Lecture 10

Logic classifiers and decision trees

Machine Learning

Andrey Filchenkov

10.04.2018

Lecture plan

- Logical rules
- Convolution
- Rules induction
- Decision trees
- Random Forest

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

Lecture plan

- Logical rules
- Convolution
- Rules induction
- Decision trees
- Random Forest

Concepts and rules

Concept is a predicate on an object set X:

$$\varphi: X \to \{0,1\}.$$

A concept **covers** an object x, if $\varphi(x) = 1$.

Rule is a logical predicate, which covers many objects from one class and just a few or none objects from other classes and can be simply interpreted.

Example (from the Russian language): *If a word is an adverb and it ends with a hissing sound ("ж", "ч" or "ш"), you should end it with "ь".*

Examples: "вска**чь**", "насте**жь**".

Exceptions: "уж", "замуж", "невтерпеж".

Interpretable concepts

Origins in knowledge discovery in databases.

A concept φ **can be interpreted** if it

1) is formulated in natural language;
2) depends on a small number of feature (1–7).

Informative concepts

A concept φ is **informative** for a class c, if

$$p(\varphi) = |\{x_i | \varphi(x_i) = 1, y_i = c\}| \to \max;$$
$$n(\varphi) = |\{x_i | \varphi(x_i) = 1, y_i \neq c\}| \to \min.$$

Can be reformulated in a probabilistic sense.

$p(\varphi)$ is True Positive;

$n(\varphi)$ is False Positive;

$p(\varphi) + n(\varphi)$ is coverage.

Lecture plan

- Logical rules
- Convolution
- Rules induction
- Decision trees
- Random Forest

Convolution choice problem

It is not obvious, how to convolute these two parameters:

- Precision:

$$\frac{p}{p+n} \to \max;$$

- Accuracy

$$p - n \to \max;$$

- Linear cost accuracy:

$$p - Cn \to \max;$$

- Relative accuracy:

$$\frac{p}{P} - \frac{n}{N} \to \max.$$

Convolution choice comparison

Compare with $P = N = 100$:

p	n	$p-n$	$p-5n$	$\frac{p}{P}-\frac{n}{N}$	$\frac{p}{n+1}$	$\frac{p}{n+p}$	I_c	IGain_c	$\sqrt{p}-\sqrt{n}$
50	0	**50**	50	0.25	50	1	22.65	23.70	7.07
100	50	**50**	−150	0	1.96	0.67	2.33	1.98	2.93
50	9	41	**5**	0.16	**5**	**0.85**	7.87	7.94	4.07
5	0	5	**5**	0.03	**5**	**1**	2.04	3.04	2.24
100	0	**100**	100	**0.5**	100	1	52.18	53.32	10.0
140	20	**120**	40	**0.5**	6.67	0.88	37.09	37.03	7.36

ε, δ-rule

$E_c(\varphi, T^\ell) = \frac{n_c(\varphi)}{p_c(\varphi)+n_c(\varphi)}$ is a share of falsely covered objects.

$D_c(\varphi, T^\ell) = \frac{p_c(\varphi)}{\ell}$ is a share of positive objects among the covered objects.

$\varphi(x)$ is **ε, δ-rule** (for class c), if $E_c(\varphi, T^\ell) \leq \varepsilon$ and $D_c(\varphi, T^\ell) \geq \delta$.

If $n_c(\varphi) = 0$, then the rule is **exact**.

Statistical rule

Assumption: the sample is simple.

Probability of pair (p, n) is described with hypergeometric distribution:

$$\mathcal{H}_{P,N}(p,n) = \frac{C_P^p C_N^n}{C_{P+N}^{p+n}}.$$

Convolution of predicate φ with sample T^ℓ:

$$I_c(\varphi, T^\ell) = -\ln \mathcal{H}_{P_c,N_c}\big(p_c(\varphi), n_c(\varphi)\big).$$

