①

Final

First-Order Logic

Objects - numbers, cars, peoples, houses, colors

properties - breezy, tall, red

relations - inside, adjacent, larger, sibling [holds between two objects]

function - father-of, best friend, age [maps object to object]

term - constant or variable or function {Jack, age(Jack), LeftOf(R)}

Universal Quantification ($\forall$) - For All

Existential Quantification ($\exists$) - There Exists

- can switch order of $\forall x \forall y$, $\exists x \exists y$, but NOT for $\forall x \exists y$ $\exists x \forall y$

Uniqueness Quantifier ($\exists !$) - $\exists ! x$ King(x) - There is only one King X.

Ex: $\forall x$ At(x, UCLA) $\Rightarrow$ smart(x)

$\exists x$ At(x, UCLA) $\Rightarrow$ tall(x)

$\forall y \exists x$ Loves(x, y)] different meaning

$\exists x \forall y$ Loves(x, y)

$\exists x, y$ sister(spot, x) $\wedge$ sister(spot, y) $\wedge \neg (x = y)$

Substitution - exchange x with y in sentence α

↳ $\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

subst({x / John}, α)

King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)

- If a sentence is entailed by a FOL knowledge base, then it is entailed by a finite subset of that propositional knowledge base.

Unification - replace variables to make two statements equivalent

Knows(John, x) Knows(John, Jane) θ = {x / Jane} → Knows(John, Jane) x 2

Resolution - works the same as in propositional logic

- KB(A) $\models$ α (query) → $\overset{CNF}{\Delta} \wedge \neg \alpha$ (prove a contradiction)

① Convert English sentences to FOL

② Convert FOL to CNF → Perform Resolution using Unification as necessary

- If there is a contradiction, then query is proven.

Definite Clause - exactly one positive literal

- $\neg A \vee \neg B \vee C \Rightarrow A \wedge B \Rightarrow C$

Horn Clause - at most one positive literal

- $\neg A \vee \neg B \vee C \Rightarrow A \wedge B \Rightarrow C$

Conversion to CNF

- $\exists x\ Crown(x)$
  ↳ $Crown(C_1)$
  skolem constant
- $\forall y \exists x\ crown(x)$
  ↳ $\forall y\ crown(F(y))$
  skolem function

{ Replace the $\exists$ variable as $F(\forall \text{variable})$

{ Then remove $\exists$ and $\forall$ terms

① Eliminate $\Rightarrow$, $\Leftrightarrow$

- $\alpha \Rightarrow \beta \rightarrow \neg\alpha \vee \beta$
- $\alpha \Leftrightarrow \beta \rightarrow (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha) \rightarrow (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \alpha)$

② push $\neg$ inwards

- $\neg\forall x P \rightarrow \exists x \neg P$
- $\neg\exists x P \rightarrow \forall x \neg P$

De Morgan's $\begin{cases} \neg(\alpha \wedge \beta) \rightarrow (\neg\alpha \vee \neg\beta) \\ \neg(\alpha \vee \beta) \rightarrow (\neg\alpha \wedge \neg\beta) \end{cases}$

③ Skolemize (see above) and Drop $\forall$, $\exists$ terms

④ Distribute $\vee$ over $\wedge$

- $(\alpha \wedge \beta) \vee \gamma \rightarrow (\alpha \vee \gamma) \wedge (\beta \wedge \gamma)$

Probability

Classical logic is "monotonic" - things that are true will remain true even with the addition of new information (ignores possible contradictions)

Degrees of Belief - how much do I believe that something is true? $[0, 1]$

Rules: ① $0 \leq Pr(\alpha) \leq 1$

② α is inconsistent $\Leftrightarrow Pr(\alpha) = 0$

③ α is valid $\Leftrightarrow Pr(\alpha) = 1$

④ $Pr(\alpha) + Pr(\neg\alpha) = 1$

⑤ $Pr(\alpha \vee \beta) = Pr(\alpha) + Pr(\beta) - Pr(\alpha \wedge \beta)$ [α, β not mutually exclusive]

