Lecture 13

Label irregularity

Information Systems
(Machine Learning)

Andrey Filchenkov

13.12.2018

Lecture plan

- Label irregularity
- Semi-supervised learning
- Active learning
- Data sampling
- One-shot learning
- Anomaly detection
- Outliers

- The presentation is prepared with materials of
  - K.V. Vorontsov's course "Machine Leaning"
  - V. Gulins's course "Data Mining"
  - A. Ng's course "Deep Learning"
- Slides are available online: **goo.gl/BspjhF**

Lecture plan

- Label irregularity
- Semi-supervised learning
- Active learning
- Data sampling
- One-shot learning
- Anomaly detection
- Outliers

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.

Hidden assumption and its problems

We assume that the classes are equal and equally represented with labeled data. But we may have

1. Classes that are not equal (“non-ducks” is not a class)
2. Imbalanced classes
3. Only a few of labeled objects in a class
4. Extremely many classes
5. None of labeled objects in a class

Lecture plan

- Label irregularity
- Semi-supervised learning
- Active learning
- Data sampling
- One-shot learning
- Anomaly detection
- Outliers

Problem formulation

A training sample is given, which is

$$\{(x_1, y_1), \dots, (x_\ell, y_\ell), x_{\ell+1}, \dots, x_{\ell+m}\} = T^\ell \cup U^m,$$

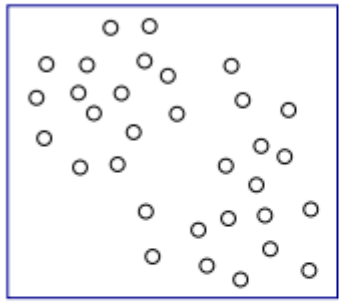
where $\ell \ll m$.

Solve as supervised problem (on T^ℓ, "forgetting" about U^m)

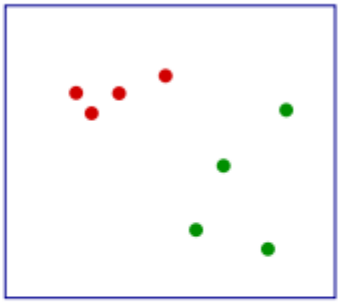
Solve as unsupervised problem (on $X^\ell \cup U^m$, "forgetting" about Y^m).

Semi-supervised and active learning

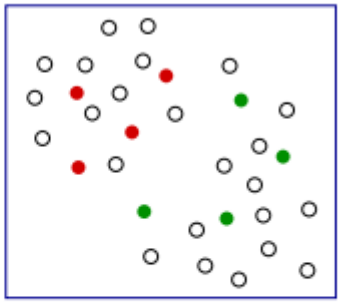
Active learning is a case of **semi-supervised learning**



Unlabeled points



Supervised learning



Semisupervised and active learning

Why is it important to solve this problem?

Usually, it is cheap to get objects and it is expensive to label objects.

Object mining is automated and object-labeling is expert-based.

Typical example: data from Internet (posts, pictures, articles) or generic data.

Semi-supervised learning approaches

Three approaches:

- Solve with native methods
- Solve with supervised algorithms without estimating error on unlabeled objects
- Solve with unsupervised algorithm achieving clusters which contains at least one object and all objects belonging to a cluster have the same label

Self-training algorithm

Train some f on T^{ℓ}

Predict $f(x)$ on X^{m}

Add $(x, f(x))$ to T^{ℓ} and go to start

Options:

- Add the most confident x
- Add all x
- Add all x weighed with confidence

Self-training algorithm

Train some f on T^{ℓ}

Predict $f(x)$ on X^{m}

Add $(x, f(x))$ to T^{ℓ} and go to start

Options:

