Lecture 1b:

Parameter Learning in Fully Observed Graphical Models

Sam Roweis

Thursday August 17, 2006
CIAR Summer School, Toronto

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, ...)

Learning Graphical Models from Data

- In AI the bottleneck is often knowledge acquisition.
- Human experts are rare, expensive, unreliable, slow.
  But we have lots of machine readable data.
- Want to build systems automatically based on data and a small amount of prior information (e.g. from experts).

$\Rightarrow$ Sam Roweis

$\Rightarrow$ Geoff Hinton

- For now, our "systems" will be probabilistic graphical models.
- Assume the prior information we have specifies type & structure of the GM, as well as the mathematical form of the parent-conditional distributions or clique potentials.
- In this case learning $\equiv$ setting parameters.
  ("Structure learning" is also possible but we won't consider it now.)

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

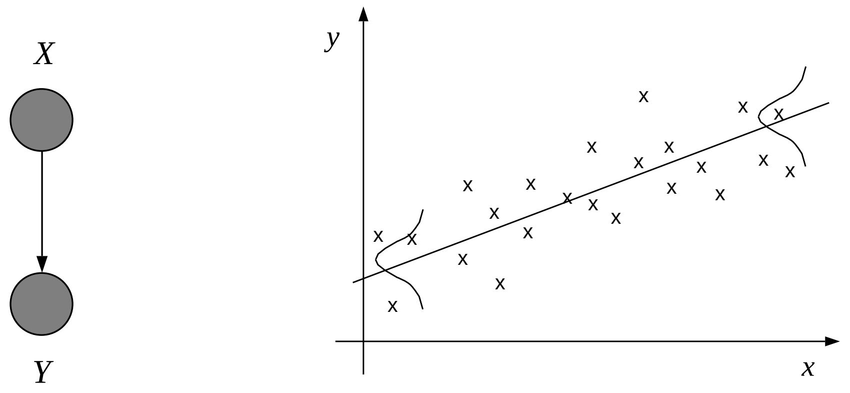
- We observe M iid coin flips: $\mathcal{D}$=H,H,T,H,...
- Model: $p(H) = \theta \quad p(T) = (1-\theta)$
- Likelihood:

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

- Take derivatives and set to zero:

$$\begin{aligned}\frac{\partial\ell}{\partial\theta} &= \frac{N_{\mathrm{H}}}{\theta} - \frac{N_{\mathrm{T}}}{1-\theta}\\ \Rightarrow \theta^*_{\mathrm{ML}} &= \frac{N_{\mathrm{H}}}{N_{\mathrm{H}}+N_{\mathrm{T}}}\end{aligned}$$

Example: Linear Regression



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

- 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:

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

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

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

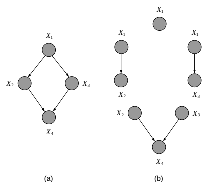
- The ML parameters can be solved for using linear least-squares:

$$\begin{aligned}\frac{\partial\ell}{\partial\theta} &= -\sum_m(y^m-\theta^\top\mathbf{x}^m)\mathbf{x}^m\\ \Rightarrow \theta^*_{\mathrm{ML}} &= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}\end{aligned}$$

- "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.

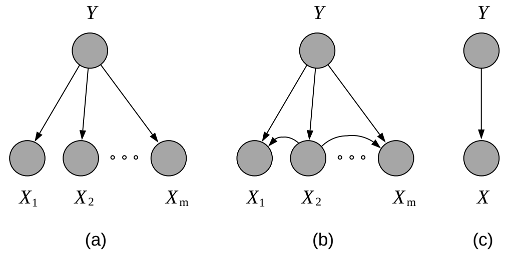


Bayesian Approach

- The Bayesian programme (after Rev. Thomas Bayes) treats *all unknown quantities as random variables* and represents uncertainty over those quantities using probability distributions.
- Thus, unknown parameters are treated as random variables just like latent (hidden) variables or missing data.
  This means we have probability distributions over the parameters.
- We can (and should) put priors $p(\theta)$ over them, and can compute things like posteriors $p(\theta|\mathcal{D})$.
- Crucially, we want to integrate/sum out all unobserved quantities (even parameters) just as we did with things like cluster assignment variables or continuous latent factors.

