GLMs (wrap-up) and Generative Models for Supervised Learning

CS772A: Probabilistic Machine Learning

Piyush Rai

Plan for today and announcements

- Wrap up GLM
- Testing conditional independence in directed graphical models
- Generative models for supervised learning
- Also, Quiz 1 on Monday (Sept 5) at 7pm
- Rescheduled class on Saturday (Sept 3) at 6pm
- Mid-sem exam on Sept 19 in L17 (18:00-20:00), ERES seating arrangement

GLM with Canonical Response Function

- For GLM with Canon Resp Func (a.k.a., canonical GLM)

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

The simple form of canonical GLM (nat. param just a linear function $\boldsymbol{w}^\top \boldsymbol{x}$) makes parameter estimation via MLE/MAP easy since gradient and Hessian have simple expressions (though the Hessian may be expensive to compute/invert)

- Consider doing MLE (assuming N i.i.d. responses). The log likelihood

$$L(\eta) = \log p(Y|\eta) = \log \prod_{n=1}^{N} h(y_n)\exp(y_n \mathbf{w}^\top \mathbf{x}_n - A(\eta_n)) = \sum_{n=1}^{N} \log h(y_n) + \mathbf{w}^\top \sum_{n=1}^{N} y_n \mathbf{x}_n - \sum_{n=1}^{N} A(\eta_n)$$

- Convexity of $A(\eta)$ guarantees a global optima. Gradient of log-lik w.r.t. $\boldsymbol{w}$

$$\mathbf{g} = \sum_{n=1}^{N}\left(y_n \mathbf{x}_n - A'(\eta_n)\frac{d\eta_n}{d\mathbf{w}}\right) = \sum_{n=1}^{N}(y_n \mathbf{x}_n - \mu_n \mathbf{x}_n) = \sum_{n=1}^{N}(y_n - \mu_n)\mathbf{x}_n$$

Exp of suff-stats $\mathbb{E}[y_n]$

Corrective updates for $\boldsymbol{w}$

The Hessian can also be shown to be

$$\mathbf{H} = -\nabla \mathbf{g} = \sum_{n=1}^{N} f'(\eta_n)\boldsymbol{x}_n\boldsymbol{x}_n^\top$$

- Note $\mu_n = f(\xi_n) = f(\boldsymbol{w}^\top \boldsymbol{x}_n)$ and $f = \psi^{-1}$ ("inverse link") depends on the model
  - Real-valued y (linear regression): f is identity, i.e., $\mu_n = \boldsymbol{w}^\top \boldsymbol{x}_n$
  - Binary y (logistic regression): f is sigmoid function, i.e., $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$
  - Count-valued y (Poisson regression): f is exp, i.e., $\mu_n = \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)$
  - Non-negative y (gamma regression): f is inverse negative i.e., $\mu_n = -1/(\boldsymbol{w}^\top \boldsymbol{x}_n)$

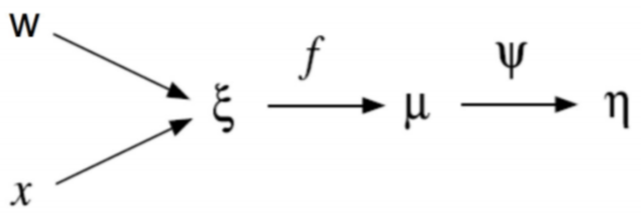
Fully Bayesian Inference for GLMs

- Most GLMs, except linear regression with Gaussian lik. and Gaussian prior, do not have conjugate pairs of likelihood and priors (recall logistic regression)

- Posterior over the weight vector $\boldsymbol{w}$ is intractable

- Approximate inference methods needed, e.g.,
  - Laplace approximation (have already seen): Easily applicable since derivatives (first and second) can be easily computed (note that we need $\boldsymbol{w}_{MAP}$ and Hessian)
  - MCMC or variational inference (will see later)

