CSC 412/2506 Spring 2017

Probabilistic Graphical Models

Lecture 2: Generative Classifiers

Slides based on Rich Zemel's

All lecture slides will be available on the course website:
www.cs.toronto.edu/~duvenaud/courses/CS412

Some of the figures are provided by Kevin Murphy
from his book: "Machine Learning: A Probabilistic Perspective"

Basic Statistical Problems

- Basic problems: density est., clustering, classification, regression.
- Can always do joint density estimation and then condition:
  - Regression: $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y},\mathbf{x})/p(\mathbf{x}) = p(\mathbf{y},\mathbf{x})/\int p(\mathbf{y},\mathbf{x})d\mathbf{y}$
  - Classification: $p(c|\mathbf{x}) = p(c,\mathbf{x})/p(\mathbf{x}) = p(c,\mathbf{x})/\sum_c p(c,\mathbf{x})$
  - Clustering: $p(c|\mathbf{x}) = p(c,\mathbf{x})/p(\mathbf{x})$ c unobserved
  - Density estimation: $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{y},\mathbf{x})/p(\mathbf{x})$ $\boldsymbol{y}$ unobserved

In general, if certain things are *always* observed we may not want to model their density:

Regression/Classification

If certain things are always unobserved they are called *hidden* or *latent* variables (more later):

Clustering/Density Est.

Fundamental Operations

- What can we do with a probabilistic graphical model?
- *Generate* data.
  For this you need to know how to sample from local models (directed) or how to do Gibbs or other sampling (undirected)
- *Compute probabilities*.
  When all nodes are either observed or marginalized the result is a single number which is the prob. of the configuration.
- *Inference.*
  Compute expectations of some things given others which are observed or marginalized.
- *Learning.* (today)
  Set the parameters of the local functions given some (partially) observed data to maximize the probability of seeing that data

Learning Graphical Models from Data

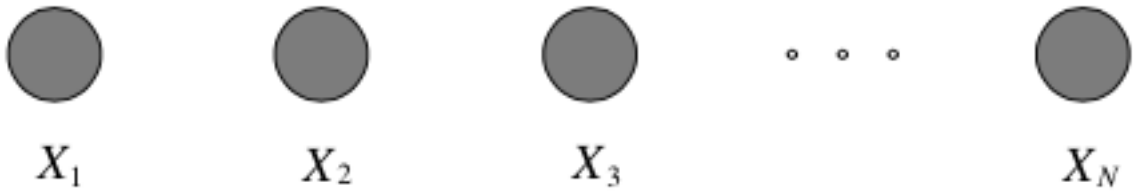
- Want to build prediction systems automatically based on data, and as little as possible on expert information
- In this course, we'll use probability to combine evidence from data and make predictions
- We'll use graphical models as a visual shorthand language to express and reason about families of model assumptions, structures, dependencies and information flow, without specifying exact distributional forms or parameters
- In this case learning ≡ setting parameters of distributions given a model structure.
  ("Structure learning" is also possible but we won't consider it now.)

Multiple Observations, Complete IID Data

- A single observation of the data $\mathbf{X}$ is rarely useful on its own.
- Generally we have data including many observations, which creates a set of random variables: $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(M)}\}$
- We will sometimes assume two things:

1. Observations are independently and identically distributed according to joint distribution of graphical model: *i.i.d. samples*.
2. We observe all random variables in the domain on each observation: *complete data*, or *fully observed model*.

- We shade the nodes in a graphical model to indicate they are observed. (Later we will work with unshaded nodes corresponding to missing data or latent variables.)



Likelihood Function

- 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 $\mathbf{x}$:

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

This function is called the (log) "likelihood".

- Choose θ to maximize some cost or loss function $L(\theta)$ which includes $\ell(\theta)$:

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

$L(\theta) = \ell(\theta; \mathcal{D}) + r(\theta)$ maximum a posteriori (MAP)/penalized ML

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

Maximum Likelihood

- For IID data, the log likelihood is a sum of identical functions

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

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

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

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

- Very commonly used 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.

Sufficient Statistics

- A statistic is a (possibly vector valued) deterministic function of a (set of) random variable(s).
- $T(\mathbf{X})$ is a "sufficient statistic" for $\mathbf{X}$ if

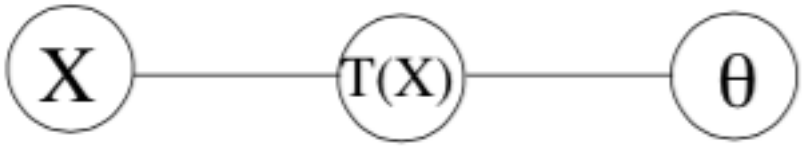
$$T(\mathbf{x}^{(1)}) = T(\mathbf{x}^{(2)}) \Rightarrow L(\theta; \mathbf{x}^{(1)}) = L(\theta; \mathbf{x}^{(2)}) \quad \forall \theta$$

