Lecture 2:

Parameter Learning in Fully Observed Graphical Models

Sam Roweis

Monday July 24, 2006
Machine Learning Summer School, Taiwan

Likelihood Functions

- So far we have focused on the (log) probability function $p(\mathbf{x}|\theta)$ which assigns a probability (density) to any joint configuration of variables $\mathbf{x}$ given fixed parameters θ.
- But in learning we turn this on its head: we have some fixed data and we want to find parameters.
- Think of $p(\mathbf{x}|\theta)$ as a function of θ for fixed $\mathbf{x}$:

$$Z(\theta; \mathbf{x}) = p(\mathbf{x}|\theta)$$
$$\ell(\theta; \mathbf{x}) = \log p(\mathbf{x}|\theta)$$

  This function is called the (log) "likelihood".
- Chose θ to maximize some loss function $L(\theta)$ which includes $\ell(\theta)$:

$L(\theta) = \ell(\theta; \mathcal{D})$ maximum likelihood (ML)

$L(\theta) = \ell(\theta; \mathcal{D}) + \log p(\theta)$ maximum a posteriori (MAP)/penalizedML

(also cross-validation, Bayesian estimators, BIC, AIC, ...)

Maximum Likelihood

- For IID data:

$$p(\mathcal{D}|\theta) = \prod_m p(\mathbf{x}^m|\theta)$$

$$\ell(\theta; \mathcal{D}) = \sum_m \log p(\mathbf{x}^m|\theta)$$

- Idea of maximum likelihod estimation (MLE): pick the setting of parameters most likely to have generated the data we saw:

$$\theta^*_{\mathrm{ML}} = \mathrm{argmax}_\theta \ \ell(\theta; \mathcal{D})$$

- Commonly used as a "baseline" model in statistics.
  Often leads to "intuitive", "appealing", or "natural" estimators.
- For a start, the IID assumption makes the log likelihood into a sum, so its derivative can be easily taken term by term.

Example: Bernoulli Trials

- We observe M iid coin flips: $\mathcal{D}$=H,H,T,H,...
- Model: $p(H) = \theta \quad p(T) = (1-\theta)$
- Likelihood:

$$\begin{aligned}
\ell(\theta; \mathcal{D}) &= \log p(\mathcal{D}|\theta) \\
&= \log \prod_m \theta^{\mathbf{x}^m} (1-\theta)^{1-\mathbf{x}^m} \\
&= \log\theta \sum_m \mathbf{x}^m + \log(1-\theta) \sum_m (1-\mathbf{x}^m) \\
&= \log\theta N_{\mathrm{H}} + \log(1-\theta) N_{\mathrm{T}}
\end{aligned}$$

- Take derivatives and set to zero:

$$\frac{\partial \ell}{\partial \theta} = \frac{N_{\mathrm{H}}}{\theta} - \frac{N_{\mathrm{T}}}{1-\theta}$$

$$\Rightarrow \theta^*_{\mathrm{ML}} = \frac{N_{\mathrm{H}}}{N_{\mathrm{H}} + N_{\mathrm{T}}}$$

Example: Univariate Normal

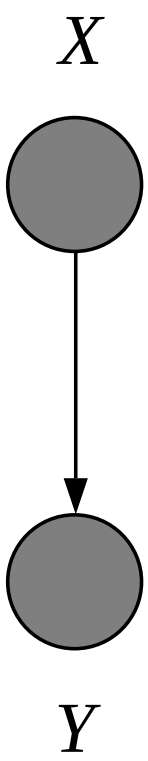
- We observe M iid real samples: $\mathcal{D}$=1.18,-.25,.78,...
- Model: $p(x) = (2\pi\sigma^2)^{-1/2} \exp\{-(x-\mu)^2/2\sigma^2\}$
- Likelihood (using probability density):

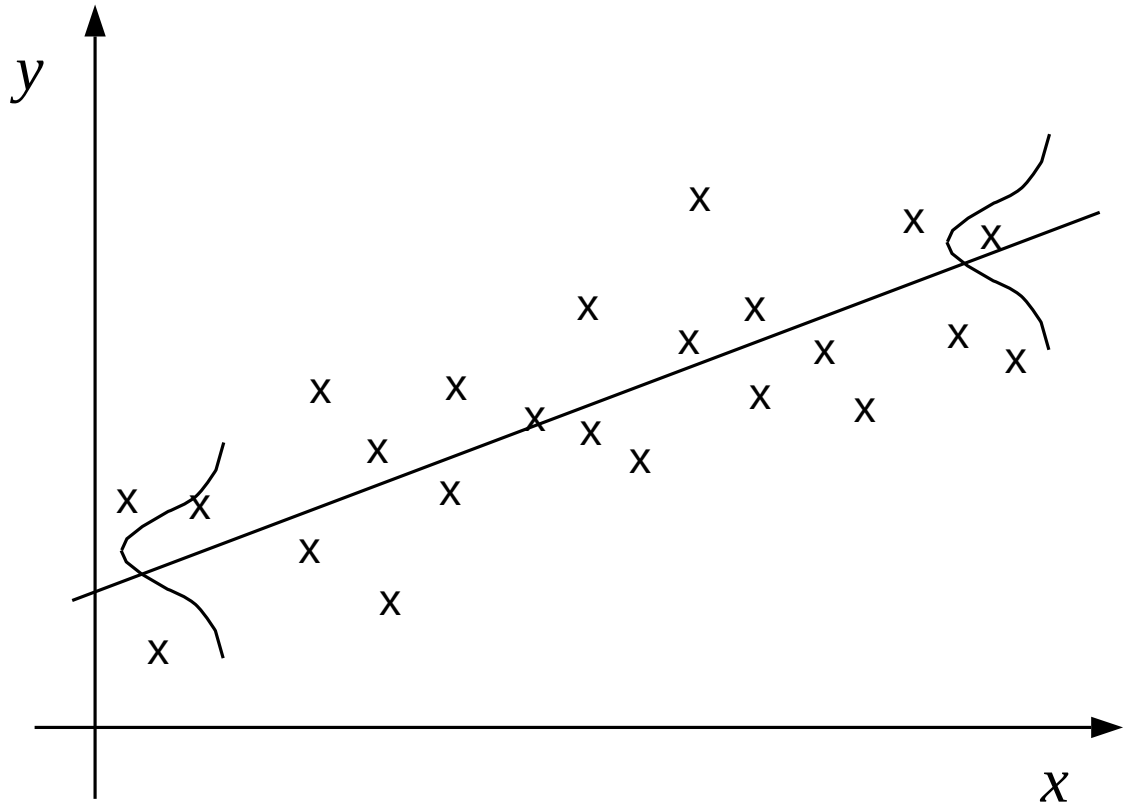
$$\begin{aligned}\ell(\theta; \mathcal{D}) &= \log p(\mathcal{D}|\theta) \\ &= -\frac{M}{2}\log(2\pi\sigma^2) - \frac{1}{2}\sum_m \frac{(x^m - \mu)^2}{\sigma^2}\end{aligned}$$

