vote of its k neighbors, **given a similarity measure** (e.g. Euclidean distance).

K-NN assumes no knowledge on the underlying classes; it is based on the training patterns alone. Note that K-NN has no training phase.

A Metric-based Classifier: K-Nearest Neighbors

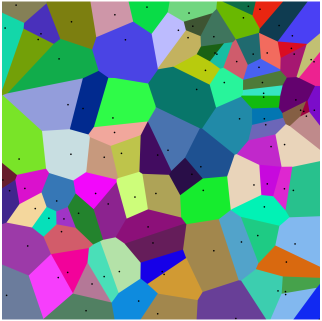
An interesting case is obtained for $K = 1$, where we get a decomposition of the feature space in n convex regions called **Voronoi cells**:



Note: if the number of training samples n is very large, then the error rate of 1-NN is never worse than twice the Bayes (minimum) error rate, awesome for such a simple algorithm!

A Metric-based Classifier: K-Nearest Neighbors

An interesting case is obtained for $K = 1$, where we get a decomposition of the feature space in n convex regions called **Voronoi cells**:



Some limitations of K-NN:

- Computationally expensive in both time and memory.
- Classes with more frequent examples tend to dominate the classification.

* **Note:** K-NN can also be used for regression.

A Non-Metric Classifier: Decision Trees

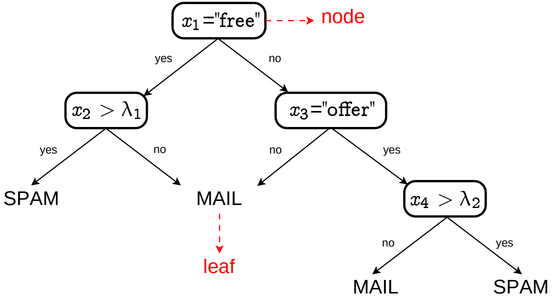
Consider a feature space with no similarity metric, e.g., nominal features (for continuous features, we do not consider any *distance* among them).

How to construct a classifier with no-metric features?

We can build a partition of the feature space by **asking multiple questions.**

The next question depends on the previous answer; questions do not repeat.

These questions build a **decision tree**; we use only binary questions.



A Non-Metric Classifier: Decision Trees

How do we build such a tree?

A Non-Metric Classifier: Decision Trees

How do we build such a tree?

- At each node N, we make the question that minimizes the *impurity* in the immediate descendant nodes.
- The most popular impurity measure is the **entropy impurity**:

$$\begin{aligned} i(N) &= -\sum_{j=1}^{c} P(w_j) \log_2 P(w_j), \quad i(N) \in [0, \log_2(c)] \\ P(w_j) &= \% \text{ patterns at } N \in w_j \end{aligned}$$

A Non-Metric Classifier: Decision Trees

How do we build such a tree?

- At each node N, we make the question that minimizes the *impurity* in the immediate descendant nodes.
- The most popular impurity measure is the **entropy impurity**:

$$\begin{aligned} i(N) &= -\sum_{j=1}^{c} P(w_j)\log_2 P(w_j), \;\; i(N) \in [0, \log_2(c)] \\ P(w_j) &= \% \text{ patterns at } N \in w_j \end{aligned}$$

- At node N, a question on feature x_i reduces the impurity by $\Delta i(N)$:

  $\Delta i(N) = i(N) - P_L i(N_L) - P_R i(N_R),\; P_L$ % patterns $\in$ left node N_L

A Non-Metric Classifier: Decision Trees

How do we build such a tree?

- At each node N, we make the question that minimizes the *impurity* in the immediate descendant nodes.
- The most popular impurity measure is the **entropy impurity**:

$$
\begin{aligned}
i(N) &= -\sum_{j=1}^{c} P(w_j) \log_2 P(w_j), \;\; i(N) \in [0, \log_2(c)] \\
P(w_j) &= \% \text{ patterns at } N \in w_j
\end{aligned}
$$

- At node N, a question on feature x_i reduces the impurity by $\Delta i(N)$:

  $\Delta i(N) = i(N) - P_L i(N_L) - P_R i(N_R), \; P_L$ % patterns $\in$ left node N_L

- So at each step, we create a new node by taking the feature that maximizes $\Delta i(N)$.
- This recursive-growing approach is the one used in $\mathrm{ID3}$ and its successor $\mathrm{C4.5}$ trees.

A Non-Metric Classifier: Decision Trees

Stopping Criterion:

- Growing the tree to the minimum impurity may cause **overfitting**.
- In the practice, there is a post-pruning of the tree to reduce overfitting.
- Occam's razor principle: prefer compact trees with few nodes.

A Non-Metric Classifier: Decision Trees

Stopping Criterion:

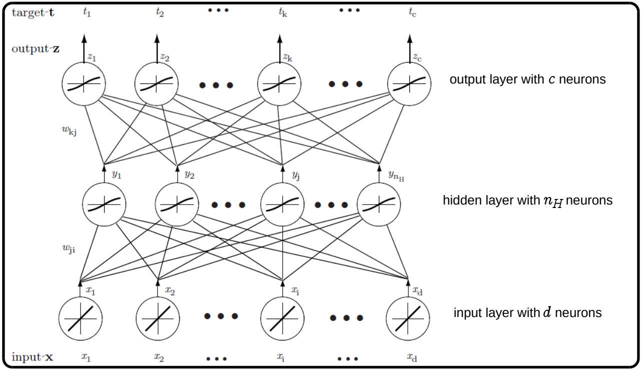
- Growing the tree to the minimum impurity may cause **overfitting**.
- In the practice, there is a post-pruning of the tree to reduce overfitting.
- Occam's razor principle: prefer compact trees with few nodes.

Properties of Decision Trees:

- Very easy to interpret, provide basic filtering rules.
- Very fast classification.
- It is simple to include domain knowledge from experts.
- Explicitly shows the importance of different features.

Multilayer Feed-forward Neural Networks

Neural networks provide a powerful model for classification and regression. We describe a particular model: **3-layers feed-forward neural network**:



c discriminant functions $g_k(\mathbf{x}) = f\left(\sum_{j=1}^{n_H} w_{kj}\, f\left(\sum_{i=1}^{d} w_{ji}\, x_i + w_{j0}\right) + w_{k0}\right)$

Multilayer Feed-forward Neural Networks

In this 3-layers model:

- Neurons in one layer connect to the next through **neural weights** w_{ji} .
- Each input neuron i just copies its input x_i at the output.
- The output of hidden neuron j is a non-linear function f applied to the weighted sum of input layer outputs.
- The output of output neuron k is a non-linear function f applied to the weighted sum of hidden layer outputs.

Multilayer Feed-forward Neural Networks

The neural network training (i.e., estimating the neural weights $\mathbf{w}$) is done from the set of training patterns, minimizing the squared estimation error:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{c}\left(g_k(\mathbf{x}) - z_k(\mathbf{x})\right), \quad z_k(\mathbf{x}) \text{ is the ground truth output}$$

which is generally achieved by gradient descent. **Backpropagation** is the simplest method for doing this supervised learning of the weights $\mathbf{w}$.

NOTE: the number of input and output neurons is defined by the problem itself, but for n_H we are free to choose; n_H generally has an important influence on the performance of the network (i.e., overfitting, input/output mapping, etc).

Multilayer Feed-forward Neural Networks

The neural network training (i.e., estimating the neural weights $\mathbf{w}$) is done from the set of training patterns, minimizing the squared estimation error:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{c}\left(g_k(\mathbf{x}) - z_k(\mathbf{x})\right), \quad z_k(\mathbf{x}) \text{ is the ground truth output}$$

which is generally achieved by gradient descent. **Backpropagation** is the simplest method for doing this supervised learning of the weights $\mathbf{w}$.

NOTE: the number of input and output neurons is defined by the problem itself, but for n_H we are free to choose; n_H generally has an important influence on the performance of the network (i.e., overfitting, input/output mapping, etc).

Universal approximation theorem:
Any continuous input/output function can be implemented in a 3-layer-ff-net, given sufficient number of hidden neurons, proper non-linearities, and weights.

Outline

1. What is Machine Learning (ML) and why ML in Networking?
2. General overview on Machine Learning techniques:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Ensemble Learning
3. Features Extraction and Features Selection
   - Feature Extraction
   - Feature Selection
4. Final Remarks: Overfitting and Learning Evaluation
5. Machine Learning in Networking:
   - PSQA: Neural Networks for QoE Assessment
   - Sub-Space Clustering for Self Network Defense

Unsupervised Learning

In **unsupervised learning**, the set of patterns for training has **no labels**.

This is the case in many (or most) real life applications, where labeling is a very expensive and difficult (sometimes even impossible) to achieve task.

Therefore, unsupervised learning is about **finding relevant structures in the data** (overlapping with data-mining).

Unsupervised Learning

In **unsupervised learning**, the set of patterns for training has **no labels**.

This is the case in many (or most) real life applications, where labeling is a very expensive and difficult (sometimes even impossible) to achieve task.

Therefore, unsupervised learning is about **finding relevant structures in the data** (overlapping with data-mining).

Standard approaches to unsupervised learning include:

- Parametric: mixture-resolving or identifying modals in data.
- Non-Parametric: find natural groupings or **clusters**.

So what is Clustering about?

The objective of clustering is to **divide a set of unlabeled patterns into homogeneous groups of similar characteristics**, based on some measure of similarity.

The Clustering Problem:

- Given a set of n d-dimensional **unlabeled patterns** $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$
- and given some measure of **similarity** among these patterns,
- **divide** this set into **homogeneous groups of similar characteristics** or **clusters**.

So what is Clustering about?

The objective of clustering is to **divide a set of unlabeled patterns into homogeneous groups of similar characteristics**, based on some measure of similarity.

The Clustering Problem:

- Given a set of n d-dimensional **unlabeled patterns** $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$
- and given some measure of **similarity** among these patterns,
- **divide** this set into **homogeneous groups of similar characteristics** or **clusters**.

- Clustering is the first step when analyzing unknown data (i.e. unlabeled data).
- Clustering is a natural classification process: degree of similarity among forms.
- Clustering is about data exploration: discover underlying structure in the data, generate hypotheses, detect anomalies.
- **Cluster analysis is an exploratory tool**.

Clustering Algorithms

- Clustering analysis first appeared in the title of a paper analyzing anthropological data back in 1954.
- Today, we have hundreds of clustering algorithms to choose from.

Clustering Algorithms

- Clustering analysis first appeared in the title of a paper analyzing anthropological data back in 1954.
- Today, we have hundreds of clustering algorithms to choose from.

Most clustering algorithms are divided in two groups:

- **Partitional clustering**: produce a single partition of the patterns in k clusters, optimizing some performance criterion.
- **Hierarchical clustering**: produce multiple "nested" partitions in a hierarchical structure.

Clustering Algorithms

- Clustering analysis first appeared in the title of a paper analyzing anthropological data back in 1954.
- Today, we have hundreds of clustering algorithms to choose from.

Most clustering algorithms are divided in two groups:

- **Partitional clustering**: produce a single partition of the patterns in k clusters, optimizing some performance criterion.
- **Hierarchical clustering**: produce multiple "nested" partitions in a hierarchical structure.

A bit of history in Clustering developments:

1957 Hierarchical Clustering
1967 k-Means
1970 Mixture models
1971 Graph-theoretic methods
1973 Fuzzy Clustering (soft clustering)
1982 Self Organization Maps (based on ANN)
1992 Vector Quantization (density identification of High Dimensional data)
1996 Density-based Clustering (DBSCAN)
1998 Sub-Space Clustering (High Dimensional data)
2000 Spectral Clustering (dimensionality reduction)
2002 Ensemble Clustering (combine weak partitions)
2004 Semi-Supervised Clustering
... and the list goes on

The User's Dilemma

Clustering involves taking many decisions:

- What is a cluster?