- Equivalently (by the Neyman factorization theorem) we can write:

$$p(\mathbf{x}|\theta) = h(\mathbf{x}, T(\mathbf{x}))g(T(\mathbf{x}), \theta)$$

- Example: exponential family models:

$$p(\mathbf{x}|\eta) = h(\mathbf{x}) \exp\{\eta^T T(\mathbf{x}) - A(\eta)\}$$



Example: Bernoulli Trials

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

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

- Take derivatives and set to zero:

$$\frac{\partial \ell}{\partial \theta} = \frac{N_H}{\theta} - \frac{N_T}{1-\theta}$$

$$\Rightarrow \theta^*_{ML} = \frac{N_H}{N_H + N_T}$$

Example: Multinomial

- We observe M iid die rolls (K-sided): $\mathcal{D} = 3, 1, K, 2, \ldots$
- Model: $p(k) = \theta_k \qquad \sum_k \theta_k = 1$

- Likelihood (for binary indicators $[\mathbf{x}^{(m)} = k]$):

$$\begin{aligned}\ell(\theta; \mathcal{D}) &= \log p(\mathcal{D}|\theta) \\ &= \log \prod_m \theta_1^{[\mathbf{x}^{(m)}=k]} \ldots \theta_k^{[\mathbf{x}^{(m)}=k]} \\ &= \sum_k \log \theta_k \sum_m [\mathbf{x}^{(m)} = k] = \sum_k N_k \log \theta_k\end{aligned}$$

- Take derivatives and set to zero (enforcing $\sum_k \theta_k = 1$):

$$\frac{\partial \ell}{\partial \theta_k} = \frac{N_k}{\theta_k} - M$$

$$\Rightarrow \theta_k^* = \frac{N_k}{M}$$

Example: Univariate Normal

- We observe M iid real samples: $\mathcal{D} = 1.18, -.25, .78, \ldots$
- Model:

$$p(x) = (2\pi\sigma^2)^{-1/2} \exp\{-(x-\mu)^2/2\sigma^2\}$$

- Likelihood (using probability density):

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

- Take derivatives and set to zero :

$$\frac{\partial \ell}{\partial \mu} = \frac{1}{\sigma^2}\sum_m (x_m - \mu) \qquad \frac{\partial \ell}{\partial \sigma^2} = -\frac{M}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_m (x_m - \mu)^2$$

$$\Rightarrow \mu_{ML} = (1/M)\sum_m x_m \qquad \Rightarrow \sigma^2_{ML} = (1/M)\sum_m x_m^2 - \mu^2_{ML}$$

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 parents we use the model:

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

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

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

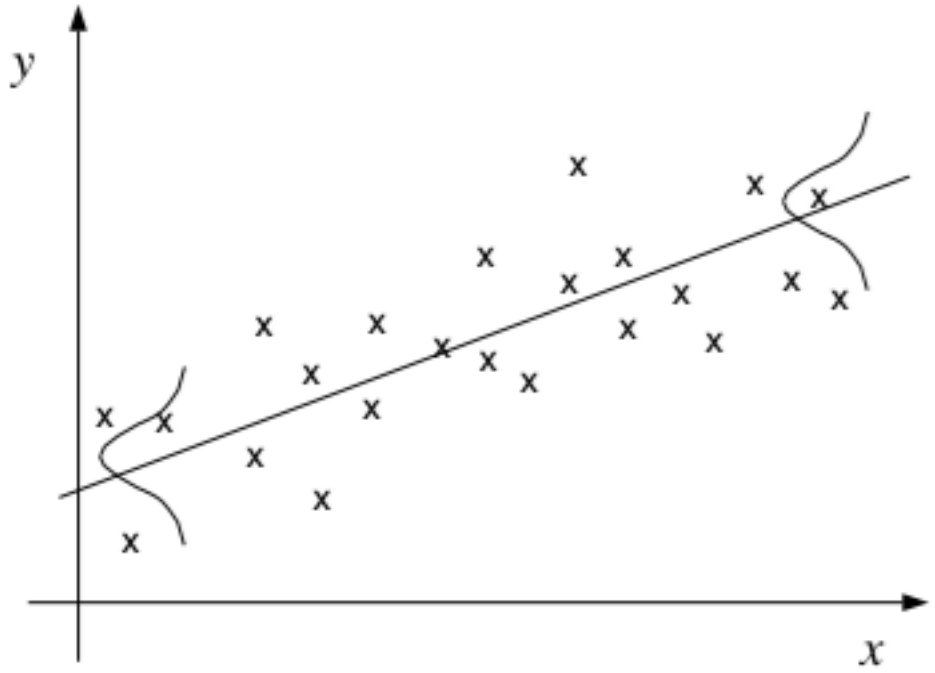
- The ML parameters can be solved for using linear least-squares:

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

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

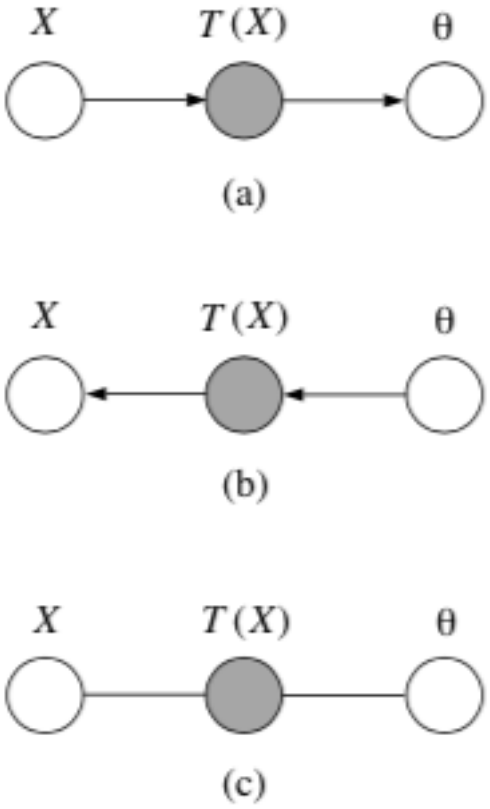
- Sufficient statistics are input correlation matrix and input-output cross-correlation vector.

