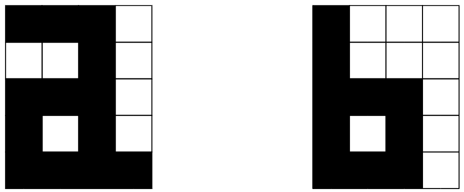
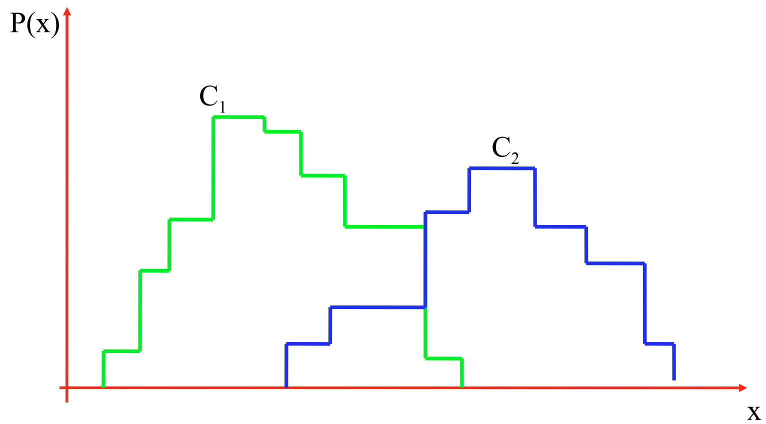
Bayesian Classification

Simple Classification

- Based on images, perform letter classification
- Get lots of images of each letter
- Train a classifier



Feature Histograms



Probability

- We are dealing with probabilities
- We make the histogram from our examples
- Joint probability $P(C_i, X_j)$
- Conditional probability $P(X_j|C_i)$

Prior Knowledge

- Suppose we know that Class 1 is more likely that Class 2
  - Distribution of letters in English text
- We should be able to include this information into the classifier: $P(C_1)$

$$P(C_i, X_j) = P(C_i|X_j)P(X_j)$$
$$P(C_i, X_j) = P(X_j|C_i)P(C_i)$$

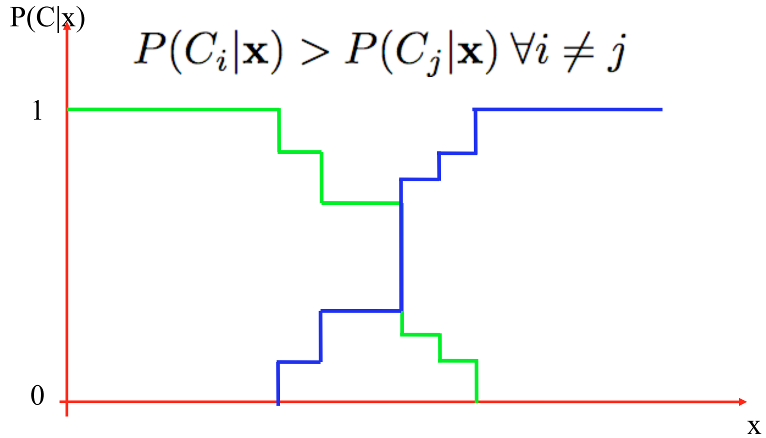
Bayes' Rule

$$P(C_i|X_j) = \frac{P(X_j|C_i)P(C_i)}{P(X_j)}$$

- Most important equation in machine learning
- Combine things that are easy to find to get useful answers
- Denominator normalises it so probabilities sum to 1

Posterior Probability

$$P(C_i|\mathbf{x}) > P(C_j|\mathbf{x})\ \forall i \neq j$$



Classification Process

- Inference
  - Compute posterior probabilities from data
- Make decisions
  - Use posterior probabilities to classify new data
- We do this here by maximising the posterior probability

