

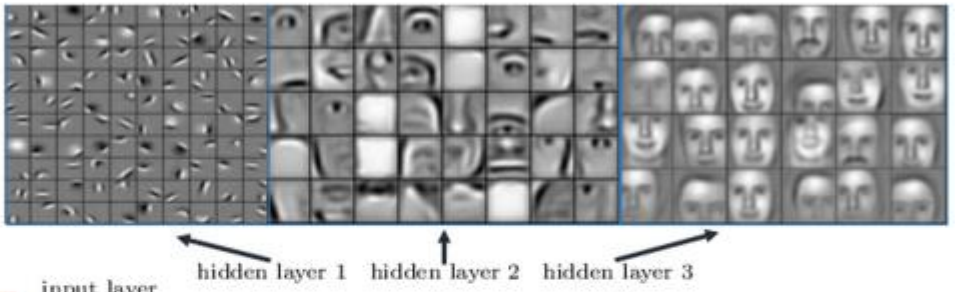

CS-541
Wireless Sensor Networks

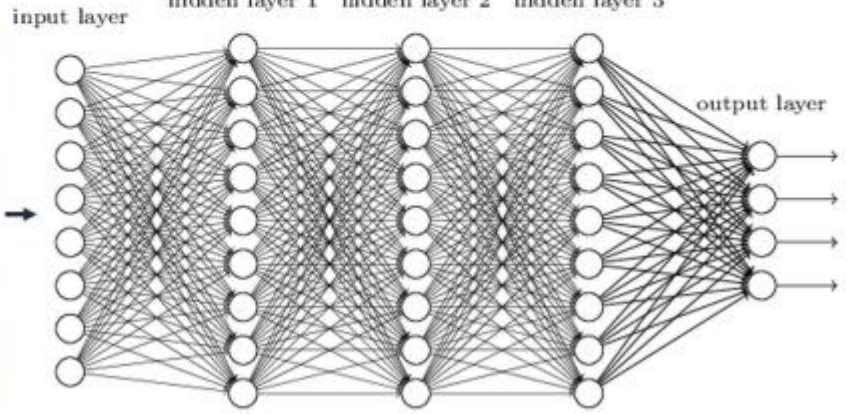
Lecture 11: Machine Learning in WSN

Spring Semester 2017-2018

Prof Panagiotis Tsakalides, Dr Athanasia Panousopoulou, Dr Gregory Tsagkatakis

Machine Learning





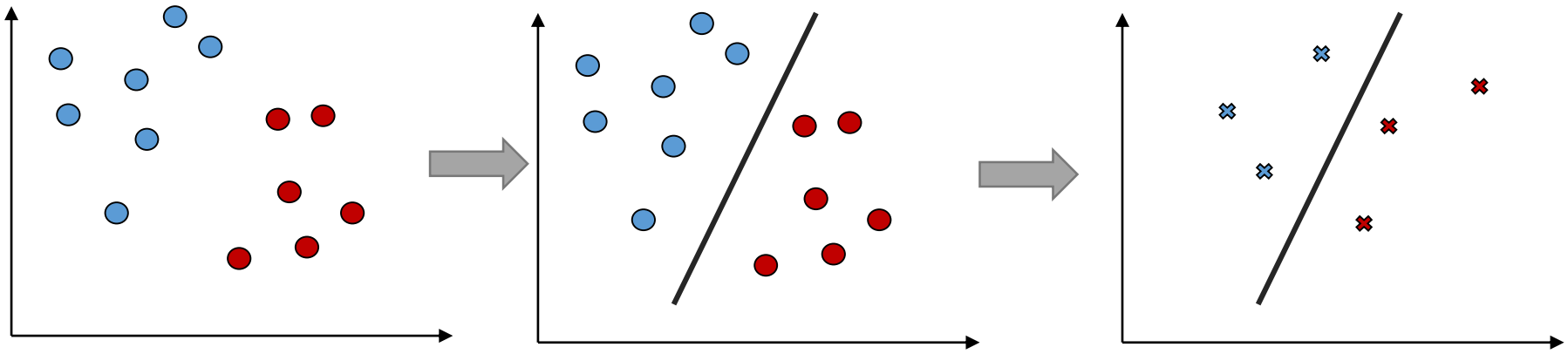
Machine Learning

Machine learning: construction and study of algorithms that can learn from data

- Models of example inputs (training data) → make predictions or decisions on new inputs (testing data)
- Data: characteristics
- Prior assumptions: a priori knowledge
- Representation: How do we represent the data
- Model / Hypothesis space: Hypotheses to explain the data
- Feedback / learning signal: Learning signal (delayed, labels)
- Learning algorithm: Model update
- Evaluation: Check quality

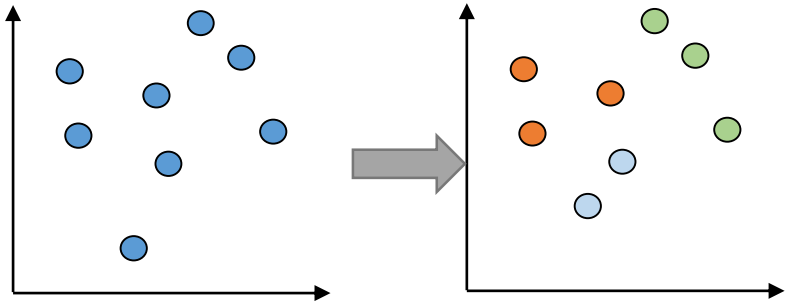
Types of ML

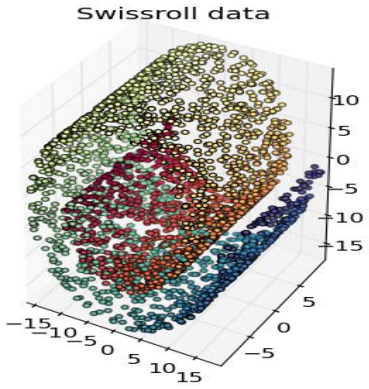
Supervised learning: present example inputs and their desired outputs (**labels**) → learn a general rule that maps inputs to outputs.

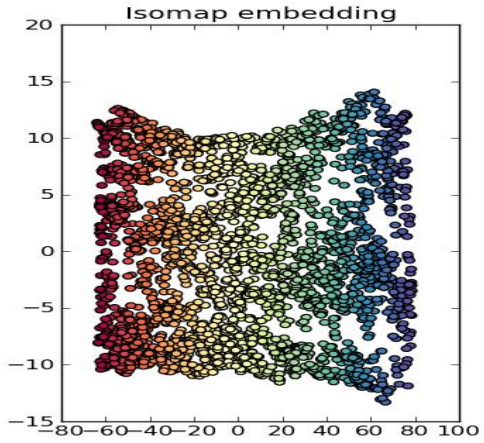


Types of ML

Unsupervised learning: no labels are given → find structure in input.

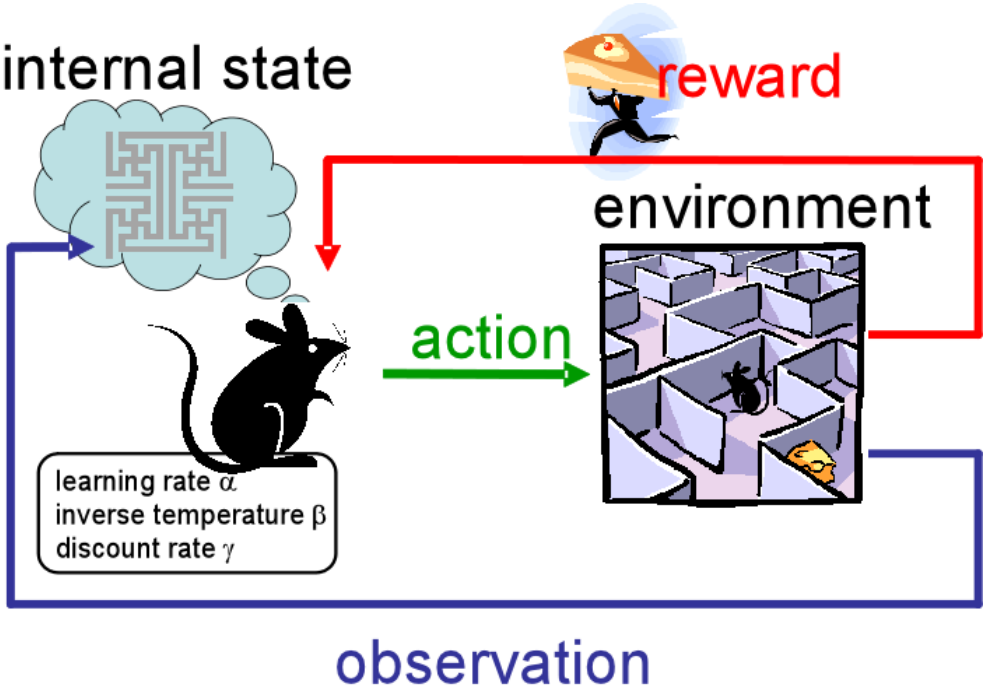






Types of ML

Reinforcement learning: system interacts with environment and must perform a certain goal without explicitly telling it whether it has come close to its goal or not.



Applications in WSNs

Network performance optimization

- Routing
- Distributed regression framework
- Data Aggregation
- Localization and Objects Targeting
- Medium Access Control

Data Mining

- Activity recognition
- Event Detection and Query Processing

Unsupervised learning - Clustering

What is a cluster?

groups of data instances that are similar to each other in one cluster and data instances that are very different from each other into different clusters

Hard vs. Soft

- *Hard*: belong to single cluster
- *Soft*: belong to multiple clusters

Flat vs. Hierarchical

- *Flat*: clusters are flat
- *Hierarchical*: clusters form a tree

K-means clustering

- K-means is a partitional clustering algorithm
- Let the set of data points (or instances) D be

  $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$,

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and r is the number of attributes (dimensions) in the data.
- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

K-means algorithm

Given *k*, the *k-means* algorithm works as follows:

1)Randomly choose *k* data points (seeds) to be the initial centroids, cluster centers

2)Assign each data point to the closest centroid

3)Re-compute the centroids using the current cluster memberships.

4)If a convergence criterion is not met, go to 2).

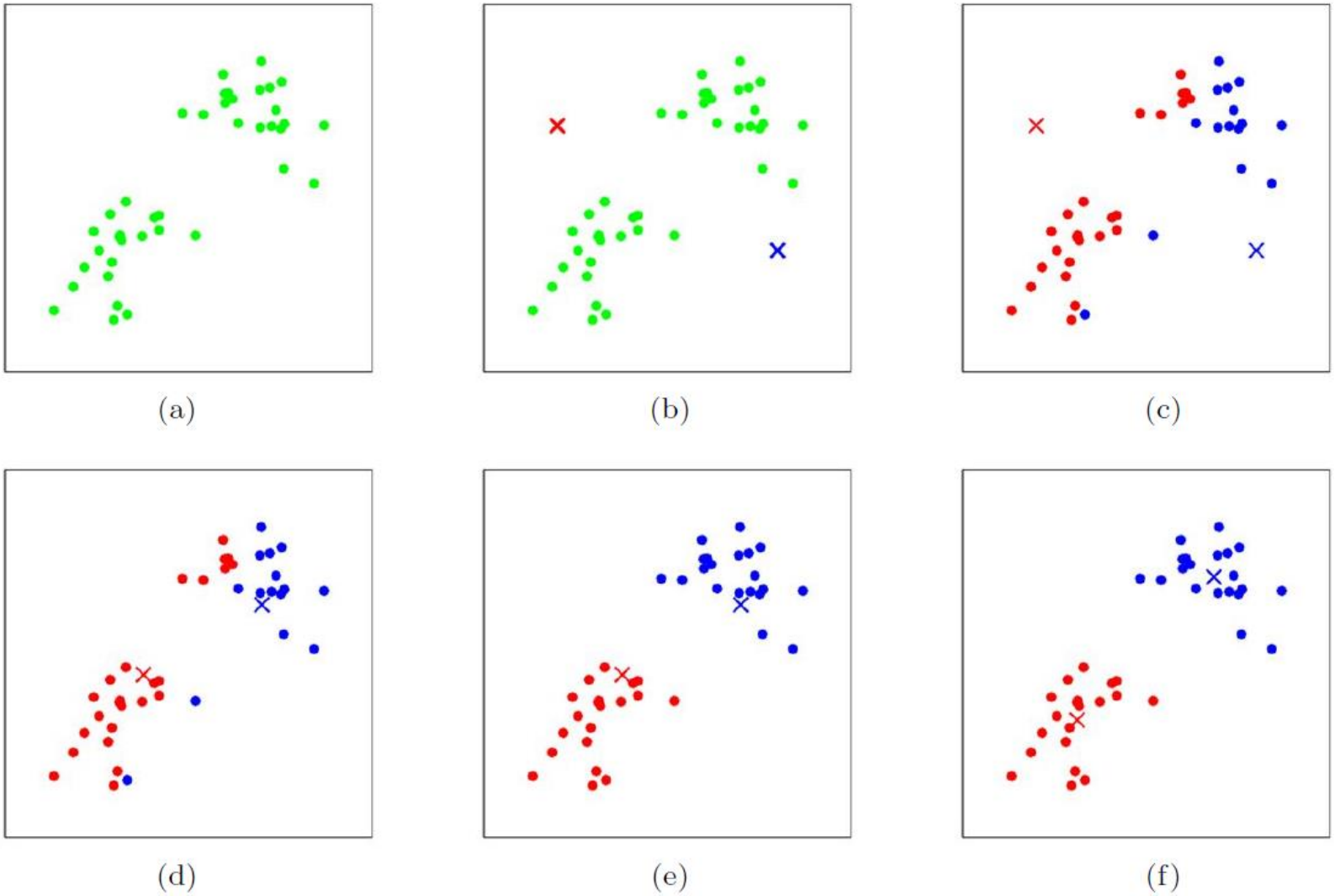
Stopping criteria

- no re-assignments of data points to different clusters
- no change of centroids
- minimum decrease in the $SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2$

