Decision Trees

...based on Toronto and UPenn ML class

6/10/2023

Supervised Classification: Problem Setting

Input : Training labeled examples $\{(\mathbf{x}^{(i)}, y^{(i)})\}$ of unknown target function f such as $y = f(\mathbf{x})$

- Examples $x^{(i)}$ described by their values on some set of features or attributes
- Unknown target function $f : X \to Y$
  - X Set of possible instances
  - Y label space

per ogni label Y(i) viene associata una istanza X(i).
Y(1)=Vento <- Forte=X(1).
Y(2)=Tempo <- Sole=X(2)
queste tuple formano una ISTANZA ETICHETTATA.

Output: Hypothesis $h \in H$ that (best) approximates target function f

- Set of function hypotheses $H = \{h | h : X \to Y\}$

quindi, date delle istanze X e delle label Y, cerchiamo una funzione che approssima al meglio delle possibilità queste istanze nelle label.
NB: una singola label Y può assumere uno tra i valori delle possibili istanze X, che cambia da label a label.

Supervised Classification: Problem Setting

Input : Training labeled examples $\{(\mathbf{x}^{(i)}, y^{(i)})\}$ of unknown target function f such as $y = f(\mathbf{x})$

- Examples $x^{(i)}$ described by their values on some set of features or attributes
- Unknown target function $f : X \rightarrow Y$
  - X Set of possible instances
  - Y label space

Output: Hypothesis $h \in H$ that (best) approximates target function f

- Set of function hypotheses $H = \{h | h : X \rightarrow Y\}$
  - hypothesis h are decision trees

Sample Dataset

- Columns denote features of X
- Rows denote labeled instances $(\mathbf{x}^{(i)}, y^{(i)})$
- Class label (whether a tennis game was played)

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

$\langle x^{(i)}, y^{(i)} \rangle$

Sample Dataset

- Columns denote features of X
- Rows denote labeled instances $(\mathbf{x}^{(i)}, y^{(i)})$
- Class label (whether a tennis game was played)

$\langle x^{(i)}, y^{(i)} \rangle$

Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

- What about

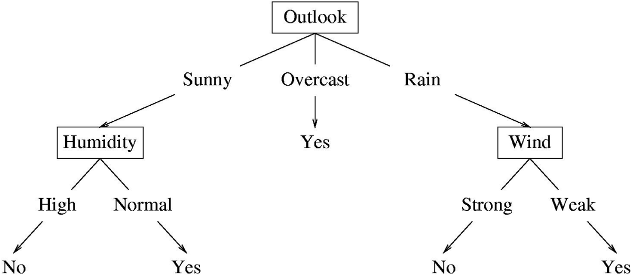
$$x = (Overcast, Mild, Normal, Weak)$$

Decision Trees

queste "Ipotesi H" sono alberi decisionali!

- Make predictions by splitting on features according to a tree structure.

Questa struttura permette una facile comprensione anche senza avere basi matematiche, ma dipende dai dati che ho. Se avessi tutti gli eventi possibili, l'albero sarebbe affidabilissimo, ma anche banale perché non predice nulla.



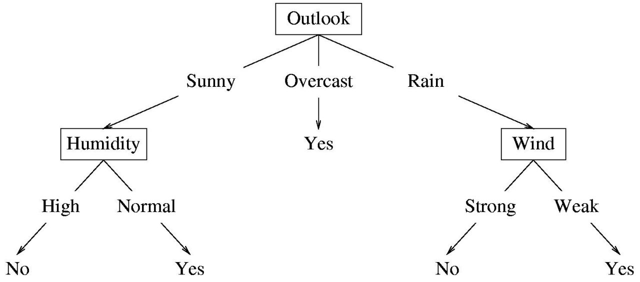
- Each internal node tests one attribute
- Each branch from a node selects one value of that attribute
- Each leaf nodes predicts Y

Decision Trees

- Make predictions by splitting on features according to a tree structure.

L'ideale sarebbe avere qualche caso, lavorare bene su quello e poi avere buoni riferimenti per dati non osservati!

Se non li osservo, come faccio a giudicarli? uso validation set.

