Lecture 1

Basic models: nearest neighbors and trees

Machine Learning

Ivan Smetannikov

02.07.2019

Lecture plan

- Nearest neighbor classifier
- Generalized distance-based classifiers
- Non-parametric regression
- Decision trees
- Splitting criteria
- Regression trees

- The presentation is partly prepared with materials of the K.V. Vorontsov's course "Machine Leaning".

Lecture plan

- Nearest neighbor classifier
- Generalized distance-based classifiers
- Non-parametric regression
- Decision trees
- Splitting criteria
- Regression trees

Problem formulation

X is object set, Y is 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 problem formulation

X is object set, Y is 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.

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

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

Commonly used metrics

Minkowski distance:

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

$p = 2$ is the Euclidian distance;

$p = 1$ 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.

Entropic distance measure (in K^*)

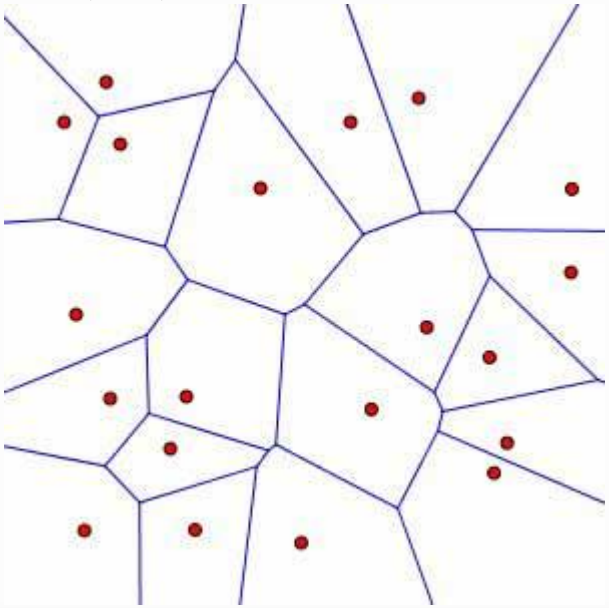
Nearest neighbor method (1NN)

$x_{(u,1)}$ is **nearest** (1 = number one in sorted distances list) **neighbor of** classified object u:

$$x_{(u,1)} = \operatorname{argmin}_{x \in X^\ell} \rho(u, x).$$

Classifier: $a\left(u, T^\ell\right) = y(x_{(u,1)}) = y(u, 1).$

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

- Nearest neighbor classifier
- Generalized distance-based classifiers
- Non-parametric regression
- Decision trees
- Splitting criteria
- Regression trees

How can it be improved?

Very simple model

- More complex model (more parameters)

How to choose “closeness”?

- Distance choosing
- Dimension reduction

Finding nearest object is too slow

- Usage of good structures for data storage
- Storage of only useful objects

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:

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

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

Optimization of k

How to find a proper k?

k in this case is called **hyperparameter.**

We will work with this problem during the next lecture.

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 weight 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.

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

What can be chosen as w?

$w(i,u)$:

- linearly decreasing functions;
- exponentially decreasing functions;

Parzen-Rosenblatt window

With fixed window width:

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

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

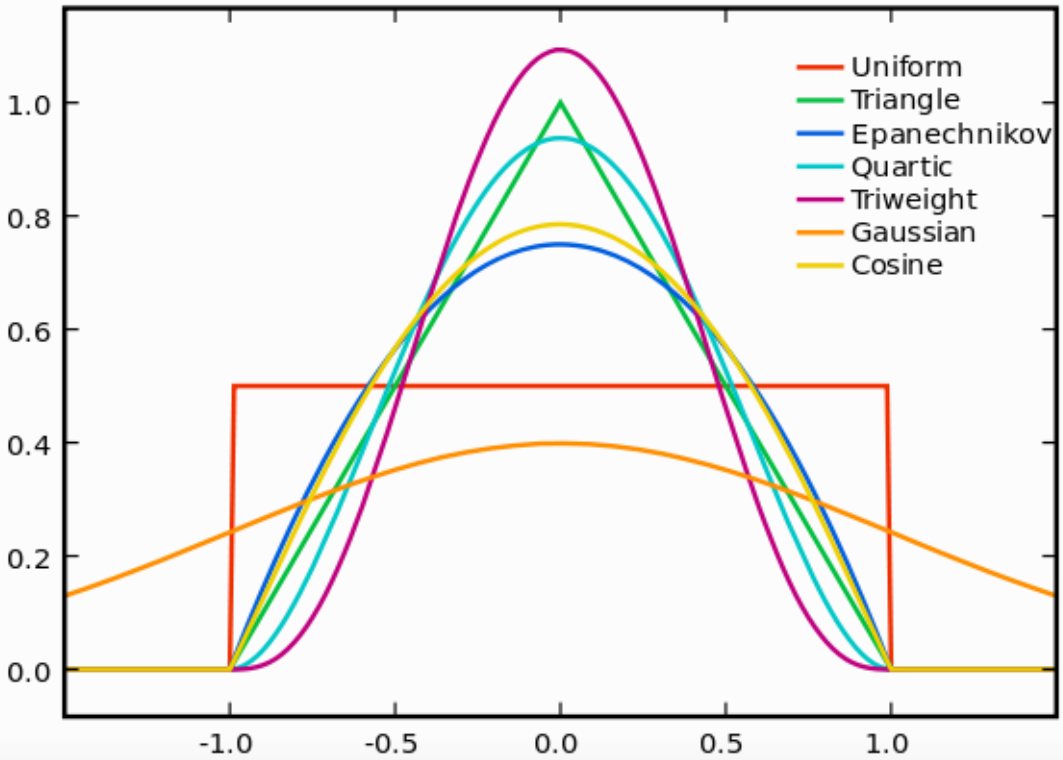
With variable window width:

