

Probability.

A random variable is some aspect of the world about which we have uncertainty.
 $P(X=x) \geq 0$, and $\sum_x P(X=x) = 1$

Joint distribution: $P(X_1=x_1, X_2=x_2, \dots, X_n=x_n)$ or $P(x_1, x_2, \dots, x_n)$
 $\Rightarrow P(x_1, x_2, \dots, x_n) \geq 0$, $\sum_{x_1, x_2, \dots, x_n} P = 1$

An event is a set E of outcomes.

$$P(E) = \sum_{x_1, x_2, \dots, x_n \in E} P(x_1, x_2, \dots, x_n)$$

$P(X_1=x_1) = \sum_{x_2, \dots, x_n} P(x_1, x_2, \dots, x_n)$ or like this kind.

Conditional Distributions, $P(a|b) = \frac{P(a,b)}{P(b)}$

P(T, W)		
T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$$P(W = \text{sun} | T = \text{hot}) = \frac{P(W = \text{sun}, T = \text{hot})}{P(T = \text{hot})} = \frac{0.4}{0.5} = 0.8$$

$$P(W = \text{rain} | T = \text{hot}) = \frac{P(W = \text{rain}, T = \text{hot})}{P(T = \text{hot})} = \frac{0.1}{0.5} = 0.2$$

$$P(W = \text{sun} | T = \text{cold}) = \frac{P(W = \text{sun}, T = \text{cold})}{P(T = \text{cold})} = \frac{0.2}{0.5} = 0.4$$

$$P(W = \text{rain} | T = \text{cold}) = \frac{P(W = \text{rain}, T = \text{cold})}{P(T = \text{cold})} = \frac{0.3}{0.5} = 0.6$$

$$P(x_i | x_{-i}) = \frac{P(x_i, x_{-i})}{P(x_{-i})} = \frac{P(x_i, x_{-i})}{\sum_{x_i} P(x_i, x_{-i})}$$

Probabilistic Inference: compute a desired probability from other known probabilities.

Product Rule: $P(x, y) = P(x|y) \cdot P(y) \Leftrightarrow P(x|y) = \frac{P(x,y)}{P(y)}$

Chain Rule:

$$P(x_1, x_2, \dots, x_n) = P(x_1) \cdot P(x_2 | x_1) \cdot P(x_3 | x_1, x_2) \dots$$

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | x_1, \dots, x_{i-1})$$

Bayes' Rule: $P(x_i | y) = \frac{P(x_i | y) \cdot P(y)}{P(y)}$
 $\Rightarrow P(x_i | y) = \frac{P(y | x_i) \cdot P(x_i)}{P(y)}$

Example: Diagnostic probability from causal probability:

$$P(\text{cause} | \text{effect}) = \frac{P(\text{effect} | \text{cause}) P(\text{cause})}{P(\text{effect})}$$

Example:

- M. meningitis, S. still sick

$P(+m) = 0.0001$	Example gates
$P(+s +m) = 0.8$	
$P(+s -m) = 0.01$	

$$P(+m | +s) = \frac{P(+s | +m) P(+m)}{P(+s | +m) P(+m) + P(+s | -m) P(-m)} = \frac{0.8 \times 0.0001}{0.8 \times 0.0001 + 0.01 \times 0.9999} \approx 0.08$$

Note: posterior probability of meningitis still very small
 Note: you should still get into medical school!

$$P(\text{sun} | \text{dry}) = \frac{P(\text{dry} | \text{sun}) \cdot P(\text{sun})}{P(\text{dry})} = \frac{0.9 \times 0.4}{0.9 \times 0.4 + 0.3 \times 0.2} = \frac{0.36}{0.36 + 0.06} = \frac{0.36}{0.42} \approx 0.86$$

Independence:

Two variables are independent if: $\forall x, y: P(x, y) = P(x) P(y)$

or another form: $\forall x, y: P(x|y) = P(x)$

Write as $X \perp\!\!\!\perp Y$

Conditional independence:

X is conditionally independent of Y given Z if and only if:

$$\forall x, y, z: P(x, y | z) = P(x | z) \cdot P(y | z)$$

or equivalently, $\forall x, y, z: P(x | y, z) = P(x | z)$

Bayes' Net

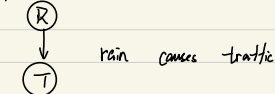
Probabilistic Models

Bayes' Net Notation.