⑥ $Pr(\alpha \vee \beta) = Pr(\alpha) + Pr(\beta)$ [α, β are mutually exclusive / independent]

Belief Change - introduction of new information, must revise probabilities

$$Pr(w \mid \underset{\text{new info}}{\beta}) = \begin{cases} 0 & \text{if } w \models \neg\beta \text{ [if } \beta \text{ is 0 in that world, change probability to 0]} \\ \frac{Pr(w)}{Pr(\beta)} & \text{if } w \models \beta \text{ [original probability of world} \div \text{new total probability after zeroing out stuff]} \end{cases}$$

Bayes Conditioning: $Pr(\alpha \mid \beta) = \frac{Pr(\alpha \wedge \beta)}{Pr(\beta)}$ [if α, β independent - $Pr(\alpha \wedge \beta) = Pr(\alpha) \cdot Pr(\beta)$]

Independence: $[Pr(\alpha \mid \beta) = Pr(\alpha)]$, $[\frac{Pr(\alpha \wedge \beta)}{Pr(\beta)} = Pr(\alpha)]$ check if $\{P(E) = P(E \mid B)$ & $P(B) = P(B \mid E)$

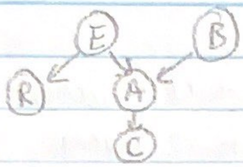
Chain Rule: $Pr(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) = Pr(\alpha_1 \mid \alpha_2 \ldots \alpha_n) Pr(\alpha_2 \mid \alpha_3 \ldots \alpha_n) \ldots Pr(\alpha_n)$

↳ For Graph: use each nodes' parents/children $\rightarrow Pr(\overset{p}{a}) Pr(\overset{p}{b}) Pr(\overset{c}{c} \mid \overset{p}{a}) Pr(\overset{c}{d} \mid \overset{p}{a}, b) \ldots$

Case Analysis: Find probability when not explicitly given $\rightarrow$ use all possible pairings for needed variable $P(T) = P(T \mid O) P(O) + P(T \mid \neg O \wedge \neg G) P(\neg O \wedge \neg G$

Bayes Rule: $Pr(\alpha \mid \beta) = \frac{Pr(\alpha \wedge \beta)}{Pr(\beta)} = \frac{Pr(\beta \mid \alpha) Pr(\alpha)}{Pr(\beta)}$ [$Pr(\alpha \wedge \beta) = Pr(\alpha \mid \beta) Pr(\beta)$]

Bayesian Networks



Parents : A : E, B   C : A   R : E   B : ∅

Descendants : E : R, A, C   C : ∅   B : A, C

Non-Descendants : A : R   C : R, E, B   E : B

↳ exclude V and its direct parents

Markovian Assumptions : I (V, Parents (V), Non-Descendants (V))
independence   node   ← write assumptions for all nodes in a graph

↳ Every node in a Bayesian Network is conditionally independent of its non-descendants given its parents → I(A, ∅, BE) means A independent of B, E

d-seperation : if any path from X → Y is open, then it is not d-seperated.
If at least one path from X → Y is blocked, then it is d-seperated.

dsep(B, EC, R)   [Known = Z]
start known end

① sequential : → W → [blocked iff W ∈ Z]

② divergent : ← W → [blocked iff W ∈ Z]

③ convergent : → W ← [blocked iff W ∉ Z AND descendents(W) ∉ Z]

Prior Marginal - CPT before we have any evidence (probability a random person has disease

Posterior Marginal - CPT after acquiring evidence (probability person has disease after test res

Most Probable Explanation (MPE) - assume the final consequence and search for the most probable query [uses all variables]

Maximum a Posteriori (MAP) - same idea as MAP, but don't include all variables [generally more complex, less efficient]

Same Decision Probability (SDP) - what is the probability that our decision [to prescribe medicine] will stay the same after getting more information [tests]

Complexity of Inference (MAR) - variable elimination, conditioning

- Time/Space complexity : $O(nd^w)$ n: # variables   d: # of values [0,1]   w = tree width
- normal tree : w = 1 ; polytree : nodes have multiple parents, w = max # of parents a node