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

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

Kernel function

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



Distance selection (learning)

Distance can be learned (chosen).

Distance can be parametrized. Example (weighted Minkowski):

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

Now the problem is how to choose coefficients ω_i.

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

How to choose nearest neighbor?

Brute force (no data structure)

- py: bruteforce
- weka: LinearNNSearch

K-D Tree

- py: kd_tree
- weka: KDTree

Ball Tree

- py: ball_tree
- weka: BallTree

Cover Tree

- weka: CoverTree

Nearest centroid classifier

Just create **centroids** for each class.

For a new object, choose the label of the nearest centroid.

Lecture plan

- Nearest neighbor classifier
- Generalized distance-based classifiers
- Non-parametric regression
- Decision trees
- Splitting criteria
- Regression trees

Main idea

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

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

Main idea: use kernel smoothing:

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

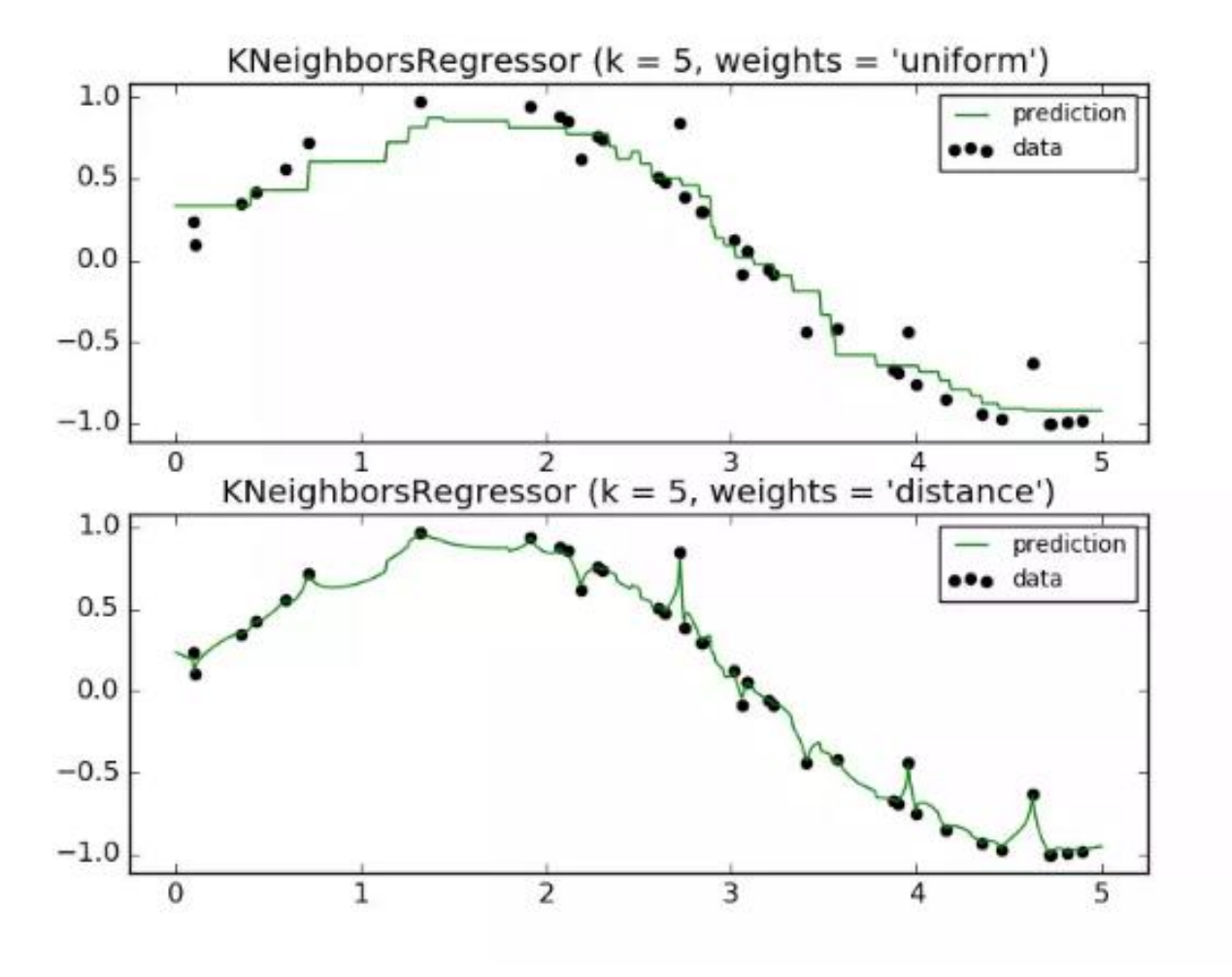
where h is window width.

