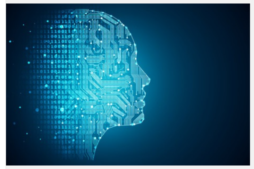


Data Science for Researchers and Scholars

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science and Engineering, Bioinformatics & Genomics, Public Health Sciences and Neuroscience
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

From generative to discriminative models

- Generative models model the data generative process
- Discriminative models focus on discriminating between classes
- Discrimination is easier than generative modeling
- Decision trees are an example of discriminative classifiers

Learning to predict class labels by playing "20 questions"

Learning to predict whether Joe will enjoy machine learning

- **You:** Is the course a Data Science course?
- **Me:** Yes
- **You:** Has Joe done well in programming?
- **Me:** Yes
- **You:** Has Joe done well in calculus?
- **Me:** No
- **You:** I predict this student will not like this course.

- Goal of learner: Figure out what questions to ask, and in what order, and what to predict when you have answered enough questions

Decision tree representation

In the simplest case,

- Each internal node tests on an attribute
- Each branch corresponds to an attribute value
- Each leaf node corresponds to a class label

In general,

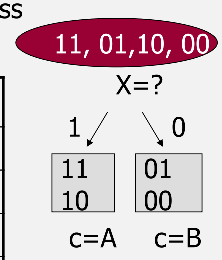
- Each internal node corresponds to a test (on input data sample)
    - Test outcomes are mutually exclusive and exhaustive
    - Tests may be univariate or multivariate
    - Each branch corresponds to an outcome of a test
    - Each leaf node corresponds to a class label

A data set may be represented by many decision trees

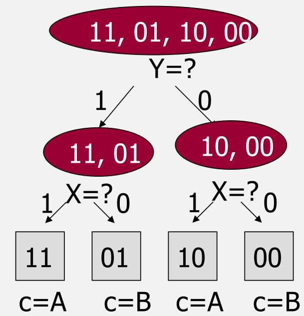
Attributes (x, y) — Class (c)

	x	y	c
1	1	1	A
2	0	1	B
3	1	0	A
4	0	0	B

Data set



Tree 1



Tree 2

Should we choose Tree 1 or Tree 2? Why?

Learning Decision Tree Classifiers

- Decision trees are especially well suited for representing simple rules for classifying data samples that are described by discrete attribute values
- Decision tree learning algorithms
  - Implement Ockham's razor as a model selection bias (simpler decision trees are preferred over more complex trees)
  - Are relatively efficient – linear in the size of the decision tree and the size of the data set
  - Produce easy-to-understand classifiers
  - Are often among the first to be tried on a new data set

Learning Decision Tree Classifiers

- Ockham's razor recommends that we pick the simplest decision tree that is consistent with the training set
- Simplest tree is one that takes the fewest bits to encode (we will see why in a bit)
- There are far too many trees that are consistent with training data
- Searching for the simplest tree that is consistent with the training set is not typically computationally feasible

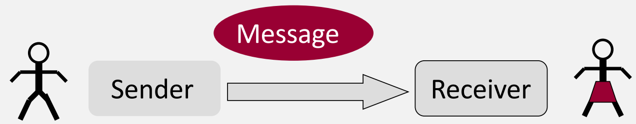
Solution

- Use a greedy algorithm – not guaranteed to find the simplest tree – but works well in practice
- Or restrict the space of hypothesis to a subset of simple trees

Information – Some intuitions

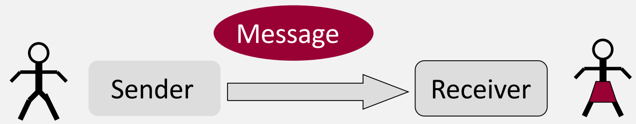
- Information reduces uncertainty
- Information is relative – to what you already know
- Information in a message is related to how surprising the message is
- Information depends on context

Digression: Information and Uncertainty



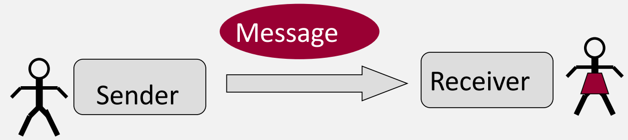
- You are stuck inside.
- You and I are both generally familiar with the weather in Pennsylvania.
- I do not lie, and you trust me.
- You send me out to report back to you on what the weather is like.
  - On a July afternoon in Pennsylvania, I walk into the room and tell you it is hot outside
  - On a January afternoon in Pennsylvania, I walk into the room and tell you it is hot outside