Various Types of GLMs



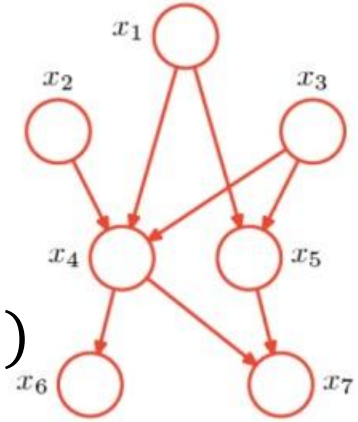
Type of response	Type of GLM	Link Function Ψ	Response Function f (Inv Link Func if canon. GLM) (Operates on $\xi = \boldsymbol{w}^\top \boldsymbol{x}$)	Activation
Real	Gaussian	Identity	Identity	Linear
Binary	Logistic	Log-odds: $\log \frac{\mu}{1-\mu}$	Sigmoid	Sigmoid
Binary	Probit	Inv CDF: $\Phi^{-1}(\mu)$	Φ (CDF of N(0,1))	Probit
Categorical	Multinoulli	Log-odds: $\log \frac{\mu_k}{1-\mu_k}$	Softmax	Softmax
Count	Poisson	$\log \mu$	exp	
Non-negative real	gamma	Negative of inverse	Negative of inverse	
Binary	Gumbel	Gumbel Inv CDF: log(-log())	Gumbel CDF: exp(-exp(-))	

.. and several others (exponential, inverse Gaussian, Binomial, Tweedie, etc)

(Directed) Graphical Models

- Most models we study can be represented via directed graphical models (DGMs)
- A DGM is a graph with nodes denoting random variables and edges their dependences

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$$
$$= p(x_1)p(x_2)p(x_3)p(x_4|x_2, x_3)$$
$$p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$



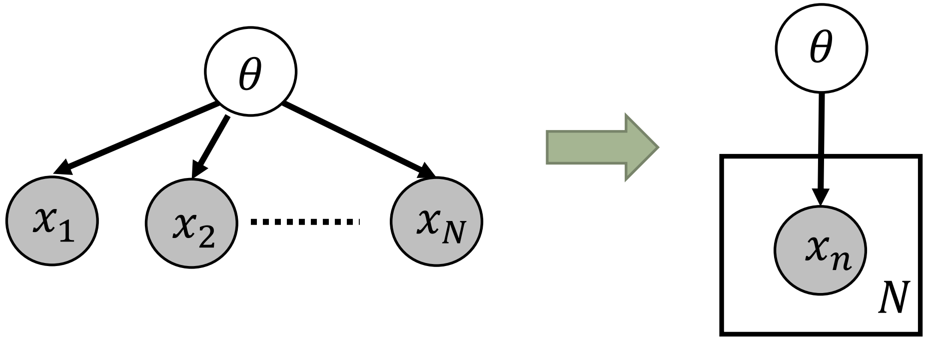
$$\mathbf{x} = (x_1, x_2, \ldots, x_K)$$

$$p(\mathbf{x}) = \prod_{i=1}^{K} p(x_i|\text{pa}(x_i))$$

Joint distribution of $\mathbf{x}$ written as a product of local conditional distributions

Set of parent nodes of x_i in the DGM

- Plate notation is a compact way of representing DGMs, e.g.,



Also remember: shaded = observed, unshaded = unknown

DGMs and Independence

- Goal: Test if two nodes $\boldsymbol{a}$ and $\boldsymbol{b}$ are independent in presence of a third node $\boldsymbol{c}$

