**Search Algorithms**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Complete | Optimal | Time | Space | Notes |
| BFS | Yes | No | O(bd) | O(bd) | Space biggest problem |
| UCS | Yes | Yes |  |  | Chooses cheapest total path cost |
| DFS | No (inf d) | No | O(bm) | O(bm) | Can delete branches after expansion |
| IDS | Yes | Yes | O(bd) | O(bd) |  |
| Greedy | Yes | No | O(bm) | O(bm) | Choses cheapest heuristic |
| A\* | Yes | Yes (h adm) | O(bd) | Exp | min (path + heuristic) |

Heuristic H admissible if H(n) < true cost for all nodes => A\* complete and optimal

Heuristic h consistent if h(p) ≤ cost(p,c) + h(c) (f is non-decreasing along any path) (implies ^)

Hill climbing: keep only a single state in memory if neighbor is better move to it otherwise stop

Beam: like HC but uses k random states not 1. Generate all successors of k states and chose best k.

Genetics: choose best options for reproduction fitness determined by function

Supervised learning – target output is known for training

Unsupervised – no known target (clustering)

Holdout Procedure – split into training and test and sometimes validation

**Algorithms:**

0R: predict based of majority class

1R: predict based of attribute that classifies training examples with smallest error

**Naïve Bayes**: assumes attributes are conditionally independent and equally important

Need to multiply by P(H) and P(E) cancels so can ignore

For numeric columns use normal distribution

A picture containing font, handwriting, line, diagram

Description automatically generatedA picture containing font, number, symbol, typography

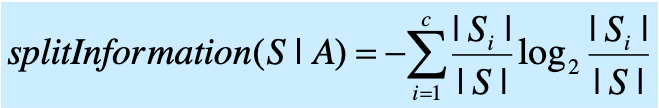
Description automatically generated A picture containing font, handwriting, line, number

Description automatically generated

Laplace Correction: if one p is 0 for all P(Ei | X) add 1 to numerator and |X| to denom

**Decision Tree:**

H(s) = I(s) = information gain is reduction in entropy after split

A picture containing text, font, line, calligraphy

Description automatically generated

Prune to avoid overfitting

* Pre-pruning: stop growing tree before it classifies everything perfectly
* Post-pruning: fully grow then prune – widely used
  + Subtree replacement – replace subtree with majority leaf class
  + Subtree-raising – more complex restricted to most popular branch
  + Rule – convert full tree to rules and remove non harmful pre condition

**Perceptron:**

a = step(w . p + b), t = true value

Tentative learning rule: , bnew = bold + (t - a)

**Backpropagation**

Propagate each training ex and find output. Check MSE and update weights to reduce the error until it is below threshold. Backpropagate back to earlier levels of neurons. Can speed up with batching (stochastic gradient descent) and using linear filters to average updates letting a larger learning rate be used.

**Autoencoders**

Kind of NN that predicts its own input. Can be used to pretrain other NN with stacked autoencoders for each layer. Useful for leaning features from unlabelled data. Particularly sensory data applications

**Convolutional NN**

Designed to recognise visual patterns directly from pixel images with minimal pre-processing.

Local connectivity – restrict connections so each neuron is only connected to small number of adjacent neurons (pixels)

Weight sharing – number of connections reduced by weight sharing. Neurons organised into groups which all apply the standard weight as the filter in a convolution. During forward pass single weight is used. For backward pass each weight is calculated and summed for final

Pooling – used with convolution layer and takes max value of set of neurons. Invariant to shifts in the inputs in its receptive field

Local contrast normalization – convolution and pooling layers can be stacked and then normalized. Allows for brightness invariance

Finally there is a smaller fully connected network. For multiple channels (rgb) filters are shared but weights are not. Can use dropout in fully connected layer to reduce overfitting.

**SVM**

Maximises margin of hyperplane separating support vectors. Uses kernel functions satisfying mercers theorem to transform into higher dimensional space so boundary is not necessarily linear. Kernel fns need to satisfy for some f f(u)•f(v). = (u•v)2. Max can be found using lagrange multipliers which are nonzero for support vectors. Then max decision boundary is

**Bagging**

Weighted error e is the sum of p of the incorrect rows. Probability modification m = e / (1-e). Multiply correct rows by m and normalise by dividing by the sum of all rows.

**Clustering**

Centroid = means, medoid = median

K means => k clusters

Nearest neighbour: iteratively add to existing cluster if threshold is met otherwise make new

Hierarchical: iteratively merge into new clusters. Single link = min distance complete = max