Digression: Information and Uncertainty



- On a July afternoon in Pennsylvania, I walk into the room and tell you it is hot outside
- On a January afternoon in Pennsylvania, I walk into the room and tell you it is hot outside
- Which of the above two messages is more informative?
- Whichever is more surprising given what you know about what the Pennsylvania weather is like in different months of the year!
  - “That it is hot outside on a January afternoon”
- It is this intuition that Claude Shannon formalized in information theory

Digression: Information and Uncertainty



- How much information does a message contain?
- If my message to you describes a scenario that you expect with certainty, the information content of the message for you is zero
- The more surprising the message to the receiver, the greater the amount of information conveyed by the message
- What does it mean for a message to be surprising?

Digression: Information and Uncertainty

- Suppose I have a coin with heads on both sides and you know that I have a coin with heads on both sides.
- I toss the coin, and without showing you the outcome, tell you that it came up heads.
- How much information did I give you? Zero!
- Suppose I have a fair coin and you know that I have a fair coin.
- I toss the coin, and without showing you the outcome, tell you that it came up heads.
- How much information did I give you? 1 bit

Measuring Information

- Without loss of generality, assume that messages are binary – made of 0s and 1s.
- Conveying the outcome of a fair coin toss requires 1 bit of information
  - Must specify which one out of two equally likely outcomes occurred
- Conveying the outcome of a random experiment (the value of a random variable) with 8 equally likely outcomes requires 3 bits
- Conveying an outcome of that is certain takes 0 bits
- In general, if an outcome has a probability *p*, the information content of the corresponding message is

$$I(p) = -\log_2 p \qquad I(0) = 0$$

Information is Subjective

- Suppose there are 3 agents – Sahar, Neil, David, in a world where a fair dice has been tossed.
- Sahar observes the outcome is a "6" and whispers to Neil that the outcome is "even" but David knows nothing about the outcome.

  - Information gained by Sahar by looking at the outcome of the dice $= -\log_2\left(\frac{1}{6}\right) = \log_2 6$ bits.
  - Sahar's uncertainty about the outcome $=\ 0$
  - Information conveyed by Sahar to David = 0 bits.
  - David's uncertainty about the outcome $= \log_2 6$ bits.
  - Information gained by Neil from Sahar $= \log_2 3$ bits.
  - Neil's uncertainty about the outcome after hearing from Sahar $=\ \log_2 6 -\ \log_2 3$ bits

Information and Shannon Entropy

- Suppose we have a message that conveys the result of a random experiment with *m* possible discrete outcomes, with probabilities

$$p_1, p_2, \ldots p_m$$

The expected information content of such a message is called the entropy of the probability distribution

$$H(p_1, p_2, .. p_m) = \sum_{i=1}^{m} p_i I(p_i)$$

$$I(p_i) = -\log_2 p_i \ \text{ provided } \ p_i \neq 0$$

$$I(p_i) = 0 \ \text{ otherwise}$$

Shannon's entropy as a measure expected information

- Let $\vec{P} = (p_1, \cdots p_n)$ be a discrete probability distribution over the n outcomes of a random experiments (values of a random variable)
- Then the Shannon Entropy $H(\vec{P})$ of the distribution $\vec{P}$ is given by

$$H(\vec{P}) = -\sum_{p=1}^{n} p_i \, I\,(p_i)$$

- For example, $H\left(\frac{1}{2}, \frac{1}{2}\right) = -\sum_{p=1}^{n} p_i \log_2(p_i)$

$$= -\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) = 1 \text{ bit}$$

$$H(1,0) = -(1)\log_2(1) - 0(0) = 0 \text{ bit}$$

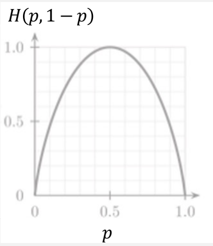
Shannon entropy offers a measure of expected information supplied by the

- outcome of a random experiment
- value of a random variable

Properties of Shannon Entropy

- $\forall \vec{P}\ H(\vec{P}) \geq 0$
- $H(\vec{P}) \leq n$
- If $\forall i\ p_i = \frac{1}{n}$ (all outcomes are equally likely) $H(\vec{P}) = \log_2 n$
- If $\exists i\ p_i = 1$ (one outcome is certain) $H(\vec{P}) = 0$