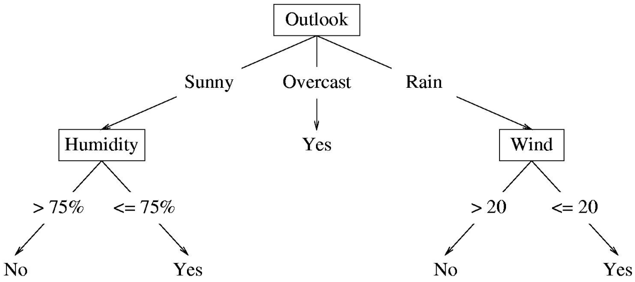


- What prediction would we make for

$$x = (Overcast, Mild, Normal, Weak)$$

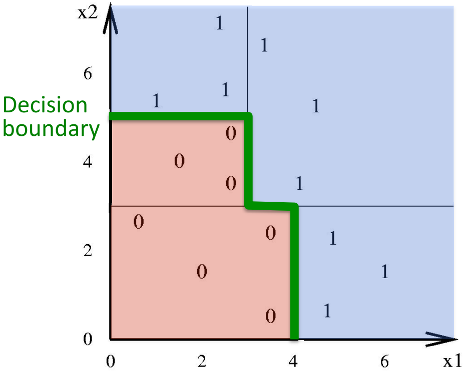
Decision Trees

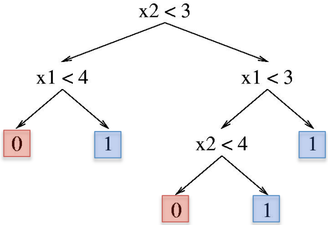
- If features are continuous, internal nodes can test the value of a feature against a threshold



Decision Trees - Decision Boundary

- Decision trees divide the feature space into axis-parallel (hyper-)rectangles
- Each rectangular region is labeled with one label





Another Example (Russel & Norivg)

Model a patron's decision whether to wait for a table

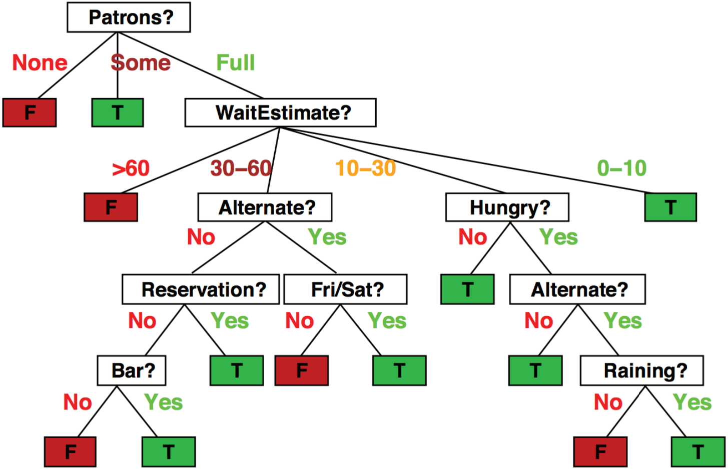
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

Features:

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai or Burger).
10. WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

A possible Decision Tree

- Will I eat at this restaurant?



Core Aspects in Decision Tree & Supervised Learning

- How to automatically find a good hypothesis for training data?
  - This is an algorithmic question, the main topic of computer science
- When do we generalize and do well on unseen data?
  - Learning theory quantifies ability to generalize as a function of the amount of training data and the hypothesis space
  - Occam's razor: use the simplest hypothesis consistent with data!

Core Aspects in Decision Tree & Supervised Learning

- How to automatically find a good hypothesis for training data?
  - This is an algorithmic question, the main topic of computer science
- When do we generalize and do well on unseen data?
  - Learning theory quantifies ability to generalize as a function of the amount of training data and the hypothesis space
  - Occam's razor: use the simplest hypothesis consistent with data!

- Decision trees: find a small decision tree that explains data well
  - NP-hard problem
  - Very nice practical heuristics; top down algorithms, e.g , ID3

Anche se NP-Hard, non vuol dire che non si possa mai usare, se lavoro con piccole istanze si, il problema nasce con tante istanze!

Ockham's Razor

- Principle stated by William of Ockham (1285-1347)
- "Entia non sunt multiplicanda praeter necessitatem"
- entities are not to be multiplied beyond necessity

- **Idea**: The simplest consistent explanation is the best
  - Therefore, the smallest decision tree that correctly classifies all of the training examples is best
    - Finding the provably smallest decision tree is NP-hard
    - So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

La spiegazione più semplice è anche la migliore. Trovare l'albero più semplice (è un problema NP-Hard) è trovare l'albero migliore.