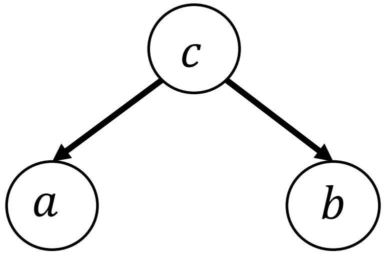
Summing over to get this since $\boldsymbol{c}$ is unobserved

$$p(a,b,c) = p(a|c)p(b|c)p(c)$$

$$p(a,b) = \sum_c p(a|c)p(b|c)p(c)$$

Thus $\boldsymbol{a}$ and $\boldsymbol{b}$ not marginally independent

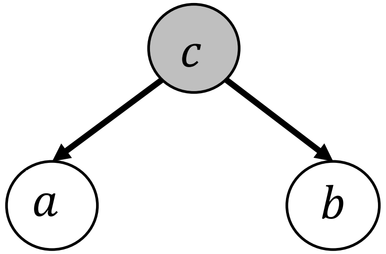
$$\neq p(a)p(b)$$



$$p(a,b|c) = p(a,b,c)|p(c)$$

$$= p(a|c)p(b|c)$$

Thus $\boldsymbol{a}$ and $\boldsymbol{b}$ are conditionally independent given $\boldsymbol{c}$

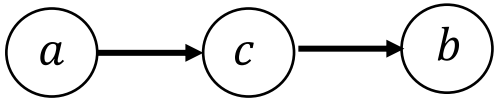


$$p(a,b,c) = p(a)\,p(b,c|a)$$

$$p(a,b) = p(a)\sum_c p(b,c|a)$$

Thus $\boldsymbol{a}$ and $\boldsymbol{b}$ not marginally independent

$$= p(a)p(b|a)$$

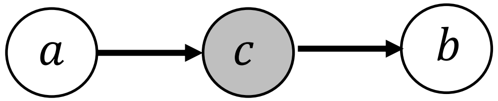


$$p(a,b,c) = p(a,c)\,p(b|c)$$

$$p(a,b|c) = \frac{p(a,c)\,p(b|c)}{p(c)}$$

$$= p(a|c)\,p(b|c)$$

Thus $\boldsymbol{a}$ and $\boldsymbol{b}$ are conditionally independent given $\boldsymbol{c}$

