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: $P(X_1=x_1, X_2=x_2, \ldots, X_n=x_n)$ or $P(x_1, x_2, \ldots, x_n)$

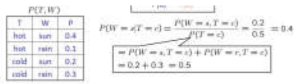
$\Rightarrow P(x_1, x_2, \ldots, x_n) \geq 0$, $\sum_{(x_1, x_2, \ldots, x_n)} P = 1$

An event is a set E of outcomes.

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

$P(X_1 = x_1) = \sum_{x_2} P(X_1 = x_1, X_2 = x_2)$, or ~~sth~~ like this kind.

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



$P(x_1 | x_2) = \frac{P(x_1, x_2)}{P(x_2)} = \frac{P(x_1, x_2)}{\sum_{x_1} P(x_1, x_2)}$

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

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

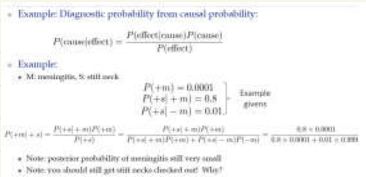
Chain Rule:

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

$P(x_1, \ldots, x_n) = \prod_i P(x_i | x_1 \cdots x_{i-1})$.

Bayes' Rule. $P(x, y) = P(x|y) \cdot P(y) = P(y|x) \cdot P(x)$

$\Rightarrow P(x|y) = \frac{P(y|x) \cdot P(x)}{P(y)}$



$P(\text{sun} | \text{dry}) = \frac{P(\text{dry} | \text{sun}) \cdot P(\text{sun})}{P(\text{dry})} = \frac{0.9 \times 0.8}{P(\text{dry}|\text{sun}) \cdot P(\text{sun}) + P(\text{dry}|\text{rain}) \cdot P(\text{rain})} = \frac{0.9 \times 0.8}{0.9 \times 0.8 + 0.3 \times 0.2} = \frac{0.72}{0.78}$

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 $X \perp\!\!\!\perp Y | Z$ if and only if.

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

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

Bayes' Nets

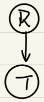
Probabilistic Models

Bayes' Net Notation.

Nodes: variables (with domains).

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

Example: R: rain, T: traffic.



rain causes traffic

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

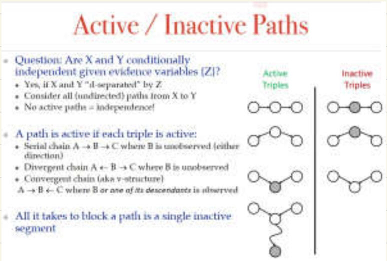
Size of a Bayes' Net

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.



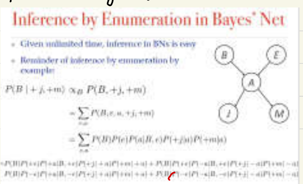
D-separation:

Query: $X_i \perp\!\!\!\perp X_j | \{X_{k_1}, \ldots, X_{k_n}\}$?

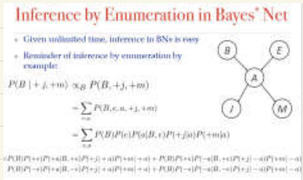
Check all (undirected!) paths between X_i and X_j.

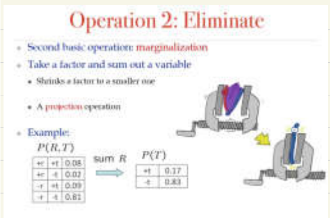
If one or more active, independence not guaranteed.

If All inactive, independence.

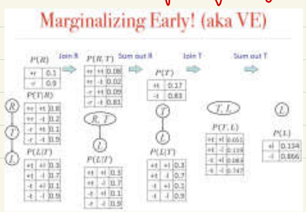


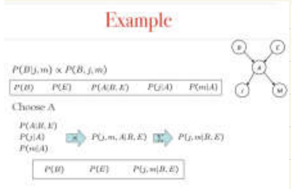
Enumerate Inference

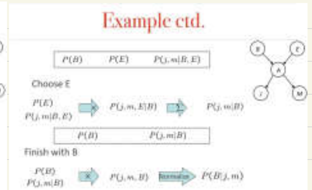


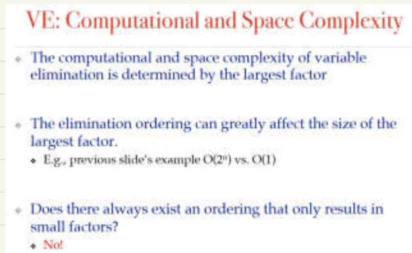


Variable Elimination = Marginalizing Early









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.