Bayes Classifier

$$P(C_i|\mathbf{x}) > P(C_j|\mathbf{x}) \; \forall i \neq j$$

- Need to compute $P(\mathbf{x}|C_i)$
- Often have high dimensional feature vectors

$$P(X_j^1 = a_1, X_j^2 = a_2, \ldots, X_j^n = a_n | C_i)$$

- Curse of dimensionality applies - need lots and lots of data

Naïve Bayes Classifier

- What if we assume that the features are independent?
- Then can just compute:

$$P(X_j^1 = a_1|C_i) \times P(X_j^2 = a_2|C_i) \times \ldots \times P(X_j^n = a_n|C_i)$$
$$= \prod_k P(X_j^k = a_k|C_i)$$

- Gross simplification
- Surprisingly effective

What to Maximise

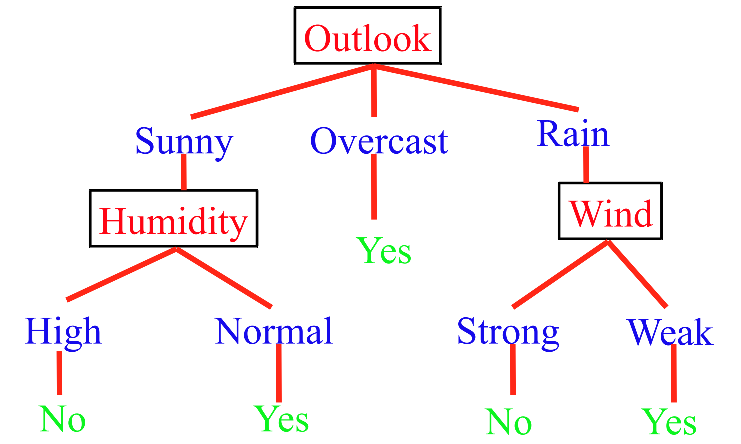
- We maximised posterior probability
- There are other choices
  - Maximise likelihood
  - Minimise risk
- Medical data - better to think somebody has a disease than not if unsure
- Loss matrix

Decision Trees

Decision Trees

- Split classification down into a series of choices about features in turn
- Lay them out in a tree
- Progress down the tree to the leaves

Example



Rules and Decision Trees

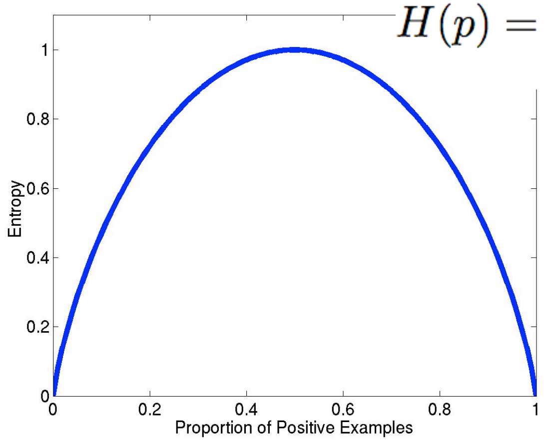
- Can turn the tree into a set of rules:
  - (outlook = sunny & humidity = normal) |
    (outlook = overcast) |
    (outlook = rain & wind = weak)
- How do we generate the trees?
  - Need to choose features
  - Need to choose order of features

Entropy

- Tells us how much extra information we get from knowing p_i
- Measures the amount in impurity in the set of features
- Makes sense to pick the features that provides the most information