The User's Dilemma

Clustering involves taking many decisions:

- What is a cluster?
- **Which features to use?**

The User's Dilemma

Clustering involves taking many decisions:

- What is a cluster?
- **Which features to use?**
- How to define pair-wise similarity?

The User's Dilemma

Clustering involves taking many decisions:

- What is a cluster?
- **Which features to use?**
- How to define pair-wise similarity?
- **Which clustering algorithm?**

The User's Dilemma

Clustering involves taking many decisions:

- What is a cluster?
- **Which features to use?**
- How to define pair-wise similarity?
- **Which clustering algorithm?**
- How many clusters?

The User's Dilemma

Clustering involves taking many decisions:

- What is a cluster?
- **Which features to use?**
- How to define pair-wise similarity?
- **Which clustering algorithm?**
- How many clusters?
- Does the data has a clustering tendency/underlying structure?

The User's Dilemma

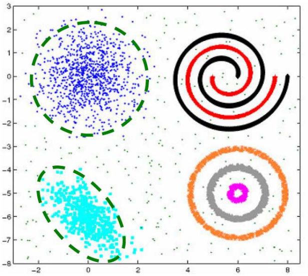
Clustering involves taking many decisions:

- What is a cluster?
- **Which features to use?**
- How to define pair-wise similarity?
- **Which clustering algorithm?**
- How many clusters?
- Does the data has a clustering tendency/underlying structure?
- **Are the discovered clusters and partition valid?**

What is a Cluster?

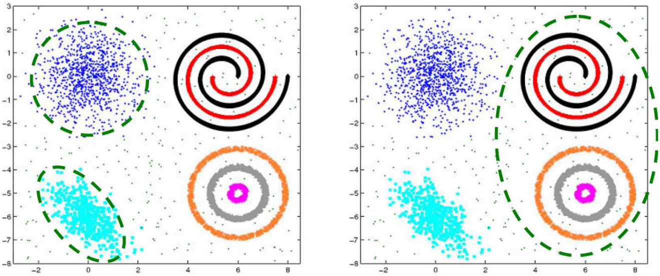
- Our notions of cluster comes from a 3-D world: **compact and isolated regions**...
- ...but cluster's definition depends on how we define similarity:

What is a Cluster?



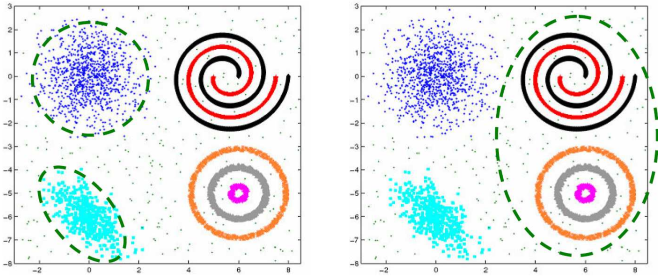
- Our notions of cluster comes from a 3-D world: **compact and isolated regions**...
- ...but cluster's definition depends on how we define similarity:
- **Compact clusters:** intra-cluster distance < inter-cluster distance

What is a Cluster?



- Our notions of cluster comes from a 3-D world: **compact and isolated regions**...
- ...but cluster's definition depends on how we define similarity:
- **Compact clusters:** intra-cluster distance < inter-cluster distance
- **Connected clusters:** intra-cluster connectivity > inter-cluster connectivity

What is a Cluster?



- Our notions of cluster comes from a 3-D world: **compact and isolated regions**...
- ...but cluster's definition depends on how we define similarity:
- **Compact clusters:** intra-cluster distance < inter-cluster distance
- **Connected clusters:** intra-cluster connectivity > inter-cluster connectivity
- Different algorithms use different notions of cluster $\longrightarrow$ they provide different identification results.
- Domain specific knowledge is useful in determining the most useful cluster shape.

Which Features to Use?

- A good representation leads to compact and isolated clusters.
- Using the **best** and **least** features is paramount in Clustering.
- **Feature Engineering if the key** in any machine learning algorithm.
- We talk about **Dimensionality Reduction**.

Which Features to Use?

- A good representation leads to compact and isolated clusters.
- Using the **best** and **least** features is paramount in Clustering.
- **Feature Engineering if the key** in any machine learning algorithm.
- We talk about **Dimensionality Reduction**.

And what for?

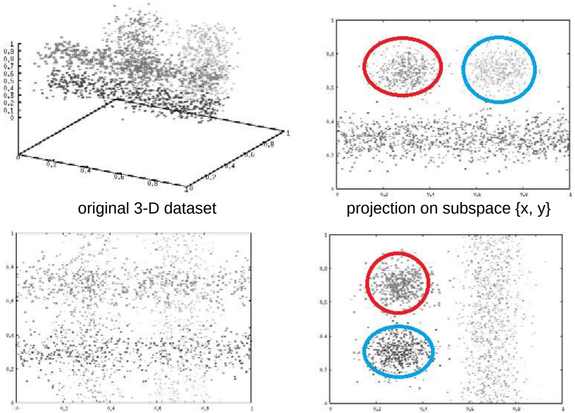
- Improving accuracy of the analysis.
- Reduce measurement costs.
- Create faster systems with less memory constraints.
- Simplify the interpretation of results.

Dimensionality Reduction

Naive approach: adding more features does not hurt, since at worst they provide no new information $\longrightarrow$ **WRONG!**

Dimensionality Reduction

Naive approach: adding more features does not hurt, since at worst they provide no new information $\longrightarrow$ **WRONG!**



original 3-D dataset

projection on subspace {x, y}

projection on subspace {x, z}

projection on subspace {y, z}

- **Irrelevant features mask real clusters and complicates clustering.**

Feature Extraction and Feature Selection

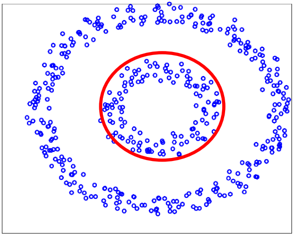
- Patterns are generally located in low dimensional manifolds embeded in the input space. **How to find them?**

Feature Extraction and Feature Selection

- Patterns are generally located in low dimensional manifolds embeded in the input space. **How to find them?**

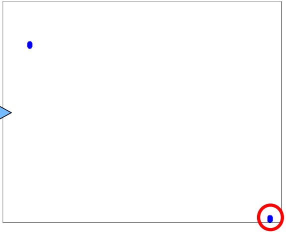
Feature extraction

- Transform the input space into a new space of smaller dimensions.
- Eliminating redundancy and extracting relevant information.
- New features may not have a clear physical meaning.



2D Space

$\mathcal{F}\{\cdot\}$



Representation based on eigenvectors of RBF kernel

Feature Extraction and Feature Selection

- Patterns are generally located in low dimensional manifolds embeded in the input space. **How to find them?**

Feature selection

- Identify a sub-set of m out of the d original features.
- Optimizing some performance criterion (e.g. max correlation).
- Heuristics to search for optimal sub-sets.

Feature Extraction and Feature Selection

- Patterns are generally located in low dimensional manifolds embeded in the input space. **How to find them?**

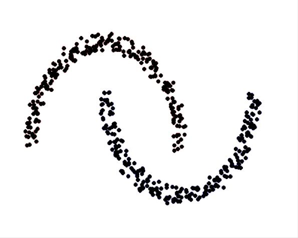
Feature selection

- Identify a sub-set of m out of the d original features.
- Optimizing some performance criterion (e.g. max correlation).
- Heuristics to search for optimal sub-sets.

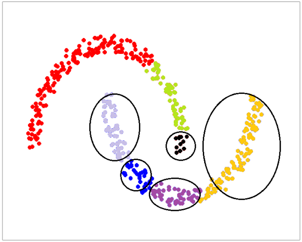
- Problem of feature extraction and selection in Clustering: **we do not have the ground truth**.
- The very nature of clustering means that in many cases, **we know little about the clusters to uncover**.

Which Algorithm?

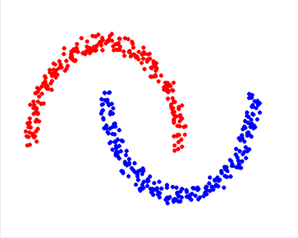
- Each algorithm imposes a structure on data.
- Good fit between model and data $\longrightarrow$ success.



2 Semi-Rings



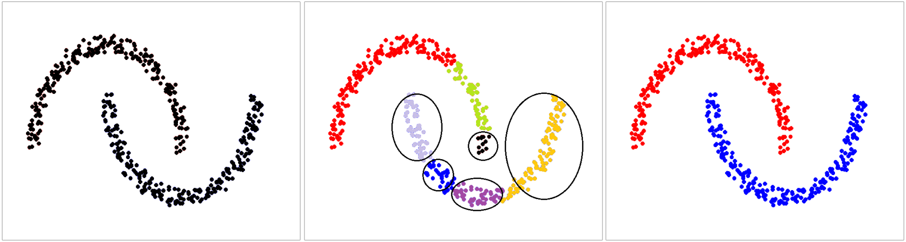
Spherical-based Clustering



Density-based Clustering

Which Algorithm?

- Each algorithm imposes a structure on data.
- Good fit between model and data $\longrightarrow$ success.

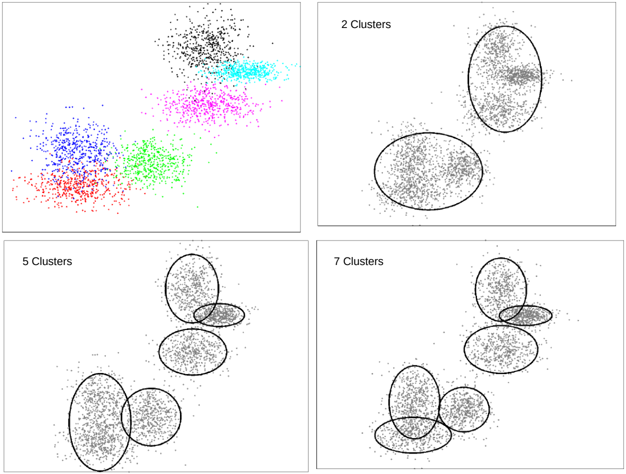


2 Semi-Rings | Spherical-based Clustering | Density-based Clustering

- There is no *silver bullet* in Clustering.

How Many Clusters?

- Some algorithms need the number of clusters as input.
- Difficult to know, requires knowledge on the structure of data.



The most well-known partitioning algorithm: k-means

The k-means algorithm separates the n patterns $p_j \in S$ in k clusters (predefined number), iteratively assigning p_j to the *closest cluster*.

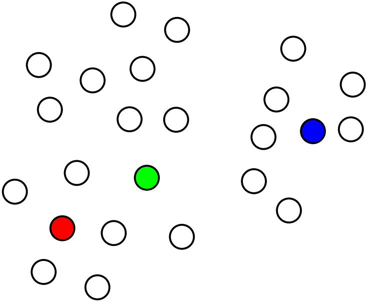
The algorithm:

1. Select an initial random partition in k clusters.
2. Compute the centroids μ_i, $i = 1, \ldots, k$ of each cluster.
3. For each p_j, (re)assign it to the cluster which minimizes distance to μ_i.
4. Continue until no re-assignations are possible.

The most well-known partitioning algorithm: k-means

The algorithm:

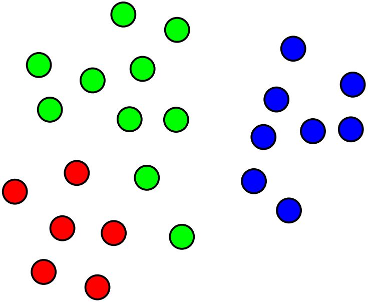
1. Select an initial random partition in k clusters.
2. Compute the centroids μ_i, $i = 1, \ldots, k$ of each cluster.
3. For each p_j, (re)assign it to the cluster which minimizes distance to μ_i.
4. Continue until no re-assignations are possible.