- Entropy of the outcome of a fair coin toss is maximized when the two outcomes are equally likely
- Entropy of the outcome of a fair coin toss is zero if the outcome is certain



Shannon's entropy as a measure of information

- For any distribution $\vec{P}$, $H(\vec{P})$ is the optimal number of binary questions required on average to determine an outcome drawn from P.
- We can extend these ideas to talk about how much information is conveyed by the observation of the outcome of one experiment about the possible outcomes of another (mutual information)

Coding Theory Perspective

- Suppose you and I both know the distribution $\vec{P}$
- I choose an outcome according to $\vec{P}$
- Suppose I want to send you a message about the outcome
- You and I could agree in advance on the questions
- I can simply send you the answers
- Optimal message length on average is $H(\vec{P})$

Entropy of a random variable

For a random variable X taking values $a_1 \ldots . a_n$,

$$H(X) = -\sum_X P(X)\log_2 P(X)$$

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

If $\mathbf{X}$ is a set of random variables,

$$H(\mathbf{X}) = -\sum_{\mathbf{x}} P(\mathbf{X})\log_2 P(\mathbf{X})$$

Joint Entropy and Conditional Entropy

For random variables X and Y, the joint entropy

$$H(X,Y) = -\sum_{X,Y} P(X,Y)\log_2 P(X,Y)$$

Conditional entropy of X given Y

$$H(X|Y) = -\sum_{X,Y} P(X,Y)\log_2 P(X|Y)$$

$$= \sum_{Y} P(Y)H(X|Y=a)$$

$$H(X|Y=a) = -\sum_{X} P(X,Y=a)\log_2 P(X|Y=a)$$

Joint Entropy and Conditional Entropy

Some Useful results :

$$\left.\begin{aligned} H(X,Y) &\le H(X) + H(Y) \\ H(Y \mid X) &\le H(Y) \end{aligned}\right\}$$

(When do we have equality?) ← When Y is independent of X

Chain rule for Entropy

$$\begin{aligned} H(X,Y) &= H(X) + H(Y|X) \\ &= H(Y) + H(X|Y) \end{aligned}$$

Proof follows from the definition of entropy and the laws of probability

Example of entropy calculations

$P(X = H; Y = H) = 0.2. \ P(X = H; Y = T) = 0.4$

$P(X = T; Y = H) = 0.3. \ P(X = T; Y = T) = 0.1$

$H(X, Y) = -0.2 \log_2 0.2 - 0.3 \log_2 0.3 - 0.4 \log_2 0.4 - 0.1 \log_2 0.1$ 1.85

$P(X = H) = 0.6. \ H(X) = 0.97$

$P(Y = H) = 0.5. \ H(Y) = 1.0$

$P(Y = H | X = H) = 0.2/0.6 = 0.333$

$P(Y = T | X = H) = 1 - 0.333 = 0.667$

$P(Y = H | X = T) = 0.3/0.4 = 0.75$

$P(Y = T | X = T) = 0.1/0.4 = 0.25$

$H(Y|X)$ $H(X,Y) - H(X) = 0.88$

Mutual Information

For a random variable X and Y, the average mutual information between X and Y

$$I(X,Y) = H(X) + H(Y) - H(X,Y)$$

Or by using chain rule

$$H(X,Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y)$$

$$I(X,Y) = H(X) - H(X \mid Y)$$

$$I(X,Y) = H(Y) - H(Y \mid X)$$

In terms of probability distributions,

$$I(X,Y) = \sum_{X,Y} P(X=a, Y=b) \log_2 \frac{P(X=a, Y=b)}{P(X=a)P(Y=b)}$$

Question: When is $I(X,Y) = 0$?

