Artificial Intelligence

Lecture 12: Decision Tree & Naive Bayes

Credit: Ansaf Salleb-Aouissi, and "Artificial Intelligence: A Modern Approach", Stuart Russell and Peter Norvig, and "The Elements of Statistical Learning", Trevor Hastie, Robert Tibshirani, and Jerome Friedman, and "Machine Learning", Tom Mitchell.

Decision Tree

Tree Classifiers (Decision Tree)

- Popular classification methods.
- Easy to understand, simple algorithmic approach.
- No assumption about linearity.
- **History:**
  - CART (Classification And Regression Trees): Friedman 1977.
  - ID3 and C4.5 family: Quilan 1979-1983.
  - Refinements in mid 1990's (e.g., pruning, numerical features etc.).
- **Applications**:
  - Botany (e.g., New Flora of the British Isles Stace 1991).
  - Medical research (e.g., Pima Indian diabetes diagnosis, early diagnosis of acute myocardial infarction).
  - Computational biology (e.g., interaction between genes)

Tree Classifiers

- The terminology **Tree** is graphic.
- However, a decision tree is grown from the root downward. The idea is to send the examples down the tree, using the concept of information entropy.
- **General Steps to build a tree:**
  1. Start with the root node that has all the examples.
  2. Greedy selection of the next best feature to build the branches. The splitting criteria is *node purity*.
  3. Class majority will be assigned to the leaves.

Classification

Given: Training data: $(x_1, y_1), \ldots, (x_n, y_n)$, $x_i \in \mathbb{R}^d$ and y_i is discrete (categorical/qualitative), $y_i \in Y$.

Example $Y = \{-1, +1\}$, $Y = \{0, 1\}$

Task: Learn a classification function, $f: \mathbb{R}^d \to Y$

In the case of Tree Classifiers:

1. No need for $x_i \in \mathbb{R}^d$, so no need to turn categorical features into numerical features.
2. The model is a tree.

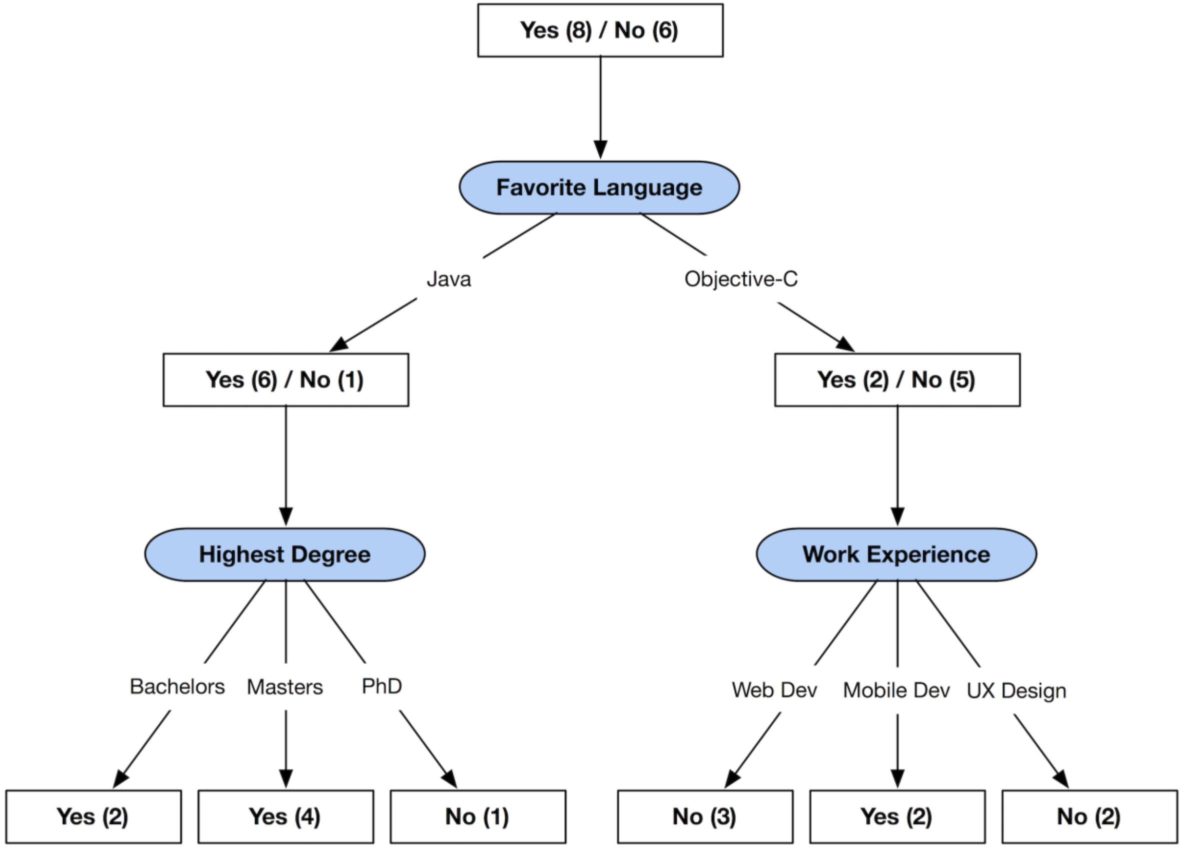
Toy example

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Masters	UX Design	Java	TRUE	?

