Candy example

- Favorite candy sold in two flavors: Cherry (yum), Lime (ugh)
- Same wrapper for both flavors
- Sold in bags with different ratios:
  - 100% cherry
  - 75% cherry + 25% lime
  - 50% cherry + 50% lime
  - 25% cherry + 75% lime
  - 100% lime
- You bought a bag of candy but don't know its flavor ratio
- After eating k candies:
  - What's the flavor ratio of the bag?
  - What will be the flavor of the next candy?

Candy example

- Hypothesis H: probabilistic theory of the world
  - h_1: 100% cherry
  - h_2: 75% cherry + 25% lime
  - h_3: 50% cherry + 50% lime
  - h_4: 25% cherry + 75% lime
  - h_5: 100% lime
- Data D: evidence about the world
  - d_1: 1st candy is cherry
  - d_2: 2nd candy is lime
  - d_3: 3rd candy is lime
  - ...

Bayesian Learning

- Prior: $Pr(H)$
- Likelihood: $Pr(d|H)$
- Evidence: $d = \langle d_1, d_2, \ldots, d_n \rangle$
- Computing the posterior using Bayes'Theorem:

$$Pr(H|d) = \alpha Pr(d|H) Pr(H)$$

Bayesian Prediction

- Suppose we want to make a prediction about an unknown quantity X (*i.e.*, the flavor of the next candy)

$$P(X|d) = \sum_i P(X|d, h_i)P(h_i|d) = \sum_i P(X|h_i)P(h_i|d)$$

- Predictions are weighted averages of the predictions of the individual hypotheses
- Hypotheses serve as "intermediaries" between raw data and prediction

Candy Example

- Hypothesis H:
  - h_1: 100% cherry
  - h_2: 75% cherry + 25% lime
  - h_3: 50% cherry + 50% lime
  - h_4: 25% cherry + 75% lime
  - h_5: 100% lime
- Assume prior $P(H) = \langle 0.1, 0.2, 0.4, 0.2, 0.1 \rangle$
- Assume candies are i.i.d. (identically and independently distributed), *i.e.*, $P(d|h) = \Pi_j P(d_j|h)$
- Suppose first 10 candies all taste lime:
  - $P(d|h_5) = 1^{10} = 1$,
  - $P(d|h_3) = 0.5^{10} = 0.00097$
  - $P(d|h_1) = 0^{10} = 0$

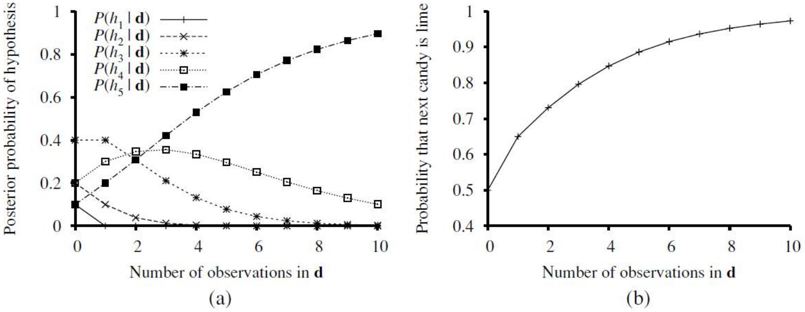


Figure 20.1 (a) Posterior probabilities $P(h_i \mid d_1, \ldots, d_N)$ from Equation (20.1). The number of observations N ranges from 1 to 10, and each observation is of a lime candy. (b) Bayesian prediction $P(d_{N+1} = lime \mid d_1, \ldots, d_N)$ from Equation (20.2).