Example: Linear Regression



Sufficient Statistics are Sums

- In the examples above, the sufficient statistics were merely sums (counts) of the data:
  - Bernoulli: # of heads, tails
  - Multinomial: # of each type
  - Gaussian: mean, mean-square
  - Regression: correlations
- As we will see, this is true for all exponential family models: sufficient statistics are the average natural parameters.
- Only exponential family models have simple sufficient statistics.

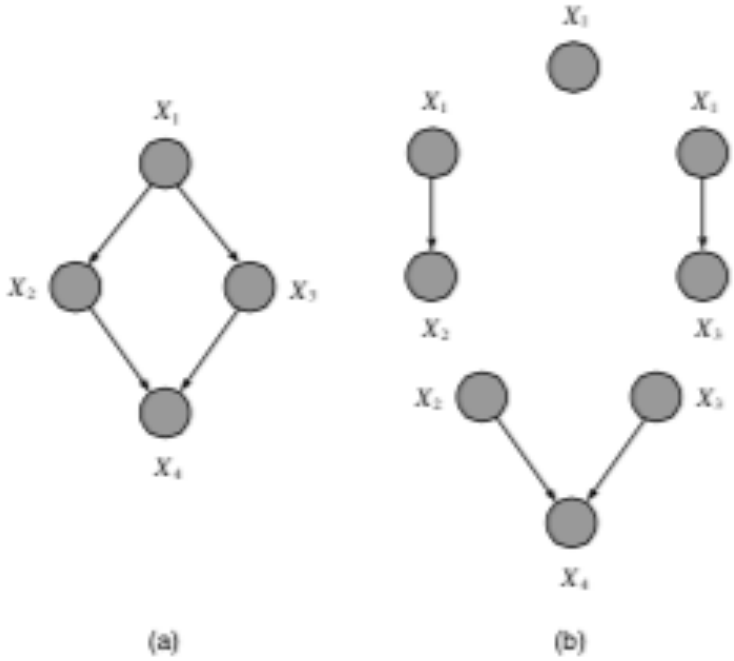


MLE for Directed GMs

- 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 $\mathbf{x}_i$ and its parents in order to estimate θ_i
- Furthermore, if $\mathbf{x}_i, \mathbf{x}_{\pi_i}$ have sufficient statistics only need those.
- 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.

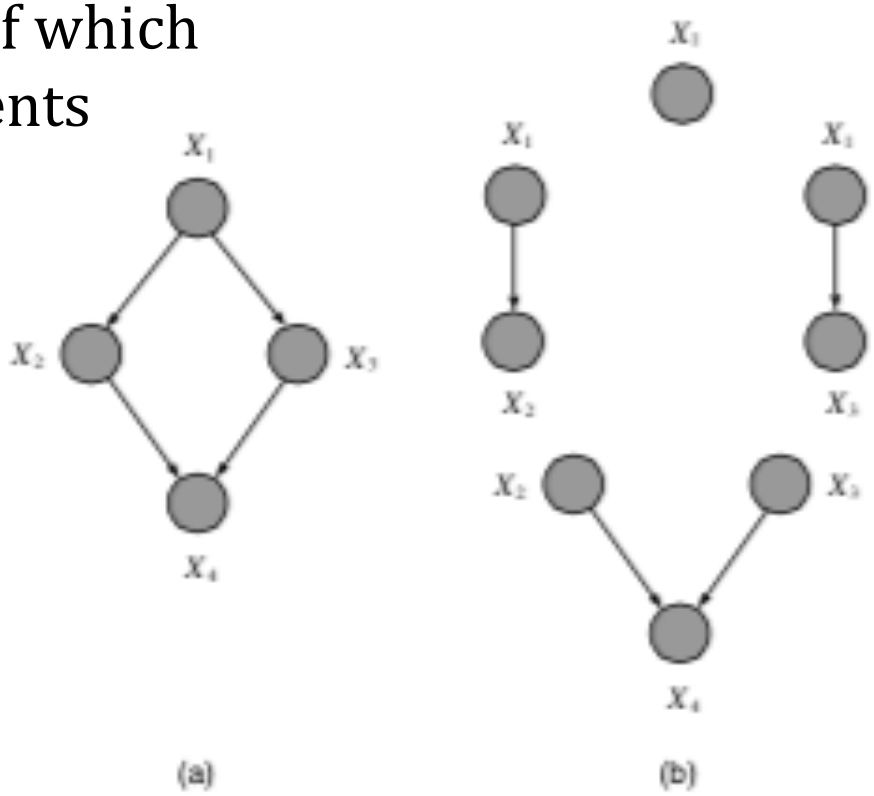


Example: A Directed Model

- Consider the distribution defined by the DAGM:

$$p(\mathbf{x}|\theta) = p(\mathbf{x}_1|\theta_1)p(\mathbf{x}_2|\mathbf{x}_1, \theta_2)p(\mathbf{x}_3|\mathbf{x}_1, \theta_3)p(\mathbf{x}_4|\mathbf{x}_2, \mathbf{x}_3, \theta_4)$$

