Lecture 2

Non-parametric methods

Intellectual systems
(Machine Learning)

Andrey Filchenkov

13.09.2018

Lecture plan

- Similarity-based classification and regression
- One nearest neighbor
- *k*NN and hyperparameter tuning
- Generalized distance-based classifiers
- Nonparametric regression
- Prototype selection
- Performance measures

- The presentation is prepared with materials of the K.V. Vorontsov's course "Machine Leaning".
- Slides are available online: **goo.gl/BspjhF**

Lecture plan

- Similarity-based classification and regression
- One nearest neighbor
- kNN and hyperparameter tuning
- Generalized distance-based classifiers
- Nonparametric regression
- Prototype selection
- Performance measures

Problem formulation

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

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

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

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

Task: return an algorithm $a: X \to Y$.

What is this task?

Classification problem formulation

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

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

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

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

Task: return an algorithm $a: X \to Y$.

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 / may have similar target function values.

Main idea for classification: 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

Lecture plan

- Similarity-based classification and regression
- One nearest neighbor
- kNN and hyperparameter tuning
- Generalized distance-based classifiers
- Nonparametric regression
- Prototype selection
- Performance measures

Formalization of “similarity”

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

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

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.

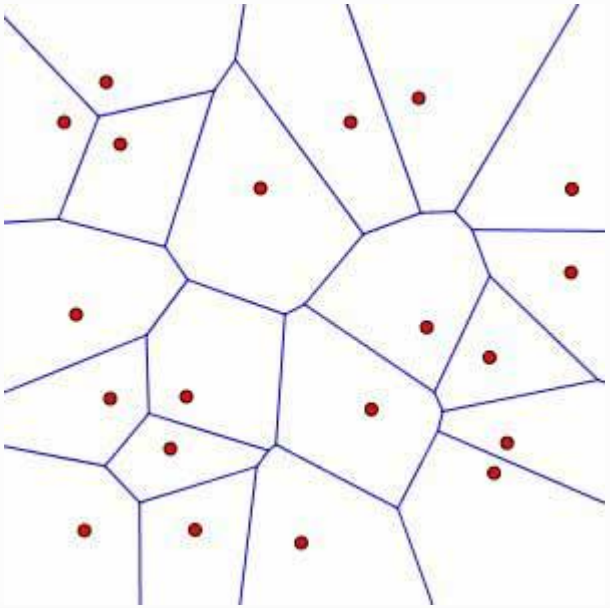
Nearest neighbor method (1NN)

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

Classifier:

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

Voronoi diagram:



1NN discussion

Advantages:

- simplicity;
- lucidity;
- interpretability.

Disadvantage:

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

1NN for regression

Could you suggest 1NN regressor?

1NN for regression

Could you suggest 1NN regressor?

Assign a value, which has the most similar object

Is this somehow useful?

1NN for regression

Could you suggest 1NN regressor?

Assign a value, which has the most similar object

Is this somehow useful?

It is not so bad as it could be, but none uses it

Lecture plan

- Similarity-based classification and regression
- One nearest neighbor
- kNN and hyperparameter tuning
- Generalized distance-based classifiers
- Nonparametric regression
- Prototype selection
- Performance measures

kNN

Choose a distance ρ.

Sort objects:

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

Algorithm kNN:

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

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

Hyperparameters and parameters

We can work with any value of k.

If an algorithm can work with any value of k but can not distinguish them in its sole direction, k is a **hyperparameter** for this algorithm.

If it can choose its value, then k is a **parameter** of this algorithm.

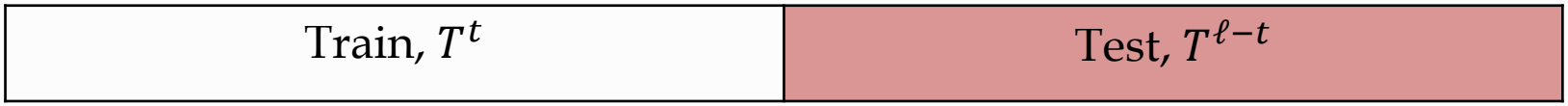
How to turn k to a parameter?

