

Lecture 2

# Distance-based classifiers

Intellectual system  
(Machine Learning)  
Andrey Filchenkov

12.09.2017

# Lecture plan

- Distance-based classifiers (1NN)
  - Parameters tuning
  - Generalized distance-based classifiers
  - Prototype selection
  - Anomaly detection
- 
- The presentation is partly prepared with materials of the K.V. Vorontsov's course "Machine Learning".
  - Slides are available online:  
**[goo.gl/fDBgMq](http://goo.gl/fDBgMq)**

# Lecture plan

- Distance-based classifiers (1NN)
- Parameters tuning
- Generalized distance-based classifiers
- Prototype selection
- Anomaly detection

# Problem formulation

$X$  is an object set,  $Y$  is an answer set,

$y : X \rightarrow Y$  is an unknown dependency,  $|Y| \ll \infty$

$X^\ell = \{x_1, \dots, x_n\}$  is a training sample,

$T^\ell = \{(x_1, y_1), \dots, (x_\ell, y_\ell)\}$  is a set of examples.

**Task:** return an algorithm  $a : X \rightarrow Y$ .

What is this task?

# Classification problem formulation

$X$  is an object set,  $Y$  is an answer set,

$y : X \rightarrow Y$  is unknown dependency,  $|Y| \ll \infty$

$X^\ell = \{x_1, \dots, x_n\}$  is training sample,

$T^\ell = \{(x_1, y_1), \dots, (x_\ell, y_\ell)\}$  is set of examples.

**Task:** return an algorithm  $a : X \rightarrow Y$ .

What is this task?

Classification, because  $|Y| \ll \infty$ .



# Duck test

## Duck test:

If it looks like a duck, swims like a duck,  
and quacks like a duck, then it **probably** is a  
duck.

# Duck test

Duck test: *if it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck.*

	Looks	Swims	Quacks	A duck?
	like a duck	like a duck	like a duck	Probably, a duck
	totally not like a duck	can be a duck	not like a duck	Probably, not a duck

# How is the classifier formalized?

What is the training sample?

Many ducks, many non-ducks (unducks).

What is classification procedure?

1. Ducks were described with **key features**.
2. **Similarity concept** was used.
3. Logical separator was used for classification.



# Main idea

**Key hypothesis:** similar objects belong to same class.

**Main idea:** for an object we have to find a class, in which objects are the most similar to the given one.

- Reasoning by analogy (case-based)
- Lazy learning

# Formalization of “similarity”

“Similarity” is a distance between objects. We will talk about **metrics**.

**Distance:**  $\rho: X \times X \rightarrow [0; +\infty)$ .

**Metric space** is a set with a metric  $\rho(x, y)$ , defined on it.

# Commonly used metrics

## Minkowski distance:

$$p(x, y) = \left( \sum_i |x_i - y_i|^p \right)^{\frac{1}{p}},$$

when  $p = 2$ , it is the Euclidian distance;  
when  $p = 1$ , it is the Manhattan distance.

## Mahalanobis distance:

$$p(x, y) = \sqrt{(x - y)^\top S^{-1} (x - y)},$$

where  $S$  is covariance matrix for  $x$  and  $y$ .

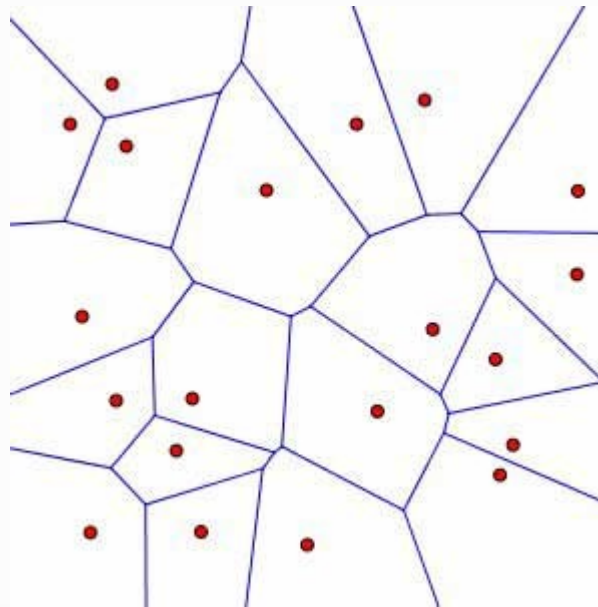
# Nearest neighbor method (1NN)

$x_{(u,1)}$  is **nearest neighbor** of  $u$ :  $x_{(u,1)} = \operatorname{argmin}_{x \in X^\ell} \rho(u, x)$ .

