INM431 Machine Learning

Artur S. d'Avila Garcez
a.garcez@city.ac.uk
http://www.staff.city.ac.uk/~aag/

Content

Information Theory applied to Decision Trees

Random Forests for regression and classification

Decision tree (recall)

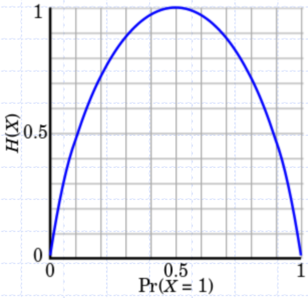
- Decision tree learning performs hill-climbing search (with information gain as a heuristic) over the space of all decision trees, with no backtracking
- Decision tree learning might generate non-minimal trees (local minima)
- Bias: shorter trees are preferred; trees that place attributes with higher information gain closest to the root are preferred

Entropy H(X)

A measure of the amount of *uncertainty* associated with the value of discrete random variable X

Entropy is maximized when all the messages (data) in the message space are equiprobable $p(x) = 1/n$
i.e. most unpredictable

In this case $H(X) = \log n$



Information Theory

- How to send a message reliably through a noisy channel? Communication theory
- Source – Encoder – Channel – Decoder – Recipiente
- Entropy gives shortest possible average length of a lossless encoding
- Claude Shannon's MSc dissertation: https://dspace.mit.edu/handle/1721.1/11173

Information Content

Suppose you want to find an answer to some question, whose n possible answers v_i have probabilities $P(v_i)$.

Then the information content/entropy of the answer is:

$$H(P(v_1),\ldots,P(v_n)) = -\sum_{i=1,\ldots,n} P(v_i) \log_2 P(v_i)$$

where $-\log_2 P(v_i)$ is the information content of answer v_i (measured in bits)

Example (flipping a coin): what will the outcome be?

v_1 = heads, v_2 = tails

Coin is fair: P(heads)=P(tails)=1/2

$$H(1/2,1/2) = -1/2 \log_2 1/2 - 1/2 \log_2 1/2 = 1 \text{ bit}$$

Coin is biased: P(heads) = 0, P(tails) = 1

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

Information theory for decision tree learning

- You want to find an answer to the question: given a new input **x**, which is its correct classification?
- Before any attribute has been tested for **x**, the information content/entropy of the actual answer is:

  $H(p/(p+n),n/(p+n)) =$

  $- p/(p+n) \log_2 p/(p+n) - n/(p+n) \log_2 n/(p+n)$

  where the training set E has p positive and n negative examples
- Testing an attribute will give some useful info:

  An attribute A which can take *v* values divides E into subsets $E_1 \ldots E_v$ each E_j having p_j positive and n_j negative examples

Information theory for decision tree learning (cont.)

If we choose attribute A, then $H(p_j/(p_j+n_j),n_j/(p_j+n_j))$ bits will be needed to decide which is the correct classification for **x**, if **x** follows the branch with value j

- The probability of **x** having value j for A is:

  $(p_j+ n_j)/(p+n)$
- After testing A, on average one will need:

  $After(A)=\Sigma_{j=1,..,v} (p_j+ n_j)/(p+n) H(p_j/(p_j+n_j),n_j/(p_j+n_j))$

  bits of information to classify **x**.
- The information gain from choosing A is then given by:

  $H(p/(p+n),n/(p+n)) - After(A)$

Thus, at step 5 in the decision tree learning algorithm:
***Choose** attribute A giving the highest information gain!*

Random Forests

- An ensemble of decision trees used for classification or regression
- Decision trees can overfit their training data and cannot handle noise
- RFs average multiple decision trees, trained on different parts of the training set
- This comes at the expense of some loss of interpretability, but generally boosts performance greatly.
- RFs provide state-of-the-art performance in many computer vision applications

Random Forests (cont.)

- Use bootstrapping (or bagging) of the training data:
- For $b = 1,\ldots,B$ select with replacement n examples $(X\ Y)$; call these (X_b, Y_b)
- Train a decision tree f_b on (X_b, Y_b)
- Average the predictions (for regression) or take the majority vote (for classification)

Random Forest Training

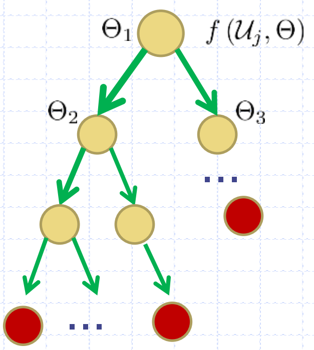
Split node:

- Splitting function:

$$f(\mathcal{U}_j, \Theta) = \begin{cases} Left & \text{if } \mathcal{U}_j(w) < \tau, \\ Right & \text{otherwise.} \end{cases}$$

- Learned parameter:

$$\Theta = (w, \tau)$$



U_j = data set at node j
w = a randomly chosen dimension, i.e. attribute
τ = split threshold
f splits U_j into two sets U_j^{left} and U_j^{right}

We want to find good values for θ:
We normally select attributes randomly and use information gain for the choice of split thresholds

Random Forest (cont.)