- Add the most confident x
- Add all x
- Add all x weighed with confidence

Co-training and multi-view training

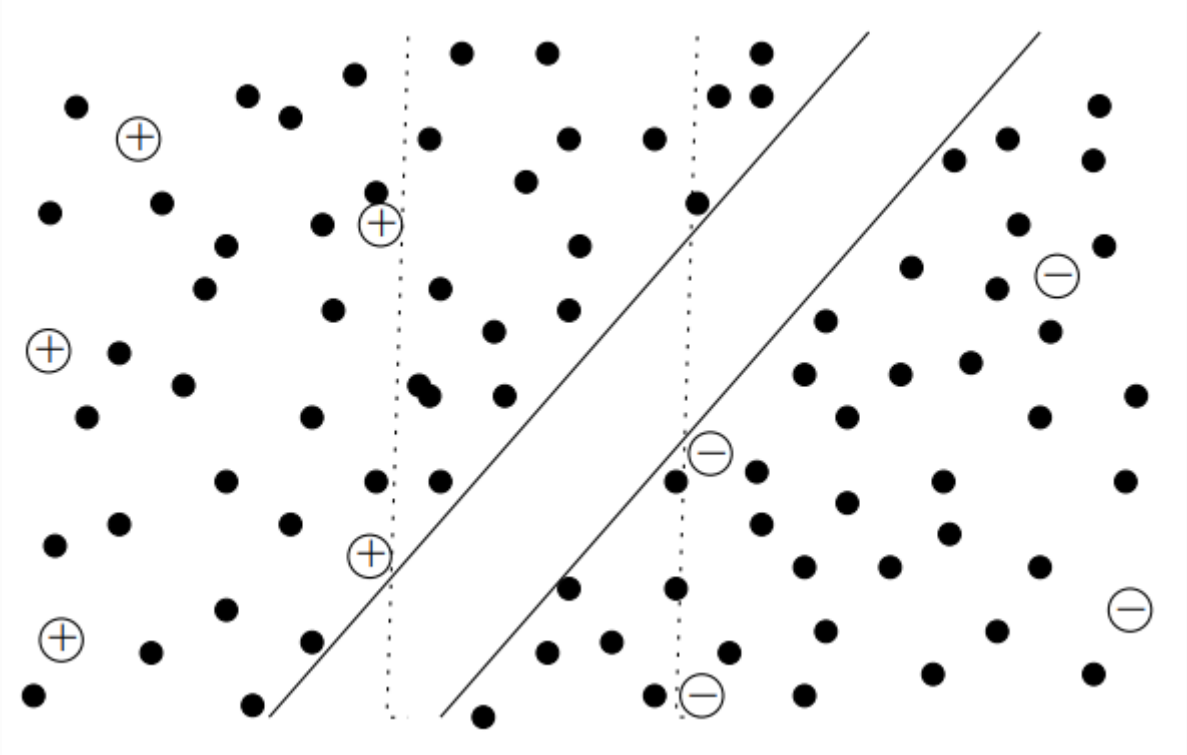
Take several independent algorithms and train them independently

For co-training, we use only 2 classifiers and adds labels given by the first classifier to the training set of the second one and visa verse.

For multi-view training, we do the very same in the one-versus-all manner.

Semi-supervised support vector machine

Main idea of S3VM is maximizing “unlabeled data margin”