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:

$$\mathrm{HO}\left(\mu, T^{t}, T^{\ell-t}\right) = Q\left(\mu(T^{t}), T^{\ell-t}\right) \to \min$$

Complete cross-validation

Choose value of t.

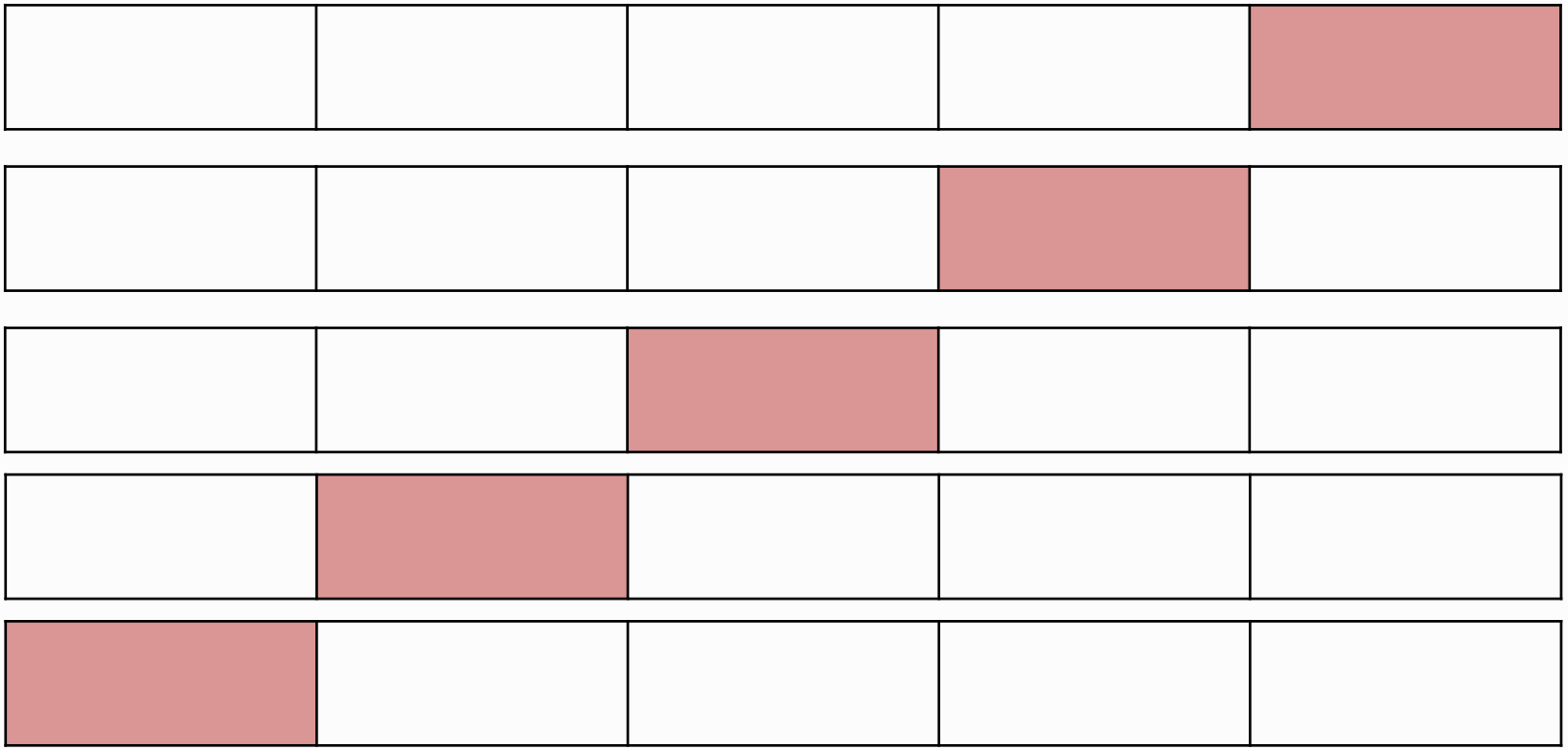
Split the sample with all the possible ways on T^t and $T^{\ell-t}$.

