Machine Learning (Lab support)

Decision trees and Ensemble learning

Abdelkrime Aries
Laboratoire de la Communication dans les Systèmes Informatiques (LCSI)
École nationale Supérieure d'Informatique (ESI, ex. INI), Algiers, Algeria

Academic year: 2024-2025

Machine Learning (Lab support)

DT & Ensemble: Introduction

- Decision trees
  - how to create a decision tree using ID3
  - how to create a decision tree using CART
- Random Forests
  - Ensemble learning
  - How a random forest works

Machine Learning (Lab support)

DT & Ensemble: Plan

Section 1

Decision trees

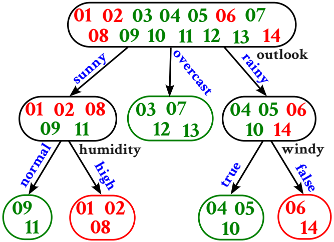
DT & Ensemble

Decision trees

Example of expected model

id	outlook	temp	humidity	windy	play
01	sunny	hot	high	false	no
02	sunny	hot	high	true	no
03	overcast	hot	high	false	yes
04	rainy	mild	high	false	yes
05	rainy	cool	normal	false	yes
06	rainy	cool	normal	true	no
07	overcast	cool	normal	true	yes
08	sunny	mild	high	false	no
09	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

```
if outlook == 'sunny':
    if humidity == 'high':
        return 'no'
    else: # 'normal'
        return 'yes'
elif outlook == 'overcast':
    return 'yes'
else: # 'rainy'
    if not windy:
        return 'yes'
    else:
        return 'no'
```



- **GOAL**: Create a tree by splitting the training data. The leafs must contain homogeneous samples (same class)
- **Estimation**: navigate the tree from the root till a leaf based on the sample's features

DT & Ensemble

Decision trees: Some algorithms

- **ID3 (Iterative Dichotomiser 3)**: developed in 1986 by Ross Quinlan. It can be applied only to nominal characteristics. It is used for ranking.
- **C4.5**: an extension of ID3 by Ross Quinlan. It can be applied on all types of features. It is used for ranking.
- **C5.0**: a commercial extension of C4.5, again by Ross Quinlan.
- **CART (Classification and Regression Trees)**: like C4.5 but uses other metrics. Also, the algorithm supports regression.

DT & Ensemble: Decision trees

Algorithms: Construction (Training)

Data: X, Y
Result: A decision tree
return `build`(X, Y);`// return the tree's root`
function `build`(X', Y')
  n ← new Node();
  if Y' *is homogeneous or stop criteria is reached* **then**
    $s.class \leftarrow \arg\max_k |\{y \in Y' / y = k\}|$ `//` n `is a leaf`
  else
    determine the feature j of X' which better divides Y';
    split (X, Y) into $(X_1, Y_1), \ldots, (X_K, Y_K)$ based on the K values of X'_j;
    $n.feature = j$;
    foreach $k \in \{1, \ldots, K\}$ **do** $n.children[k] \leftarrow$ `build`$(X_k, Y_k))$;
  end
  return n;
end

Algorithm 1: Generating a decision tree (General algorithm)

DT & Ensemble: Decision trees

Algorithms: Search (prediction)

Data: x: a sample; $T = (V, E)$: a decision tree
Result: y: the result class
Let n_{root} be the root node of T;
return `Search` $(n_{root};\ x)$;
function `Search`(n: *a node;* x: *a sample*)
    if n *is a leaf* // `it has no children`
     then
        return $n.class$;
    else
        $k = x[n.feature]$;
        return `Search` $(n.children[k];\ x)$;
    end
    return n;
end

Algorithm 2: Traversing a decision tree (general algorithm)

DT & Ensemble: Decision trees

Stop conditions

- **Homogeneity**: the observations in the node have the same class;
- **Minimum impurity**: the impurity or classification/regression error of observations in the node is less than or equal to this threshold;
- **Minimum number of observations**: the number of observations in a node is less than or equal to this threshold;
- **Depth**: the depth of the node in the tree is less than this threshold.

DT & Ensemble: Decision trees

Review: Advantages

- simple to understand and interpret. We can visualize the trees. Also, we can easily explain the obtained results.
- can work on data with little preparation. For example, they do not need data normalization.
- accept numeric and nominal data. Other learning algorithms are specialized in a single type of data.
- perform well even if their assumptions are somewhat violated by the actual model from which the data was generated.