S3VM loss

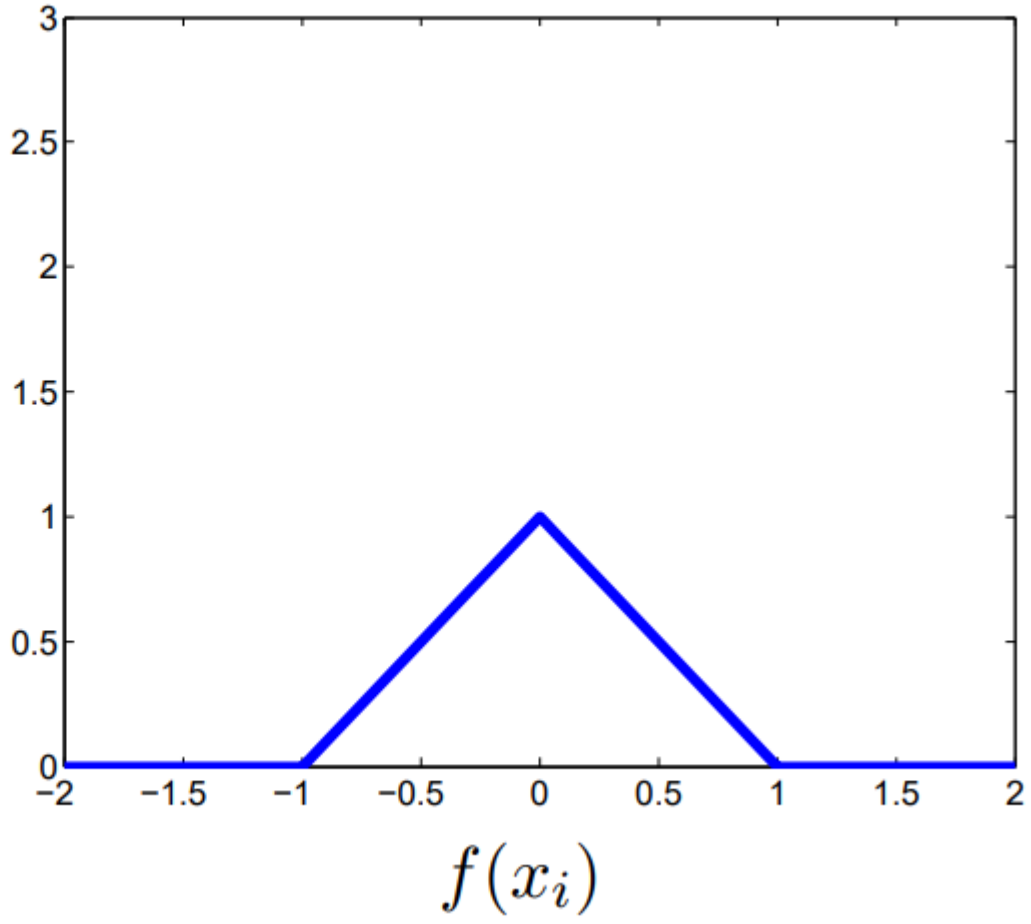
This is archived by using loss that incorporates distance to the margin of unlabeled points:

$$\sum_{i=1}^{\ell}\left(1 - M_i(w, w_0)\right)_+ + \frac{1}{2C}||w||^2 +$$

$$+ \sum_{j=1}^{m}\left(1 - |M_j(w, w_0)|\right)_+ \to \min_{w, w_0}.$$

Hat loss

$$(1 - |f(x_i)|)_+$$



Hat loss problem

It turns problem into non-convex optimization

Ways to solve:

- Use fine-tuned non-convex optimization
- Make hat loss smooth and apply gradient descent
- Use upper bound and split into several convex optimization problems

Lecture plan

- Label irregularity
- Semi-supervised learning
- Active learning
- Data sampling
- One-shot learning
- Anomaly detection
- Outliers

Active learning

In semi-supervised learning we need to build a model. It is between supervised and unsupervised learning.

In **active learning** we start as in semi-supervised learning, but we can ask **Oracle** about labels. This questions are expensive.

The goal: to find the best strategy to achieve good model quality with minimal number of questions.

Uncertainty sampling

Main idea: query the instances on which you are most uncertain.

Say that $\hat{y} = \arg\max_y P_\theta(y|x)$ is the class with the highest posteriori probability under the model parametrized with θ.

Uncertainty sampling variants

Last confident:

$$x_{LC}^{*} = \arg\max_{x} 1 - P_{\theta}(\hat{y}|x)\,,$$

Margin sampling:

$$x_{M}^{*} = \arg\max_{x} P_{\theta}(\hat{y}_1|x) - P_{\theta}(\hat{y}_2|x)\,,$$

General uncertain (entropy):

$$x_{M}^{*} = \arg\max_{x} -\sum_{i} P_{\theta}(y_i|x)\log P_{\theta}(y_i|x)\,.$$

Query-by-committee

Assume we have a committee of trained models $C = \{\theta^{(1)}, \dots, \theta^{(|C|)}\}$. Each member is allowed to vote on one of instances from a fix set H.