- Take derivatives and set to zero:

$$\begin{aligned}\frac{\partial \ell}{\partial \mu} &= (1/\sigma^2)\textstyle\sum_m (x_m - \mu) \\ \frac{\partial \ell}{\partial \sigma^2} &= -\frac{M}{2\sigma^2} + \frac{1}{2\sigma^4}\textstyle\sum_m (x_m - \mu)^2 \\ \Rightarrow \mu_{\mathrm{ML}} &= (1/M)\textstyle\sum_m x_m \\ \sigma^2_{\mathrm{ML}} &= (1/M)\textstyle\sum_m x_m^2 - \mu^2_{\mathrm{ML}}\end{aligned}$$

Example: Linear Regression





Example: Linear Regression

- At a linear regression node, some parents (covariates/inputs) and all children (responses/outputs) are continuous valued variables.
- For each child and setting of discrete parents we use the model:

$$p(y|\mathbf{x}, \theta) = \text{gauss}(y|\theta^\top \mathbf{x}, \sigma^2)$$

- The likelihood is the familiar "squared error" cost:

$$\ell(\theta; \mathcal{D}) = -\frac{1}{2\sigma^2} \sum_m (y^m - \theta^\top \mathbf{x}^m)^2$$

- The ML parameters can be solved for using linear least-squares:

$$\frac{\partial \ell}{\partial \theta} = -\sum_m (y^m - \theta^\top \mathbf{x}^m)\mathbf{x}^m$$

$$\Rightarrow \theta^*_{\text{ML}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

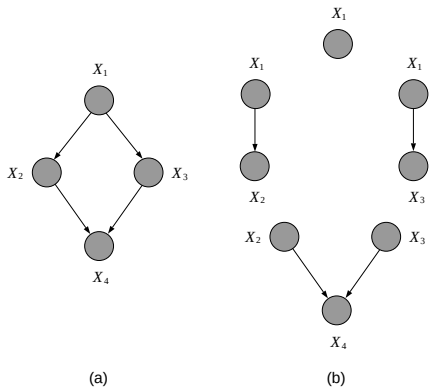
- "Sufficient statistics" are input correlation matrix and input-output cross-correlation vector.

MLE for Directed GMs

- For a directed GM, the likelihood function has a nice form:

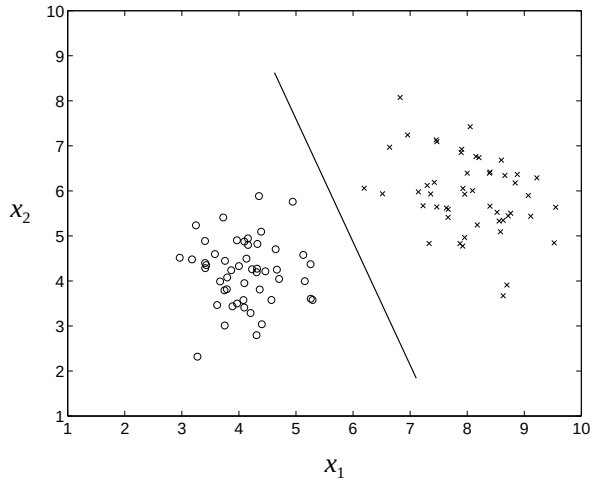
$$\log p(\mathcal{D}|\theta) = \log \prod_m \prod_i p(\mathbf{x}_i^m | \mathbf{x}_{\pi_i}, \theta_i) = \sum_m \sum_i \log p(\mathbf{x}_i^m | \mathbf{x}_{\pi_i}, \theta_i)$$

- The parameters *decouple*; so we can maximize likelihood independently for each node's function by setting θ_i.
- Only need the values of x_i and its parents in order to estimate θ_i.
- In general, for fully observed data if we know how to estimate params at a single node we can do it for the whole network.



Reminder: Classification

- Given examples of a discrete *class label* y and some *features* $\mathbf{x}$.
- Goal: compute label (y) for new inputs $\mathbf{x}$.
- Two approaches:
  Generative: model $p(\mathbf{x}, y) = p(y)p(\mathbf{x}|y)$;
  use Bayes' rule to infer conditional $p(y|\mathbf{x})$.
  Discriminative: model discriminants $f(y|\mathbf{x})$ directly and take max.
- Generative approach is related to conditional *density estimation* while discriminative approach is closer to *regression*.