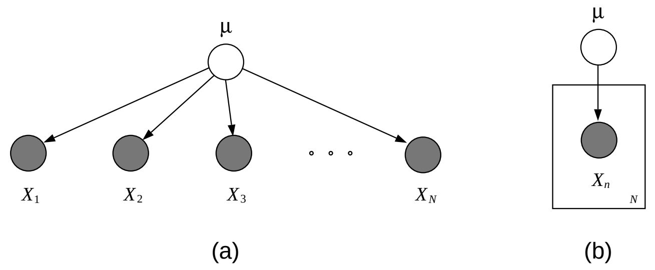
Three Key Regularization Ideas

- 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 (!).



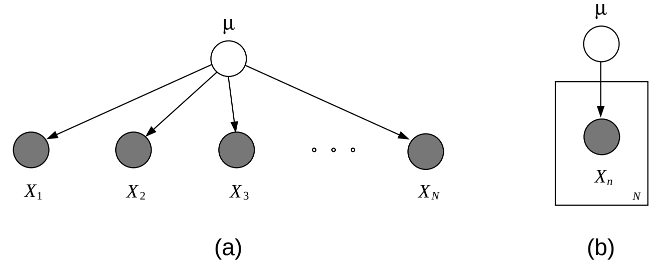
Plates

- Since Bayesian methods treat parameters as random variables, we would like to include them into the graphical model.
- One way to do this is to repeat all the iid observations explicitly and show the parameter only once.
- A better way is to use "plates", in which repeated quantities that are iid are put in a box.



Plates are Macros for Repeated Structure

- Plates are like "macros" that allow you to draw a very complicated graphical model with a simpler notation.
- The rules of plates are simple: repeat every structure in a box a number of times given by the integer in the corner of the box (e.g. N), updating the plate index variable (e.g. n) as you go.
- Duplicate every arrow going into the plate and every arrow leaving the plate by connecting the arrows to each copy of the structure.



Posterior Over Parameters

- If θ is a random variable, we can view the likelihood as a conditional probability and use Bayes rule:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$$

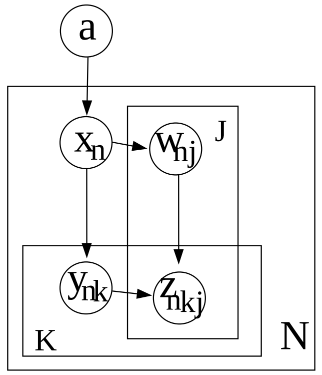
- This crucial equation can be written in words:

$$\textit{posterior} \propto \textit{likelihood} \times \textit{prior}$$

- Computing the posterior requires conditioning on the data and having a *prior* over parameters.
- In contrast, frequentists consider various "estimators" of θ and hope to show that they have desirable properties, e.g. ML, "unbiased", "minimum variance", etc.

Nested/Intersecting Plates

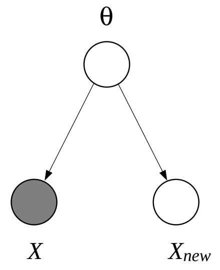
- Plates can be nested, in which case there arrows get duplicated also, according to the rule: draw an arrow from every copy of the source node to every copy of the destination node.
- Plates can also cross (intersect), in which case the nodes at the intersection have multiple indices and get duplicated a number of times equal to the product of the duplication numbers on all the plates containing them.



Model Averaging

- Posterior $p(\theta|\mathcal{D})$ is used in all future Bayesian computations.
- For example, to do prediction of a new value $\mathbf{x}_{\text{new}}$ given some iid data, we compute the conditional posterior:

$$\begin{aligned} p(\mathbf{x}_{\text{new}}|\mathbf{X}) &= \int p(\mathbf{x}_{\text{new}}, \theta|\mathbf{X})d\theta \\ &= \int p(\mathbf{x}_{\text{new}}|\theta, \mathbf{X})p(\theta|\mathbf{X})d\theta \\ &= \int p(\mathbf{x}_{\text{new}}|\theta)p(\theta|\mathbf{X})d\theta \end{aligned}$$



- This means the Bayesian prediction is based on averaging predictions from lots of models, weighted by the posterior probability of the model's parameters.