Main idea: query instances where agreement is not achieved.

Vote entropy:

$$x^*_{VE} = \arg\max_{x \in H} -\sum_i \frac{V(y_i)}{|C|} \log \frac{V(y_i)}{|C|},$$

where $V(y_i)$ is number of votes for label y_i for object x. Can be generalized with Kullback-Leibler Divergence.

Recalling semi-supervised learning

uncertainty sampling
query instances the model is least confident about

self-training
propagate confident labelings among unlabeled data

query-by-committee (QBC)
use ensembles to rapidly reduce the version space

co-training
multi-view learning
use ensembles with multiple views to constrain the version space w.r.t. unlabeled data

Expected model change

Main idea: query instances which can give the highest model change:

$$x^*_{EGL} = \arg\max_x - \sum_i P_\theta(y_i|x)\|\nabla L(x, y_i)\|.$$

where $\nabla L(x, y_i)$ is loss function gradient.

Expected error reduction

Main idea: query the instances which help to reduce the generalization error the most.

Minimization of 0-1-loss:

$$x^*_{0/1} = \arg\max_x \sum_i P_\theta(y_i|x) \sum_{x' \in H} 1 - P_{\theta^+(x', y_i)}(\hat{y}|x'),$$

Minimization of expected log-loss:

$$x^*_{0/1} = \arg\max_x \sum_i P_\theta(y_i|x) \left(-\sum_j \sum_{x' \in H} P_+(j) \log P_+(j) \right),$$

where $P_+(j) = P_{\theta^+(x', y_j)}(y_j|x')$

Lecture plan

- Label irregularity
- Semi-supervised learning
- Active learning
- Data sampling
- One-shot learning
- Anomaly detection
- Outliers

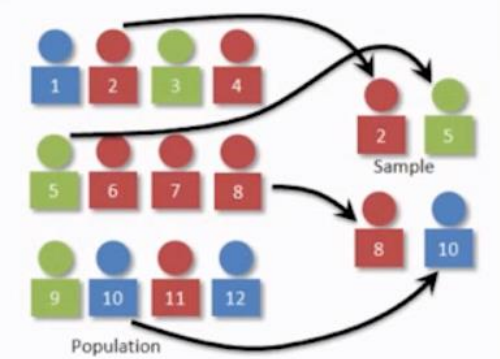
Two sampling strategies

We can sample from a bigger class a subsample, or upsample from a smaller class

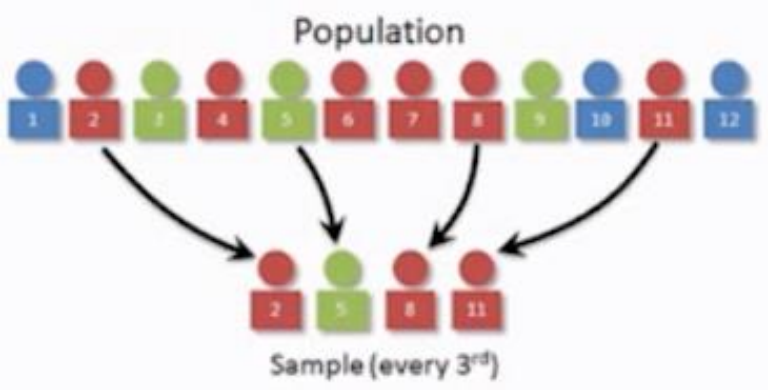
Subsampling this days is used mostly for data exploration and results validation

