

Chap. 20

Learning Probabilistic Models

How does the agent learn the probability theory of the world from experience to handle uncertainty?

Outline

- Bayesian learning
 - Maximum A Posteriori (MAP) and Maximum Likelihood (ML) learning
- Methods for learning probability models: Bayesian network
 - ML parameter learning with Complete Data: discrete models
 - Naïve Bayes models
 - Linear regression: continuous models
- Learning with Hidden Variables: EM algorithm

Introduction

- View learning as a form of *uncertain reasoning* from observations
- Learn the probabilistic theories of the world from experience, prior to handling uncertainty.
- How to formulate the learning task as a process of probabilistic inference?
- Bayesian view of learning is powerful for the problems of noise, overfitting, optimal prediction.
- A method for learning probability models – Bayesian networks.

Statistical Learning

- Data vs. Hypotheses
 - **Data**: evidence, i.e. instantiations of some/all of the random variables describing the domain.
 - **Hypotheses**: probabilistic theories of how the domain works, including logical theories as a special case.
- **H** is the hypothesis variable, values h_1, h_2, \dots , prior $P(H)$.
- **D** : all the data, j^{th} observation d_j gives the outcome of random variable D_j , training data $\mathbf{d} = d_1, \dots, d_N$
- Bayesian learning:
 - calculates the *probability of each hypothesis, given the data*
 - makes *predictions* on that basis.
 - In this way, learning is reduced to probabilistic inference.

Full Bayesian Learning

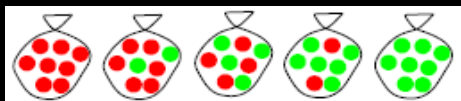
- View learning as Bayesian updating of a probability distribution over the **hypothesis space**.
 - H is the hypothesis variable, values h_1, h_2, \dots , prior $P(H)$.
 - j^{th} observation d_j gives the outcome of random variable D_j
training data $\mathbf{d} = d_1, \dots, d_N$
- Given the data so far, each hypothesis has a **posterior probability**:


$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i) P(h_i)$$
 where $P(\mathbf{d}|h_i)$ is called the **likelihood** of the data under each hypothesis and $P(h_i)$ is the **hypothesis prior**.
- Predictions** (a/b an unknown quantity X) are **weighted average** over the predictions of the individual hypotheses:

$$P(X|\mathbf{d}) = \sum_i P(X|\mathbf{d}, h_i) P(h_i|\mathbf{d}) = \sum_i P(X|h_i) P(h_i|\mathbf{d})$$
- No need to pick one best-guess hypothesis!

Example

- Suppose there are five kinds of bags of candies:
 - 10% are h_1 : 100% cherry candies
 - 20% are h_2 : 75% cherry candies + 25% lime candies
 - 40% are h_3 : 50% cherry candies + 50% lime candies
 - 20% are h_4 : 25% cherry candies + 75% lime candies
 - 10% are h_5 : 100% lime candies



- Then we observe candies drawn from some bag: 
- What kind of bag is it? What flavor will the next candy be?
-- to infer a theory of its world.

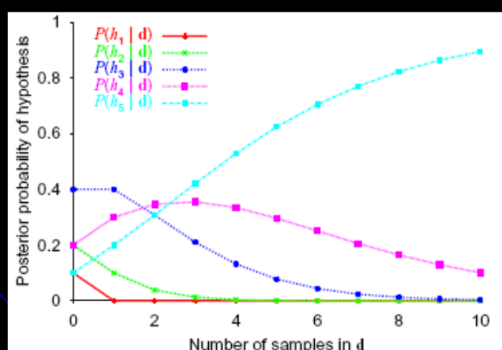
Example

- $P(H) = \langle p(h_1), p(h_2), \dots, p(h_5) \rangle = \langle .1, .2, .4, .2, .1 \rangle$
- The **likelihood of the data** is calculated under the assumption that the observations are i.i.d. (independently and identically distributed), so

$$P(\mathbf{d}|h_i) = \prod_j P(d_j|h_i)$$
- Suppose that the bag is really an all-lime bag (h_5) and the 1st 10 candies are all lime; then $P(\mathbf{d}|h_5) = 0.5^{10}$
- How the posterior probabilities of the 5 hypotheses changes as the sequence of 10 lime candies is observed: Figure (slide 8)
 - The posterior probability of any false hypothesis will eventually vanish.
 - $P(d_{N+1} = \text{lime} | \mathbf{d}) \rightarrow 1$: prediction prob. (slide 9)

$$P(X|\mathbf{d}) = \sum_i P(X|\mathbf{d}, h_i) P(h_i|\mathbf{d}) = \sum_i P(X|h_i) P(h_i|\mathbf{d})$$
 e.g.) $P(d_{11} = \text{lime} | d_{1-10}) = .973025$
 - The true hypothesis eventually dominates the Bayesian prediction.
- The Bayesian prediction is optimal, whether the data set be small or large.
- For real learning problems, the hypothesis space is usually very large or infinite. – Summing over the hypothesis space is often intractable.

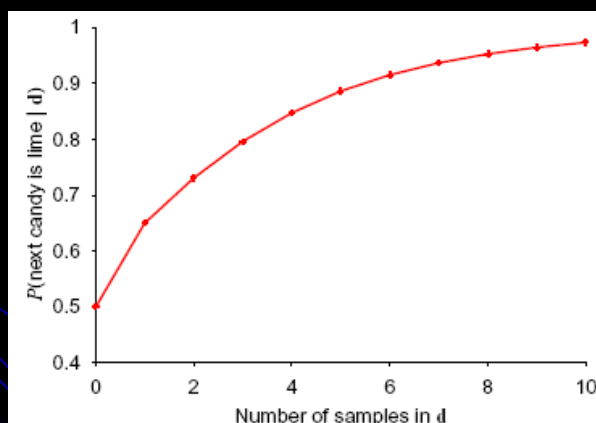
Posterior probability of hypotheses



The posterior probability of any false hypothesis will eventually vanish.

$$P(h_i|\mathbf{d}) = \alpha P(\mathbf{d}|h_i) P(h_i)$$

Prediction probability



$$P(d_{N+1} = \text{lime} \mid \mathbf{d}) \rightarrow 1$$

The true hypothesis eventually dominates the Bayesian prediction.