K-means example

Complexity is O(n * K * I * d)

n = number of points, K = number of clusters,
I = number of iterations, d = dimensionality

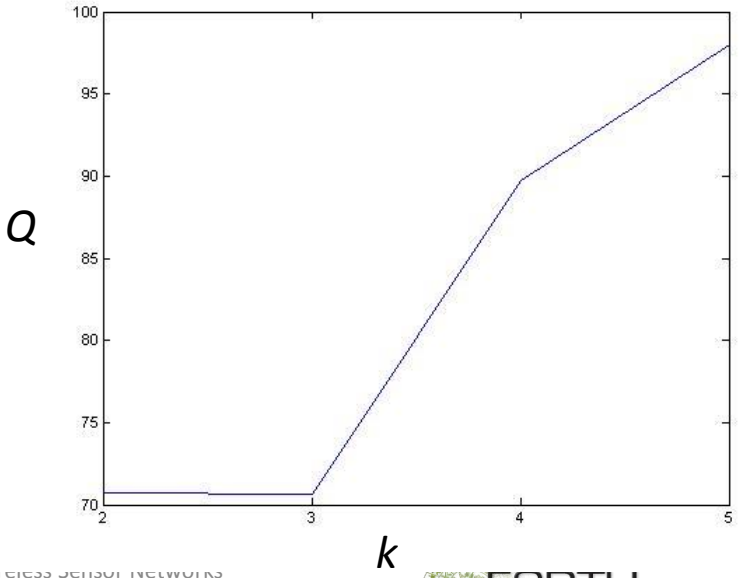


Issues with K-means

- Random initialization -> different clusters each time
- Data points are assigned to only one cluster
- Implicit assumptions about the “shapes” of clusters
- You have to pick the number of clusters…

Cluster tightness

$$Q = \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{\boldsymbol{x} \in C_i} d(\boldsymbol{x}, \mu_i)$$

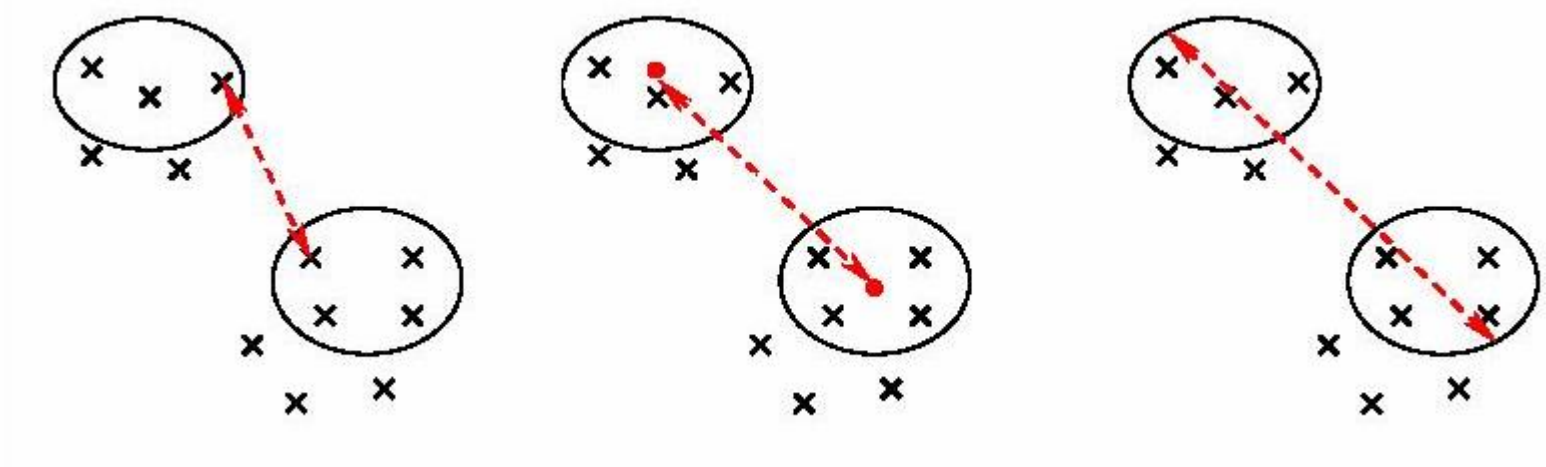


Distance Between Two Clusters

single-link clustering: distance between clusters -> shortest distance between any two members.

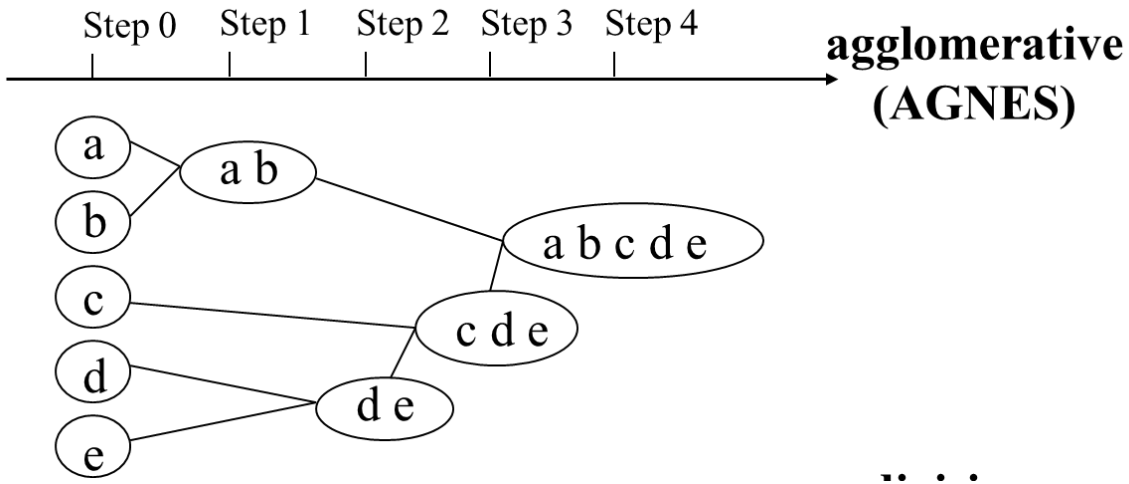
complete-link clustering: distance between clusters -> longest distance between any two members.

average-link clustering: distance between clusters -> average distance between any two members

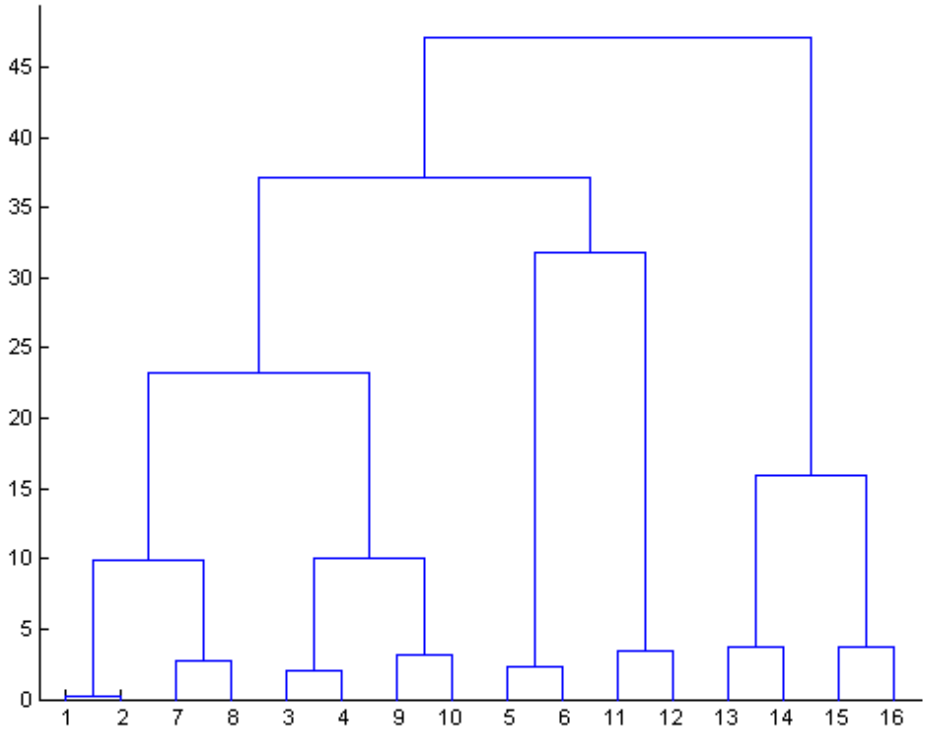


Hierarchical Agglomerative Clustering

- We start with every data point in a separate cluster
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left
- This is called a bottom-up or agglomerative method



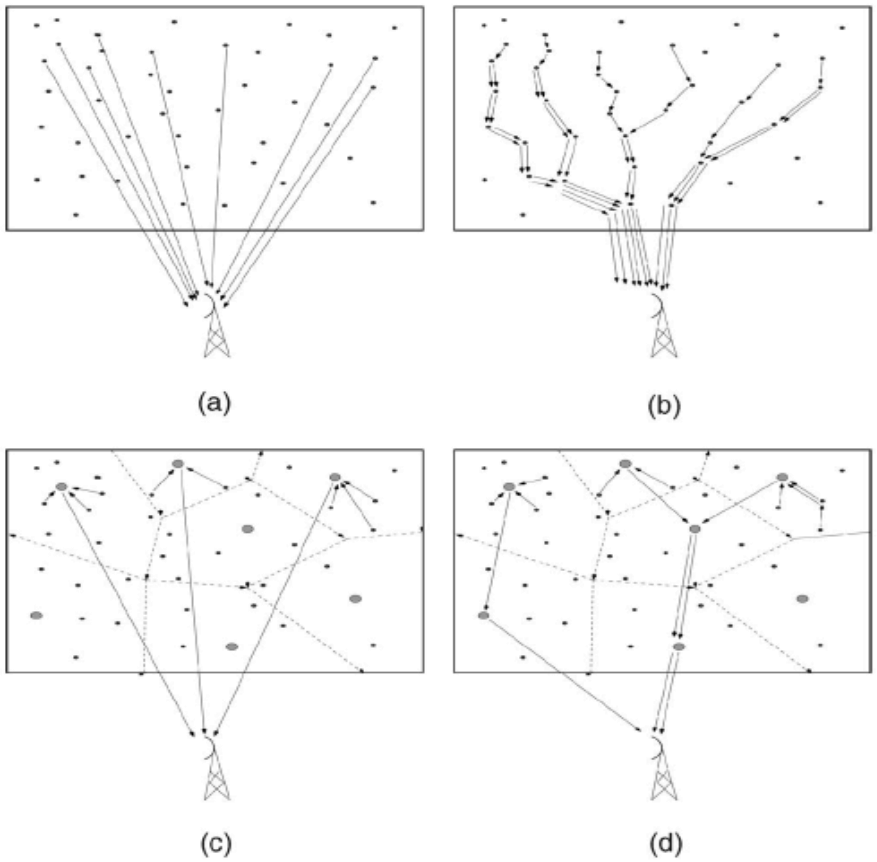
Hierarchical Clustering (cont.)



- This produces a binary tree or ***dendrogram***
- The final cluster is the root and each data item is a leaf
- The height of the bars indicate how close the items are

Clustering in WSN

- Scalability:
  - Reduce routing tables to within cluster
- Data Aggregation
  - Energy reduction vs. full data transmission
  - CH based data fusion
  - multi-hop tree structure aggregation
- Load Balancing
  - Eliminate redundant data transmissions
  - Communications between CHs
- Energy reduction
  - Selective sampling within cluster
  - Short-range communications with CH
- Robustness & Fault tolerance
  - Support node failure/recovery
  - mobility of sensors
  - noisy measurements etc.
- Efficiency
  - Collision avoidance (intra vs. inter cluster communications)
  - Latency reduction by reducing hops
  - Network life-time maximization
  - Quality-of-service