Decision Tree Classifiers

On average, the information needed to convey the class membership of a random instance drawn from nature is $H(\vec{P})$

Nature

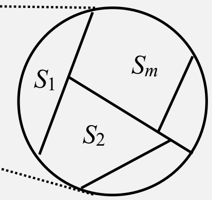
Instance

Classifier

Class label

☺

Training Data S



$$H(\hat{P}) = -\sum_{i=1}^{i=m} (\hat{p}_i)\log_2(\hat{p}_i) = \hat{H}(X)$$

where $\hat{P}$ is an estimate of P

and X is a random variable with distribution P

S_i is the multi-set of examples belonging to class C_i

Learning Decision Tree Classifiers

- Nature encodes a certain amount of information in the training data
- The amount of information present in the training data is equal to the entropy of the distribution
- The task of the learner then is to identify a series of questions that optimally extract the information needed to classify samples from the distribution
- The learned model is stored in the form of a decision tree

Learning Decision Tree Classifiers

Information gain based decision tree learner

- Start with the entire training set at the root
- Recursively add nodes to the tree
    - The nodes correspond to the questions that yield the greatest expected reduction in entropy (or the largest expected information gain)
        - until some termination criterion is met (e.g., the training data at every leaf node has zero entropy)

Information gain based decision tree learner

- Base case: If all data belong to the same class, create a leaf node with that label
- Otherwise, recursively
  - Calculate the information gain for each feature if we use it to split the data
  - Pick the feature with the highest information gain, partition the data based on values of the feature

Learning Decision Tree Classifiers - Example

Samples: ordered 3-tuples of attribute values corresponding to

Height (tall, short)

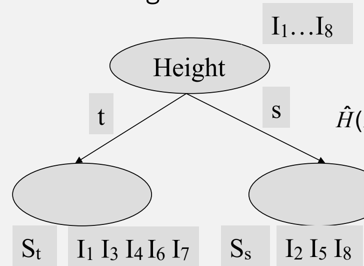
Hair (dark, blonde, red)

Eye (blue, brown)

Training Data

Sample	Class label	
I_1	(t, d, l)	+
I_2	(s, d, l)	+
I_3	(t, b, l)	–
I_4	(t, r, l)	–
I_5	(s, b, l)	–
I_6	(t, b, w)	+
I_7	(t, d, w)	+
I_8	(s, b, w)	+

Learning Decision Tree Classifiers - Example



$$\hat{H}(X) = -\frac{3}{8}\log_2\frac{3}{8} - \frac{5}{8}\log_2\frac{5}{8} = 0.954 bits$$

$$\hat{H}(X \mid Height = t) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971 bits$$

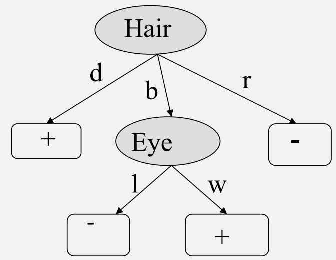
$$\hat{H}(X \mid Height = s) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918 bits$$

$$\hat{H}(X \mid Height) = \frac{5}{8}\hat{H}(X \mid Height = t) + \frac{3}{8}\hat{H}(X \mid Height = s) = \frac{5}{8}(0.971) + \frac{3}{8}(0.918) = 0.95 bits$$

Similarly, $\hat{H}(X \mid Eye) = 0.607 bits$ and $\hat{H}(X \mid Hair) = 0.5 bits$

Hair is the most informative because it yields the largest reduction in entropy. Test on the value of Hair is chosen to correspond to the root of the decision tree

Learning Decision Tree Classifiers - Example



In practice, we need some way to prune the tree to avoid overfitting the training data.

How do we prevent over-fitting?

- Early stopping
- Post pruning

How do we prevent over-fitting?

Base case:

- If all data belong to the same class, create a leaf node with that label
- ***OR*** all the data has the same feature values
- ***OR*** We've reached a particular depth in the tree

Idea:

- Stop building the tree early
- Check if the information gain of the split being considered the is statistically significantly better than that of a random split

Prune if information gain is not significantly > 0

- Evaluate Candidate split to decide if the resulting information gain is significantly greater than zero as determined using a suitable hypothesis testing method at a desired significance level

Example: χ^2 statistic

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}}$$

In the 2-class, binary (L, R) split case,

- n_1 samples belong to class 1, n_2 to class 2; $N = n_1 + n_2$
- Split sends pN to L and $(1-p)N$ to R
- Random split would send pn_1 of class 1 to L and pn_2 of class 2 to L
- Null hypothesis – the split is not better than random
- The critical value of χ^2 depends on the degrees of freedom which is 1 in this case (for a given p, n_{1L} fully specifies n_{2L}, n_{1R} *and* n_{2R})