Classifier:

$$a(u, T^\ell) = y_{(u,1)}.$$

**Voronoi diagram:**



# 1NN discussion

Advantages:

- simplicity;
- lucidity;
- interpretability.

Disadvantage:

- sensibility to noise;
- low efficacy;
- no parameters (explicitly);
- necessity to store all the examples.

# Lecture plan

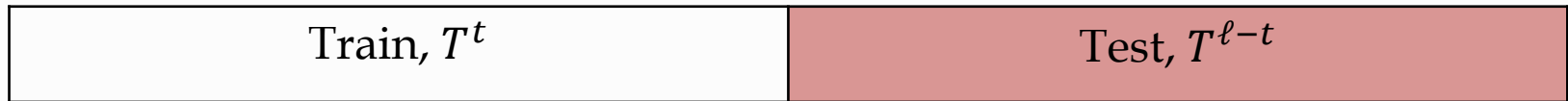
- Distance-based classifiers (1NN)
- Parameters tuning
- Generalized distance-based classifiers
- Prototype selection
- Anomaly detection

# Hold-out validation

Hold-out validation, HO

Split training sample into two parts:

$$T^\ell = T^t \cup T^{\ell-t}$$



Solve the optimization problem:

$$\text{HO}(\mu, T^t, T^{\ell-t}) = Q(\mu(T^t), T^{\ell-t}) \rightarrow \min$$

# Complete cross-validation

Choose value of  $t$ .

Split the sample with all the possible ways on  $T^t$  and  $T^{\ell-t}$ .



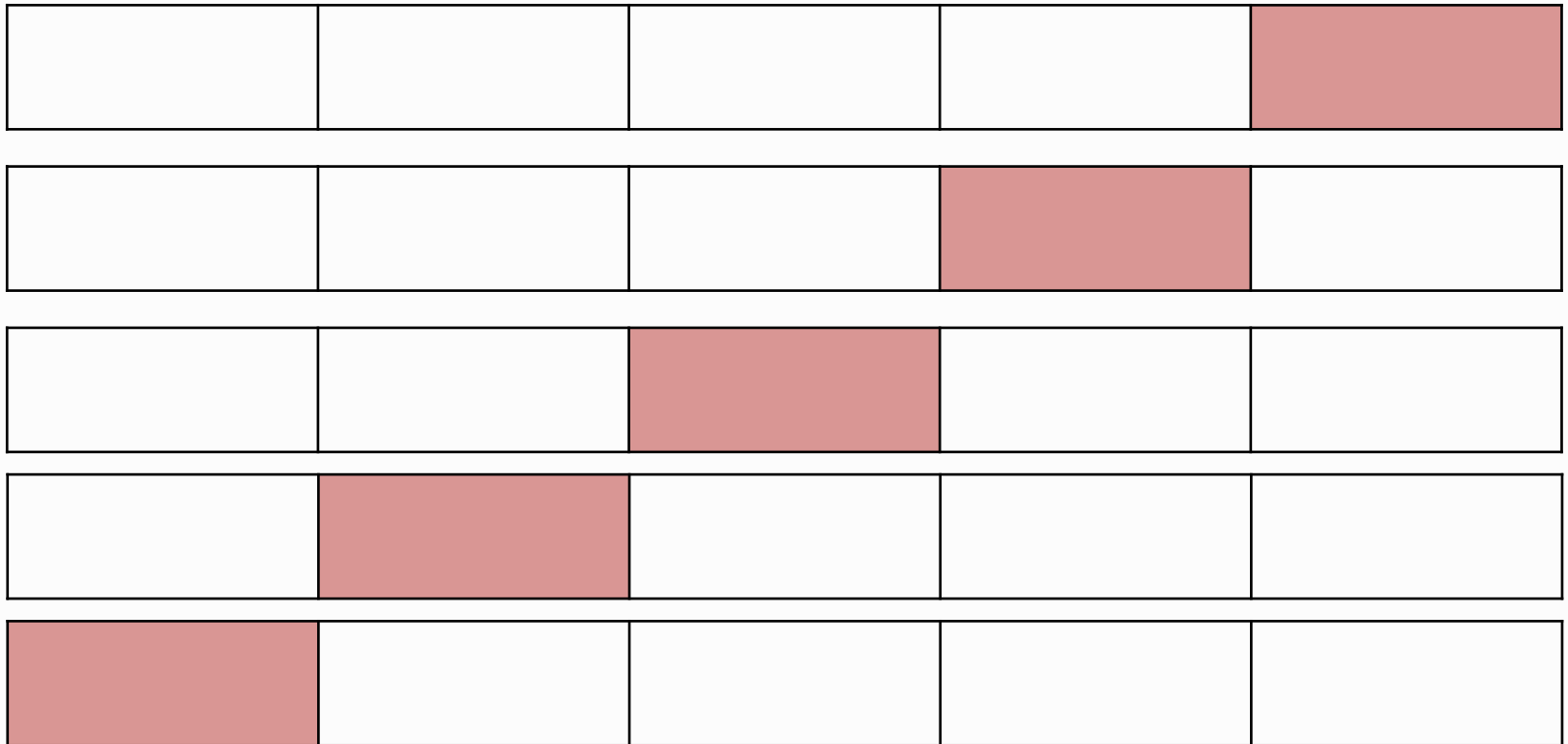
Solve the optimization problem:

$$\text{CVV}_t = \frac{1}{C_{\ell}^{\ell-t}} \sum_{T^{\ell} = T^{\ell-t} \cup T^t} Q(\mu(T^t), T^{\ell-t}) \rightarrow \min$$



# Cross-validation

Split sample to  $k$  parts  $k$  times



# *k*-fold cross-validation

*k*-fold cross-validation

Each of *k* blocks is a test sample once.

*k* is usually 10 (5 is small sample size).

Split  $T^\ell = F_1 \cup \dots \cup F_k$ ,  $|F_i| \approx \frac{\ell}{k}$ .

Solve the optimization problem:

$$CV_k = \frac{1}{k} \sum_{i=1}^k Q(\mu(T^\ell \setminus F_i), F_i) \rightarrow \min.$$

# $t \times k$ -fold cross-validation

Repeat  $t$  times: split sample on  $k$  blocks, each of  $k$  blocks is a test sample once.

$k$  is usually 10 ,  $t$  is usually 10 or less.

Split  $T^\ell$   $t$  times randomly:

$$T^\ell = F_{(1,1)} \cup \dots \cup F_{(k,1)} = \dots = F_{(1,t)} \cup \dots \cup F_{(k,t)},$$

$$|F_{(i,j)}| \approx \frac{\ell}{k}.$$

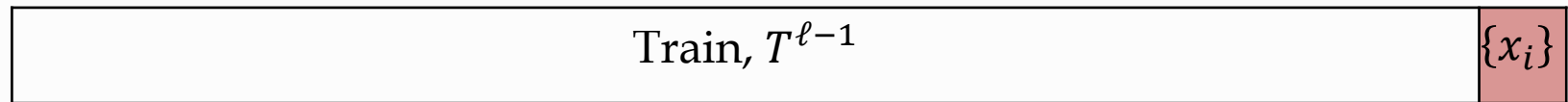
Solve the optimization problem:

$$\text{CV}_{t \times k} = \frac{1}{tk} \sum_{j=1}^t \sum_{i=1}^k Q(\mu(T^\ell \setminus F_{(i,j)}), F_{(i,j)}) \rightarrow \min.$$

# Leave one out

Leave-one-out cross-validation, LOO

Split sample into  $\ell - 1$  and 1 objects  $\ell$  times.



Solve the optimization problem:

$$\text{LOO} = \frac{1}{\ell} \sum_{i=1}^{\ell} Q(\mu(T^{\ell} \setminus p_i), p_i) \rightarrow \min.$$

where  $p_i = (x_i, y_i)$ .

# Lecture plan

- Distance-based classifiers (1NN)
- Parameters tuning
- Generalized distance-based classifiers
- Prototype selection
- Anomaly detection

# How can it be improved?

- More complicated model (more parameters)
- Distance choosing
- Dimension reduction
- Usage of good structures for storing data
- Object set thinning
- Noise filtering
- Prototype selection

# $k$ NN

Choose a distance  $\rho$ .