The most well-known partitioning algorithm: k-means

The algorithm:

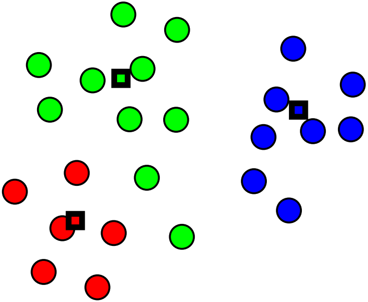
1. Select an initial random partition in k clusters.
2. Compute the centroids μ_i, $i = 1, \ldots, k$ of each cluster.
3. For each p_j, (re)assign it to the cluster which minimizes distance to μ_i.
4. Continue until no re-assignations are possible.



The most well-known partitioning algorithm: k-means

The algorithm:

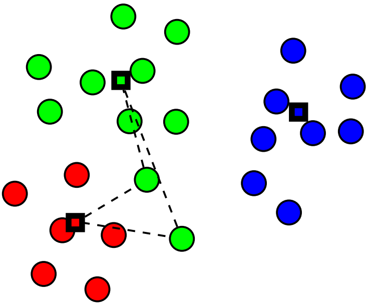
1. Select an initial random partition in k clusters.
2. Compute the centroids μ_i, $i = 1, \ldots, k$ of each cluster.
3. For each p_j, (re)assign it to the cluster which minimizes distance to μ_i.
4. Continue until no re-assignations are possible.



The most well-known partitioning algorithm: k-means

The algorithm:

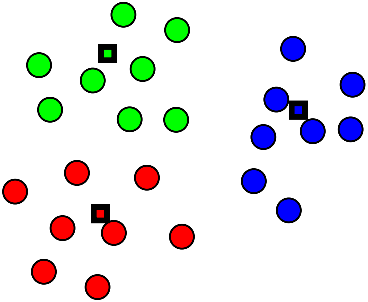
1. Select an initial random partition in k clusters.
2. Compute the centroids μ_i, $i = 1, \ldots, k$ of each cluster.
3. For each p_j, (re)assign it to the cluster which minimizes distance to μ_i.
4. Continue until no re-assignations are possible.



The most well-known partitioning algorithm: k-means

The algorithm:

1. Select an initial random partition in k clusters.
2. Compute the centroids μ_i, $i = 1, \ldots, k$ of each cluster.
3. For each p_j, (re)assign it to the cluster which minimizes distance to μ_i.
4. Continue until no re-assignations are possible.



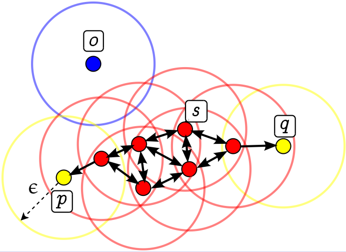
DBSCAN: a density-based notion of clusters

DBSCAN identifies clusters using a notion of density: clusters are high-density regions separated by low-density regions.

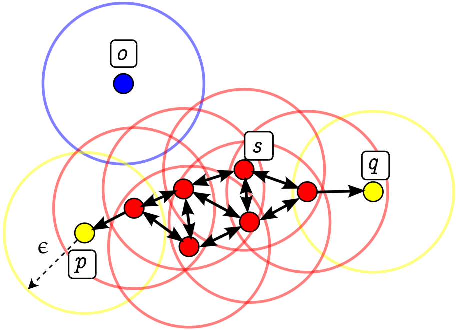
DBSCAN: a density-based notion of clusters

The notion of density in DBSCAN:

1. Two parameters: search distance ϵ and minimum cluster size m.
2. The ϵ-neighborhood of pattern p, $N_\epsilon(p)$ is the set of q_i closer than ϵ.
3. p is *directly density reachable* from q if $p \in N_\epsilon(q)$ and $\#N_\epsilon(q) > m$.
4. p is *density reachable* (dr) from q if there is a chain of inter-directly density reachable patterns between them.
5. p and q are *density connected* (dc) is there is s such that both p and q are (dr) from s.



DBSCAN: a density-based notion of clusters



A DBSCAN cluster C_i is a sub-set of S satisfying the following conditions:

- $\forall\, p, q$: if $p \in C_i$ and q is dr from $p \rightarrow q \in C_i$.
- $\forall\, p, q \in C_i$, p and q are dc.
- Any pattern o_j not belonging to any cluster C_i is defined as *noise* (**outliers**).

k-means vs DBSCAN

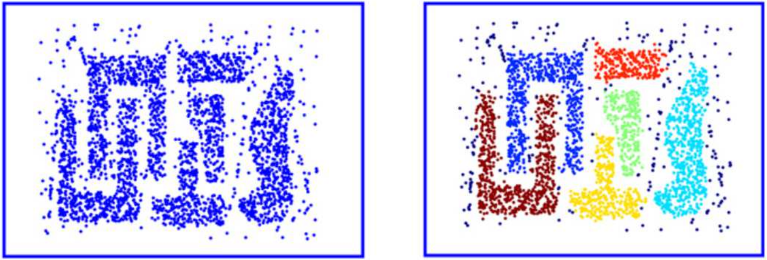
Which is "better"?

- k-means is faster than DBSCAN (multiple implementations of both algorithms improve computational time).

k-means vs DBSCAN

Which is "better"?

- k-means is faster than DBSCAN (multiple implementations of both algorithms improve computational time).
- k-means works well only for spherical-like clusters.
- DBSCAN finds clusters of arbitrary shapes and sizes.



k-means vs DBSCAN

Which is "better"?

- The number of clases k must be defined a-priori (heuristics).
- DBSCAN does not need to know the number of classes.

k-means vs DBSCAN

Which is “better”?

- The number of clases k must be defined a-priori (heuristics).
- DBSCAN does not need to know the number of classes.
- k-means is very sensitive to the initial conditions (heuristics).
- DBSCAN is very sensitive to ϵ in data sets with large differences in densities.

k-means vs DBSCAN

Which is “better”?

- The number of clases k must be defined a-priori (heuristics).
- DBSCAN does not need to know the number of classes.
- k-means is very sensitive to the initial conditions (heuristics).
- DBSCAN is very sensitive to ϵ in data sets with large differences in densities.
- DBSCAN is deterministic, k-means depends on the initial conditions.

k-means vs DBSCAN

Which is "better"?

- The number of clases k must be defined a-priori (heuristics).
- DBSCAN does not need to know the number of classes.
- k-means is very sensitive to the initial conditions (heuristics).
- DBSCAN is very sensitive to ϵ in data sets with large differences in densities.
- DBSCAN is deterministic, k-means depends on the initial conditions.
- DBSCAN uses the notion of outliers (heuristics in k-means).

High Dimensional Data

- In multiple data analysis problems, we have to deal with **high dimensional** and **massive datasets**.

Clustering High Dimensional Data

High Dimensional Data

- In multiple data analysis problems, we have to deal with **high dimensional** and **massive datasets**.
- **Heuristics** to scale-up with the size of the datasets.

High Dimensional Data

- In multiple data analysis problems, we have to deal with **high dimensional** and **massive datasets**.
- **Heuristics** to scale-up with the size of the datasets.
- High dimensional data is more and more common in Networking.

Clustering High Dimensional Data

High Dimensional Data

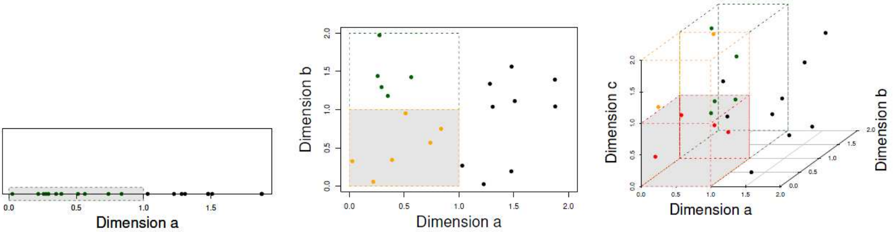
- In multiple data analysis problems, we have to deal with **high dimensional** and **massive datasets**.
- **Heuristics** to scale-up with the size of the datasets.
- High dimensional data is more and more common in Networking.

Clustering high dimensional data is challenging:

- Structure-masking by irrelevant features (i.e., noise).
- **The Curse of Dimensionality**

The Curse of Dimensionality

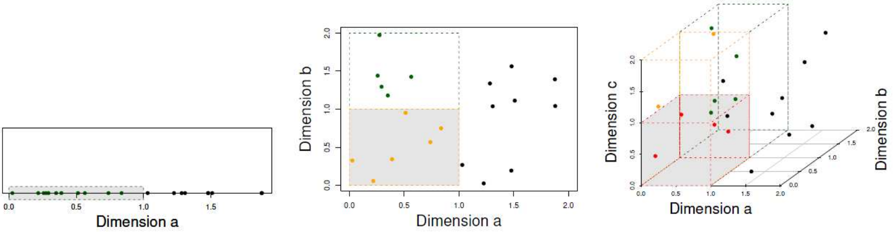
- The term was first coined by Bellman in 1961 to refer to multiple problems associated with high-dimensional data analysis.
- When **dimensionality increases**, the volume of the space increases so fast that the available **data becomes sparse**.



(a) 11 Objects in One Unit Bin (b) 6 Objects in One Unit Bin (c) 4 Objects in One Unit Bin

The Curse of Dimensionality

- The term was first coined by Bellman in 1961 to refer to multiple problems associated with high-dimensional data analysis.
- When **dimensionality increases**, the volume of the space increases so fast that the available **data becomes sparse**.



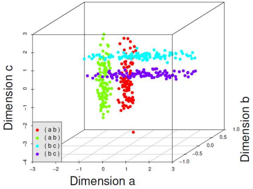
(a) 11 Objects in One Unit Bin (b) 6 Objects in One Unit Bin (c) 4 Objects in One Unit Bin

The notion of cluster in high-dimensional data vanishes:

- Inter-pattern **distance** becomes increasingly **meaningless**.
- Data becomes sparse and **patterns** tend to be **equidistant**.
- **Intuition fails in high dimensions**: the volume of an hyper-sphere is in the shell!

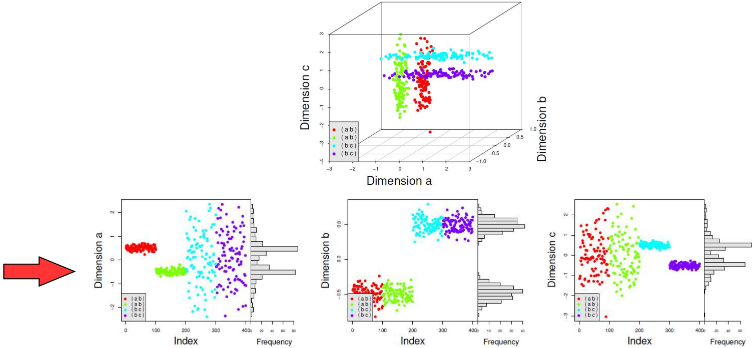
Subspace Clustering - A Graphical Example

The key to find clusters is to identify the correct subspaces:



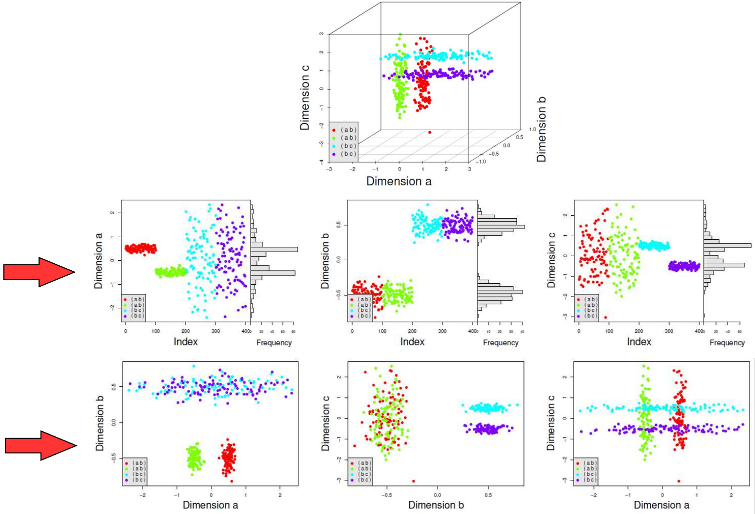
Subspace Clustering - A Graphical Example

The key to find clusters is to identify the correct subspaces:



Subspace Clustering - A Graphical Example

The key to find clusters is to identify the correct subspaces:



(a) Dims a & b (b) Dims b & c (c) Dims a & c

Subspace Clustering (SSC)

SSC: automatically find clusters in different subspaces

- SSC is an approach to do clustering in high-dimensional data.
- An **unsupervised** extension for **feature selection**.
- SSC algorithms **search for relevant dimensions**, finding clusters in multiple, possibly overlapping subspaces.
- SSC algorithms **find low-dimensional clusters in high-dimensional data**.
- SSC algorithms are distinguished by their **search strategy**.

