Machine learning

Lecture 3: Rule-based approach in Machine learning

Aleksei Platonov
DingTalkID: aplatonov

6 April 2020
HDU

- Logical rules in machine learning.
- Rules induction.
- Decision Trees: ID3, CART, ODT, RandomForest

The definition of logical rule

- Let's remember the machine learning task definition. There are training set $X^l = (x_i, y_i)_{i=1}^l$, we need to find an approximation of function $y_i = y(x_i)$.
- The goal of machine learning is to create an "intelligent" algorithm solving practical tasks.
- Human is the most intelligent agent in Nature that we know. How can we repeat Human intelligence?
- The first idea - using logical rules.

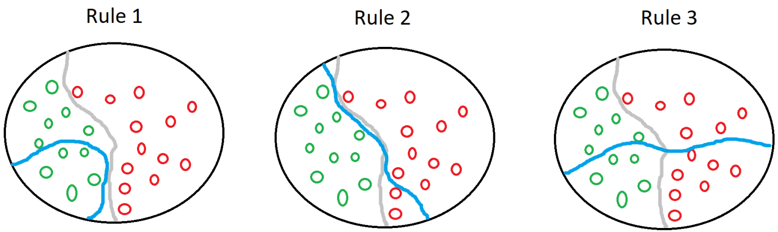
The definition of logical rule

Let's try to define logical rule mathematically:

Logical rule is a function $R : X \to 0, 1$. Logical rule must satisfy several requirements:

1. R must be interpretable:
   - R is a phrase on natural language.
   - R consists of set of logical predicates (no more that 7).
2. R must be informative regarding some class $c \in Y$:
   - $p(R) = |x_i : R(x_i) = 1 \ and \ y_i = c| \to max$
   - $n(R) = |x_i : R(x_i) = 1 \ and \ y_i \neq c| \to min$

Useful and useless rules

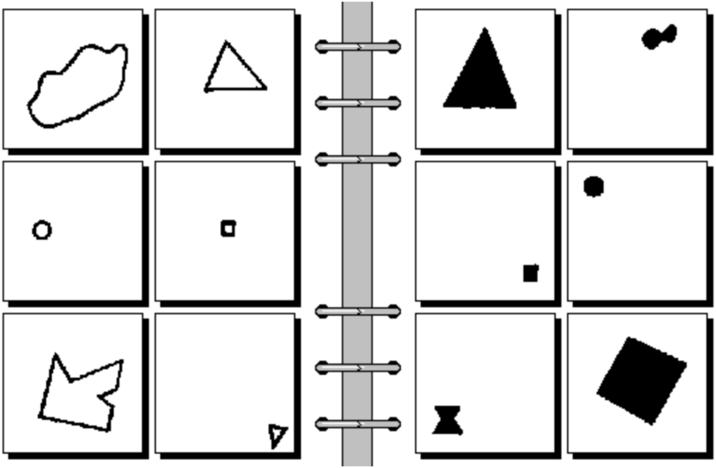


- Rule 1 is a consistent "pure" rule.
- Rule 2 is an informative useful rule.
- Rule 3 is a useless rule.

Examples of useful logical rules

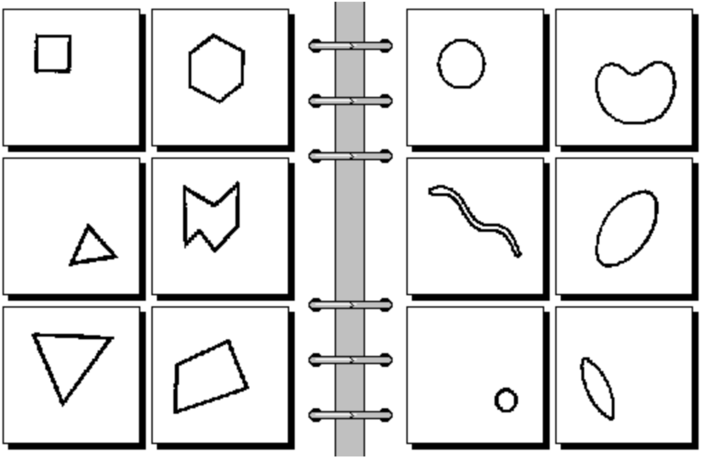
- If patient's age ≥ 60 and patient suffered a heart attack, then we don't do an operation, risk of death $= 60\%$.
- If a potential borrower wrote his/her home phone and his/her salary $\geq 2000\$$ and loan amount $\leq 5000\$$ then the loan could be approved, default risk $= 5\%$.
- If the book author is Arthur Clarke or Liu Cixin and text of the book contains words "spaceship", "planet" and "alien" then the book corresponds to the science fiction genre.
- If the email contains words "sale", "buy", "discount" and the email's sender sent more than one email last two days, then the email is spam with the level of confidence $= 80\%$.