MAP Approximation

- Summing over the hypothesis space is often intractable
→ a need of approximation or simplification.
- Maximum A Posteriori (MAP)** learning: choose h_{MAP} maximizing $P(h_i | \mathbf{d})$
-- make predictions based on a **single most probable hypothesis** (science).
- $P(X | \mathbf{d}) \approx P(X | h_{MAP})$:
predictions by an MAP hypothesis are approximately Bayesian prediction.
- i.e. maximize $P(\mathbf{d} | h_i)P(h_i)$ of $\log P(\mathbf{d} | h_i) + \log P(h_i)$
i.e. minimize $-\log P(\mathbf{d} | h_i) - \log P(h_i)$
- Log terms can be viewed as (negative of)
bits to encode data given hypothesis + bits to encode hypothesis
-- no bits are required if the hypothesis predicts the data exactly.
- Thus, MAP learning is choosing the hypothesis that provides *maximum compression* of the data.
- This is the basic idea of **minimum description length (MDL)** learning
-- to minimize the size of hypothesis and data encodings.
- For deterministic hypotheses, $P(\mathbf{d} | h_i)$ is $\begin{cases} 1 & \text{if consistent,} \\ 0 & \text{otherwise} \end{cases}$
⇒ MAP = simplest consistent hypothesis

ML Approximation

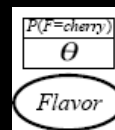
- Assume that a uniform prior over the hypothesis space.
 - For large data sets, prior becomes irrelevant.
- **Maximum Likelihood (ML)** learning: choose h_{ML} maximizing $P(d|h_i)$
- i.e. simply get the best fit to the data; identical to MAP for uniform prior (which is reasonable if all hypotheses are of the same complexity)
- ML is the “standard” (non-Bayesian) statistical learning method.
 - a good approximation to Bayesian and MAP learning for a *large data* sets.

Learning with Complete Data

- **Density Estimation**: the general task of learning a probability model, given data that are assumed to be generated from that model.
- **Complete data**: data are complete when each data point contains values for every variable in the probability model being learned.
- **Parameter learning** : find the numerical parameters for a probability model whose structure is fixed.
- **Learning with Hidden variables**: parameter learning with incomplete data.

ML Parameter Learning in Bayes nets: discrete models

- Bag from a new manufacturer; fraction θ of cherry candies?
- Any θ is possible: continuum of hypotheses h_θ
- θ is a **parameter** for this simple (binomial) family of models
- Suppose we unwrap N candies, c cherries and $\ell = N - c$ limes.
- These are i.i.d. (independent, identically distributed) observations, so



$$P(\mathbf{d}|h_\theta) = \prod_{j=1}^N P(d_j|h_\theta) = \theta^c \cdot (1 - \theta)^\ell$$

i.e. simply get the best fit to the data; identical to MAP for uniform prior (which is reasonable if all hypotheses are of the same complexity)

- To get h_{ML} , maximize this w.r.t. θ - which is easier for the **log-likelihood**:

$$L(\mathbf{d}|h_\theta) = \log P(\mathbf{d}|h_\theta) = \sum_{j=1}^N \log P(d_j|h_\theta) = c \log \theta + \ell \log(1 - \theta)$$

$$\frac{dL(\mathbf{d}|h_\theta)}{d\theta} = \frac{c}{\theta} - \frac{\ell}{1 - \theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c + \ell} = \frac{c}{N}$$

- h_{ML} asserts that the actual proportion of cherries in the bag (θ) = the observed proportion in the candies unwrapped so far (c/N).

ML Parameter Learning in Bayes nets: Discrete models

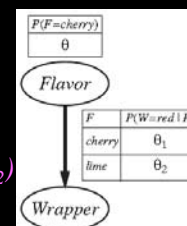
- Standard method for ML parameter learning:
 1. Write down an expression for the likelihood of the data as a function of the parameter(s).
 2. Convert it to the Log-likelihood of data.
 3. Write down the derivative of the log likelihood w.r.t. each parameter.
 4. Find the parameter values s.t. the derivatives = 0.

Find the parameter value which maximize the log likelihood of data.
- Seems sensible, but causes problems with 0 counts!
i.e. when the data set is small enough that some events have not yet been observed,
the maximum likelihood hypothesis assigns 0 probability to those events.
- Trick: initialize the counts for each event to 1, instead of 0.

Multiple parameter Learning: contd.

- Red/green wrapper depends probabilistically on flavor.
- Likelihood for, e.g., cherry candy in green wrapper:

$$\begin{aligned} P(F=\text{cherry}, W=\text{green} \mid h_{\theta, \theta_1, \theta_2}) \\ = P(F=\text{cherry} \mid h_{\theta, \theta_1, \theta_2}) P(W=\text{green} \mid F=\text{cherry}, h_{\theta, \theta_1, \theta_2}) \\ = \theta (1 - \theta_1) \end{aligned}$$



- N candies, r_c red-wrapped cherry candies, etc.:

$$P(\mathbf{d} \mid h_{\theta, \theta_1, \theta_2}) = \theta^c (1 - \theta)^l \cdot \theta_1^{r_c} (1 - \theta_1)^{g_c} \cdot \theta_2^{r_l} (1 - \theta_2)^{g_l}$$

$$\begin{aligned} L = [c \log \theta + l \log (1 - \theta)] \\ + [r_c \log \theta_1 + g_c \log (1 - \theta_1)] \\ + [r_l \log \theta_2 + g_l \log (1 - \theta_2)] \end{aligned}$$

Multiple parameters: contd.