Entropy

$$H(p) = \sum_i p_i \log_2 p_i$$



Information Gain

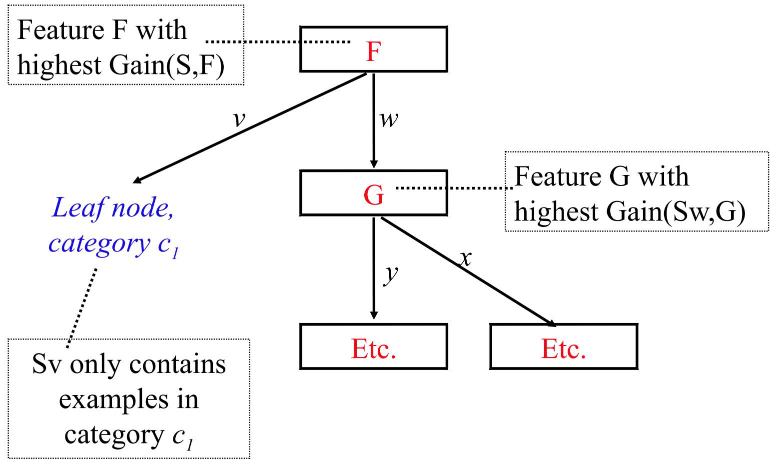
$$\text{Gain}(S, F) = \text{Entropy}(S) - \sum_{f \in values(F)} \frac{|S_f|}{|S|} \text{Entropy}(S_f)$$

- Choose the feature that provides the highest information gain over all examples
- That is all there is to ID3:
  - At each stage, pick the feature with the highest information gain

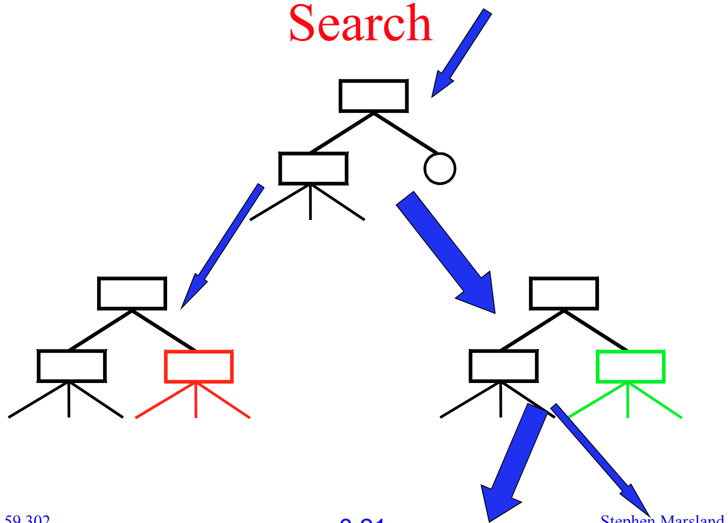
ID3 (Quinlan)

- Search over all possible trees
  - Greedy search - no backtracking
  - Susceptible to local minima
  - Uses all features - no pruning
- Can deal with noise
  - Labels are most common value of examples

ID3



Search

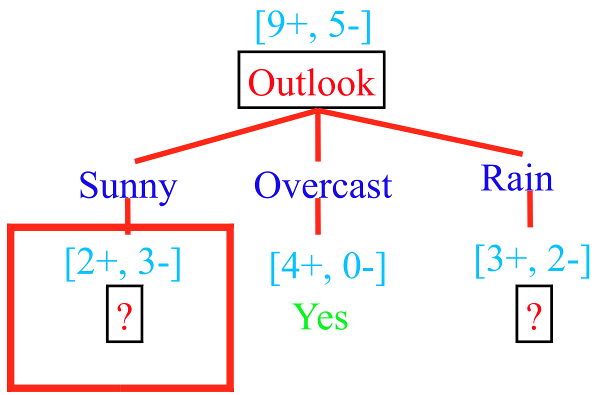


Day	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Example

- Values(Wind) = Weak, Strong
- S = [9+, 5-]
- S(Weak) <- [6+, 2-]
- S(Strong) <- [3+, 3-]
  - Gain(S,Wind) = Entropy(S) - (8/14) Entropy(S(Weak)) - (6/14)Entropy(S(Strong))
  - = 0.94 - (8/14)0.811 - (6/14)1.00

Example



Inductive Bias

- How does the algorithm generalise from the training examples?
  - Choose features with highest information gain
  - Minimise amount of information is left
  - Bias towards shorter trees
  - Occam's Razor (KISS)
  - Put most useful features near root