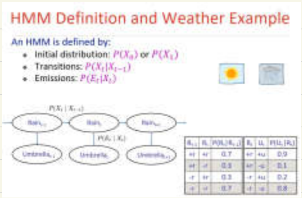
$X_0 \to X_1 \to X_2 \to X_3 \dashrightarrow$

$P(X_0)$. $P(X_t | X_{t-1})$

Stationary Distributions.

The limiting distribution (if exists) is called the stationary distribution P_∞ of the chain. $\begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix}$

Hidden Markov Models.



Joint distribution for Markov Model.

$P(X_0, \dots, X_T) = P(X_0) \prod_{t=1}^{T} P(X_t | X_{t-1})$

Joint distribution for hidden Markov model:

$P(X_0, X_1, \cdots, X_T, E_1, \cdots, E_T) = P(X_0) \prod_{t=1}^{T} P(X_t|X_{t-1}) P(E_t | X_t)$

Filtering: $P(X_t | e_{1:t})$

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

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

Smoothing: $P(X_k | e_{1:t})$ for $0 \le k < t$

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

Forward Algorithm.

$P(X_{t+1} | e_{1:t+1}) = \underset{\text{Normalize}}{\alpha} \underbrace{P(e_{t+1} | X_{t+1})}_{\text{update}} \cdot \underbrace{\sum_{x_t} P(x_t | e_{1:t}) P(X_{t+1} | x_t)}_{\text{Predict}}$

$P(X_{t+1} \| e_{1:t+1}) = P(X_{t+1} \| e_{1:t}, e_{t+1})$	
$= \alpha P(e_{t+1} \| X_{t+1}, e_{1:t}) P(X_{t+1} \| e_{1:t})$	Bayes rule, α normalization
$= \alpha P(e_{t+1} \| X_{t+1}) P(X_{t+1} \| e_{1:t})$	Conditional ind.
$= \alpha P(e_{t+1} \| X_{t+1}) \sum_{x_t} P(x_t \| e_{1:t}) P(X_{t+1} \| x_t, e_{1:t})$	Law of total prob.
$= \alpha P(e_{t+1} \| X_{t+1}) \sum_{x_t} P(x_t \| e_{1:t}) P(X_{t+1} \| x_t)$	Conditional ind.

$f_{1:t+1} = \text{Forward}(f_{1:t}, e_{t+1})$

Particle Filtering.

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

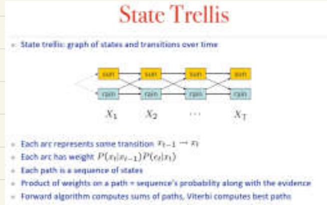
$x' = \text{sample}(P(X'|x))$

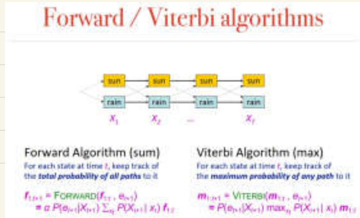
New particles approximates:

$\sum_{x_t} P(X_{t+1} | x_t) f_{1:t}(x_t)$

Most Likely explanation (MLE)

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





Time complex: $O(|X|^2 T)$

Space: $O(|X| T)$

Number of paths. $O(|X|^T)$

Dynamic Bayes' Nets.

Machine learning

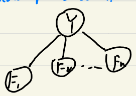
Model-based Classification

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

Naïve Bayes for Digit.

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

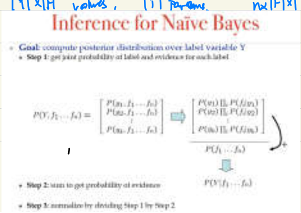
$P(Y | F_{0,0}, \dots, F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j} | Y)$.



general Naive Bayes Model:

$P(Y | F_1 \cdots F_n) = P(Y) \cdot \prod_i P(F_i | Y)$

$|Y| \times |F|^n$ values, $|Y|$ params, $n \times |F| \times |Y|$ params.



