|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Group distributional robust optimisation: maximum group loss!= average loss. Minimise maximum group loss. Predicted label = sign(w. ϕ (x)). Score = **w**. ϕ (x) how confident are we in predicting +1? Margin = **w**. ϕ (x).y how correct we are. More features: makes learning harder because need to eval weight for each feature. Hypothesis class of quadratic predictors is superset of hypothesis class of linear predictors. | | | NNs build nonlinear predictors without feature extractors by hand You may not need O(d2) features to represent desired non-linear predictor = compact.  NNs can’t deal with zero gradients so use Activation functions still a problem when gradients are very small which makes optimising hard. | | | | | |
|  | |  | | | | | Definition of search problem   1. Sstart Start state 2. Action(s): possible actions from states 3. Cost(s,a): cost of the action from state s 4. Succ(s,a): successor state from state s given action a 5. IsEnd(s): reached end state? Don’t forget to terminate if pitfall! | |
| Approximation vs. estimation error trade off: increase hypothesis class means you can get closer to magical f\* but that increases estimation error as hypothesis class size increases.  How to control hypothesis class size to reduce estimation error?   1. Reduce dimensionality d (number of features) . Manually drop unhelpful features. Automatic feature selection: forward selection, boosting, L1 regularisation 2. L2 Regularisation: add a term to penalise the length of **w**. Note: SVMs are exactly hinge loss + L2 regularisation. 3. Early stopping | | | | |  | | | |
|  | | | | |  | | | |
|  | | | |  | | | | |
| **Laplace Smoothing:** against overfitting. add lambda to each possible value whether observed or not.  **Unsupervised with EM:** handle missing data setting where training data consists of partial assignments. breaks the problem into probabilistic inference + supervised algorithm.  Estep (freeze params, infer hidden): evaluate the expected joint distributions for the missing variable. Then evaluate the P(missing variable|evidence). You can use any inference algorithm for posterior inference in the E-step: gibbs, forward-backward, particle filtering. Output is q(h) which is    Mstep (freeze hidden, update params): add the weighted number of times that param was used and recount instances with halllucinated weight and then normalize to get probabilities: effectively applying maximum likelihood | | | |  | | | | |
|  | | |  | | | | | If ∀ then comes with implication ->  If ∃ then comes with conjunction or disjunction v  If you see a verb then it is a predicate  CNF. P→Q into ¬P v Q  De morgans’ law: ¬(¬want ∧ ¬ invite). (want v invite). flips ∀ into ∃ and vice versa  Resolution rule. Get rid of p and ¬P and join the two statements. It is a single inference rule that operates on CNFs.  Prove f is entailed KB via resolution  Equivalent to proving KB U ¬f. |
| expectiminimax is variation of minimax. It’s for 2-player stochastic games. Max aims to maximise expected utility (EU) while min minimise EU (two player dice games). Expectimax is only one player trying max EU. | | | | Minimax Recurrence | | | | |
|  | MCV, LCV, Arc consistency or forward checking    Backtrack: extend partial assignments, exact, exponential, approx., linear | | | | | **Beam search** extend partial assignments, approximate, linear. Exhaustive, K = inf is BFS so tie O(bn). Greedy K=1: DFS only one path for one partial assignment. K=1. Time = O(nb). n variables (depth), branching factor: b = domain len, beam size = K, Time = O(nKblogK)  **ICM**. Modify complete assignments, Returns value of assignments, approx., linear. Max weight assignment. Choose best value. Local optimum**.**  exploit locality structure of graphs: When evaluating possible re-assignments to Xi only need to consider the factors that depend on Xi. Not guaranteed to find optimal, when stuck then stuck forever!  **Gibbs sampling** (probabilistic) returns the normalised distribution. Stochastic exploration, marginal converge to correct answer but can take exponential time. You can make direct inferences about anything in the entire MN after running gibbs because we’ve eventually converged to the right answer (spring final 2021) | | |
|  | | | | |
| Particle filtering query for HMMs (**approx)**, linear with numParticles, extend partial. practical relies only on last timestep.Partial because each particle = guess of current state. Set that to be at least as big as the domain size. more particles always better.  Propose (transitions): given currParitialAssignments get transition probability because it’s fast.  Weight (emisions) old particles based on evidence: w(hi) = ∏P(ej|hi) for all emissions 1,…j at hi. Weights are assigned to each particle as a tuple pair.  Resample with replacement kills off particles with low weights, then particles behave as if they were sampled from the distribution we care about p(H=h| latest evidence). Reallocate the *same number of particles* at the higher weighted locations.  Gibbs sampling (GS)isn’t practical for HMMs because requires indep OK for BNs. initialise the first hiddden variable in the BN either as a fixed value or specify a prior distribution P(B1) from which to sample | | | |  | | | | |