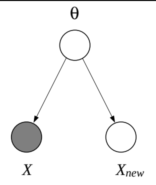
Multiple Model Averaging

- Imagine that we wanted to compute the probability of some new data (e.g. the density of a new point) taking into account the predictions of *all models*. We can compute:

$$\begin{aligned} p(\mathbf{x}_{\text{new}}|\mathbf{X}) &= \int\int p(\mathbf{x}_{\text{new}}, \theta, m|\mathbf{X}) d\theta dm \\ &= \int\int p(\mathbf{x}_{\text{new}}|\theta, m, \mathbf{X}) p(\theta, m|\mathbf{X}) d\theta dm \\ &= \int\int p(\mathbf{x}_{\text{new}}|\theta, m, \mathbf{X}) p(\theta|m, \mathbf{X}) p(m|\mathbf{X}) d\theta dm \end{aligned}$$

- This requires two posteriors, $p(m|\mathbf{X})$ (see later) and $p(\theta|\mathbf{x}, m)$.
- Remember: maximum likelihood alone cannot be used to do either model selection or model averaging since it always is subject to overfitting. Bayesian methods in principle can never overfit, since we integrate over all unknown quantities.

Integrate or Optimize?



- Normally, Bayesian statistics needs to perform an integral in order to do predictions.
  Frequentist statistics uses a "plug-in" estimator such as ML.
- We can be "pseudo-Bayesian" by using single estimators such as Bayes-point or MAP.
- Notice that both the Bayesian approach and the ML (frequentist) approach need to calculate the likelihood function $p(\mathbf{x}|\theta)$, which is what the graphical model specifies.
- So all the work we have done so far to is applicable to both Bayesian and ML frameworks.

ML vs. Maximum-A-Posteriori (MAP)

- If we forced a Bayesian to pick a *single* value for the parameters rather use the entire posterior $p(\theta|\mathcal{D})$, what would they do?
- *Bayes point* (mean of posterior):

$$\theta_{Bayes} = \int \theta p(\theta|\mathcal{D}) d\theta$$

- *MAP* (mode of posterior):

$$\begin{aligned} \theta_{MAP} &= \text{argmax}_\theta \;\; p(\theta|\mathcal{D}) \\ &= \text{argmax}_\theta \;\; \log p(\mathcal{D}|\theta) + \log p(\theta) \end{aligned}$$

- The *maximum a-posteriori* (MAP) estimate looks exactly maxmimum likelihood except for an extra term which depends only on the parameters.
- This is often called "penalized maximum likelihood", and it's what we've seen so far.

Example: Scalar Gaussian Model

- Consider a univariate Gaussian, with fixed, known variance σ^2.
- We want to put a prior $p(\mu)$ on the mean, μ and then compute its posterior, $p(\mu|\mathbf{X})$ using the Gaussian likelihood $p(\mathbf{X}|\mu)$.
- What should the prior be? Try another Gaussian:

$$p(\mu) = \frac{1}{2\pi\tau^2} \exp\left\{ -\frac{1}{2\tau^2}(\mu - \mu_0)^2 \right\} \quad = \mathcal{N}(\mu|\mu_0, \tau)$$

- Now the joint probability can be written as:

$$\begin{aligned} p(\mathbf{X}, \mu) &= p(\mathbf{X}|\mu)p(\mu) \\ &= \frac{1}{2\pi\sigma^2} \exp\left\{ -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 \right\} \frac{1}{2\pi\tau^2} \exp\left\{ -\frac{1}{2\tau^2}(\mu - \mu_0)^2 \right\} \end{aligned}$$

- We need to marginalize this joint with respect to μ to obtain the posterior $p(\mu|\mathbf{X})$. This normalization can be done using the conditional Gaussian formulas or by explicitly completing the square.