Weighted Model Counting (WMC) - more general version of #SAT

- instead of finding the number of SAT worlds, find the sum of the weights of SAT worlds
- weight of a world is the product of weights assigned to its literals
- compile Δ formula into smooth, decomposable, deterministic NNF circuit → WMC in linear ti

Modeling Logic as a Bayesian Network

① Decide variables/values ② Decide edges ③ Create CPTs {cold, flu, ... } example

- Bipartite - edges only go from the causes to the effects, 2 layers
- complete data - no ? in the data, everything filled out {model with Bayesian Net}
- incomplete data - have at least one V? unknown in the data {model with Expectation maximization}
  - EM - iterative method to find local maximum or MAP estimates of parameters

Maximum likelihood principle - learn the parameters that maximize the likelihood of observing a certain set of data

calculate likelihood : $\theta_1 = Pr_1(\underset{\text{example (like a patient)}}{e_1}) \cdot Pr_1(e_2) \cdot \ldots \cdot Pr_1(e_n)$ [Based on Bayesian Net #1]

$\theta_2 = Pr_2(e_2) \cdot Pr_2(e_2) \cdot \ldots \cdot Pr_2(e_n)$ [Based on Bayesian Net #2]

Choose the greater one as your parameters/Bayesian Network

MLP with complete data (given a complete dataset)

① Find Empirical Distribution - for all possibilities of the variables, what is the probability $Pr_D(\cdot)$ of randomly picking that one from the dataset?

② Find the MLP (ex: $\theta_{\bar{s}|h} = \frac{Pr(\bar{s},h)}{Pr(h)} = \frac{Pr_D(\bar{s},h)}{Pr_D(h)} = \frac{10/16}{12/16} = \frac{5}{6}$

MLP with incomplete data (given an incomplete dataset)

① Use Expectation Maximization (EM)

- basically guess random values for unknown variable and choose the assignment that yields the highest probability; use this in the CPT
- repeat as necessary for any and all unknown variables
- once dataset has been filled up, follow steps for complete dataset

Optimizing only using MLP can lead to overfitting [model is too specific to the training data and doesn't generalize to test data]

Supervised Learning [Query-Oriented, Labeled Data] - classification, regression

Unsupervised Learning [Model-oriented, Unlabeled Data] - clustering

Loss Function - evaluates how close test results are to the expected result (i.e. mean-squared)

Arithmetic Circuits - given a query, perform WMC on equivalent arithmetic circuit

- $A = T \{\lambda_a = 1, \lambda_{\bar{a}} = 0\}$ $A = F \{\lambda_a = 0, \lambda_{\bar{a}} = 1\}$ $A = ? \{\lambda_a = 1, \lambda_{\bar{a}} = 1\}$
- Evaluating AC is $O(n)$, but converting query to AC is $O(nd^w)$
- values for the θ stuff in the AC should be given in problem

Cross Entropy (Loss Function): $CE = \sum_x Q(x) \log_2 P(x)$ P(x): prediction Q(x): label [prefer smaller CE]

Entropy - uncertainty in a single distribution, used for decision trees

Cross Entropy - Measure similarity between two distributions, loss function for gradient descent

Machine Learning

Decision Trees and Random Forests - examples of classifiers (supervised)

Entropy - less entropy means more certainty $[ENT(X) = -\sum_x Pr(x)\log_2 Pr(x)]$

Conditional Entropy - more commonly used

- If we know ENT(X) and $Y = y$, then $ENT(X|y) = -\sum_x Pr(x|y)\log_2 P(x|y)$
- If we don't know Y but want to observe it, then $ENT(X|Y) = -\sum_y Pr(y)\,ENT(X|y)$

Additional information can never increase entropy $\rightarrow ENT(X|Y) \leq ENT(X)$ [information gain]

Decision Tree - interpretable classifier

- best to split into training data and test data to train tree and check accuracy
- cross-validation: perform the train/test process multiple times but splitting data differently
- determine how to create decision tree / how to split it based on entropy

47

① Look at all Input Attributes, choose one with lowest conditional entropy based on result/class [greatest information gain] and split on that