Solve the optimization problem:

$$\mathrm{CVV}_t = \frac{1}{C_\ell^{\ell-t}} \sum_{T^\ell = T^{\ell-t} \cup T^t} Q\left(\mu(T^t), T^{\ell-t}\right) \to \min$$

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

Solve the optimization problem:

$$\mathrm{CV}_k = \frac{1}{k}\sum_{i=1}^{k} Q\left(\mu\left(T^\ell \backslash F_i\right), F_i\right) \to \min.$$

$t \times k$-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^ℓ t times randomly:

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

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

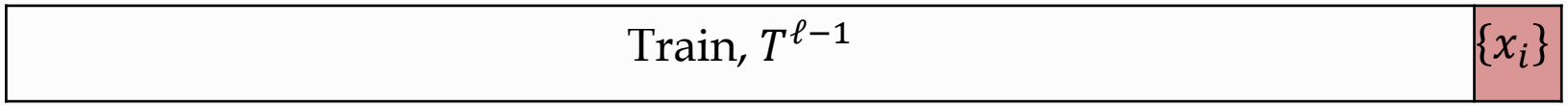
Solve the optimization problem:

$$\mathrm{CV}_{t\times k} = \frac{1}{tk}\sum_{j=1}^{t}\sum_{i=1}^{k} Q\left(\mu\left(T^\ell \backslash F_{(i,j)}\right), F_{(i,j)}\right) \to \min.$$

Leave one out

Leave-one-out cross-validation, LOO

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



Solve the optimization problem:

$$\text{LOO} = \frac{1}{\ell}\sum_{i=1}^{\ell} Q\left(\mu\left(T^{\ell}\backslash p_i\right), p_i\right) \to \min.$$

where $p_i = (x_i, y_i)$.

Choosing hyperparameters

You cannot choose hyperparameters on a test set.
Why?

Choosing hyperparameters

You cannot choose hyperparameters on a test set, because it will lead to overfitting. You will choose the best hyperparameters that will improve performance on this dataset, but not in general.

How to fix it?

Choosing hyperparameters

You cannot choose hyperparameters on a test set, because it will lead to overfitting. You will choose the best hyperparameters that will improve performance on this dataset, but not in general.

You should split your train set. You will learn parameters on one part, and then tune hyperparameters on another part. This is why datasets are typically split to train, validation and test subsets.

Lecture plan

- Similarity-based classification and regression
- One nearest neighbor
- kNN and hyperparameter tuning
- Generalized distance-based classifiers
- Nonparametric regression
- Prototype selection
- Performance measures

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

Optimization of k

Is equal to the problem of LOO quality functional minimization:

$$\mathrm{LOO}\left(k, T^{\ell}\right) = \sum_{i=1}^{\ell} \left[a\left(x_i; T^{\ell} \backslash \{(x_i, y_i)\}, k\right) \neq y_i\right] \rightarrow$$

$$\rightarrow \min_k$$

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 ith 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.