Bayesian learning properties

- Optimal (*i.e.*, given prior, no other prediction is correct more often than the Bayesian one)
- No overfitting (all hypotheses weighted and considered)
- There is a price to pay:
  - When hypothesis space is large, Bayesian learning may be intractable
  - *i.e.*, sum (or integral) over hypothesis often intractable
- Solution: approximate Bayesian learning

Maximum a posteriori (极大后验,MAP)

- Idea: make prediction based on most probable hypothesis
  - $h_{\mathsf{MAP}} = \mathsf{argmax}_{h_i} P(h_i|d)$
  - $P(X|d) \approx P(X|h_{\mathsf{MAP}})$
- In contrast, Bayesian learning makes prediction based on all hypotheses weighted by their probability

Candy Example (MAP)

- Prediction after
  - 1 lime: $h_{\text{MAP}} = h_3$, $Pr(lime|h_{\text{MAP}}) = 0.5$
  - 2 limes: $h_{\text{MAP}} = h_4$, $Pr(lime|h_{\text{MAP}}) = 0.75$
  - 3 limes: $h_{\text{MAP}} = h_5$, $Pr(lime|h_{\text{MAP}}) = 1$
  - 4 limes: $h_{\text{MAP}} = h_5$, $Pr(lime|h_{\text{MAP}}) = 1$
  - ...
- After only 3 limes, it correctly selects h_5
- But what if correct hypothesis is h_4?
- After 3 limes, MAP incorrectly predicts h_5
  - MAP yields $P(lime|h_{MAP}) = 1$
  - Bayesian learning yields $P(lime|d) = 0.8$

MAP properties

- MAP prediction less accurate than Bayesian prediction since it relies only on one hypothesis h_{MAP}
- But MAP and Bayesian predictions converge as data increases
- Controlled overfitting (prior can be used to penalize complex hypotheses)
- Finding h_{MAP} may be intractable:
  - $h_{MAP} = \text{argmax}_h P(h|d)$
  - Optimization may be difficult

MAP computation

- Optimization:
  - $h_{\text{MAP}} = \text{argmax}_h P(h|d) = \text{argmax}_h P(h)P(d|h) = \text{argmax}_h P(h)\Pi_i P(d_i|h)$
- Product induces non-linear optimization
- Take the log to linearize optimization
  - $h_{\text{MAP}} = \text{argmax}_h \log P(h) + \sum_i \log P(d_i|h)$

Maximum Likelihood (极大似然,ML)

- Idea: simplify MAP by assuming uniform prior (*i.e.*, $P(h_i) = P(h_j)$ for all i, j)
  - $h_{\text{MAP}} = \text{argmax}_h P(h)P(d|h)$
  - $h_{\text{ML}} = \text{argmax}_h P(d|h)$
- Make prediction based on h_{ML} only:
  - $P(X|d) \approx P(X|h_{\text{ML}})$

ML properties

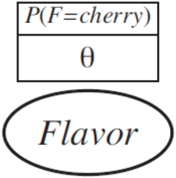
- ML prediction less accurate than Bayesian and MAP predictions since it ignores prior info and relies only on one hypothesis h_{ML}
- But ML, MAP and Bayesian predictions converge as data increases
- Subject to overfitting (no prior to penalize complex hypothesis that could exploit statistically insignificant data patterns)
- Finding h_{ML} is often easier than h_{MAP}
    - $h_{\text{ML}} = \text{argmax}_h \sum_i \log P(d_i|h)$

Statistical Learning

- Use Bayesian Learning, MAP or ML
- Complete data:
  - When data has multiple attributes, all attributes are known
  - Easy
- Incomplete data:
  - When data has multiple attributes, some attributes are unknown
  - Harder

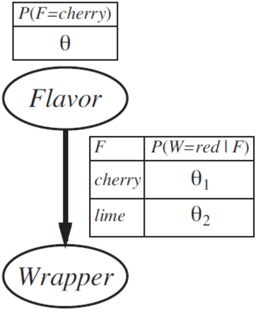
Simple ML example

- Hypothesis h_θ
  - $P(cherry) = \theta$ and $P(lime) = 1 - \theta$
- Data d:
  - c cherries and l limes



- $P(d|h_\theta) = \theta^c(1-\theta)^l$
- $\log P(d|h_\theta) = c \log \theta + l \log(1-\theta)$
- $d(logP(d|h_\theta))/d\theta = c/\theta - l/(1-\theta)$
- $c/\theta - l/(1-\theta) = 0 \Rightarrow \theta = c/(c+l)$

More complicated ML example



- Hypothesis $h_{\theta,\theta_1,\theta_2}$
- Data d:
  - c cherries: g_c green and r_c red
  - l limes: g_l green and r_l red

