Contents

Chapter 1

Decision Trees

Let's start with training set:

Definition 1.0.1: Training set

is defined training set a *set of examples*, where: $\langle x^{(i)}, y^{(i)} \rangle$ where:

- i is the instance of the example
- $x^{(i)} \in X$ is the set of *input*
- $y^{(i)} \in Y$ is the set of *output*

the problem of machine learning is to find a function $h : X \to Y$ that approximates the real function $f : X \to Y$. We have two types of problems:

- **Classification**: Y is a discrete set of values (e.g. $\{0, 1\}$)
- **Regression**: Y is a continuous set of values (e.g. $\mathbb{R}$)

1.1 Hypothesis space

In machine learning the *hypothesis space* H is defined as the set of all possible functions that can be used to approximate the real function $f : X \to Y$. Formally:

Definition 1.1.1: Hypothesis space

A hypothesis space H is defined as the set: $H = \{h | h : X \to Y\}$ where:

- h is a function (hypothesis) that maps input X to output Y
- X the input space (features, domain of data).
- Y the output space (labels, range of data).
- $|H|$ is the size of the hypothesis space (number of possible hypotheses)

this let us to define the model:

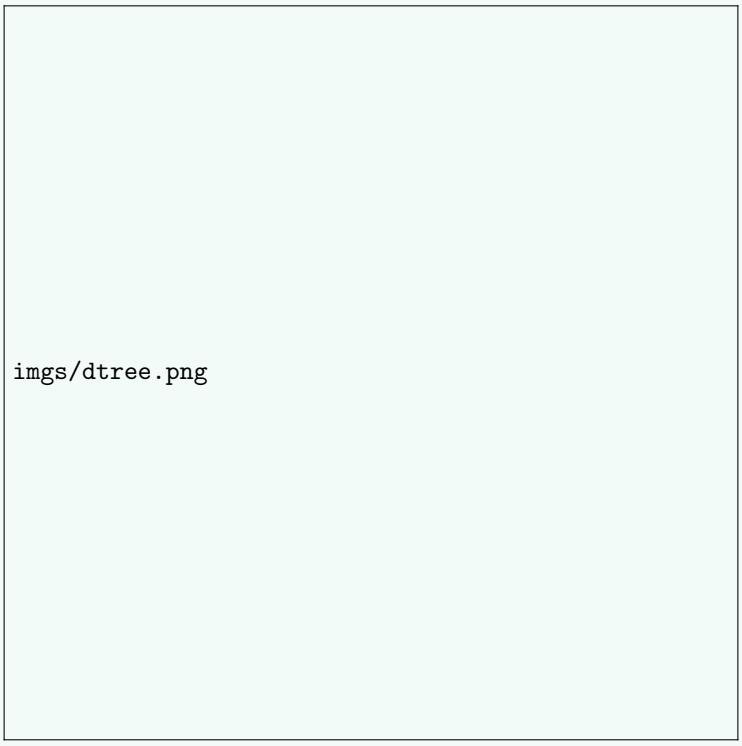
Definition 1.1.2: Model

A model is a way to compute a function $h \in H$ from the training set.

Example 1.1.1 (Decision tree)

A good day to play tennis? Our function F is:

$$\mathbb{F} : Outlook \times Humidity \times Wind \times Temp \rightarrow PlayTennis?$$

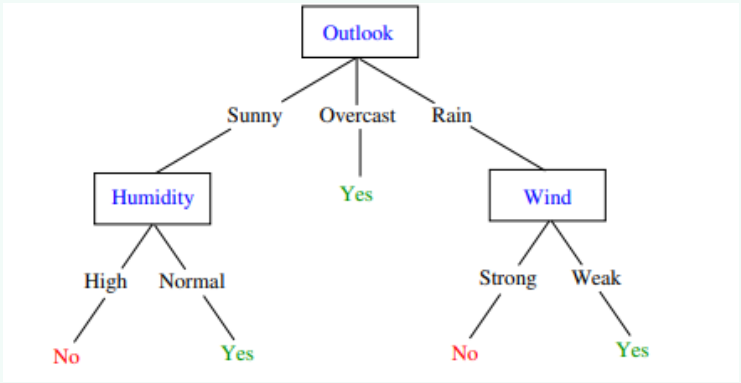


where:

