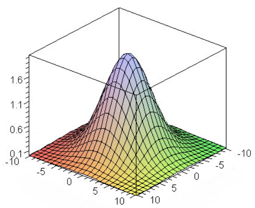
Virtual Things for Machine Learning Applications

Gérôme Bovet[1, 2] – Antonio Ridi[2, 3] – Jean Hennebert[2]
[1]Telecom ParisTech France
[2]University of Applied Sciences Western Switzerland
[3]University of Fribourg Switzerland

gerome.bovet@hefr.ch

Introduction

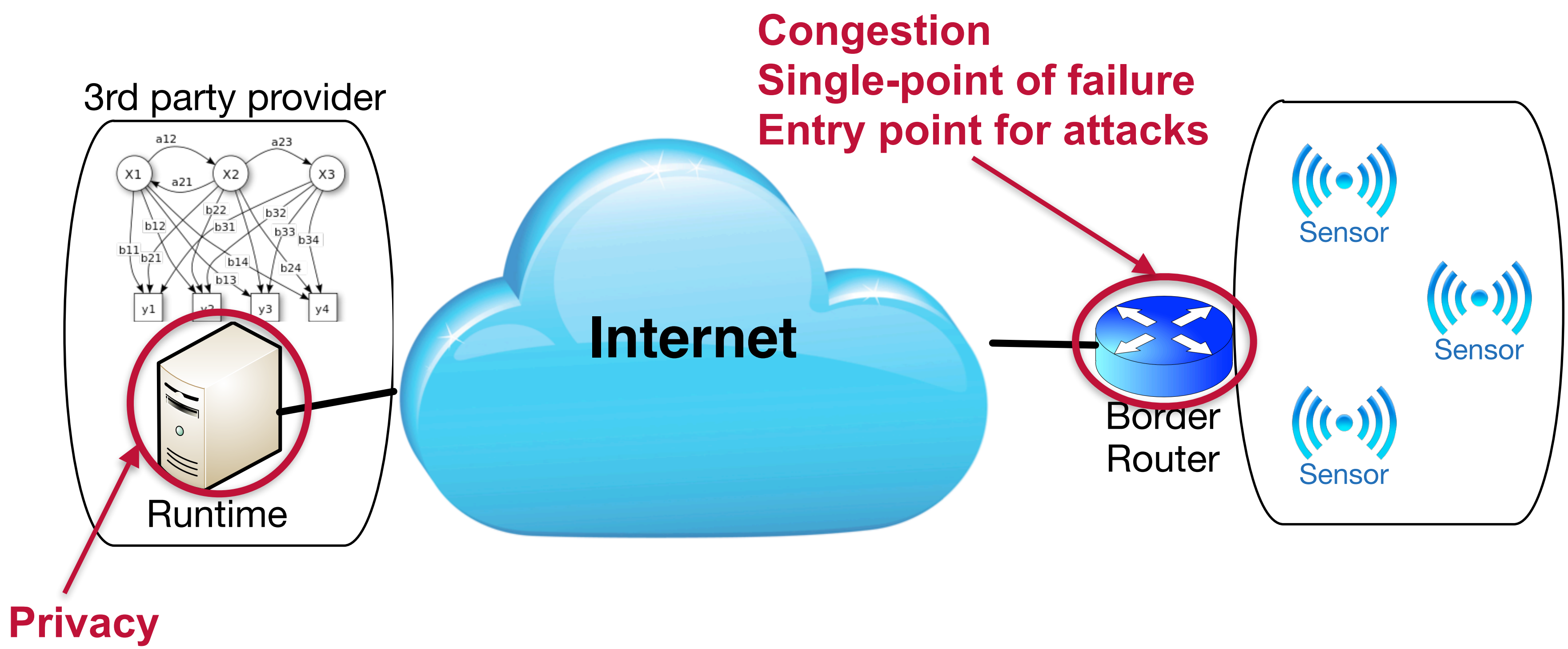


- **Data-driven machine-learning techniques are increasingly used for analyzing sensor data**
  - Trained models performing high accuracy
  - Executed server-side

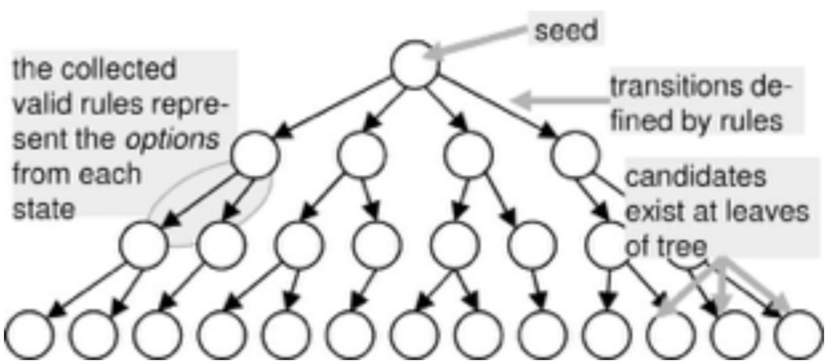
- **Sensing devices are empowered with high computational capabilities**
  - Often underexploited CPU and memory