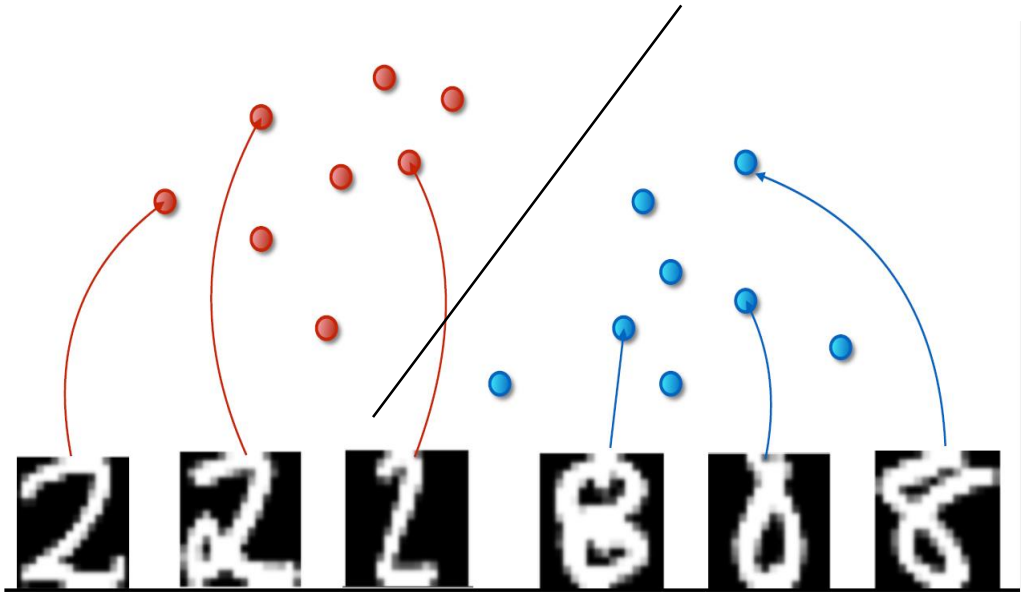


Supervised Learning

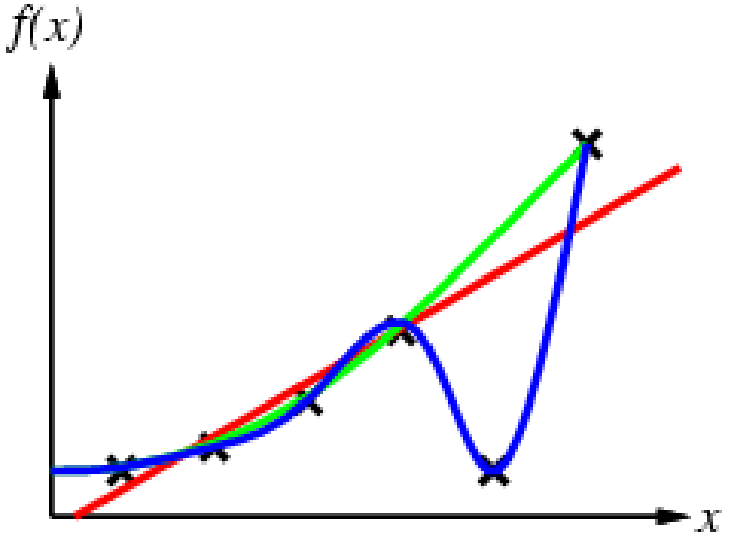
Input **x**∈ $\mathcal{X}$ → y = f(**x**) → Output y∈ $\mathcal{Y}$

Learn f(**x**) from training examples

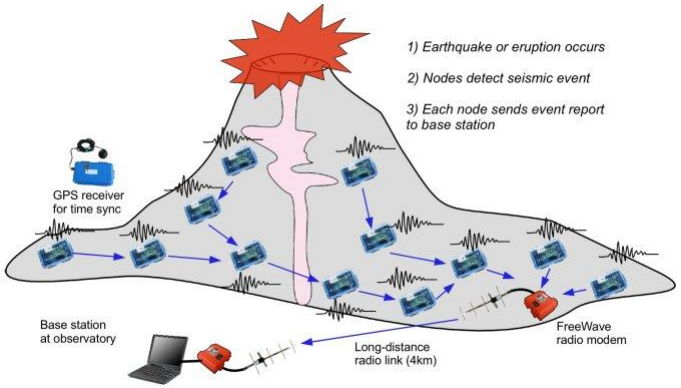
Decision rule

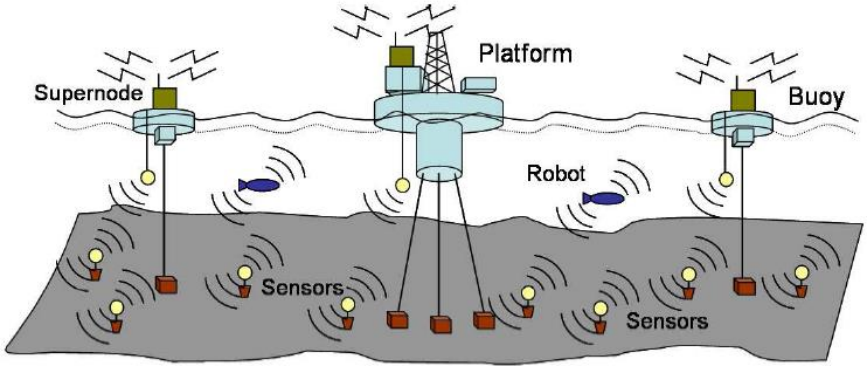


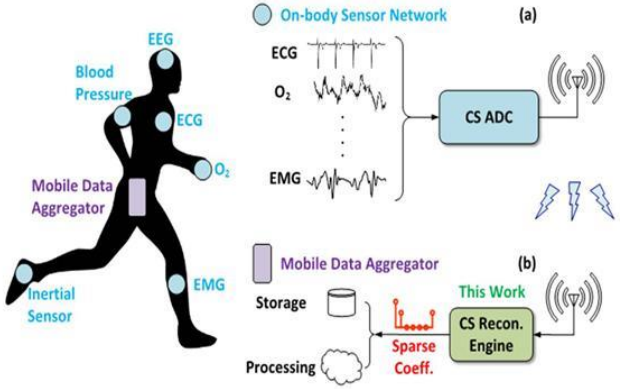
Generalization: Ockham's razor

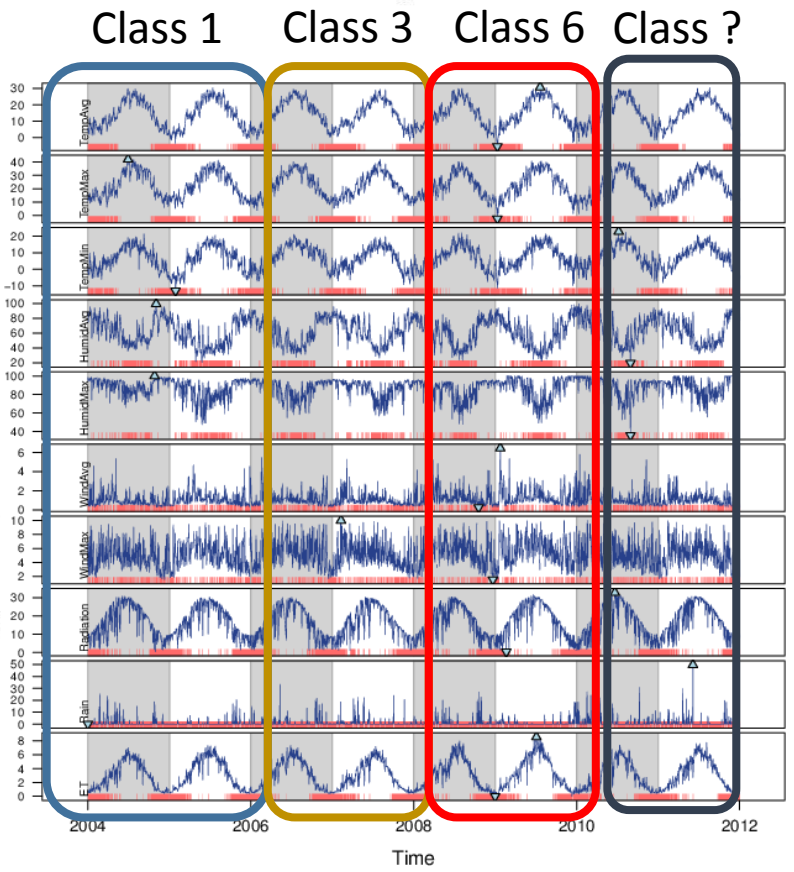


Applications









Support Vector Machines (SVMs)

SVMs are linear classifiers that find a hyperplane to separate two class of data, positive and negative

Training examples *D* be $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$,

- **input vector:** $\mathbf{x}_i = (x_1, x_2, \ldots, x_n)$ $X \in R^n$
- **class label:** $y_i \in \{1, -1\}$.

SVM finds a linear function of the form (**w**: weight vector)

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$

$$y_i = \begin{cases} 1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

V. Vapnik in 1970s in Russia (became known in 1992).

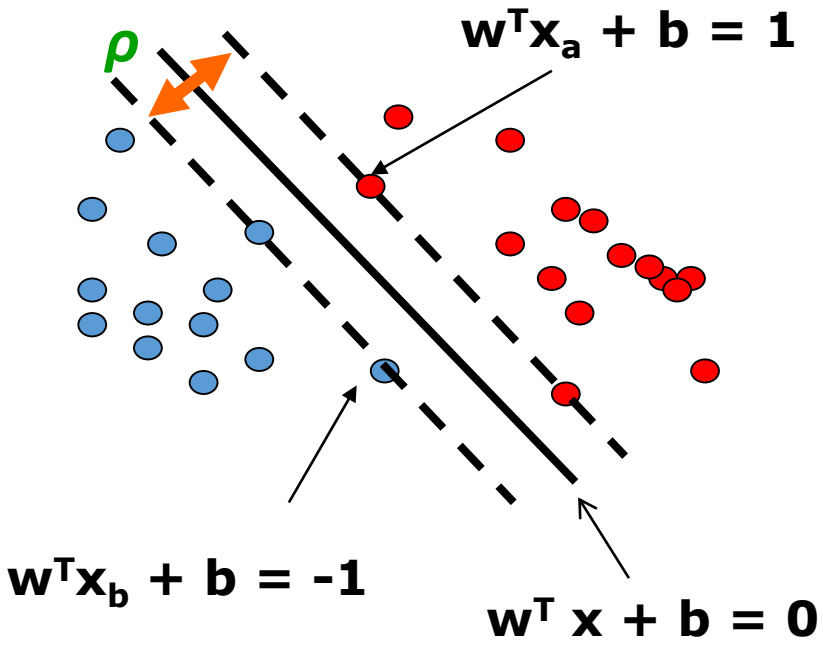
Support Vector Machines

- SVMs maximize the *margin* around the separating hyperplane.
- The decision function is fully specified by a subset of training samples, *the support vectors*
- *Max Margin classification*

$$\mathbf{w}^T\mathbf{x}_i + b \geq 1 \quad \text{if } y_i = 1$$

$$\mathbf{w}^T\mathbf{x}_i + b \leq -1 \quad \text{if } y_i = -1$$

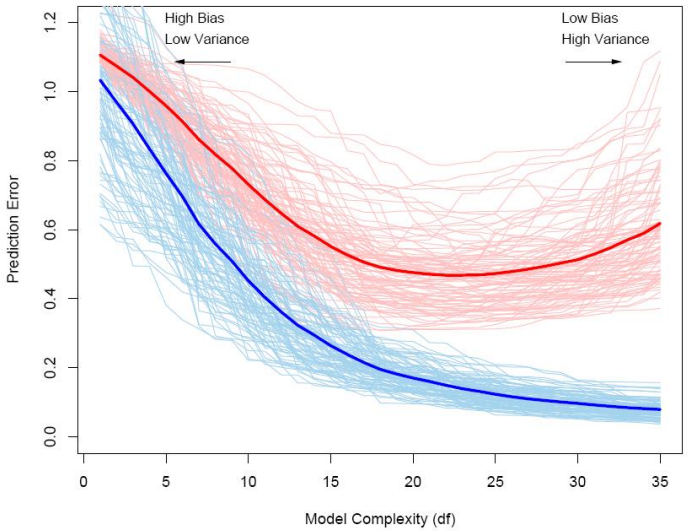
$$\textit{margin} \quad r = \frac{2}{\|\mathbf{w}\|}$$



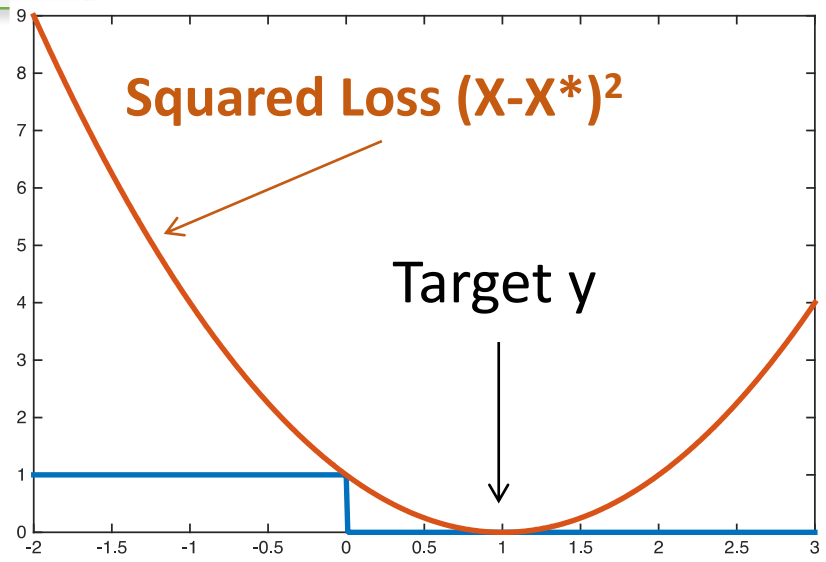
Support Vector Machines

Specification of loss function

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{w}^T\mathbf{x}_i)$$

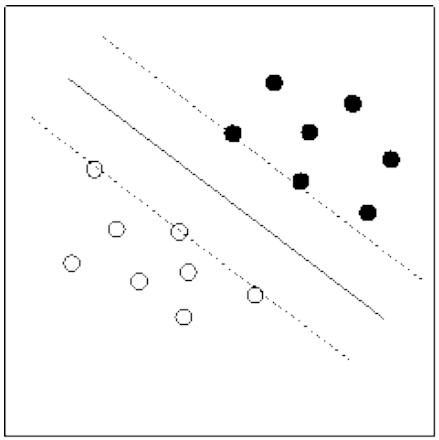


- support vector machines
- logistic regression
- lasso regression
- ridge regression