Scalar Gaussian: Posterior over the Mean

- Amazingly, the posterior is another Gaussian:

$$\begin{aligned} p(\mu|\mathbf{X}) &= \frac{p(\mathbf{X}|\mu)p(\mu)}{\int p(\mathbf{X},\mu)d\mu} \\ &= \frac{1}{2\pi s^2}\exp\left\{-\frac{1}{2s^2}(\mu-m)^2\right\} \\ &= \mathcal{N}(\mu|m,s^2) \\ m &= \frac{N/\sigma^2}{N/\sigma^2+1/\tau^2}\mu_{ML} + \frac{1/\tau^2}{N/\sigma^2+1/\tau^2}\mu_0 \\ s^2 &= \left(\frac{N}{\sigma^2}+\frac{1}{\tau^2}\right)^{-1} \end{aligned}$$

  where μ_{ML} is the sample mean.

Example: Multinomial

- Bayesian methods can also be used to estimate the density of discrete quantities (e.g. spam/nospam, shoe colour).
- If we use a *multinomial* distribution over K settings as the likelihood model, the *conjugate prior* is called the *Dirichlet distribution* defined as:

$$\begin{aligned} p(\theta) &= C(\alpha)\theta_1^{\alpha_1-1}\theta_2^{\alpha_2-1}\ldots\theta_K^{\alpha_K-1} \\ C(\alpha) &= \Gamma(\sum_k \alpha_k)/\prod_k \Gamma(\alpha_k) \end{aligned}$$

  where $\Gamma(\cdot)$ is the gamma function and the $\alpha-1$ is a convention.
- This is a funny density, because it is a density over the *simplex*, i.e. over vectors whose components are non-negative and sum to one.
- In the binary case, the multinomial becomes a *binomial* $p(x|\theta)=\theta^x(1-\theta)^{1-x}$ and the Dirichlet becomes a *beta* distribution $p(\theta)=C(\alpha)\theta^{\alpha_1-1}(1-\theta)^{\alpha_2-1}$.

Conjugate Priors

- In the example we just worked out, the posterior had the same form as the prior (both were Gaussian).
- When this happens, the prior is called the *conjugate prior* for the parameters with respect to the *likelihood function*.
- Conjugate priors are very nice to work with because the posterior and prior have the same parameter types and the effect of the data is just to update the parameters from the prior to the posterior.
- In these settings, the prior can often be interpreted as some "pseudo-data" which we observed before we saw the real data.
- Remember Laplace smoothing? That's just a pseudo-count of unity, which in turn is just a conjugate prior for the multinomial...

Multinomial Posterior

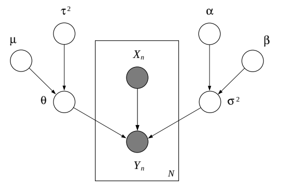
- The posterior is also of the form of a Dirichlet:

$$\begin{aligned} p(\theta|\mathbf{X}) \propto p(\mathbf{X}|\theta)p(\theta) &= \prod_k \theta_k^{\sum_n [x_n=k]}\theta_k^{\alpha_k-1} \\ &= \prod_k \theta_k^{\alpha_k-1+\sum_n [x_n=k]} \end{aligned}$$

  which has parameters $\alpha_k' = \alpha_k + \sum_n [x_n=k]$.
- We see that to update the prior into a posterior, we simply add the observed counts to the priors.
- So we can think of the priors as "pseudo-counts".

Heirarchical Bayes and Structure Learning

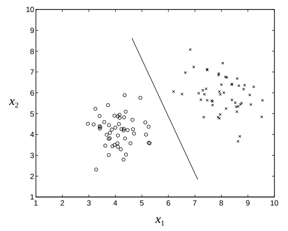
- What about the parameters of the parameter priors? In a full Bayesian formulation, they also have priors, called *hyperpriors* and we treat them in the same way.
- In theory we should do this upwards forever, but in practice we usually stop after only one or two levels.



- What about the model structure? In a full Bayesian formulation we also have a prior on that, and attempt to get a posterior. Sometimes this can be done (e.g. fully observed tree learning was just maximum likelihood over structure).

Back to 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 discriminant $p(y|\mathbf{x})$ directly and take max.
- Generative approach is related to conditional *density estimation* while discriminative approach is closer to *regression*.