- **Sensor networks are following the Web-of-Things paradigm**
  - Strong interaction style between heterogeneous devices

Motivations

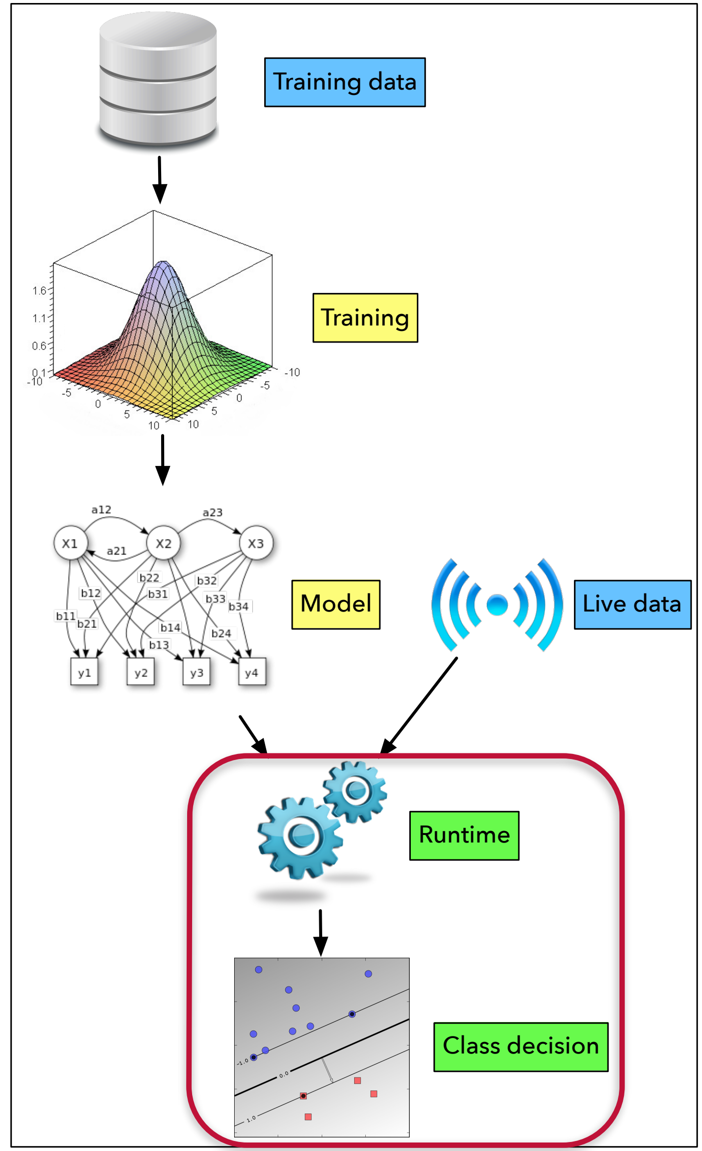


Machine learning



- **Discriminative models**
  - Tree-based decision models
  - Simple to implement
  - Require no particular capabilities

Machine learning process



Machine Learning - Approaches

- **Generative approach: separately model class conditional densities and priors**

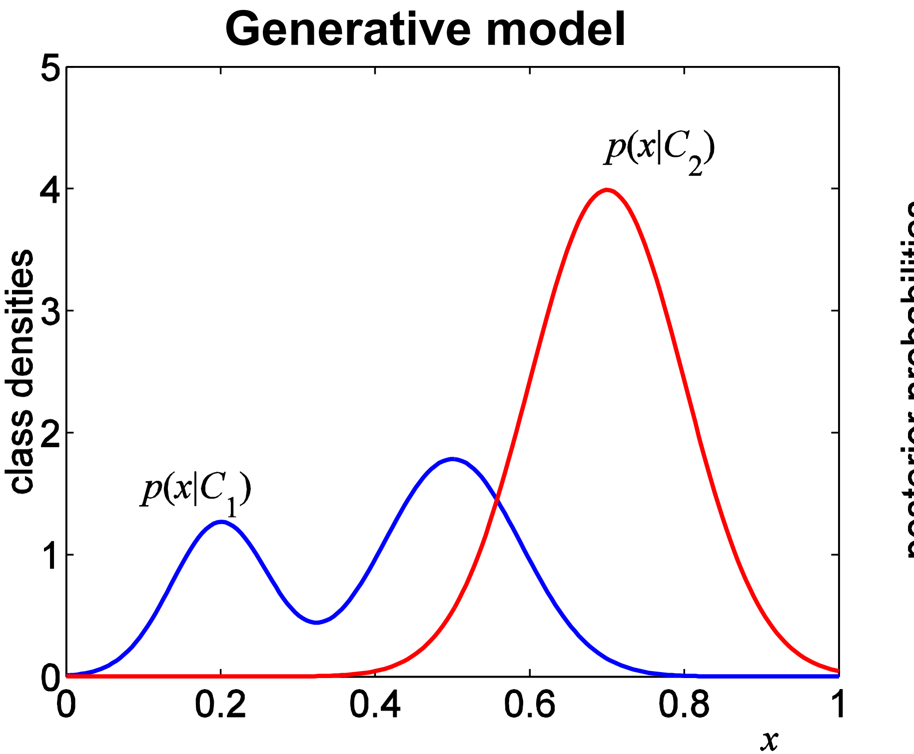
  $$p(\mathbf{x}|\mathcal{C}_k), \qquad p(\mathcal{C}_k)$$