Subspace Clustering (SSC)

SSC: automatically find clusters in different subspaces

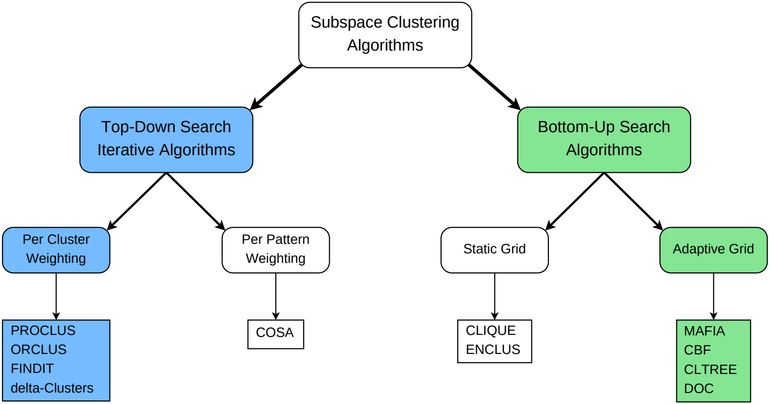
- SSC is an approach to do clustering in high-dimensional data.
- An **unsupervised** extension for **feature selection**.
- SSC algorithms **search for relevant dimensions**, finding clusters in multiple, possibly overlapping subspaces.
- SSC algorithms **find low-dimensional clusters in high-dimensional data**.
- SSC algorithms are distinguished by their **search strategy**.

Two major branches of SSC algorithms:

- **Bottom-Up** search SSC algorithms
- Iterative **Top-Down** search SSC algorithms

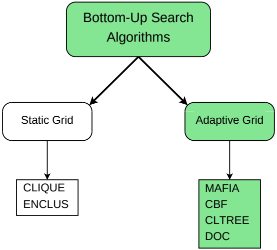
SSC Taxonomy

- **Search heuristics** are optimized for working in massive datasests.
- Different **measures of locality** to recognize clusters in subspaces.



Bottom-Up search SSC

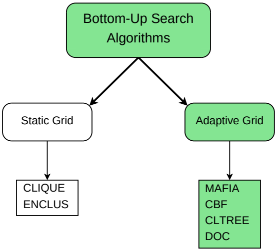
Bottom-Up search



- **Downward closure property** to reduce the search space:

Bottom-Up search SSC

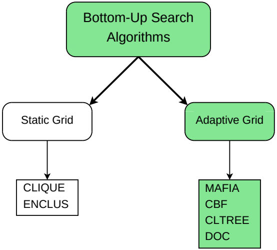
Bottom-Up search



- **Downward closure property** to reduce the search space:
- **Density** in k**-dimensional** space $\longrightarrow$ **density** in all $k-1$ dimensional **projections**.

Bottom-Up search SSC

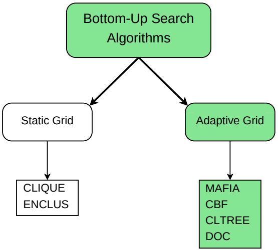
Bottom-Up search



- **Downward closure property** to reduce the search space:
- **Density** in k**-dimensional** space $\longrightarrow$ **density** in all $k-1$ dimensional **projections**.
- Each **dimension is discretized** into bins using a **grid**.

Bottom-Up search SSC

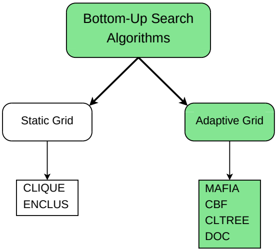
Bottom-Up search



- **Downward closure property** to reduce the search space:
- **Density** in k**-dimensional** space $\longrightarrow$ **density** in all $k-1$ dimensional **projections**.
- Each **dimension is discretized** into bins using a **grid**.
- Start in 1-d spaces: histogram on each dimension to **select the most dense bins** (threshold).

Bottom-Up search SSC

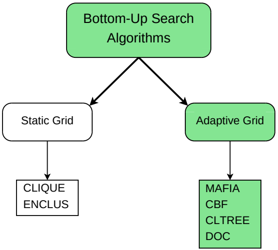
Bottom-Up search



- **Downward closure property** to reduce the search space:
- **Density** in k**-dimensional** space $\longrightarrow$ **density** in all $k-1$ dimensional **projections**.
- Each **dimension is discretized** into bins using a **grid**.
- Start in 1-d spaces: histogram on each dimension to **select the most dense bins** (threshold).
- Build candidate 2-d spaces using only the dimensions with dense bins.

Bottom-Up search SSC

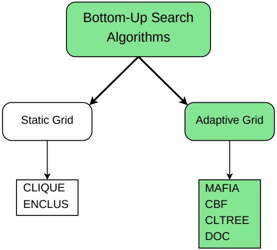
Bottom-Up search



- **Downward closure property** to reduce the search space:
- **Density** in k**-dimensional** space $\longrightarrow$ **density** in all $k-1$ dimensional **projections**.
- Each **dimension is discretized** into bins using a **grid**.
- Start in 1-d spaces: histogram on each dimension to **select the most dense bins** (threshold).
- Build candidate 2-d spaces using only the dimensions with dense bins.
- Stop when **no new higher-dimensional spaces can be added**.

Bottom-Up search SSC

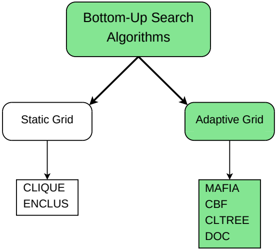
Bottom-Up search



- **Downward closure property** to reduce the search space:
- **Density** in k**-dimensional** space $\longrightarrow$ **density** in all $k-1$ dimensional **projections**.
- Each **dimension is discretized** into bins using a **grid**.
- Start in 1-d spaces: histogram on each dimension to **select the most dense bins** (threshold).
- Build candidate 2-d spaces using only the dimensions with dense bins.
- Stop when **no new higher-dimensional spaces can be added**.
- Different **heuristics to combine and prune dense regions** and form clusters.

Bottom-Up search SSC

Bottom-Up search

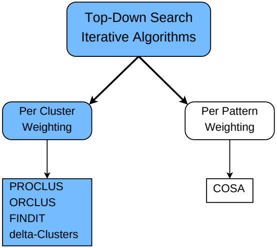


Some observations:

- Bottom-Up algorithms leads to **overlapping clusters**.
- **Grids** can be of **fixed** or **dinamic**, data-based **size**.
- Clusters can be of **arbitrary shape** and size.
- **No need** to specify the **number of clusters** to identify.

Iterative Top-Down search SSC

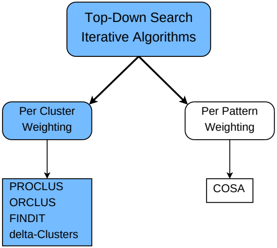
Iterative Top-Down search



- Different algorithms use **different heuristics** and **clustering techniques**.

Iterative Top-Down search SSC

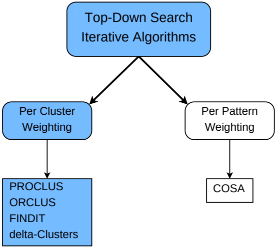
Iterative Top-Down search



- Different algorithms use **different heuristics** and **clustering techniques**.
- Starts by finding an **initial** approximation of the **clusters considering ALL dimensions** (generally using sampling).

Iterative Top-Down search SSC

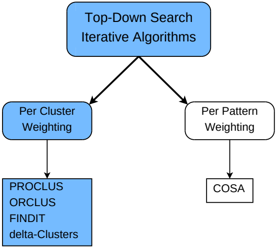
Iterative Top-Down search



- Different algorithms use **different heuristics** and **clustering techniques**.
- Starts by finding an **initial** approximation of the **clusters considering ALL dimensions** (generally using sampling).
- Different **dimensions are weighted** differently according to the **quality of the clusters** (different locality measures).

Iterative Top-Down search SSC

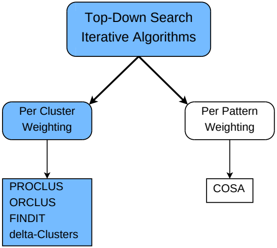
Iterative Top-Down search



- Different algorithms use **different heuristics** and **clustering techniques**.
- Starts by finding an **initial** approximation of the **clusters considering ALL dimensions** (generally using sampling).
- Different **dimensions are weighted** differently according to the **quality of the clusters** (different locality measures).
- **Pruning:** clusters are refined by **selecting the top-weighted dimensions**.

Iterative Top-Down search SSC

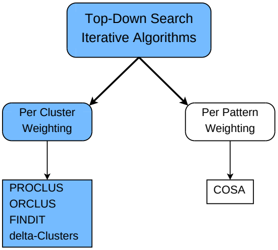
Iterative Top-Down search



- Different algorithms use **different heuristics** and **clustering techniques**.
- Starts by finding an **initial** approximation of the **clusters considering ALL dimensions** (generally using sampling).
- Different **dimensions are weighted** differently according to the **quality of the clusters** (different locality measures).
- **Pruning:** clusters are refined by **selecting the top-weighted dimensions**.
- Different **stopping conditions**, but relative to the **stability of the obtained results** (i.e., no more changes between iterations)

Iterative Top-Down search SSC

Iterative Top-Down search

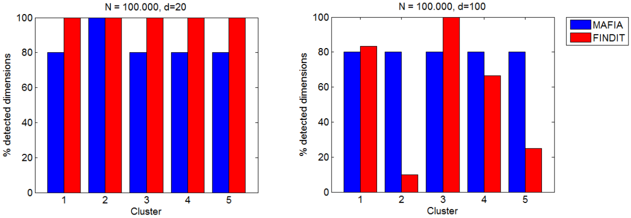


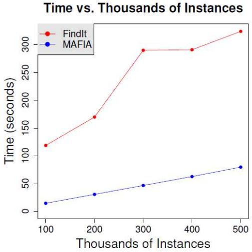
Some observations:

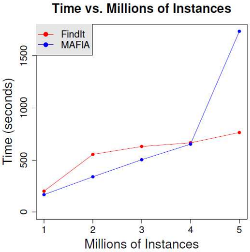
- Top-Down algorithms require to specify the **number of clusters**.
- Tend to **find spherical clusters** in the same or **similar** sized **subspaces**.
- **Sampling** is fundamental to **scale-up** in massive datasets.

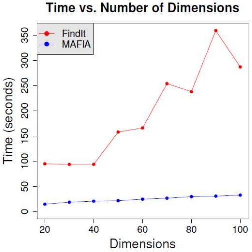
Which SSC Approach to Use?