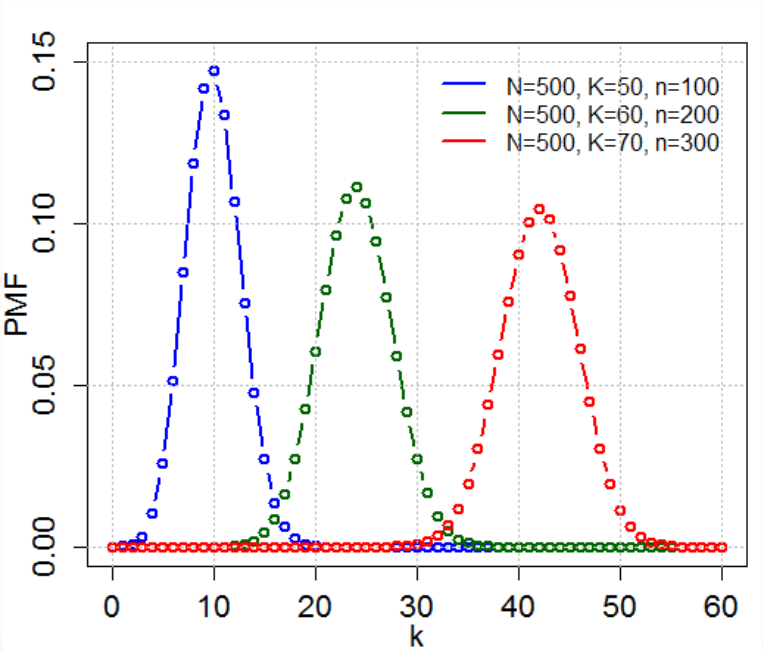
$\varphi(x)$ is **statistical rule** (for class c), if

$$I_c\left(\varphi, T^\ell\right) \geq \alpha$$

with α being high enough (**Fisher exact test**).

Hypergeometric distribution

Hypergeometric distribution is a discrete probability distribution that describes the probability of getting k target objects in n draws (without replacement) from a finite population of size N containing exactly K target objects.



Entropy-based rule

Entropy of two outcomes:

$$H(q_0, q_1) = -q_0 \log_2 q_0 - q_1 \log_2 q_1 .$$

Entropy of the sample:

$$\widehat{H}(P, N) = H\left(\frac{P}{P+N}, \frac{N}{P+N}\right).$$

$$\widehat{H}_\varphi(P, N, p, n) =$$

$$= \frac{p+n}{P+N}\widehat{H}(p, n) + \frac{P+N-p-n}{P+N}\widehat{H}(P-p, N-n).$$

$$\mathrm{IGain}_c\left(\varphi, T^\ell\right) = \widehat{H}(P, N) - \widehat{H}_\varphi(P, N, p, n).$$

$\varphi(x)$ is **entropy-based rule** (for class c), if $\mathrm{IGain}_c\left(\varphi, T^\ell\right) \geq G_0$ with a certain G_0 being high enough.

Good criterions (convolutions)

- IGain_c, I_c

  Theorem: $\lim\limits_{\ell\to\infty} \mathrm{IGain}_c\left(\varphi, T^\ell\right) = \frac{1}{\ell \ln 2} I_c\left(\varphi, T^\ell\right)$.

- Boosting criterion:

$$\sqrt{p} - \sqrt{n} \to \max.$$

- Normalized boosting criterion:

$$\sqrt{\frac{p}{P}} - \sqrt{\frac{n}{N}} \to \max.$$

Lecture plan

- Logical rules
- Convolution
- Rules induction
- Decision trees
- Random Forest

Rule definition and examples (1/2)

Rule is an interpretable highly informative single-class classifiers with refusals.

Examples

1. **Conjunction of boundaries** (terms):

$$R(x) = \bigwedge_{j \in J} [a_j \leq f(x_j) \leq b_j].$$

Rule definition and examples (2/2)

2. **Syndrome** is at least d terms of J are true:

$$R(x) = \left[\sum_{j \in J} [a_j \le f(x_j) \le b_j] \ge d\right],$$

(when $d = |J|$, it is conjunction, when $d = 1$, it is disjunction).

3. **Half-plane**:

$$R(x) = \left[\sum_{j \in J} w_j f_j(x) \ge w_0\right].$$

4. **Ball** is threshold similarity function:

$$R(x) = [r(x, x_0) \le r_0].$$

Where to get concepts and how to choose rules

Concepts can be:

- created by a researcher;
- learnt;
- given from experts / literature.

Rules can be learnt with

- optimization problem solvers;
- heuristic methods;
- special machine learning algorithms.