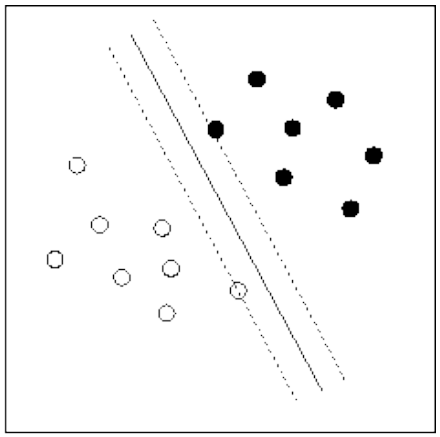


SVM optimization

Quadratic optimization problem:



(a) Larger margin



(b) Smaller margin

Find $\mathbf{w}$ and b such that

$$r = \frac{2}{\|\mathbf{w}\|} \text{ is maximized; and for all } \{(\mathbf{x_i}, y_i)\}$$

$\mathbf{w^T x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w^T x_i} + b \leq -1$ if $y_i = -1$

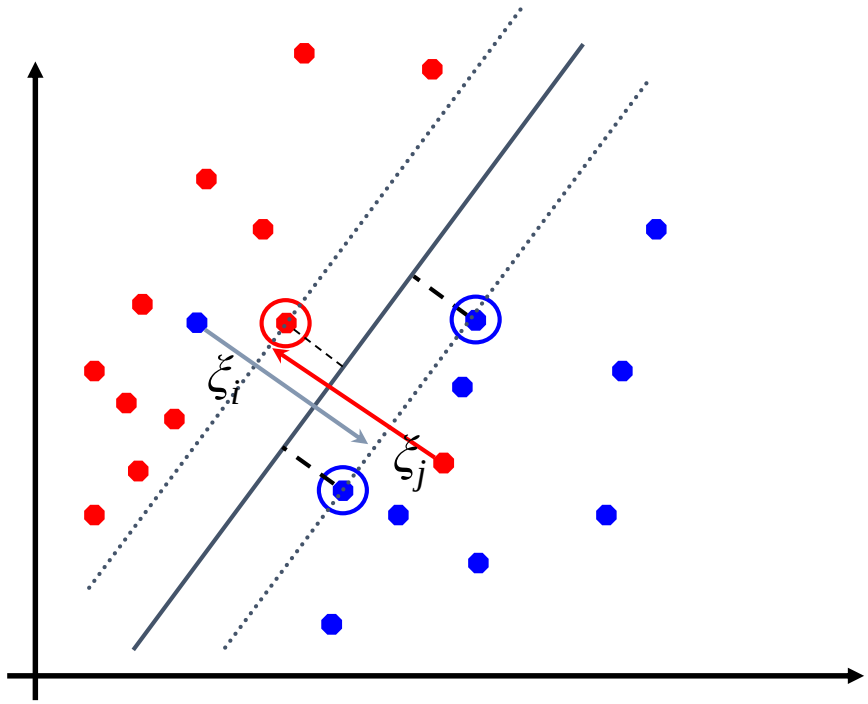
Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ is minimized;

and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i\,(\mathbf{w^T x_i} + b) \geq 1$

Soft Margin Classification

- If the training data is not linearly separable, *slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- Allow some errors
  - Let some points be moved to where they belong, at a cost
- Still, try to minimize training set errors, and to place hyperplane "far" from each class (large margin)



Soft Margin Classification Mathematically

- The old formulation:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- The new formulation incorporating slack variables:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$

$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- Parameter C can be viewed as a way to control overfitting
  - A regularization term

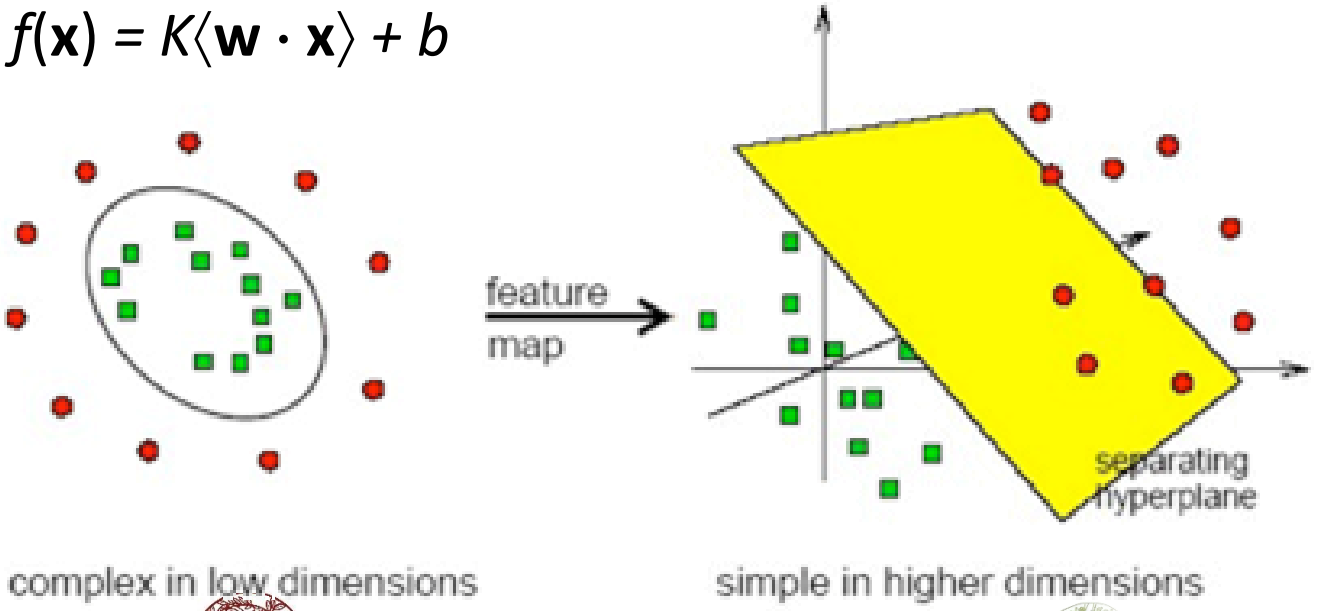
Non-linear SVM

Linear separable case is the ideal situation.

What about Non Linear?

- map the data in the input space X to a feature space F via a nonlinear mapping $\varphi \rightarrow$ **kernel trick**

SVM: $f(\mathbf{x}) = K\langle \mathbf{w} \cdot \mathbf{x} \rangle + b$



,

The "Kernel Trick"

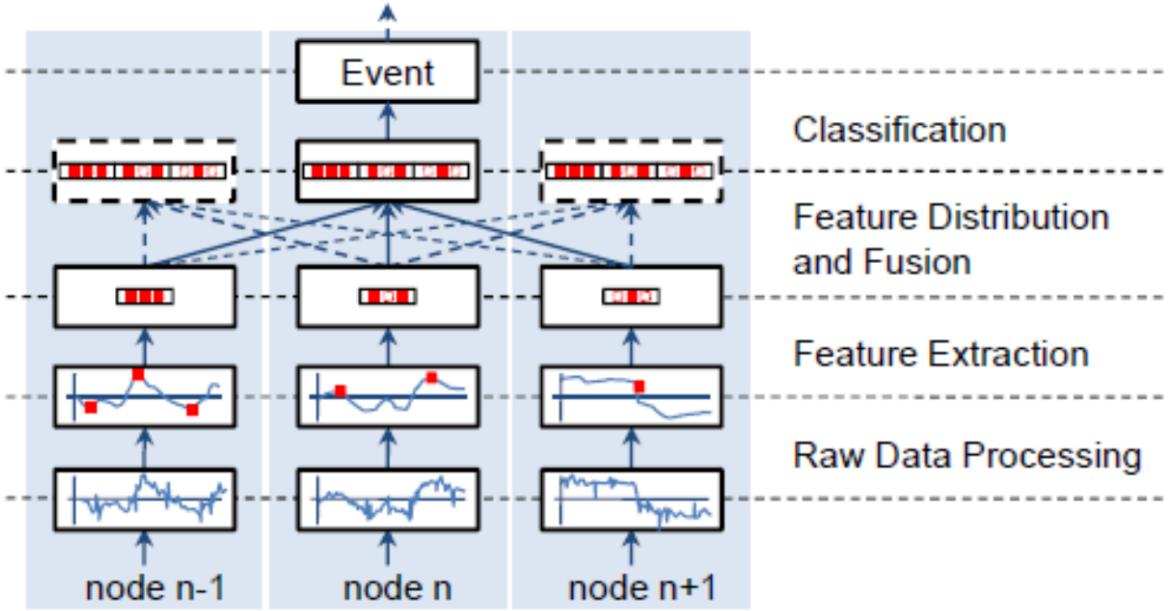
- The inner product of two observations is given by $\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}.$
- The linear support vector classifier can be written as $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$
- The solution function can take the form $f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$

d^{th} degree polynomial	$K(x, x') = (1 + < x, x' >)^d$
Radial kernel	$K(x, x') = \exp(- \| x - x' \|^2 / c)$
Neural network	$K(x, x') = \tanh(k_1 < x, x' > + k_2)$

Supervised Event Detection

Objective

Event Detection -> the network has to identify which application-specific incident has occurred based on the raw data gathered by individual sensor nodes.



Supervised Event Detection

Feature extraction

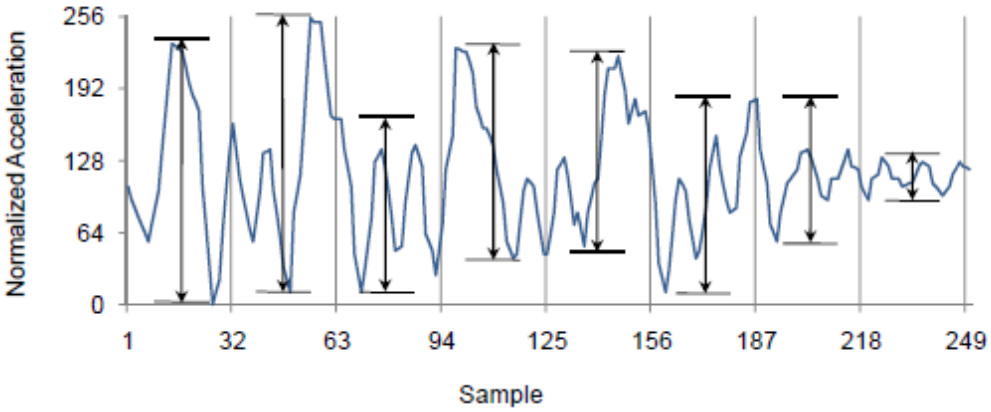
- Min, max, avg, wavelet…
- Discretization

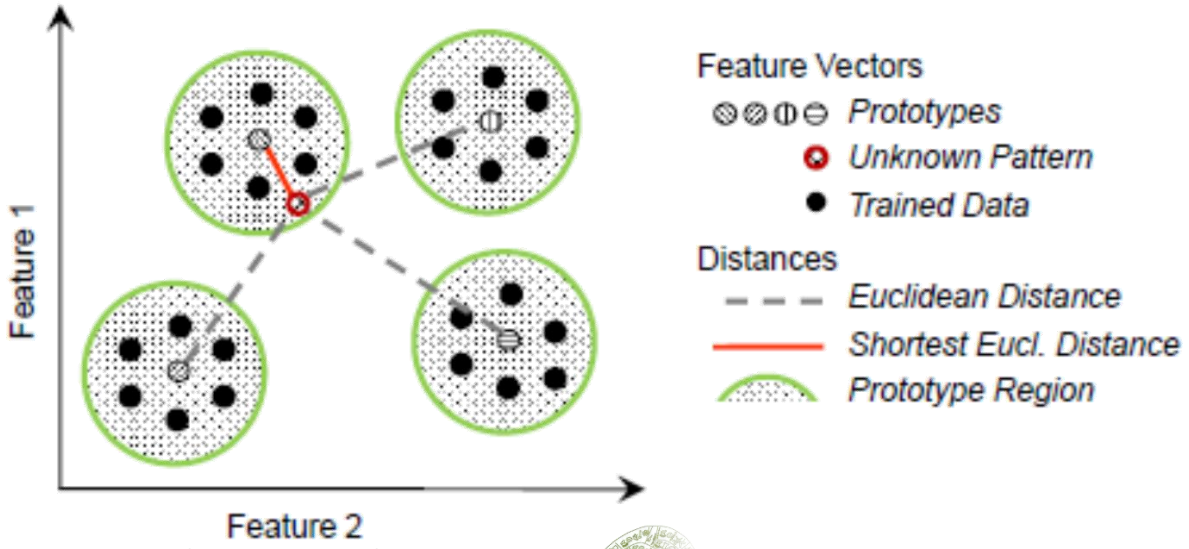
Classification

- DT, SVM, kNN

Training vs. Testing

- Data
- Resources
- Overheads





Community Seismic Network

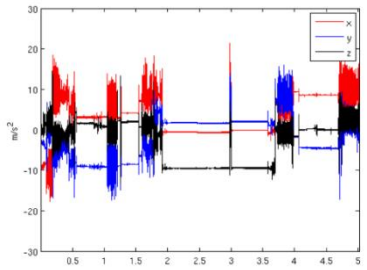
Objective

Detect seismic motion using accelerometers in smartphones and other consumer devices
And issue real-time alerts

