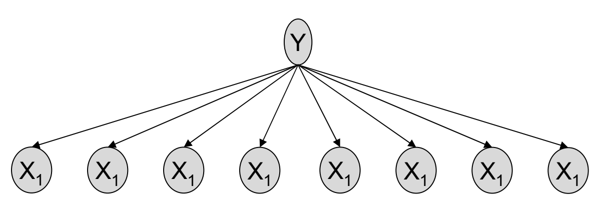
Statistical Learning: The Complex Cases

- Case 0: Bayesian Network structure known, all variables observed
  - Easy: Just count!
- Case 1: Bayesian Network structure known, but some variables unobserved
- Case 2: Bayesian Network structure unknown, but all variables observed
- Case 3: Structure unknown, some variables unobserved

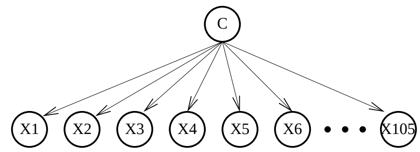
Case 1: Known structure, unobserved variables

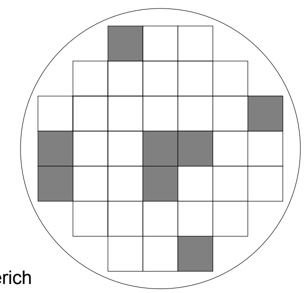
- Simplest case: Finite Mixture Model
- Structure: Naïve Bayes network
- Missing variable: The class!



Example Problem: Cluster Wafers for HP

- We wish to learn $\underline{P}(C,X_1,X_2, \ldots, X_{105})$
- C is a hidden “class” variable





Complete Data and Incomplete data

Wafer	X_1	X_2	…	X_{105}	C
1	1	1	…	0	?
2	0	1	…	1	?
3	0	1	…	1	?
4	1	0	…	1	?

- The given data are incomplete. If we could guess the values of C, we would have complete data, and learning would be easy

“Hard” EM

- Let W = (X_1, X_2, …, X_{105}) be the observed wafers
- Guess initial values for C (e.g., randomly)
- Repeat until convergence
  - Hard M-Step: (Compute maximum likelihood estimates from complete data)
    - Learn P(C)
    - Learn P(X_i|C) for all I
  - Hard E-Step: (Re-estimate the C values)
    - For each wafer, set C to maximize P(W|C)

Hard EM Example

- Suppose we have 10 chips per wafer and 2 wafer classes. Suppose this is the “true” distribution:

C	P(C)
0	0.58
1	0.42

P(X_i=1|C)	0	1
X_1	0.34	0.41
X_2	0.19	0.83
X_3	0.20	0.15
X_4	0.69	0.19
X_5	0.57	0.53
X_6	0.71	0.93
X_7	0.34	0.68
X_8	0.43	0.04
X_9	0.13	0.65
X_{10}	0.14	0.89

Draw 100 training examples and 100 test examples from this distribution

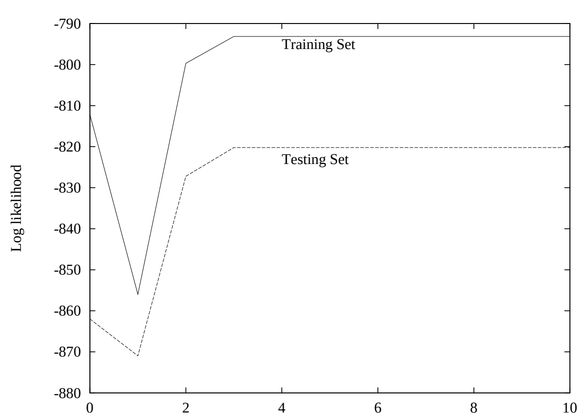
Fit of Model to Fully-Observed Training Data

C	P(C)
0	0.61
1	0.39

$P(X_i=1\|C)$	0	1
X_1	0.28	0.41
X_2	0.15	0.85
X_3	0.15	0.13
X_4	0.67	0.23
X_5	0.49	0.51
X_6	0.74	0.97
X_7	0.39	0.69
X_8	0.34	0.03
X_9	0.10	0.67
X_{10}	0.16	0.87

- Hard-EM could achieve this if it could correctly guess C for each example

EM Training and Testing Curve



Hard EM Fitted Model

C	P(C)
0	0.43
1	0.57

$P(X_i=1\|C)$	0	1
X_1	0.35	0.32
X_2	0.81	0.12
X_3	0.09	0.18
X_4	0.26	0.68
X_5	0.60	0.42
X_6	0.95	0.74
X_7	0.65	0.40
X_8	0.02	0.37
X_9	0.67	0.05
X_{10}	0.86	0.12

