**Artificial Intelligence**

**Session 6**

1. Classification: why is it hard?
   1. Classes not directly observable
      1. classes represent high-level information (identity, behaviour...) but only low-level attributes (pixel values, # clicks on web page...) observed
      2. human brain is good at making sense of data, but don’t know how
   2. Objects have many attributes
      1. each attribute individually has little correlation with classes
         1. need lots of attributes
      2. relationship between attributes and classes has to be learnt
   3. There are many examples
      1. if not, it could be done by hand #
      2. learning algorithms may have high computational cost
2. Machine learning for classification
   1. Classifier: takes features describing object, returns object’s class
      1. can label any possible example
   2. Machine learning approach
      1. learn classifier from training set, then apply it to problem examples
         1. supervised learning: classes of training examples are known
         2. training set is much smaller than the problem set
   3. Learning step may be costly but is performed only once
   4. How can we ensure classifier will work on the problem data?
      1. cannot be certain
      2. Occam’s razor: simplest solution is often the right one
3. Classification: geometrical view
   1. Examples lie in the attribute space
      1. each attribute is a dimension
      2. each example is a point
      3. in these slides, classes are represented by colours
   2. A classifier embodies a partition of the attribute space
      1. the whole space is divided into non-overlapping pieces (subspaces)
   3. Learning a classifier is partitioning training set into homogeneous subsets
      1. each subspace contains training examples of exactly one class
   4. Applying classifier is just checking in which subspace a point lies
4. Classification: geometrical view
   1. Consider a graph with attributes as dimensions and examples as points.   
        
      Consider two sets of points separated linearly on the graph.  
        
      Classification on this graph can be done by:
      1. Colour the whole space
      2. Colour only the subspaces
      3. Colour the points
5. Structure of the classifier
   1. Subspaces of classifier do not have to be connected
   2. Border can be much more complex than single line
   3. Generally, types of borders allowed are set by algorithm
      1. need expert knowledge of problem
      2. no single algorithm solves every problem
6. Attribute Types
   1. Symbolic attributes
      1. finite, discrete valued: e.g. gender, marital status,...
      2. no notion of order or distance between values, only Boolean comparison
      3. can be enumerated
      4. use logic to combine different attributes after testing them
   2. Numeric attributes
      1. real or integer valued: e.g. grades, temperature, pixels, audio samples, ...
      2. can be measured
      3. can use algebra to combine different attributes before testing them
         1. can create new attributes
7. Attribute selection/engineering
   1. Learning is hard
      1. learning algorithms usually costly & suboptimal
   2. Learning is easier if one starts with “good” attributes
      1. fewer attributes decrease cost of learning
      2. good initial guess helps finding the best classifier
   3. What is a good attribute?
      1. easiest case (trivial): classes themselves are attributes
      2. hardest case (insoluble): no attributes at all
      3. an attribute that carries a lot of “information”
      4. an attribute whose “information” is related to the class
8. Greedy Algorithm ID3 ((Iterative Dichotomiser 3)  
     
   A tree with left node 3 and two sub nodes 99, 8. Right node 12 with sub nodes 5, 6.  
   1. Goal: reach largest sum
   2. Greedy algorithm will take what appears to be optimal choice at each step
      1. no backtracking
   3. So it will choose 12 instead of 3 at second step
   4. Will not reach best solution, which contains 99.
9. ID3 takes training set & builds decision tree so that each leaf is homogeneous (represents single class label)
   1. builds tree from root to leaves  
        
      ID3(Training\_Set) =
      1. if all points in Training\_Set have same class C
         1. return Leaf(C);
      2. elseif no questions remain for Training\_Set
         1. return Leaf(MajorityClass(Training\_Set));
      3. else Question = Find\_Best\_Question(Training\_Set);
         1. [Set\_1,...,Set\_n] = Split(Training\_set, Question);
         2. return Tree(Question, ID3(Set\_1),...,ID3(Set\_n));
   2. ID3 restricts itself to questions that test only one attribute
      1. less questions to choose from
      2. geometrically, all borders orthogonal to one axis
      3. works well for symbolic attributes
         1. symbolic attributes can be enumerated
         2. symbolic attributes cannot be combined
      4. Questions chosen so that resulting subsets are as well-classified as possible
         1. entropy of class variable c within set S measures how classified it is
         2. probability of item belonging to c is proportion of items belonging to c in training set S  
              