Naive Bayes for Text

$P(Y, W_1, \dots, W_n) = P(Y) \cdot \prod_i P(W_i | Y)$ — W_i: Word at position i

Training and Testing.

Data. labelled instances. e.g. emails marked spam / ham.

1. Training set
2. Validation set
3. 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

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

Estimate that maximize the likelihood of data.

$L(x, \theta) = \prod_i P_\theta(x_i) = \theta \cdot \theta \cdot (1-\theta)$

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

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

Maximum Likelihood Estimation.

Data: observed set D of α_H Head and α_T tail.

Hypothesis space: Binomial distributions: $\text{Bin}(\theta, \alpha_H + \alpha_T)$

Learning: finding θ is an optimization problem.

objective function, $P(D|\theta) = \theta^{\alpha_H} \cdot (1-\theta)^{\alpha_T}$

MLE: Choose θ to maximize probability of D.

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

$= \arg\max_\theta \ln P(D|\theta) = \arg\max_\theta \ln \theta^{\alpha_H} (1-\theta)^{\alpha_T}$.

Set derivative to zero, and solve:

$\frac{d}{d\theta} \ln P(D|\theta) = \frac{d}{d\theta} (\ln \theta^{\alpha_H} (1-\theta)^{\alpha_T})$

$= \frac{d}{d\theta} (\alpha_H \ln \theta + \alpha_T \ln(1-\theta))$

$= \alpha_H \cdot \frac{1}{\theta} - \alpha_T \cdot \frac{1}{1-\theta} = \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1-\theta} = 0$

$\Rightarrow \hat{\theta}_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T}$

Smoothing.

Laplace smoothing.

$P_{LAP}(x) = \frac{c(x) + k}{\sum_x (c(x) + k)} = \frac{c(x) + k}{N + k|X|}$, N is total samples,

conditional: $P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$

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.

$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0-\alpha)\hat{P}(x)$

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

Tuning on Validation data.

Two kinds of unknown:

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

Baselines.

Confidences from a classifier.

Posterior over the top label.

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

Discriminative learning.

Linear classifiers.

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

$\text{activation}_w(x) = \sum_i w_i \varphi_i(x) = w \cdot \varphi(x)$

If activation is positive, output +1, negative, output −1

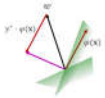
Learning: Binary Perceptron

- Start with weights = 0
- For each training instance (x, y^*):
  - Classify with current weights

$$\hat{y} = \begin{cases} +1 & \text{if } w \cdot \varphi(x) \geq 0 \\ -1 & \text{if } w \cdot \varphi(x) < 0 \end{cases}$$

  - If correct (i.e., $\hat{y} = y^*$), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector. Subtract if y^* is -1.

$$w = w + y^* \cdot \varphi(x)$$



- Maximum likelihood estimation

$$\theta_{ML} = \arg\max_\theta P(\mathbf{X}|\theta) = \arg\max_\theta \prod_i P_\theta(X_i)$$

- Maximum *conditional* likelihood estimation

$$\theta^* = \arg\max_\theta P(\mathbf{Y}|\mathbf{X},\theta) = \arg\max_\theta \prod_i P_\theta(y_i|x_i)$$

$$\ell(w) = \prod_i \frac{e^{w_{y_i} \cdot x_i}}{\sum_y e^{w_y \cdot x_i}}$$

$$\ell\ell(w) = \sum_i \log P_w(y_i|x_i) = \sum_i w_{y_i} \cdot x_i - \log \sum_y e^{w_y \cdot x_i}$$

$$\ell\ell(w) = \sum_i \log P_w(y_i|x_i)$$
$$= \sum_i w_{y_i} \cdot x_i - \log \sum_y e^{w_y \cdot x_i}$$

$$\frac{d}{dw_y} \log P_w(y_i|x_i) = \begin{cases} x_i - x_i \dfrac{e^{w_y \cdot x_i}}{\sum_{y'} e^{w_{y'} \cdot x_i}} & \text{if } y = y_i \\ -x_i \dfrac{e^{w_y \cdot x_i}}{\sum_{y'} e^{w_{y'} \cdot x_i}} & \text{otherwise} \end{cases}$$