Toy example

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no



Splitting criteria in ID3

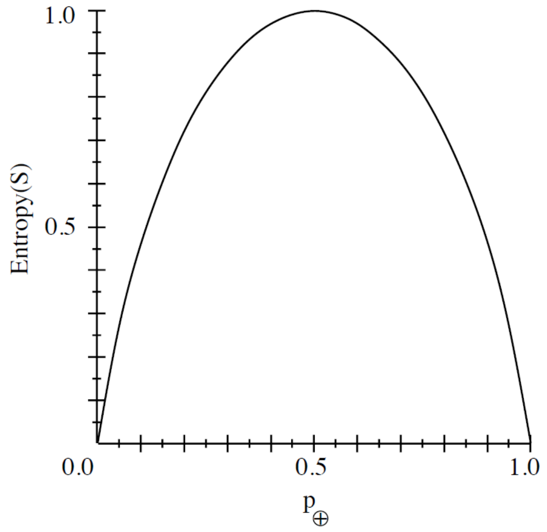
1. The central choice is selecting the next attribute to split on.
2. We want some criteria that measure the homogeneity or impurity of examples in the nodes:
   (a) Quantify the mix of classes at each node.
   (b) Maximum if equal number of examples from each class.
   (c) Minimum if the node is pure.
3. A perfect measure commonly used in *Information Theory*:

$$Entropy(S) = -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$$

$p_{\oplus}$ is the proportion of positive examples.
$p_{\ominus}$ is the proportion of negative examples.

Splitting criteria in ID3

$$Entropy(S) = -\, p_{\oplus}\log_2 p_{\oplus} - \; p_{\ominus}\log_2 p_{\ominus}$$



In general, for c classes:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

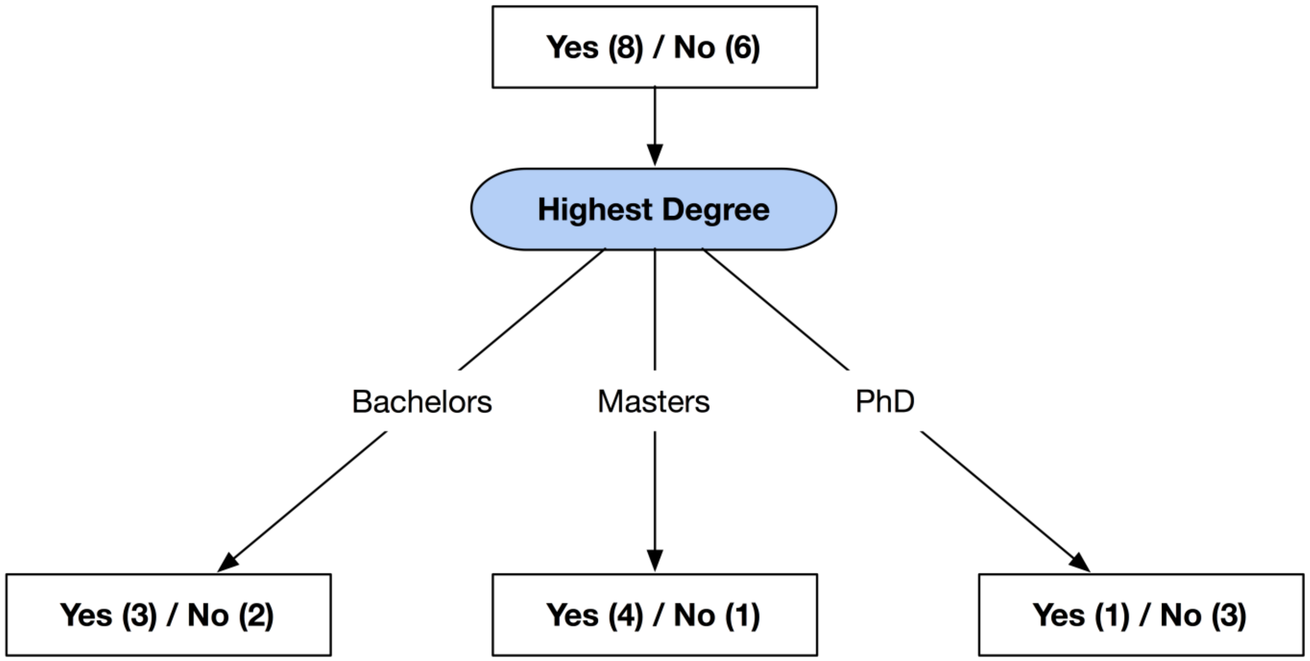
Splitting criteria in ID3

- Now each node has some entropy that measures the homogeneity in the node.
- How to decide which attribute is best to split on based on entropy?
- We use **Information Gain** that measures the expected reduction in entropy caused by partitioning the examples according to the attributes:

$$Gain(S, A) = Entropy(S) - \sum_{v \in \mathrm{Value(A)}} \frac{|S_v|}{|S|} Entropy(S_v)$$