In general, the number of degrees of freedom can be > 1

Prune if information gain is not significantly > 0

$$\chi^2 = \sum_{j=1}^{Branches-1}\sum_{i=1}^{Classes}\frac{\left(n_{ij}-n_{ie_j}\right)^2}{n_{ie_j}}$$

$$N = n_1 + n_2 + \ldots n_{Classes}$$

$$p = [p_1 p_2 \ldots . p_{Branches}]; \quad \sum_{j=1}^{Branches} p_j = 1$$

$$n_{ie_j} = p_j n_i$$

- The greater the value of χ^2 the less likely it is that the split is random.
- For a sufficiently high value of χ^2, the difference between the expected (random) split is statistically significant and we reject the null hypothesis that the split is random.

$$\text{Degrees of freedom} = (Classes - 1)(Branches - 1)$$

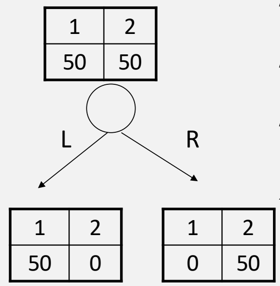
χ^2 Table

Chi-square Distribution Table

d.f.	.995	.99	.975	.95	.9	.1	.05	.025	.01
1	0.00	0.00	0.00	0.00	0.02	2.71	3.84	5.02	6.63
2	0.01	0.02	0.05	0.10	0.21	4.61	5.99	7.38	9.21
3	0.07	0.11	0.22	0.35	0.58	6.25	7.81	9.35	11.34
4	0.21	0.30	0.48	0.71	1.06	7.78	9.49	11.14	13.28
5	0.41	0.55	0.83	1.15	1.61	9.24	11.07	12.83	15.09
6	0.68	0.87	1.24	1.64	2.20	10.64	12.59	14.45	16.81
7	0.99	1.24	1.69	2.17	2.83	12.02	14.07	16.01	18.48
8	1.34	1.65	2.18	2.73	3.49	13.36	15.51	17.53	20.09
9	1.73	2.09	2.70	3.33	4.17	14.68	16.92	19.02	21.67
10	2.16	2.56	3.25	3.94	4.87	15.99	18.31	20.48	23.21
11	2.60	3.05	3.82	4.57	5.58	17.28	19.68	21.92	24.72
12	3.07	3.57	4.40	5.23	6.30	18.55	21.03	23.34	26.22
13	3.57	4.11	5.01	5.89	7.04	19.81	22.36	24.74	27.69
14	4.07	4.66	5.63	6.57	7.79	21.06	23.68	26.12	29.14
15	4.60	5.23	6.26	7.26	8.55	22.31	25.00	27.49	30.58
16	5.14	5.81	6.91	7.96	9.31	23.54	26.30	28.85	32.00
17	5.70	6.41	7.56	8.67	10.09	24.77	27.59	30.19	33.41
18	6.26	7.01	8.23	9.39	10.86	25.99	28.87	31.53	34.81
19	6.84	7.63	8.91	10.12	11.65	27.20	30.14	32.85	36.19
20	7.43	8.26	9.59	10.85	12.44	28.41	31.41	34.17	37.57
22	8.64	9.54	10.98	12.34	14.04	30.81	33.92	36.78	40.29
24	9.89	10.86	12.40	13.85	15.66	33.20	36.42	39.36	42.98
26	11.16	12.20	13.84	15.38	17.29	35.56	38.89	41.92	45.64
28	12.46	13.56	15.31	16.93	18.94	37.92	41.34	44.46	48.28
30	13.79	14.95	16.79	18.49	20.60	40.26	43.77	46.98	50.89
32	15.13	16.36	18.29	20.07	22.27	42.58	46.19	49.48	53.49
34	16.50	17.79	19.81	21.66	23.95	44.90	48.60	51.97	56.06
38	19.29	20.69	22.88	24.88	27.34	49.51	53.38	56.90	61.16

Prune if information gain is not significantly > 0

- Evaluate Candidate split to decide if the resulting information gain is significantly greater than zero as determined using a suitable hypothesis testing method at a desired significance level



$$n_1 = 50, n_2 = 50, N = n_1 + n_2 = 100$$

$$n_{1L} = 50, n_{2L} = 0, n_L = 50, \ p = \frac{n_L}{N} = 0.5$$

$$n_{1e} = pn_1 = 25, n_{2e} = pn_2 = 25$$

$$\chi^2 = \frac{(n_{1L} - n_{1e})^2}{n_{1e}} + \frac{(n_{2L} - n_{2e})^2}{n_{2e}} = 25 + 25 = 50$$