- This is exactly like learning four separate small DAGMs, each of which consists of a node and its parents



MLE for Categorical Networks

- Assume our DAGM contains only discrete nodes, and we use the (general) categorical form for the conditional probabilities.
- Sufficient statistics involve counts of joint settings of $\mathbf{x}_i, \mathbf{x}_{\pi_i}$ summing over all other variables in the table.
- Likelihood for these special "fully observed categorical networks":

$$\ell(\eta; \mathcal{D}) = \log \prod_{m,i} p(\mathbf{x}_i^{(m)} | \mathbf{x}_{\pi_i}^{(m)}, \theta_i)$$

$$= \log \prod_{i, \mathbf{x}_i, \mathbf{x}_{\pi_i}} p(\mathbf{x}_i | \mathbf{x}_{\pi_i}, \theta_i)^{N(\mathbf{x}_i, \mathbf{x}_{\pi_i})} = \log \prod_{i, \mathbf{x}_i, \mathbf{x}_{\pi_i}} \theta_{\mathbf{x}_i | \mathbf{x}_{\pi_i}}^{N(\mathbf{x}_i, \mathbf{x}_{\pi_i})}$$

$$= \sum_i \sum_{\mathbf{x}_i, \mathbf{x}_{\pi_i}} N(\mathbf{x}_i, \mathbf{x}_{\pi_i}) \log \theta_{\mathbf{x}_i | \mathbf{x}_{\pi_i}}$$

$$\Rightarrow \theta^*_{\mathbf{x}_i | \mathbf{x}_{\pi_i}} = \frac{N(\mathbf{x}_i, \mathbf{x}_{\pi_i})}{N(\mathbf{x}_{\pi_i})}$$

MLE for General Exponential Family Models

- Recall the probability function for models in the exponential family:

$$p(\mathbf{x}|\theta) = h(\mathbf{x}) \exp\{\eta^T T(\mathbf{x}) - A(\eta)\}$$

- For i.i.d. data, the sufficient statistic vector is

$$\ell(\eta; \mathcal{D}) = \log p(\mathcal{D}|\eta) = \left(\sum_m \log h(\mathbf{x}^{(m)}) \right) - MA(\eta) + \left(\eta^T \sum_m T(\mathbf{x}^{(m)} \right)$$

- Take derivatives and set to zero:

$$\frac{\partial \ell}{\partial \eta} = \sum_m T(\mathbf{x}^{(m)}) - M \frac{\partial A(\eta)}{\partial \eta}$$

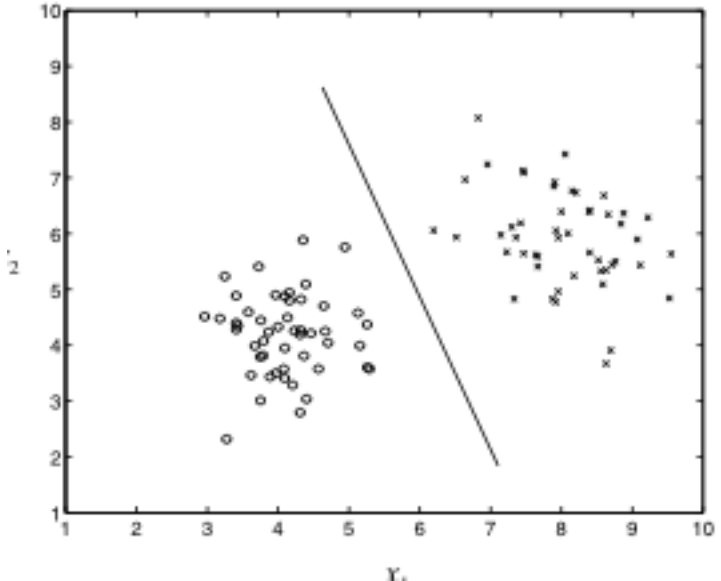
$$\Rightarrow \frac{\partial A(\eta)}{\partial \eta} = 1/M \sum_m T(\mathbf{x}^{(m)})$$

$$\eta_{ML} = 1/M \sum_m T(\mathbf{x}^{(m)})$$

recalling that the natural moments of an exponential distribution are the derivatives of the log normalizer.