- Derivatives of contain only the relevant parameter.

$$\frac{\partial L}{\partial \theta} = \frac{c}{\theta} - \frac{\ell}{1 - \theta} = 0 \quad \Rightarrow \quad \theta = \frac{c}{c + \ell}$$

$$\frac{\partial L}{\partial \theta_1} = \frac{r_c}{\theta_1} - \frac{g_c}{1 - \theta_1} = 0 \quad \Rightarrow \quad \theta_1 = \frac{r_c}{r_c + g_c}$$

$$\frac{\partial L}{\partial \theta_2} = \frac{r_l}{\theta_2} - \frac{g_l}{1 - \theta_2} = 0 \quad \Rightarrow \quad \theta_2 = \frac{r_l}{r_l + g_l}$$

- The solution for $\theta_1(\theta_2)$: the probability that a cherry (lime) candy has a red wrapper is the observed fraction of cherry (lime) candies with red wrappers.
- With **complete data**, **parameters can be learned separately**
- For the parameter values for a variable, given its parents, are just the observed frequencies of the variable values for each setting of the parent values.
- Still be careful with 0 count!

Naïve Bayes Models

- The most common Bayesian Network model
- The class variable C (to be predicted): the root,
The attribute variables X_i : the leaves
- Assumes the attributes are conditionally independent of each other, given the class – the *naïve* model.
- $\theta = P(C=true), \theta_{i1} = P(X_i=true/C=true), \theta_{i2} = P(X_i=true/C=false).$
- E.g.) $\theta = P(F=cherry),$
 $\theta_{11} = P(Wrapper=red \mid F=cherry),$
 $\theta_{12} = P(Wrapper=red \mid F=lime).$
- Use it to classify new examples (test data) for which C is unobserved:

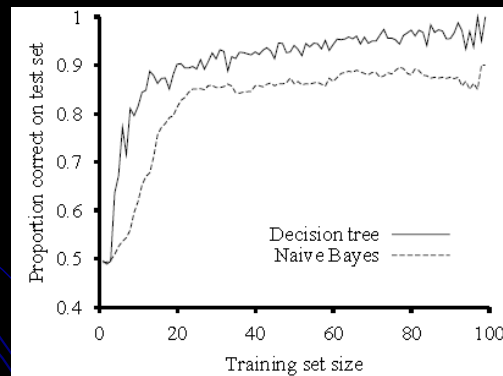
$$P(C/x_1, \dots, x_n) = \alpha \cdot P(C) \cdot \prod_i P(x_i/C)$$
- $c_{NB} = \underset{c_j}{\operatorname{argmax}} P(c_j) \prod_i P(x_i/c_j)$: the most likely class.
- Prediction will be made based on the most likely class.

Naïve Bayes Models

- Whenever the assumption of conditional independence is satisfied, Naïve Bayes classification c_{NB} = MAP classification, h_{MAP}
- Difference b/t NB learning vs. Other learning methods:
 - NB learning scales well to very large problems:
with n boolean attributes, there are just $2n+1$ parameters.
 - No explicit search for h_{ML} through the hypotheses space
-- the maximum-likelihood naïve Bayes hypothesis.
 - Its hypotheses space is the space of possible assignable values to $P(c_j)$ and $P(x_i/c_j)$ terms.
 - The hypothesis is formed by counting the frequency of various data combinations w/i the training examples.
 - No difficulty with noisy data and can give probabilistic predictions when appropriate.

Naïve Bayes Models

- Prediction result is not as well as decision tree learning because the true hypothesis is not representable exactly using a Naïve Bayes model.



Example: Naïve Bayes Models

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

TABLE 3.2

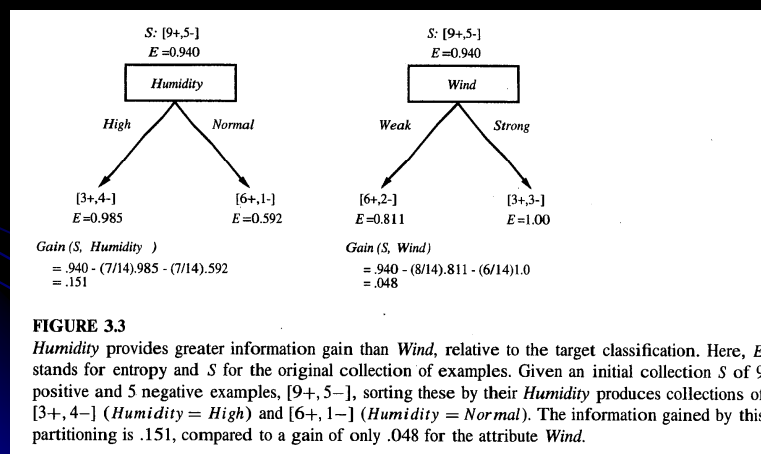
Training examples for the target concept *PlayTennis*.