This split is significantly better than random with confidence > 99% because $\chi^2 > 6.63$

Minimizing over fitting – Post pruning

- Grow full tree, then prune
- Minimize *size* (*tree*) + *size* (*exceptions* (*tree*))

Reduced error pruning

Do until further pruning is harmful:

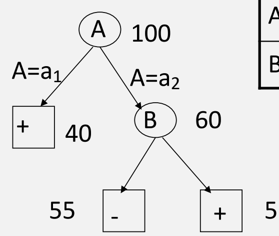
- Evaluate impact on validation set of pruning each candidate node
- Greedily select a node which most improves the performance on the validation set when the sub tree rooted at that node is pruned

Potential Drawback

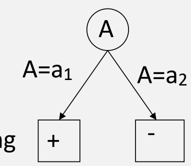
- holding back the validation set limits the amount of training data available
- not desirable when data set is small

Reduced error pruning – Example

Node	Accuracy gain on the validation set by pruning
A	-20%
B	+10%

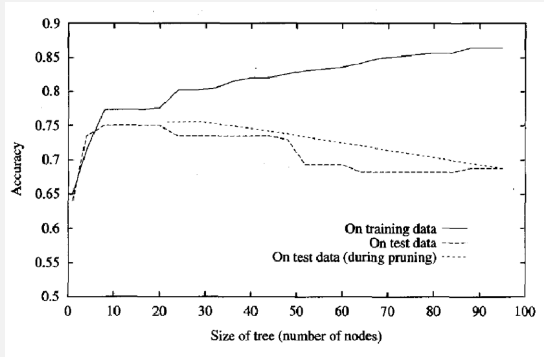


Before Pruning



After Pruning

Reduced error pruning



Classification of samples using a decision tree

- Unique classification – possible when each leaf has zero entropy and there are no missing attribute values
- Most likely classification – based on distribution of classes at a node when there are no missing attribute values
- Probabilistic classification – based on distribution of classes at a node when there are no missing attribute values

Two-way versus multi-way splits

- Entropy criterion favors many-valued attributes
- Pathological behavior – what if in a medical diagnosis data set, social security number is one of the candidate attributes?

Solutions

Only two-way splits (CART) $A = value \text{ versus } A = \neg value$

Entropy ratio (C4.5)

$$EntropyRatio(S \mid A) \equiv \frac{Entropy(S \mid A)}{SplitEntropy(S \mid A)}$$

$$SplitEntropy(S \mid A) \equiv -\sum_{i=1}^{|Values(A)|} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Incorporating Attribute costs

- Not all attribute measurements are equally costly or risky
- In Medical diagnosis
  - Blood-Test has cost $150
  - Exploratory-Surgery may have a cost of $3000
- Goal: Learn a Decision Tree Classifier which minimizes cost of classification

$$\frac{\text{Info-gain}^2(S,A)}{\text{Cost}(A)}$$

Dealing with Missing Attribute Values (Solution 1)

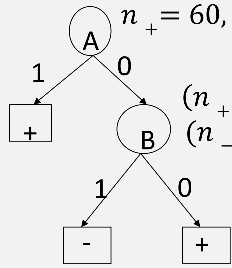
- Sometimes, the fact that an attribute value is missing might itself be informative –
- Missing blood sugar level might imply that the physician had reason not to measure it
- Introduce a new value (one per attribute) to denote a missing value
- Decision tree construction and use of tree for classification proceeds as before

Dealing with Missing Attribute Values

- During decision tree construction
  - Generate several fractionally weighted training examples based on the distribution of values for the corresponding attribute at the node
- During use of tree for classification
  - Generate multiple *instances* by assigning candidate values for the missing attribute based on the distribution of samples at the node
  - Sort each such sample through the tree to generate candidate labels and assign the most probable class label or probabilistically assign class label

Dealing with Missing Attribute Values

$n_+ = 60, n_- = 40$



$(n_+ | A = 1) = 50$

$(n_+ | A = 0) = 10;$
$(n_- | A = 0) = 40$

$(n_- | A = 0, B = 1) = 40$

$(n_+ | A = 0, B = 0) = 10$

- Suppose B is missing
- Assume B=1 with probability 40/50
- Assume B=0 with probability 10/50
- Choose the most likely class over the two options

Handling different types of attribute values

Types of attributes

- Nominal – values are names (as above)
- Ordinal – values are ordered
- Cardinal (Numeric) – values are numbers (hence ordered)

....