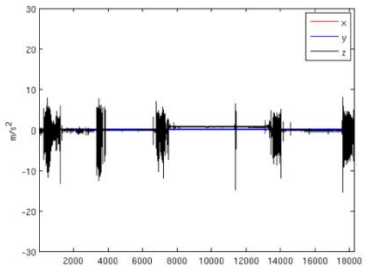
- Limit false alarm

Hypothesis testing

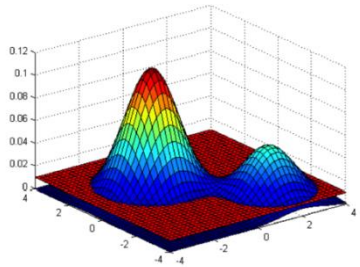
$$\frac{\mathbb{P}\left[X_{s,t} \mid E_t = 1\right]}{\mathbb{P}\left[X_{s,t} \mid E_t = 0\right]} \geq \tau.$$



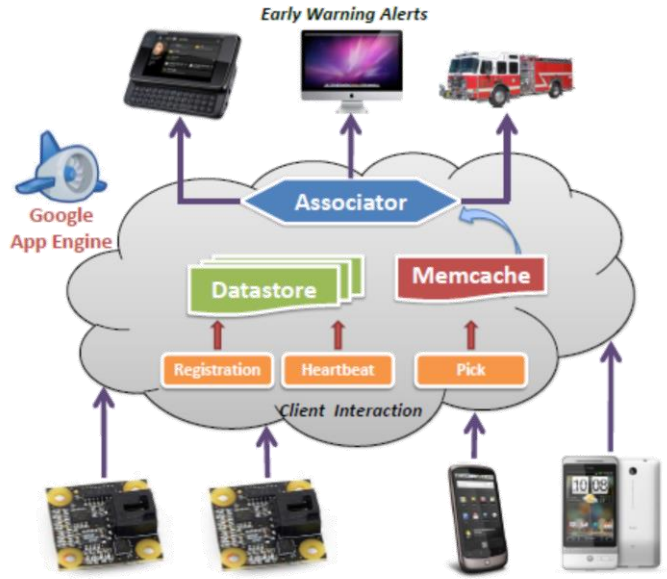
(a) *Before gravity correction*

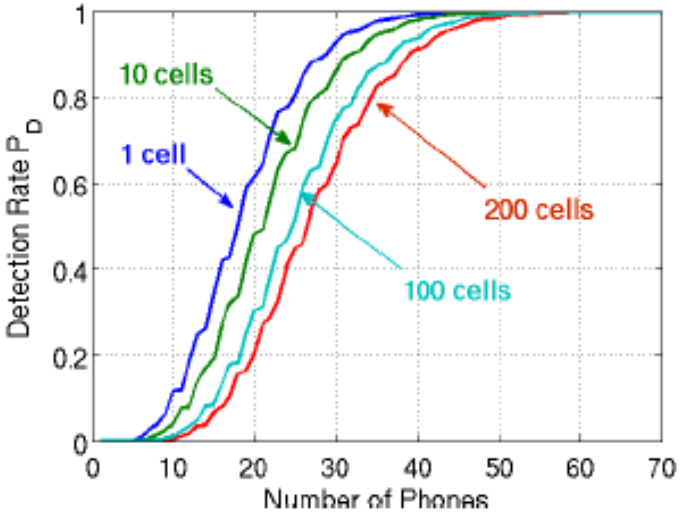


(b) *After gravity correction*



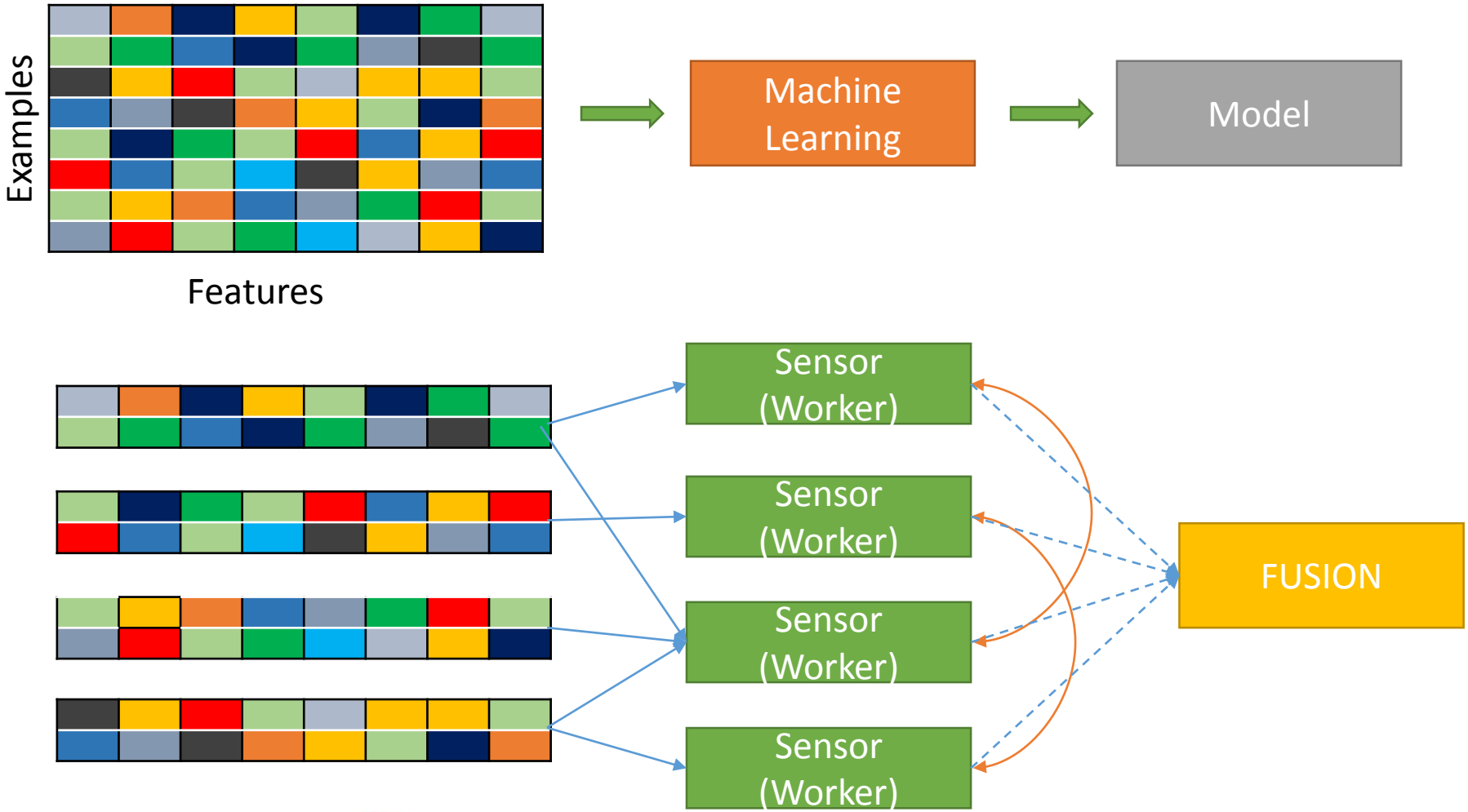
(c) *GMM based picking*





(c) *Detection rates*

Distributed Learning in WSN



Gossip based SVM training

Problem Formulation

→ Graph of *n* nodes

→ Each edge is a communication link

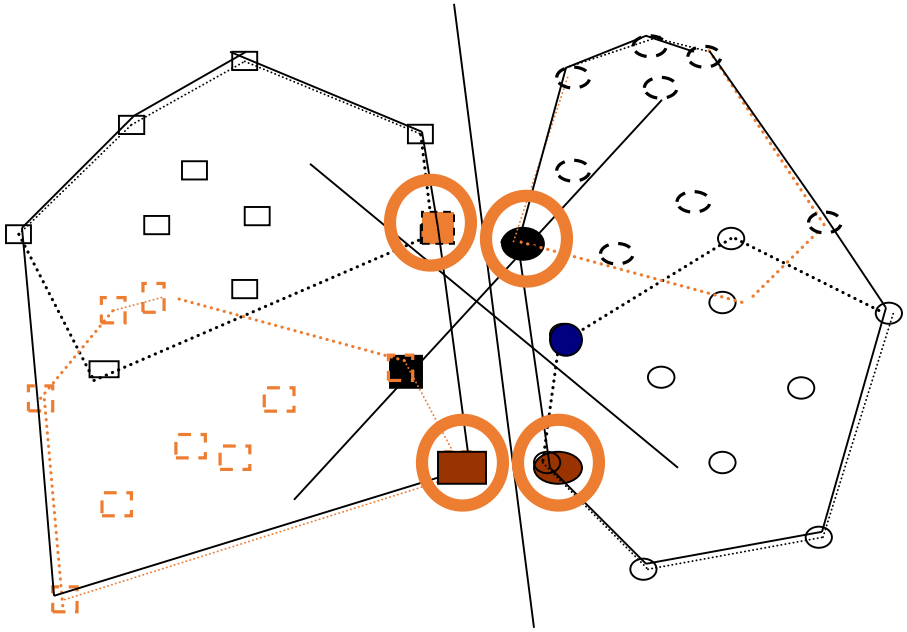
→ N_i neighboring sensors of *i*

MSG-SVM

- Train locally SVM at each sensor $SV_i(0) = \left\{ x_k(0), y_l(0) : \sum_k \theta_k x_k(0) - \sum_l \gamma_l y_l(0) = w_i(0) \right\}$
- $SV_i(0)$ are exchanged to nodes N_i
- Data at time step *t,* at node *i:* $SV_i(t)$ and $SV_{N(i)}(t)$
- Train SVM locally and construct $w_i(t+1))$

> Energy Efficiency: transmit to neighbors only the support vectors that were not transmitted in previous steps.

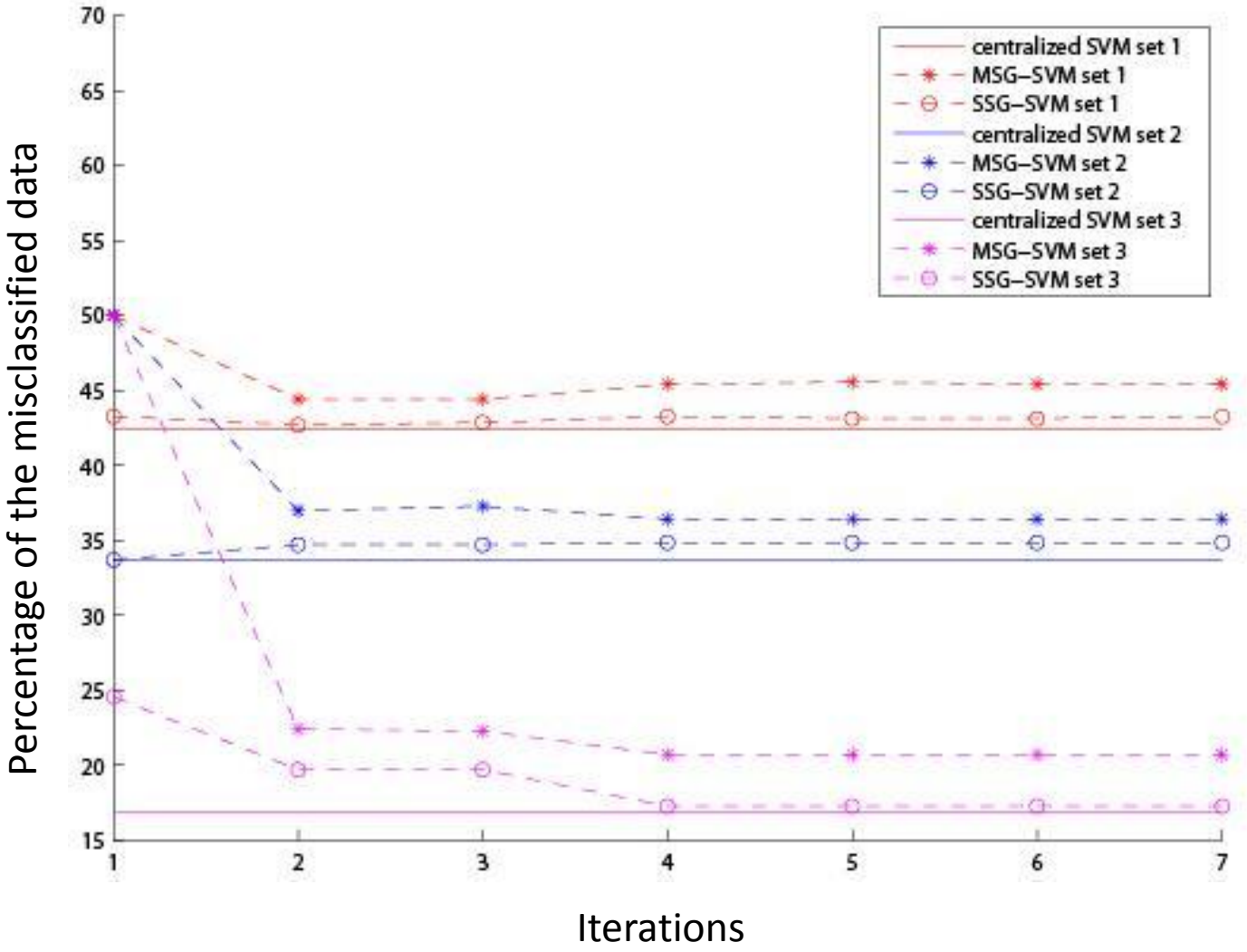
Gossip-based Distributed SVM



$x_k^{(1)}$, $y_l^{(1)}$ } S_1

$x_k^{(2)}$, $y_l^{(2)}$ } S_2

Gossip-based Distributed SVM



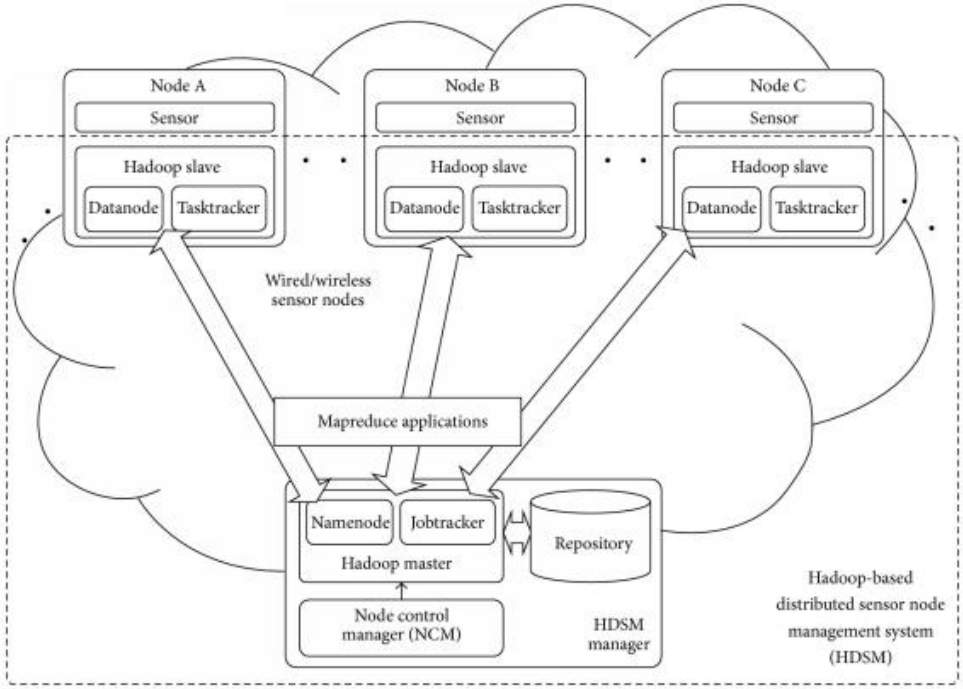
Large Scale Distributed & Parallel Processing