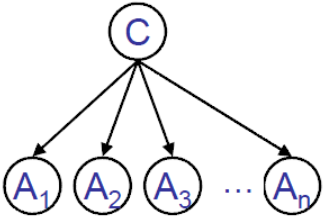
- $P(d|h_{\theta,\theta_1,\theta_2}) = \theta^c(1-\theta)^l\theta_1^{r_c}(1-\theta_1)^{g_c}\theta_2^{r_l}(1-\theta_2)^{g_l}$
- $c/\theta - l/(1-\theta) = 0 \Rightarrow \theta = c/(c+l)$
- $r_c/\theta_1 - g_c/(1-\theta_1) = 0 \Rightarrow \theta_1 = r_c/(r_c+g_c)$
- $r_l/\theta_2 - g_l/(1-\theta_2) = 0 \Rightarrow \theta_2 = r_l/(r_l+g_l)$

Laplace Smoothing

- An important case of overfitting happens when there is no sample for a certain outcome
  - *e.g.*, no cherries eaten so far
  - $P(cherry) = \theta = c/(c+l) = 0$
  - Zero prob. are dangerous: they rule out outcomes
- Solution: Laplace (add-one) smoothing
  - Add 1 to all counts
  - $P(cherry) = \theta = (c+1)/(c+l+2) > 0$
  - Much better results in practice

Naive Bayes models

- Want to predict a class C based on attributes $A_1, \ldots, A_n$
- Parameters:
  - $\theta = P(C = true)$
  - $\theta_{i1} = P(A_i = true | C = true)$
  - $\theta_{i2} = P(A_i = true | C = false)$
- Assumption: A_i's are independent given C



Naive Bayes learning

- Notation: $p = \#(c), n = \#(-c), p_i^+ = \#(c, a_i)$, $n_i^+ = \#(c, -a_i), p_i^- = \#(-c, a_i), n_i^- = \#(-c, -a_i)$
- $P(d|h) = \theta^p (1-\theta)^n \Pi_i \theta_{i1}^{p_i^+} \theta_{i2}^{p_i^-} (1-\theta_{i1})^{n_i^+} (1-\theta_{i2})^{n_i^-}$
- $\theta = p/(p+n)$, $\theta_{i1} = p_i^+/(p_i^+ + n_i^+)$, $\theta_{i2} = p_i^-/(p_i^- + n_i^-)$,
- $P(C|a_1, \ldots, a_n) = \alpha P(C) \Pi_i P(a_i|C)$
- Choose the most likely class

Bayesian network parameter learning (ML)

- Parameters $\theta_{V,pa(V)=\mathbf{v}}$:
  - CPTs: $\theta_{V,pa(V)=\mathbf{v}} = P(V|pa(V)=\mathbf{v})$
- Data **d**:
  - $d_1 : \langle V_1=v_{1,1}, V_2=v_{2,1}, \ldots, V_n = v_{n,1}\rangle$
  - $d_2 : \langle V_1=v_{1,2}, V_2=v_{2,2}, \ldots, V_n = v_{n,2}\rangle$
  - ...
- Maximum likelihood:
  - Set $\theta_{V,pa(V)=\mathbf{v}}$ to the relative frequencies of the values of V given the values **v** of the parents of V

$$\theta_{V,pa(V)=\mathbf{v}} = \#(V,pa(V)=\mathbf{v}) \ / \ \#(pa(V)=\mathbf{v})$$

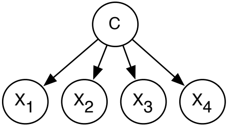
Exercise: Candy example

- Prior $P(H) = \langle 0.1, 0.2, 0.4, 0.2, 0.1 \rangle$
- Evidence $d = \langle lime, cherry, lime \rangle$
- Make predictions using Bayesian, MAP and ML learning

EM algorithm

- Used for soft clustering — examples are probabilistically in classes.
- k-valued random variable C