Lecture plan

- Logical rules
- Convolution
- Rules induction
- Decision trees
- Random Forest

Decision trees

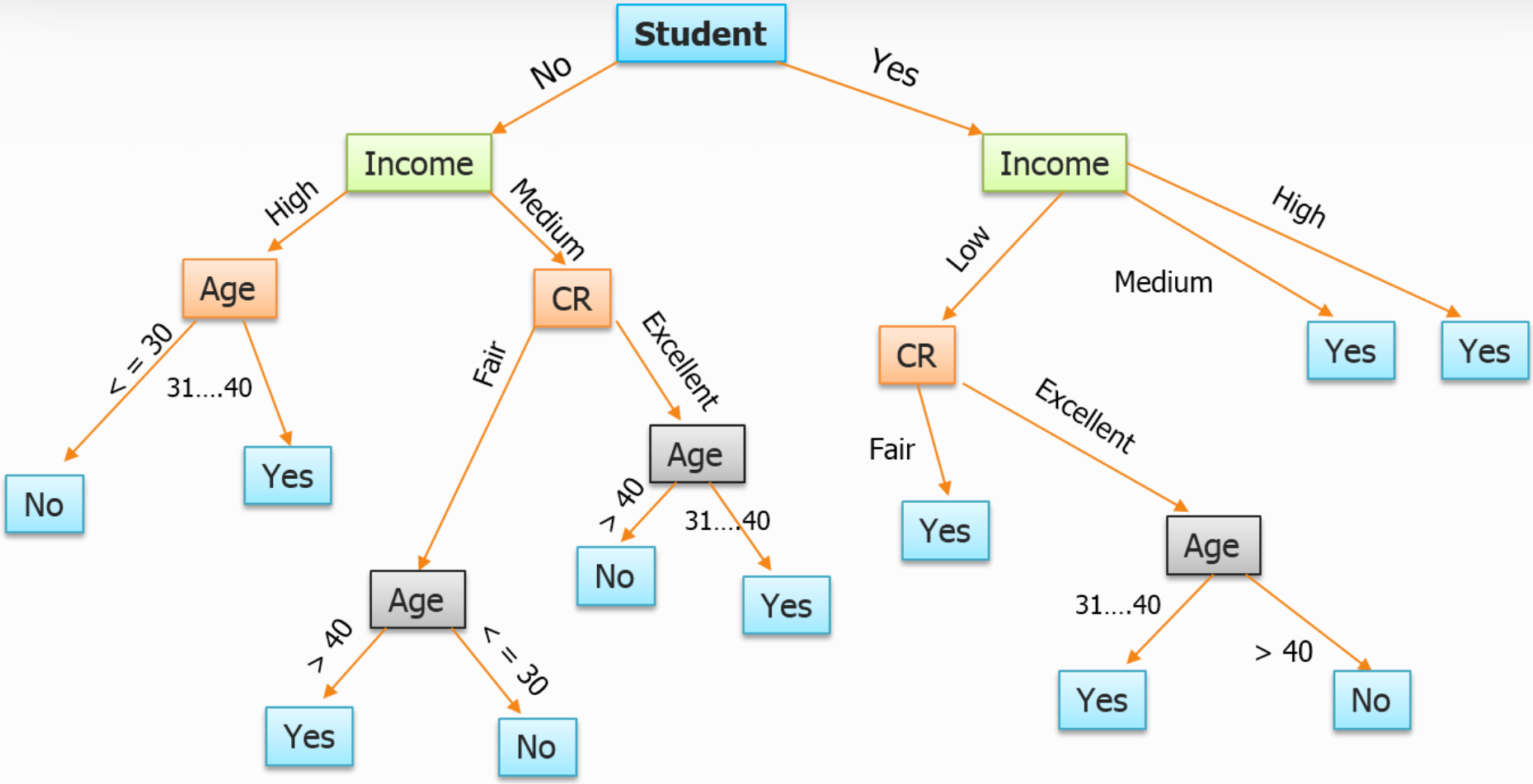
Decision tree is a classifier and regression algorithm.

Nodes contain splitting rules (questions).

Each edge is a possible answer to its node question.