Classification, Revisited

- Given examples of a discrete *class label y* and some *features* **x**.
- Goal: compute label (y) for new inputs **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 is closer to *regression*



Probabilistic Classification: Bayes Classifier

- Generative model: $p(\mathbf{x}, y) = p(y)p(\mathbf{x}|y)$
  $p(y)$ are called class priors (relative frequencies).
  $p(\mathbf{x}|y)$ are called class-conditional feature distributions
- For the class frequency prior we use a Bernoulli or categorical:

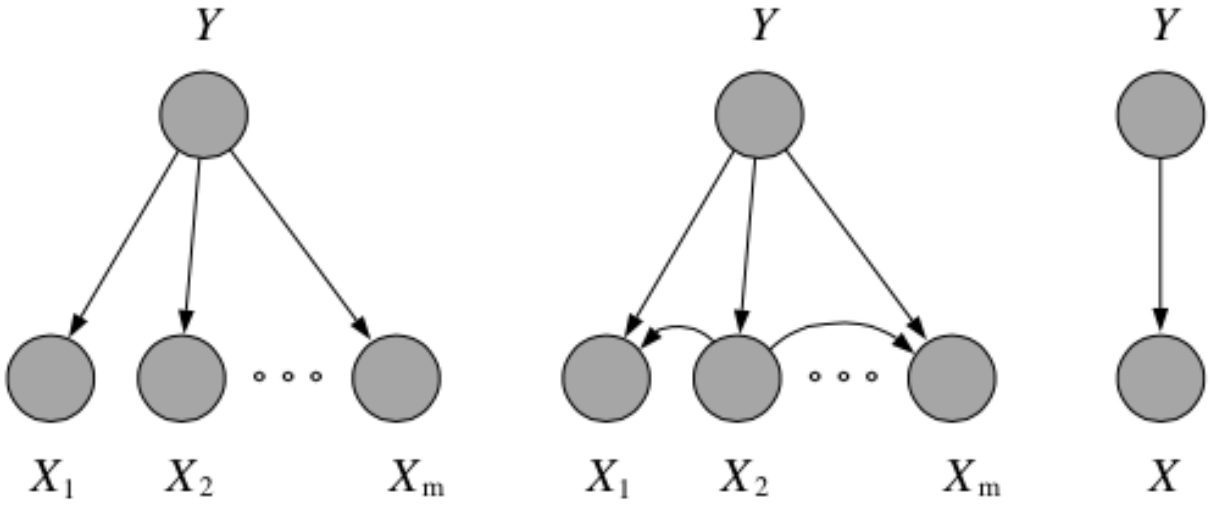
$$p(y = k|\pi) = \pi_k \qquad \sum_k \pi_k = 1$$

- Fitting by maximum likelihood:
  - Sort data into batches by class label
  - Estimate $p(y)$ by counting size of batches (plus regularization)
  - Estimate $p(\mathbf{x}|y)$ separately within each batch using ML (also with regularization)
- Two classification rules (if forced to choose):
  - ML: $\text{argmax}_y\ p(\mathbf{x}|y)$ (can behave badly if skewed frequencies)
  - MAP: $\text{argmax}_y\ p(y|\mathbf{x}) = \text{argmax}_y\ \log p(\mathbf{x}|y) + \log p(y)$ (safer)

Three Key Regularization Ideas

To avoid overfitting, we can:

- put *priors* on the parameters. Maximum likelihood + priors = maximum a posteriori (MAP). Simple and fast. Not Bayesian.
- Integrate over all possible parameters. Also requires priors, but protects against overfitting for totally different reasons.
- Make *factorization* or *independence* assumptions. Fewer inputs to each conditional probability. Ties parameters together so that fewer of them are estimated.



Gaussian Class-Conditional Distribution

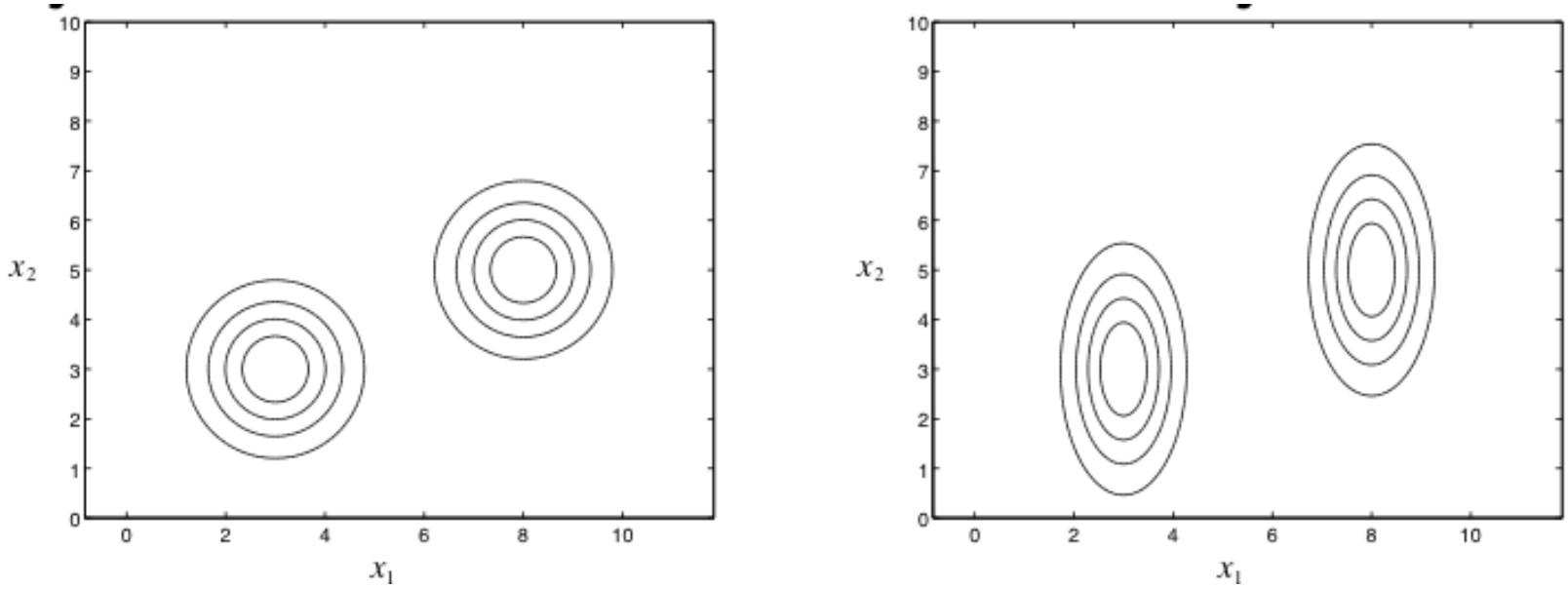
- If all features are continuous, a popular choice is a Gaussian class-conditional.