- $Outlook \in \{Sunny, Overcast, Rain\}$
- $Humidity \in \{High, Normal\}$
- $Wind \in \{Weak, Strong\}$
- $PlayTennis? \in \{Yes, No\}$

Every node tests an attribute. Each branch corresponds to one of the possible values for that attribute. Each leaf node assigns a classification (Yes or No), in other words predicts the answer Y.
The prblem configurationg is the following:

- X is the set of all possible $x \in X$ that corresponds to a vector of attributes $(Outlook, Humidity, Wind, Temp)$
- Target function $f : X \rightarrow Y$ is the function that maps the attributes to the target variable $PlayTennis?$ (booleans)
- Hypothesis space $H = \{h | h : X \rightarrow Y\}$ is the set of all possible decision trees that can be constructed using the attributes in X to predict the target variable Y



1.1.1 Top-down inductive construction

Let $X = X_1 \times X_2 \cdots \times X_n$ where $X_i = \{\text{Truee}, \text{False}\}$
Can we represent, for instance, $Y = X_2 \wedge X_5$? or $Y = X_2 \wedge X_5 \vee (\neg X_3) \wedge X_4 \wedge X_1$?
and:

- do we have a decision tree for each h in the space hypothesis?
- if the tree exists, is it unique?
- if it is not unique, do we have a preference?

1.1.2 Entropy

In other words, Entropy meausers the *degree of uncertainty* of the information. It is maximal when X is uniformly distributed (all classes have the same probability) and minimal (zero) when all examples belong to the same class (pure set)

Missing image: imgs/entropy.png

Information Theory (Shannon 1948)

The entropy is the average amount of information produced by a stochastic source of data. The *information* is associated to the *probability* of each datum (the surprise element):

- An event with probability 1 (certain event) provides no information (no surprise): $I(1) = 0$.
- An event with probability 0 (impossible event) provides infinite information (really surprising): $I(0) = \infty$.
- Given two independent events A and B, the information provided by both events is the sum of the information provided by each event:
$$I(A \cap B) = I(A) + I(B)$$

So is natoral defining

$$I(p) = -\log_2(p)$$

Code Theory (Shannon-Fano 1949, Huffman 1952)

The entropy is also related to the avarage number of bits required to transmit outcomes produces by a stochastic source process x.
Let suppose to have n events with same probability $p_i = \frac{1}{n}$. Home many bits do we need to encode these events? The answer is $\log_2(n)$ bits. For instance, if we have 4 events, we need 2 bits to encode them:.
In this case:

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i) = -\sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = \log_2(n)$$

1.1.3 Information Gain

In a decision tree, the goal is to maximize the information gain during the execution of the algorithm. In other words, the final split should result in the minimum possible impurity. Here are the main formulas:

Theorem 1.1.2 Entropy of X

Theorem 1.1.2 Entropy of X

$$H(X) = -\sum_{i=1}^{n} P(X = i) \log_2 P(X = i)$$

Theorem 1.1.3 Conditional Entropy of X given a specific $Y = v$

Theorem 1.1.3 Conditional Entropy of X given a specific $Y = v$

$$H(X \mid Y = v) = -\sum_{i=1}^{n} P(X = i \mid Y = v) \log_2 P(X = i \mid Y = v)$$

This measures the entropy of X restricted to the subgroup where $Y = v$.

Theorem 1.1.4 Conditional Entropy of X given Y

Theorem 1.1.4 Conditional Entropy of X given Y

$$H(X \mid Y) = \sum_{v=1}^{m} P(Y = v)\, H(X \mid Y = v)$$

This is the generalization of **??**, used to evaluate the utility of an attribute. It measures the average impurity that remains in X after splitting the data using all possible values of Y.