- Low-dimensional clusters ($k = 2, ..., 7$) embeded in d-dimensional data.
- Evaluate the number of correctly detected dimensions when d increase.
- Evaluate computational time when N = n^{o} patterns and d increase.









Outline

1. What is Machine Learning (ML) and why ML in Networking?
2. General overview on Machine Learning techniques:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Ensemble Learning
3. Features Extraction and Features Selection
   - Feature Extraction
   - Feature Selection
4. Final Remarks: Overfitting and Learning Evaluation
5. Machine Learning in Networking:
   - PSQA: Neural Networks for QoE Assessment
   - Sub-Space Clustering for Self Network Defense

Semi-Supervised Learning: between Supervised and Unsupervised

- In some supervised learning applications we would like to reduce as much as possible the size of labeled data.
- Some applications may provide little information for training issues, but still we would like to use it to improve the analysis.

Semi-Supervised Learning: between Supervised and Unsupervised

- In some supervised learning applications we would like to reduce as much as possible the size of labeled data.
- Some applications may provide little information for training issues, but still we would like to use it to improve the analysis.

In **semi-supervised learning**, we combine a small amount of labeled data with a large amount of unlabeled data for training.

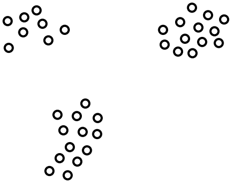
When used in conjunction with a small amount of labeled data, and **under certain assumptions**, unlabeled data can produce considerable improvement in the learning accuracy!

The semi-supervised literature is extensive and there is a whole spectrum of interesting ideas on how to learn from combining labeled and unlabeled data.

Semi-Supervised Learning: between Supervised and Unsupervised

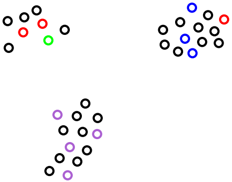
A very intuitive and basic example: build a classifier using clustering and a maximum-likelihood labeling with a small set of labeled flows:

- We first cluster a set of unlabeled patterns.



A very intuitive and basic example: build a classifier using clustering and a maximum-likelihood labeling with a small set of labeled flows:

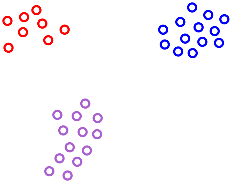
- We first cluster a set of unlabeled patterns.



- Then we consider the labels of a small fraction λ of patterns.

A very intuitive and basic example: build a classifier using clustering and a maximum-likelihood labeling with a small set of labeled flows:

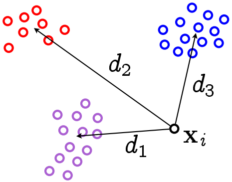
- We first cluster a set of unlabeled patterns.



- Then we consider the labels of a small fraction λ of patterns.
- Maximum-Likelihood Labeling: label each cluster with the most present label among the λ patterns.

A very intuitive and basic example: build a classifier using clustering and a maximum-likelihood labeling with a small set of labeled flows:

- We first cluster a set of unlabeled patterns.



- Then we consider the labels of a small fraction λ of patterns.
- Maximum-Likelihood Labeling: label each cluster with the most present label among the λ patterns.
- Classify an unknown pattern $\mathbf{y}_i$ based on its distance to the centroid of each cluster $\mathbf{o}_k$:

$$l_i = \text{label}\left(\arg\min_k d(\mathbf{x}_i, \mathbf{o}_k)\right)$$

Outline

1. What is Machine Learning (ML) and why ML in Networking?
2. General overview on Machine Learning techniques:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Ensemble Learning
3. Features Extraction and Features Selection
   - Feature Extraction
   - Feature Selection
4. Final Remarks: Overfitting and Learning Evaluation
5. Machine Learning in Networking:
   - PSQA: Neural Networks for QoE Assessment
   - Sub-Space Clustering for Self Network Defense

Ensemble Learning: Combining Multiple Algorithms

Union and diversity provide strength - combining multiple (independent) learnings can be useful in many situations:

- Use different algorithms on the same data to improve performance through diversity.
- Different descriptions of the same problem with different kinds of data (i.e., identify botnets by analyzing flow descriptors, geographical data, dns-based features, etc.).
- Multiple training sets available, collected at different time and different environment (i.e., build a flow classifier with traffic from different ISPs).
- Use the same algorithm with different parametrizations and/or initial conditions (multiple attempts to learn).

Ensemble Learning: Combining Multiple Algorithms

Union and diversity provide strength - combining multiple (independent) learnings can be useful in many situations:

A typical combination scheme consists of an ensemble of individual algorithms and a combiner which merges the results of the individual approaches.

Architecture of combining schemes:

- Parallel combination - individual algorithms are used independently.
- Serial combination - from simple to more complex algorithms.
- Hierarchical combination - refined algorithms for particular data characteristics.

A very large number of ensemble approaches are proposed in the literature.

Outline

1. What is Machine Learning (ML) and why ML in Networking?
2. General overview on Machine Learning techniques:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Ensemble Learning
3. Features Extraction and Features Selection
   - Feature Extraction
   - Feature Selection
4. Final Remarks: Overfitting and Learning Evaluation
5. Machine Learning in Networking:
   - PSQA: Neural Networks for QoE Assessment
   - Sub-Space Clustering for Self Network Defense

Dimensionality Reduction

Using the best and the least features to describe a learning problem is extremely important in Machine Learning. In the feature space terminology, we talk about **Dimensionality Reduction**. And what for?

Dimensionality Reduction

Using the best and the least features to describe a learning problem is extremely important in Machine Learning. In the feature space terminology, we talk about **Dimensionality Reduction**. And what for?

- $\boxed{\text{Improving accuracy of the analysis.}}$
- Reduce measurement costs.
- Create faster systems with less memory constraints.
- Simplify the interpretation of results.

Dimensionality Reduction

Reducing the number of features may lead to a loss in discrimination power, so why performance would degrade when using more features?

- In clustering: working in higher dimensions makes feature spaces become sparser, blurring the notions of similarity.
- In supervised learning: tradeoff between number of features, size of the training set, and algorithm complexity (degrees of freedom).

The Curse of Dimensionality: as the number of features increases, the training set has to increase exponentially to avoid degradations. The more complex the algorithm, the worse it gets.

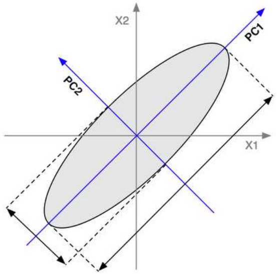
Outline

1. What is Machine Learning (ML) and why ML in Networking?
2. General overview on Machine Learning techniques:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Ensemble Learning
3. Features Extraction and Features Selection
   - Feature Extraction
   - Feature Selection
4. Final Remarks: Overfitting and Learning Evaluation
5. Machine Learning in Networking:
   - PSQA: Neural Networks for QoE Assessment
   - Sub-Space Clustering for Self Network Defense

Feature Extraction

Feature extraction uses a (non)-linear transformation of the feature space into a new space of smaller dimensions, eliminating redundancy and extracting particular information. New features may not have a clear physical meaning.

- Principal Components Analysis (PCA) - standard linear mapping: simple rotation of axes to capture the most of the “energy” of the data.



Other approaches: ICA (linear, assumes independence of sources), kernel PCA (non-linear), SOM (non-linear, based on grids of neurons), etc.

Outline

Feature Selection

Feature selection identifies a sub-set of m out of the d original features, optimizing some performance criterion.

Feature selection consists in two tasks:

- Defining the evaluation criterion used to assess the quality of a sub-set.
- Defining the search strategy to look for the candidate sub-set (heuristic-based search, using graph exploration; optimal exhaustive search is prohibitive!).

Three different approaches for Feature Selection (FS):

- Filter FS: the evaluation criterion is independent of the ML algorithm.
- Wrapper FS: the evaluation criterion is the performance of a certain ML algorithm (i.e., depends o the ML algorithm to be used).
- Embedded FS: the feature selection is part of the ML algorithm itself (e.g., decision trees,).

Feature Selection

Three different approaches for Feature Selection (FS):

- Filter FS: the evaluation criterion is independent of the ML algorithm.
- Wrapper FS: the evaluation criterion is the performance of a certain ML algorithm (i.e., depends o the ML algorithm to be used).
- Embedded FS: the feature selection is part of the ML algorithm itself (e.g., decision trees,).

An example of heuristic-search and filter FS:

- Evaluation criterion - Correlation-based FS (CFS): selects sub-sets with small inter-pattern correlation but highly correlated with the classes.
- Search strategy - Best First search (BF): explores a tree-like graph of features, adding or removing features to improve the criterion; BF permits backtracking to avoid local minima.

Note: “FS can also be done” in clustering $\longrightarrow$ Sub-Space Clustering.

Outline

Some Practical Concepts in Machine Learning

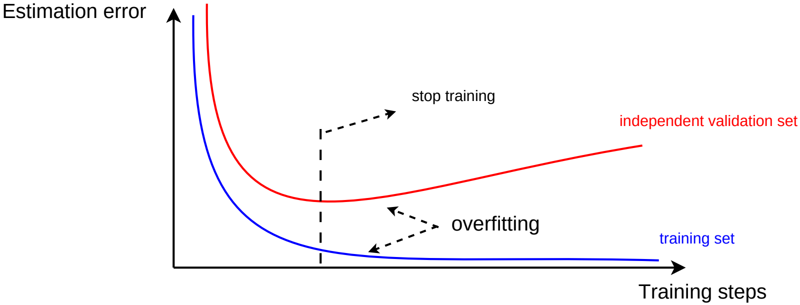
A usual problem in learning is **overfitting**: “learn to remember the training patterns but fail to predict for unseen ones”.

Why overfitting occurs?

- The training set is small w.r.t. the number of parameters to estimate (excessively complex models).
- The number of features is big w.r.t. the size of the training set (curse of dimensionality).
- The training procedure is not stopped at the right moment (“learn” the training set).

Avoiding overfitting

- **Early stopping**: stop the training when the algorithm stops learning the underlying model.
- Train in a sub-set of the training set S, evaluate the predictive expression with the rest of the patterns.



Avoiding overfitting

- k**-fold cross validation**: split the training set in k separated sub-sets.
- Learn from $k-1$ sub-sets, evaluate in the remaining set.
- Rotate sub-sets until covering all of them.

Avoiding overfitting

- k**-fold cross validation**: split the training set in k separated sub-sets.
- Learn from $k-1$ sub-sets, evaluate in the remaining set.
- Rotate sub-sets until covering all of them.

Rule of thumb: use at least 10 times as many training patterns per class n_i as the number of features d:

$$n_i/d > 10$$

The more complex the machine learning model, the larger this ratio should be.

Evaluation of a Machine Learning algorithm

The evaluation of a machine learning algorithm depends on the particular learning approach and on the specific application:

- **Classification**: true positives, false positives, global accuracy, recall, precision, ROC curves.
- **Regression**: estimation/prediction error.
- **Clustering**: cluster homogeneity, number of clusters, outliers analysis.

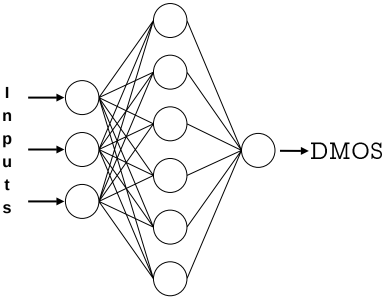
- Always **favor proper and focused evaluations** (less is more).
- **Don't forget sensitivity analysis**: it is easy to find particular cases, but if you want to get useful results, provide robust analysis.