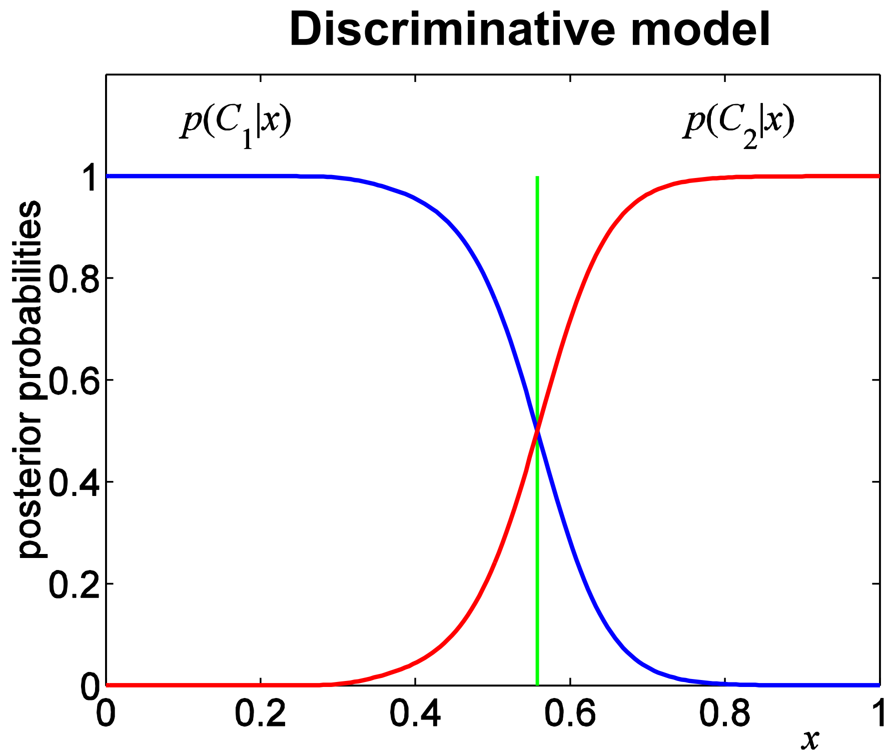
  then evaluate posterior probabilities using Bayes' theorem

  $$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)}$$

- **Discriminative approach: directly model posterior probabilities** $p(\mathcal{C}_k|\mathbf{x})$

Machine Learning - Comparison





Probability density functions as function of *x*

- New classes can be added without re-training using all the data
- Training converges faster
- Need to compute likelihood for each class