Nodes: variables (with domains).

Arcs: interactions: indicate "direct influence" between variables.

Example: R: rain, T: traffic.



$$P(x_1, x_2, \dots, x_n) = \prod_i P(x_i | \text{Parents}(x_i))$$

Size of a Bayes' Net

How big is a joint distribution over N boolean variables: 2^N .

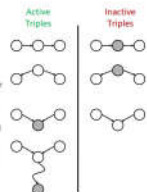
How big is an N -node net if nodes have up to k parents: $O(n \cdot 2^{k+1})$

D-separation.

Active / Inactive Paths.

Active / Inactive Paths

- Question: Are X and Y conditionally independent given evidence variables $\{Z\}$?
- Yes, if X and Y "d-separated" by Z
- Consider all (undirected!) paths from X to Y
- No active paths = independent!



D-separation:

Query: $X \perp\!\!\!\perp Y | \{x_{i_1}, \dots, x_{i_n}\}$?

Check all (undirected!) paths between X_i and X_j .

If one or more active, independence not guaranteed.

If all inactive independence.

Inference by Enumeration in Bayes' Net

- Given unlimited time, inference in BNs is easy
- Reminder of inference by enumeration by example:

$$P(B | +j, +m) \propto_{\theta} P(B, +j, +m)$$

$$= \sum_{i, n} P(B, i, n, +j, +m)$$

$$= \sum_{i, n} P(B) P(i) P(n | B, i) P(+j | i) P(+m | n)$$

$$= P(B) P(i) P(n | B, i) P(+j | i) P(+m | n)$$

Enumerate Inference

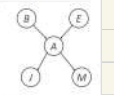
Inference by Enumeration in Bayes' Net

- Given unlimited time, inference in BNs is easy
- Reminder of inference by enumeration by example:

$$P(B | +j, +m) \propto_{\theta} P(B, +j, +m)$$

$$= \sum_{i, n} P(B, i, n, +j, +m)$$

$$= \sum_{i, n} P(B) P(i) P(n | B, i) P(+j | i) P(+m | n)$$



Operation 2: Eliminate

- Second basic operation: marginalization
- Take a factor and sum out a variable
- Shrink a factor to a smaller one

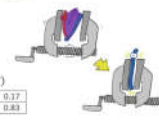
A projection operation

Example:

$$P(R, T) \xrightarrow{\text{sum } R} P(T)$$

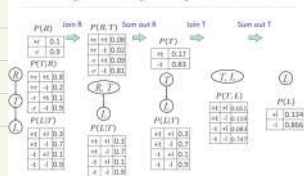
++	0.08
+-	0.02
-+	0.09
--	0.81

$$\Rightarrow P(T) = \begin{matrix} ++ & 0.37 \\ -+ & 0.83 \end{matrix}$$



Variable Elimination = Marginalizing Early

Marginalizing Early! (aka VE)



Example

$$P(B | +j, +m) \propto_{\theta} P(B, +j, +m)$$

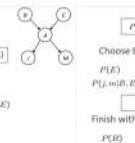
$$= P(B) P(E) P(A | B, E) P(G | A) P(M | A)$$

Choose A:

$$P(A | B, E) P(G | A) P(M | A) \Rightarrow P(A, m | B, E) \Rightarrow P(A, m | B, E)$$

Finish with B:

$$P(B) P(E) P(A, m | B, E) \Rightarrow P(B, m | B, E)$$



Example ctd.

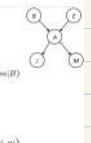
$$P(B) P(E) P(A, m | B, E) \Rightarrow P(B, m | B, E)$$

Choose E:

$$P(E) P(A, m | B, E) \Rightarrow P(A, m | B) \Rightarrow P(A, m | B)$$

Finish with B:

$$P(B) P(A, m | B) \Rightarrow P(A, m) \Rightarrow P(A, m)$$



VE: Computational and Space Complexity

- The computational and space complexity of variable elimination is determined by the largest factor.
- The elimination ordering can greatly affect the size of the largest factor.
- E.g., previous slide's example $O(2^4)$ vs. $O(1)$
- Does there always exist an ordering that only results in small factors?
- Not!

Approximate Inference: Sampling.

Basic idea: of sampling.