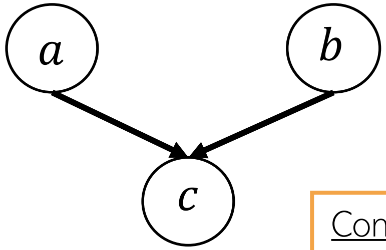


$$p(a,b,c) = p(a)p(b)p(c|a,b)$$

$$p(a,b) = p(a)p(b)\sum_c p(c|a,b)$$

Thus $\boldsymbol{a}$ and $\boldsymbol{b}$ are independent

$$= p(a)p(b)$$

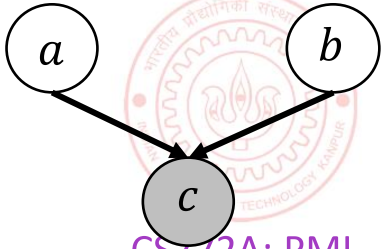


$$p(a,b,c) = p(a)p(b)p(c|a,b)$$

$$p(a,b|c) = \frac{p(a)p(b)p(c|a,b)}{p(c)}$$

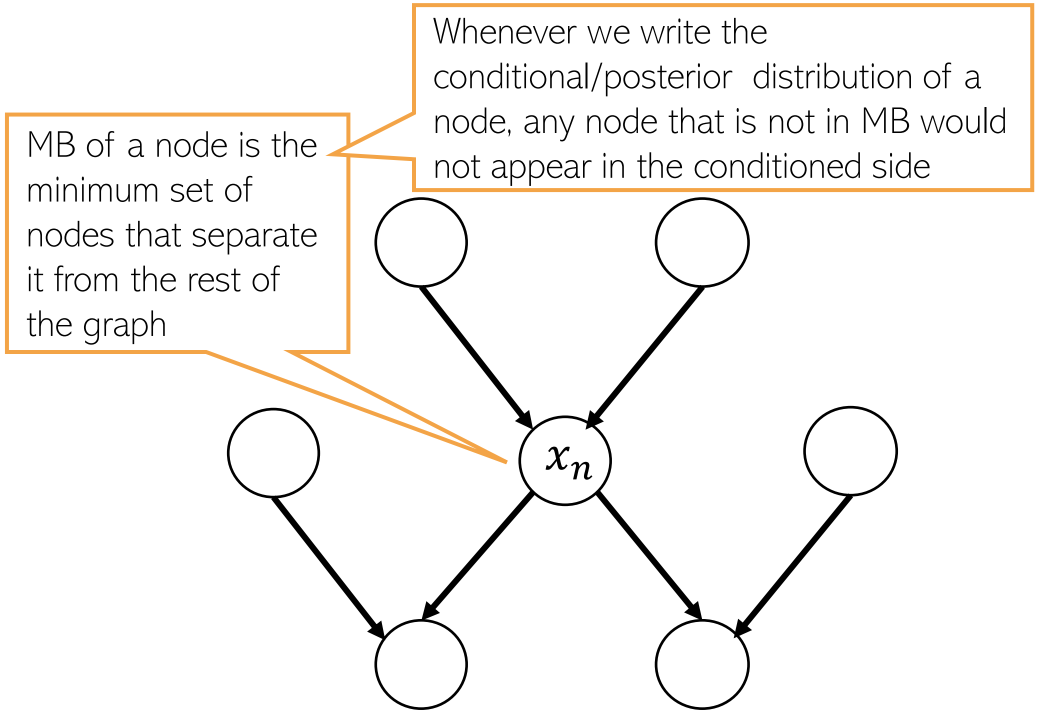
Conditionally NOT independent given $\boldsymbol{c}$

$$\neq p(a|c)p(b|c)$$



DGMs and Independence

- A node in a DGM is independent of all other nodes given its Markov Blanket
- Markov Blanket (MB) consists of a nodes parents, children, and co-parents



Generative Supervised Learning

Have already seen the discriminative approach to learn $p(y|\boldsymbol{x})$ (prob linear regression, logistic regression, and also GLM)

- Want to learn the conditional distribution $p(y|\boldsymbol{x})$ for supervised learning (reg/class.)
- Generative approach assumes a model for the joint distribution $p(\boldsymbol{x}, y)$
- Assuming a classification setting, we can write

All these distributions here, $p(y), p(\boldsymbol{x}|y)$ (and consequently $p(y|\boldsymbol{x})$ too) depend on some params/hyperparams (not shown for brevity)

Can compute it for each possible value of $y \in \{1,2,\ldots,K\}$

Class-prior distribution (a discrete distribution since y is discrete)

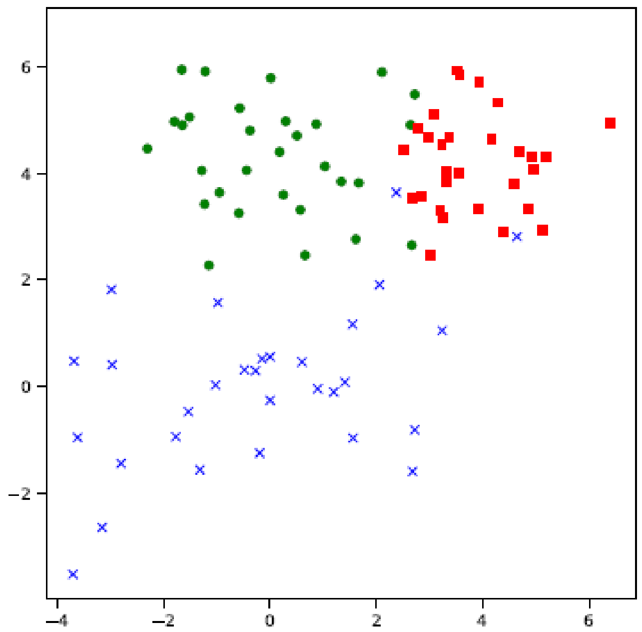
Class-conditional distribution of inputs from class y