Back to the example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



Entropy(3, 2) = 0.971

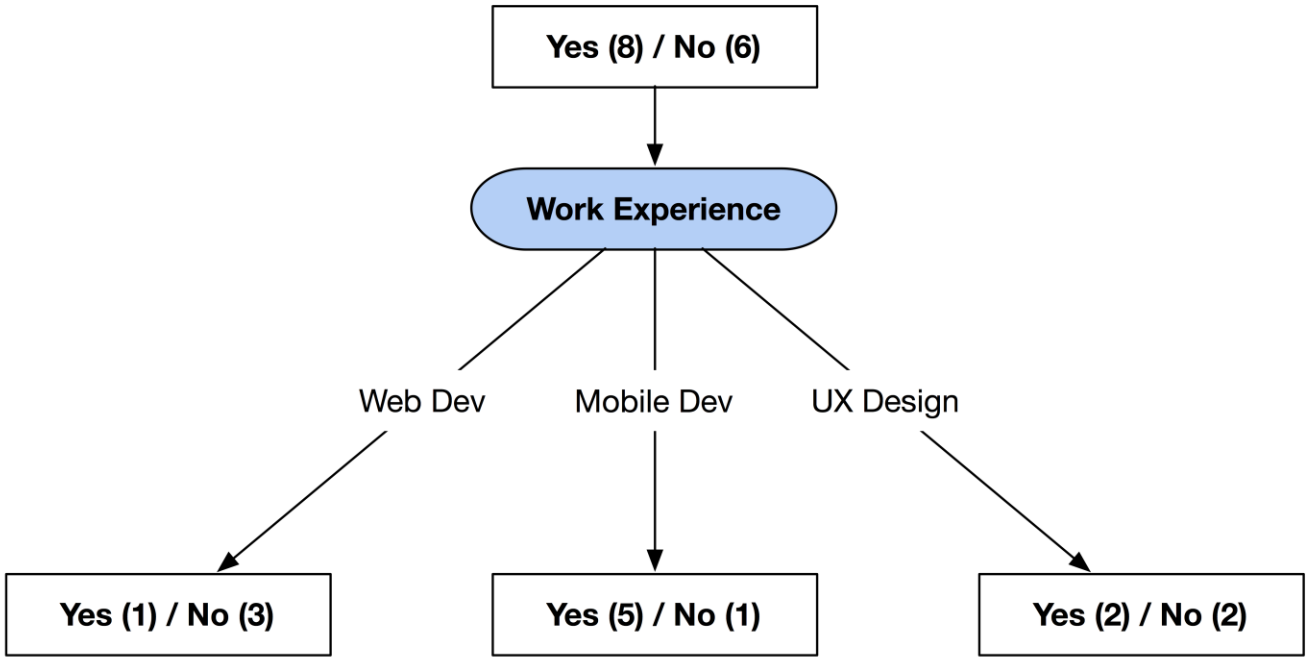
Entropy(4, 1) = 0.722

Entropy(1, 3) = 0.811

Gain(S, Highest Degree) = 0.985 – (5/14) x 0.971 – (5/14) x 0.722 – (4/14) x 0.811 = 0.149