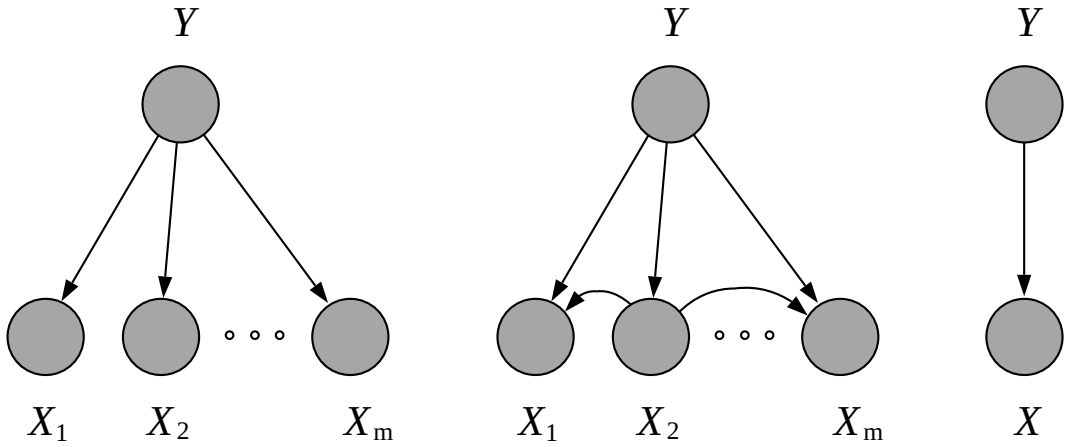


Probabilistic Classification: Bayes Classifiers

- Generative model: $p(\mathbf{x}, y) = p(y)p(\mathbf{x}|y)$.
  $p(y)$ are called class *priors*.
  $p(\mathbf{x}|y)$ are called *class conditional feature distributions*.
- For the prior we use a Bernoulli or multinomial:
  $p(y = k|\pi) = \pi_k$ with $\sum_k \pi_k = 1$.
- Classification rules:
  ML: $\operatorname{argmax}_y p(\mathbf{x}|y)$ (can behave badly if skewed priors)
  MAP: $\operatorname{argmax}_y p(y|\mathbf{x}) = \operatorname{argmax}_y \log p(\mathbf{x}|y) + \log p(y)$ (safer)
- Fitting: maximize $\sum_n \log p(\mathbf{x}^n, y^n) = \sum_n \log p(\mathbf{x}^n|y^n) + \log p(y^n)$
  1) Sort data into batches by class label.
  2) Estimate $p(y)$ by counting size of batches (plus regularization).
  3) Estimate $p(\mathbf{x}|y)$ separately within each batch using ML.
  (also with regularization).

Three Key Regularization Ideas

- To avoid overfitting, we can put *priors* on the parameters of the class and class conditional feature distributions.
- We can also *tie* some parameters together so that fewer of them are estimated using more data.
- Finally, we can make *factorization* or *independence* assumptions about the distributions. In particular, for the class conditional distributions we can assume the features are fully dependent, partly dependent, or independent (!).



Gaussian Class-Conditional Distributions

- If all features are continuous, a popular choice is a Gaussian class-conditional.

$$p(\mathbf{x}|y=k,\theta) = |2\pi\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)\Sigma^{-1}(\mathbf{x}-\mu_k)\right\}$$

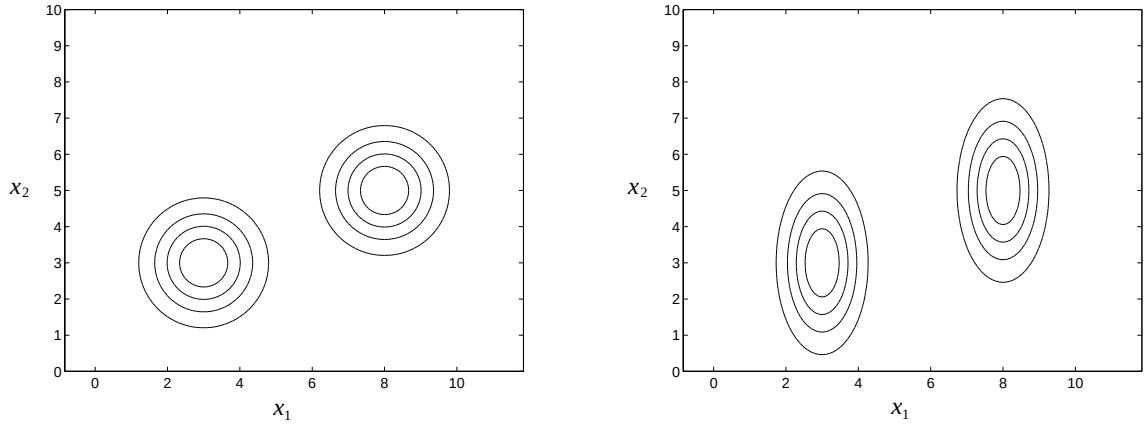
- Fitting: use the following amazing and useful fact.
  The maximum likelihood fit of a Gaussian to some data is the Gaussian whose mean is equal to the data mean and whose covariance is equal to the sample covariance.

  [Try to prove this as an exercise in understanding likelihood, algebra, and calculus all at once!]

- Seems easy. And works amazingly well.
  But we can do even better with some simple regularization...

Regularized Gaussians

- Idea 1: assume all the covariances are the same (tie parameters). This is exactly Fisher's linear discriminant analysis.



- Idea 2: Make independence assumptions to get diagonal or identity-multiple covariances. (Or sparse inverse covariances.) More on this in a few minutes...
- Idea 3: add a bit of the identity matrix to each sample covariance. This "fattens it up" in directions where there wasn't enough data. Equivalent to using a "Wishart prior" on the covariance matrix.