            H(s) = -(summation of p\_{c} \* log\_2 p\_{c}), where p\_{c} = P(x contained within c given x contained within S)
   3. Finding the best question
      1. Entropy of partition is sum of entropies of subsets weighted by their size
      2. Entropy also provides stopping criterion: leaf is node with entropy 0
      3. Information gain defined as difference between total entropies before and after partitioning set S into subsets Si
         1. Its also mutual information between attribute & class, conditioned to previous splits  
              
            G(S, Q) = H(S) – (sum of (S\_i / S) \* (S\_i))
      4. Compute G for all possible questions Q
      5. Select Q with largest G(S,Q)
   4. Which questions to ask?
      1. Symbolic attributes (gender, hair colour,...)
         1. only finite number of possible values for all data
         2. one branch for each different value
      2. Numerical attributes (temperature, pixel value,...)
         1. infinite range even if only finite number of values present in training set
         2. so choose threshold t and ask: is value < t ?
      3. Threshold choice
         1. information gain criterion still holds
         2. best threshold must be between two training examples of different classes
         3. try all such candidates, using midpoint between opposite training examples
   5. Strong and weak points of ID3
      1. Each question only checks one feature:
         1. works well with attributes of different types (e.g. size and gender)
         2. designs rectangular regions only, even if it does not fit data
      2. Generates shortest possible tree:
         1. fast classification after learning
         2. helps to generalise the model
         3. helps to understand the generated rules
      3. Training is expensive:
         1. try all possible questions at each node
         2. for each question, compute the answer with each data point of the subset
         3. continuous range attributes generate many different questions
   6. A word on overfitting
      1. Noise blurs boundaries between classes
      2. Sampling (choice of training examples) can alter perception of problem
         1. selection bias
         2. rare events
      3. General solution: more data! (not always possible)
      4. Overfitting
         1. learning a tree that models properties specific to the training set: this is bad!
            1. the tree works well on the training set but not on other data
            2. perfect classification of the training set is usually too much
10. Reduced-error pruning
    1. Split training set into two (usually unequal) parts:
       1. one for learning
       2. one for pruning
    2. Goal: learn tree that classifies pruning set well though not trained on it
    3. First build tree with ID3 on training set (learning subset)
    4. Then greedily prune nodes to improve classification of pruning set
    5. What if we don’t have enough data to make a pruning set?
11. Rule pruning
    1. Tree can be written as set of logical rules
    2. each path is a single conjunction (“and”) of tests ‣ there is a disjunction (“or”) between different paths leading to the same class
12. Rule post-pruning
    1. First convert tree into rules
    2. Then greedily prune rules by removing conditions
    3. Only prune if precision of rule does not decrease
       1. Precision = fraction of points of correct class among points picked by rule
    4. Pruning is done on training set directly
       1. no need to split the set in two
       2. more used in practice
13. Summary
    1. Information as a measurable quantity
       1. information as structure in data
       2. quantified by entropy
    2. ID3: method of using information to decide on structure
       1. build decision trees which are simple and interpretable by humans
    3. Avoid overfitting
       1. pruning to re-generalise
14. Information theory (Shannon 1948)
    1. Communication problem: how to compress signal, i.e. stream of independent random variables X?
       1. e.g. signal = succession of coin tosses
    2. Quantity of information in signal = minimum bit rate needed to transmit it, i.e. average number of bits per symbol.
    3. It is a function of probability distribution of the signal
       1. distribution represents what is known before transmission, does not depend on actual samples
       2. bit rate chosen before transmission
    4. more uncertain the distribution → more information in the content
       1. if coin is double-headed, then there is nothing to transmit
    5. Probabilities are multiplicative, information is additive
       1. if ones tosses coin n times, there are 2^n possible outcomes, but can all be written using n bits only
       2. information content depends on logarithm of probability distribution
    6. Impossible events should not matter
       1. adding “coin can rest on its side with probability 0” should not change amount of information
       2. rare events matter a little; common events matter a lot
       3. weight logarithms by probability px of each event x from set X of all possible events
    7. Entropy:  
         