Theorem 1.1.5 Information Gain between X and Y

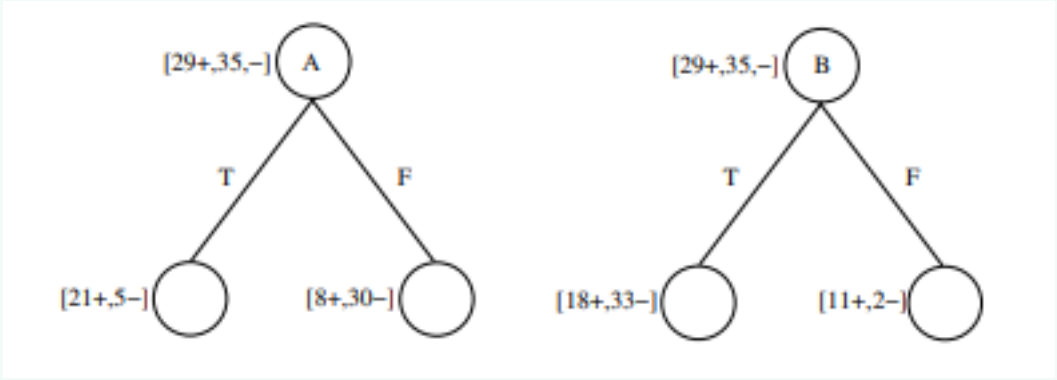
Theorem 1.1.5 Information Gain between X and Y

Here we are! $I(X, Y) = H(X) - H(X \| Y) = H(Y) - H(Y \| X)$

Example 1.1.2 (Information gain)

Example 1.1.2 (Information gain)

Let us measure the entropy reduction of the target variable Y due to some attribute X, that is the information gain $I(Y, X)$ between Y and X



$$H(Y) = -\frac{29}{64}\log_2\left(\tfrac{29}{64}\right) - \frac{35}{64}\log_2\left(\tfrac{35}{64}\right) = 0.994$$

$$H(Y \mid A = T) = -\frac{21}{26}\log_2\left(\tfrac{21}{26}\right) - \frac{5}{26}\log_2\left(\tfrac{5}{26}\right) = 0.706$$

$$H(Y \mid A = F) = -\frac{8}{38}\log_2\left(\tfrac{8}{38}\right) - \frac{30}{38}\log_2\left(\tfrac{30}{38}\right) = 0.742$$

$$H(Y \mid A) = 0.706 \cdot \tfrac{26}{64} \; + \; 0.742 \cdot \tfrac{38}{64} = 0.726$$

$$I(Y, A) = H(Y) - H(Y \mid A) = 0.994 - 0.726 = 0.288$$

$$H(Y \mid B) = 0.872$$

$$I(Y, B) = 0.122$$

Chapter 2

Overfitting

Let us consider the error of the hypothesis h

- on the training set, $error_{train}(h)$
- on the full data set $\mathcal{D}$, $error_{\mathcal{D}}(h)$

Definition 2.0.1 Overfitting

Definition 2.0.1 Overfitting

It's said taht h *overfits* the training set if there exixsts another hypotheses h' such that:

$$\text{error}_{\text{train}(h)} < \text{error}_{\text{train}(h')}$$

but

$$\text{error}_{\mathcal{D}(h)} > \text{error}_{\mathcal{D}(h')}$$

These models (h and h') represent two different situations. The first corresponds to a model that fits the training dataset very closely, including its uncertainty and noise. The second is simpler: it captures only the general trend of the training data and avoids fitting the noise. As a consequence, the error with respect to the true data distribution $\mathcal{D}$ is larger for the first model than for the second. The second one is better! Let's generalised.
But *We do not know* $\mathcal{D}$

2.1 Avoiding the over fitting

2.1.1 Detecting the Overfitting: validation set

For Detecting the Overfitting it's usefull divinding the dats aviables in two disjoint sets:

- **Training set**: set of datas that the model *use for learning.* The dtree is built by this datas
- **validation set**: This set is not shown during the training- It's used as "test" for evaluating the accuracy of the model

2.1.2 Early stopping

This is a proactive strategy. Instead of let the tree grows untill his major complexity, it's sopped first the possibility of Overfitting. The growing of a branch is stopped if these two conditions is verified:

- **The improvement is too small**: if a possible divsion of datas produces a gain of information below a certain threshold, it means that it's not usefull to continue
- **There are not enaugh datas**: if a node contains a number of examples too musch low, any decision taken would be statistically unreliable and probably based on noise. The tree stops to avoid creating rules based on coincidences.

2.1.3 Post - Pruning

This strategy is **reactive**. The decision tree is let grow completely on the training set, which may lead to overfitting, and then the useless or harmful branches are pruned.

Definition 2.1.1 Reduce-Error Post-Pruning

Definition 2.1.1 Reduce-Error Post-Pruning

The *reduce-error post-pruning* technique works as follows:

- build the tree completely
- evaluate each branch using a validation set
- prune the branch whose removal improves accuracy the most
- repeat until no further pruning improves the accuracy