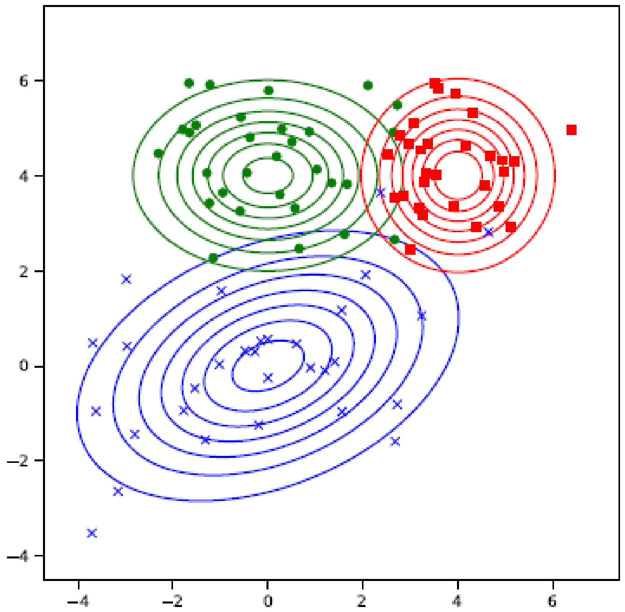
$$p(y|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, y)}{p(\boldsymbol{x})} = \frac{p(y)p(\boldsymbol{x}|y)}{p(\boldsymbol{x})} = \frac{p(y)p(\boldsymbol{x}|y)}{\sum_y p(y)p(\boldsymbol{x}|y)}$$

- Note that the inputs of each class have a specific distribution $p(\boldsymbol{x}|y)$
  - Thus the inputs are also assumed to be "generated" via a process defined by $p(\boldsymbol{x}|y)$
- We learn $p(y)$ and $p(\boldsymbol{x}|y)$ using training data $(\boldsymbol{X}, \boldsymbol{y}) = \{(x_n, y_n)\}_{n=1}^N$
  - To estimate $p(y = k|\theta)$ or $p(y|\theta)$, the data is $\{y_n\}_{n=1}^N$: Bernoulli ($K = 2$) or multinoulli ($K > 2$) To estimate $p(\boldsymbol{x}|y = k, \theta)$, the data is all the inputs from class k, i.e., $\boldsymbol{X}_{\text{k}} = \{\boldsymbol{x}_n : y_n = k\}$
- These distributions can be estimated using point estimation or using fully Bayesian inference

Generative Supervised Learning: Classification

- Training: Fit an appropriate probability distribution for data from each class





- Test time: Evaluate under which class, the test input has largest probability $p(x|y=k)$ or largest class posterior probability $p(y=k|x) \propto p(y=k)p(x|y=k)$

Generative Classification: A Generative Story

- Assuming binary labels, can define a "generative story" for each example $(\boldsymbol{x}_i, y_i)$
  - First draw ("generate") a binary label $y_i \in \{0,1\}$

$$y_i \sim \text{Bernoulli}(\pi)$$

For multi-class problems, we will have a multinoulli instead

  - Now draw ("generate") the input $\boldsymbol{x}_i$ from the distribution of class $y_i \in \{0,1\}$

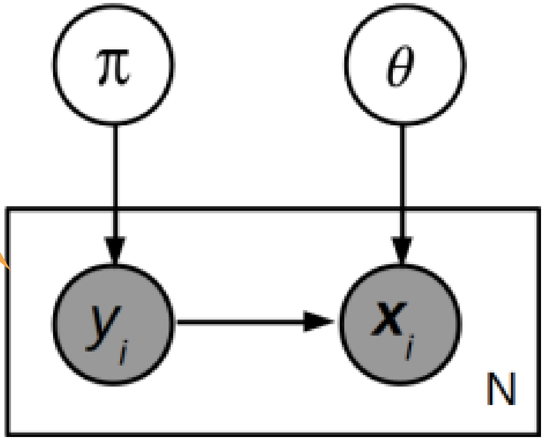
$$\boldsymbol{x}_i | y_i \sim p(\boldsymbol{x} | \theta_{y_i})$$

Most generative models (supervised as well as unsupervised/semi-supervised) can be expressed via such a story