Decision Trees: ID3 algorithm

ID3 è greedy, parte da vuoto, e poi sceglie la feature migliore, su cui inizierò a dividere, e in base a cui partizionerò i dati.
Poi procedo in modo ricorsivo.

ID3: Iterative Dichotomiser 3 (Ross Quinlan)

- greedy approach to build a decision tree top down from the root

Algorithm:

- Start with the whole training set and an empty decision tree.
- Pick the "best" feature/attribute
- Split on that feature and recurse on subpartitions.

Mi fermo quando, per ogni training ho la classificazione corretta.

Decision Trees: ID3 algorithm

ID3 algorithm

node $\leftarrow$ root
repeat
    $A \leftarrow$ the "best" decision attribute for next level nodes
    forall *value a of A* **do**
        add a new descendent node corresponding to attribute a
    end
    Assign training examples to leaf nodes
until *all training examples are perfectly classified*;

Decision Trees: ID3 algorithm

ID3 algorithm

```
node ← root
repeat
    A ← the "best" decision attribute for next level nodes
    forall value a of A do
        add a new descendent node corresponding to attribute a
    end
    Assign training examples to leaf nodes
until all training examples are perfectly classified;
```

Key Question: Which attribute is the best?

Choosing the Best Attribute

Key Problem : choosing which attribute to split a given set of examples

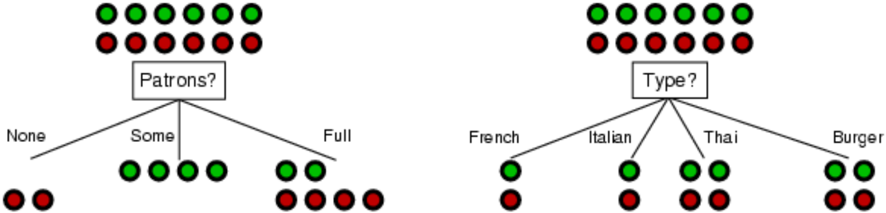
- Some possibilities are:
  - **Random** Select any attribute at random
  - **Least-Values** Choose the attribute with the smallest number of possible values
  - **Most-Values** Choose the attribute with the largest number of possible values
  - **Max-Gain** Choose the attribute that has the largest expected *information gain*

    aspetto che vedremo più nel dettaglio!
    - i.e., attribute that results in smallest expected size of subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

chi genera rami che si "fermano" subito, perché l'informazione di base mi permette di decidere in maniera decisa! Non serve altro.

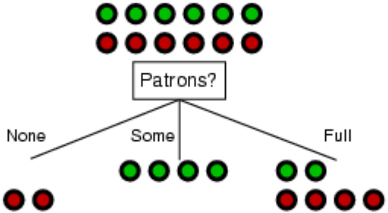
Choosing an Attribute

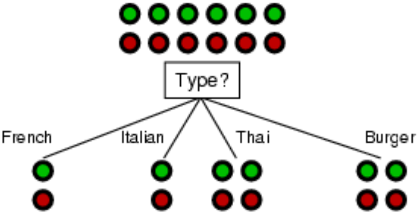
- **Idea** a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Choosing an Attribute

- **Idea** a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





qui, a parte il caso "full" in cui ho "indecisione", negli altri casi ho scelte più "decise".

Qui, in ogni ramificazione, non ho mai una risposta "certa", la avrò andando più in fondo nella ramificazione.

- which split is more informative?

Choosing a Good Split

- How can we quantify uncertainty in prediction for a given leaf node?
  - If all examples in leaf have same class: good, low uncertainty
  - If each class has same amount of examples in leaf: bad, high uncertainty
- **Idea**: Use counts at leaves to define probability distributions; use a probabilistic notion of uncertainty to decide splits.
- A brief detour through information theory...

Quantifying Uncertainty

Uso l'entropia per misurare l'incertezza!

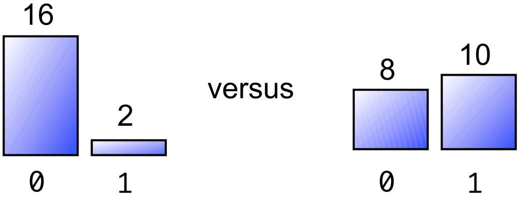
- The entropy of a discrete random variable is a number that quantifies the uncertainty inherent in its possible outcomes.
- The mathematical definition of entropy that we give in a few slides may seem arbitrary, but it can be motivated axiomatically.
- To explain entropy, consider flipping two different coins...