Sort objects:

$$\rho(u, x_{(u,1)}) \leq \rho(u, x_{(u,2)}) \leq \dots \leq \rho(u, x_{(u,\ell)}).$$

**Algorithm  $k$ NN:**

$$a(u; T^\ell) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u, i) = y][i \leq k],$$

$$a(u; T^\ell) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^k [y(u, i) = y].$$

# Optimization of $k$

Is equal to the problem of LOO quality functional minimization:

$$\text{LOO}(k, T^\ell) = \sum_{i=1}^{\ell} [a(x_i; T^\ell \setminus \{(x_i, y_i)\}, k) \neq y_i] \rightarrow \min_k$$



# Generalized metric classifier

$$a(u; T^\ell) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u, i) = y] w(i, u),$$

where  $w(i, u)$  is a function representing importance of  $i$ th neighbor of  $u$ .

$C_y(u) = \sum_{i=1}^{\ell} [y(u, i) = y] w(i, u)$  is estimation of object  $u$  closeness to class  $y$ .

$$a(u; T^\ell) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} C_y(u).$$

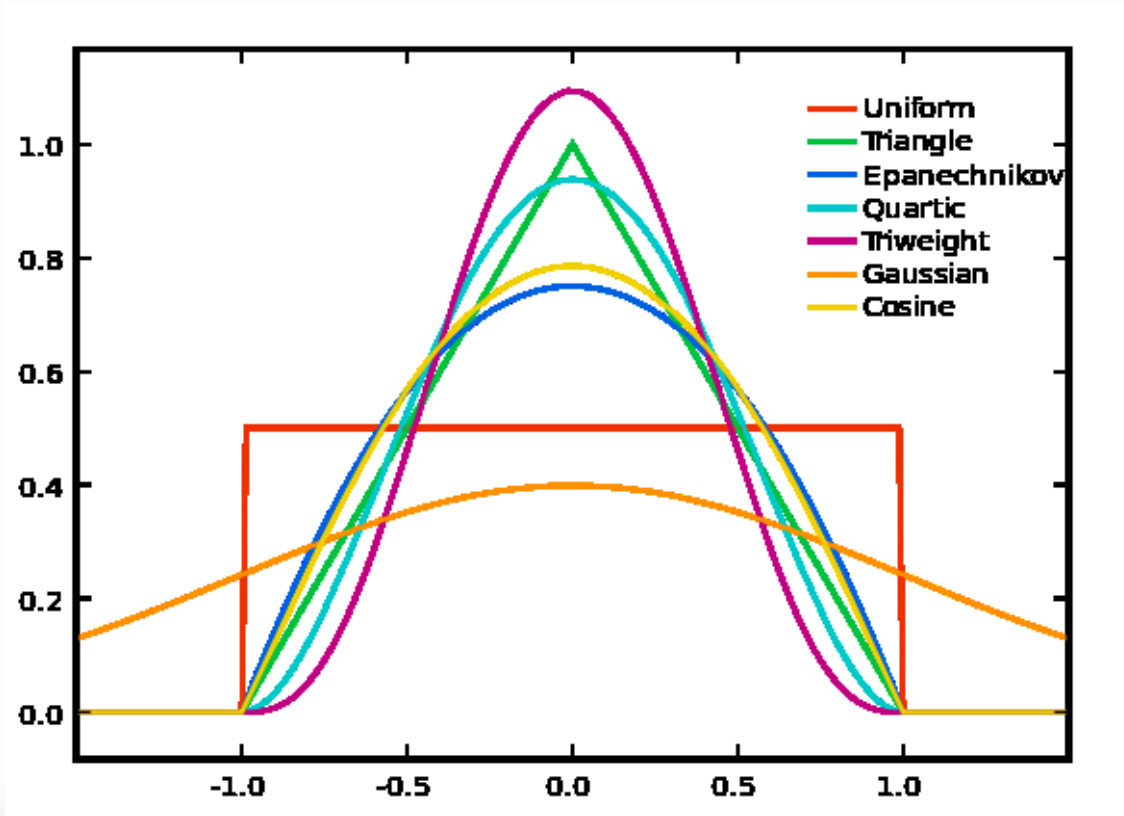
# What can be chosen as $w$ ?

$w(i, u)$ :

- linearly decreasing functions;
- exponentially decreasing functions;
- kernel functions.