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

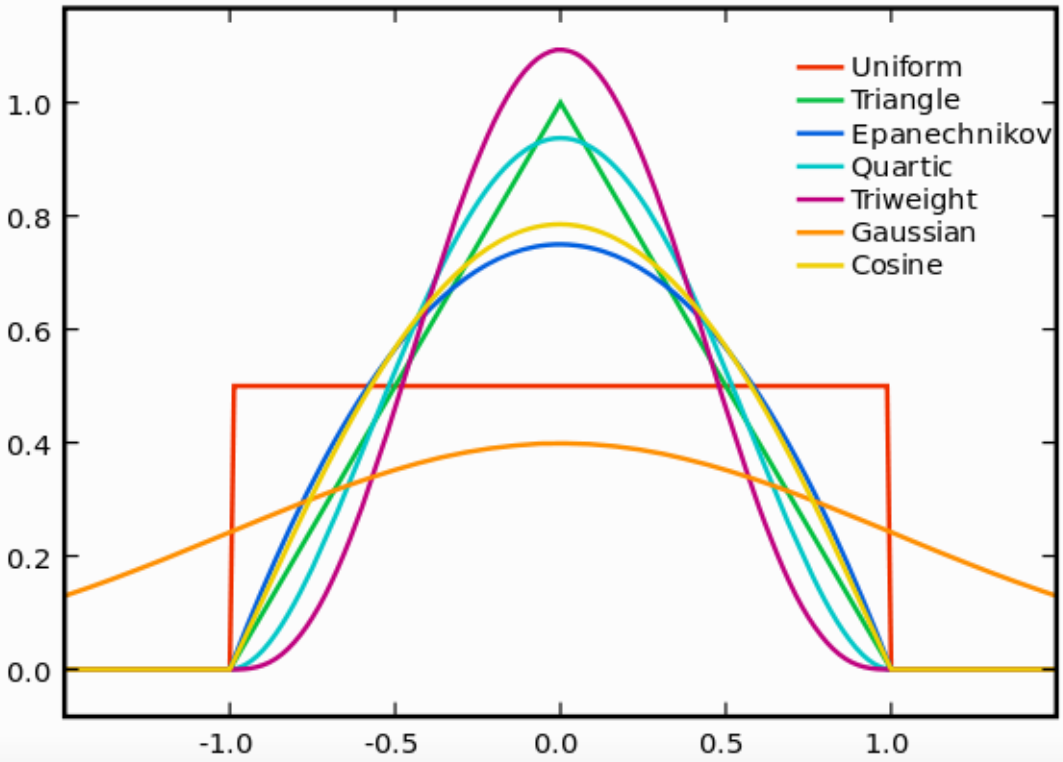
What can be chosen as w?

$w(i,u)$:

- 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:

$$a\left(u; T^{\ell}; h; K\right) =$$

$$= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u,i) = y] K\left(\frac{\rho\left(u, x_{(u,i)}\right)}{h}\right),$$

With variable window width:

$$a\left(u; T^{\ell}; k; K\right) =$$

$$= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u,i) = y] K\left(\frac{\rho\left(u, x_{(u,i)}\right)}{\rho\left(u, x_{(u,k+1)}\right)}\right).$$

Parzen-Rosenblatt window

With fixed window width:

$$a\left(u; T^{\ell}; h; K\right) =$$

$$= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u,i) = y] K\left(\frac{\rho\left(u, x_{(u,i)}\right)}{h}\right),$$

With variable window width:

$$a\left(u; T^{\ell}; k; K\right) =$$

$$= \operatorname{argmax}_{y \in Y} \sum_{i=1}^{\ell} [y(u,i) = y] K\left(\frac{\rho\left(u, x_{(u,i)}\right)}{\rho\left(u, x_{(u,k+1)}\right)}\right).$$

Distance selection (learning)

Distance can be learned.

Example (weighted Minkowski):