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

- 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 all directions and prevents collapse. Related 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:

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

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

Softmax/Logit

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

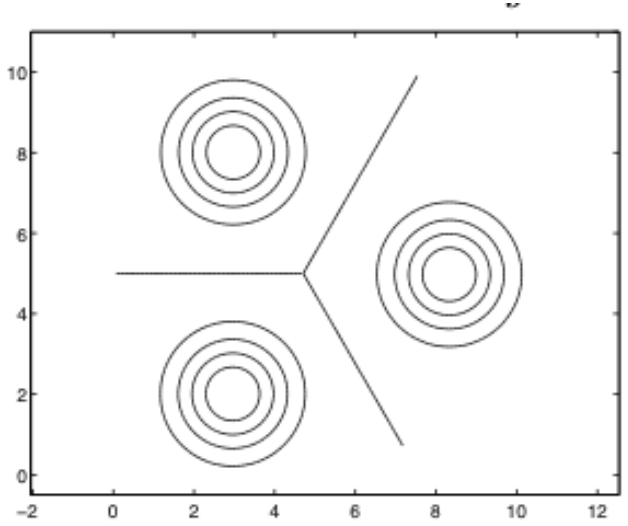
$$\phi_k(\mathbf{z}) \equiv \frac{e^{z_k}}{\sum_j e^{z_j}} \qquad g(\eta) = \frac{1}{1+e^{-\eta}}$$

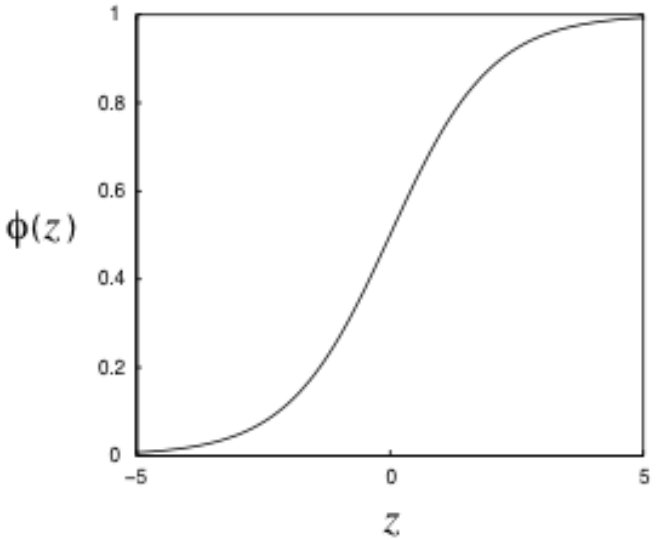
- It is invertible (up to a constant):

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

- Derivative is easy:

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





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:

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

- 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 $\mathbf{t} = \Sigma^{-1}\mathbf{x}$.
- Class 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.

Exponential Family Class-Conditionals

- Bayes Classifier has the same softmax form whenever the class-conditional densities are *any* exponential family density:

$$p(\mathbf{x}|y=k,\eta_k) = h(\mathbf{x})\exp\{\eta_k^T\mathbf{x} - a(\eta_k)\}$$

$$p(y=k|\mathbf{x},\eta) = \frac{p(\mathbf{x}|y=k,\eta_k)p(y=k|\pi)}{p(\mathbf{x}|y=j,\eta_j)p(y=j|\pi)}$$

$$= \frac{\exp\{\eta_k^T\mathbf{x} - a(\eta_k)\}}{\sum_j \exp\{\eta_j^T\mathbf{x} - a(\eta_j)\}}$$

$$= \frac{e^{\beta_k^T\mathbf{x}}}{\sum_j e^{\beta_k^T\mathbf{x}}}$$

- Where $\beta_k = [\eta_k; -a(\eta_k)]$ and we have augmented x with a constant component always equal to 1 (bias term)
- Resulting classifier is linear in the sufficient statistics