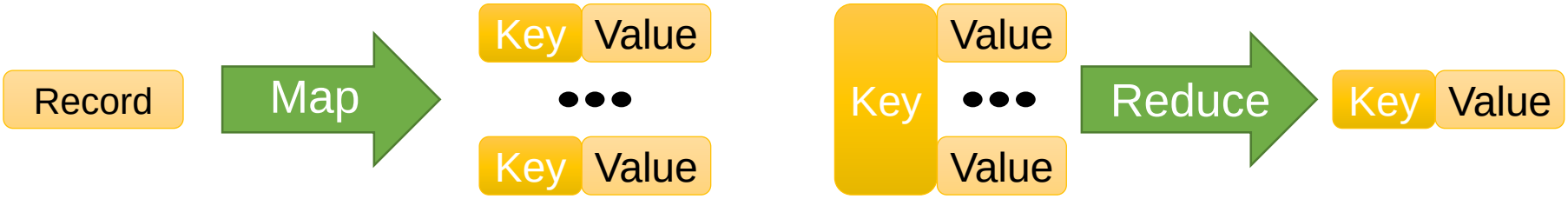
Objectives

- Highly decentralized
- Asynchronous
- Robust (fault & dynamics)

Architectures

- Google MapReduce,
- Apache Spark





Distributed Supervised Learning

Collaborative training for Distributed Learning

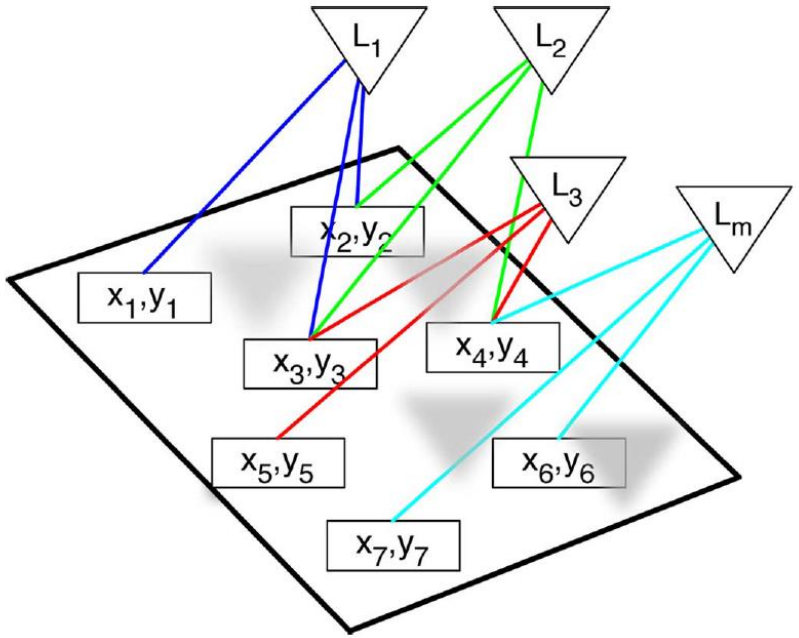
- Require only local communications

Message-passing algorithms

- Bipartite graphs
- Iterative updates

Architecture

- *m* agents (sensors)
- $S=\{(x_i,y_i)\}_{i=1\ldots n}$ data sources
- For WSN: n = neighbors



Collaborative training for Distributed Learning

AN ALGORITHM FOR TRAINING COLLABORATIVELY

Input: Ensemble $\{S_n^j\}_{j=1}^m$

Initialize: $\mathbf{z} = \mathbf{y}$

$g_{j,0} = 0,\ j = 1, \ldots, m$

Train: for $t = 1, \ldots, T$

for $j = 1, \ldots, m$

Compute:

$$g_n^{j,t} := \arg\min_{f \in \mathcal{H}_K} \left[\sum_{i \in \bar{S}_j} (f(x_i) - z_{i,})^2 + \lambda_j \| f - g_n^{j,t-1} \|_{\mathcal{H}_K}^2 \right]$$

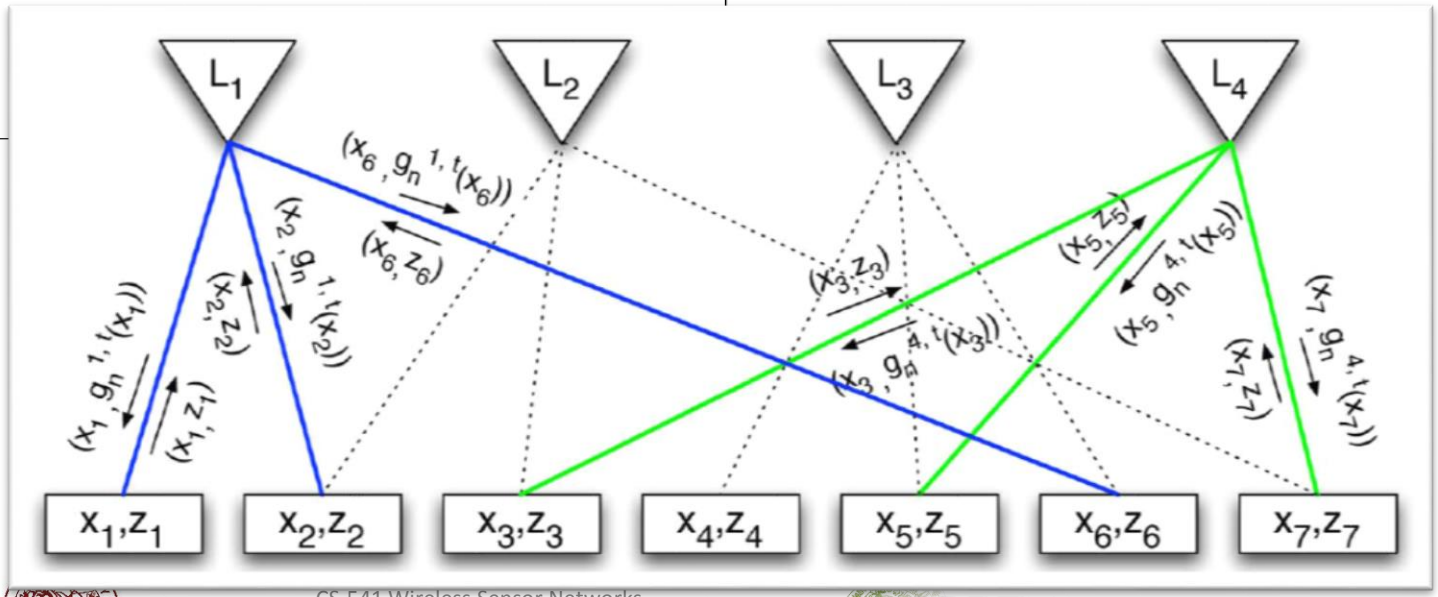
Update:

$$z_i \leftarrow g_n^{j,t}(x_i) \ \ \forall i \in \bar{S}_n^j$$

end

end

Output: $\{g_n^{j,T}\}_{j=1}^m$, $\mathbf{z}_T := \mathbf{z}$



Decision Trees & Forests

Rule-based prediction

Trees hierarchically encode rule-based prediction

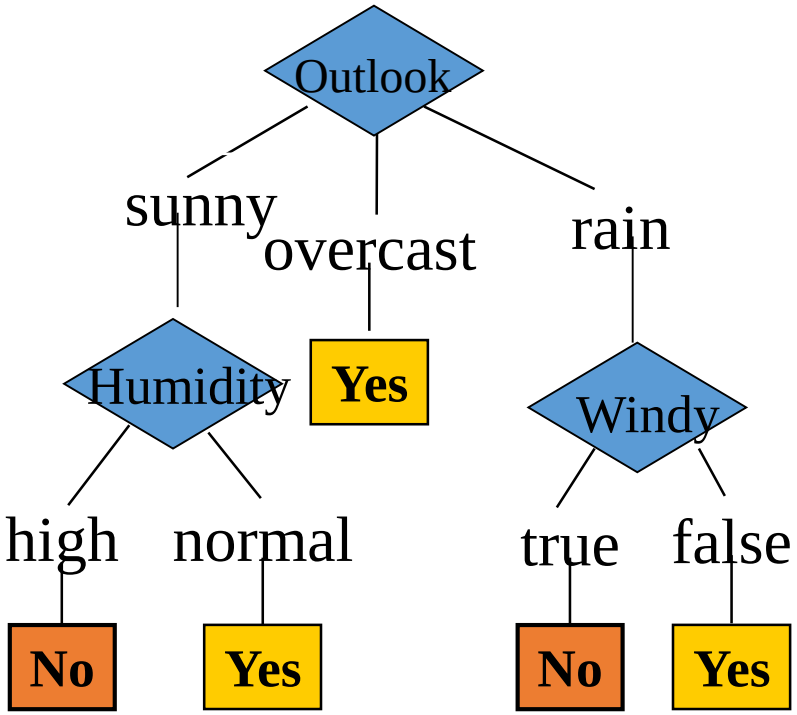
- Nodes test features to branch
- Leaves produce predictions

Decision tree -> set of rules

- Each path from the root to a leaf

Splitting rule

- Information gain
- Sensitivity to attribute selection

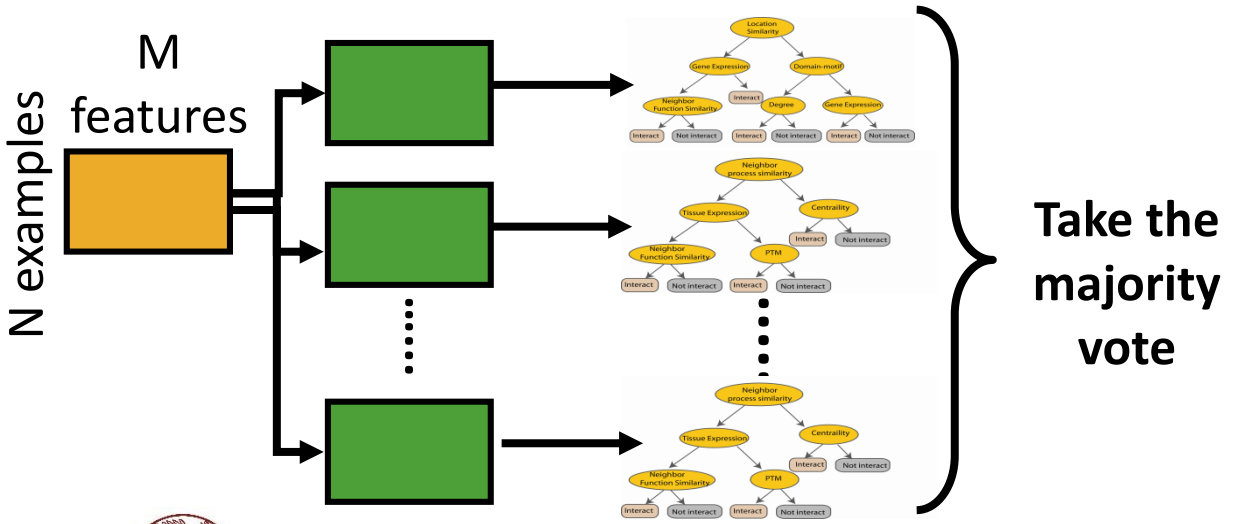


Decision Trees & Forests

Random Forests grows many classification trees.

Classify a new object from an input vector,

- run the input vector down each of the trees in the forest.
- Each tree gives a classification-> "vote" for that class.
- The forest chooses the classification having the most votes



Multi-label classification

- Is it eatable?
- Is it sweet?
- Is it a fruit?
- Is it a banana?

Is it a banana?

Is it an apple?

Is it an orange?

Is it a pineapple?