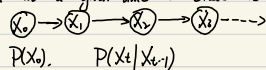
Draw N samples from a sampling distribution S .

Compute an approximate posterior probability

Show this converges to the true probability P .

Markov Models.

Value of X at a given time is called the state.



Stationary Distributions

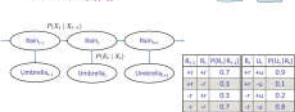
The limiting distribution (if exists) is called the stationary distribution P_{∞} of the chain. (0.5)

Hidden Markov Models.

HMM Definition and Weather Example

An HMM is defined by:

- Initial distribution: $P(X_0)$ or $P(X_1)$
- Transitions: $P(X_i | X_{i-1})$
- Emissions: $P(E_i | X_i)$



	Sunny	Rainy	Cloudy
Sunny	0.5	0.4	0.1
Rainy	0.4	0.7	0.2
Cloudy	0.1	0.3	0.6

Joint distribution for Markov Model.

$$P(X_0, \dots, X_n) = P(X_0) \prod_{i=1}^n P(X_i | X_{i-1})$$

Joint distribution for hidden Markov model.

$$P(X_0, X_1, \dots, X_n, E_1, \dots, E_n) = P(X_0) \prod_{i=1}^n P(X_i | X_{i-1}) P(E_i | X_i)$$

Filtering: $P(X_k | e_{1:k})$

Belief State: input to the decision process of a rational agent.

Prediction: $P(X_{k+1} | e_{1:k})$ for $k > 0$.

Smoothing: $P(X_k | e_{1:n})$ for $0 \leq k \leq n$

Most likely explanation: $\arg \max_{x_{1:n}} P(x_{1:n} | e_{1:n})$

Forward Algorithm

$$P(X_{n+1} | e_{1:n}) = \frac{P(e_{n+1} | X_{n+1}) \cdot \sum_{x_n} P(X_n | e_{1:n}) P(X_{n+1} | x_n)}{\text{Norm}} \quad \text{Predict}$$

Norm: Normalise update: update Predict: Predict

$$P(X_{n+1} | e_{1:n+1}) = P(X_{n+1} | e_{1:n}, e_{n+1})$$

$$= \alpha P(e_{n+1} | X_{n+1}) P(X_{n+1} | e_{1:n})$$

$$= \alpha P(e_{n+1} | X_{n+1}) P(X_{n+1} | e_{1:n})$$

$$= \alpha P(e_{n+1} | X_{n+1}) \sum_{x_n} P(X_n | e_{1:n}) P(X_{n+1} | x_n)$$

$$= \alpha P(e_{n+1} | X_{n+1}) \sum_{x_n} P(X_n | e_{1:n}) P(X_{n+1} | x_n)$$

Bayes rule
or normalization

Conditional ind.

Law of total prob.

Conditional ind.

filter: Forward (filter, emit)

Particle Filtering.

Each particle is moved by sampling its next position from the transition model:

$$x' = \text{sample}(P(X' | x))$$

New particles approximates:

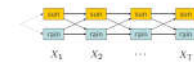
$$\sum_{x_i} P(X_{n+1} | x_i) f(x_i)$$

Most Likely explanation (MLE)

$\arg \max_{x_{1:n}} P(X_{1:n} | e_{1:n})$ Viterbi algorithm

State Trellis

State trellis: graph of states and transitions over time



- Each arc represents some transition $x_{i-1} \rightarrow x_i$
- Each arc has weight $P(x_i | x_{i-1}) P(e_i | x_i)$
- Each path is a sequence of states
- Product of weights on a path = sequence's probability along with the evidence
- Forward algorithm computes sums of paths, Viterbi computes best path

Forward / Viterbi algorithms



Forward Algorithm (sum)

For each state at time i , keep track of the total probability of all paths to it

$$f_{i+1} = \text{FORWARD}(f_i, \theta_{i+1})$$

$$= \alpha P(e_{i+1} | X_{i+1}) \sum_{x_i} P(X_i | e_{1:i}) f_i$$

Viterbi Algorithm (max)

For each state at time i , keep track of the maximum probability of any path to it

$$m_{i+1} = \text{VITERBI}(m_i, \theta_{i+1})$$

$$= P(e_{i+1} | X_{i+1}) \max_{x_i} P(X_i | e_{1:i}) f_i$$

Time complex: $O(n^2 T)$

Space: $O(nT)$

Number of paths: $O(n^T)$

Dynamic Bayes' Nets.