# Kernel function

**Kernel function**  $K(x)$  is symmetric non-negative function,  $\int_{-\infty}^{+\infty} K(x) dx = 1$ .



# Parzen-Rosenblatt window

With fixed window width:

$$\begin{aligned} a(u; T^\ell; \textcolor{red}{h}; K) &= \\ &= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u, i) = y] K \left( \frac{\rho(u, x_{(u, i)})}{\textcolor{red}{h}} \right), \end{aligned}$$

With variable window width:

$$\begin{aligned} a(u; T^\ell; \textcolor{red}{k}; K) &= \\ &= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u, i) = y] K \left( \frac{\rho(u, x_{(u, i)})}{\textcolor{red}{\rho(u, x_{(u, k+1)})}} \right). \end{aligned}$$

# Parzen-Rosenblatt window

With fixed window width:

$$\begin{aligned} a(u; T^\ell; \textcolor{red}{h}; K) &= \\ &= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u, i) = y] K \left( \frac{\rho(u, x_{(u, i)})}{\textcolor{red}{h}} \right), \end{aligned}$$

With variable window width:

$$\begin{aligned} a(u; T^\ell; \textcolor{red}{k}; K) &= \\ &= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u, i) = y] K \left( \frac{\rho(u, x_{(u, i)})}{\textcolor{red}{\rho(u, x_{(u, k+1)})}} \right). \end{aligned}$$

# Distance selection (learning)

Distance can be learned.

Example (weighted Minkowski):

$$p(x, y) = \left( \sum_i w_i |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

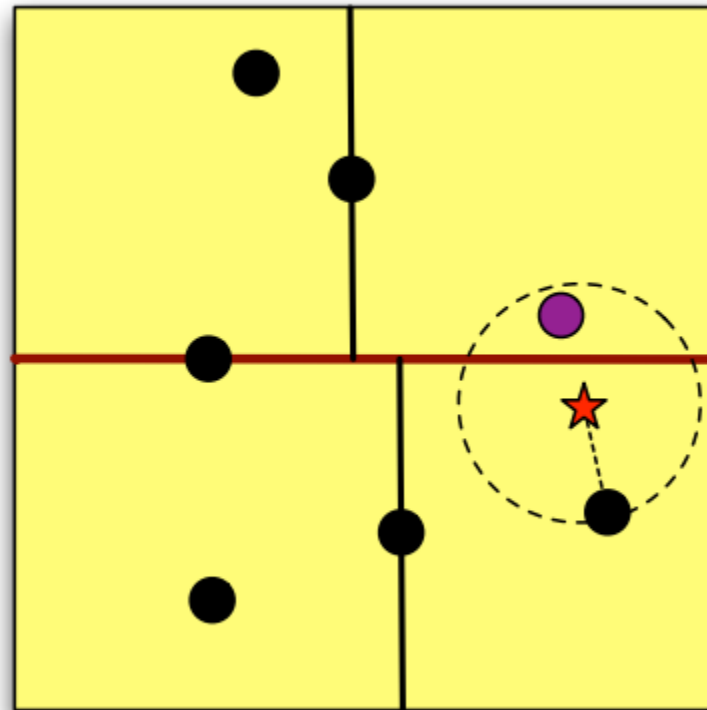
Now the problem is how to choose coefficients  $w_i$ .

When  $w_i = 0$ , the feature is thrown away (feature selection).

A kernelization can be applied.

# Structure for storing data

Different greed-like structures can be used. The most effective is *k-d-trees*:



# Lecture plan

- Distance-based classifiers (1NN)
- Parameters tuning
- Generalized distance-based classifiers
- **Prototype selection**
- Anomaly detection



# Margin

**Margin** of object  $x_i$  with respect to algorithm  $a(u)$ :

$$M(x_i) = C_{y_i}(x_i) - \max_{y \in Y \setminus \{y_i\}} C_y(x_i).$$

Margin is the measure of object “typicalness” for its class. The higher margin is, the more typical the object is.