- Note that the classes are "reversed": The learned class 0 corresponds to the true class 1. But the likelihoods are the same if the classes are reversed

The search can get stuck in local minima

C	P(C)
0	0.93
1	0.07

$P(X_i=1\|C)$	0	1
X_1	0.35	0.00
X_2	0.42	0.43
X_3	0.12	0.43
X_4	0.47	0.86
X_5	0.53	0.14
X_6	0.83	0.86
X_7	0.51	0.57
X_8	0.16	1.00
X_9	0.34	0.00
X_{10}	0.47	0.00

- Parameters can go to zero or one!
- Should use Laplace Estimates

The Expectation-Maximization (EM) Algorithm

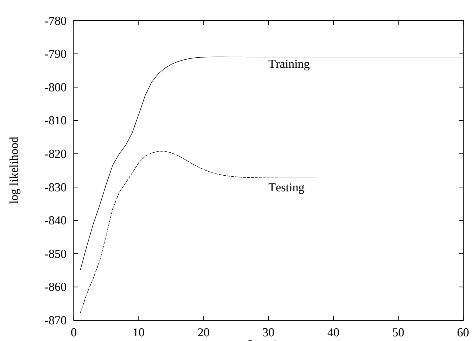
- Initialize the probability tables randomly
- Repeat until convergence
  - E-Step: For each wafer, compute P'(C|W)
  - M-Step: Compute maximum likelihood estimates from weighted data (S)

$$P(C) = \frac{\sum_i P'(C|W_i)}{|S|}$$

$$P(X=x|C=c) = \frac{\sum_{\{i|X_i=x\}} P'(C=c|W_i)}{\sum_i P'(C=c|W_i)}$$

We treat P'(C|W) as fractional "counts". Each wafer W_i belongs to class C with probability P'(C|W).

EM Training Curve



- Each iteration is guaranteed to increase the likelihood of the data. Hence, EM is guaranteed to converge to a *local* maximum of the likelihood.

EM Fitted Model

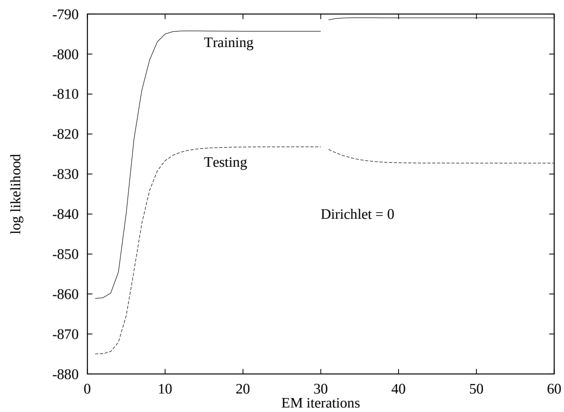
C	P(C)
0	0.35
1	0.65

$P(X_i=1\|C)$	0	1
X_1	0.41	0.28
X_2	0.81	0.21
X_3	0.11	0.15
X_4	0.26	0.63
X_5	0.56	0.47
X_6	0.97	0.75
X_7	0.74	0.38
X_8	0.00	0.34
X_9	0.76	0.08
X_{10}	0.96	0.16

Avoiding Overfitting

- Early stopping. Hold out some of the data, monitor log likelihood on this holdout data, and stop when it starts to decrease
- Laplace estimates
- Full Bayes

EM with Laplace Corrections

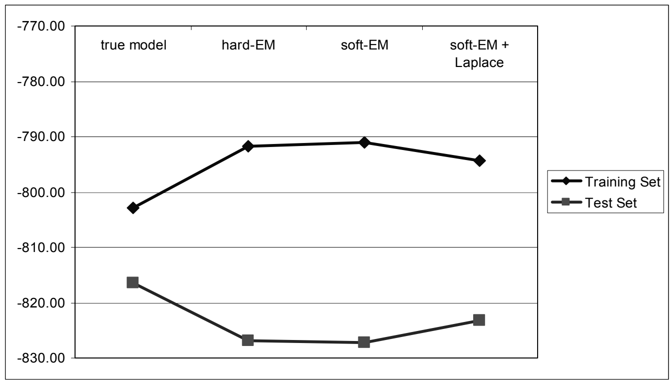


- When correction is removed, EM overfits immediately

Comparison of Results

Method	Training Set	Test Set
true model	-802.85	-816.40
hard-EM	-791.69	-826.94
soft-EM	-790.97	-827.27
soft-EM + Laplace	-794.31	-823.19

Graphical Comparison



- hard-EM and soft-EM overfit
- soft-EM + Laplace gives best test set result

Unsupervised Learning of an HMM

- Suppose we are given only the Umbrella observations as our training data
- How can we learn $\underline{P}(R_t|R_{t-1})$ and $\underline{P}(U_t|R_t)$?