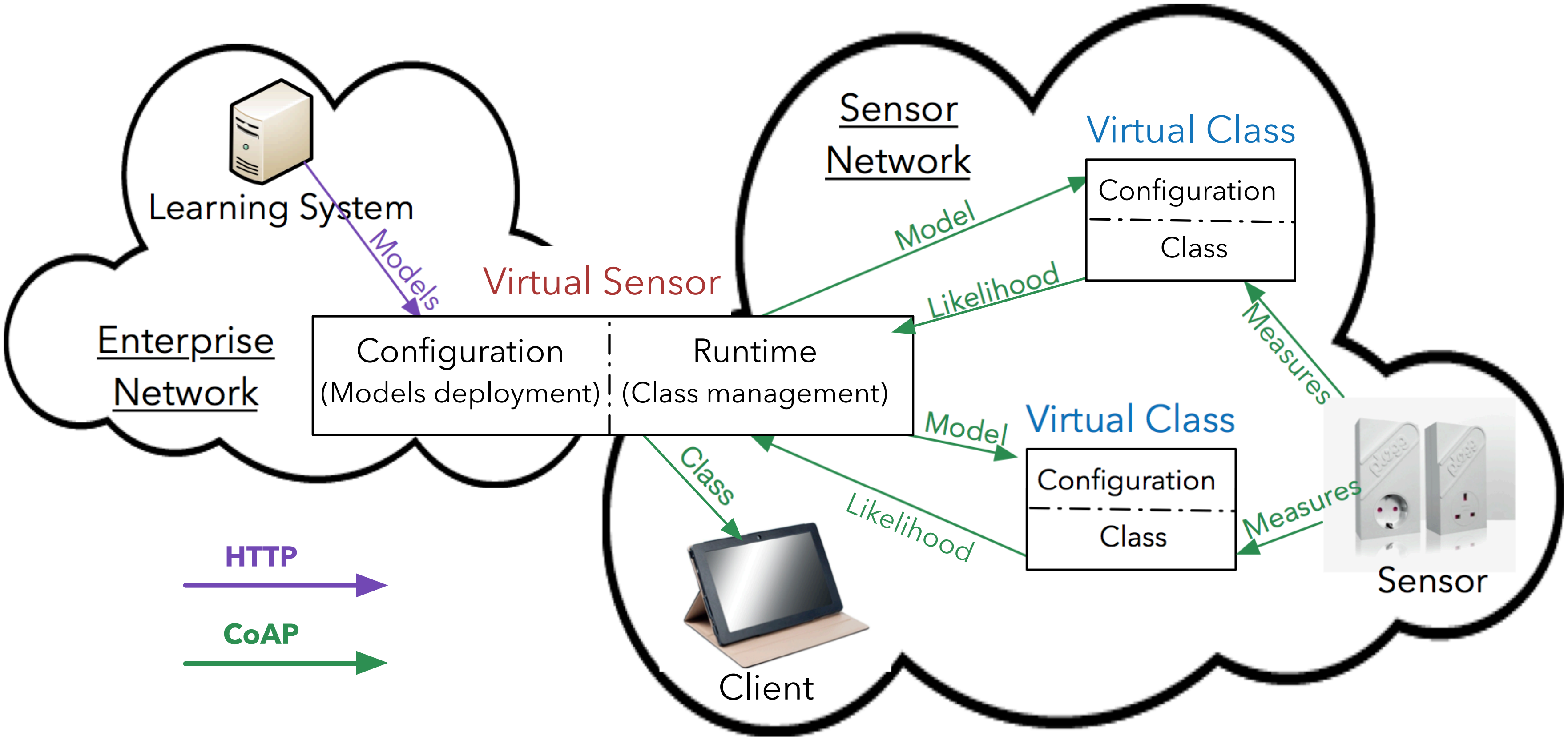
Probabilities as function of *x*

- Very fast once trained
- Training converges slower
- Need to re-train with all the data when adding/removing classes

Architectural Design - Objectives

- **Perform machine-learning within the sensor network**
- **Reuse computational capacities of already installed devices (sensors and actuators)**
- **Inherit and extend key properties of the Web**
  - Strong interaction style
  - Independence regarding hard-/software platforms
  - Scalable architecture
- **Formalize the exchange of machine-learning models**
- **Extend the perception of things to virtual system components**

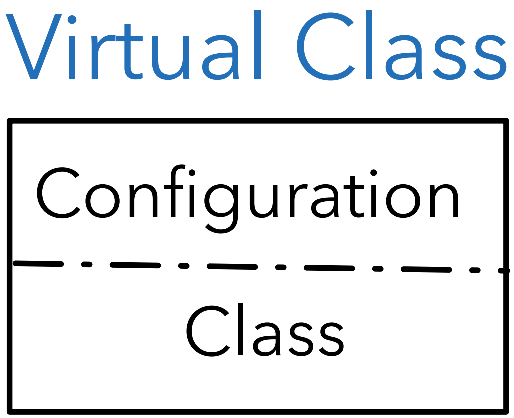
Architecture



Virtual Class

- **Preloaded agent with runtime algorithm (HMM or GMM)**
  - Discovery by location and available capabilities
- **Represents a single class**
  - Output is the likelihood for the class (probability)

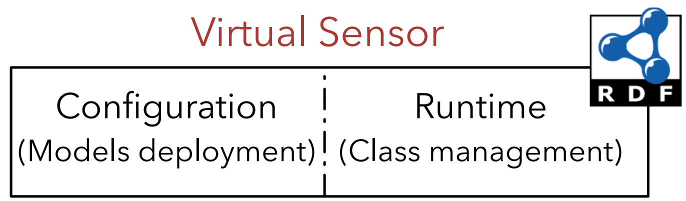
Deploying models in JSON and binary



Observable resource returning the current probability

Virtual Sensor

- **Represents a high-level sensor (non-physical)**
  - Extracts knowledge from multiple sensors of different nature
  - Machine learning tasks abstraction level
  - Reusable component for performing mashups (semantic description of the sensor)



Entry point for deploying machine learning models

Performs validation of the MaLeX input (JSON schema)