Image recognition: Bongard Problems



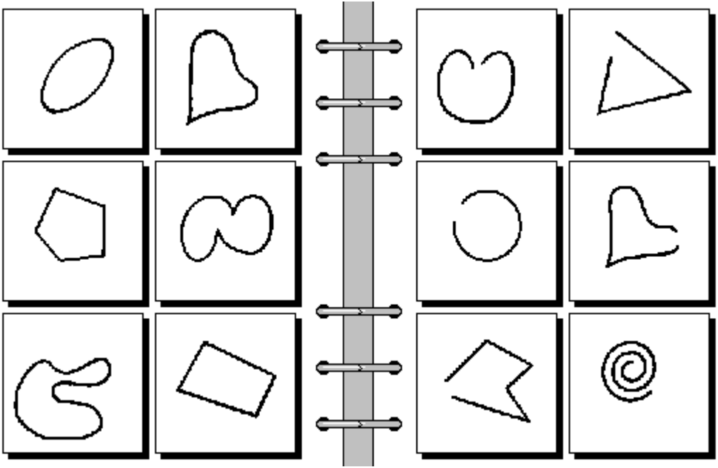
[0]http://www.foundalis.com/res/bps/bpidx.htm

Image recognition: Bongard Problems



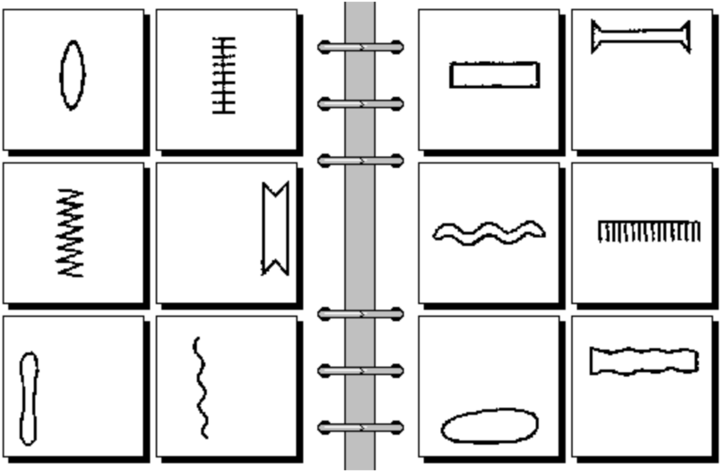
[0] http://www.foundalis.com/res/bps/bpidx.htm

Image recognition: Bongard Problems



[0] http://www.foundalis.com/res/bps/bpidx.htm

Image recognition: Bongard Problems



[0] http://www.foundalis.com/res/bps/bpidx.htm

The problems

- How can we find features $f_1(x) \dots f_n(x)$? It is always art of feature engineering.
- What's kind of logical rules $R_i(x)$ we need? We need a set of simple interpretable rules. How can we write down them in mathematical manner?
- How can we select useful logical rules? How can we unite the pair of requirements of informativeness?

$$p(R) = |x_i : R(x_i) = 1 \ and \ y_i = c| \to max$$

$$n(R) = |x_i : R(x_i) = 1 \ and \ y_i \neq c| \to min$$

- How can we find logical rules using set of feature functions $f_1(x) \dots f_n(x)$?
- How could logical rules $R_1(x) \dots R_n(x)$ be used in the classification and regression tasks?

Types of logical rules

1. The conjunction of threshold conditions:

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

For example: "[60 $\leq$ the patient's age $\leq$ 80] and [the number of operations in the past $\leq$ 5]"

Features could have two bounds of one. If feature has just one bound then another bound will be equal to $-\inf$ or $+\inf$.

Where (a_j, b_j) are threshold values, $f_j(x)$ - feature function, J - the set of thresholds for features.

Types of logical rules

2. Syndrome - the linear boolean function:

$$R(x) = \left[\sum_{j \in J} [a_j \leq f_j(x) \leq b_j] \geq d \right]$$

For example: "the patient's state matches at least three enumerated conditions: [has a cough, has a runny nose, body temperature ≥ 38, has a pain in back, has a headache]"

Features with boolean values also could have threshold values: $f_i(x) < 1$ means $f_i(x) = \mathit{false}$ or $f_i(x) > 0$ means $f_i(x) = \mathit{true}$.