EM for HMMs: "The Forward-Backward Algorithm"

- Initialize probabilities randomly
- Repeat to convergence
  - E-step: Run the forward-backward algorithm on each training example to compute $P'(R_t|U_{1:N})$ for each time step t.
  - M-step: Re-estimate $\underline{P}(R_t|R_{t-1})$ and $\underline{P}(U_t|R_t)$ treating the $P'(R_t|U_{1:N})$ as fractional counts
- Also known as the Baum-Welch algorithm

Hard-EM for HMMs: Viterbi Training

- EM requires forward and backward passes. In the early iterations, just finding the single best path usually works well
- Initialize probabilities randomly
- Repeat to convergence
  - E-step: Run the Viterbi algorithm on each training example to compute $R'_t = \text{argmax}_{Rt}\ \underline{P}(R_t|U_{1:N})$ for each time step t.
  - M-step: Re-estimate $\underline{P}(R_t|R_{t-1})$ and $\underline{P}(U_t|R_t)$ treating the R'_t as if they were correct labels

Case 2: All variables observed; Structure unknown

- Search the space of structures
  - For each potential structure
    - Apply standard maximum likelihood method to fit the parameters
- Problem: How to score the structures?
  - The complete graph will always give the best likelihood on the training data (because it can memorize the data)

MAP Approach: M = model; D = data

$\text{argmax}_M\ P(M \mid D) = \text{argmax}_M\ P(D \mid M) \cdot P(M)$

$\text{argmax}_M \log P(M \mid D) = \text{argmin}_M - \log P(D \mid M) - \log P(M)$

$-\log P(M)$ = number of bits required to represent M (for some chosen representation scheme)

Therefore:

- Choose a representation scheme
- Measure description length in this scheme
- Use this for $-\log P(M)$

Representation Scheme

- Representational cost of adding a parent p to a child node c that already has k parents
  - Must specify link: $\log_2 n(n-1)/2$ bits
  - c already requires 2^k parameters. Adding another (boolean) parent will make this 2^{k+1} parameters, so the increase is $2^{k+1} - 2^k = 2^k$ each of which requires, say, 8 bits. This gives $8 \cdot 2^k$ bits
  - Total: $8 \cdot 2^k + \log_2 n(n-1)/2$
- Min: $-\log P(D \mid M) + \lambda [8 \cdot 2^k + \log_2 n(n-1)/2]$
  - λ is adjusted (e.g., by internal holdout data) to give best results

Note: There are many other possible representation schemes

- Example: Use joint distribution plus the graph structure
  - Joint distribution always has 2^N parameters
  - Describe graph by which edges are *missing*!
  - This scheme would assign the smallest description length to the complete graph!
- The chosen representation scheme implies a prior belief that graphs that can be described compactly under the scheme have higher prior probability P(M)

Search Algorithm

- Search space is all DAGs with N nodes
  - Very large!
- Greedy method
  - Operators: Add an edge, Delete an edge, Reverse an edge
  - At each step,
    - Apply each operator to change the structure
    - Fit the resulting graph to the data
    - Measure total description length
    - Take the best move
  - Stop when local maximum is reached

Alternative Search Algorithm

- Operator:
  - Delete a node and all of its edges from the graph
  - Compute the optimal set of edges for the node and re-insert it into the graph
    - Surprisingly, this can be done efficiently!
- Apply this operator greedily

Initializing the Search

- Compute the best tree-structured graph using Chou-Liu Algorithm

Chou-Liu Algorithm

- for all pairs (X_i, X_j) of variables do
  - compute mutual information:

$$I(X_i; X_j) = \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}$$

- Construct complete graph G such that the edge (X_i, X_j) has weight $I(X_i; X_j)$
- Compute maximum weight spanning tree
- Choose root node arbitrarily and direct edges away from it recursively

Case 3: Unknown structure AND hidden variables

- Structural EM algorithm (Friedman, 1997)
- Repeat
  - E-step: Compute "complete data" from current network structure and parameters
  - Structural M-Step: Apply structure learning algorithm to find MAP structure from complete data
  - Standard M-Step: Find ML estimate of the network parameters
- Until convergence
- Works ok if there are not too many hidden variables

Statistical Learning Summary

- Case 0: Bayesian Network structure known, all variables observed
  - Easy: Just count!
- Case 1: Bayesian Network structure known, but some variables unobserved
  - EM Algorithm
- Case 2: Bayesian Network structure unknown, but all variables observed
  - Greedy structure search with MDL to penalize complex networks
- Case 3: Structure unknown, some variables unobserved
  - Structural EM: Combine greedy structure search with EM