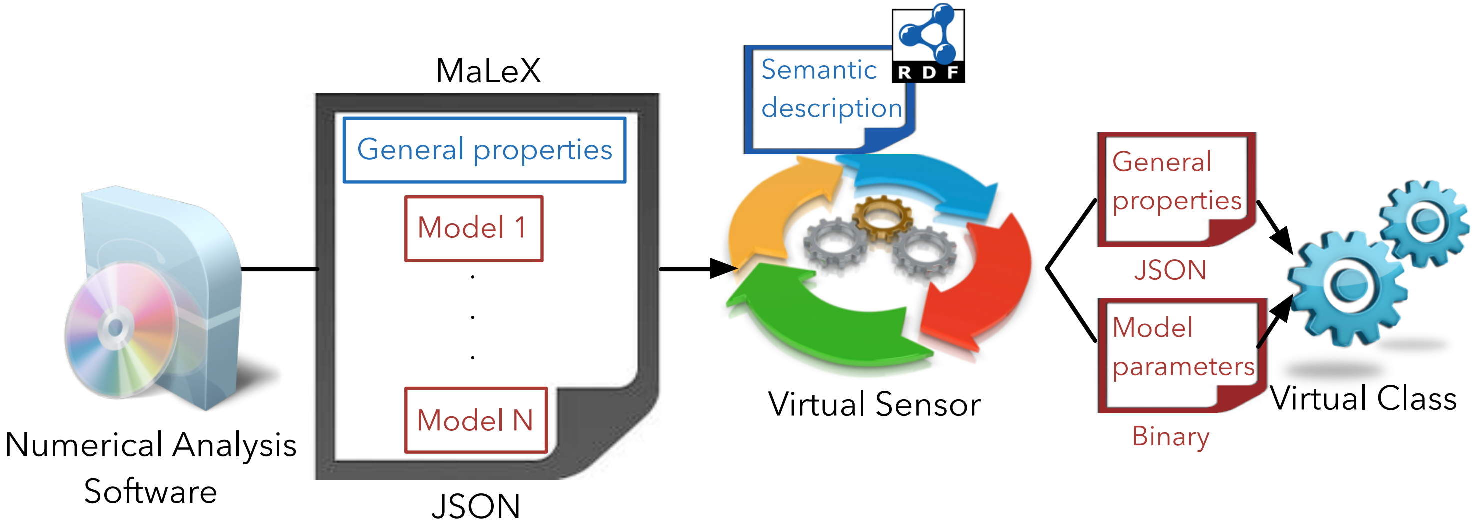
Distributes the classes on nearly located agents

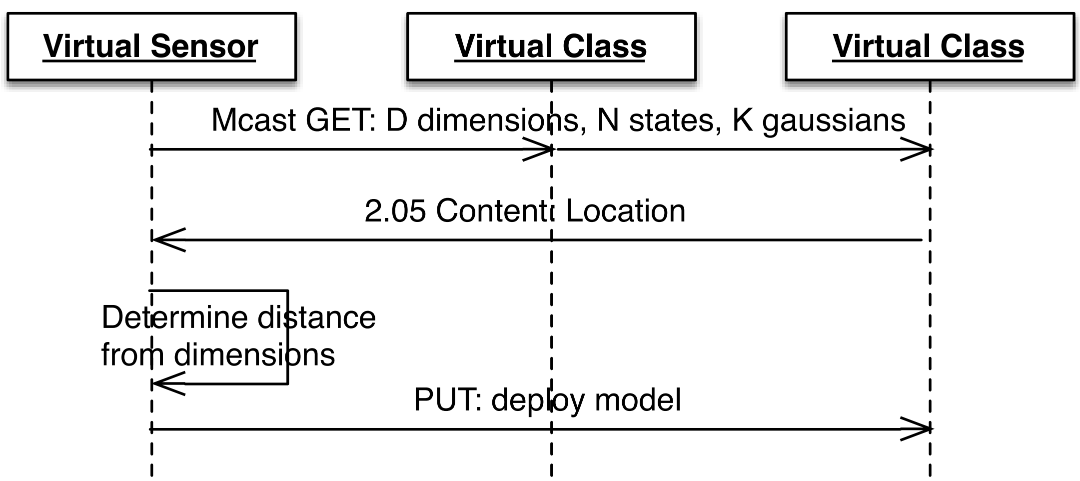
Performs the decision making (class with highest likelihood)