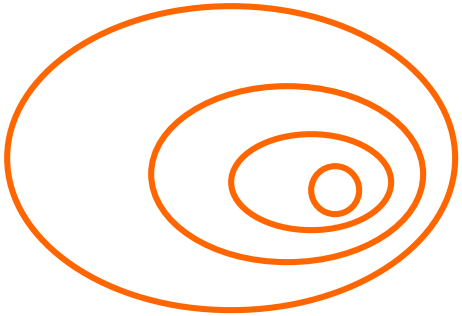
Is it a banana?

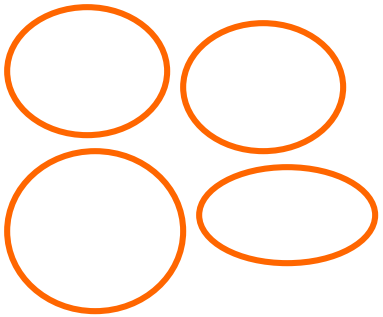
Is it yellow?

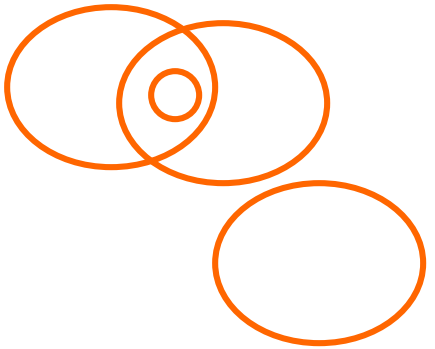
Is it sweet?

Is it round?

Different structures







Nested/ Hierarchical

Exclusive/ Multi-class

General/Structured

Online Machine Learning

Sequential/Stream data processing

For t=1,2,… inf

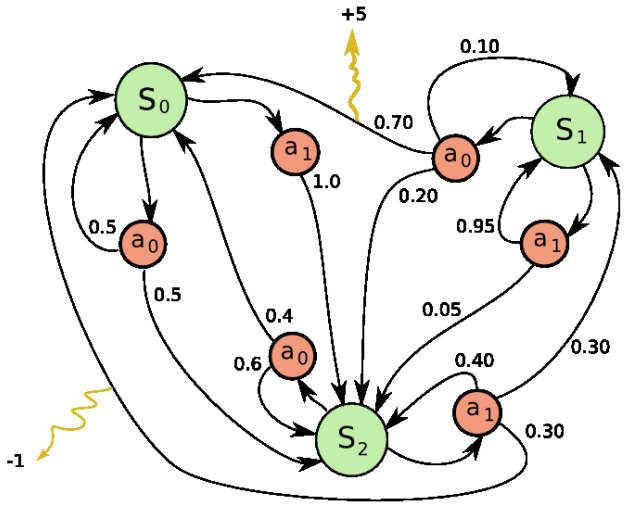
- Receive input X
- Apply predictor $y=H_t(x)$
- Receive true label y*
- Suffer loss $L(y-y^*)$
- Update predictor $H_{t+1} = F(H_t(x), L(y-y^*))$

Goal: perform almost as well as someone who observes the entire sequence and picks the best prediction strategy

Reinforcement Learning

Modeling as a Markov Decision Process (MDP)

- a set of environment states *S*
- a set of actions *A*
- rules of transitioning between states $\pi(A_t, A_{t+1})$
- rules for the reward of a transition $R(S_t, A_t)$
- rules that describe what the agent observes



Goal:

Reinforcement learning methods specify how the agent changes its policy as a result of experience.

Types of Reinforcement Learning

- Search-based: evolution directly on a policy
  - E.g. genetic algorithms
- Model-based: build a model of the environment
  - Then you can use dynamic programming
  - Memory-intensive learning method
- Model-free: learn a policy without any model
  - Temporal difference methods (TD)
  - Requires limited episodic memory (though more helps)
- Q-learning
  - The TD version of Value Iteration
  - This is the most widely used RL algorithm

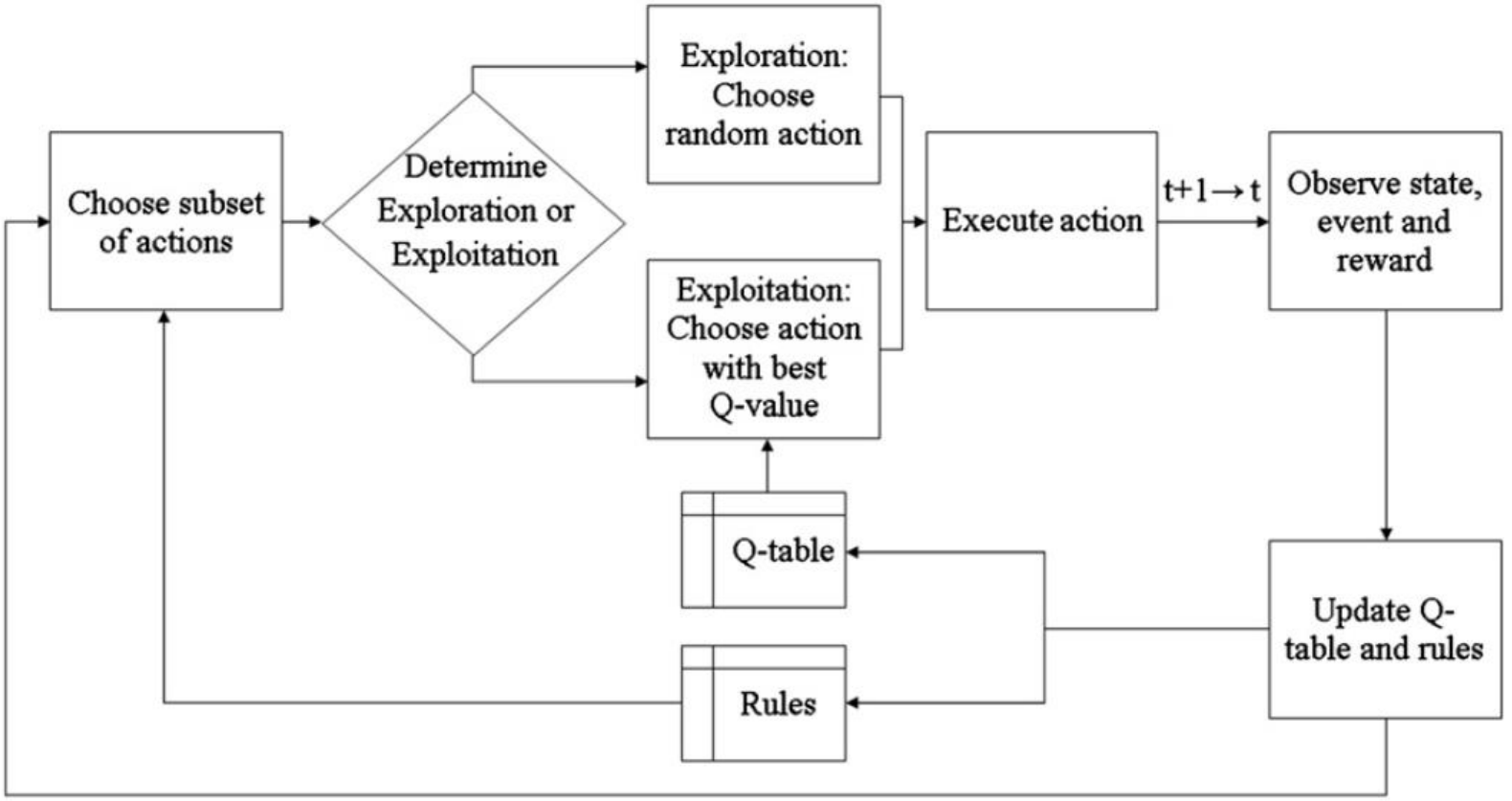
Q-Learning

Define **Q*(s,a):** "Total reward if an agent in state **s** takes action **a**, then acts optimally at all subsequent time steps"

Optimal policy: $\boldsymbol{\pi^*(s)=\text{argmax}_a Q^*(s,a)}$

- **Q(s,a)** is an estimate of **Q*(s,a)**
- Q-learning motion policy: $\boldsymbol{\pi(s)=\text{argmax}_a Q(s,a)}$
- Update Q recursively:

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a') \qquad 0 < \gamma < 1$$


Choose subset of actions
Determine Exploration or Exploitation
Exploration: Choose random action
Exploitation: Choose action with best Q-value
Execute action
t+1 → t
Observe state, event and reward
Q-table
Rules
Update Q-table and rules

Multi-Agent Learning

Centralized Learning (isolated learning)

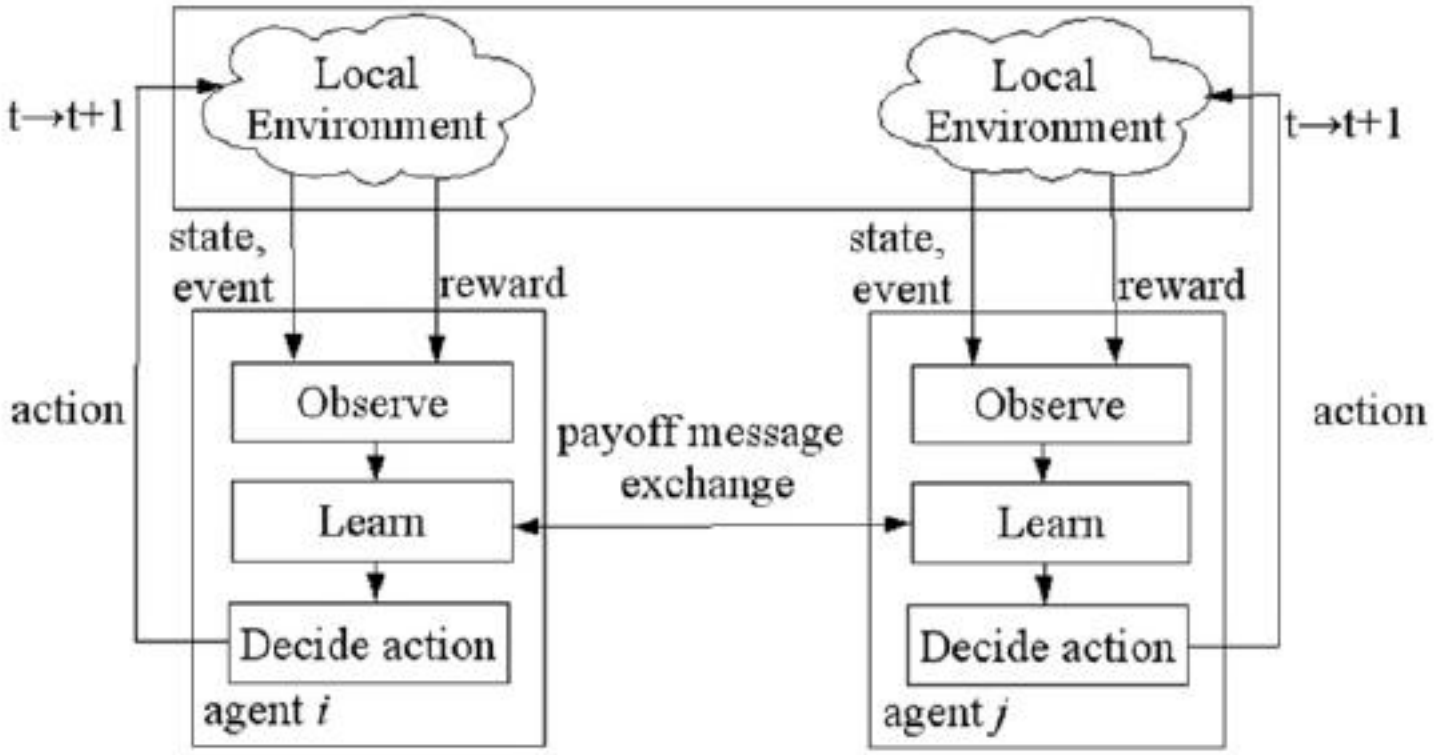
- Learning executed by a single agent (no interaction)
- Several learners -> different or identical goals at the same time

- Decentralized Learning (interactive learning)
  - Several agents are engaged in the same learning process
  - Groups of agents -> different or identical goals at the same time
- Single agent may be involved in several learning processes at the same time

Question:

- Communications overhead
- Convergence speed

Multi-Agent Learning

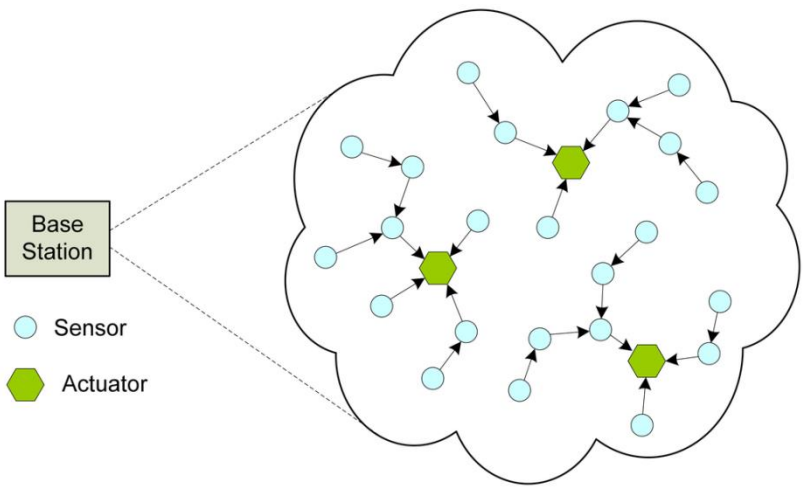


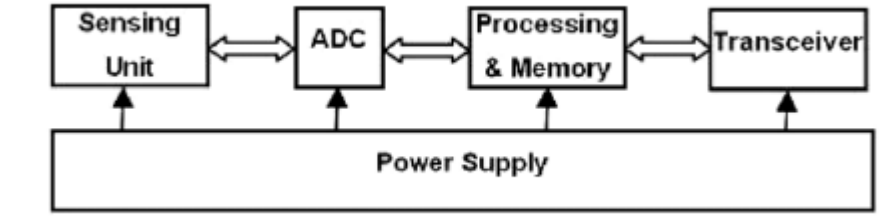
Scheme	Type of wireless network	Performance enhancement compared to traditional schemes
Cooperative communications	Ad hoc	Higher number of successful packet transmissions per transmitted power
	Wireless sensor	Higher throughput (or higher packet delivery rate or lower packet loss rate) Lower end-to-end delay
Rate adaptation	Single link	Higher throughput
Scheduling	Cellular	Higher throughput Lower end-to-end delay
	Wireless personal area	Lower decoding failure rate
Power management	Ad hoc Single link Wireless sensor	Without third-party involvement in power control Lower number of backlogged packets Higher throughput (or lower packet loss rate) per total consumed energy Lower energy consumption
Security management	Ad hoc	Higher throughput
Queue management	Ad hoc	Lower packet dropping rate
Medium access control	Cognitive radio Wireless sensor	Higher throughput (or lower packet loss rate) Lower energy consumption Higher throughput and lower energy consumption
Network coding	Cognitive radio Ad hoc	Higher throughput Faster code construction rate
Service discovery	Ad hoc	Higher average bandwidth savings