Machine learning

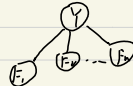
Model-based Classification

- build a model, where both the label and features are random variables
- Instantiate any observed features
- Query for the distribution of the label conditioned on the features.

Naïve Bayes for Digs.

Naïve Bayes: Assume all features are independent effects of the label.

$$P(Y | F_0, \dots, F_{n-1}) \propto P(Y) \prod_{j=0}^{n-1} P(F_j | Y)$$



General Naïve Bayes Model:

$$P(Y | F_1, \dots, F_n) = P(Y) \prod_{j=1}^n P(F_j | Y)$$

$|Y| \times |F|^n$ values. $|Y|$ possible $|F| \times |Y|$ pairs.

Inference for Naïve Bayes

- Goal: compute posterior distribution over label variable Y
- Step 1: get joint probability of label and evidence from each label

$$P(Y, f_1, \dots, f_n) = \begin{bmatrix} P(Y, f_1, \dots, f_n) \\ P(Y, f_1, \dots, f_n) \\ P(Y, f_1, \dots, f_n) \end{bmatrix}$$

$$P(Y, f_1, \dots, f_n) = P(Y) \prod_{j=1}^n P(f_j | Y)$$

Step 2: sum to get probability of evidence

Step 3: normalize by dividing Step 1 by Step 2

Naïve Bayes for Text

$P(Y, w_1, \dots, w_n) = P(Y) \prod_{j=1}^n P(w_j | Y)$ Word at position j

Training and Testing.

Data: labeled instances, e.g. emails marked spam / ham.

- Training set
- Validation set
- Test set.

Features: attribute-value pairs which characterize each x .

- Overfitting and generalization
- Want a classifier which does well on test data
- Overfitting: fitting the training data very closely, but not generalizing well
- Underfitting: fits the training set poorly

Relatively frequency parameters will overfit the training data

Parameter estimation

$$P_{ML}(x) = \frac{\text{count}(x)}{\text{total count}}$$

Estimate that maximizes the likelihood of data

$$L(x, \theta) = \prod_{i=1}^n P(x_i | \theta) = \theta^{\sum x_i} (1-\theta)^{n - \sum x_i}$$

$$P_{ML}(r) = 2/3$$

$$P_\theta(x = \text{red}) = \theta$$

$$P_\theta(x = \text{blue}) = 1 - \theta$$

Maximum Likelihood Estimation

Data: observed set D of size n and n trials.

Hypothesis space: Binomial distributions: $\text{Bin}(\theta, n)$

Learning: finding θ is an optimization problem

objective function: $P(D | \theta) = \theta^{\sum x_i} (1-\theta)^{n - \sum x_i}$

MLE: Choose θ to maximize probability of D .

$$\hat{\theta} = \arg \max_{\theta} P(D | \theta)$$

$$= \arg \max_{\theta} \ln P(D | \theta) = \arg \max_{\theta} \ln \theta^{\sum x_i} (1-\theta)^{n - \sum x_i}$$

Set derivative to zero, and solve:

$$\frac{d}{d\theta} \ln P(D | \theta) = \frac{d}{d\theta} (\sum x_i \ln \theta + (n - \sum x_i) \ln (1-\theta))$$

$$= \frac{d}{d\theta} (\sum x_i \ln \theta + (n - \sum x_i) \ln (1-\theta))$$

$$= \sum x_i \cdot \frac{1}{\theta} - (n - \sum x_i) \cdot \frac{1}{1-\theta} = \frac{\sum x_i}{\theta} - \frac{n - \sum x_i}{1-\theta} = 0$$

$$\Rightarrow \hat{\theta}_{MLE} = \frac{\sum x_i}{n}$$

Smoothing

Laplace Smoothing

$$P_{Lap}(x) = \frac{\text{count}(x) + 1}{\sum_{x'} (\text{count}(x') + 1)} = \frac{\text{count}(x) + 1}{N + |X|}$$

Conditional: $P_{Lap}(x, y) = \frac{\text{count}(x, y) + 1}{\sum_{x'} (\text{count}(x', y) + 1)}$

Estimation: Linear Interpolation.