Types of logical rules

3. Hyperplane threshold function [1]:

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

4. The ball condition [2].

$$R(x) = [\rho(x, x_0) \leq w_0]$$

Remind that: [*condition*] means that if *condition is true* then [*condition*] $= 1$ otherwise [*condition*] $= 0$.

---

[1] It corresponds to linear binary classifier. We will study them on the next lecture

[2] x_0 - the prototype, It corresponds to kNN methods

Information criteria

How can we unite different information criteria?

- $p_c(R) = |x_i : R(x_i) = 1 \; and \; y_i = c| \to max$
- $n_c(R) = |x_i : R(x_i) = 1 \; and \; y_i \neq c| \to min$

We need to create new criteria $I(p_c, n_c) \to max$. We could quickly generate a lot of useful functions:

- Precision: $\frac{p_c}{p_c + n_c} \to max$
- Accuracy: $p_c - n_c \to max$
- Linear cost accuracy: $p_c - Cn_c \to max$
- Relative accuracy: $\frac{p_c}{P_c} - \frac{n_c}{N_c} \to max$

Where P_c - the number of examples in training set of target class c, N_c - the number of examples in training set of other classes.

Information criteria

$P = 200, \; N = 100$

p	n	$\frac{p}{p+n}$	$p - n$	$p - 5n$	$\frac{p}{P} - \frac{n}{N}$
50	0	1.00	50	50	0.25
100	50	0.67	50	-150	0
50	9	0.85	41	5	0.16
5	0	1.0	5	5	0.03
100	0	1.0	100	100	0.5
140	20	0.88	120	40	0.5

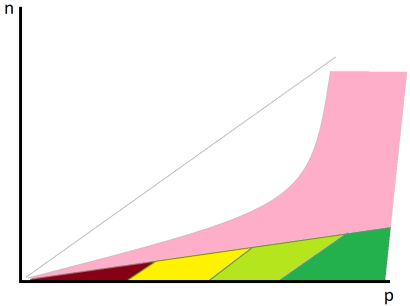
Information criteria

But that criteria don't consider entropy in the training set. There are other criteria:

- Information gain:
  $IGain(p, n) = h(\frac{p}{l}) - \frac{p+n}{l} h(\frac{p}{p+n}) - \frac{l-p-n}{l} h(\frac{P-p}{l-p-n}) \to max$
  Where $h(q) = -q \cdot log_2(q) - (1-q) \cdot log_2(1-q)$, l - the size of the training set.
- Fisher's test: $IStat(p, n) = -log_2(\frac{C_P^p \cdot C_N^n}{C_{P+N}^{p+n}}) \to max$

(p,n) - plane

(n,p)-plane clarifies what kind of rules we need...



We could find in on this plane:

- Less informative rules (red zone).
- Informative rules (yellow-green zone).
- Logical rules (right green zone).
- Statistical patterns (pink zone).

Rules induction: pseudocode

There exists a lot of rules generation methods:

- Genetic programming.
- Branch and bound method.
- Stochastic local search.

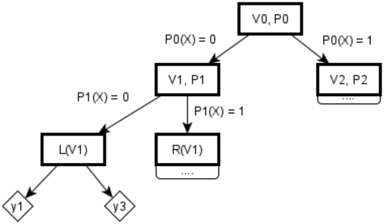
Algorithm 1 Rules induction principle

procedure $\mathrm{RI}(X^l, k)$
    $Z \leftarrow \mathit{starting\_rules\_set}(X^l)$
    $\mathit{last\_inf\_gain} \leftarrow -\inf$
    while True **do**
        $Z^{'} \leftarrow \mathit{generate\_rules\_modification}(X^l, Z)$
        $Z^{'} \leftarrow \mathit{delete\_similar\_rules}(Z \cap Z^{'})$
        $Z^{'} \leftarrow \mathit{select\_most\_informative}(Z^{'})$
        $\mathit{inf\_gain} \leftarrow \mathit{estimate\_informativeness}(Z^l)$
        if $|\mathit{inf\_gain} - \mathit{last\_inf\_gain}| < \epsilon$ **then**
            return Z
        $Z \leftarrow Z^{'}$

The definition of BDT

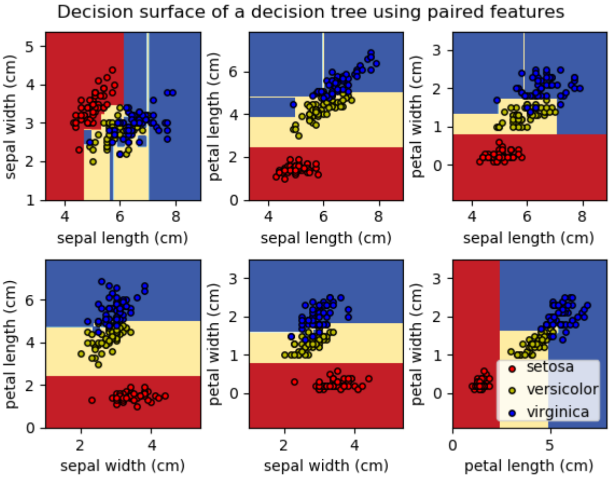
Binary decision tree (BDT) is a data structure corresponds to classic binary search tree with logical predicates in inner nodes and classification results in leafs:

1. $\forall v \in V_{inner} \exists \beta_v : X \rightarrow 0, 1$
2. $\forall v \in V_{leaf} \exists c_v \in Y$



Where $R(V_i)$ - right child of inner node V_i,
$L(V_i)$ - left child,
β_i - predicate of inner node V_i

Iris dataset example



[1] https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html

Learning BDT: pseudocode

Algorithm 2 Rules induction principle

procedure $\textsc{TrainID3}(U \subset X^l)$
    if $\forall (x_i, y_i) \in U : y_i = c$ **then** ▷ All class labels are the same
        return new Leaf(C_v = c)
    $\beta_{max} \leftarrow arg \max_{\beta \in B} I(\beta, U)$ ▷ Find the most informative predicate
    $U_0 \leftarrow \{x \in U, \beta_{max}(x) = 0\}$
    $U_1 \leftarrow \{x \in U, \beta_{max}(x) = 1\}$
    if $U_0 = \emptyset$ or $U_1 = \emptyset$ **then**
        return new Leaf(C_v = majority class)
    $V \leftarrow$ new InnerNode(β_{max})
    $L(V) \leftarrow$ TrainID3(U_0)
    $R(V) \leftarrow$ TrainID3(U_1)
    return v

Information criteria for BDT

How can we choose predicate (line 4 of pseudocode)? We need information criteria:

- The same as for logical rules (see previous slides).
- Gini criterion:

$$I(\beta_v, X^l) = |\{(x_i, , x_j) : \beta_v(x_i) = \beta_v(x_j), y_i = y_j\}|$$

- Dual criterion [D-criterion of Donskoy]:

$$D(\beta_v, X^l) = |\{(x_i, , x_j) : \beta_v(x_i) \neq \beta_v(x_j), y_i \neq y_j\}|$$

Working with skips in dataset

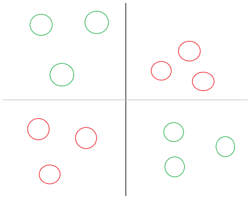
Decision tree is the first algorithm that can process skips in dataset. How does it work?

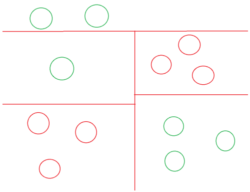
- At the training stage:
  1. If $\beta_v(x)$ is undefined, then we skip the object x during calculation of $I(\beta_v(x), U)$.
  2. Save $q_0 = \frac{|U_0|}{|U|}$ - the probability of the left branch.
- At the stage of usage:
  1. $P_v(y|x) = (1 - \beta_v(x))P_{L(v)}(y|x) + \beta_v(x)P_{R(v)}(y|x)$ - the conditional probability of class y in the node P_v. This rule is used if $\beta_v(x)$ is defined.
  2. For leafs: $P_v(y|x) = [y = C_v]$.
  3. For undefined $\beta_v(x)$: $P_v(y|x) = (1 - q_0)P_{L(v)}(y|x) + q_0 P_{R(v)}(y|x)$
- For such classifier the answer will be: $y = arg \max\limits_{y \in Y} P_{V_0}(y|x)$, where V_0 is a root of Decision Tree.

BDT: advantages an disadvantages

+ It is the interpretable algorithm.
+ It is simple in use.
+ Decision trees work with skips in data.
+ Decision trees support different types of data.

- The greedy algorithm TrainID3 doesn't guarantee the best tree.
- BDT is highly noise sensitivity.
- BDT highly fragment dataset. Usually there are no statistically significant answers in leafs.





CART algorithm

What we can do with BDT disadvantages? There is technique, named as **pruning** (a kind of regularization). Here is the main idea:

- Train a decision tree on training set.
- Get a dataset different from the training set [development set].
- Spread all objects from the development set to created the decision tree.
- Calculate how many inner nodes left without objects and delete them.
- Calculate how many inner nodes could be replaced by their child-nodes or leafs and replace them.

So, we've developed the CART algorithm (**C**lassification **A**nd **R**egression **T**ree)!