Subsampling strategies

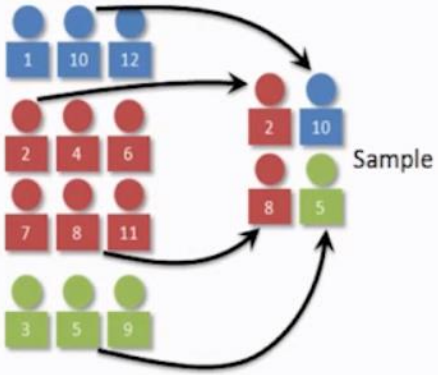
Simple random sampling



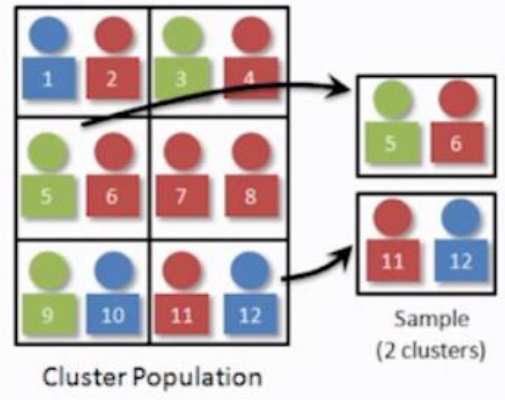
Systematic sampling



Stratified sampling



Cluster sampling



SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) for more intelligent data sampling

1) Take a point a
2) Find its k neighbors of the same class
3) Randomly choose one, b
4) Randomly choose a point in segment (a, b)

What else?

- Data augmentation
- Generative adversarial networks
- Noisy labelled data

Important to remember

All artificial data can be contained only in the test set.

Lecture plan

- Label irregularity
- Semi-supervised learning
- Active learning
- Data sampling
- One-shot learning
- Anomaly detection
- Outliers

One-shot learning concept

The main idea is to learn a system that can simply add new classes without retraining all the parameters.

Think of face recognition system.

How can we design such a system?