Back to the example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985

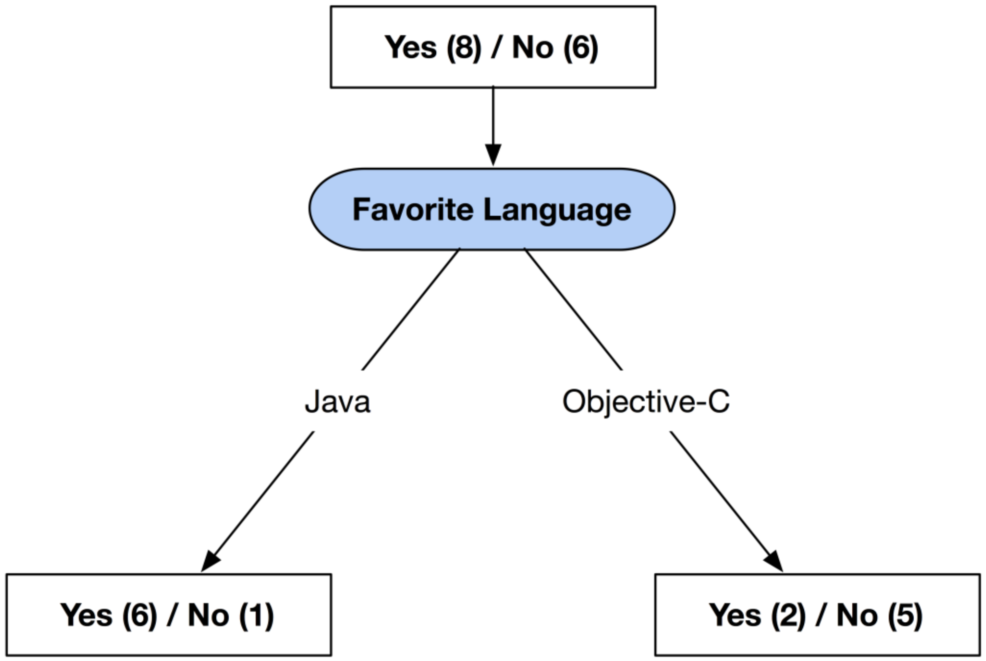


Entropy(1, 3) = 0.811 Entropy(5, 1) = 0.650 Entropy(2, 2) = 1.000

Gain(S, Work Experience) = 0.985 – (4/14) x 0.811 – (6/14) x 0.650 – (4/14) x 1.000 = 0.189

Back to the example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985



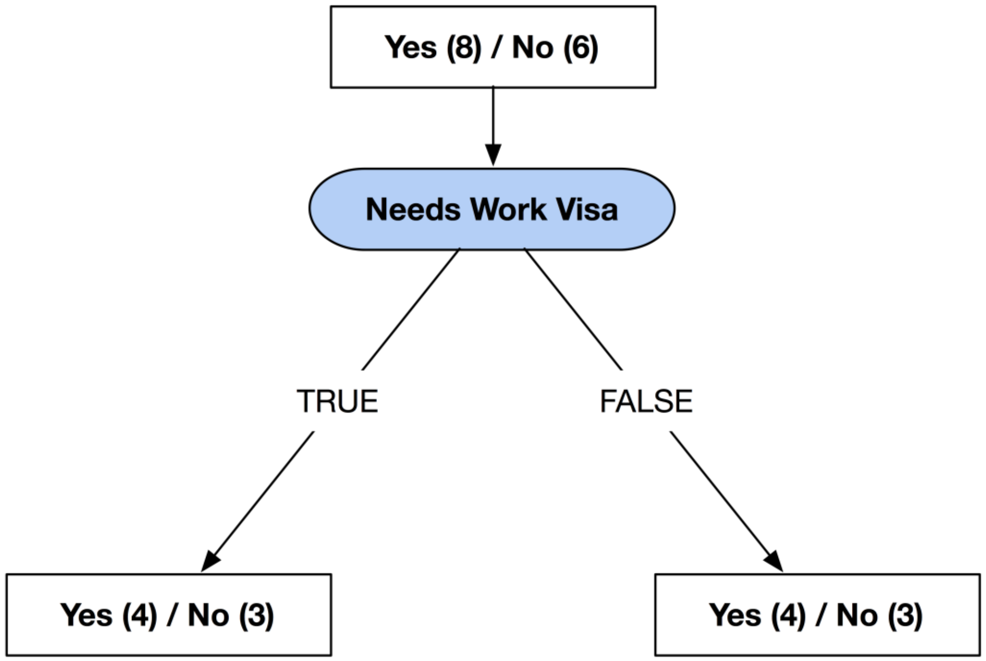
Entropy(6, 1) = 0.592

Entropy(2, 5) = 0.863

Gain(S, Favorite Language) = 0.985 – (7/14) x 0.592 – (7/14) x 0.863 = 0.258

Back to the example

Entropy(8, 6) = - (8/14) x log(8/14) - (6/14) x log(6/14) = 0.985

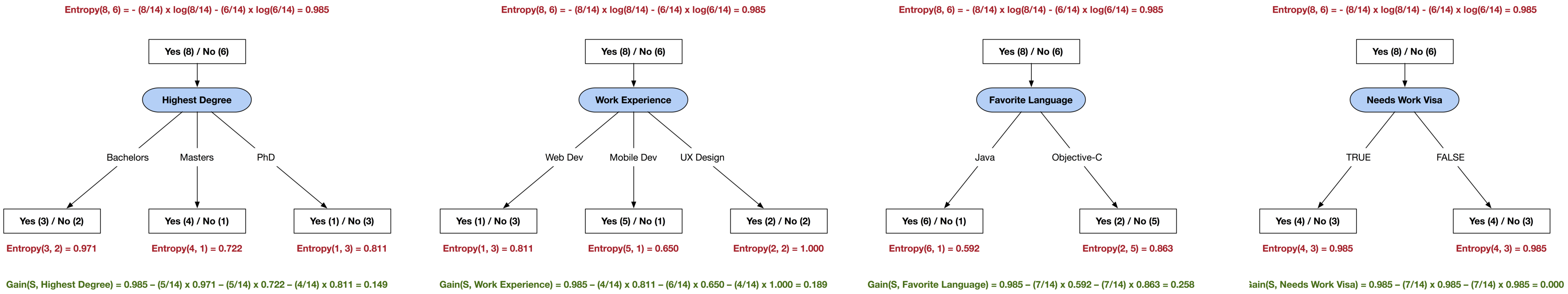


Entropy(4, 3) = 0.985

Entropy(4, 3) = 0.985

Gain(S, Needs Work Visa) = 0.985 – (7/14) x 0.985 – (7/14) x 0.985 = 0.000

Back to the example



Back to the example

Feature	Information Gain
Highest Degree	0.149
Work Experience	0.189
Favorite Language	**0.258**
Needs Work Visa	0.000

At the first split starting from the root, we choose the attribute that has the max gain.