Example: Naïve Bayes Models

- Use it to classify new examples (test data) for which C is unobserved:
- $P(C/x_1, \dots, x_n) = \alpha P(C) \prod_i P(x_i/C)$
- $c_{NB} = \operatorname{argmax}_{c_j} P(c_j) \prod_i P(x_i/c_j)$ - most likely class
- $\langle x_1, x_2, x_3, x_4 \rangle$
 $= \langle \text{Outlook}=\text{sunny}, \text{Temperature}=\text{cool}, \text{Humidity}=\text{high}, \text{Wind}=\text{strong} \rangle$ -- *
- Task: to predict the target value of *PlayTennis* for this new instance.
- $c_{NB} = \operatorname{argmax}_{c_j} P(c_j) \prod_i P(x_i/c_j)$
 $= \operatorname{argmax}_{c_j} P(c_j) \cdot P(\text{Outlook}=\text{sunny}/c_j) \cdot P(\text{Temperature}=\text{cool}/c_j)$
 $\quad \cdot P(\text{Humidity}=\text{high}/c_j) \cdot P(\text{Wind}=\text{strong}/c_j)$
 $P(\text{PlayTennis}=\text{yes}) = 9/14 = .64; \quad P(\text{PlayTennis}=\text{no}) = 5/14 = .36$
 $P(\text{Wind}=\text{strong}|\text{PlayTennis}=\text{yes}) = 3/9 = .33;$
 $P(\text{Wind}=\text{strong}|\text{PlayTennis}=\text{no}) = 3/5 = .60. \quad \text{etc.}$
 So, $P(\text{yes}) \cdot P(\text{sunny}|\text{yes}) \cdot P(\text{cool}|\text{yes}) \cdot P(\text{high}|\text{yes}) \cdot P(\text{strong}|\text{yes}) = .0053;$
 $P(\text{no}) \cdot P(\text{sunny}|\text{no}) \cdot P(\text{cool}|\text{no}) \cdot P(\text{high}|\text{no}) \cdot P(\text{strong}|\text{no}) = .0206.$
 Thus, $c_{NB} = (\text{PlayTennis}=\text{no})$ for the above new instance *.
 Normalize it; $P(\text{PlayTennis}=\text{no} | x_1, x_2, x_3, x_4) = .0206 / (.0206 + .0053) = .795.$

Example: Decision Tree Learning

Which attribute is the best classifier?



Example: Decision Tree Learning

Which attribute is the best classifier at the root?

$$H(<9/14, 5/14>) = -9/14 \log_2 9/14 - 5/14 \log_2 5/14 = 0.94$$

$$\begin{aligned} \text{Gain (Outlook)} &= H(<9/14, 5/14>) - \text{Remainder(Outlook)} \\ &= 0.94 - [5/14 \cdot H(<2/5, 3/5>) + 4/14 \cdot H(<4/4, 0/4>) + 5/14 \cdot H(<3/5, 2/5>)] \\ &= 0.246; \end{aligned}$$

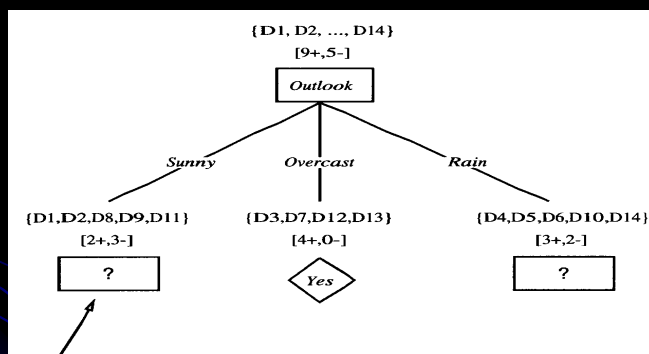
$$\begin{aligned} \text{Gain (Humidity)} &= 0.94 - \text{Remainder(Humidity)} \\ &= 0.94 - [7/14 \cdot H(<3/7, 4/7>) + 7/14 \cdot H(<6/7, 1/7>)] \\ &= 0.151; \end{aligned}$$

$$\begin{aligned} \text{Gain (Wind)} &= 0.94 - \text{Remainder(Wind)} \\ &= 0.94 - [8/14 \cdot H(<6/8, 2/8>) + 6/14 \cdot H(<3/6, 3/6>)] \\ &= 0.94 - [8/14 \cdot (-6/8 \log_2 6/8 - 2/8 \log_2 2/8) \\ &\quad + 6/14 \cdot (-3/6 \log_2 3/6 - 3/6 \log_2 3/6)] \\ &= 0.94 - [8/14 \cdot 0.811 + 6/14 \cdot 1] \\ &= 0.048; \end{aligned}$$

$$\text{Gain (Temperature)} = 0.0029$$

Example: Decision Tree Learning

Which attribute is the best classifier at the subtree of <outlook=Sunny>?



$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

Example: Decision Tree Learning

Which attribute is the best classifier at the subtree of <outlook=Sunny>?

$$H(<2/5, 3/5>) = -2/5 \log_2 2/5 - 3/5 \log_2 3/5 = 0.97$$

$$\begin{aligned} \text{Gain (Humidity)} &= 0.97 - \text{Remainder(Humidity)} \\ &= 0.97 - [3/5 H(<3/3, 0/3>) + 2/5 H(<2/2, 0/2>)] = 0.97; \end{aligned}$$

$$\begin{aligned} \text{Gain (Wind)} &= 0.97 - \text{Remainder(Wind)} \\ &= 0.97 - [3/5 H(<1/3, 2/3>) + 2/5 H(<1/2, 1/2>)] \\ &= 0.19; \end{aligned}$$

$$\begin{aligned} \text{Gain (Temperature)} &= 0.97 - \text{Remainder(Temperature)} \\ &= 0.97 - [2/5 H(<0/2, 2/2>) + 2/5 H(<1/2, 1/2>) + 1/5 H(<1/1, 0/1>)] \\ &= 0.570 \end{aligned}$$

Which attribute is the best classifier at the subtree of <outlook=Rain>?

$$H(<3/5, 2/5>) = -3/5 \log_2 3/5 - 2/5 \log_2 2/5 = 0.97$$

$$\begin{aligned} \text{Gain (Wind)} &= 0.97 - \text{Remainder(Wind)} \\ &= 0.97 - [3/5 H(<3/3, 0/3>) + 2/5 H(<0/2, 2/2>)] = 0.97; \end{aligned}$$

$$\begin{aligned} \text{Gain (Temperature)} &= 0.97 - \text{Remainder(Temperature)} \\ &= 0.97 - [0/5 H(<0/2, 2/2>) + 3/5 H(<2/3, 1/3>) + 2/5 H(<1/2, 1/2>)] \end{aligned}$$