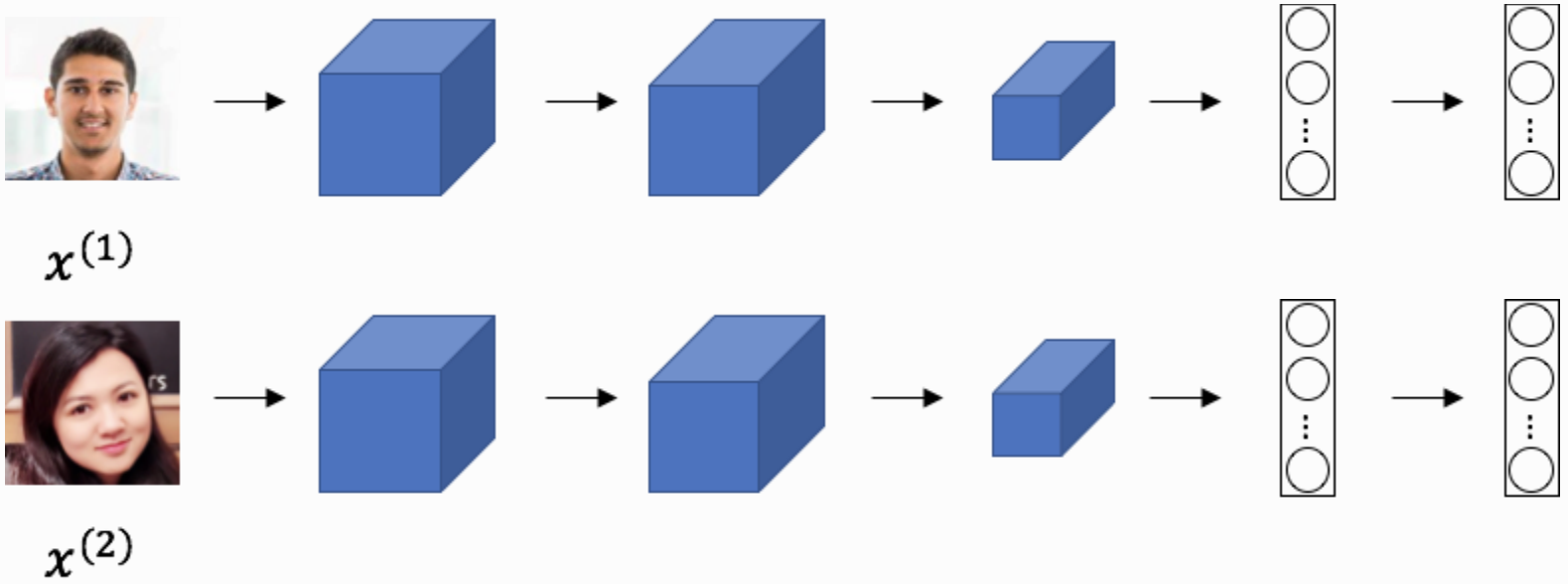
Interpretable concepts

One-shot learning is based on distance-based learning and performing distant-based classification.

Adding centroids does not make you to reevaluate all other centroids.

Siamese network

Siamese network is two identical networks that process two inputs and must decide if they belong to the same class or not



Triplet loss

To train this network, we use **triplet loss**, show similar and dissimilar pairs of objects:

$$L = \max(\text{dist}(a, p) - \text{dist}(a, n) + \varepsilon)$$

Training and applying

Training. The network is shown triplets. For each triplet, we evaluate triplet loss and then use gradient descent.

Applying. We store a single object per class (centroid) and compare each new object with each centroid. To increase the number of classes, we simple add corresponding centroids.

Lecture plan

- Label irregularity
- Semi-supervised learning
- Active learning
- Data sampling
- One-shot learning
- Anomaly detection
- Outliers

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.

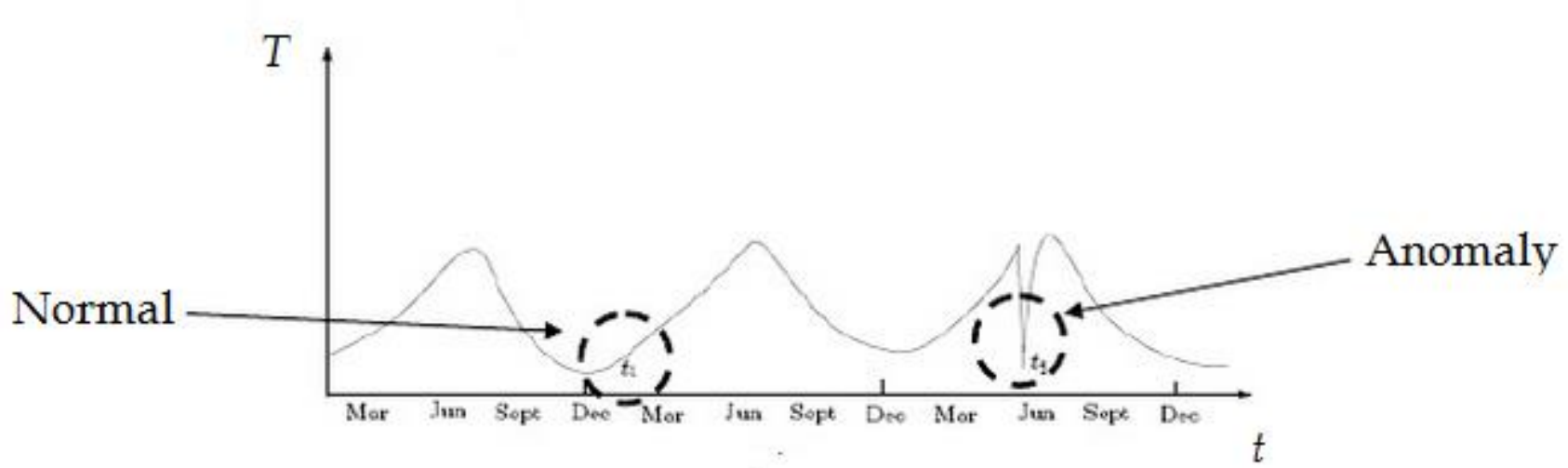
Anomaly (outlier, exception, surprise) **detection** is the problem of one-class classification.