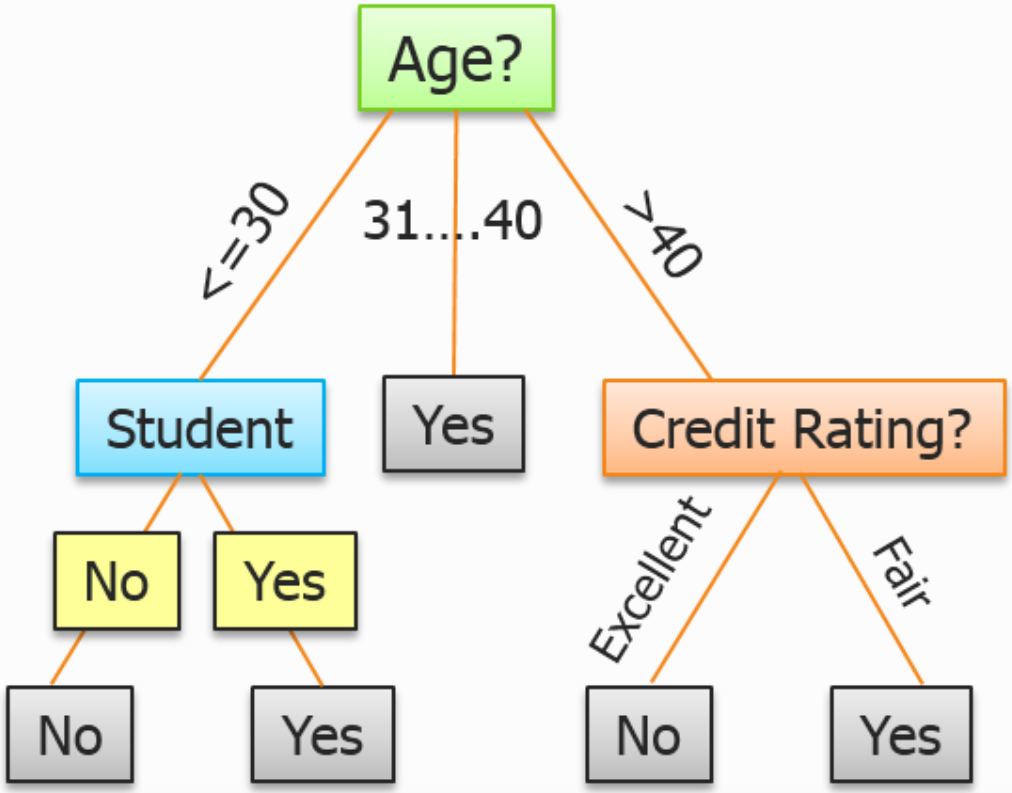
Leafs contain decisions (a class for classification problem and a number for regression problem).

Decision tree example (1/2)



Decision tree example (2/2)

The same classification can be achieved in a much more simpler way



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 a leaf of the class c and stop.

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

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

3. Prune the resulting tree.

Three main questions

How to choose splitting rules?

How to choose a splitting criterion?

How to prune the tree?

Selecting splitting rules family

Can be any family of classifiers.

- In most of the cases, it is 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 are $\ell - 1$ options to split the 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 a sample is split each time into 2 ($k = 2$), then $\mathcal{B}$ is a family of binary rules, tree is binary.
- If a feature is categorical, several edges can be added.
- If a feature is real, discretization / binarization is applied.

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

Selecting split quality criterion

Split quality Φ can be sometimes represented as:

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

The most popular:

- $\phi_h(S)$ is 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.

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 control 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).

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).

Trees discussion

Advantages:

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

Disadvantage:

- sensitive to noise;
- easily get overfitted.

Lecture plan

- Logical rules
- Convolution
- Rules induction
- Decision trees
- Random Forest

Random Forest

For sample $T^{\ell} = \{x_i, y_i\}_{i=1}^{\ell}$ with n features.

1. Choose a subsample size of ℓ with bootstrap.
2. Synthesize decision tree for that sample, for each vertex choose n' (usually $n' = \sqrt{n}$) random features.
3. No pruning is applied.

This can be done 100, 1000, … times.

Why does it work?

- It is voting
- Trees are easily get overfitted to very different subsamples
- With the growth of the sample, its performance converges

More details

What to combine?

- Simple votes of trees
- Probabilities (frequency of class in the resulting leaf)

How to improve?

- **Extremely randomized trees**: use random values for splitting (it is faster).

Ensemble method comparison