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

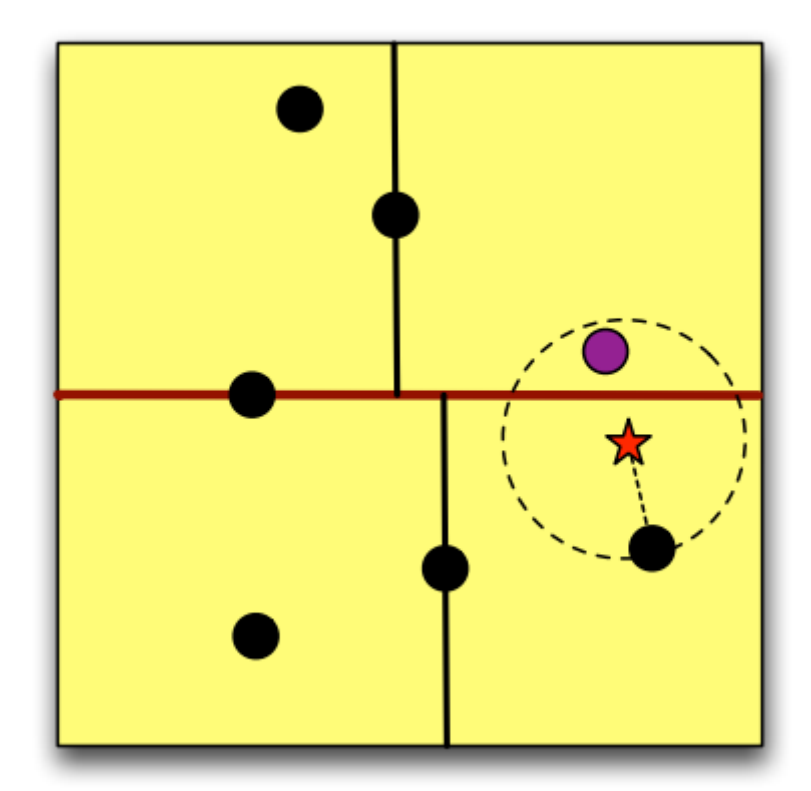
Now the problem is how to choose coefficients w_i.

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

- Similarity-based classification and regression
- One nearest neighbor
- kNN and hyperparameter tuning
- Generalized distance-based classifiers
- Nonparametric regression
- Prototype selection
- Performance measures

Problem formalization

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

$y: X \to Y$ is an unknown dependency, $Y \in \mathbb{R}$

$X^\ell = \{x_1, \ldots, x_\ell\}$ is training sample,

$T^\ell = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ set of instances.

Problem: find $a: X \to Y$.

`

$a(x) = f(x, \theta)$ is dependency model, $\theta \in \mathbb{R}^t$.

Main idea

Basic idea: let think that $\theta(x) = \theta$ nearby $x \in X$:

$$Q\left(\theta, T^{\ell}\right) = \sum_{i=1}^{\ell} w_i(x)(\theta - y_i)^2 \to \min_{\theta \in \mathbb{R}}.$$

Main idea: use kernel smoothing:

$$w_i(x) = K\left(\frac{\rho(x_i, x)}{h}\right),$$

where h is window width.

Kernel smoothing

Nadaraya-Watson kernel smoothing:

$$a_h\left(x, T^\ell\right) = \frac{\sum_{i=1}^{\ell} y_i w_i(x)}{\sum_{i=1}^{\ell} w_i(x)} = \frac{\sum_{i=1}^{\ell} y_i K\left(\frac{\rho(x_i, x)}{h}\right)}{\sum_{i=1}^{\ell} K\left(\frac{\rho(x_i, x)}{h}\right)}.$$

Basic theorem

Theorem. If

1) sample T^ℓ is simple, distributed with $p(x, y)$;

2) $\int_0^\infty K(r)dr < \infty$, $\lim_{r\to\infty} rK(r) = 0$;

3) $\mathrm{E}\left(y^2 \middle| x\right) < \infty \; \forall x \in X$;

4) $\lim_{\ell\to\infty} h_\ell = 0$, $\lim_{\ell\to\infty} \ell h_\ell = \infty$,

then $a_h\left(x, T^\ell\right) \to^P \mathrm{E}(y|x)$ in any $x \in X$,

when $E(y|x), p(x), D(y|x)$ are continuing, $p(x) > 0$.

Method discussion

- kernel function has impact on smoothness;
- kernel function has small impact on approximation quality;
- h impacts on approximation quality;
- k can be tuned;
- sensitive to noise.