Anomaly detection taxonomy

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

Contextual anomaly detection

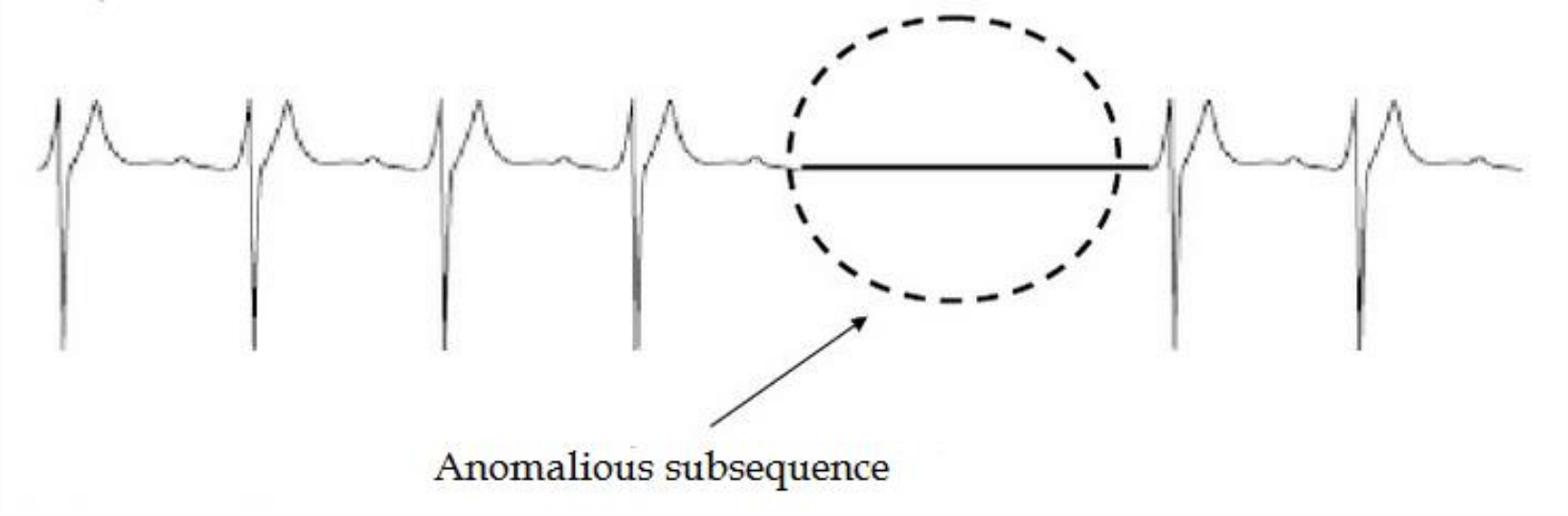
Assumption: all normal instances within a context will be similar (in terms of behavioral attributes), while the anomalies will be different.