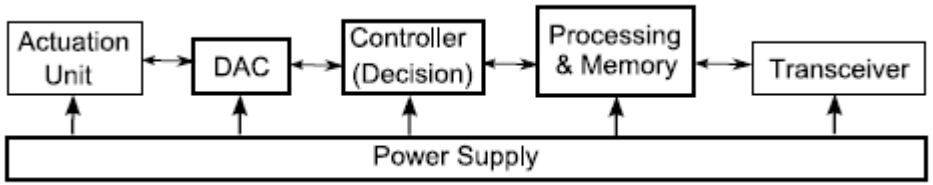
Wireless Sensor & Actuator Networks (WSAN)

WSANs: sense, interact, and change the physical world

- Sensors gather information about the state of physical world
- Sensors transmit the collected data to actuators (single/multi-hop)
- Actuators make the decision about how to react to this information
- Actuators perform actions to change the physical environment.
- The base station is monitoring and managing the overall network



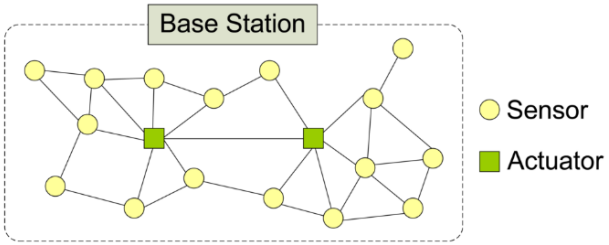




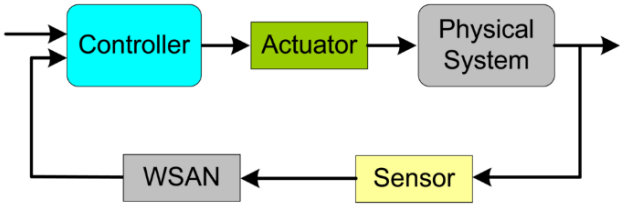
WSANs Characteristics

- *Throughput:* the effective number of data flow transported within a certain period of time
- *Delay:* queuing delay, switching delay, propagation delay, etc.
- *Jitter*: variations in delay. Difference in queuing delays experienced by consecutive packets.
- *Packet loss rate:* congestion, bit error, or bad connectivity.
- Resources: data processing capability, transmission rate, battery energy, and memory
- *Platform Heterogeneity*
- *Dynamic Network Topology:* stationary sensor & mobile actuators
- *Mixed Traffic:* periodic and aperiodic data

Models for WSANs

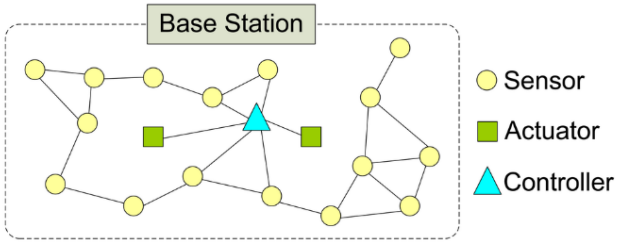


(a) Network topology

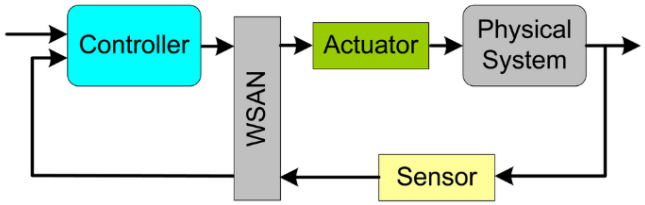


(b) Abstraction of control application

- no explicit controller
- data gathered by sensors will be transmitted directly to the corresponding actuators via single-hop or multi-hop communications



(a) Network topology



(b) Abstraction of control application

- one or more controller entities explicitly exist in the WSAN
- both the sensor data and control commands need to be transmitted wirelessly in a single-hop or multi-hop fashion

WSAN applications

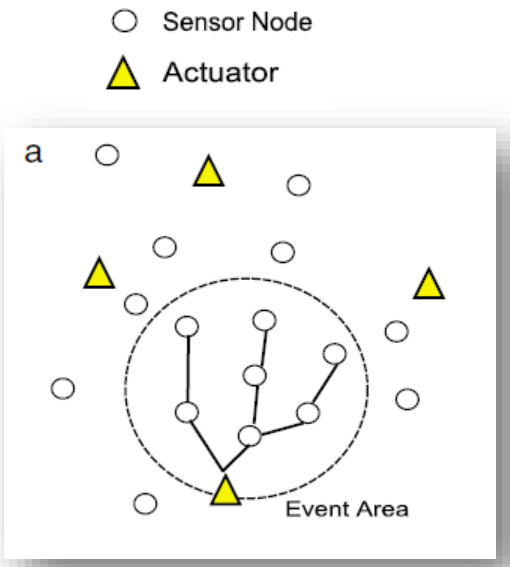
Applications	Sensor types	Actuator types	Delay tolerance	Number of sensors	Number of actuators
Home automation [37,18,25]	Temperature, humidity, light, acoustic and occupancy sensors.	Relays, light, switches, motors, noise-masking system adjusting and valves.	2 s	More than 100 sensor nodes	20–50 actuator nodes
Animal control [55]	Location and velocity meter sensors	Stimuli board	500 ms	20–30 sensor nodes	20–30 actuator nodes
Infrastructure health monitoring [44,65,28,29,23]	Velocity meter and piezoelectric sensors	Hydraulic or piezoelectric shakers	1 Day	20–100 sensor nodes	20–100 actuator nodes
Precision agriculture [32,42,17,61]	Temperature, humidity, light and soil moisture sensors	Valves, relays light switches	10 min	More than 100 sensor nodes	20–30 actuator nodes
Intelligent Buildings [35,11,26,39]	Smoke detectors, glass break and motion sensors	Water sprinklers and cameras	2 s	20–50 sensor nodes	20–30 actuator nodes
Sewer management [52,31]	Water level	Valves or city sewers	14 min	More than 100 sensor nodes	More than 100 actuator nodes
Traffic light control [68,46]	Magnetic sensor	Relays of traffic lamp	10 s	More than 100 sensor nodes	More than 100 actuator nodes

Coordination among sensors

- sensor nodes in the event area communicate with each other to find the nearest actuator node(s) that covers the area & sufficient energy for the task.

(+) actuator nodes are excluded from this coordination effort.

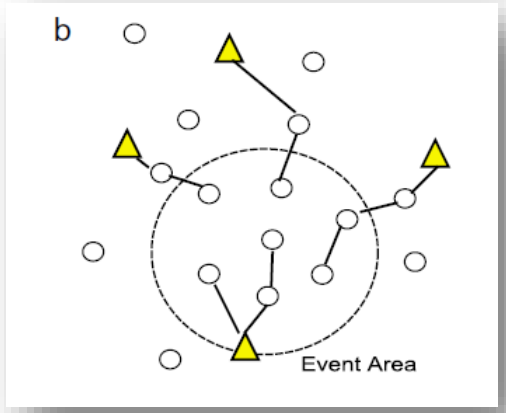
(-) sensor nodes around the event area will deplete their energy quicker.



- each sensor node independently selects an actuator node

(+) No sensor-to-sensor coordination,

(-) may overload a given actuator or cause sensor nodes around an actuator node to deplete their energy quicker.

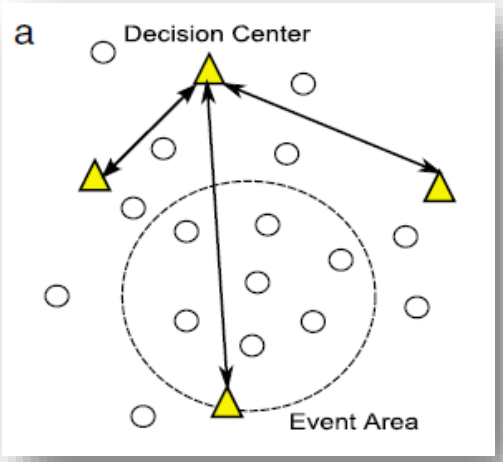


Coordination among actuators

- Centralized decision category

(+) the decision center is able to select the best actuator nodes to carry out a task

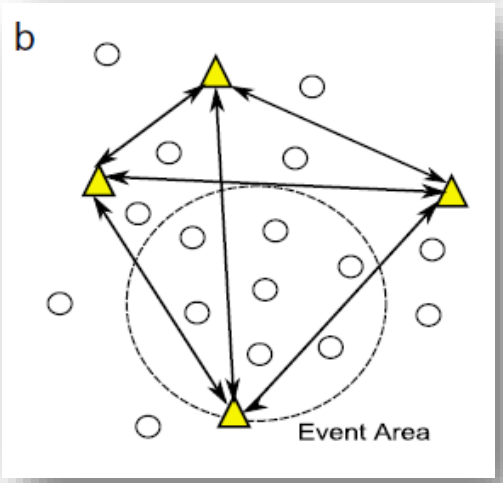
(-) increased end-to-end delays



- Distributed decision category

(+) reduced actuator response time

(-) increased energy consumption (communications)



Comparison of sensor-to-actuator proposals.

Prior works	Coordination	Actuator localization	Actuator mobility	Energy conservation techniques
Melodia et al. [40]	Multiple actuators	Dynamic clustering	No	Rotate cluster head role and use alternate paths
Shah et al. [45]	Single actuator	Static clustering	No	Event-based clustering
Zeng et al. [64]	Multiple actuators	Actuator nodes periodically broadcast their position	Yes	No consideration
Melodia et al. [39]	Multiple actuators	Voronoi diagram and Kalman filtering to predict position.	Yes	Exploit Voronoi diagrams to scope broadcast messages

Comparison of routing protocols.

Prior works	Route discovery	Route maintenance	Delay bound	Energy conservation techniques
Dinh et al. [51]	Sensor nodes	Sensor nodes periodically exchange routing tables	No	None
Hu et al. [22]	Actuator nodes	Actuator nodes	Select the nearest actuator node	Anycast tree
Boukerche et al. [7]	Actuator nodes	Actuator nodes periodically broadcast subscription messages	Select the path that meets packet delivery time	Distributes packets between existing paths
Cayirci et al. [14]	Actuator nodes	Actuator nodes periodically broadcast subscription messages	Vary transmission range	Multicast tree
Durresi et al. [36]	Sensor nodes	On demand	No	Sleep and wake-up
Sanchez et al. [43]	Sensor nodes	None.	No	Multicast tree

Comparison of transport protocols.

Prior works	Delay bound	Reliability	Service priority?	Energy efficiency
Zhou et al. [34]	By changing transmission range of sensor nodes to shorten or elongate a forwarding path	No consideration	Yes	Forwarding paths are selected based on required delivery time of packets
Gungor et al. [2]	Scheduling packets according to their deadlines and dynamic adjustment of sampling rate	TCEF scheduling algorithm and the changing sampling rate	Yes	During congestion, sampling rate of sensor nodes is reduced
Ngai et al. [67,66]	Forwards packets on the least congested path. They also consider the use of mobile actuators	Measuring links loss rate and sending multiple copies of a packet along many paths	Yes	No
Melodia et al. [40]	Probabilistic transmission schedule, and increasing nodes' transmission range to reduce route length	Probabilistic recruitment of sensor nodes	No	Adjust sensor nodes transmission range
Melodia et al. [39]	Changing sensor nodes transmission range and shifting load to less congestion actuator nodes	Changing sensor nodes' transmission range and shifting packets onto less congested paths or actuators	No	Reduce sensor nodes' transmission range if event reliability is above a given threshold

Reading List

Abu Alsheikh, Mohammad, et al. "Machine learning in wireless sensor networks: Algorithms, strategies, and applications." *Communications Surveys & Tutorials, IEEE* 16.4 (2014): 1996-2018.

Flouri, K., B. Beferull-Lozano, and P. Tsakalides. "Optimal gossip algorithm for distributed consensus SVM training in wireless sensor networks." *Digital Signal Processing, 2009 16th International Conference on*. IEEE, 2009.

Predd, Joel B., Sanjeev R. Kulkarni, and H. Vincent Poor. "A collaborative training algorithm for distributed learning." *Information Theory, IEEE Transactions on* 55.4 (2009): 1856-1871.