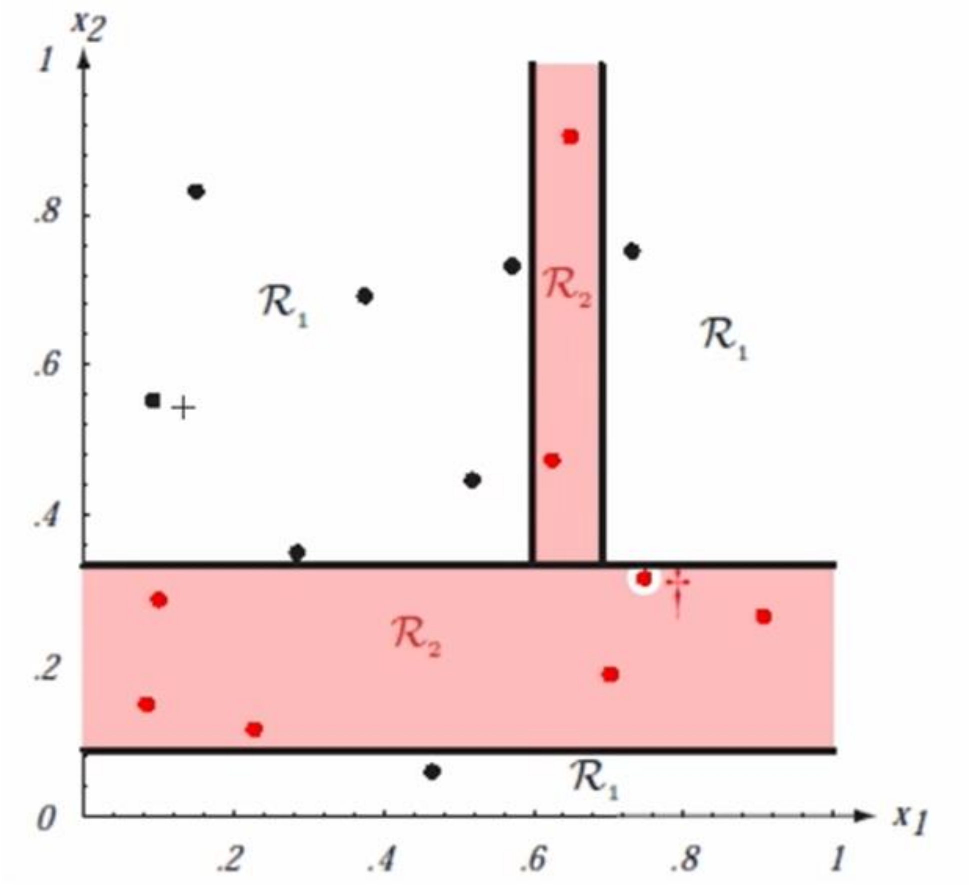
Then, we re-start the same process at each of the children nodes (if node not pure).