Gaussian Bayes Classifier

- Maximum likelihood estimates for parameters:
  priors π_k: use observed frequencies of classes (plus smoothing)
  means μ_k: use class means
  covariance Σ: use data from single class or pooled data
  $(\mathbf{x}^m - \mu_{y^m})$ to estimate full/diagonal covariances
- Compute the posterior via Bayes' rule:

$$
\begin{aligned}
p(y = k|\mathbf{x}, \theta) &= \frac{p(\mathbf{x}|y = k, \theta)p(y = k|\pi)}{\sum_j p(\mathbf{x}|y = j, \theta)p(y = j|\pi)} \\
&= \frac{\exp\{\mu_k^\top \Sigma^{-1}\mathbf{x} - \mu_k^\top \Sigma^{-1}\mu_k/2 + \log \pi_k\}}{\sum_j \exp\{\mu_j^\top \Sigma^{-1}\mathbf{x} - \mu_j^\top \Sigma^{-1}\mu_j/2 + \log \pi_j\}} \\
&= e^{\beta_k^\top \mathbf{x}} / \sum_j e^{\beta_j^\top \mathbf{x}} = \exp\{\beta_k^\top \mathbf{x}\}/Z
\end{aligned}
$$

  where $\beta_k = [\Sigma^{-1}\mu_k\,;\, (\mu_k^\top \Sigma^{-1}\mu_k + \log \pi_k)]$ and we have augmented $\mathbf{x}$ with a constant component always equal to 1 (bias term).

Softmax/Logit

- The squashing function is known as the *softmax* or *logit*:

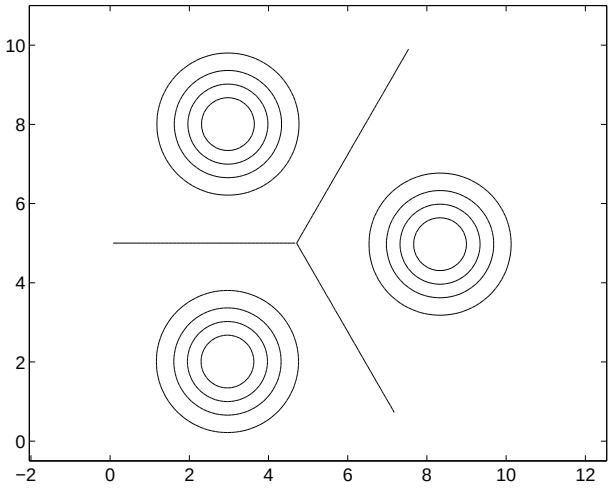
$$\phi_k(\mathbf{z}) \equiv \frac{e^{z_k}}{\sum_j e^{z_j}} \qquad g(\eta) = \frac{1}{1+e^{-\eta}}$$

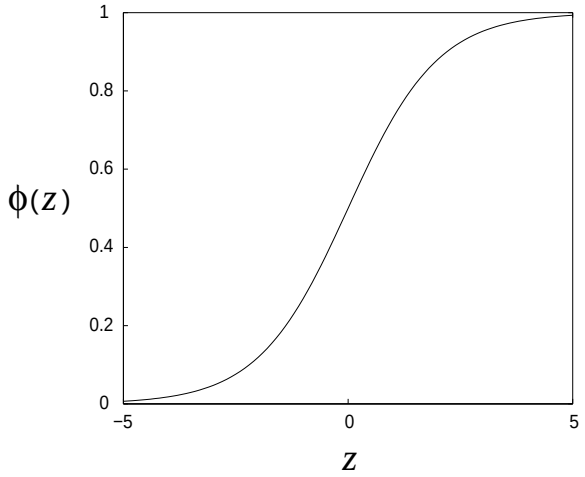
- It is invertible (up to a constant):

$$z_k = \log \phi_k + c \qquad \eta = \log(g/1-g)$$

- Derivative is easy:

$$\frac{\partial \phi_k}{\partial z_j} = \phi_k(\delta_{kj} - \phi_j) \qquad \frac{dg}{d\eta} = g(1-g)$$





Log Linear Geometry

- Taking the ratio of any two posteriors (the "odds") shows that the contours of equal pairwise probability are linear surfaces in the feature space:

$$\frac{p(y=k|\mathbf{x},\theta)}{p(y=j|\mathbf{x},\theta)} = \exp\left\{(\beta_k - \beta_j)^\top \mathbf{x}\right\}$$

- The pairwise discrimination contours $p(y_k) = p(y_j)$ are orthogonal to the differences of the means in feature space when $\Sigma = \sigma I$. For general Σ shared b/w all classes the same is true in the transformed feature space $\mathbf{w} = \Sigma^{-1}\mathbf{x}$.
- The priors do not change the geometry, they only shift the operating point on the logit by the log-odds $\log(\pi_k/\pi_j)$.
- Thus, for equal class-covariances, we obtain a *linear classifier*.
- If we use different covariances, the decision surfaces are conic sections and we have a quadratic classifier.

Discrete Bayesian Classifier

- If the inputs are discrete (categorical), what should we do?
- The simplest class conditional model is a joint multinomial (table):

$$p(x_1 = a, x_2 = b, \ldots | y = c) = \eta^c_{ab\ldots}$$

- This is conceptually correct, but there's a big practical problem.
- Fitting: ML params are observed counts:

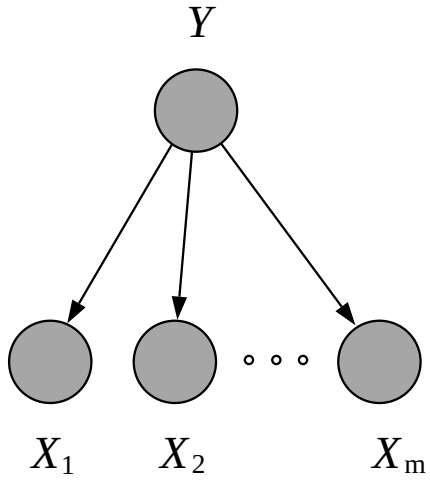
$$\eta^c_{ab\ldots} = \frac{\sum_n [y_n = c][x_1 = a][x_2 = b][\ldots][\ldots]}{\sum_n [y_n = c]}$$

- Consider the 16x16 digits at 256 gray levels.
- How many entries in the table? How many will be zero?
  What happens at test time? Doh!
- We obviously need some regularlization.
  Smoothing will not help much here. Unless we know about the relationships between inputs beforehand, sharing parameters is hard also. But what about independence?