Example: Decision Tree Learning

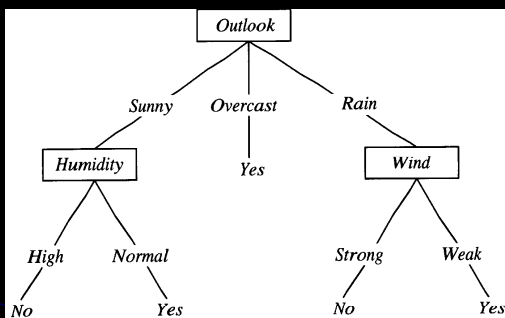


FIGURE 3.1

A decision tree for the concept *PlayTennis*. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, *Yes* or *No*). This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

$$h = (\text{Outlook}=\text{Sunny} \wedge \text{Humidity}=\text{Normal}) \vee (\text{Outlook}=\text{Overcast}) \vee (\text{Outlook}=\text{Rain} \wedge \text{Wind}=\text{Weak})$$

For a test data <Outlook=sunny, Temperature=cool, Humidity=high, Wind=strong>,

PlayTennis=NO – same prediction as the Naïve Bayes Model

ML Parameter Learning: Continuous models

- How to learn continuous models from data?
- Learning the parameters of a Gaussian density function on a single variable
i.e. the data are

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- The parameters: μ (mean) and σ (standard deviation).
- Let the observed values be x_1, x_2, \dots, x_N .
- The Log-Likelihood is:

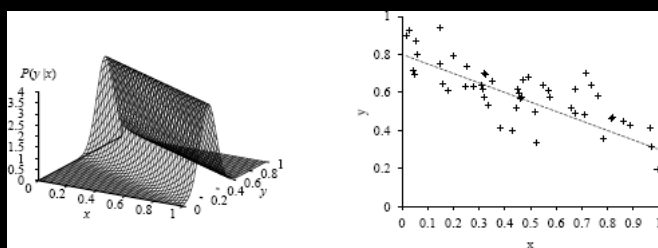
$$L = \sum_{j=1}^N \log \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_j-\mu)^2}{2\sigma^2}} = N(-\log \sqrt{2\pi} - \log \sigma) - \sum_{j=1}^N \frac{(x_j - \mu)^2}{2\sigma^2}$$

- Set the derivatives to zero:

$$\begin{aligned} \frac{\partial L}{\partial \mu} &= -\frac{1}{\sigma^2} \sum_{j=1}^N (x_j - \mu) = 0 & \Rightarrow \mu &= \frac{\sum_{j=1}^N x_j}{N} \\ \frac{\partial L}{\partial \sigma} &= -\frac{N}{\sigma} + \frac{1}{\sigma^3} \sum_{j=1}^N (x_j - \mu)^2 = 0 & \Rightarrow \sigma &= \sqrt{\frac{\sum_{j=1}^N (x_j - \mu)^2}{N}} \end{aligned}$$

the maximum-likelihood value of the mean(μ) is the sample average
the maximum-likelihood value of the standard deviation(σ) is the square root of the sample variance.

Example: linear Gaussian model



- Consider a linear Gaussian model with 1 continuous parent X and a child Y.
- To learn $P(Y|X)$, find the maximum-likelihood values of the parameters:

- Maximizing $P(y|x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-(\theta_1 x + \theta_2))^2}{2\sigma^2}}$ w.r.t. θ_1, θ_2

$$= \text{minimizing } E = \sum_{(j=1..N)} (y_j - (\theta_1 x_j + \theta_2))^2$$

E = the **sum of squared errors** between the actual value y and the prediction $\theta_1 x_j + \theta_2$
– the quantity that is minimized by the standard linear regression procedure.

- Minimizing the sum of squared errors gives the ML solution for a linear fit
assuming Gaussian noise of fixed variance of data:

Example: linear Gaussian model

- minimizing $E = \sum_{j=1..N} (y_j - (\theta_1 x_j + \theta_2))^2$
 E = the **sum of squared errors** between the actual value y
 and the prediction $\theta_1 x_j + \theta_2$
 which is the quantity that is minimized by the standard linear regression procedure:
 e.g.) least squares fitting.

$$\frac{\partial E}{\partial \theta_1} = -2 \sum_{j=1}^N (y_j - (\theta_1 x_j + \theta_2)) x_j = 0$$

$$\frac{\partial E}{\partial \theta_2} = -2 \sum_{j=1}^N (y_j - (\theta_1 x_j + \theta_2)) = 0$$

$$\theta_1 = \frac{\sum_{j=1}^N x_j y_j - N \bar{x} \bar{y}}{\sum_{j=1}^N x_j^2 - N \bar{x}^2} = \frac{N \text{cov}(x, y)}{N \sigma_x^2}$$

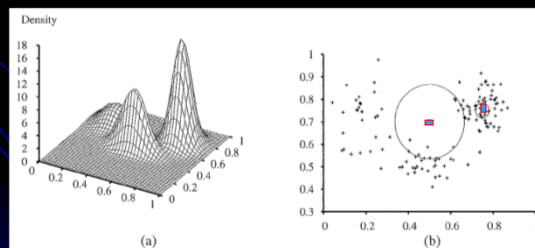
$$\theta_2 = \frac{\bar{y} (\sum_{j=1}^N x_j^2) - \bar{x} \sum_{j=1}^N x_j y_j}{\sum_{j=1}^N x_j^2 - N \bar{x}^2} = \bar{y} - \theta_1 \bar{x}$$

where

$$\bar{x} = \sum_{j=1}^N x_j \quad \text{and} \quad \bar{y} = \sum_{j=1}^N y_j$$

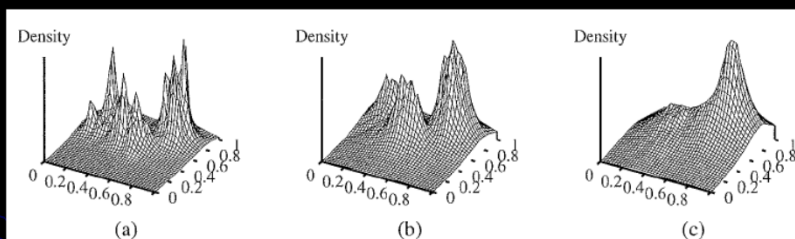
Density Estimation with Nonparametric Models

- Learn a probability model with no assumption about its structure and parameterization.
- Nonparametric density estimation in continuous domain.
- K-nearest-neighbors model:
 - For the estimation of probability density at a query point x , measure the density of data points in the neighborhood of x .
- Example: A pdf on a space defined by 2 continuous variables.
 2 query points.