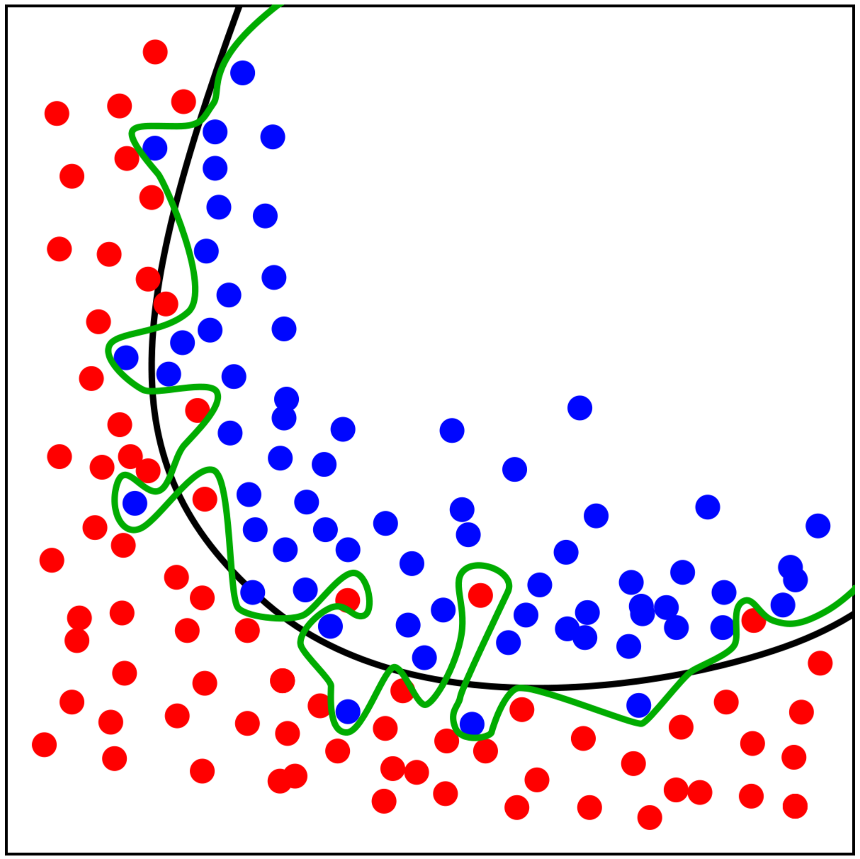
Numerical features

Papers Published	Years of Work	Grade Point Average	Needs Work Visa	Hire
0 paper(s)	5 year(s)	3.20	TRUE	yes
5 paper(s)	1 year(s)	3.64	FALSE	yes
4 paper(s)	6 year(s)	2.92	TRUE	yes
10 paper(s)	4 year(s)	4.00	TRUE	yes
12 paper(s)	3 year(s)	3.21	TRUE	no
0 paper(s)	8 year(s)	3.37	TRUE	no
0 paper(s)	5 year(s)	4.00	FALSE	yes
8 paper(s)	3 year(s)	2.59	FALSE	no
0 paper(s)	7 year(s)	3.70	FALSE	yes
4 paper(s)	7 year(s)	3.78	TRUE	no
2 paper(s)	9 year(s)	4.00	FALSE	yes
9 paper(s)	4 year(s)	4.00	FALSE	no
7 paper(s)	4 year(s)	2.71	TRUE	yes
0 paper(s)	2 year(s)	3.03	FALSE	no

Overfitting the data

决策树天生过拟合，因为它可以分到每个样本一类，error是0，所以需要控制它





Pruning strategies

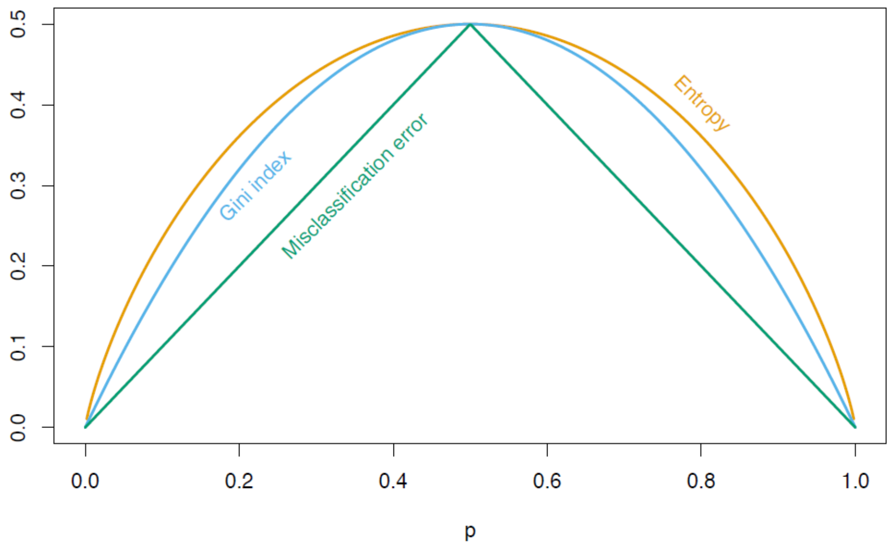
To get suitable tree sizes and avoid overfitting:

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training examples. (difficult to know when to stop!).
- Grow a complex tree then to prune it back (Best strategy found). Use a validation set to evaluate the utility of post-pruning (remove a subtree if the performance of the new tree is no worse than the original tree).

CART

- Adopt same greedy, top-down algorithm.
- Binary splits instead of multiway splits.
- Uses Gini Index instead of information entropy.