Pruning algorithm: pseudocode (part 1)

Algorithm 3 Pruning algorithm

procedure PruneTree(dt: DecisionTree, X^l, X^k) ▷ X^k - dev. set
    for $(x_i, y_i) \in X^k$ **do**
        $v \leftarrow root(dt)$
        while $type(v) \neq Leaf$ **do**
            $v.cnt \leftarrow v.cnt + 1$
            $v \leftarrow$ **if** $\beta_v(x_i)$ **then** $R(v)$ **else** $L(v)$
    for $v \in V_{inner}$ *and* $v.cnt = 0$ **do**
        replace v by $Leaf(C_v = majority\_class(v, X^l))$
        ...

Pruning algorithm: pseudocode

Algorithm 4 Pruning algorithm

procedure PRUNETREE(dt: DecisionTree, X^l, X^k) ▷ X^k - dev. set

  . . .

  for $v \in V_{inner}$ **do**

    $r(v) \leftarrow errors(v)$ ▷ The number of errors on X^k in v

    $r_l(v) \leftarrow errors(L(v))$ ▷ The number of errors in the left child of v

    $r_r(v) \leftarrow errors(R(v))$ ▷ The number of errors in the right child of v

    $r_c(v) \leftarrow errors(v\ as\ Leaf)$ ▷ The number of errors if v is a leaf

    $r_{min} \leftarrow min(r(v), r_l(v), r_r(v), r_c(v))$

    if $r_l(v) = r_{min}$ **then** replace v by $L(v)$

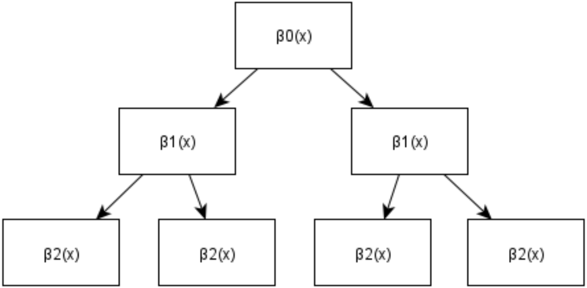
    else if $r_r(v) = r_{min}$ **then** replace v by $R(v)$

    else if $r_c(v) = r_{min}$ **then** replace v by $v\ as\ Leaf$

ODT algorithm

Goal: Simplify the algorithm of BDT training (it is a high-complexity algorithm) and make a regularization automatically.

Idea: Create oblivious decision trees (ODT). These trees contains the same conditions β on each level of tree:



Regression using BDT

We can easily transform a **classification** decision tree to **regression** decision tree:

- We can use the same classes of predicates $\beta_v(x)$. They work in the same way for both trees' types.
- We need to modify information criterion. Use the Variance from statistics:

$$I(U) = \frac{1}{|U|} \sum_{(x_i, y_i) \in U} (y_i - E[y])^2 ,$$

where

$$E[y] = \frac{1}{|U|} \sum_{(x_i, y_i) \in U} y_i$$

- The answer in Leaf node will be an average value of objects' answers.

Random forest: the main idea

We noticed that BDT is the statistically unstable algorithm - it could have a very few objects in leafs to deduce answers. How can we fix it except pruning?

There is a very popular approach: the composition of algorithms. Here is the idea:

- Select k different subsets of the training set.
- Train k models.
- For each new object x call each model from the previous step and:
  - If it is classification task use the majority voting.
  - If it is a regression task use the average value of models' answers.

Random forest regression: pseudocode

Algorithm 5 Train random forest regression tree

procedure TRAINRANDOMFOREST(X^l, k, S_o, S_f) ▷ S_o - the number of objects in a subsample, S_f - the number of features in a subsample
  $models \leftarrow []$
  for $i \in 1 \dots k$ **do**
    $R_o \leftarrow [random(1 \dots l)]_{j=1}^{S_o}$ ▷ Random object indexes
    $R_f \leftarrow [random(1 \dots n)]_{j=1}^{S_f}$ ▷ Random feature indexes
    $X_i \leftarrow \{[f_j(x_i)], y_i : i \in R_o, j \in R_f, (x_i, y_i) \in X^l\}$
    models.add($TrainID3(X_i)$)
  return new RandomForestRegression(models)

Algorithm 6 Random forest regression tree

procedure APPLYRANDOMFOREST(rf: RandomForestRegression, x)
  return $\frac{1}{|rf.models|} \sum_{dt \in rf.models} rf.apply(x)$

Conclusions

- We learned about another approach in ML: the rule-based classification.
- We studied how to apply the rule-based approach to train decision trees.
- We learned how to simplify decision trees.
- We studied how to transform classification tree into regression tree and random forest.