Model Selection

- In principle we could do model structure learning in a Bayesian way also. Consider a fixed class of models, indexed by $m = 1 \ldots M$ (e.g. Gaussian mixtures with m components).
- Since m is unknown, the Bayesian way is to treat it as a random variable and to compute its posterior:

$$p(m|\mathbf{X}) = \frac{p(\mathbf{X}|m)p(m)}{p(\mathbf{X})}$$

- Notice that we require a prior $p(m)$ on models as well as the *marginal likelihood*:

$$p(\mathbf{X}|m) = \int p(\mathbf{X}, \theta|m)d\theta = \int p(\mathbf{X}|\theta, m)p(\theta|m)d\theta$$

- We could try to compute the model with the highest posterior, in which case we don't have to compute $p(\mathbf{X})$.
- Or else we could use all of the models, *weighted by their posteriors* to do predictions at test time. This was called "model averaging".

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 rule:
  MAP: $\mathrm{argmax}_y \; p(y|\mathbf{x}) = \mathrm{argmax}_y \; \log p(\mathbf{x}|y) + \log p(y)$
- 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).

Gaussian Class-Conditional Distributions

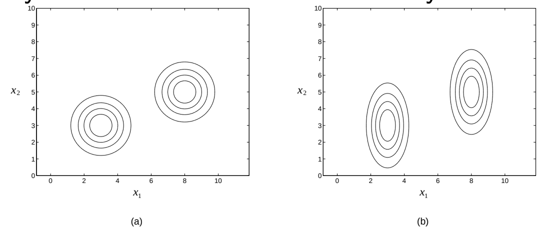
- If all features are continuous, a popular choice is a Gaussian class-conditional.
$$p(\mathbf{x}|y=k,\theta) = |2\pi\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)\Sigma^{-1}(\mathbf{x}-\mu_k)\right\}$$
- Fitting: use the following amazing and useful fact.
The maximum likelihood fit of a Gaussian 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...

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).

Regularized Gaussians

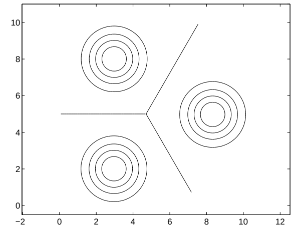
- Idea 1: assume all the covariances are the same (tie parameters).
This is exactly Fisher's linear discriminant analysis.

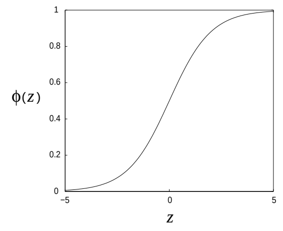


- Idea 2: Make independence assumptions to get diagonal or identity-multiple covariances. (Or sparse inverse covariances.)
More on this in a few minutes...
- Idea 3: add a bit of the identity matrix to each sample covariance.
This "fattens it up" in directions where there wasn't enough data.
Equivalent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

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

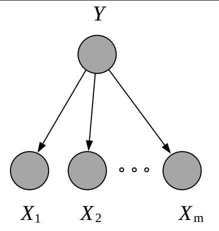




Log Linear Geometry

- 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.

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 $\operatorname{argmax}_c\, p(c|\mathbf{x})$ using
$$c(\mathbf{x}) = \operatorname{argmax}_c\, p(c|\mathbf{x}) = \operatorname{argmax}_c \left[\log p(\mathbf{x}|c) + \log p(c)\right] = \operatorname{argmax}_c \left[\log p(c) + \sum_i \log p(x_i|c)\right]$$

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?

Discrete (Multinomial) Naive Bayes

Discrete features x_i, assumed independent given the class label y.
$$p(x_i = j|y=k) = \eta_{ijk}$$
$$p(\mathbf{x}|y=k,\eta) = \prod_i \prod_j \eta_{ijk}^{[x_i=j]}$$

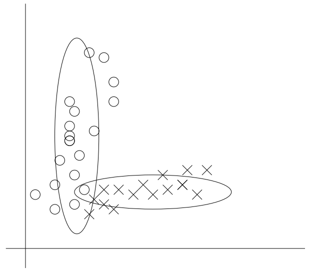
Classification rule:
$$p(y=k|\mathbf{x},\eta) = \frac{e^{\beta_k^\top \mathbf{x}}}{\sum_q e^{\beta_q^\top \mathbf{x}}} = \frac{\pi_k \prod_i \prod_j \eta_{ijk}^{[x_i=j]}}{\sum_q \pi_q \prod_i \prod_j \eta_{ijq}^{[x_i=j]}}$$