Lecture plan

- Similarity-based classification and regression
- One nearest neighbor
- kNN and hyperparameter tuning
- Generalized distance-based classifiers
- Nonparametric regression
- Prototype selection
- Performance measures

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

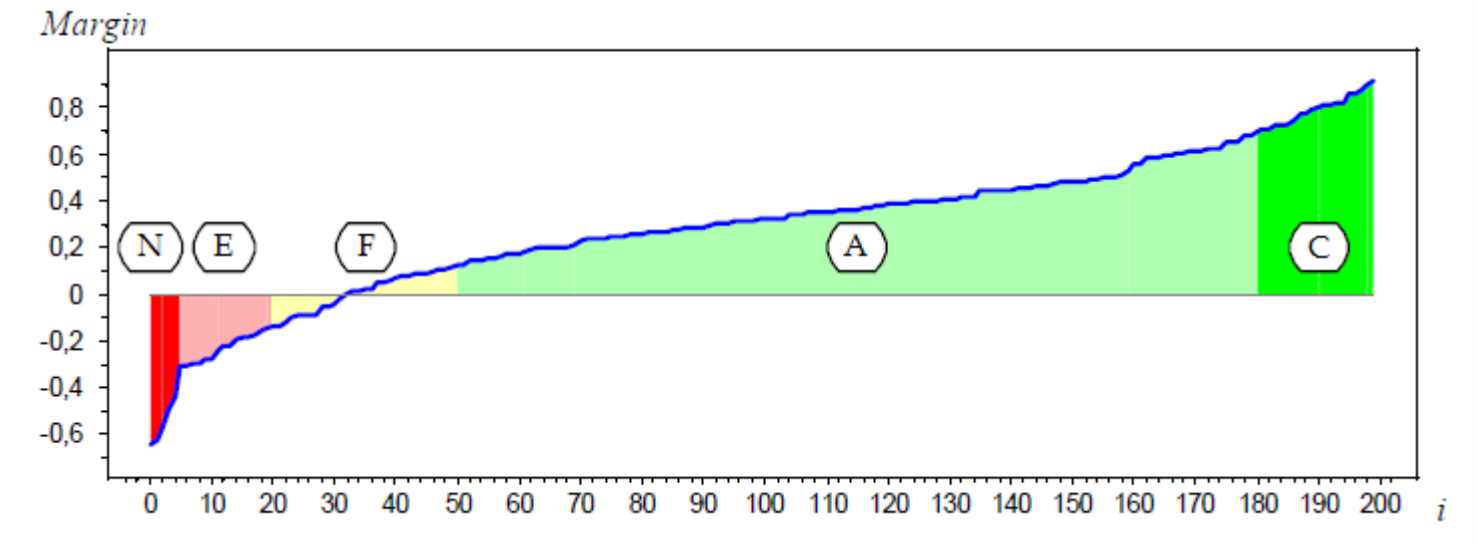
C — **core** (base for classification)

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

- Similarity-based classification and regression
- One nearest neighbor
- kNN and hyperparameter tuning
- Generalized distance-based classifiers
- Nonparametric regression
- Prototype selection
- Performance measures

Contingency table

	Positive	Negative
Classified as positive	**TP** = True Positive	**FP** = False Positive
Classified as negative	**FN** = False Negative	**TN** = True Negative

FN in math. stat. — I type error

FP in math. stat. — II type error

P = **TP** + **FN** — number of positive examples

N = **FP** + **TN** — number of negative examples

Some definitions

Sensitivity or **Recall**:

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{P}}$$

Specificity:

$$\text{SPC} = \frac{\text{TN}}{\text{N}}$$

Precision:

$$\text{Precision} = \text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Accuracy:

$$\text{Accuracy} = \text{ACC} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

F-measure

We will not lose much in accuracy performing badly on small classes.