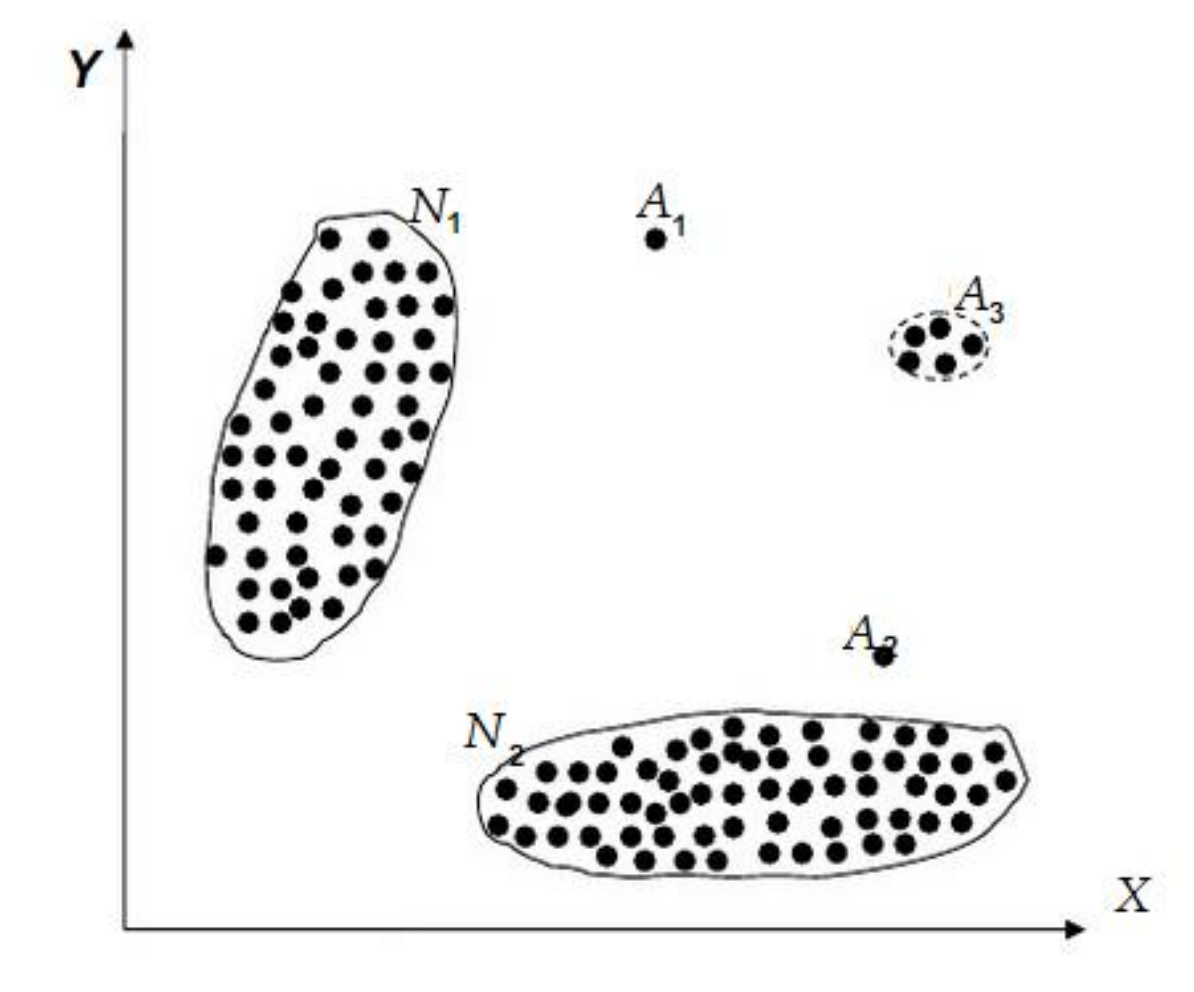
Collective anomaly detection

Data instances are related:

- sequential data
- spatial data
- graph data



Point anomaly detection



Point 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

Lecture plan

- Label irregularity
- Semi-supervised learning
- Active learning
- Data sampling
- One-shot learning
- Anomaly detection
- Outliers

Cleaning noisy objects

Anomalies for regression are used to called outliers. In general, we can filter them out.

However, there are methods for building robust models that can handle outliers by themselves.

Weighting objects

$\varepsilon_i = \text{LOO}(x_i) = L(a(x_i; T^\ell \backslash \{x_i\}))$.

This can be used as a weight in general approach:

$$Q(a, T^\ell) = \sum_{i=1}^{\ell} K(\varepsilon_i)(f(x_i, \theta) - y_i)^2 \to \min_\theta .$$

LOWESS method (LOcally WEighted Scatter plot Smoothing) for nonparametric regression:

use $\varepsilon_i = |a - y_i|$ as a loss function;

use kernel function $K(\varepsilon_i) = K_Q\left(\frac{\varepsilon_i}{6\text{med}\varepsilon_i}\right)$.

Robust regression

Regression model:

$$a(x) = f(x, \theta).$$

Meshalkin function:

$$L(x) = 1 - \exp(x^2).$$