Discrete Bayesian Classifier

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

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

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

$$\eta^c_{ab...} = \frac{\sum_n [y^{(n)} = c][x_1 = a][x_2 = b][...][...]}{\sum_n [y^{(n)} = c]}$$

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

Naïve 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

$$c(\mathbf{x}) = \arg\max_c p(c|\mathbf{x}) = \arg\max_c [\log p(\mathbf{x}|c) + \log p(c)]$$

$$= \arg\max_c [\log p(c) + \sum_i \log p(x_i|c)]$$

Discrete (Categorical) Naïve Bayes

- Discrete features x_i assumed independent given class label y

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

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

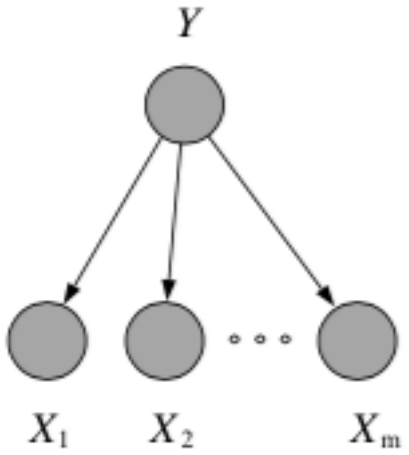
- Classification rule:

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

$$= \frac{e^{\beta_k^T \mathbf{x}}}{\sum_q e^{\beta_q^T \mathbf{x}}}$$

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

$$\mathbf{x} = [x_1 = 1; x_2 = 2; ...x_i = j; ...; 1]$$



Fitting Discrete Naïve Bayes

- ML parameters are class-conditional frequency counts:

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

- How do we know? Write down the likelihood:

$$\ell(\eta; \mathcal{D}) = \sum_m \log p(y^{(m)}|\pi) + \sum_{m,i} \log p(x_i^{(m)}|y^{(m)}, \eta)$$

- and optimize it by setting its derivative to zero (careful! enforce normalization with Lagrange multipliers):

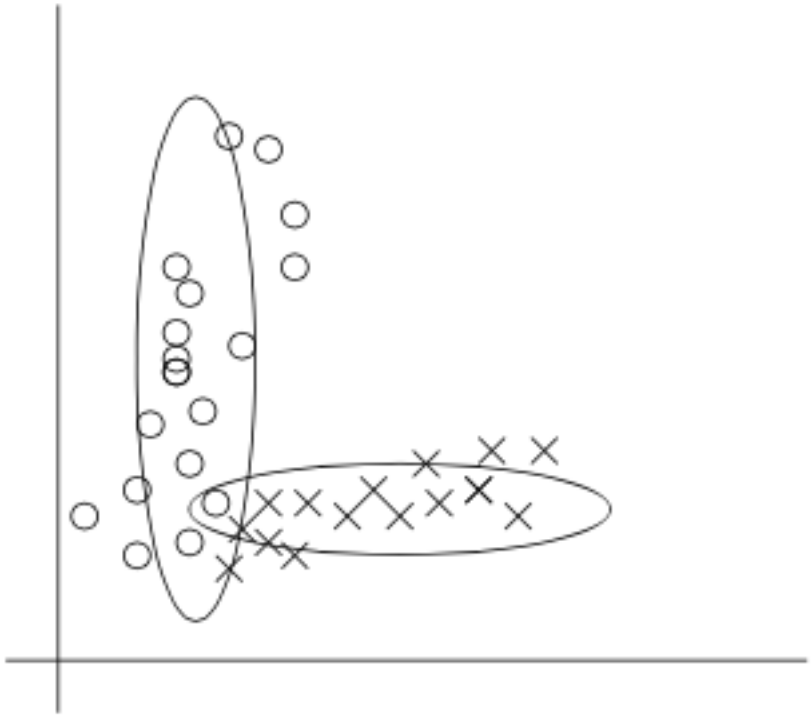
$$\ell(\eta; \mathcal{D}) = \sum_m \sum_{ijk} [x_i^{(m)} = j][y^{(m)} = k] \log \eta_{ijk} + \sum_{ik} \lambda_{ik} (1 - \sum_j \eta_{ijk})$$

$$\frac{\partial \ell}{\partial \eta_{ijk}} = \frac{\sum_m [x_i^{(m)} = j][y^{(m)} = k]}{\eta_{ijk}} - \lambda_{ik}$$

$$\frac{\partial \ell}{\partial \eta_{ijk}} = 0 \quad \Rightarrow \lambda_{ik} = \sum_m [y^{(m)} = k] \quad \Rightarrow \eta^*_{ijk} = \text{above}$$

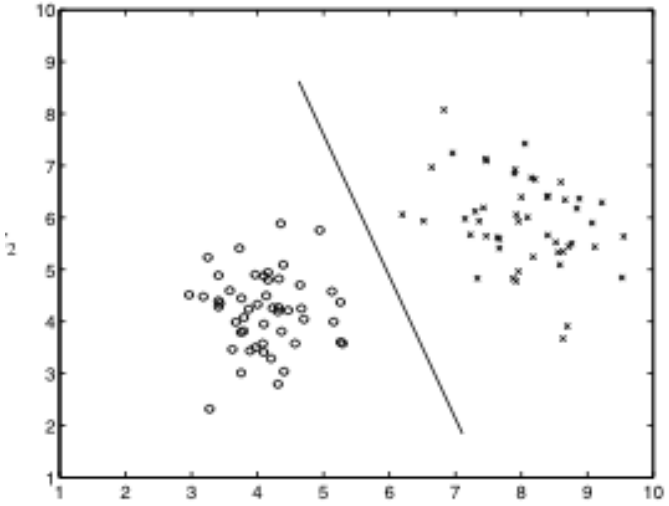
Gaussian Naïve Bayes

- 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 the same cost function 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.