Density Estimation with Nonparametric Models

- 3 plots of density estimation using k-nearest neighbors for k=3, 10, 40.

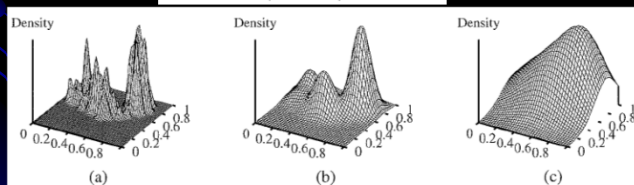


- (a) too spiky (k is too small)
- (b) k is about right
- (c) too smooth (k is too large).
- The best value of k can be chosen by cross validation.

Cont.

- Use kernel function.
- To apply a kernel model to density estimation, assume that each data point generates its own little density function, using a Gaussian kernel.
- The estimated density at a query point \mathbf{x} is the average density as given by each kernel function:
$$\hat{P}(\mathbf{x}) = \frac{1}{N} \sum_{j=1}^N \mathcal{K}(\mathbf{x}, \mathbf{x}_j)$$
- Assume spherical Gaussian with standard deviation w along each axis: d is # of dimensions in \mathbf{x} , D is the Euclidean distance function.

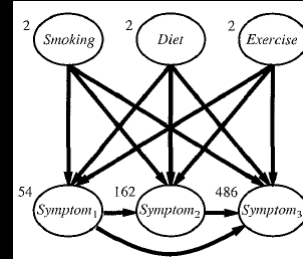
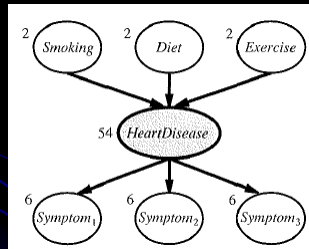
$$\mathcal{K}(\mathbf{x}, \mathbf{x}_j) = \frac{1}{(w^2 \sqrt{2\pi})^d} e^{-\frac{D(\mathbf{x}, \mathbf{x}_j)^2}{2w^2}}$$

(a) $w = 0.02$ (b) $w = 0.07$ (c) $w = 0.2$

- Choose a suitable value for kernel width w by cross validation.

Learning with Hidden Variables: EM Algorithm

- Problems: Hidden Factors
 - Unobservable / Latent / Hidden
 - Make them as variables → Simplicity of the model



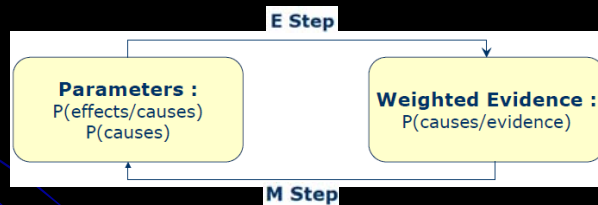
- Assume that each variable has 3 possible values: 78 parameters → 708 parameters.
- Hidden variables can dramatically reduce the # of parameters required to specify a BN; thus, it reduces the amount of data need to learn the parameters.
- Solution: EM algorithm

EM Algorithm: the general form

- Iterative Parameter Estimation Method in 2 steps:
 - Expectation – Maximization
- Not based on the gradient calculations as in stochastic approximation methods.
- EM is used to estimate *parameters of a mixture model* via ML.
- Learn the distribution of the *hidden variable*.
- Principle: the general form
 - Given: Cause (or Factor), Evidence
 - Compute: Probability distribution. Parameters in Hidden variables.
- 2 steps:
 - E Step: Compute expectation of the complete data log likelihood .
 - M step: Find the parameters that maximize the expected complete data log likelihood.
- No guarantee that the solution is the global maximum; but, a slow convergence on many problems.

EM Algorithm: the general form

- 2 steps:
 - E Step: *Compute expectation of the complete data log-likelihood .*
For each evidence (E),
use the current estimate for parameters to compute probability distribution.



- M step: *Find the parameters that maximize the expected complete data log-likelihood.*
Update the estimates of parameters based on weighted evidence.

EM Algorithm: the general form

- An iterative method for finding *maximum likelihood(ML) estimates of parameters* in probability model which depends on unobserved hidden variables. The algorithm alternates between performing an *Expectation (E) step* and a *Maximization (M) step*.

$$\theta^{(i+1)} = \underset{\theta}{\operatorname{argmax}} \sum_{\mathbf{z}} P(\mathbf{Z}=\mathbf{z} | \mathbf{x}, \theta^{(i)}) L(\mathbf{x}, \mathbf{Z}=\mathbf{z} | \theta) .$$

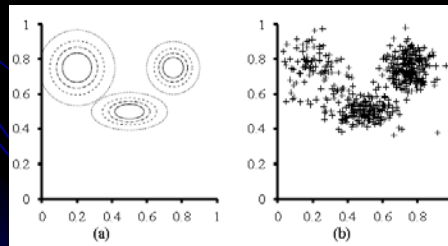
- E Step:
 - Perception step
 - Using the current estimate for the parameter,
Computes the expectation of the log-likelihood of the completed data w.r.t. $P(\mathbf{Z}=\mathbf{z} | \mathbf{x}, \theta^{(i)})$, the posterior over the hidden variables given the data.
- M step:
 - Learning step
 - Computes parameters maximizing the expected log-likelihood found on the E step.
 - Use these parameter-estimates to determine the distribution of the hidden variables in the next E-step.