We Flip Two Different Coins

Lanciamo una moneta, nel primo caso sono molto più certo che, dopo un lancio, possa uscire uno "0", la moneta sembra truccata! Nel secondo caso, sono molto più incerto.

Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:
0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?



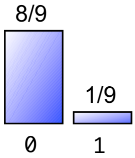
Quantifying Uncertainty

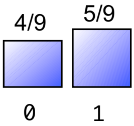
- The entropy of a loaded coin with probability p of heads is given by

$$-plog_2(p) - (1-p)log_2(1-p)$$

(si usa sempre base 2 nel log)

Il segno meno è necessario in quanto, lavorando con un logaritmo tra 0 ed 1, avremmo valori negativi!





entropia 0 = certezza totale
entropia 1 = incertezza totale

$$-\frac{8}{9}\log_2\frac{8}{9} - \frac{1}{9}\log_2\frac{1}{9} \approx \frac{1}{2}$$

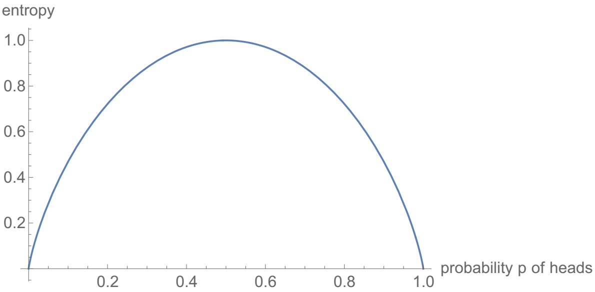
$$-\frac{4}{9}\log_2\frac{4}{9} - \frac{5}{9}\log_2\frac{5}{9} \approx 0.99$$

- Notice: the coin whose outcomes are more certain has a lower entropy.
- In the extreme case $p = 0$ or $p = 1$, we were certain of the outcome before observing. So, we gained no certainty by observing it, i.e., entropy is 0.

entropia 0 = sono sicuro di ciò che esce, ovvero una cosa esce con probabilità 1 (esce sempre), o non esce mai (probabilità 0).

Quantifying Uncertainty

- Can also think of entropy as the expected information content of a random draw from a probability distribution.



- Claude Shannon showed: you cannot store the outcome of a random draw using fewer expected bits than the entropy without losing information.
- Interpretation from information theory: expected number of bits needed to encode label of a randomly drawn example in S.
- So units of entropy are bits; a fair coin flip has 1 bit of entropy.

Entropy

- More generally, the entropy of a discrete random variable Y is given by
$$H(Y) = -\sum_{y \in Y} P(y) \log_2 P(y)$$
- "High Entropy"
  - Variable has a uniform like distribution over many outcomes
  - Flat histogram
  - Values sampled from it are less predictable
- "Low Entropy"
  - Distribution is concentrated on only a few outcomes
  - Histogram is concentrated in a few areas
  - Values sampled from it are more predictable

Entropy

- Suppose we observe partial information X about a random variable Y
  - For example, X = sign(Y).
- We want to work towards a definition of the expected amount of information that will be conveyed about Y by observing X.
  - Or equivalently, the expected reduction in our uncertainty about Y after observing X.

Quando è che conoscere X ci può dare info utili su Y?

Entropy of a Joint Distribution

- Example: X = {Warm , Cool}, Y = {Not Play, Play}

	Not Play	Play
Warm	24/100	1/100
Cool	25/100	50/100

Con coppie di variabili aleatorie, la formula non cambia!
L'entropia sarà >1, perchè ho più info!

$$H(X,Y) = -\sum_{x \in X}\sum_{y \in Y} P(x,y) log_2 P(x,y)$$
$$= -\frac{24}{100} log_2 \frac{24}{100} - \frac{1}{100} log_2 \frac{1}{100} - \frac{25}{100} log_2 \frac{25}{100} - \frac{50}{100} log_2 \frac{50}{100}$$
$$\approx 1.56 bits$$

Entropy of Marginal Distribution

- Example: X = {Warm , Cool}, Y = {Not Play, Play}

	Not Play	Play
Warm	24/100	1/100
Cool	25/100	50/100

$$
\begin{aligned}
H(Y) &= -\sum_{y \in Y} P(y) log_2 P(y) \\
&= -\frac{49}{100} log_2 \frac{49}{100} - \frac{51}{100} log_2 \frac{51}{100} \\
&\approx 1 bits
\end{aligned}
$$