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

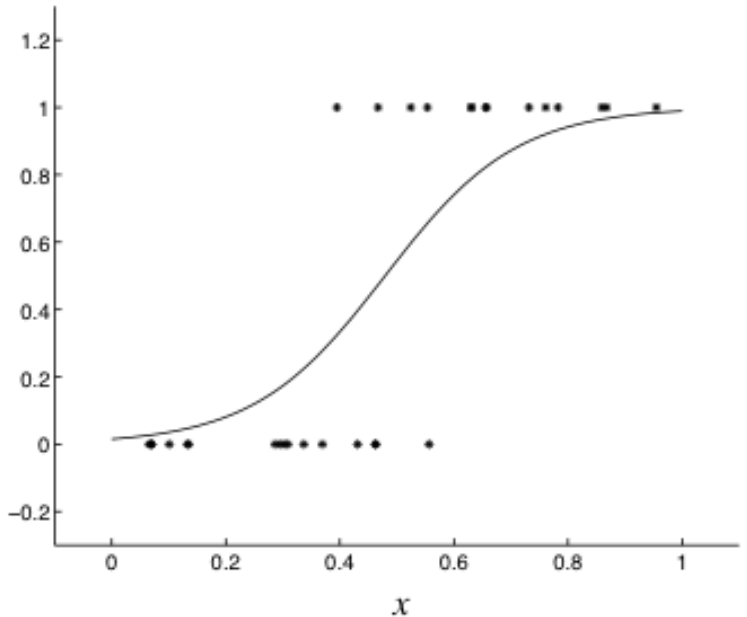
- Fitting: now we optimize the conditional likelihood:

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

$$\frac{\partial \ell}{\partial \theta_i} = \sum_{m,k} \frac{\partial \ell_k^{(m)}}{\partial p_k^{(m)}} \frac{\partial p_k^{(m)}}{\partial z_i^{(m)}} \frac{\partial z_i^{(m)}}{\partial \theta_i}$$

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

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



More on Logistic Regression

- Hardest Part: picking the feature vector **x**.
- The likelihood is convex in the parameters θ. No local minima!
- Gradient is easy to compute; so easy to optimize using gradient descent or Newton-Raphson.
- Weight decay: add $\varepsilon\theta^2$ to the cost function, which subtracts $2\varepsilon\theta$ from each gradient

- Why is this method called logistic regression?
- It should really be called "softmax linear regression".
- Log odds (logit) between any two classes is linear in parameters.
- A classification neural net always has linear regression as the last layer – no hidden layers = logistic regression

Classification via Regression

- Binary case: $p(y = 1|\mathbf{x})$ is also the conditional expectation.
- So we could forget that y was a discrete (categorical) random variable and just attempt to model $p(y|\mathbf{x})$ using regression.
- One idea: do regression to an indicator matrix.
- For two classes, this is related to LDA. For 3 or more, disaster…
- Weird idea. Noise models (e.g., Gaussian) for regression are inappropriate, and fits are sensitive to outliers. Furthermore, gives unreasonable predictions < 0 and > 1.

Other Models

- Noisy-OR (see slides)
- Classification via Regression (see slides)
- Non-parametric (e.g. K-nearest-neighbor).
- Semi-parametric (e.g. kernel classifiers, support vector machines, Gaussian processes).
- Probit regression.
- Complementary log-log.
- Generalized linear models.
- Some return a value for *y* without a distribution.

Joint vs. Conditional Models

- Both Naïve Bayes and logistic regression have same conditional form and can have same parameterization.
- But the criteria used to choose parameters is different

- Naive Bayes is a joint model; it optimizes

$$p(\mathbf{x}, y) = p(\mathbf{x})p(y|\mathbf{x}).$$

- Logistic Regression is conditional: optimizes $p(y|\boldsymbol{x})$ directly

- Pros of discriminative: More flexible, directly optimizes what we care about. Why not choose optimal parameters?
- Pros of generative: Easier to think about, check model, and incorporate other data sources (semi-sup learning)

Joint vs. Conditional Models: Yin and Yang

- Each generative model implicitly defines a conditional model
- $p(z|x)$ has complicated form if $p(x|z)$ is at all complicated. Expensive to compute naively, necessary for learning.

- Autoencoders: Given interesting generative model $p(x|z)$, force approximate $q(z|x)$ to have a nice form.
- So, designing inference methods for generative models involves designing discriminative recognition networks.

- Thursday: Tutorial on optimization.