ML parameters are class-conditional frequency counts:
$$\eta^*_{ijk} = \frac{\sum_m [x_i{}^m = j][y^m = k]}{\sum_m [y^m = k]}$$

$\beta_k = \log[\eta_{11k} \ldots \eta_{1jk} \ldots \eta_{ijk} \ldots \log \pi_k]$ Log-Linear!
$\mathbf{x} = [x_1{=}1; x_1{=}2; \ldots; x_i{=}j; \ldots; 1]$

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...



Logistic/Softmax Regression

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

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

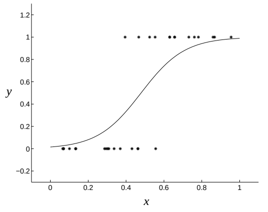
- Fitting: now we optimize the *conditional* likelihood:

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

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

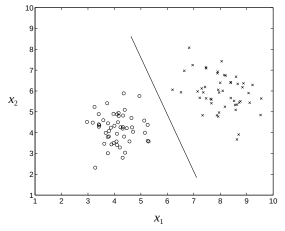
$$= \sum_{mk} \frac{y_k^m}{p_k^m} p_k^m (\delta_{ik} - p_i^m) \mathbf{x}^m$$

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



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.

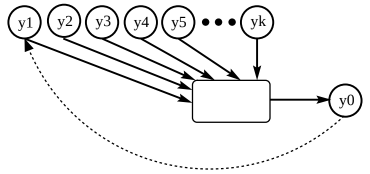


Chains: Markov Models

- 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:

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

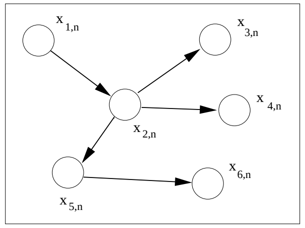
Directed Tree Graphical Models

- 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.



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*.

$$\begin{tikzcd}
s_1 \arrow[r] & s_2 \arrow[r] & s_3 \arrow[r] & \bullet\bullet\bullet \arrow[r] & s_T \\
f_1 \arrow[u] & f_2 \arrow[u] & f_3 \arrow[u] & & f_T \arrow[u]
\end{tikzcd}$$

- 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.

Likelihood function for Trees

- 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:

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

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

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

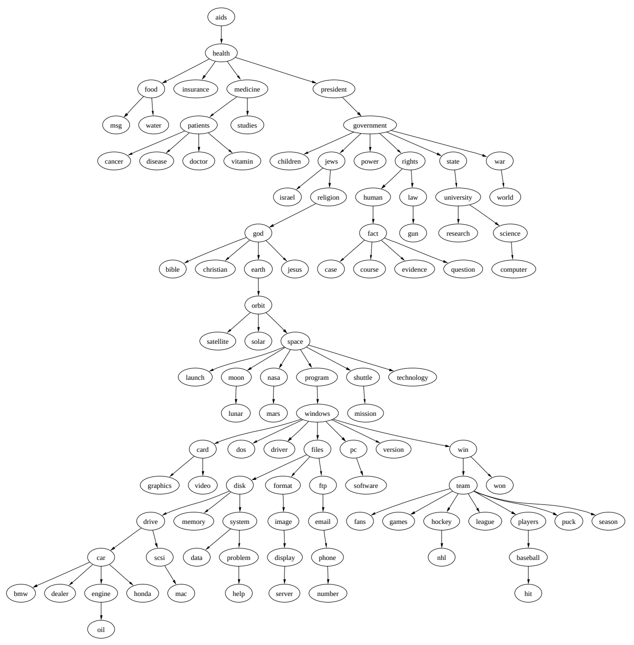
where $N_i(\mathbf{v}) = \sum_n [\mathbf{y}_i^n = \mathbf{v}]$ and $p_i(\mathbf{v}_i) = p(x_i | \mathbf{x}_{\pi_i})$.
- Trees are in the exponential family with $\mathbf{y}_i$ as sufficient statistics.

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