Handling numeric attributes

Attribute T	40	48	50	54	60	70
Class	N	N	Y	Y	Y	N

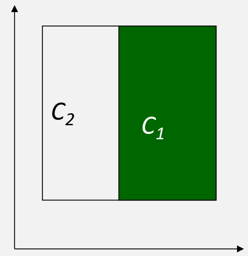
Candidate splits $T > \frac{(48+50)}{2}?$ $T > \frac{(60+70)}{2}?$

$$E(S \mid T > 49\,?) = \frac{2}{6}(0) + \frac{4}{6}\left(-\left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right) - \left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right)\right)$$

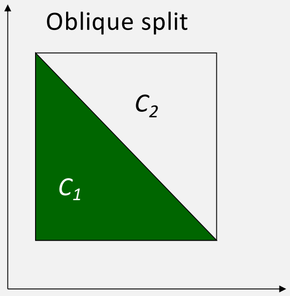
- Sort instances by value of numeric attribute under consideration
- For each attribute, find the test which yields the lowest entropy
- Greedily choose the best test across all attributes

Handling numeric attributes

Axis-parallel split



Oblique split



- Oblique splits cannot be realized by univariate tests
- We can train any binary classifier, e.g., SVM, to perform the split

Incorporating class-dependent misclassification costs

- Not all misclassifications are equally costly
- An occasional false alarm about a nuclear power plant meltdown is less costly than the failure to alert when there is a chance of a meltdown
- Use weighted Gini Impurity in place of entropy

$$Impurity(S) = \sum_{ij} \lambda_{ij} \left(\frac{|S_i|}{|S|} \right) \left(\frac{|S_j|}{|S|} \right)$$

λ_{ij} is the cost of wrongly assigning an instance belonging to class *i* to class *j*

Forests are better than trees

- Suppose a single decision tree does not perform well
  - Why?
    - Overfitting
    - Limited Expressive power
- But, it is super fast
- Can we learn multiple trees and combine them?
- Yes – Random forests

Random Forests

Outline

Bagging

Random Forests

Boosting

Random Forests

- We build a number of decision trees on bootstrapped training samples
- Each time a split in a tree is considered, a random sample of *m* features is chosen as split candidates from the full set of *p* predictors.

Random Forests Algorithm

For b = 1 to B:

(a) Draw a bootstrap sample Z of size *N* from the training data.

(b) Grow a random-forest tree from the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until a fixed depth is reached or the size of the data gets smaller than n_{min}.

i. Select ***m*** variables at random from the *p* variables.

ii. Pick the best variable/split-point among the *m*.

iii. Split the node into two daughter nodes.

Output the ensemble of trees.

To make a prediction at a new point *x* we do:

For regression: average the results

For classification: majority vote

Random Forests Tuning

Recommended defaults

- For classification, the default value for *m* is $\sqrt{p}$ and the minimum size of the data is one.
- For regression, the default value for *m* is *p/3* and the minimum node size is five.

In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters (optimized using cross-validation on a validation data set)

Example

- 4,718 genes measured on tissue samples from 349 patients.
- Each gene has different expression
- Each of the patient samples has a qualitative label with 15 different levels: either normal or 1 of 14 different types of cancer.

Use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.

There are around 20,000 genes in humans , and individual genes
23 chromosomes (2 x)

Random Forests Issues

When the number of variables is large, but the fraction of relevant variables is small, random forests are likely to perform poorly when *m* is small

Why?

Because:

At each split the chance can be small that the relevant variables will be selected

For example, with 3 relevant and 100 not so relevant variables the probability of any of the relevant variables being selected at any split is ~0.25

Can RF overfit?

Random forests "cannot overfit" the data wrt to number of trees.

Why?

The number of trees, *B* does not mean increase in the flexibility of the model

Summary

- Decision trees offer an attractive approach when the information necessary for classification consists of nonlinear interactions between a small number of informative variables
- Random forests provide a simple way to combine multiple decision trees to improve predictive performance in settings where there is no single decision tree that achieves the desired performance