Outline

Outline

PSQA: Neural Networks for QoE Assessment

The **Pseudo-Subjective Quality Assessment** approach (Gerardo Rubino, INRIA/IRISA, France) relies on Neural Networks (NN) to build an estimation model for QoE in multimedia services:

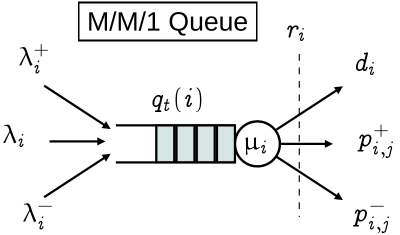


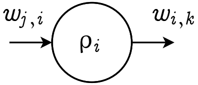
- PSQA uses a particular NN model: Random Neural Networks (RNN).
- Inputs: QoS network features $\{x_n\}$ and sequence characteristics $\{y_m\}$.
- Training step, using subjective tests and inputs $(\{x_i\}, \{y_i\}, \mathrm{DMOS})$.

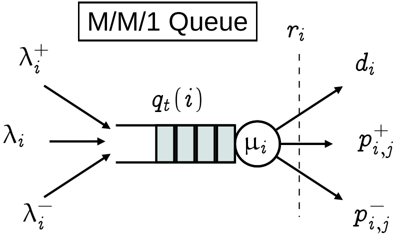
PSQA mapping function:

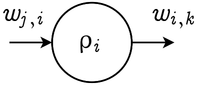
$$\boxed{\mathrm{DMOS} = \mathcal{F}\left(\{x_1, ., x_n\}, \{y_1, ., y_m\}\right)}$$

The Random Neuron Model



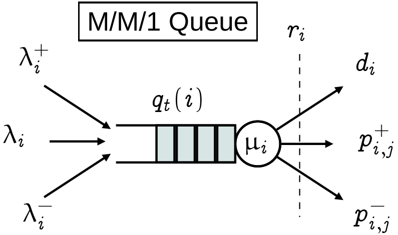


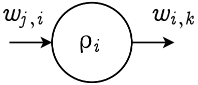




$$d_i \quad + \quad \sum_{j=1}^{N} \left(p_{i,j}^+ + p_{i,j}^- \right) \; = \; 1$$

The Random Neuron Model

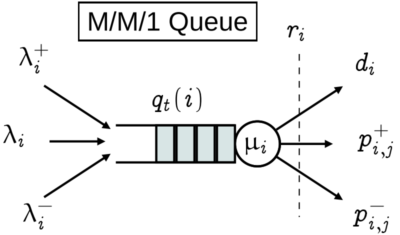


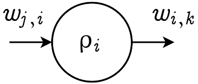


$$d_i + \sum_{j=1}^{N} \left(p_{i,j}^{+} + p_{i,j}^{-} \right) = 1$$

$$\rho_i = \lim_{t \to \infty} \Pr \left(q_t(i) > 0 \right)$$

The Random Neuron Model





$$d_i \quad + \quad \sum_{j=1}^{N} \left(p_{i,j}^+ + p_{i,j}^- \right) \; = \; 1$$

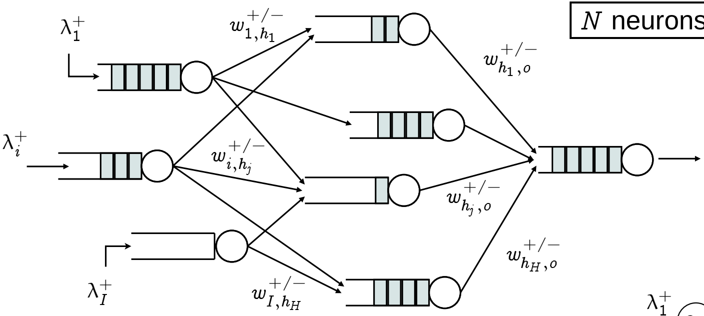
$$\rho_i \quad = \quad \lim_{t \to \infty} \text{Pr} \left(q_t(i) > 0 \right)$$

$$\rho_i \quad = \quad \frac{\lambda_i}{\mu_i}$$

$$\lambda_i \quad = \quad \lambda_i^+ + \sum_{j=1}^{N} \rho_j \, r_j \, p_{j,i}^+$$

$$\mu_i \quad = \quad r_i + \lambda_i^- + \sum_{j=1}^{N} \rho_j \, r_j \, p_{j,i}^-$$

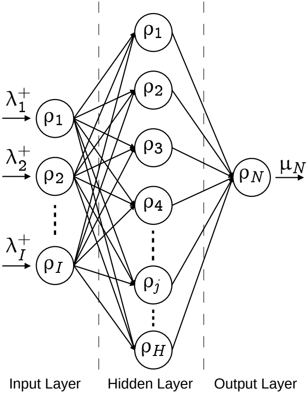
The Random Neural Network Model



N neurons

$$
\begin{aligned}
\lambda_i^- &= 0,\ \forall i \\
w_{i,j}^+ &= r_i p_{i,j}^+ \\
w_{i,j}^- &= r_i p_{i,j}^-
\end{aligned}
$$

$2H(I+1)$ weights to calibrate
gradient descent



Using the RNN for QoE Estimation

3-layer Feed Forward RNN Model:

$$\rho_i = \frac{\lambda_i^+}{r_i} \qquad \forall \text{ input neuron } i$$

$$\rho_h = \frac{\sum_{\text{input neuron } i} \rho_i w_{i,h}^+}{r_h + \sum_{\text{input neuron } i} \rho_i w_{i,h}^-} \qquad \forall \text{ hidden neuron } h$$

$$\rho_o = \frac{\sum_{\text{hidden neuron } h} \rho_h w_{h,o}^+}{r_o + \sum_{\text{hidden neuron } h} \rho_h w_{h,o}^-} \qquad o \equiv N$$

Using the RNN for QoE Estimation

3-layer Feed Forward RNN Model:

$$\rho_i = \frac{\lambda_i^+}{r_i} \qquad \forall \text{ input neuron } i$$

$$\rho_h = \frac{\sum_{\text{input neuron } i} \rho_i w_{i,h}^+}{r_h + \sum_{\text{input neuron } i} \rho_i w_{i,h}^-} \qquad \forall \text{ hidden neuron } h$$

$$\rho_o = \frac{\sum_{\text{hidden neuron } h} \rho_h w_{h,o}^+}{r_o + \sum_{\text{hidden neuron } h} \rho_h w_{h,o}^-} \qquad o \equiv N$$

$$d_i = 0,\ \forall i \neq o, \qquad d_o = 1, \qquad r_o \rightarrow \text{free design parameter}$$

Using the RNN for QoE Estimation

3-layer Feed Forward RNN Model:

$$\rho_i = \frac{\lambda_i^+}{r_i} \qquad \forall \text{ input neuron } i$$

$$\rho_h = \frac{\sum_{\text{input neuron } i} \rho_i w_{i,h}^+}{r_h + \sum_{\text{input neuron } i} \rho_i w_{i,h}^-} \qquad \forall \text{ hidden neuron } h$$

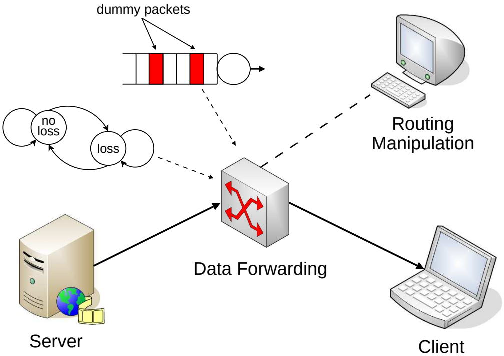
$$\rho_o = \frac{\sum_{\text{hidden neuron } h} \rho_h w_{h,o}^+}{r_o + \sum_{\text{hidden neuron } h} \rho_h w_{h,o}^-} \qquad o \equiv N$$

$$d_i = 0,\ \forall i \neq o, \qquad d_o = 1, \qquad r_o \rightarrow \text{free design parameter}$$

$$r_i = \sum_{\text{hidden neuron } h} \left(w_{i,h}^+ + w_{i,h}^- \right) \qquad \forall \text{ input neuron } i$$

$$r_h = w_{h,o}^+ + w_{h,o}^- \qquad \forall \text{ hidden neuron } h$$

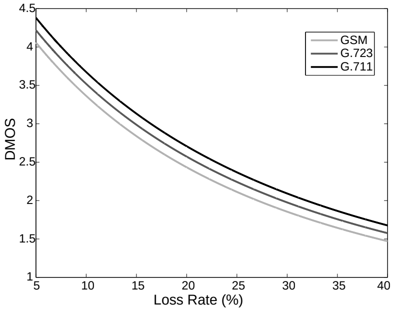
Evaluation Testbed



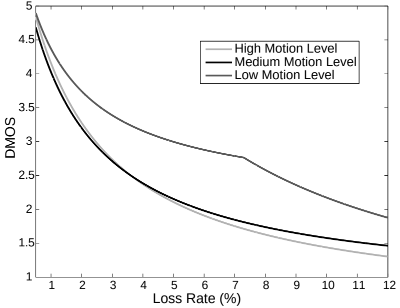
- Intermediate router generates losses and jitter (simple Bernoulli loss model, losses in bursts).
- Short video and audio sequences transmitted from the endpoints.
- Complete Dataset for audio and video, after subjective tests.

QoE analysis through PSQA

DMOS vs loss rate (Mean Loss Burst Length = 5 packets).



(a) Audio Codecs (G.711, G.723, GSM-LPC)

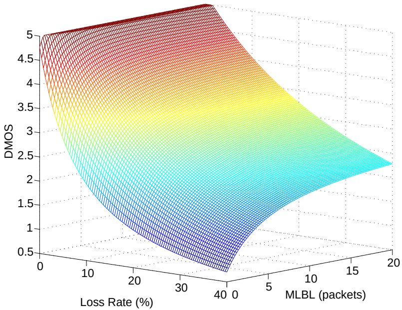


(b) Different video motion levels

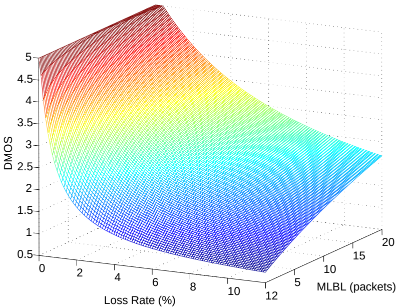
- Audio results are as expected, less impacted by losses than video.
- Video motion level may impact QoE.

QoE analysis through PSQA

$\mathtt{DMOS}$ vs loss rate and mean loss burst length.



(a) Audio Evaluation (G.711 coding)



(b) Video Evaluation (MPEG4 coding)

- QoE in audio is less sensitive to losses than in video (visual system is more developed than the auditory system).
- For the same loss rate, QoE increases with the Mean Loss Burst Length (we prefer concentrated to spread losses).

Outline

Unsupervised NIDS based on Clustering Analysis

We propose a NIDS based on clustering analysis and outliers detection.

The problem to tackle: current network security is based on an **"acquired knowledge"** perspective:

Unsupervised NIDS based on Clustering Analysis

We propose a NIDS based on clustering analysis and outliers detection.

The problem to tackle: current network security is based on an **"acquired knowledge"** perspective:

Signatures-based: detect what I ALREADY KNOW

- $(+)$ highly effective to detect what it is programmed to alert on.
- $(-)$ can not defend the network against unknown attacks.
- $(-)$ signatures are expensive to produce: human manual inspection.

Unsupervised NIDS based on Clustering Analysis

We propose a NIDS based on clustering analysis and outliers detection.

The problem to tackle: current network security is based on an **"acquired knowledge"** perspective:

Signatures-based: detect what I ALREADY KNOW

- $(+)$ highly effective to detect what it is programmed to alert on.
- $(-)$ can not defend the network against unknown attacks.
- $(-)$ signatures are expensive to produce: human manual inspection.

