|  |
| --- |
| **Chain rule:** |
| **Extended chain rule:** |
| **Conditional probability as a distribution:** |
| **Independence:** If , then and are independent for all values of |
| **Conditional independence:** |
| **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. |
| **Loss function:** A means to measure accuracy of a classifier function. Must be differentiable so as to apply gradient descent. Loss over an epoch/minibatch is taken as the average loss over the each of the individual predictions |
| **Softmax loss/Cross entropy**: Given output values , take probabilities as . Loss is computed as . Useful for probabilities and classification tasks |
| **Mean Square Error (MSE) loss**: Given output values , take loss as . Useful for predicting numerical values/continuous variables |
| **Neuron input**: For inputs into a neuron, each neuron computes , where is the individual weight learned and is the bias. Each neuron thus has m+1 trainable parameters. |
| **Activation functions:** Sigmoid , tanh , ReLu (rectified linear unit) – nonlinear functions applied to values computed by neurons to improve usefulness of the model |
| **ANN Hyperparameters**: Hidden layers, nodes in each layer, activation function, output nodes, initial weights & bias, learning rate. optimization algos, batch size, epochs |
| **CNN**: Special type of ANN that extracts features from images, and can be highly generalizable for different tasks |
| **Filter**: set of values that slides over the image spatially and computes dot products. is a hyperparameter, is the depth of the input volume. Each filter can only produce an activation map of depth = 1 |
| **Activation maps:** Each filter tries to interpret certain features, and activates in areas where that feature is found |
| **Stride**: How many rows/columns the filter is shifted by when convolving over the image |
| **Padding**: Adding zeros uniformly to the image on all sides to avoid fractional outputs for activation maps |
| **Convolutional layer summary**: Given input volume , with filters, filter size (assume square), stride , and padding , produces an output volume (, . Trainable parameters = |
| **Max pooling**: To make activation map smaller and more tractable, we have a filter that simply takes the max value of all those under that filter’s view. No training parameters needed |
| **Dropout**: Randomly drop out neurons during training to force every neuron to learn something useful. |
| **Batch normalization**: Normalise a layer’s input by subtracting the mini-batch mean and dividing it by standard deviation, to ensure that those inputs have mean = 0 and s.d. = 1. Scale and shift the normalized value |
| **Dependency parsing**: Relations between words can be represented using a dependency tree |
| **Named entity recognition**: Label each noun with the concepts they represent (entity, org., person) |
| **Coreference resolution**: Find all expressions that refer to the same entity |
| **Co-occurrence matrix**: N x N, symmetric matrix storing frequencies of words occurring within a certain window of each other within a corpus of text. |
| **X=USVT­­­­­­**: SVD process applied on co-occurrence matrix |
| **Continuous bag of words**: Predict word given context. Training data = context, prediction = word |
| **Skip-gram**: Predict context given word. Training data = word, prediction = context. Often performs better due to additional training data w Naïve Bayes assumption, better for rare words |
| **Negative sampling**: Reduce complexity by updating small subset of weights |
|  |
| **Recurrent NN**:  where is an activation function, where is output activation |
| **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 not exceed min cost |
| **Consistent heuristic**: states (triangle inequality) |
| **MDP terms**: *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:** |
| **Multi-armed bandit**: Given tries , action count , action rewards ,  expected payoff   |  |  | | --- | --- | | **Greedy**-pick | **-greedy**- (1-) of the time else random | | **Softmax**- | **Upper confidence bound**-, with constant | |
| **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 , where are weights and are features. Saves on memory as weights can be stored as a single vector independent of state, learning becomes more generalizable. |