Naive (Idiot's) Bayes Classifier

- Assumption: conditioned on class, attributes are independent.

$$p(\mathbf{x}|y) = \prod_i p(x_i|y)$$



- Sounds crazy right? Right! But it works.
- Algorithm: sort data cases into bins according to y_n. Compute marginal probabilities $p(y = c)$ using frequencies.
- For each class, estimate distribution of i^{th} variable: $p(x_i|y = c)$.
- At test time, compute $\text{argmax}_c\ p(c|\mathbf{x})$ using

$$\begin{aligned} c(\mathbf{x}) = \text{argmax}_c\ p(c|\mathbf{x}) &= \text{argmax}_c\ [\log p(\mathbf{x}|c) + \log p(c)] \\ &= \text{argmax}_c\ [\log p(c) + \sum_i \log p(x_i|c)] \end{aligned}$$

Discrete (Multinomial) Naive Bayes

Discrete features x_i, assumed independent given the class label y.

$$p(x_i = j | y = k) = \eta_{ijk}$$

$$p(\mathbf{x} | y = k, \eta) = \prod_i \prod_j \eta_{ijk}^{[x_i = j]}$$

Classification rule:

$$p(y = k | \mathbf{x}, \eta) = \frac{e^{\beta_k^\top \mathbf{x}}}{\sum_q e^{\beta_q^\top \mathbf{x}}}$$

$$= \frac{\pi_k \prod_i \prod_j \eta_{ijk}^{[x_i = j]}}{\sum_q \pi_q \prod_i \prod_j \eta_{ijq}^{[x_i = j]}}$$

ML parameters are class-conditional frequency counts:

$$\eta^*_{ijk} = \frac{\sum_m [x_i{}^m = j][y^m = k]}{\sum_m [y^m = k]}$$

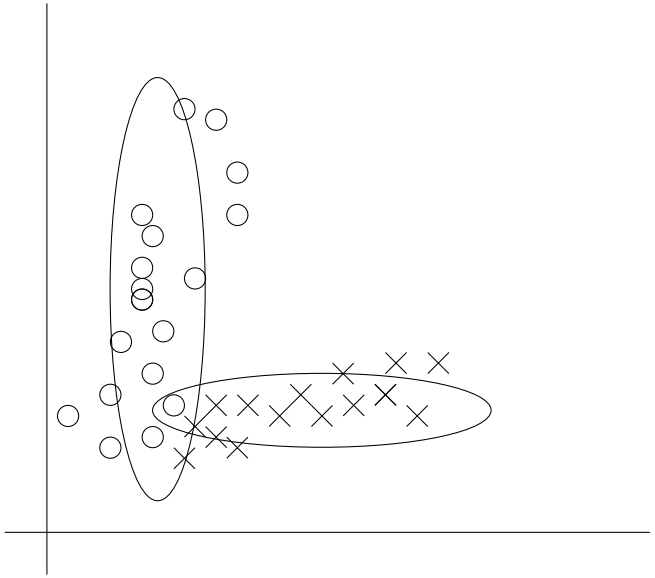
$\beta_k = \log[\eta_{11k} \ldots \eta_{1jk} \ldots \eta_{ijk} \ldots \log \pi_k]$

$\mathbf{x} = [x_1{=}1; x_1{=}2; \ldots; x_i{=}j; \ldots; 1]$

Log-Linear!

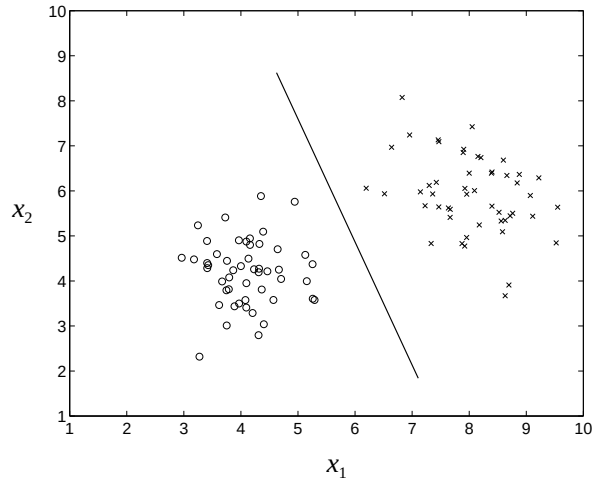
Gaussian Naive Bayes

- This is just a Gaussian Bayes Classifier with a separate diagonal covariance matrix for each class.
- Equivalent to fitting a one-dimensional Gaussian to each input for each possible class.
- Decision surfaces are quadratics, not linear...



Discriminative Models

- Parametrize $p(y|\mathbf{x})$ directly, forget $p(\mathbf{x}, y)$ and Bayes' rule.
- As long as $p(y|\mathbf{x})$ or discriminants $f(y|\mathbf{x})$ are linear functions of $\mathbf{x}$ (or monotone transforms), decision surfaces will be piecewise linear.
- Don't need to model the density of the features.
  Some density models have lots of parameters.
  Many densities give same linear classifier.
  But we cannot generate new labeled data.
- Optimize a cost function closer to the one we use at test time.