EM: Learning Mixtures of Gaussians

- Issues in Mixture of Gaussian
 - Unsupervised Clustering :
 - A problem of discerning multiple *unknown* categories in a collection of objects.
 - Set of data points (Evidences)
 - Data generated from mixture distribution (P) where a distribution has k components (C):

$$P(\mathbf{x}) = \sum_{i=1}^k P(C=i) P(\mathbf{x} | C=i)$$

- Continuous data: Mixture of Gaussians
 - What kind of probability distribution might have generated the data?

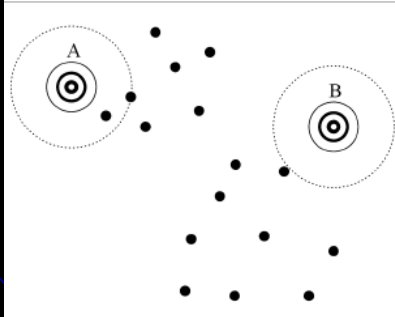


EM: Gaussian Mixture Model (cont.)

- What For?
 - Find parameters of each component: $P(\mathbf{x}) = \sum_{i=1}^k P(C=i) P(\mathbf{x} | C=i)$
 - (mixing) Weight : $w_i = P(C=i)$ - weight of each component i .
 - Mean: μ_i
 - Covariance: Σ_i
- How?
 - Guess the probability distribution with the parameters
 - Infer the probability that each data point belongs to each component : E-step
 - Refit the components to the data (i.e. recompute the parameters) : M-step
 - Iterate this process until convergence.
 - Complete the data by inferring probability distributions over the *hidden variables* – which *component* each data point belongs to – based on the current model.

EM Process: GMM

- Initialization:
 - initialize the parameters by random values.



EM Process: GMM

- E-step: Expectation step**

Compute the expected values p_{ij} of the hidden indicator variables Z_{ij} where $z_{ij} = \begin{cases} 1 & \text{if datum } \mathbf{x}_j \text{ was generated by component-}i. \\ 0 & \text{o.w.} \end{cases}$

 - Compute the probabilities $p_{ij} = P(C=i/\mathbf{x}_j)$.
 - the probability that datum \mathbf{x}_j was generated by component- i .
$$p_{ij} = P(C=i/\mathbf{x}_j) = \alpha \frac{P(\mathbf{x}_j/C=i)P(C=i)}{\sum_k P(\mathbf{x}_j/C=k)P(C=k)}$$

where
 $P(\mathbf{x}_j/C=i)$: the probability at \mathbf{x}_j of the i -th Gaussian
 $P(C=i)$: the weight parameter for the i -th Gaussian.

Define $n_i = \sum_j z_{ij}$, the effective # of data points currently assigned to component i .

EM Process: GMM

- **M-step: Maximization**

Find the new values of the parameters that maximize the log likelihood of the data, given the expected values of the hidden indicator variables.

- i.e. Compute the new mean, covariance, and component weights:

$$\begin{aligned}\mu_i &\leftarrow \sum_j p_{ij} \mathbf{x}_j / n_i \\ \Sigma_i &\leftarrow \sum_j p_{ij} (\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T / n_i \\ w_i &\leftarrow n_i / N\end{aligned}$$

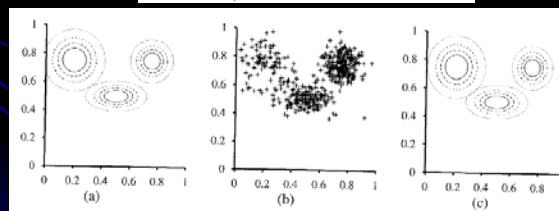


Figure 20.11 (a) A Gaussian mixture model with three components; the weights (left-to-right) are 0.2, 0.3, and 0.5. (b) 500 data points sampled from the model in (a). (c) The model reconstructed by EM from the data in (b).

EM Process: GMM (cont.)

- EM increases the log likelihood of the data at every iteration.

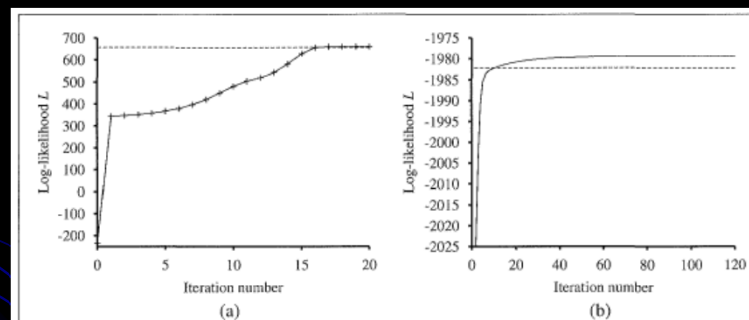


Figure 20.12 Graphs showing the log likelihood of the data, L , as a function of the EM iteration. The horizontal line shows the log likelihood according to the true model. (a) Graph for the Gaussian mixture model in Figure 20.11. (b) Graph for the Bayesian network in Figure 20.13(a).

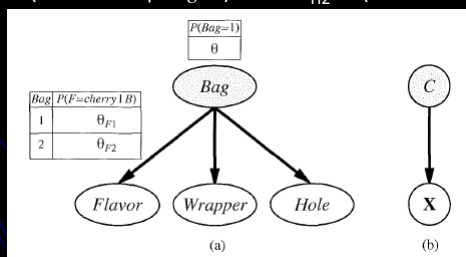
EM: Learning BN with hidden variables

- 2 bags of the mixed candies with 3 features: *Flavor*, *Wrapper*, *Hole*
- Distribution of candies in each bag is described by a Naïve Bayes Model.
- A mixture model.
- Parameters: θ : $P(\text{candy is from Bag}=1)$

$$\theta_{F1} : P(\text{Flavor}=\text{cherry} \mid \text{Bag}=1) \quad \theta_{F2} : P(\text{Flavor}=\text{cherry} \mid \text{Bag}=2)$$

$$\theta_{W1} : P(\text{Wrapper}=\text{red} \mid \text{Bag}=1) \quad \theta_{W2} : P(\text{Wrapper}=\text{red} \mid \text{Bag}=2)$$

$$\theta_{H1} : P(\text{Hole}=\text{true} \mid \text{Bag}=1) \quad \theta_{H2} : P(\text{Hole}=\text{true} \mid \text{Bag}=2)$$



- Recover the descriptions of two bags by observing candies from the mixture.