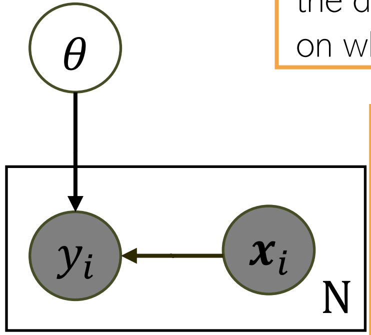
- Writing $\boldsymbol{\theta} = (\theta_0, \theta_1)$, the above generative model shown in "plate notation"

Order of generation in this story depends on what part of the data/parameters depend on what data/params

Note that in this generative process, we assume y is generated first since the generation of $\boldsymbol{x}$ depends on what $\boldsymbol{y}$ is



Compare this with the plate notation diagram of a discriminative model such as prob linear regression or logistic regression

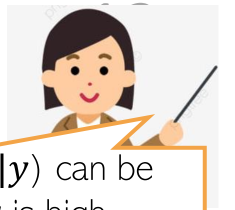


A discriminative model (no model for $\boldsymbol{x}_i$'s)

Often we also show the generation of parameters/unknowns as well (via their respective distributions)

Note that in this generative process, we assume y is generated first since the generation of $\boldsymbol{x}$ depends on what $\boldsymbol{y}$ is

Generative Classification: Learning Procedure



Note: Estimating $p(\boldsymbol{x}|y)$ can be difficult especially if $\boldsymbol{x}$ is high-dimensional and we don't have enough data from each class

- Recall the generative classification model

Prior probability of belonging to class k

Probability (density) of input $\boldsymbol{x}$ under class k

(Posterior) Probability of belonging to class k, conditioned on the input $\boldsymbol{x}$

$$p(y = k|\boldsymbol{x}) = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{\sum_k p(y = k)p(\boldsymbol{x}|y = k)}$$

A way to handle this is to assume simpler forms for $p(\boldsymbol{x}|y)$ (e.g., Gaussian with diagonal/spherical covar – naïve Bayes) but it might sacrifice accuracy too

- We need to learn $\boldsymbol{p(y)}$ and $\boldsymbol{p(x|y)}$ here given training data $(\boldsymbol{X}, \boldsymbol{y}) = \{(x_n, y_n)\}_{n=1}^N$

- Class prior distribution $\boldsymbol{p(y)}$ will always be a discrete distribution, e.g.,
  - For $y \in \{0,1\}$, $p(y) = p(y|\pi) = \text{Bernoulli}(y|\pi)$ with $\boldsymbol{\pi} \in (0,1)$
  - For $y \in \{1,2,\ldots,K\}$, $p(y) = p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\boldsymbol{\pi})$ where $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$

$\sum_{k=1}^{K} \pi_k = 1$

- Class conditional distribution $\boldsymbol{p(x|y)}$ will depend on the nature of inputs, e.g.,
  - For $\boldsymbol{x} \in \mathbb{R}^D$, $p(\boldsymbol{x}|y = k)$ can be a multivariate Gaussian (one per class)

For π, can use Beta or Dirichlet (we have already seen these examples)

$$p(\boldsymbol{x}|y = k) = p(\boldsymbol{x}|\theta_k) = \mathcal{N}(\boldsymbol{x}|\mu_k, \Sigma_k)$$

Note: When estimating θ_k, we only need inputs from class k
$\boldsymbol{X}_k = \{\boldsymbol{x}_n : y_n = k\}$

Will need appropriate prior distributions for π and $\{\theta_k\}_{k=1}^K$

- Can estimate $\boldsymbol{\pi}$ and $\{\theta_k\}_{k=1}^K$ using $(\boldsymbol{X}, \boldsymbol{y})$ via point est. or fully Bayesian infer.