Logistic/Softmax Regression

- Model: y is a multinomial random variable whose posterior is the softmax of linear functions of *any* feature vector.

$$p(y = k|\mathbf{x}, \theta) = \frac{e^{\theta_k^\top \mathbf{x}}}{\sum_j e^{\theta_j^\top \mathbf{x}}}$$

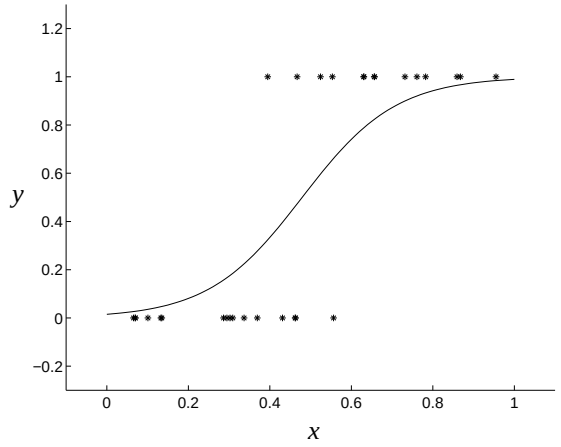
- Fitting: now we optimize the *conditional* likelihood:

$$\ell(\theta; \mathcal{D}) = \sum_{mk} [y^m = k] \log p(y = k|\mathbf{x}^m, \theta) = \sum_{mk} y_k^m \log p_k^m$$

$$\frac{\partial \ell}{\partial \theta_i} = \sum_{mk} \frac{\partial \ell_k^m}{\partial p_k^m} \frac{\partial p_k^m}{\partial z_i^m} \frac{\partial z_i^m}{\partial \theta_i}$$

$$= \sum_{mk} \frac{y_k^m}{p_k^m} p_k^m (\delta_{ik} - p_i^m) \mathbf{x}^m$$

$$= \sum_m (y_k^m - p_k^m) \mathbf{x}^m$$

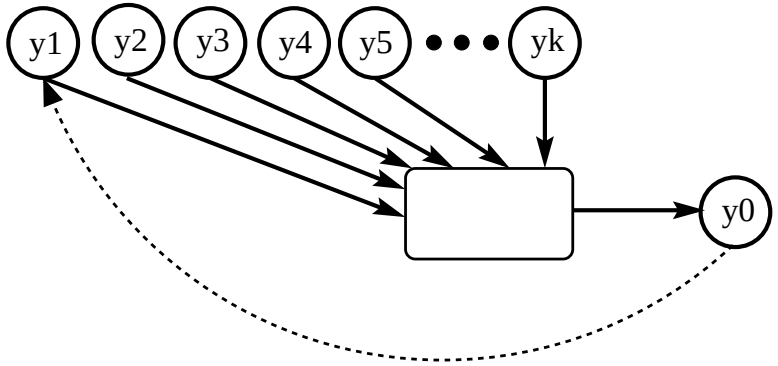


Chains: Markov Models

- If variables have some temporal/spatial order, we can model their joint distribution as a dynamical/diffusion system.
- Simple idea: next output depends only on k previous outputs:

$$\mathbf{y}t = f[\mathbf{y}t-1, \mathbf{y}t-2, \ldots, \mathbf{y}t-k]$$

k is called the *order* of the Markov Model



- Add noise to make the system probabilistic:

$$p(\mathbf{y}t|\mathbf{y}t-1, \mathbf{y}t-2, \ldots, \mathbf{y}t-k)$$

Learning Markov Models

- The ML parameter estimates for a simple Markov model are easy:

$$p(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T) = p(\mathbf{y}_1 \ldots \mathbf{y}_k) \prod_{t=k+1}^{T} p(\mathbf{y}_t | \mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots, \mathbf{y}_{t-k})$$

$$\log p(\{\mathbf{y}\}) = \log p(\mathbf{y}_1 \ldots \mathbf{y}_k) + \sum_{t=k+1}^{T} \log p(\mathbf{y}_t | \mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots, \mathbf{y}_{t-k})$$

- Each window of $k+1$ outputs is a training case for the model $p(\mathbf{y}_t | \mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots, \mathbf{y}_{t-k})$.
- Example: for discrete outputs (symbols) and a 2nd-order markov model we can use the multinomial model:

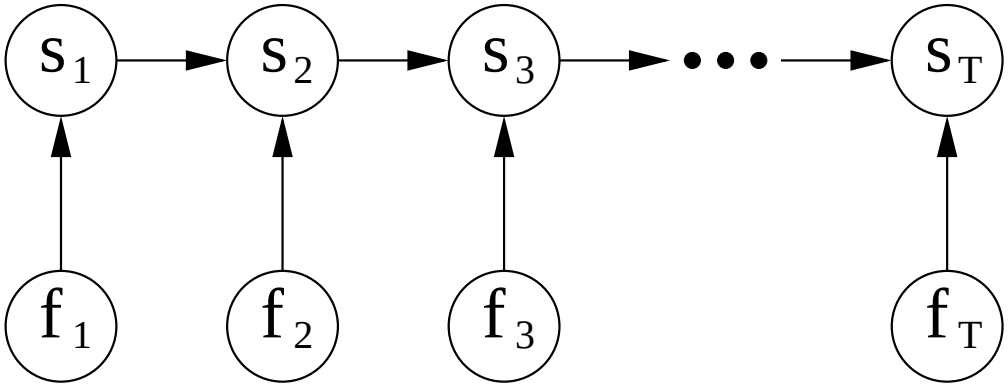
$$p(y_t = m | y_{t-1} = a, y_{t-2} = b) = \alpha_{mab}$$

The maximum likelihood values for α are:

$$\alpha^*_{mab} = \frac{\text{num}[t \; s.t. \; y_t = m, y_{t-1} = a, y_{t-2} = b]}{\text{num}[t \; s.t. \; y_{t-1} = a, y_{t-2} = b]}$$

Maximum Entropy Markov Models

- We can extend this idea to a "logistic regression through time" type of conditional model called a *maximum entropy markov model*.