Kernel smoothing

Nadaraya-Watson kernel smoothing:

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

Nearest neighbor regression



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

- Nearest neighbor classifier
- Generalized distance-based classifiers
- Non-parametric regression
- Decision trees
- Splitting criteria
- Regression trees

Decision trees

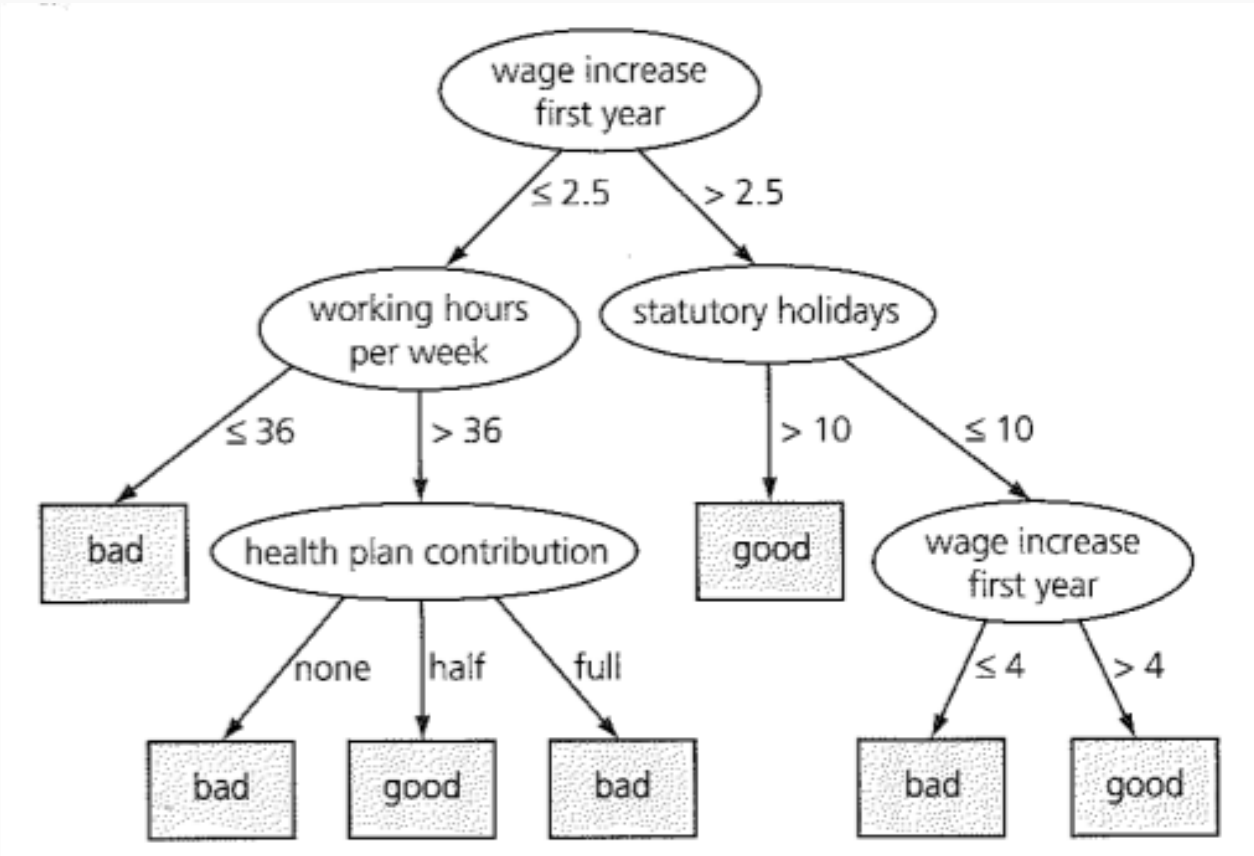
Decision tree is a classifier and even regression algorithm.