# Ranking based on margin

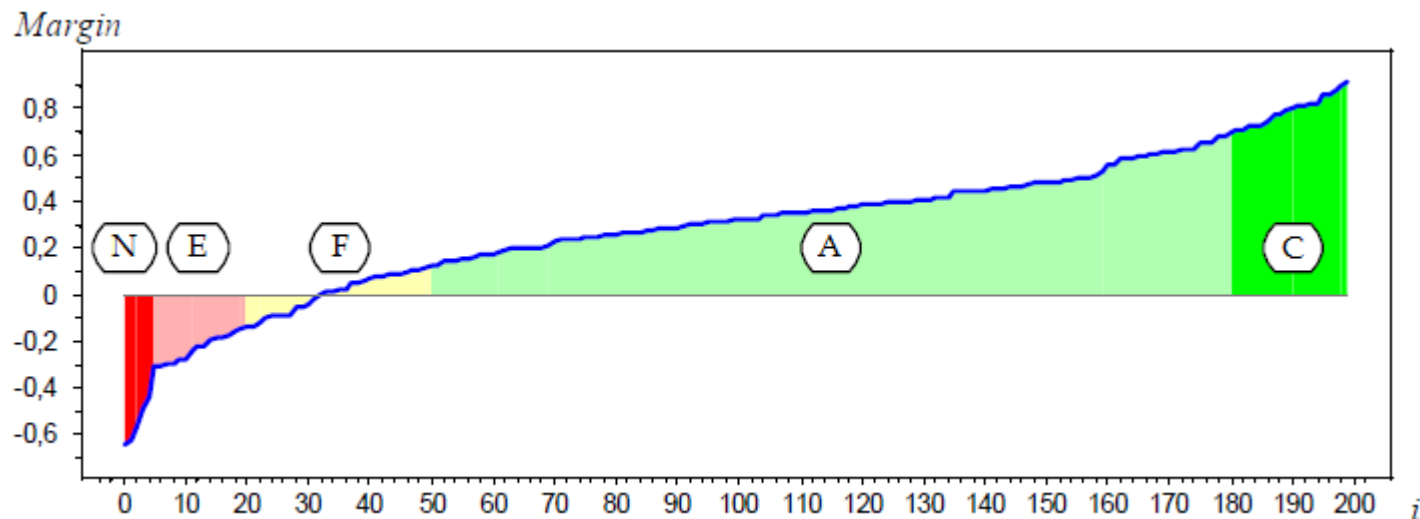
C – **core** (base for classification)

A – **accompaniment** (can be deleted)

F – **frontier** (classification is instable)

E – **erroneous** (misclassification because of bad model)

N – **noisy** (misclassification because of bad data)



# Prototype selection problem

**Prototype selection problem** is how to choose optimal subset of object  $\Omega \subseteq X^\ell$ .

Distance-based classifier:

$$a(u; \Omega) = \operatorname{argmax}_{y \in Y} \sum_{x_i \in \Omega} [y(u, i) = y] w(i, u) .$$

# Prototype selection solutions

The problem is NP-hard, so there are lots of approximate solutions.

Two main approaches:

- 1) linear programming relaxation;
- 2) greedy algorithms.

Plenty of heuristic approaches.

# Lecture plan

- Distance-based classifiers (1NN)
- Parameters tuning
- Generalized distance-based classifiers
- Prototype selection
- **Anomaly detection**

# How is the classifier formalized?

What is the training sample?

Many ducks, **many non-ducks (unducks)**.

What is classification procedure?

1. Ducks were described with **key features**.
2. **Similarity concept** was used.
3. Logical separator was used for classification.

# One-class classification problem

The duck test in one-class classifier: we can make suggestions only about belonging to a certain class (ducks). Nothing is said about other classes.

**Anomaly** (outlier, exception, surprise) **detection** is the problem of one-class classification.

# More precise definition

The definition is determined by the class of problems we solve:

- Labels are known for positive and negative examples: **supervised** rare-class detection.
- Labels are known only for positive examples: **semi-supervised** learning.
- Labels are unknown: **unsupervised** small-cluster detection.