       H(s) = - (summation of p\_{x} \* log\_2 p\_{x})
    8. Example: Coin toss:
       1. Two possible values (X = {Heads, Tails})
          1. respective frequencies p\_Heads = p and p\_Tails = 1 – p  
               
             H(s) = - p log\_2 p – (1-p) log\_2 (1-p)|  
               
             A inverted parabola shaped graph from 0 to 1 on both the x and y axis.
       2. How can one coin-toss hold less than 1 bit of information when there are still 2 possible outcomes?
          1. Result is probabilistic: for any bit rate b > H(X) and any arbitrarily low error rate e > 0, there is an encoder with bit rate b and error rate e
       3. What does non-integer quantity of information mean?
          1. It is an average rate for a stream of independent samples
          2. The stream is encoded by grouping samples together into words (sequences of flips)
          3. H(X) = 0.4 means that words of 10 samples can be encoded using 4 bits only
       4. How does it work?
          1. If H(X) is less than maximal, then there are rare events
          2. Conjunctions of rare events become rarer and rarer as word length increases
          3. At some point they become negligible and do not need to be encoded anymore
    9. Mutual Information:
       1. Objective: measure how much of the information in two random variables is shared
          1. 0 if the variables are independent
          2. maximal if one variable is a function of the other.
          3. Mutual information: I(X;Y) = H(X) + H(Y) - H(X,Y)  
               
             Two overlapping spaces H(X) and H(Y). The intersecting space is I(X;Y), while the whole space is H(X,Y)
15. Computing the entropy numerically
    1. log(0) is undefined and will give an error if you compute it
       1. as the limit of x reaches 0, log(x) = -infinity for x > 0, but x\*log(x) = 0
       2. test if probability is 0, do not compute explicitly in that case
    2. How to compute log2:
       1. many languages have log2 function (MATLAB, C, C++,...)
       2. ... but some do not (Java,...)
       3. use the fact that for any bases a and b: log\_b x = (log\_a x) / (log\_a b)
       4. in our case b=2
       5. a is usually e (2.71828…, called the natural logarithm) or 10 (does it matter?)
16. Back to attribute selection
    1. Finding attributes with high entropy:
    2. start with a compressing transform (e.g. time/frequency for sounds, wavelet or DCT for images, movement vectors for video,…) to reduce redundancy
    3. for numeric attributes, variance can be good surrogate for entropy, and easier to compute
    4. Finding features with high mutual information with the classes:
       1. expert knowledge may be available
    5. Common sense works!
       1. e.g., don’t classify text documents based on frequency of “a”, “the”, “and”, ...
17. Common classifiers:
    1. Bayesian classifiers (lecture topic)
       1. Maximum Likelihood learning: for each class C, learn probability distribution p(. given C) that maximizes product sum of p(m given C) for all m in training data belonging to C
       2. classification: for a point m, find the class C that maximises p(C given m)
          1. using Bayes’ rule
          2. scales linearly with the number of classes
       3. Support vector machines (SVMs)
          1. linear classifiers: boundaries are hyperplanes
          2. Maximal Margin learning: learn the hyperplane that is the furthest away from making a mistake
             1. maximise the distance to the closest (=worst) points of each class (the support vectors)
          3. mainly for 2 classes (binary classification)
          4. Neural networks
18. Decision trees
    1. Tool to represent decision making
    2. Decompose complex problem into sequence of simple questions
    3. Which question to ask next depends on previous answers
       1. represented as a tree
    4. Tree structure:
       1. leaves: decisions
       2. nodes: questions
       3. branches: possible answers to the parent question
19. Decision trees and classification
    1. Classification is decision problem: which label applies to data point?
    2. Classifiers represented as decision tree
       1. decision tree also partitions attribute space
       2. question: on which side of given border is point we’re interested in?
       3. several trees can represent same classifier
    3. Learning problem: which questions to ask?
    4. If one asks enough questions, one will eventually classify training set
       1. 1 question = 1 new split of the space
       2. with enough splits, one can eventually separate each training point from all others
    5. but would that classifier generalise to new data?
    6. classifier should be general
       1. classifier should be fast to apply after learning ‣ build classifier that needs to ask few questions before taking decision (shallow, bushy tree)