Nodes contain splitting rules (questions).

Each edge is a possible answer to its node question.

Leafs contain decisions (a class for classification problem and a number for regression problem).

Decision tree example



General scheme

With splitting rules space $\mathcal{B}$ and split quality functional Φ.

1. Sent S to the root.

2. On each step process sample S.

2.1. If S contains objects from a single class c, create leaf of class c and stop.

2.2. Else choose splitting rule $\beta \in \mathcal{B}$ which is the most informative with respect to Φ and split sample to $S_1, \ldots, S_k$.

2.3. If stop-criterion, is true, then return the most popular class in S, otherwise create n children with samples S_i.

3. Prune the tree.

Stop-criteria

The most popular stop-criteria:

- one of classes is empty after splitting
- $\Phi(S)$ is lower than a certain threshold
- $|S|$ is lower than a certain threshold
- tree height is higher than a certain threshold

Pruning

Premises: just first node impact on performance; decision trees tend to be overfitted.

Main idea: to cut lower branches.

Pruning is processing of created trees, when branches are delated consequently with a certain criterion (reduction number of errors, for example).

Pruning algorithm scheme

Split sample to train and test in proportion 2:1.

For each tree node apply operation, which is the best in terms of number of errors:

1) Don not change anything;
2) Replace node with its child (for each child);
3) Replace node with a leaf (for each class).

Trees discussion

Advantages:

- easily understandable and interpretable;
- learning is quick;
- can work with different data types.

Disadvantage:

- sensitive to noise;
- easily get overfitted.

Lecture plan

- Nearest neighbor classifier
- Generalized distance-based classifiers
- Non-parametric regression
- Decision trees
- Splitting criteria
- Regression trees

Selecting splitting rules family

Can be any family of classifiers.

- In most of cases it's single feature based rules like:

$$f_i(x) > m_i;$$
$$f_i(x) = d_i.$$

- Sometimes it can be a combination.

Selection of feature-based rules

There is $\ell - 1$ options to split sample.

- Check each and pick the most informative.
- Join diapasons of values .
- Skip small diapasons.

This is how you can synthesize a rule for each feature.

Sample splitting

- If sample is being split each time into 2 ($k = 2$), then $\mathcal{B}$ is family of binary rules, tree is binary.
- If a feature is categorical, several edges can be added.
- If a feature is real, discretization / banalization is applied.

On each step number of edges can differ, but usually k is fixed.

Selecting split quality criterion

Split quality Φ sometimes can be represented as:

$$\Phi(S) = \phi(S) - \sum_{i=1}^{k} \frac{|S_i|}{|S|} \phi(S_i).$$

The most popular:

- $\phi_h(S)$ is an entropy, $\Phi_h(S)$ is IGain;
- $\phi_g(S) = 1 - \sum_{i=1}^{m} p_i^2$, where p_i is probability (frequency) of ith class in sample S is **Gini index**. $\Phi_h(S)$ is GiniGain.

Many other are used

$$\text{GainRatio} = \text{IGain}(S) \,/\, \text{Enthropy}(S).$$

Split quality criterion usually does not matter.

Examples

ID3 (Quinlan, 1986):

IGain; only $\Phi(S) < 0$; no pruning.

C4.5 (Quinlan, 1993):

GainRatio; pruning.

CART (Breinman, 1984):

binary; GiniGain; pruning. Can solve regression (values in leafs).

Lecture plan

- Nearest neighbor classifier
- Generalized distance-based classifiers
- Non-parametric regression
- Decision trees
- Splitting criteria
- Regression trees

Regression trees