DT & Ensemble: Decision trees

Review: Limits

- can be too complex to not generalize well (overfitting). This can be adjusted by setting the minimum number of samples in the leaves or by setting the maximum depth of the tree.
- may be unstable due to data variations.
- there are problems that are a bit difficult to learn by decision trees. They are not easy to express, for example: XOR.
- may be biased to the dominant class. So, you have to balance the data before training the system.
- it is not guaranteed to fall on the optimal decision tree.

Section 2

ID3

DT & Ensemble

ID3

- ***Iterative Dichotomize 3***;
- only accepts nominal characteristics;
- only for classification (no regression);
- the training stops if the sets of classes in the leafs are homogeneous.

DT & Ensemble: ID3

Homogeneity of a set

- Shannon's entropy $H(Y)$ to measure the uncertainty of a set Y;
- $H(Y) = 0$: Y contains the same values (one category);
- $H(Y) \geq 1$: Y contains different values;
- given V_y the vocabulary set of Y (unique values).

$$H(Y) = -\sum_{v \in V_y} p(v/Y) \log_2 p(v/Y)$$

$$p(v/Y) = \frac{|\{y/y \in Y \wedge y = v\}|}{|S|}$$

DT & Ensemble: ID3

Set's split

- Y: a set of predictions;
- X_j: the values of the feature j;
- v: one value out of the possible values of X_j (vocabulary V_j);
- for each value $v \in V_j$, we create a set $Y_{j,v}$;
- if $|V_j| = K$, we will have K sets $Y_1, \cdots, Y_K$;

$$split(Y, X_j, v) = Y_{j,v}$$

$$Y_{j,v} = \{y^{(i)} \in Y / X_j^{(i)} \in X_j \land X_j^{(i)} = v\}$$

DT & Ensemble: ID3

Choice of split feature

- Entropy gain (information gain) $IG(Y, X_j)$ is used;

$$j_{best} = \arg\max_j IG(Y, X_j)$$

- It measures how much uncertainty in Y was reduced after Y was split using the feature j;
- IG is the difference between the entropy of Y and the weighted entropy of the split sets;

$$IG(Y, X_j) = H(Y) - \sum_{v \in V_j} p(v/X_j) H(Y_{j,v})$$

DT & Ensemble: ID3

Example

1. Create a node $ROOT$
   - $H(Y) = -p(yes \in Y)\log_2 p(yes \in Y) - p(no \in Y)\log_2 p(no \in Y) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} \approx 0.94$ **(not pure)**
   - $IG(Y, outlook) = 0.94 - [\underbrace{\frac{5}{14}(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5})}_{sunny} + \underbrace{\frac{4}{14}(-\frac{0}{4}\log_2\frac{0}{4} - \frac{4}{4}\log_2\frac{4}{4})}_{overcast} + \underbrace{\frac{5}{14}(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5})}_{rainy}]$
   - $IG(Y, outlook) \approx 0.247, IG(Y, temp) \approx 0.029, IG(Y, humidity) \approx 0.152, IG(Y, wind) \approx 0.048$ **(outlook is the best)**
   - **split the dataset into three datasets (X_1, X_2, X_3), each will be used to create a child of $ROOT$**
2. Create a node N_1 where X_1, Y_1 are split on $X[outlook] = sunny$ (5 samples)
   - $H(Y_1) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} \approx 0.97$ **(not pure)**
   - $IG(Y_1, temp) = 0.97 - [\underbrace{\frac{2}{5}(-\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2})}_{hot} + \underbrace{\frac{2}{5}(-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2})}_{mild} + \underbrace{\frac{1}{5}(-\frac{0}{1}\log_2\frac{0}{1} - \frac{1}{1}\log_2\frac{1}{1})}_{cool}]$
   - $IG(Y_1, temp) \approx 0.57, IG(Y_1, humidity) \approx 0.97, IG(Y_1, wind) \approx 0.02$ **(humidity is the best)**
   - **split the dataset into two datasets (X_{11}, X_{12}), each will be used to create a child of N_1**
3. Create a node N_2 where X_2, Y_2 are split on $X[outlook] = overcast$ (4 samples)
   - $H(Y_2) = -\frac{0}{4}\log_2\frac{0}{4} - \frac{4}{4}\log_2\frac{4}{4} = 0$ **(pure: one class)**
   - $N_2.class =' yes'$

Section 3

CART

DT & Ensemble

CART

- ***Classification and Regression Trees***;
- supports regression;
- tries to minimize a cost function;
- uses pre-pruning based on a stopping criterion;
- creates binary trees.

