|  |
| --- |
| **Chain rule:** |
| **Extended chain rule:** |
| **Conditional probability as a distribution:** |
| **Independence:** If , then and are independent for all values of |
| **Conditional independence:** |
| **Linear Programming**: Set of real-valued variables, linear function with minimization/maximization objective, and a set of linear constraints  Guaranteed *convex function*, can find a global optimum |
| **Factored joint distribution of a Bayes net:** |
| **Local Markov property**: |
| **Trail blocking:** Evidence in **cascade**, evidence in **common cause**, or no evidence in **V structure** (including descendants) |
| **D-separation**: when all trails between 2 sets of nodes are blocked given evidence set. D-separation implies conditional independence  Cascade X -> **E** -> Y  Common cause X <- **E** -> Y  V structure X -> **!(E and descendents)** <- Y |
| **Bayes net** stores joint probabilities of nodes and immediate parents. **Sum product variable elimination** exploits conditional independences in a Bayes net to compute joint probabilities without a full joint distribution table. |
| **Search problem**: Comprises *state space, operators, costs, objective* |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Search** | **Informed?** | **Metric** | **Queue replace?** | **Explore replace?** | | BFS | No | Breadth, node ordering | No | No | | DFS | No | Depth, node ordering | No | No | | BFS opt | No | Path cost | Yes | No | | Best-first | Yes | Heuristic | Yes | No | | A\* | Yes | Path and heuristic | Yes | Yes | |
| **Admissible heuristic**: states , i.e. heuristic does not exceed min cost |
| **Consistent heuristic**: states (triangle inequality) |
| **Markov Random Fields** capture correlation (undirected edges) between variables, joint distribution is the product of *clique potentials* and must be normalized |
| **MPE (most probable explanation)** achieved by variable elimination with a forward and backward pass. Given variables , maximize for |
| **Approximate inference**: Stochastically choose from parent to leaf nodes, count frequency of values |
| **Likelihood weighting**: When using approximate inference with evidence observed, weight each evidence node by to compensate |
| **MDP terms**: tuple with states , actions , transitions and rewards . Decision epoch-finite/infinite horizon, absorbing state-transition outside is impossible, markov assumption-**pastfuture|present**, transition function must be normalized |
| **EU** defined for every state and action:  **MEU** defined for every state: |
| **Policy** can be *stationary/non-stationary*, *deterministic/stochastic* |
| **Value iteration**: Set states , each time step , update :  , where discount factor    **Convergence** when  **Optimal policy guarantee:** |
| **Policy iteration**: Start with a random policy and make improvements over time. , for each state , if then |
| **Linear Programming**: Minimize s.t. |
| **Multi-armed bandit**: Given tries , action count , action rewards ,  expected payoff   |  |  | | --- | --- | | **Greedy:** | **-greedy:**  (1-) of the time else random | | **Softmax:** | **Upper confidence bound:** , with constant |   **Bellman Equation**: |
| **Q-learning**: World is a set of *discrete and finite* states and actions. An **experience** is where is **state** at time step *t*, is **action**, is **reward** and is **new state**. An **episode** is a sequence of experiences from start to a **terminal state** |
| **Tabular Q-learning**: Store Q(s,a) for each state and action in 2D array (rows are states and columns are actions). On new experience, compute , where is step size/learning rate, and is discount factor. Repeat for a fixed number of experiences, or upon some convergence condition (among or ) |
| **Feature-based learning**: Instead of states, learn specific features. Represent Q value according to weighted combination of these features , with weights and features |
| **Activation functions:** Sigmoid , tanh , ReLu (rectified linear unit) – nonlinear functions applied to values computed by neurons to improve usefulness of the model |
| **DQN**: Value function approximated by deep neural network, update using MSE loss with gradient descent Experience replay – Randomly sample transitions from replay memory to remove correlations between states  Fixed targets – freeze target NN for a fixed number of steps before updating  2 threads – 1 thread acts in environment according to -greedy, 1 thread retrieves minibatches from replay buffer |
| **Distributed Constraint Optimization (DCOP)**: Given agents , variables , domain values , constraints , mapping of variables to agents with a minimizing/maximizing assignment |
| **ADOPT**: Transform constraint graph into pseudo-tree. Starting from root node, send value (variable assignment) to child nodes and propagate costs to parent nodes. Update threshold(lower bound) with every level explored, update upper bound when deepest node has been explored. |
| **DPOP**: Transform constraint graph into pseudo-tree. Starting from leaf nodes, propagate costs to parent nodes. Once parent receives all costs from children, it chooses best value and informs children. Propagate back down. |
| **k-optimality**: No deviation by agents in the solution can decrease the solution cost. Form a grp of agents, choose best values within the grp and send to neighbours. Repeat until no improvements |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Algo** | **Sol. quality** | **Memory/agent** | **Message size** | **No. messages** | | ADOPT | Optimal |  |  |  | | DPOP | Optimal |  |  |  | | k-optimal | Suboptimal |  |  |  | |
| **Nash equilibrium:** No player has incentive to change. every finite game has nash equilibrium, pure or mixed |
| **Fictitious play**: Each agent begins with random strategy, computes best response against aggregated belief of opponents actions. Converges for potential, identical interest, zero sum gaes |
| **Potential function :** Defined such that , where are possible actions the agent can take, and are actions of other agents |
| **Correlated equilibrium**: Given a probability distribution on joint strategies, no agent has any incentive to deviate. Let be the recommended strategy for player then we have |
| **Stackelberg game**: Leader policy , follower policy .  Follower LP:  Follower dual:  Leader LP: , **S.T.**  Combining with dual: |