In practice, Laplace can perform poorly for ~~P(X|Y)~~ $P(X|Y)$

1. When $|X|$ very large
2. When $|Y|$ very large.

Linear interpolation: another option.

$$\text{Punt}(x): \alpha \hat{P}(x|y) + (1-\alpha) \hat{P}(x)$$

or get the empirical $P(X)$ from data.

Tuning on Validation data.

Two kinds of unknown.

1. parameters $P(X|Y)$, $P(Y)$.
2. hyperparameters: eg. the amount / type of smoothing to do, k , α .

Base lines.

Candidates from a classifier.

Predictor over the top labels.

$$\text{confidence}(x) = \max_y P(y|x)$$

Discriminative learning.

Linear classifiers.

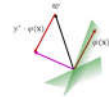
Inputs are feature values. Each feature has a weight. Sum is the activation.

$$\text{activation}(x) = \sum w_i \phi_i(x) = w \cdot \phi(x)$$

If activation is positive, output 1, negative, output -1

Learning: Binary Perceptron

- Start with weights $w=0$
- For each training instance (x, y^*)
 - Classify with current weights.
 - If correct ($\hat{y} = y^*$), no change
 - If wrong, adjust the weight vector by adding or subtracting the feature vector. Subtract if $y^* = -1$.



Maximum likelihood estimation

$$\theta_{MLL} = \arg \max_{\theta} P(X|\theta)$$
$$= \arg \max_{\theta} \prod_i P_i(X_i|\theta)$$



Maximum conditional likelihood estimation

$$\theta^* = \arg \max_{\theta} P(Y|X, \theta)$$

$$= \arg \max_{\theta} \prod_i P_i(y_i|x_i, \theta)$$

$$\ell(\theta) = \prod_i \frac{e^{\theta^T \phi_i(x_i, y_i)}}{\sum_{y_i} e^{\theta^T \phi_i(x_i, y_i)}}$$

$$\ell(\theta) = \sum_i \log P_{\theta}(y_i|x_i)$$
$$= \sum_i \theta^T \phi_i(x_i, y_i) - \log \sum_{y_i} e^{\theta^T \phi_i(x_i, y_i)}$$

$$\ell(\theta) = \sum_i \log P_{\theta}(y_i|x_i)$$

$$= \sum_i w_{y_i} \cdot x_i - \log \sum_{y_i} e^{w_{y_i} \cdot x_i}$$

$$\frac{d}{dw_{y_i}} \log P_{\theta}(y_i|x_i) = \begin{cases} x_i - x_i \frac{e^{w_{y_i} \cdot x_i}}{\sum_{y_i} e^{w_{y_i} \cdot x_i}} & \text{if } y = y_i \\ -x_i \frac{e^{w_{y_i} \cdot x_i}}{\sum_{y_i} e^{w_{y_i} \cdot x_i}} & \text{otherwise} \end{cases}$$

$$= x_i (I(y = y_i) - P(y_i|x_i))$$

Stochastic Gradient Descent

Initialize w (e.g., randomly)
repeat for K iterations:
for each example (x_i, y_i) :
compute gradient $\Delta_i = -\nabla_{w_i} \log P_{\theta}(y_i|x_i)$
 $w \leftarrow w + \alpha \Delta_i$

$$\frac{d}{dw_{y_i}} \log P_{\theta}(y_i|x_i) = x_i (I(y = y_i) - P(y_i|x_i))$$

If $y_i = y$, move w_y toward x_i with weight $(1 - P(y_i|x_i))$

If $y_i \neq y$, move w_y away from x_i with weight $P(y_i|x_i)$

compare this to the maximum likelihood estimation

Neural Nets.

Artificial Neuron.

$$\text{Perceptron (Heaviside)}: g(z) = \text{Sign}(z)$$

$$\text{Logistic regression: } g(z) = \frac{1}{1 + e^{-z}}$$

$$\text{Linear regression: } g(z) = z$$

g : activation function

How to Learn the Parameters of the Model?