DT & Ensemble: CART

Homogeneity of a set

- diversity index $Gini(Y)$ to measure the classification error of Y;
- $Gini(Y) = 0$ represents the best division;
- $Gini(Y) = 0.5$ represents the worst division;
- given V_y the vocabulary of Y (unique values or classes);
- In the case of regression, ***MSE*** is used;

$$Gini(S) = \sum_{v \in V_y} p(v/Y)(1 - p(v/Y)) = 1 - \sum_{v \in V_y} p(v/Y)^2$$

$$p(v/Y) = \frac{|\{y/y \in Y \wedge y = v\}|}{|S|}$$

DT & Ensemble: CART

Set's split

- Y: a set of predictions;
- X_j: the values of the feature j;
- v: a value out of the possible values of X_j (vocabulary V_j)
- for each value $v \in V_j$, two sets (Y_G and Y_D) are created;
- Y_G (with values $X_j > v$) and Y_D (with values $X_j \leq v$);

$$split(Y, X_j, v) = (Y_L, Y_R)$$

$$Y_L = \{y^{(i)} \in Y / X_j^{(i)} \in X_j \wedge X_j^{(i)} > v\}$$
$$Y_R = \{y^{(i)} \in Y / X_j^{(i)} \in X_j \wedge X_j^{(i)} \leq v\}$$

DT & Ensemble: CART

Choice of split feature

- Gini impurity of the split $Gini_{split}(Y_L, Y_R)$ is used;

$$j_{best}, v_{best} = \arg\min_{j, v \in X_j} Gini_{split}(Y_L, Y_R) \text{ where } (Y_L, Y_R) = split(Y, X_j, v)$$

- Here, not only the split feature j is explored, but also the value v which minimizes the Gini diversity of the division;

$$Gini_{split}(S_L, S_R) = \frac{|S_L|}{|S_L| + |S_R|} Gini(S_L) + \frac{|S_R|}{|S_L| + |S_R|} Gini(S_R)$$

- for regression, ***MSE*** is used instead of ***Gini***;
- the estimate in the case of regression is the average of the leaf's Y;

If you didn’t understand
the past slides,
good luck understanding
what is coming

Section 4

Random forests

DT & Ensemble

Random forests

- ensemble method;
- uses decision trees for estimation;
- final estimation is done by majority vote;
- uses Bootstraps (sets of random observations) for tree learning;
- uses random features to train each tree: by using fewer attributes in each tree, overfitting issues can be prevented.

DT & Ensemble: Random forests

Ensemble learning

- **Bootstrap aggregating (Bagging)**
  - Concurrently train different estimators on random data subsets.
- **Boosting**
  - Sequentially train estimators on the same data; each one improves the performance of the others.
- **Stacking**
  - Concurrently train estimators on the same data; Train an estimator that combines the predictions of the other estimators.

DT & Ensemble: Random forests

Ensemble learning: Bootstrap aggregating (Bagging)

- **Training**
  - Create **K** Bootstraps (random sets) from the training dataset
  - A bootstrap can have a subset of features
  - Train a model for each Bootstrap
- **Estimation**
  - Use the models to obtain **K** estimations
  - Final estimation (Classification): by majority vote
  - Final estimation (Regression): by mathematical average
- **Examples**
  - Random Forests

DT & Ensemble: Random forests

Ensemble learning: Boosting

- **Training**
  - Train an estimator on the training dataset
  - Make predictions on this dataset to extract samples that are not well-estimated
  - Train another estimator on the same dataset but with more weight on poorly estimated samples
  - Repeat the same operation until generating **K** estimators
- **Estimation**
  - Use the models to obtain **K** estimations
  - Final estimation (Classification): by majority vote
  - Final estimation (Regression): by mathematical average
  - **Examples**
    - AdaBoost

DT & Ensemble: Random forests

Ensemble learning: Stacking

- **Training**
  - Train **K** estimators on the same training dataset
  - These estimators should have different hyperparameters or different training algorithms
  - Train an estimator that takes the outputs of the other **K** estimators as input
  - This final estimator will learn to merge the estimations of the other estimators to produce a better one
- **Estimation**
  - Use the **K** estimators to obtain initial estimations
  - Use the final estimator to combine these estimations

DT & Ensemble: Random forests

Parameters of a Forest

- **Bootstrap**
  - Do we use the dataset as it is, or do we use ***bootstrapping***?
  - What is the size of a Bootstrap?
  - How much randomness do we use to generate the Bootstrap?
- **Feature**
  - How many features should we use per tree?
  - How do we choose these features (amount of randomness)?
- **Tree**
  - How many trees do we use for estimation?
  - Plus other parameters related to the trees

If slides are nodes of a tree,
this slide is a leaf.
SO
stop scrolling
otherwise,
you'll get a leafless tree