Anomaly detection: detect what DIFFERS from WHAT I KNOW

- $(+)$ it can detect new attacks out-of a baseline profile.
- $(-)$ requires some kind of training for profiling.
- $(-)$ robust and adaptive models are difficult to conceive, specially in an evolving context.

Unsupervised Detection of Network Attacks

- Unsupervised Detection based on CLUSTERING
- HYPOTHESIS: attacking flows are sparse and different from normal traffic....in some representation (traffic aggregation)!!

Unsupervised Detection of Network Attacks

- Unsupervised Detection based on CLUSTERING
- HYPOTHESIS: attacking flows are sparse and different from normal traffic....in some representation (traffic aggregation)!!

Benefits of Unsupervised-based Detection

- no previous knowledge: neither signatures nor labeled traffic.
- no need for traffic modeling or profiling.
- can detect unknown attacks.
- a major step towards self-aware monitoring.

Unsupervised Detection of Network Attacks

- Unsupervised Detection based on CLUSTERING
- HYPOTHESIS: attacking flows are sparse and different from normal traffic....in some representation (traffic aggregation)!!

Benefits of Unsupervised-based Detection

- no previous knowledge: neither signatures nor labeled traffic.
- no need for traffic modeling or profiling.
- can detect unknown attacks.
- a major step towards self-aware monitoring.

Clustering for Unsupervised Detection is CHALLENGING

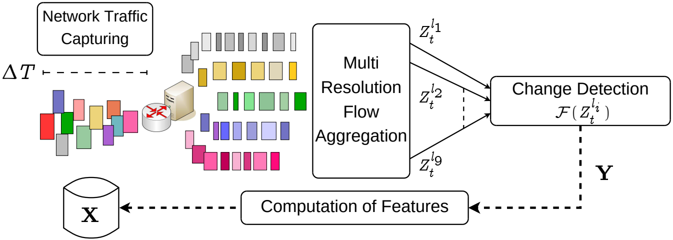
- lack of robustness: general clustering algorithms are sensitive to initialization, specification of number of clusters, etc.
- difficult to cluster high-dimensional data: structure-masking by irrelevant features, sparse spaces ("the curse of dimensionality").

UNADA: Unsupervised Detection of Network Attacks

UNADA is a 3-steps detection algorithm:

UNADA: Unsupervised Detection of Network Attacks

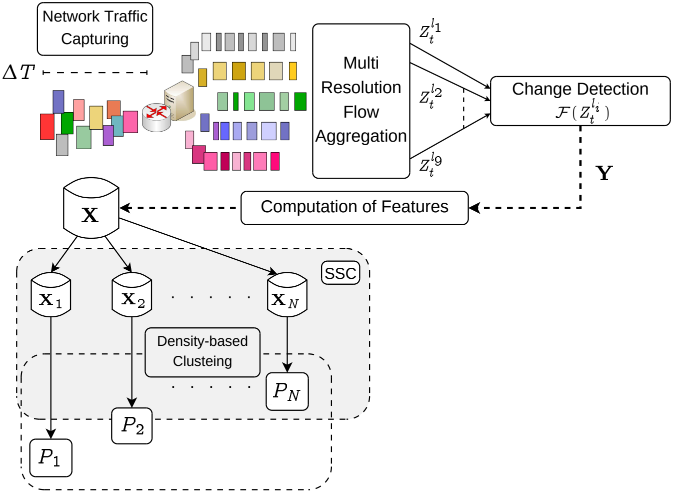
UNADA is a 3-steps detection algorithm:



(1) Multi-resolution change-detection & features computation.

UNADA: Unsupervised Detection of Network Attacks

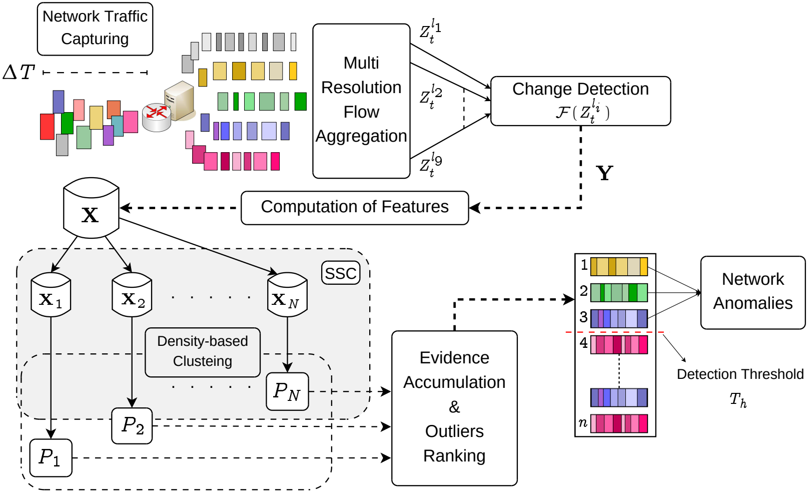
UNADA is a 3-steps detection algorithm:



(2) Sub-Space Clustering.

UNADA: Unsupervised Detection of Network Attacks

UNADA is a 3-steps detection algorithm:



(3) Evidence Accumulation and Flow Ranking.

Change-detection in Multi-resolution Traffic Flows

Traffic Aggregation and Change-Detection

- Traffic is captured and aggregated in IP flows (5-tuples) every ΔT seconds, using a temporal sliding-window.

Change-detection in Multi-resolution Traffic Flows

Traffic Aggregation and Change-Detection

- Traffic is captured and aggregated in IP flows (5-tuples) every ΔT seconds, using a temporal sliding-window.
- Change-detection in simple traffic metrics to identify an anomalous time-slot (e.g., $\#pkts$, $\#bytes$, $\#IP\ flows$).

Change-detection in Multi-resolution Traffic Flows

Traffic Aggregation and Change-Detection

- Traffic is captured and aggregated in IP flows (5-tuples) every ΔT seconds, using a temporal sliding-window.
- Change-detection in simple traffic metrics to identify an anomalous time-slot (e.g., $\#pkts$, $\#bytes$, $\#IP\ flows$).

Multi-Resolution Analysis

- Analysis at different spacial resolutions, aggregating IP flows in *macro-flows*: hash-key $\{\mathtt{IPaddress/netmask}\}$.

Change-detection in Multi-resolution Traffic Flows

Traffic Aggregation and Change-Detection

- Traffic is captured and aggregated in IP flows (5-tuples) every ΔT seconds, using a temporal sliding-window.
- Change-detection in simple traffic metrics to identify an anomalous time-slot (e.g., $\#pkts$, $\#bytes$, $\#IP\ flows$).

Multi-Resolution Analysis

- Analysis at different spacial resolutions, aggregating IP flows in *macro-flows*: hash-key $\{\mathrm{IPaddress/netmask}\}$.
- Scan traffic from coarser to finer-grained macro-flows: $\mathrm{traffic\ per\ time\text{-}slot}$, $\mathrm{IP}/8$, $\mathrm{IP}/16$, $\mathrm{IP}/24$.

Change-detection in Multi-resolution Traffic Flows

Traffic Aggregation and Change-Detection

- Traffic is captured and aggregated in IP flows (5-tuples) every ΔT seconds, using a temporal sliding-window.
- Change-detection in simple traffic metrics to identify an anomalous time-slot (e.g., $\#pkts$, $\#bytes$, $\#IP\ flows$).

Multi-Resolution Analysis

- Analysis at different spacial resolutions, aggregating IP flows in *macro-flows*: hash-key $\{\mathrm{IPaddress/netmask}\}$.
- Scan traffic from coarser to finer-grained macro-flows: $\mathrm{traffic\ per\ time\text{-}slot}$, $\mathrm{IP}/8$, $\mathrm{IP}/16$, $\mathrm{IP}/24$.
- Scan in both directions (IPsrc and IPdst) permits to detect $1\text{-}\mathrm{to}\text{-}1$, $1\text{-}\mathrm{to}\text{-}N$, and $N\text{-}\mathrm{to}\text{-}1$ attacks of different intensities.

Clustering for Anomaly Detection

- Let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the set of n macro-flows in the flagged time slot, aggregated at $\mathrm{IP}/32$.

Clustering for Anomaly Detection

- Let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the set of n macro-flows in the flagged time slot, aggregated at $\mathrm{IP}/32$.
- Each macro-flow $\mathbf{y}_i \in \mathbf{Y}$ is described by a set of m traffic features: $\mathbf{x}_i = (x_i(1), .., x_i(m)) \in \mathbb{R}^m$.

Clustering for Anomaly Detection

- Let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the set of n macro-flows in the flagged time slot, aggregated at $\mathrm{IP}/32$.
- Each macro-flow $\mathbf{y}_i \in \mathbf{Y}$ is described by a set of m traffic features: $\mathbf{x}_i = (x_i(1), .., x_i(m)) \in \mathbb{R}^m$.
- Number of sources & destinations ($\mathrm{nSrcs}, \mathrm{nDsts}$), packet rate ($\mathrm{nPkts/sec}$), fraction of SYN packets ($\mathrm{nSYN/nPkts}$), etc.

Clustering for Anomaly Detection

- Let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the set of n macro-flows in the flagged time slot, aggregated at $\mathrm{IP}/32$.

- Each macro-flow $\mathbf{y}_i \in \mathbf{Y}$ is described by a set of m traffic features: $\mathbf{x}_i = (x_i(1), .., x_i(m)) \in \mathbb{R}^m$.

- Number of sources & destinations ($\mathrm{nSrcs}, \mathrm{nDsts}$), packet rate ($\mathrm{nPkts/sec}$), fraction of SYN packets ($\mathrm{nSYN/nPkts}$), etc.

- $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$ is the complete matrix of features, referred to as the *feature space*.

Clustering for Anomaly Detection

$\mathbf{X}$ is a black box

How to detect an anomalous macro-flow in $\mathbf{X}$ via clustering?

- "Simple idea": cluster $\mathbf{X}$, big-size clusters correspond to normal-flows, outliers are anomalies.

Clustering for Anomaly Detection

$\mathbf{X}$ is a black box

How to detect an anomalous macro-flow in $\mathbf{X}$ via clustering?

- "Simple idea": cluster $\mathbf{X}$, big-size clusters correspond to normal-flows, outliers are anomalies.

Clustering for Anomaly Detection

IS NOT THAT SIMPLE!!!

How to detect an anomalous macro-flow in $\mathbf{X}$ via clustering?

- “Simple idea”: cluster $\mathbf{X}$, big-size clusters correspond to normal-flows, outliers are anomalies.

Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.

Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.

Sub-Space Clustering

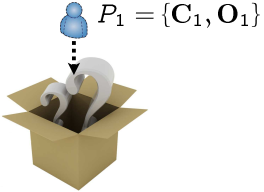
How to Improve Robustness and Clustering Performance?

- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering applied to $\mathbf{X}_i$.

Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

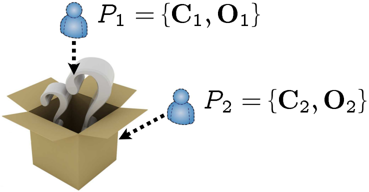
- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering applied to $\mathbf{X}_i$.



Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

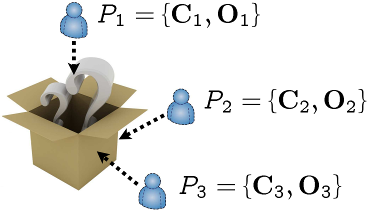
- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering applied to $\mathbf{X}_i$.



Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

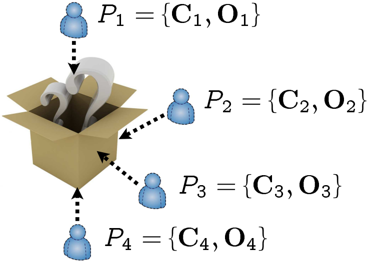
- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering applied to $\mathbf{X}_i$.



Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

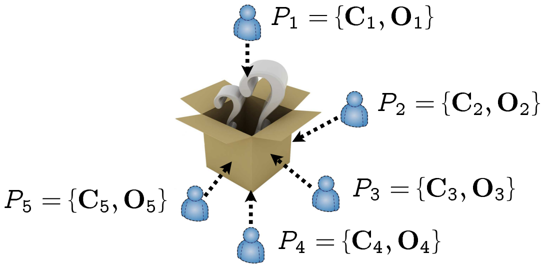
- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering applied to $\mathbf{X}_i$.



Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

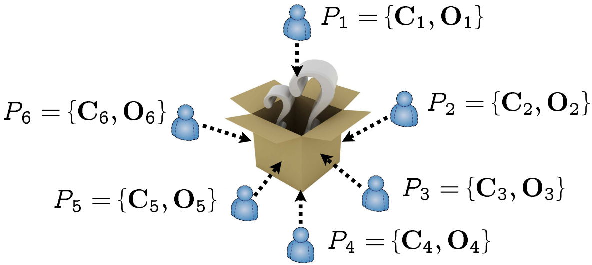
- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering applied to $\mathbf{X}_i$.



Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

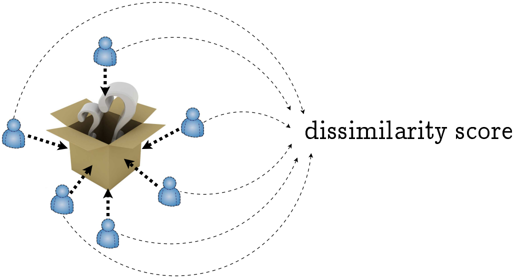
- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering applied to $\mathbf{X}_i$.



Sub-Space Clustering

How to Improve Robustness and Clustering Performance?

- Idea: combine the information provided by multiple partitions of $\mathbf{X}$ to "filter noise", easing the discovery of outliers.
- How to produce multiple partitions? $\rightarrow$ Sub-Space Clustering.
- Each sub-space $\mathbf{X}_i \subset \mathbf{X}$ is obtained by projecting $\mathbf{X}$ in k out of the m original dimensions. Density-based clustering applied to $\mathbf{X}_i$.



Evidence Accumulation for Outliers Ranking

Evidence Accumulation to combine the results of SSC:

- Build a new dissimilarity measure $D = \{d_1, d_2, \ldots, d_n\}$: d_i measures how different is flow i from the majority of the traffic.

Evidence Accumulation for Outliers Ranking

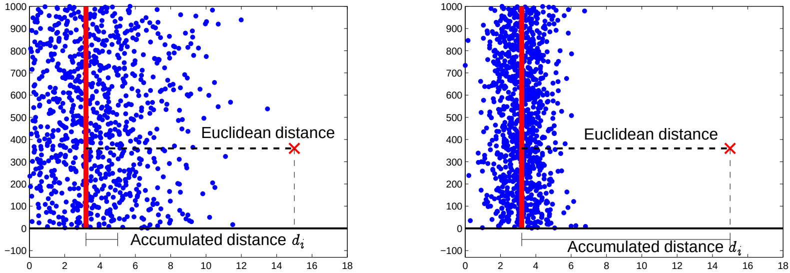
Evidence Accumulation to combine the results of SSC:

- Build a new dissimilarity measure $D = \{d_1, d_2, \ldots, d_n\}$: d_i measures how different is flow i from the majority of the traffic.
- Accumulate in d_i the *weighted* distance from outliers to biggest cluster in each sub-space.

Evidence Accumulation for Outliers Ranking

Evidence Accumulation to combine the results of SSC:

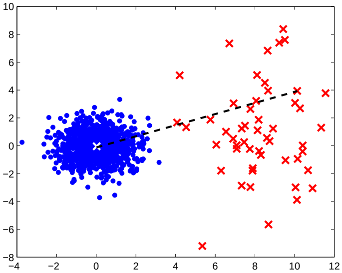
- Build a new dissimilarity measure $D = \{d_1, d_2, \ldots, d_n\}$: d_i measures how different is flow i from the majority of the traffic.
- Accumulate in d_i the *weighted* distance from outliers to biggest cluster in each sub-space.
- Most dissimilar flows w.r.t. D are flagged as anomalies.

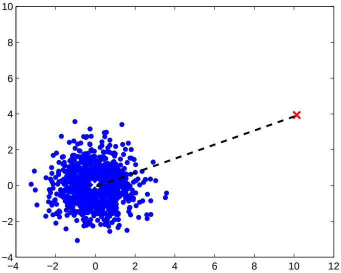


Evidence Accumulation for Outliers Ranking

Evidence Accumulation to combine the results of SSC:

- Build a new dissimilarity measure $D = \{d_1, d_2, \ldots, d_n\}$: d_i measures how different is flow i from the majority of the traffic.
- Accumulate in d_i the *weighted* distance from outliers to biggest cluster in each sub-space.
- Most dissimilar flows w.r.t. D are flagged as anomalies.



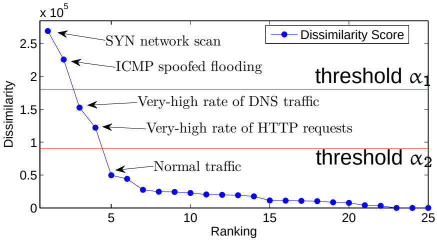


Attacks Detection in MAWI Traffic

- MAWI: packet traces from link Japan-U.S.A. of the WIDE network.
- Ex: worm scanning, ICMP flooding attack, $\mathrm{IPsrc}/32$ macro-flows.

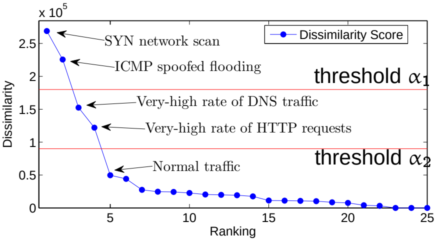
Attacks Detection in MAWI Traffic

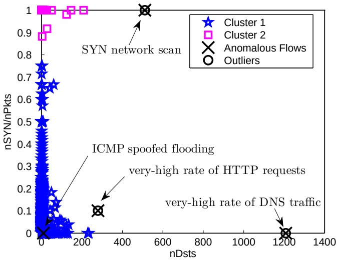
- MAWI: packet traces from link Japan-U.S.A. of the WIDE network.
- Ex: worm scanning, ICMP flooding attack, $\mathtt{IPsrc}/32$ macro-flows.

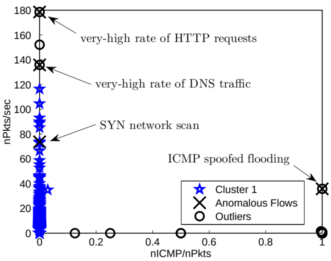


Attacks Detection in MAWI Traffic

- MAWI: packet traces from link Japan-U.S.A. of the WIDE network.
- Ex: worm scanning, ICMP flooding attack, $\mathtt{IPsrc}/32$ macro-flows.







Ground-Truth (GT) Attacks in METROSEC & MAWI

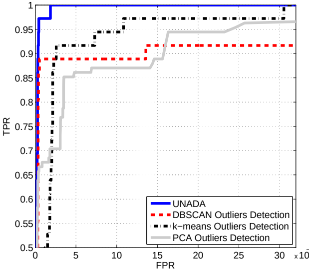
- METROSEC, DDoS attacks of different intensities (70% to 4%), $\mathrm{IPdst}/32$ macro-flows.

Ground-Truth (GT) Attacks in METROSEC & MAWI

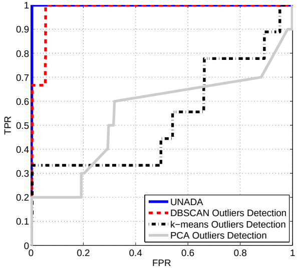
- METROSEC, DDoS attacks of different intensities (70% to 4%), $\mathrm{IPdst}/32$ macro-flows.
- MAWI, worm scanning (Sasser and Dabber), DoS/DDoS attacks, GT attacks detected by signatures + Anomaly Detection.

Ground-Truth (GT) Attacks in METROSEC & MAWI

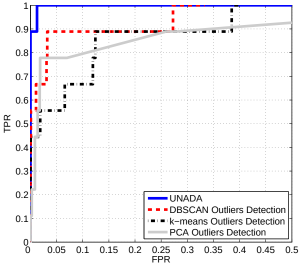
- METROSEC, DDoS attacks of different intensities (70% to 4%), $\mathrm{IPdst}/32$ macro-flows.
- MAWI, worm scanning (Sasser and Dabber), DoS/DDoS attacks, GT attacks detected by signatures + Anomaly Detection.
- Compared against traditional unsupervised approaches: DBSCAN based, k-means based, and PCA based outliers detection.



(a) MAWI, IPsrc key.



(b) MAWI, IPdst key.



(c) METROSEC, IPdst key.

Detectiong Attacks in KDD99

- DARPA - KDD99 dataset, DoS (udp storm, pod, appache flooding, etc.), scans (port, net), Remote-2-Local attacks (guess password, imap, http tunnel, etc.), User-2-Root (buffer overflows).

Detectiong Attacks in KDD99

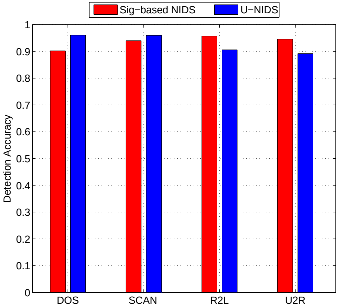
- DARPA - KDD99 dataset, DoS (udp storm, pod, appache flooding, etc.), scans (port, net), Remote-2-Local attacks (guess password, imap, http tunnel, etc.), User-2-Root (buffer overflows).
- Compared against signature-based detection $\rightarrow$ NIDS based on decision trees (C4.5).

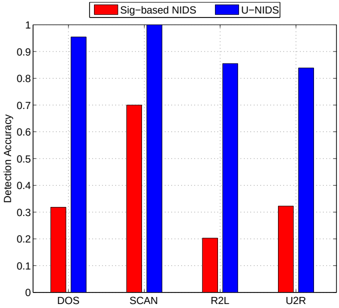
Detectiong Attacks in KDD99

- DARPA - KDD99 dataset, DoS (udp storm, pod, appache flooding, etc.), scans (port, net), Remote-2-Local attacks (guess password, imap, http tunnel, etc.), User-2-Root (buffer overflows).
- Compared against signature-based detection $\rightarrow$ NIDS based on decision trees (C4.5).
- Trees are constructed for a set of known attacks, and tested with *unknown* attacks.

Detectiong Attacks in KDD99

- DARPA - KDD99 dataset, DoS (udp storm, pod, appache flooding, etc.), scans (port, net), Remote-2-Local attacks (guess password, imap, http tunnel, etc.), User-2-Root (buffer overflows).
- Compared against signature-based detection $\longrightarrow$ NIDS based on decision trees (C4.5).
- Trees are constructed for a set of known attacks, and tested with *unknown* attacks.





References

- R. O. Duda, P. E. Hart, and D. G. Stork, “Pattern Classification (2nd edition)”, Wiley, ISBN: 978-0-471-05669-0, 2000.
- C. M. Bishop, “Pattern Recognition and Machine Learning”, Springer-Verlag, ISBN: 0-387-31073-8, 2006.
- A. K. Jain, R. P.W. Duin, and J. Mao “Statistical Pattern Recognition: A Review”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, 2000.
- A. K. Jain, “Data Clustering: 50 Years Beyond K-means”, Pattern Recognition Letters, vol. 31, issue 8, 2010.

Thank You for Your Attention!!

Remarks & Questions?