Iterative method that updates unknown weights

For perceptron:

$$w \leftarrow w + y^* \phi(x) \mathbb{1}(w \cdot \phi(x) - y^* < 0)$$

For logistic regression:

Write likelihood function: $P(X, Y|w)$

Maximize $\log P(X, Y|w)$ with gradient descent

$$w \leftarrow w + \alpha \nabla_w \log P(X, Y|w) \quad (\text{batch})$$

$$w \leftarrow w + \alpha \nabla_w \log P(x, y|w) \quad (\text{stochastic})$$

General Framework: Statistical ML

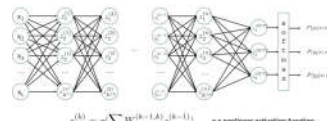
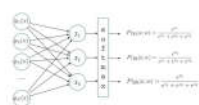
- Minimize empirical risk:
$$\min_w \sum_i \ell(\theta_w(x^i), y^i)$$
- For perceptron:
$$\ell(y, y') = \max(0, -y y')$$
- For logistic regression:
$$\ell(y, y') = -y \log(y) - (1-y) \log(1-y)$$
- For linear regression:
$$\ell(y, y') = (y - y')^2$$

Neural Networks.

Multi-class Logistic Regression

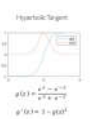
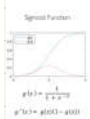
(Deep) Neural Network

Directly learns the features from data



$$z_i^{(k)} = g(\sum_j W_{ij}^{(k-1)} z_j^{(k-1)}) \quad g = \text{nonlinear activation function}$$

Common Activation Functions



Summary of Key Ideas

- Minimize empirical risk: $\min_w \sum_i \ell(\theta_w(x^i), y^i)$
- Continuous optimization
 - Gradient descent
 - Integrate negative gradient direction (gradient is just vector of partial derivatives)
 - Take step in the negative gradient direction
 - Repeat until hold out data accuracy starts to drop ("early stopping")
- Deep neural nets
 - Last layer returns expected output (e.g., probability of classes)
 - These also output mean layers before that have designed
 - comparing the features
 - the features are learned rather than hand designed
 - Universal function approximation theorems
 - if ϕ is "rich" and n is large enough
 - the number of nodes grows exponentially (assuming the training data is "early stopping")
 - Automatic differentiation gives the derivatives efficiently

Logical Agent and Propositional Logic.

A Knowledge-based Agent (KB).

Knowledge base = set of sentences in a formal language.

Propositional logic

Symbol: 1. variable that can be true or false, try to use capital letters

Operators: $\neg A$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$

Propositional logic Symbol

Given a set of propositional symbols $\{X_1, \dots, X_n\}$.

Sentence: \rightarrow Atomic Sentence / Complex Sentence.

Atomic: True / False / Symbol.

Symbol: X_1, X_2, \dots, X_n

Complex: \neg Sentence, / Sentence \wedge Sentence, ...

$$A \Rightarrow B \equiv \neg A \vee B, \quad A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A)$$

Logical Consequences.

Entailment: determine truth of sentences based on semantics.

Inference: generate new sentence from current KB.

Entailment \models iff in every world where α is true, β is also true
 $\text{model}(KB) \subseteq \text{model}(\text{query})$

Validating and Satisfiability.

A sentence is valid if it is true in every model.

$\models \mathcal{P}$ if and only if $\alpha \Rightarrow \mathcal{P}$ is valid.

A valid sentence is also called tautology

satisfiable if a sentence is true in some model.

Simple model checking.

Same recursion as backtracking. $O(2^n)$ time, linear space.

CNF: $A \wedge B \wedge C$, A is clause.

KB: set of sentences.

$$\text{Modus ponens: } \frac{\alpha \Rightarrow \beta, \alpha}{\beta} \equiv [(A \Rightarrow B) \wedge A] \Rightarrow B$$

And elimination.

$$\frac{A \wedge B}{A} \quad \frac{A \wedge B}{B}$$

$$\text{Biconditional elimination: } \frac{A \Leftrightarrow B}{(A \Rightarrow B) \wedge (B \Rightarrow A)}$$

We want inference to be sound.

If we can prove B from A ($A \vdash B$), then $A \models B$

We would like inference to be complete.

If $A \models B$, then we can prove B from A ($A \vdash B$)

$$\frac{\alpha \vdash \beta, \gamma}{\alpha} \quad \frac{\alpha \vdash \beta, \gamma}{\alpha} \quad \frac{\alpha \vdash \beta, \gamma}{\alpha} \quad \frac{\alpha \vdash \beta, \gamma}{\alpha} \quad \frac{\alpha \vdash \beta, \gamma}{\alpha}$$

show $KB \models \alpha$ by showing unsatisfiability of $(KB \wedge \neg \alpha)$