Information gain:

- $Q(\mathcal{U}_j, \Theta) = H(\mathcal{U}_j) - \sum_{b \in \{Left, Right\}} \frac{|\mathcal{U}_j^b|}{|\mathcal{U}_j|} H(\mathcal{U}_j^b)$

We want to maximise $Q(U_j, \theta)$:
Sample values for τ in the range (min(w), max(w))
Calculate $Q(U_j, \theta)$ and select the best τ
Stop splitting when there are fewer than n (e.g. n=20) examples in U_j
Typically, consider a grid search on possible values for τ and n

Leaf node:

- Leaf node probability:

$$p_t(r|\mathbf{u})$$

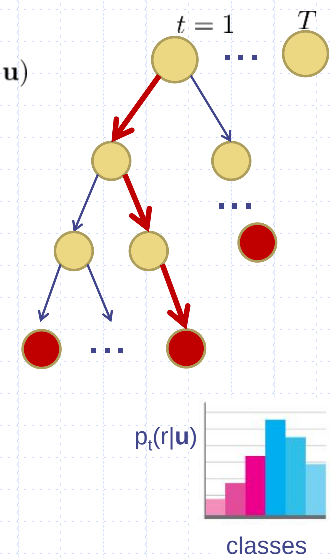
Classification: count the number of training examples in each class at each leaf node
Regression: calculate mean and variance for the label of the training examples in each leaf node, where r = label and **u** = data

Random Forest Prediction (1)

- Posterior probability: $p(r|\mathbf{u}) = \frac{1}{T}\sum_{t=1}^{T} p_t(r|\mathbf{u})$

Each example in the test/validation data ends up in a leaf and is classified into the class with the highest score according to the histogram obtained from the training data

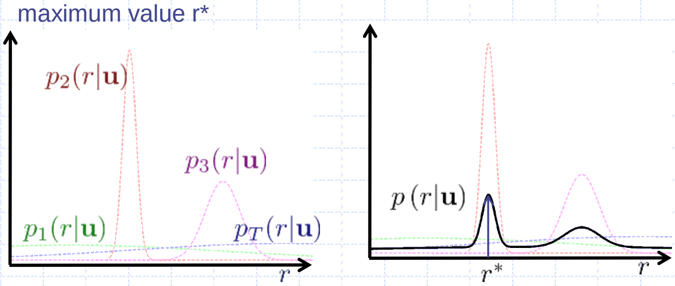
- Prediction: $r^* = \arg\max_r p(r|\mathbf{u})$

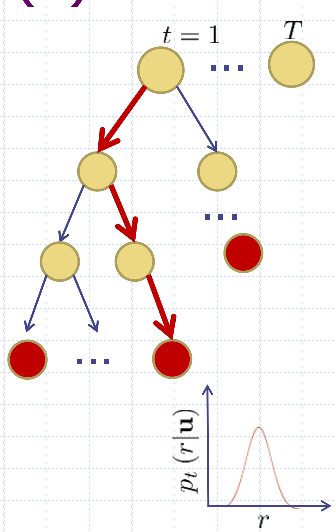


Random Forest Prediction (2)

- Posterior probability: $p(r|\mathbf{u}) = \frac{1}{T}\sum_{t=1}^{T} p_t(r|\mathbf{u})$
- Prediction: $r^* = \arg\max_r p(r|\mathbf{u})$

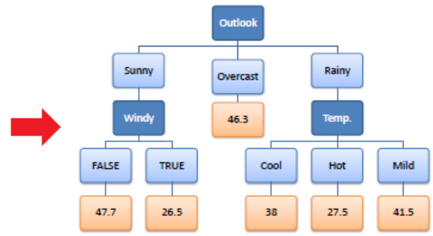
Regression: calculate mean and variance of training data in each leaf
Each example from the test/validation data ends up in a leaf node associated with a Gaussian obtained from the training data
Aggregate (add) the Gaussians into p(r/u) and take the maximum value r*

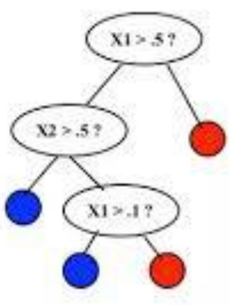


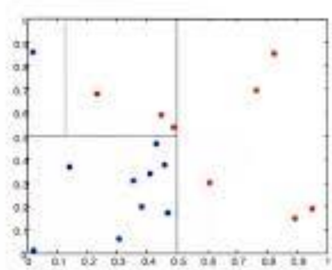


Examples

Predictors				Target
Outlook	Temp	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30







Summary

- Decision trees: very successful attribute-value baseline ML method (non-probabilistic)
- Equivalent to propositional logic; relational (FOL) ML methods exist, but are not as popular
- Information gain can help improve node splitting choice towards smaller trees (Ockam's razor)
- Random Forests: ensemble ML method with many decision trees using bagging (probabilistic)
- RFs are less interpretable than DTs, but applicable to either regression or classification problems