EM: Learning BN with hidden variables

- Recover the descriptions of two bags by observing candies from the mixture
- By EM algorithm:
 1. Data: 1000 samples, a model w/ $\theta = .5$; $\theta_{F1} = \theta_{W1} = \theta_{H1} = .8$; $\theta_{F2} = \theta_{W2} = \theta_{H2} = .3$;

The counts for 8 possible kinds of candy:

	<i>W = red</i>		<i>W = green</i>	
	<i>H = 1</i>	<i>H = 0</i>	<i>H = 1</i>	<i>H = 0</i>
<i>F = cherry</i>	273	93	104	90
<i>F = lime</i>	79	100	94	167

E step:

2. Initialize the parameters arbitrarily: $\theta^{(0)} = 0.6$, $\theta_{F1}^{(0)} = \theta_{W1}^{(0)} = \theta_{H1}^{(0)} = 0.6$, $\theta_{F2}^{(0)} = \theta_{W2}^{(0)} = \theta_{H2}^{(0)} = 0.4$

3. Update θ : expected count: $\theta^{(1)} = \hat{N}(\text{Bag}=1)/N = \sum_{j=1}^N P(\text{Bag}=1 \mid \text{flavor}_j, \text{wrapper}_j, \text{holes}_j)/N$

$$\theta^{(1)} = \frac{1}{N} \sum_{j=1}^N \frac{P(\text{flavor}_j \mid \text{Bag}=1)P(\text{wrapper}_j \mid \text{Bag}=1)P(\text{holes}_j \mid \text{Bag}=1)P(\text{Bag}=1)}{\sum_{i=1}^2 P(\text{flavor}_j \mid \text{Bag}=i)P(\text{wrapper}_j \mid \text{Bag}=i)P(\text{holes}_j \mid \text{Bag}=i)P(\text{Bag}=i)}$$

- Apply it to, say, the 273 red-wrapped *cherry* candies with holes, we get a contribution of

$$\frac{273}{1000} \cdot \frac{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)}}{\theta_{F1}^{(0)}\theta_{W1}^{(0)}\theta_{H1}^{(0)}\theta^{(0)} + \theta_{F2}^{(0)}\theta_{W2}^{(0)}\theta_{H2}^{(0)}(1 - \theta^{(0)})} \approx 0.22797$$

Continue with other 7 kinds of candy $\rightarrow \theta^{(1)} = .6124$

EM: Learning BN with hidden variables

- Update other parameters θ_{F1} , etc.

the expected count of cherry candies from bag-1:

$$\sum_{j: \text{Flavor}_j = \text{cherry}} P(\text{Bag} = 1 \mid \text{Flavor}_j = \text{cherry}, \text{wrapper}_j, \text{holes}_j)$$

Calculate them by Bayes net algorithm.

$$\theta_{F1}^{(1)} = 0.6124, \theta_{F1}^{(1)} = 0.6684, \theta_{W1}^{(1)} = 0.6483, \theta_{H1}^{(1)} = 0.6558, \\ \theta_{F2}^{(1)} = 0.3887, \theta_{W2}^{(1)} = 0.3817, \theta_{H2}^{(1)} = 0.3827.$$

- The parameter updates for Bayesian network learning with hidden variables are directly available from the results of inference on each example.
- Moreover, only local posterior probabilities are needed for each parameter.

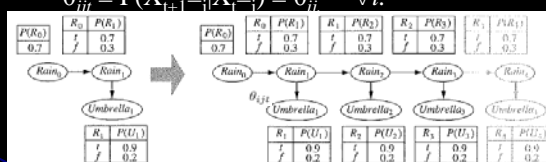
EM: Learning HMM

- Learning the **transition probabilities** from a set of observation sequences
- Complication of each distinct parameter:

a transition probability from state i to state j : $\theta_{ijt} = P(X_{t+1}=j \mid X_t=i)$

θ_{ijt} at time t are repeated across time: i.e.

$$\theta_{ijt} = P(X_{t+1}=j \mid X_t=i) = \theta_{ij} \quad \forall t.$$



- Calculate the expected proportion of times that a transition goes to state j when in state i :
$$\theta_{ij} \leftarrow \frac{\sum_t \tilde{N}(X_{t+1}=j, X_t=i)}{\sum_t \tilde{N}(X_t=i)}$$
- The expected counts are computed by HMM inference algorithm
 - Modify Forward-Backward algorithm to compute the necessary probabilities.
 - Smoothing rather than filtering – pay attention to subsequent evidence in estimating the probability that a particular transition occurred.

Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- Learning method depends on type of performance element, available feedback, type of component to be improved, and its representation
- For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance
= prediction accuracy measured on test set

Cont..

- Full Bayesian learning gives best possible predictions but is intractable.
 - MAP learning balances complexity with accuracy on training data
 - Maximum Likelihood assumes uniform prior, OK for large data sets
1. Choose a parameterized family of models to describe the data
requires substantial insight and sometimes new models
 2. Write down the likelihood of the data as a function of the parameters
may require summing over hidden variables, i.e. inference
 3. Write down the derivative of the log likelihood w.r.t. each parameter
 4. Find the parameter values such that the derivatives are zero
may be hard/impossible; modern optimization techniques help.