$$Gini = 1 - p_{\oplus}^2 - p_{\ominus}^2$$



Practical considerations

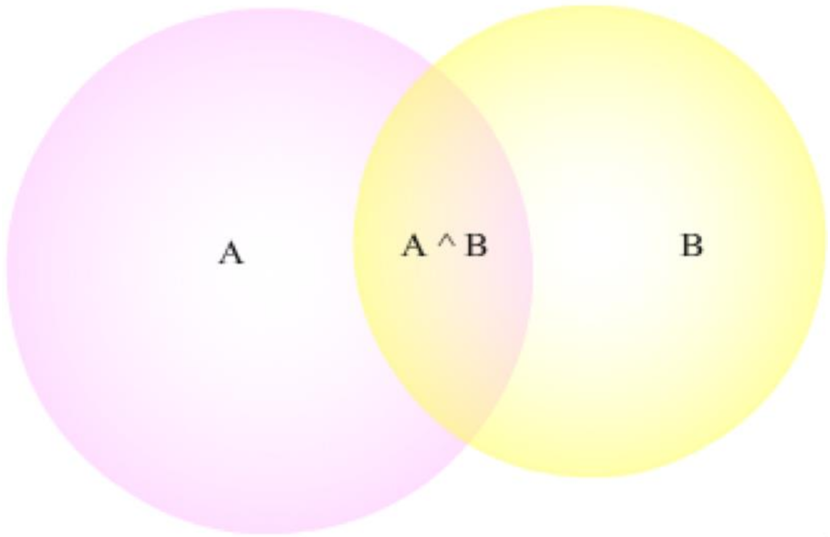
1. Consider performing dimensionality reduction beforehand to keep the most discriminative features.
2. Use ensemble methods. E.g., Random Forest, have a great performance.
3. Balance your dataset before training to prevent the tree from creating a tree biased toward the classes that are dominant.
   - Under-sampling: reduce the majority class
   - Over-sampling: Synthetic data generation for the minority class (e.g., SMOTE).

Tree classifiers: Pros & Cons

+ Intuitive, interpretable (but…).
+ Can be turned into rules.
+ Well-suited for categorical data.
+ Simple to build.
+ No need to scale the data.
- Unstable (change in an example may lead to a different tree).
- Univariate (split one attribute at a time, does not combine features).
- A choice at some node depends on the previous choices.
- Need to balance the data.

Naive Bayes

Conditional Probability



$$p(A|B) = \frac{p(A \wedge B)}{p(B)}$$

$$p(A \wedge B) = p(A|B) * p(B)$$

Bayes Rule

Writing $p(A \wedge B)$ in two different ways:

$$p(A \wedge B) = p(B|A) * p(A)$$

$$p(A \wedge B) = p(A|B) * p(B)$$

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B)}$$

$p(A|B)$ is called posterior (posterior distribution on A given B.)

$p(A)$ is called prior.

$p(B)$ is called evidence.

$p(B|A)$ is called likelihood.

后验概率，先验概率，证据，似然

Bayes Rule

	A	not A	Sum
B	P(A and B)	P(not A and B)	P(B)
Not B	P(A and not B)	P(not A and not B)	P(not B)
	P(A)	P(not A)	1

- This table divides the sample space into 4 mutually exclusive events.
- The probability in the margins are called marginals and are calculated by summing across the rows and the columns.

Another form:

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B|A) * p(A) + p(B|\neg A) * p(\neg A)}$$

Example of Using Bayes Rule

$$p(A|B) = \frac{p(B|A) * p(A)}{p(B|A) * p(A) + p(B|\neg A) * p(\neg A)}$$

- **A: patient has cancer.**
- **B: patient has a positive lab test.**

$p(A) = 0.008$	$p(\neg A) = 0.992$
$p(B|A) = 0.98$	$p(\neg B|A) = 0.02$
$p(B|\neg A) = 0.03$	$p(\neg B|\neg A) = 0.97$

$$p(A|B) = \frac{0.98 \times 0.008}{0.98 \times 0.008 + 0.03 \times 0.992} = 0.21$$

Why probabilities?

Why are we bringing here a Bayesian framework?

Recall Classification framework:

Given: Training data: $(x_1, y_1), \ldots, (x_n, y_n)$, $x_i \in \mathbb{R}^d$, $y_i \in Y$.

Task: Learn a classification function, $f\colon \mathbb{R}^d \to Y$

Learn a mapping from x to y.

We would like to find this mapping $f(x) = y$ through $p(y|x)$!

Discriminative Algorithms

- **Discriminative Algorithms:**
  - Idea: model $p(y|x)$, conditional distribution of y given x.
  - In Discriminative Algorithms: find a decision boundary that separates positive from negative example.
  - To predict a new example, check on which side of the decision boundary it falls.
  - Model $p(y|x)$ directly.

Generative Algorithms

- **Generative Algorithms** adopt a different approach:
  - Idea: Build a model for what positive examples look like.
  - Build a different model for what negative example look like.
  - To predict a new example, match it with each of the models and see which match is best.
  - Model $p(x|y)$ and $p(y)$!
  - Use Bayes rule to obtain $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$
  - To make a prediction:

$$\text{argmax}_y p(y|x) = \text{argmax}_y \frac{p(x|y)p(y)}{p(x)}$$

$$\text{argmax}_y p(y|x) \approx \text{argmax}_y p(x|y)p(y)$$

Naive Bayes Classier

- Probabilistic model.
- Highly practical method.
- Application domains to natural language text documents.
- Naive because of the strong independence assumption it makes (not realistic).
- Simple model.
- Strong method can be comparable to decision trees and neural networks in some cases.