F_β-measure:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

F_1-measure:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

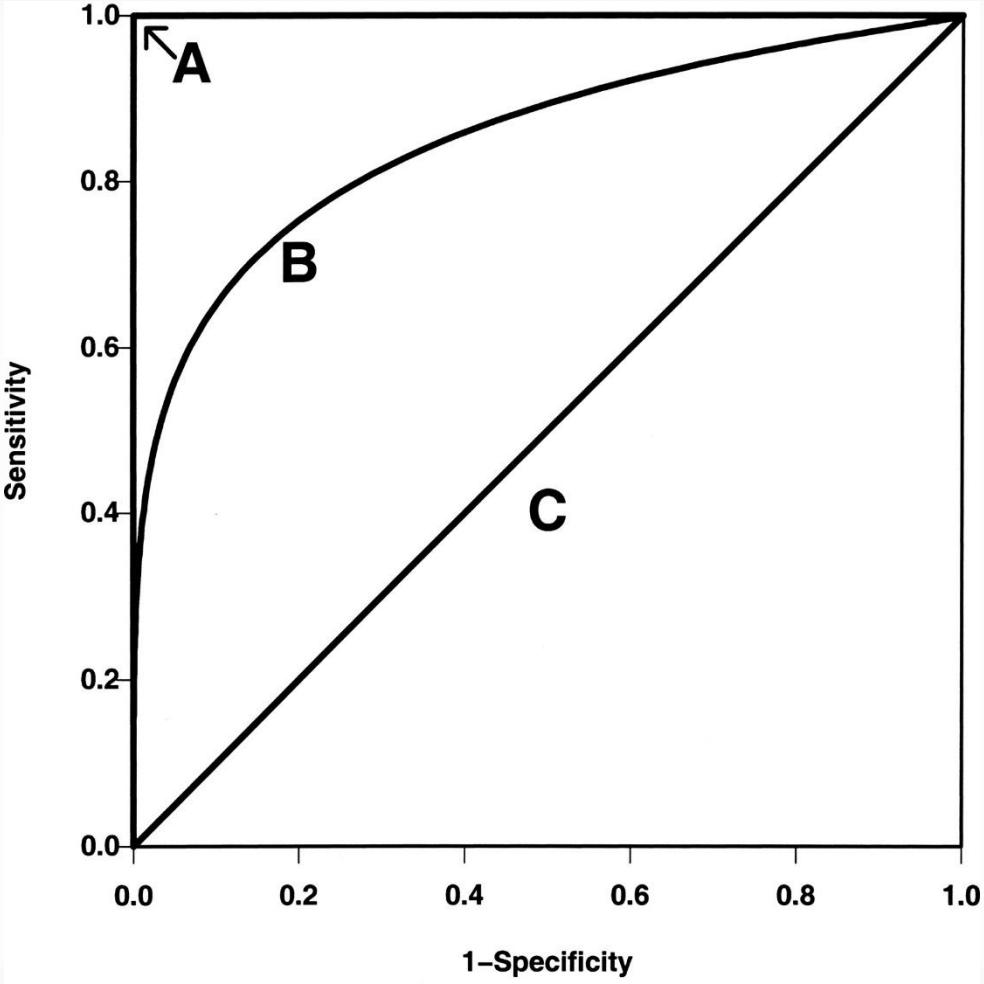
54

ROC-curve

A is the best algorithm

B is a typical algorithm

C is the worst algorithm



AUC

Area under the curve (AUC) is area under the ROC-curve.

Connected with Mann-Whitney U.

Can be expressed with Gini-index.

Out of date measure.

56

Multiclass case

- One vs one classification
- One vs all (one vs rest) classification
- Hierarchical classification
- Confusion matrix

57

Usual regression errors

- Root mean squared error (RMSE)
- Mean absolute error (MAE)
- Mean squared error (MSE)
- Symmetric mean absolute percentage error (SMAPE):

$$\text{SMAPE} = \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{2\cdot|a(x_i)-y_i|}{|a(x_i)|+|y_i|}.$$

58