- The joint distribution is now a conditional model:

$$p(s_1^T|x_1^T) = \prod_t p(s_t|s_{t-1}, f_t(x_1^T))$$

- The features f_t can be very nonlocal functions of the underlying input sequence, for example they can consult things in the past and in the future.

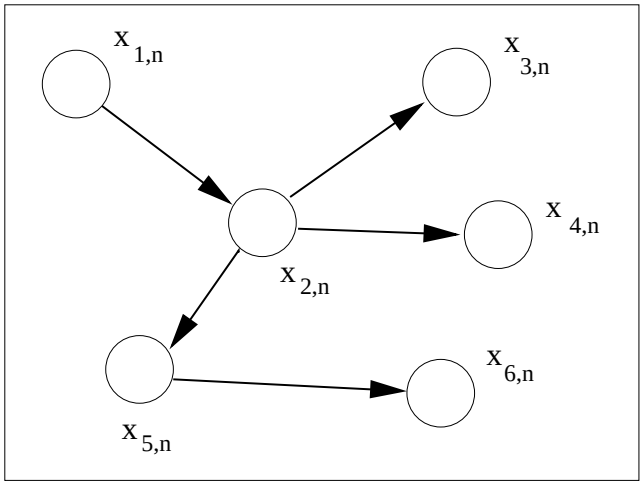
Directed Tree Graphical Models

- Directed trees are DAGMs in which each variable x_i has exactly one other variable as its parent $\mathbf{x}_{\pi_i}$ except the "root" x_{root} which has no parents. Thus, the probability of a variable taking on a certain value depends only on the value of its parent:

$$p(\mathbf{x}) = p(x_{\text{root}}) \prod_{i \neq \text{root}} p(x_i | \mathbf{x}_{\pi_i})$$

- Trees are the next step up from assuming independence. Instead of considering variables in isolation, consider them in pairs.

NB: each node (except root) has exactly one parent, but nodes may have more than one child.



Likelihood function

- Notation:

  $\mathbf{y}_i \equiv$ a node x_i and its single parent $\mathbf{x}_{\pi_i}$.

  $\mathbf{V}_i \equiv$ set of joint configurations of node i and its parent $\mathbf{x}_{\pi_i}$

  ($\mathbf{y}_{\text{root}} \equiv x_{\text{root}}$ and $\mathbf{V}_{\text{root}} \equiv \mathbf{v}_{\text{root}}$)

- Directed model likelihood:

$$\ell(\theta; \mathcal{D}) = \sum_n \log p(\mathbf{x}^n) = \sum_n \left[\log p_r(x_r^n) + \sum_{i \neq r} \log p({x_i}^n | {\mathbf{x}_{\pi_i}}^n) \right]$$

$$= \sum_n \sum_i \sum_{\mathbf{v} \in \mathbf{V}_i} [\mathbf{y}_i^n = \mathbf{v}] \log p_i(\mathbf{v}) \qquad \text{indicator trick}$$

$$= \sum_i \sum_{\mathbf{v} \in \mathbf{V}_i} N_i(\mathbf{v}) \log p_i(\mathbf{v})$$

  where $N_i(\mathbf{v}) = \sum_n [\mathbf{y}_i^n = \mathbf{v}]$ and $p_i(\mathbf{v}_i) = p(x_i | \mathbf{x}_{\pi_i})$.

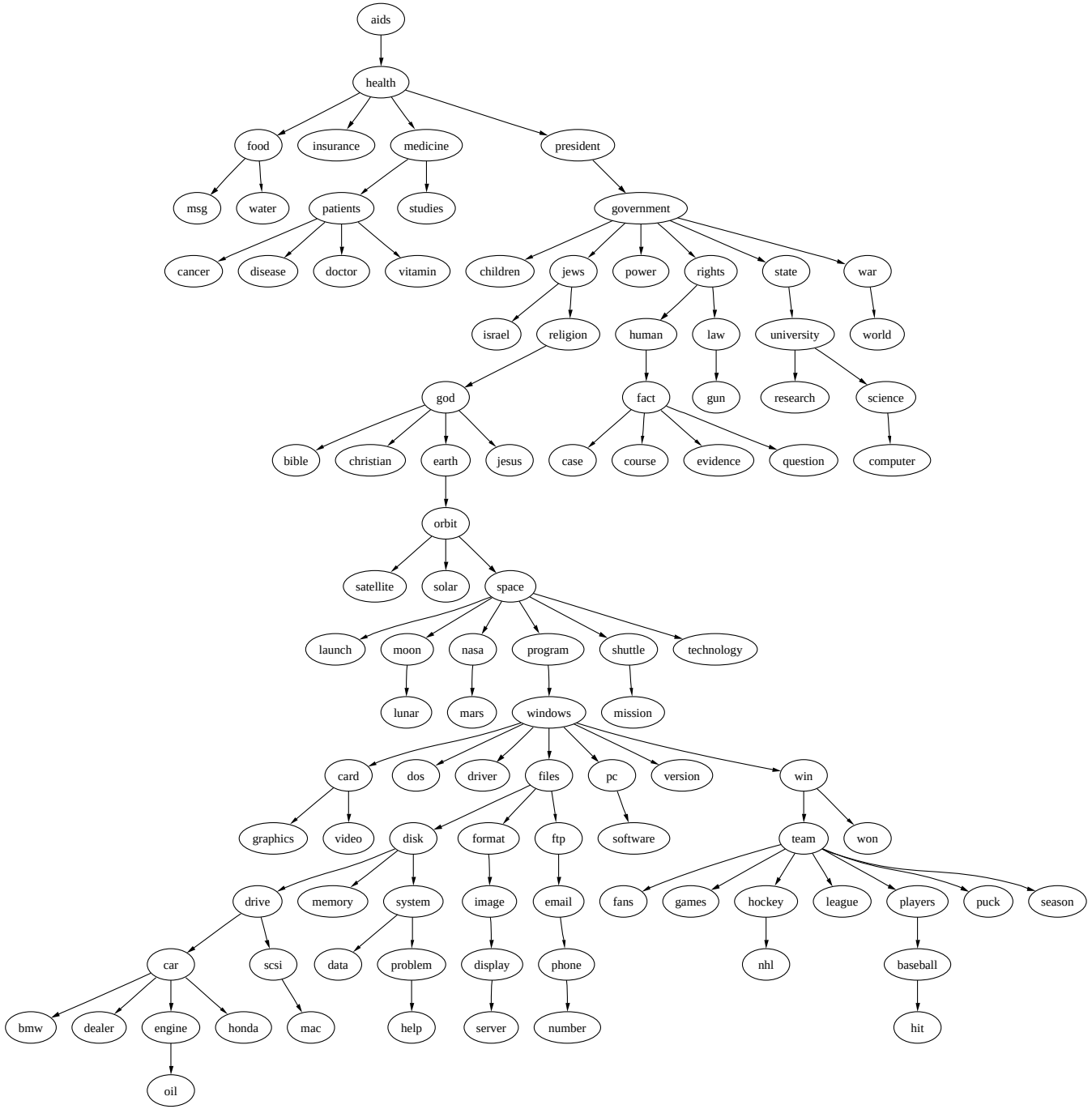
- Trees are in the exponential family with $\mathbf{y}_i$ as sufficient statistics.