MaLeX - Machine Learning Exchange Format

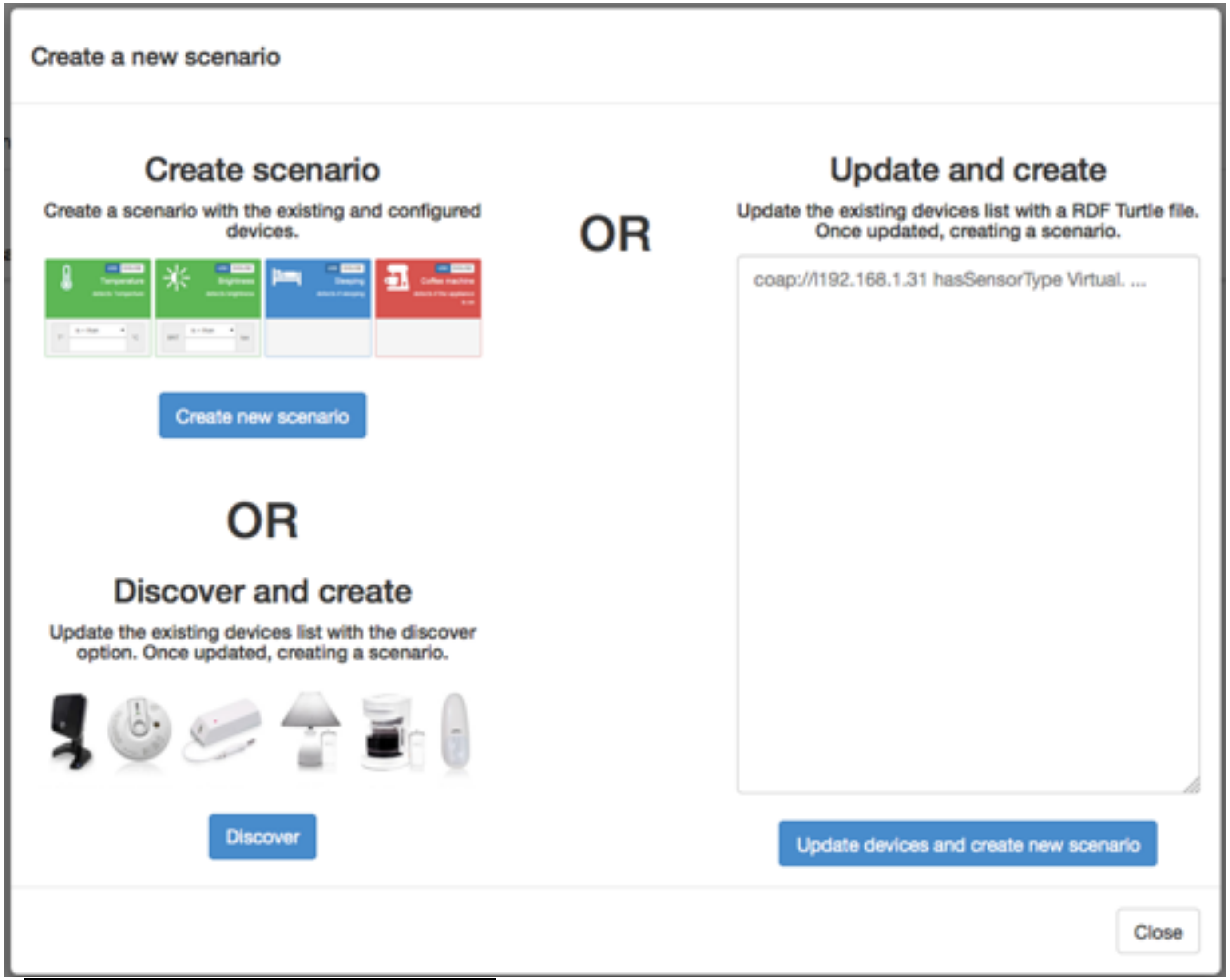
- **Inspired from PMML - Predictive Model Markup Languages (no generative models)**
- **Formalized as JSON schema**
- **Description of general entities**
  - Location, dimensions, type of sensor
- **Description specific to HMM and GMM**
  - Algorithm type, normalisation, states, matrices (mu, sigma, weights, transition)
- **Extensible to other kind of models**

Deployment


MaLeX
General properties
Model 1
Model N
JSON
Numerical Analysis Software
Semantic description
RDF
Virtual Sensor
General properties
JSON
Model parameters
Binary
Virtual Class


Virtual Sensor
Virtual Class
Virtual Class
Mcast GET: D dimensions, N states, K gaussians
2.05 Content: Location
Determine distance from dimensions
PUT: deploy model