Model



Data

X_1	X_2	X_3	X_4
t	f	t	t
f	t	t	f
f	f	t	t
	$\cdots$		

➪ Probabilities

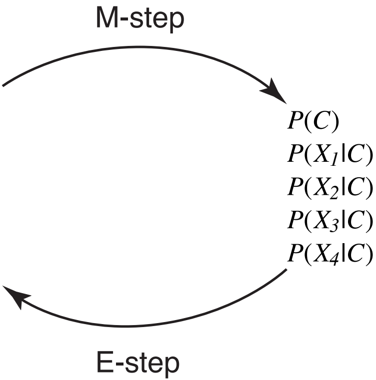
$P(C)$
$P(X_1|C)$
$P(X_2|C)$
$P(X_3|C)$
$P(X_4|C)$

EM Algorithm Overview

- Repeat the following two steps:
  - E-step give the expected number of data points for the unobserved variables based on the given probability distribution.
  - M-step infer the (maximum likelihood or maximum aposteriori probability) probabilities from the data.
- Start either with made-up data or made-up probabilities.
- EM will converge to a local maxima.

EM algorithm

X_1	X_2	X_3	X_4	C	*count*
⋮	⋮	⋮	⋮	⋮	⋮
t	f	t	t	1	0.4
t	f	t	t	2	0.1
t	f	t	t	3	0.5
⋮	⋮	⋮	⋮	⋮	⋮



Augmented data – E step

Suppose $k = 3$, and $dom(C) = \{1, 2, 3\}$.
$P(C = 1|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.407$
$P(C = 2|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.121$
$P(C = 3|X_1 = t, X_2 = f, X_3 = t, X_4 = t) = 0.472$:

X_1	X_2	X_3	X_4	*Count*
⋮	⋮	⋮	⋮	⋮
t	f	t	t	100
⋮	⋮	⋮	⋮	⋮

$\longrightarrow$

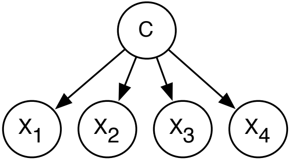
$A[X_1, \ldots, X_4, C]$

X_1	X_2	X_3	X_4	C	*Count*
⋮	⋮	⋮	⋮	⋮	⋮
t	f	t	t	1	40.7
t	f	t	t	2	12.1
t	f	t	t	3	47.2
⋮	⋮	⋮	⋮	⋮	⋮

M step

X_1	X_2	X_3	X_4	C	$Count$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
t	f	t	t	1	40.7
t	f	t	t	2	12.1
t	f	t	t	3	47.2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

$\longrightarrow$

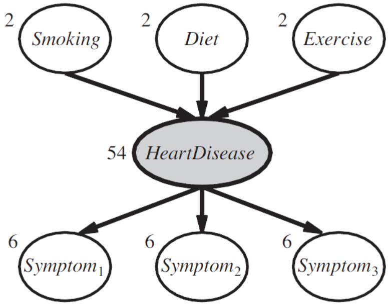


$$P(C{=}v_i) = \frac{\sum_{t \models C=v_i} Count(t)}{\sum_t Count(t)}$$

$$P(X_k = v_j | C{=}v_i) = \frac{\sum_{t \models C=v_i \wedge X_k=v_j} Count(t)}{\sum_{t \models C=v_i} Count(t)}$$

Learning with hidden variables

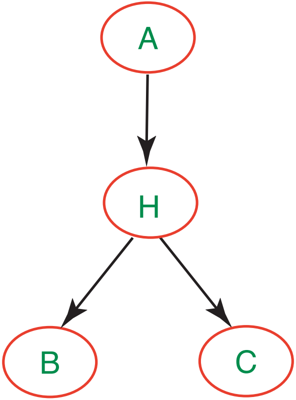
- Many real-world problems have hidden (a.k.a latent) variables



A simple diagnostic network for heart disease

- Hidden variables complicate the learning problem.

A simple example



- What if we had only observed values for A, B, C?

A	B	C
t	f	t
f	t	t
t	t	f
	$\dots$	

EM algorithm

Augmented Data

A	B	C	H	$Count$
t	f	t	t	0.7
t	f	t	f	0.3
f	t	t	f	0.9
f	t	t	t	0.1
	$\ldots$			$\ldots$

E-step

M-step

Probabilities

$P(A)$
$P(H|A)$
$P(B|H)$
$P(C|H)$