- We used: $P(y) = \sum_x P(x, y)$

Specific Conditional Entropy

- Example: X = {Warm , Cool}, Y = {Not Play, Play}

	Not Play	Play
Warm	24/100	1/100
Cool	25/100	50/100

- What is the entropy of playing tennis, given that it is warm?

entropia condizionata

$$\frac{P(Y \cap X)}{P(X)} = \quad H(Y|X = x) = -\sum_{y \in Y} P(y|x) log_2 P(y|x)$$

$$= -\frac{24}{25} log_2 \frac{24}{25} - \frac{1}{25} log_2 \frac{1}{25}$$

$$\approx 0.24 bits$$

Y = play X = temperature

Sapere se fa caldo o no si rivela una informazione utile!

- We used: $P(y|x) = \frac{P(x,y)}{P(x)}$, and $P(x) = \sum_y P(x, y)$

Conditional Entropy

- Example: X = {Warm , Cool}, Y = {Not Play, Play}

	Not Play	Play
Warm	24/100	1/100
Cool	25/100	50/100

- The expected conditional entropy:

$$
\begin{aligned}
H(Y|X) &= \sum_{x \in X} P(x) H(Y|X = x) \\
&= -\sum_{x \in X} \sum_{y \in Y} P(x, y) \log_2 p(y|x)
\end{aligned}
$$

Conditional Entropy

- Example: X = {Warm , Cool}, Y = {Not Play, Play}

	Not Play	Play
Warm	24/100	1/100
Cool	25/100	50/100

- What is the entropy of playing tennis, given the knowledge of whether or not it is warm?

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} P(x) H(Y|X = x) \\ &= \frac{1}{4} H(\text{playing}|\text{warm}) + \frac{3}{4} H(\text{playing}|\text{cool}) \\ &\approx 0.75 bits \end{aligned}$$

Sapere se fa caldo o non caldo fa passare l'entropia da 1 a 0.75, quindi un po' utile lo è.

Conditional Entropy

Some useful properties

- H is always non-negative
- Chain rule: $H(X,Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
- If X and Y are independent, then X does not affect our uncertainty about Y : $H(Y|X) = H(Y)$
- By knowing X, we can only decrease uncertainty about Y : $H(Y|X) \leq H(Y)$

Sapere Y in funzione di X al massimo può essere come sapere Y e basta, non può farmi sapere più di Y stessa!

Select the next attribute

- Example: X = {Warm , Cool}, Y = {Not Play, Play}

	Not Play	Play
Warm	24/100	1/100
Cool	25/100	50/100

- How much more certain am I about whether tennis will be played if I'm told whether it warm or cool? My uncertainty in Y minus my expected uncertainty that would remain in Y after seeing X.
- This is called the information gain $IG(Y|X)$ in Y due to X, or the mutual information of Y and X

H è la funzione coi log vista prima, non è una probabilità!

$$IG(Y|X) = H(Y) - H(Y|X)$$

- If X is completely uninformative about Y : $IG(Y|X) = 0$
- If X is completely informative about Y : $IG(Y|X) = H(Y)$

se X inutile non guadagno nulla, se X è utilissima guadagno tutto

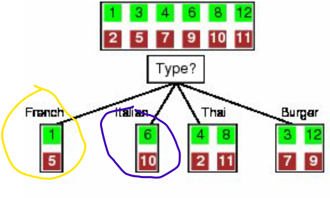
Back to Our Example

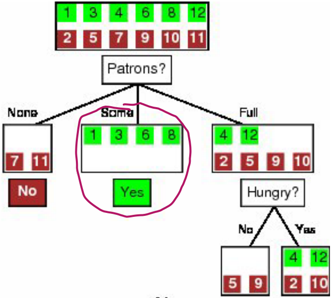
Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	$y_1 =$ Yes
$\mathbf{x}_2$	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	$y_2 =$ No
$\mathbf{x}_3$	No	Yes	No	No	Some	$	No	No	Burger	0–10	$y_3 =$ Yes
$\mathbf{x}_4$	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	$y_4 =$ Yes
$\mathbf{x}_5$	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	$y_5 =$ No
$\mathbf{x}_6$	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	$y_6 =$ Yes
$\mathbf{x}_7$	No	Yes	No	No	None	$	Yes	No	Burger	0–10	$y_7 =$ No
$\mathbf{x}_8$	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	$y_8 =$ Yes
$\mathbf{x}_9$	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	$y_9 =$ No
$\mathbf{x}_{10}$	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	$y_{10} =$ No
$\mathbf{x}_{11}$	No	No	No	No	None	$	No	No	Thai	0–10	$y_{11} =$ No
$\mathbf{x}_{12}$	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	$y_{12} =$ Yes