Setting

- A training data (x_i, y_i), x_i is a feature vector and y_i is a discrete label.
- d features, and n examples.
- Example: consider document classification, each example is a documents, each feature represents the presence or absence of a particular word in the document.
- We have a training set.
- A new example with feature values $x_{new} = (a_1, a_2, \ldots, a_d)$.
- We want to predict the label y_{new} of the new example.

Setting

$$y_{new} = \text{argmax}_{y \in \mathbb{Y}} \; p(y|a_1, a_2, \cdots, a_d)$$

Use Bayes rule to obtain:

$$y_{new} = \text{argmax}_{y \in \mathbb{Y}} \; \frac{p(a_1, a_2, \cdots, a_d|y) * p(y)}{p(a_1, a_2, \cdots, a_d)}$$

$$y_{new} = \text{argmax}_{y \in \mathbb{Y}} \; p(a_1, a_2, \cdots, a_d|y) * p(y)$$

Can we estimate these two terms from the training data?

1. $p(y)$ can be easy to estimate: count the frequency with which each label y.
2. $p(a_1, a_2, \ldots, a_d|y)$ is not easy to estimate unless we have a very very large sample. (We need to see every example many times to get reliable estimates)

Naive Bayes Classifier

Makes a simplifying assumption that the feature values are conditionally independent given the label. Given the label of the example, the probability of observing the conjunction a_1, a_2, …,a_d is the product of the probabilities for the individual features:

$$p(a_1, a_2, \cdots, a_d|y) = \prod_j p(a_j|y)$$

Naive Bayes Classier:

$$y_{new} = \text{argmax}_{y \in \mathbb{Y}} \; p(y) \prod_j p(a_j|y)$$

Can we estimate these two terms from the training data?

Yes!

Algorithm

Learning: Based on the frequency counts in the dataset:

1. Estimate all $p(y)$, $\forall y \in Y$.
2. Estimate all $p(a_j|y)$, $\forall y \in Y$, $\forall a_j$.

Classification: For a new example, use:

$$y_{new} = \text{argmax}_{y \in \mathbb{Y}} p(y) \prod_j p(a_j|y)$$

Note: No model per se or hyperplane, just count the frequencies of various data combinations within the training examples.

Example

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Bachelors	Mobile Dev	Objective-C	TRUE	yes
Masters	Web Dev	Java	FALSE	yes
Masters	Mobile Dev	Java	TRUE	yes
PhD	Mobile Dev	Objective-C	TRUE	yes
PhD	Web Dev	Objective-C	TRUE	no
Bachelors	UX Design	Objective-C	TRUE	no
Bachelors	Mobile Dev	Java	FALSE	yes
PhD	Web Dev	Objective-C	FALSE	no
Bachelors	UX Design	Java	FALSE	yes
Masters	UX Design	Objective-C	TRUE	no
Masters	UX Design	Java	FALSE	yes
PhD	Mobile Dev	Java	FALSE	no
Masters	Mobile Dev	Java	TRUE	yes
Bachelors	Web Dev	Objective-C	FALSE	no

Highest Degree	Work Experience	Favorite Language	Needs Work Visa	Hire
Masters	UX Design	Java	TRUE	?

Can we predict the class of the new example?

Example

$$y_{new} = \text{argmax}_{y \in \{yes,no\}} p(y) * p(Masters|y) * p(UX\ Design|y) * p(Java|y) * p(TRUE|y)$$

$$p(yes) = 8/14 = 0.572$$

$$p(no) = 6/14 = 0.428$$

Conditional probabilities:

$$p(masters|yes) = 4/8 \quad p(masters|no) = 1/6$$

$$p(UX\ Design|yes) = 2/8 \quad p(UX\ Design|no) = 2/6$$

$$p(Java|yes) = 6/8 \quad p(Java|no) = 1/6$$

$$p(TRUE|yes) = 4/8 \quad p(TRUE|no) = 3/6$$

$$p(yes) * p(Masters|yes) * p(UX\ Design|yes) * p(Java|yes) * p(TRUE|yes) = 0.026$$

$$p(no) * p(Masters|no) * p(UX\ Design|no) * p(Java|no) * p(TRUE|no) = 0.002$$

$$y_{new} = yes$$

Estimating probabilities

m-estimate of the probability: $$p(a_j|y) = \frac{n_c + m * p}{n_y + m}$$

where:

n_y: total number of examples for which the class is y.

n_c: total number of examples for which the class is y and feature $x_j = a_j$.

m: called equivalent sample size

Intuition:

Augment the sample size by m virtual examples, distributed according to prior p (prior estimate of each value).

If prior is unknown, assume uniform prior: if a feature has k values, we can set $p=1/k$.

Naive Bayes: Summary

1. Naive Bayes is a linear classier.
2. Incredibly simple and easy to implement.
3. Works wonderful for text.

To be continued