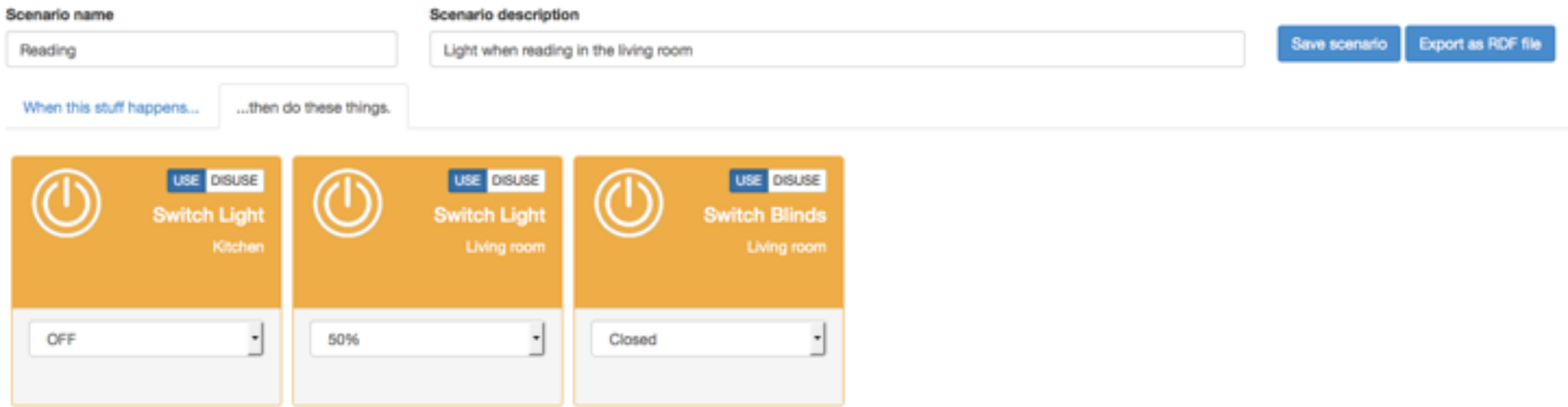
Applications - Building Automation Mashup Editor



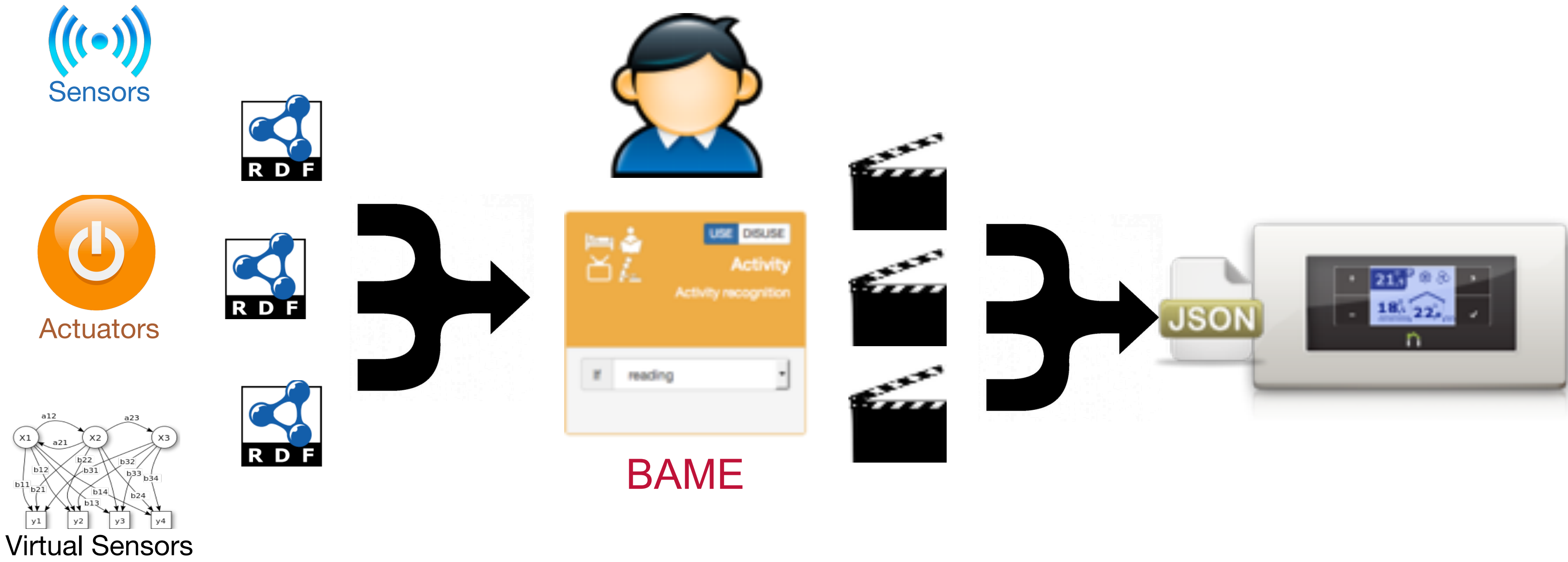
Applications - Building Automation Mashup Editor