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

Optimization Procedure: Gradient Descent

initialize w (e.g., randomly)
repeat for K iterations
  for each example (x_i, y_i)
    compute gradient $\Delta_i = -\nabla_w \log P_w(y_i|x_i)$
  compute gradient $\nabla_w \mathcal{L} = \sum_i \Delta_i$
  $w \leftarrow w - \alpha \nabla_w \mathcal{L}$

$\frac{d}{dw_y} \log P_w(y_i|x_i) = x_i(I(y = y_i) - P(y|x_i))$

- α: learning rate — hyperparameter that needs to be chosen carefully

Stochastic Gradient Descent

initialize w (e.g., randomly)
repeat for K iterations
  for each example (x_i, y_i)
    compute gradient $\Delta_i = -\nabla_w \log P_w(y_i|x_i)$
    $w \leftarrow w - \alpha \Delta_i$

$\frac{d}{dw_y} \log P_w(y_i|x_i) = x_i(I(y = y_i) - P(y|x_i))$

If $y_i = y$, move w_y toward x_i with weight $1 - P(y_i|x_i)$ — probability of incorrect answer

If $y_i \neq y$, move w_y away from x_i with weight $P(y|x_i)$ — probability of incorrect answer

compare this to the multiclass perceptron: probabilistic weighting!

Neural Nets.

Artificial Neuron.

Perceptron (感知机): $g(z) = \text{sign}(z)$

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

Linear regression: $g(z) = z$

g: activation function

How to Learn the Parameters of the Model?

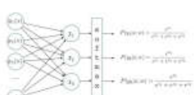
- Iterative method that updates unknown weights
- For perceptron:
  $w \leftarrow w + y^* \varphi(x) \mathbf{1}(w \cdot \varphi(x) \cdot y^* < 0)$
- For logistic regression:
  - Write likelihood function: $P(X, Y|w)$
  - Maximize $\log P(X, Y|w)$ with gradient descent
    $w \leftarrow w + \alpha \nabla_w \log P(X, Y|w)$ (batch)
    $w \leftarrow w + \alpha \nabla_w \log P(x, y|w)$ (stochastic)

General Framework: Statistical ML

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

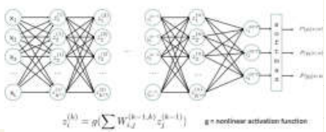
Neural Networks.

Multi-class Logistic Regression



(Deep) Neural Network

- Directly learns the features from data



$$z_i^{(k)} = g\left(\sum_j W_{i,j}^{(k-1,k)} z_j^{(k-1)}\right)$$ g = nonlinear activation function

Common Activation Functions

Summary of Key Ideas

- Minimize empirical risk: $\min_w \sum_i \ell(h_w(x^i), y^i)$
- Continuous optimization
  - Gradient descent:
- Deep neural nets
  - Last layer returns expected output (e.g., probability of classes)
  - Now also many more layers before this last layer
  - Universal function approximation theorem
- 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 Syntax

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

Sentence: → AtomicSentence | ComplexSentence.

Atomic: True | False | Symbol.

Symbol: $X_1 \mid X_2 \mid \dots \mid X_n$

Complex: $\neg$ Sentence | Sentence $\wedge$ Sentence ...

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

Logical Consequences:

Entailment: determine truth of sentences based on semantics.

Inference: generate new sentence from current KB.

Entailment: $\alpha \models \beta$ iff in every world where α is true, β is also true.

model(KB) ⊆ model(query)

Validity and Satisfiability:

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

$\alpha \models \beta$ if and only if $\alpha \Rightarrow \beta$ 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.

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

And elimination. $\frac{\alpha \wedge \beta}{\alpha}$

Biconditional elimination: $\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$

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 ($A \vdash B$)

$\frac{\alpha \vee \beta, \ \neg \beta}{\alpha}$, $\frac{\alpha \vee \beta, \ \neg \beta \vee \gamma}{\alpha \vee \gamma}$, $\frac{\alpha_1 \vee \dots \vee \alpha_n \vee \beta, \ \neg \beta \vee \gamma_1 \vee \dots \vee \gamma_m}{\alpha_1 \vee \dots \vee \alpha_n \vee \gamma_1 \vee \dots \vee \gamma_m}$

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