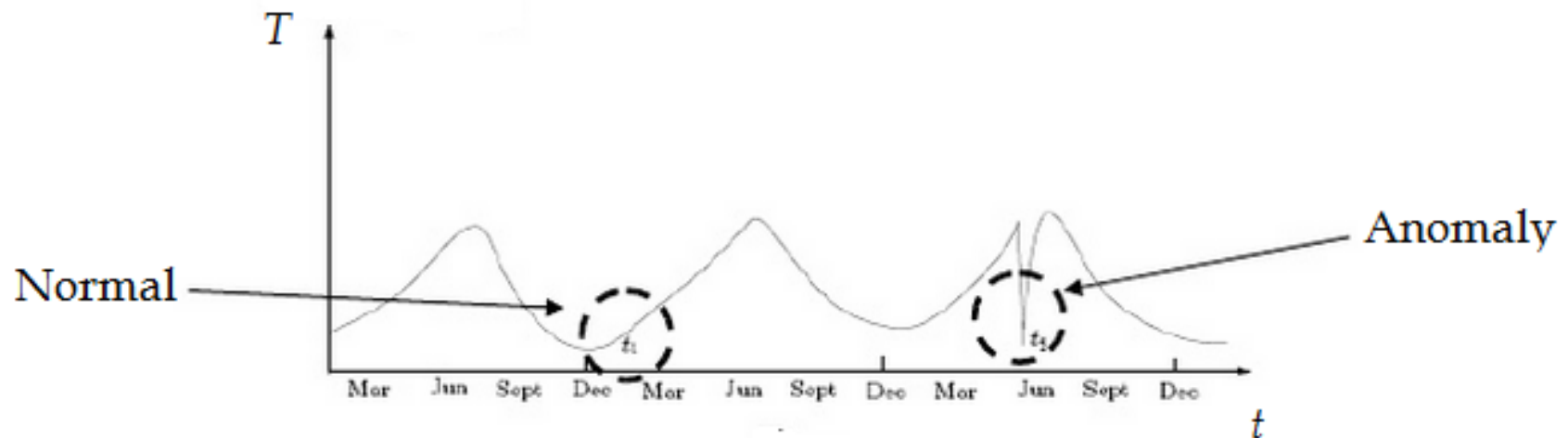


# Anomaly detection taxonomy

- Point anomaly detection
- Contextual anomaly detection
- Collective anomaly detection

# Contextual anomaly detection

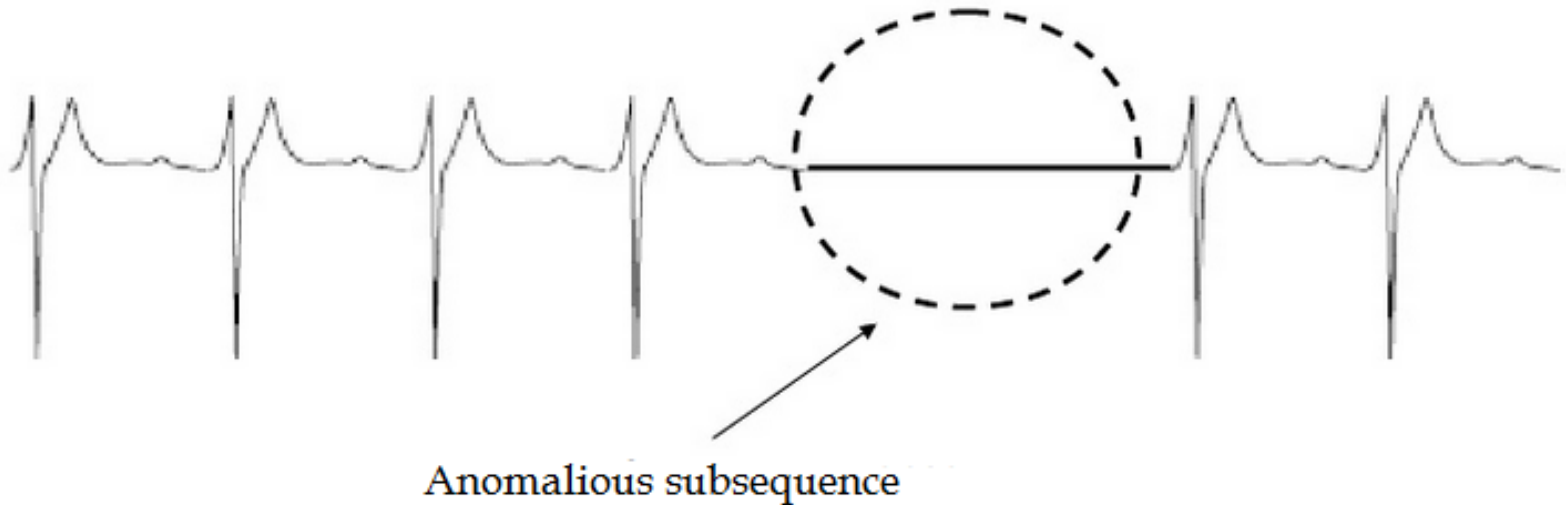
**Assumption:** all normal instances within a context will be similar (in terms of behavioral attributes), while the anomalies will be different.