- $ENT(D|a) = Pr(a) \cdot -[Pr(d|a)\log_2 Pr(d|a) + Pr(\bar{d}|a)\log_2 Pr(\bar{d}|a)]$

  $ENT(D|\bar{a}) = Pr(\bar{a}) \cdot -[Pr(d|\bar{a})\log_2 Pr(d|\bar{a}) + Pr(\bar{d}|\bar{a})\log_2 Pr(\bar{d}|\bar{a})]$

  $ENT(D|A) = ENT(D|a) + ENT(D|\bar{a})$ [minimize ENT]

- If CE = 0 when splitting on any variable, immediately split on that one

② Split the examples into True (+) or False (−) for the variable (A) that you split on, based on the class for that example

- For A = T : try to split on remaining input attributes

  For A = F : try to split on remaining input attributes

  [X_1 : A = F, D = YES → A = F, + X_1]

③ Keep splitting until leaves of all the tree are either all (+) or all (−)

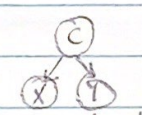
- Leaves that are all (+) become YES, Leaves that are all (−) become NO

Random Forests - more refined version of decision trees

- Decision Trees are often highly specialized to just one dataset, bad generalization
- Random Forests create multiple copies of training data that include different examples from the data, and create DTs on each copy dataset using randomly selected features (input attributes)
- When new data point enters Random Forest, put it through each little DT and select the final class / Result that has a majority vote

Bayesian Network Classifier - set a threshold (T) to classify the inputs

$$C = \begin{cases} c & \text{iff } Pr(c|a_1, a_2, \ldots, a_N) \geq T \\ \bar{c} & \text{iff } Pr(c|a_1, a_2, \ldots, a_N) < T \end{cases}$$



X, Y are independent given C

Naive Bayes Classifier - think of spam email example

- these words have probability X of being in normal email, probability Y of being in spam, open new email and use these probabilities to analyze words → decide

Neural Networks

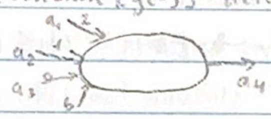
Neuron - building block of a neural network

Activations - input/outputs of a neuron, typically numbers

Weights (w_{ij}) - associated with input/output activation, value for each input multiplied by weight in calculation

Bias (b) - additional term that may be added in calculation

Activation Function [$g(\cdot)$] - determines if neuron is on/off



$a_4 = g(2a_1 - a_2 + 0a_3 + b)$

↳ a_4 can be used as input into other neurons

↳ ① step function (t = Threshold)
$$\begin{cases} g(x) = 1 & \text{if } x \geq t \\ g(x) = 0 & \text{if } x < t \end{cases}$$

② Sign function
$$\begin{cases} g(x) = 1 & \text{if } x \geq 0 \\ g(x) = -1 & \text{if } x < 0 \end{cases}$$

③ Sigmoid
$$g(x) = \frac{1}{1+e^{-x}}$$

④ ReLU
$$g(x) = \max(0, x)$$

Feed-Forward Neural Network - neuron outputs only feed forward into next neuron

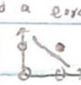
Fully-connected layer - every neuron takes input from all neurons in previous layer

depth - number of layers in neural network (including output layer, excluding input layer)

deep learning - ability to train "deep" (many layers) neural networks

Universal Function Approximators - neural networks are very expressive, can express every function up to a constant error

Neurons can represent basic logic gates as long as they are linearly separable



↳ a full neural network can create more complex, non-linear functions

Training neural networks is like an optimization problem

↳ use loss functions like cross-entropy, mean square error

perform optimization using Gradient Descent - find local min/max of function to optimize parameters, fit line to data

Accuracy - how well can the neural network classify testing data after working through the training data?

When data set is too big to analyze all at once:

Epoch - when an entire dataset (all the batches) is passed forward and backward through the neural network only once

Batch - divide the dataset into multiple batches when it's too big

Stopping criteria - split training data into TD2 (80%) and validation (20%)

- train on TD2 within an epoch, using validation data as test data
- monitor performance on validation data over epochs - when performance peaks/plateaus, we stop there