Features:

1.	Alternate: whether there is a suitable alternative restaurant nearby.
2.	Bar: whether the restaurant has a comfortable bar area to wait in.
3.	Fri/Sat: true on Fridays and Saturdays.
4.	Hungry: whether we are hungry.
5.	Patrons: how many people are in the restaurant (values are None, Some, and Full).
6.	Price: the restaurant's price range ($, $$, $$$).
7.	Raining: whether it is raining outside.
8.	Reservation: whether we made a reservation.
9.	Type: the kind of restaurant (French, Italian, Thai or Burger).
10.	WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).

Back to Our Example





Nel caso del type partiamo da H(y)=1 alla radice, ma, avendo il 50% in ogni singolo ramo, H(y|type) = 1 totale (vedendo tutti i rami sotto). Quindi non guadagno mai info, cioè IG(y|type) = 0

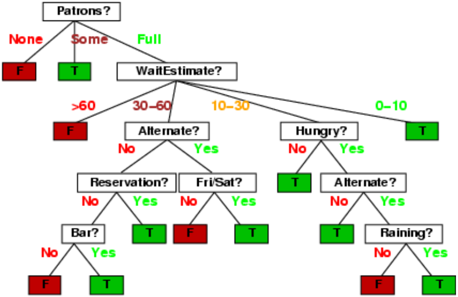
Nel caso di patrons, nei primi due rami ho un guadagno, perchè da una parte ho tutti "no", dall'altra tutti "si". (nei primi due rami ho H(y|patrons)=0, perchè sono sicuro. Ho più incertezza nel terzo ramo, ma comunque il GAIN totale è >0.

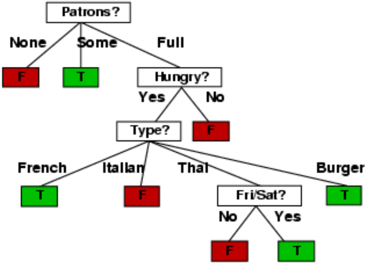
$$IG(Y|X) = H(Y) - H(Y|X)$$

$$IG(Y|type) = 1 - \left[\frac{2}{12}H(Y|Fr.) + \frac{2}{12}H(Y|It.) + \frac{4}{12}H(Y|Thai) + \frac{4}{12}H(Y|Bur.)\right] = 0$$

$$IG(Y|Patrons) = 1 - \left[\frac{2}{12}H(Y|None) + \frac{4}{12}H(Y|Some) + \frac{6}{12}H(Y|Full)\right] \approx 0,541$$

Which Tree is Better?





Decision Trees: ID3 algorithm

ID3(***X***, ***A***, ***Y***)

\# ***X*** the data, ***A*** the set of attributes, ***Y*** labels associated to ***X***

Let T a new tree (just the root node)

if *all instances in* ***X*** *have the same label* y **then**

    Label(T)=y; **return** T

end

se tutte le etichette sono uguali (tutte vere/tutte false), fine.

forall *attributes* $A \in \boldsymbol{A}$ **do**

    $IG(Y|A) = H(Y) - H(Y|A) = H(Y) - \sum_{a \in A} P(A=a) H(Y|A=a)$

end

Sennò per ogni attributo calcolo IG.

$A^* \leftarrow \arg\max_A IG(Y|A)$

Sia A* l'attributo/etichetta che massimizza il guadagno

\# Assign A^* as decision attribute for next level nodes

Label(T)=A^*

forall *value* a *of* A^* **do**

    $\boldsymbol{X_a} \leftarrow$ instances in ***X*** with $A^* = a$, $\boldsymbol{Y_a}$ the associated labels

    T_a=ID3($\boldsymbol{X_a}$,$\boldsymbol{A} \setminus \{A^*\}$, $\boldsymbol{Y_a}$)

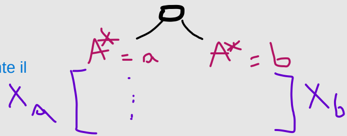
    Add a branch/edge from T to T_a labeled a

end

Return T

per ogni valore possibile di A*, prendo le istanze di X contenenti i valori di A*, ed applico iterativamente l'algoritmo.

ovviamente i sottoalberi Xa e Xb faranno ricorsivamente il lavoro visto escludendo A* dagli attributi.
In particolare Xa lavorerà con tuple aventi A*=a
e Xb lavorerà con tuple aventi A*=b



Avoiding Overfitting

Alla fine dell'albero dobbiamo avere dei nodi del tipo "vero/falso".