TELECOM ParisTech

Applications - Building Automation Mashup Editor



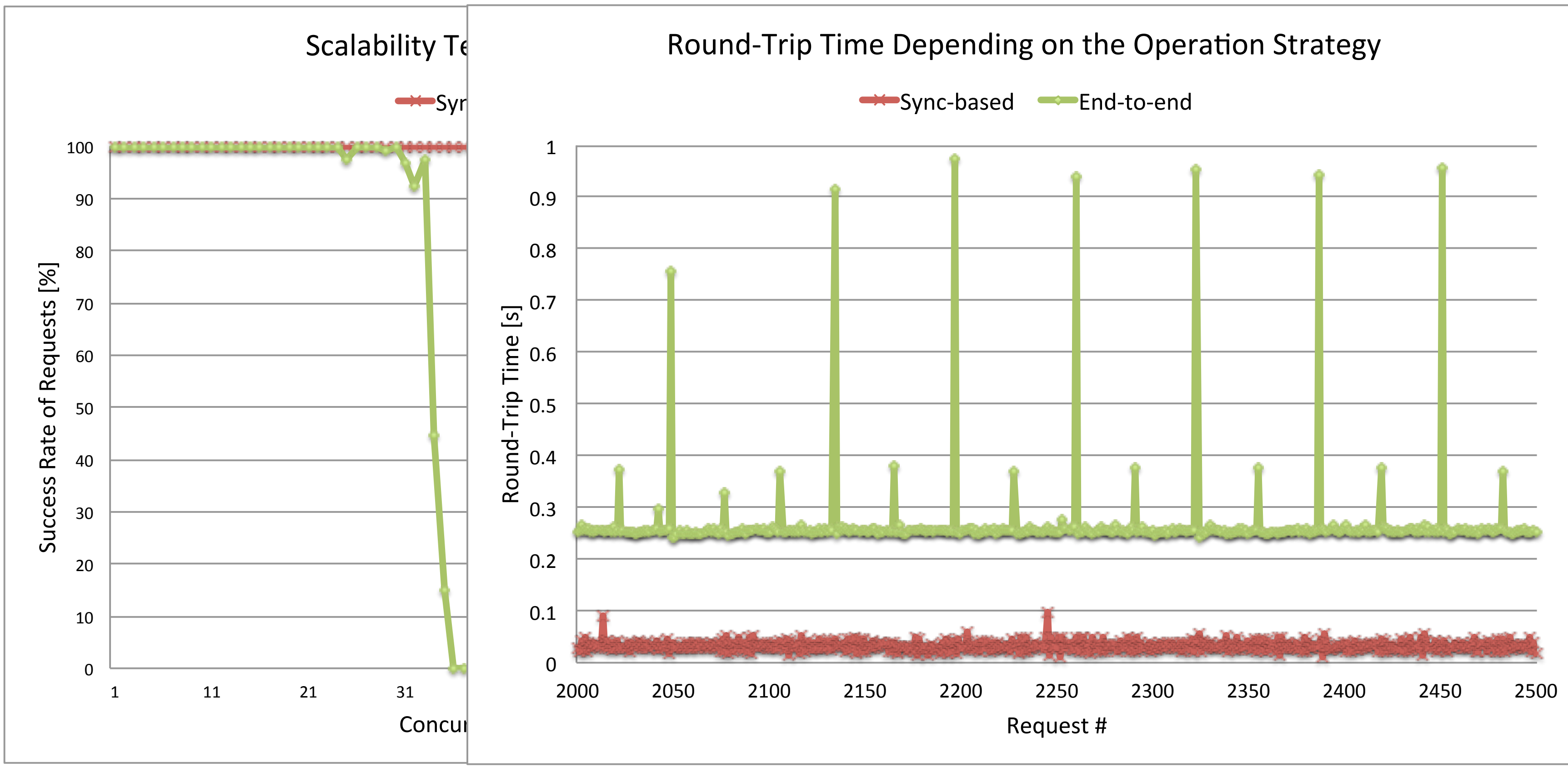
From devices to building automation



Performance - Appliance recognition

Virtual Sensor: Raspberry Pi

Virtual Classes: 5x OpenPicus Flyport Wi-Fi PRO



Contributions

- **Agent-based HMM and GMM virtual classes in C**
  - Deploy complex models (multidimensional matrices) on constrained devices
- **MaLeX - Machine Learning Exchange Format**
  - Schema describing generative algorithms
  - Embeddable into matlab to automatically export the models in the right format
- **BAME - Building Automation Mashup Editor**
  - Easily integrate new devices by relying on semantic descriptions

Conclusion and future works

- **Conclusion**
  - Sensing devices are powerful enough for running HMM
  - Reusable architecture based on RESTful APIs and semantics
  - Building automation machine learning mashups feasible
  - No dedicated infrastructure is required for data-driven analysis

- **Future works**
  - Finding efficient training algorithms segmenting the training data
  - Representing training algorithms as Web agents with semantic descriptions
  - Distributing training agents within the sensor network

Questions