Missing Data

- Suppose that one feature has no value
- Can miss out that node and carry on down the tree, following all paths out of that node
- Can therefore still get a classification
- Virtually impossible with neural networks

C4.5

- Improved version of ID3, also by Quinlan
- Use a validation set to avoid overfitting
  - Could just stop choosing features (early stopping)
- Better results from post-pruning
  - Make whole tree
  - Chop off some parts of tree afterwards

Post-Pruning

- Run over tree
- Prune each node by replacing subtree below with a leaf
- Evaluate error and keep if error same or better

Rule Post-Pruning

- Turn tree into set of if-then rules
- Remove preconditions from each rule in turn, and check accuracy
- Sort rules according to accuracy
- Rules are easy to read

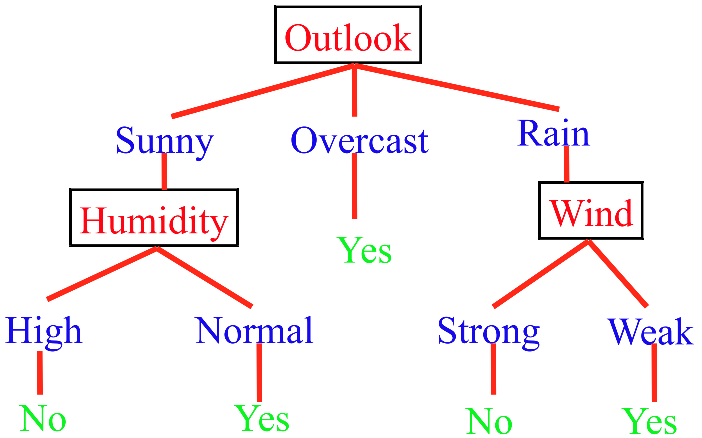
Rule Post-Pruning

- IF ((outlook = sunny) & (humidity = high))
- THEN playTennis = no
- Remove preconditions:
  - Consider IF (outlook = sunny)
  - And IF (humidity = high)
  - Test accuracy
  - If one of them is better, try removing both

Comparison Between Decision Trees and Naïve Bayes Classifier

Day	Outlook	Temp	Humid	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

ID3 Decision Tree



Test Case

- Outlook = Sunny
- Temperature = Cool
- Humidity = High
- Wind = Strong

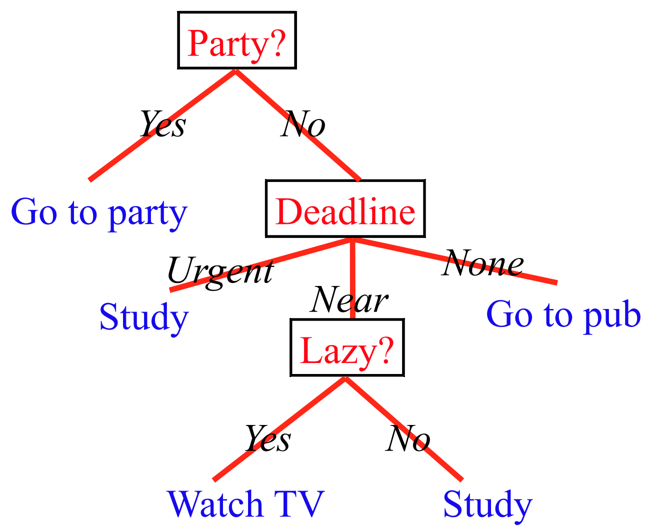
Naïve Bayes

- P(yes)*P(Outlook=Sunny|yes)*P(Temperature=Cool|yes)*P(Humidity=High|yes)*P(Wind=Strong|yes)
- Similar for no
- Count all the probabilities from the table

Naïve Bayes

- Yes: 0.0053
- No: 0.0206
- So solution is no
- Conditional probability is:

$$\frac{0.0206}{0.0206 + 0.0053} = 0.795$$


Party?
Yes
No
Go to party
Deadline
Urgent
Near
None
Study
Go to pub
Lazy?
Yes
No
Watch TV
Study