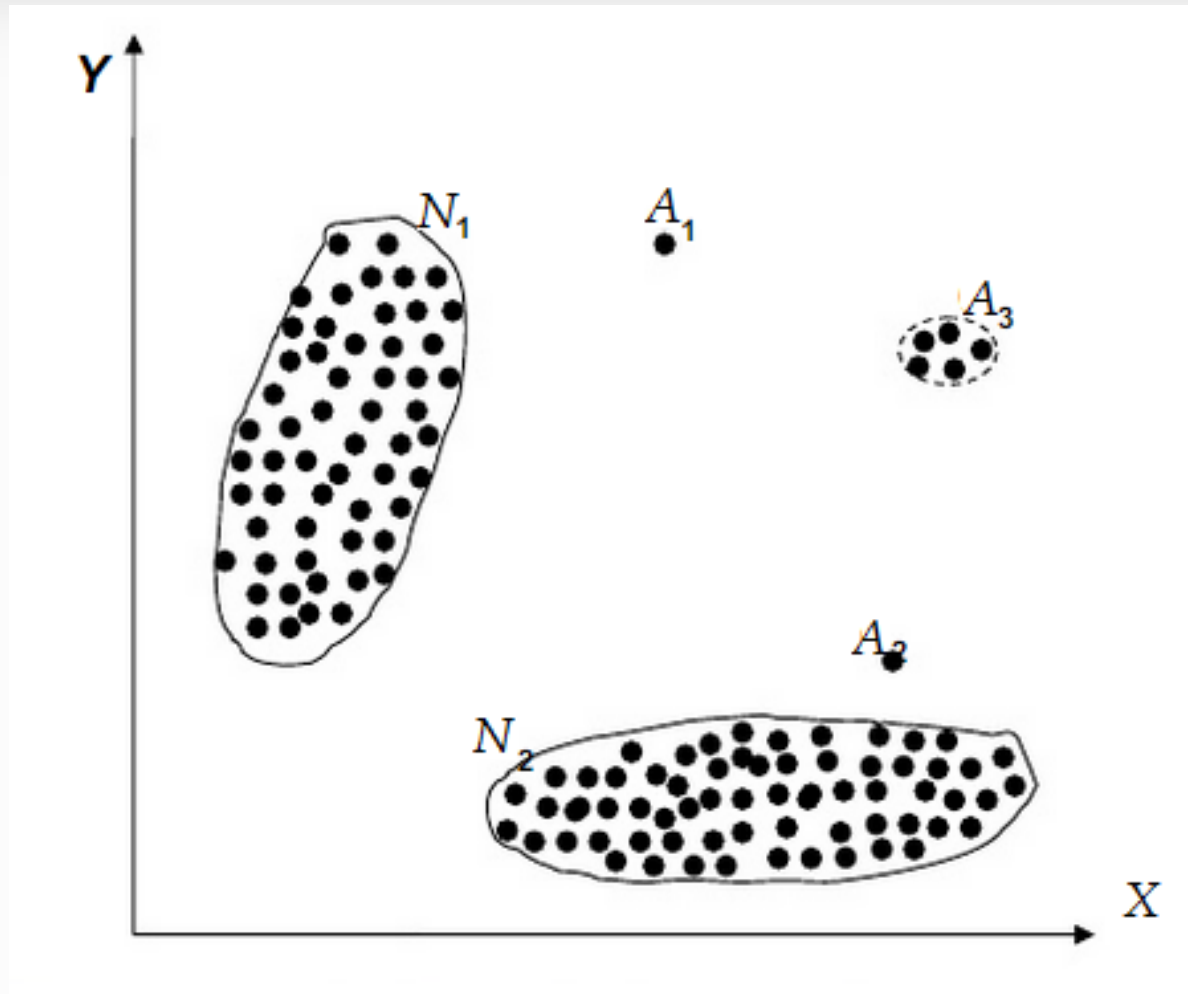
# Collective anomaly detection

Data instances are related:

- sequential data
- spatial data
- graph data



# Point anomaly detection



# NN anomaly detection

Two types of methods:

- **distance based methods:** anomalies are data points most distant from other points
- **density based methods:** anomalies are data points in low density regions