ID3 algorithm fits perfectly the training set...

- How can we avoid overfitting?
  - Stop growing when data split is not statically significant
  - Acquire more training data
  - Remove irrelevant attributes
  - Grow full tree, then post-prune
- How to select the “best” tree:
  - ...

Come evitare overfitting?
- non devo far crescere troppo l'albero
- rimuovere attributi che non portano miglioramenti
- ottenere l'albero intero, poi tolgo i nodi.

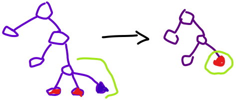
Reduced Error by Pruning

- Split data into training and validation set
- Grow tree based on training set
- Do until further pruning is harmful:
  1. Evaluate impact on validation set of pruning each possible node
  2. Greedily remove the node that most improves validation set accuracy

Decision Tress Miscellany

- Problems
  - Exponentially less data at lower levels
  - Big trees can overfit data
  - Greedy algorithms don't (necessarily) yield the global optimum
- Handling continuous attributes
  - Split based on a threshold, chosen to maximize information gain
- Decision trees can also be used for regression on real-valued outputs. Choose splits to minimize squared error, rather than maximize information gain

Ho training e validation. Se provo a togliere sottoalberi (quindi parti finali), e devo vedere se l'accuratezza migliora o meno. Se non ho miglioramenti, mi devo fermare. Togliere etichetta = metto foglia finale avente per valore il valore dell'etichetta maggiormente presente. (es: Tempo: 3 sole, 1 pioggia -> sole).



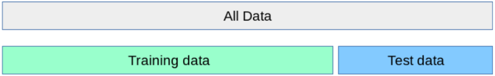
tolgo la label, il valore più probabile è rosso, allora metto foglia rossa

K-fold Cross-Validation

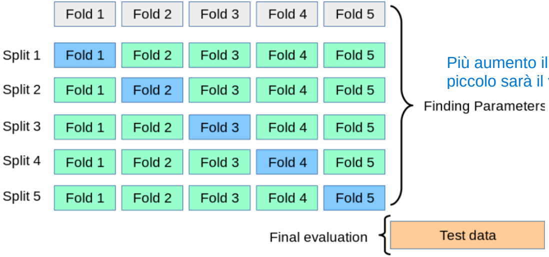
Abbiamo visto la validazione: dato training set, prelevo un "pezzo" (validation set), e lo uso per "testare" la configurazione migliore. Trovata questa, la applico al testing set (del quale non ho "risposte", sono previsioni.) Per questo uso il validation set, è un testing set coi risultati confrontabili.

Robust hyperparameter search technique

- Data is split into training set and test set (no validation set)
- Test set is further divided into *k* smaller sets called folds



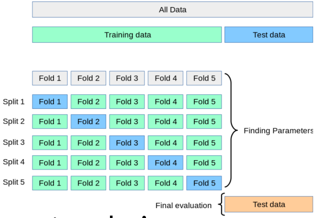
Il problema del validation set è che, oltre ad essere piccolo (circa 20% del training), potrebbe indirizzarvi verso una configurazione che va bene per quel validation set ma male per altri. Come posso provare con più dati? KFOLD CROSS VALID.



Più aumento il numero di fold, più piccolo sarà il validation set.

Diviso training set in "x" blocchi "fold", e di volta in volta (in totale lo faccio proprio "x" volte), prendo un fold per validation, e il restante per training. Poi nella seconda iterazione scelgo il secondo fold per il validation, e il restante per training etc...
Ciò che trovo alla fine verrà usato per il test vero e proprio.

K-fold Cross-Validation



Training and Hyperparameter choice

- For each value of the hyperparameter
  - For each *Split*
    1. Train the model using $k - 1$ of the folds as training data
    2. Validate the resulting model the remaining part of the data (used as a validation set)
- The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop.
- Choose the hyperparameter that yields the best performance accuracy
- Once the optimal hyperparameter is determined, train the model on the entire training set and assess its performance on the test set