Maximum Likelihood Parameters Given Structure

- Trees are just a special case of fully observed graphical models.
- For discrete data x_i with values v_i, each node stores a conditional probability table (CPT) over its values given its parent's value. The ML parameter estimates are just the empirical histograms of each node's values given its parent:

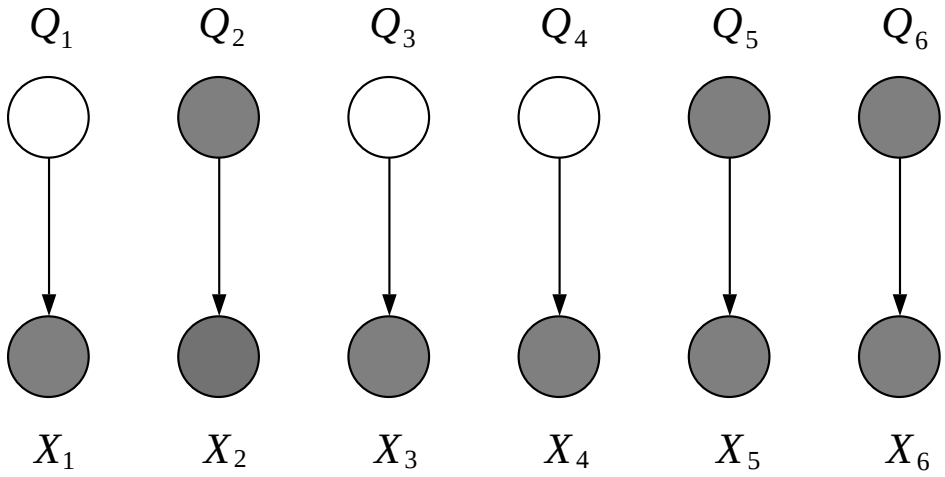
$$p^*(x_i = v_i | \mathbf{x}_{\pi_i} = v_j) = \frac{N(x_i = v_i, \mathbf{x}_{\pi_i} = v_j)}{\sum_{\mathbf{v}_i} N(x_i = v_i, \mathbf{x}_{\pi_i} = v_j)} = \frac{N_i(\mathbf{y}_i)}{N_{\pi_i}(v_j)}$$

  except for the root which uses marginal counts $N_r(v_r)/N$.
- For continuous data, the most common model is a two-dimensional Gaussian at each node. The ML parameters are just to set the mean of $p_i(\mathbf{y}_i)$ to be the sample mean of $[x_i; \mathbf{x}_{\pi_i}]$ and the covariance matrix to the sample covariance.
- In practice we should use some kind of smoothing/regularization.


aids
health
food
insurance
medicine
president
msg
water
patients
studies
government
cancer
disease
doctor
vitamin
children
jews
power
rights
state
war
israel
religion
human
law
university
world
god
fact
gun
research
science
bible
christian
earth
jesus
case
course
evidence
question
computer
orbit
satellite
solar
space
launch
moon
nasa
program
shuttle
technology
lunar
mars
windows
mission
card
dos
driver
files
pc
version
win
graphics
video
disk
format
ftp
software
team
won
drive
memory
system
image
email
fans
games
hockey
league
players
puck
season
car
scsi
data
problem
display
phone
nhl
baseball
bmw
dealer
engine
honda
mac
help
server
number
hit
oil

Unobserved Variables

- We have been assuming that we observe all the random variables in our model at training time, and all the "inputs" at test time.
- But certain variables Q in our models may be *unobserved*, either some of the time or always, either at training time or at test time.



(Graphically, we will use shading to indicate observation.)

Partially Unobserved (Missing) Variables

- If variables are occasionally unobserved they are *missing data*.
  e.g. undefinied inputs, missing class labels, erroneous target values
- In this case, we can still model the joint distribution, but we define a new cost function in which we *sum out* or *marginalize* the missing values at training or test time:

$$\begin{aligned}\ell(\theta;\mathcal{D}) &= \sum_{\text{complete}} \log p(\mathbf{x}^c, \mathbf{y}^c|\theta) + \sum_{\text{missing}} \log p(\mathbf{x}^m|\theta) \\ &= \sum_{\text{complete}} \log p(\mathbf{x}^c, \mathbf{y}^c|\theta) + \sum_{\text{missing}} \log \sum_{\mathbf{y}} p(\mathbf{x}^m, \mathbf{y}|\theta)\end{aligned}$$

[Recall that $p(x) = \sum_q p(x, q)$.]