Generative Classification: Making Predictions

- Once $\boldsymbol{\pi}$ and $\{\theta_k\}_{k=1}^K$ are learned, we are ready to make prediction for any test input $\boldsymbol{x}_*$
- Two ways to make the prediction
- Approach 1: If we have point estimates for $\boldsymbol{\pi}$ and $\{\theta_k\}_{k=1}^K$, say $\hat{\boldsymbol{\pi}}$ and $\{\hat{\theta}_k\}_{k=1}^K$. Then

$$p(y_* = k|\boldsymbol{x}_*) = \frac{p(y_* = k|\hat{\pi})p(\boldsymbol{x}_*|\hat{\theta}_k)}{\sum_k p(y = k|\hat{\pi})p(\boldsymbol{x}|\hat{\theta}_k)} \propto \hat{\pi}_k p(\boldsymbol{x}_*|\hat{\theta}_k)$$

Compute for every value of k and normalize

- Approach 2: If we have the full posterior for $\boldsymbol{\pi}$ and $\{\theta_k\}_{k=1}^K$. Then
  - Instead of using $p(y_* = k|\hat{\boldsymbol{\pi}})$, we will use $p(y_* = k|\boldsymbol{y}) = \int p(y_* = k|\pi)p(\pi|\boldsymbol{y})d\pi$ PPD of y_*
  - Instead of using $p(\boldsymbol{x}_*|\hat{\theta}_k)$, we will use $p(\boldsymbol{x}_*|\boldsymbol{X}_k) = \int p(\boldsymbol{x}_*|\theta_k)p(\theta_k|\boldsymbol{X}_k)d\theta_k$ PPD of $\boldsymbol{x}_*$
  - Using these quantities, the prediction will be made as

Compute for every value of k and normalize

$$p(y_* = k|x_*, \boldsymbol{X}, \boldsymbol{y}) = \frac{p(y_* = k|\boldsymbol{y})p(\boldsymbol{x}_*|\boldsymbol{X}_k)}{\sum_k p(y_* = k|\boldsymbol{y})p(\boldsymbol{x}_*|\boldsymbol{X}_k)} \propto p(y_* = k|\boldsymbol{y})p(\boldsymbol{x}_*|\boldsymbol{X}_k)$$

Note that we aren't using a single "best" value of the params π and θ_k unlike Approach 1

Generative Sup. Learning: Some Comments

- A very flexible approach for classification

Incorporate info about how frequent each class is in the training data ("class prior")

Incorporate info about the shape of each class

Consequently, can naturally learn nonlinear boundaries, too (without using kernel methods or deep learning)

$$p(y_* = k|\boldsymbol{x}_*) = \frac{p(y_* = k)p(\boldsymbol{x}_*|y_* = k)}{\sum_k p(y_* = k)p(\boldsymbol{x}_*|y_* = k)}$$

- Can handle missing labels and missing features
  - These can be treated as latent variables as estimated using methods such as EM

Will discuss this later

- Ability to handle missing labels makes it suitable for semi-supervised learning
- The choice of the class-conditional and proper estimation is important
  - Can leverage advances in deep generative models to learn very flexible forms for $p(\boldsymbol{x}|y)$
- Can also use it for regression (define $p(\boldsymbol{x}, y)$ via some distr. and obtain $p(y|\boldsymbol{x})$ from it)
- Can also combine generative and discriminative approaches for supervised learning

Hybrids of Discriminative and Generative Models

- Both discriminative and generative models have their strengths/shortcomings
- Some aspects about discriminative models for sup. learning
  - Discriminative models have usually fewer parameters (e.g., just a weight vector)
  - Given "plenty" of training data, disc. models can usually outperform generative models

Recall prob linear regression and logistic reg

- Some aspects about generative models for sup. learning
  - Can be more flexible (we have seen the reasons already)
  - Usually have more parameters to be learned
  - Modeling the inputs (learning $\boldsymbol{p(x|y)}$) can be difficult for high-dim inputs
- Some prior work on combining discriminative and generative models. Examples:

$$\alpha \log p(y|x;\theta) + \beta \log p(x;\theta)$$

$$p(x, y, \theta_d, \theta_g) = p_{\theta_d}(y|x) p_{\theta_g}(x) p(\theta_d, \theta_g)$$

$$p(x, y, z) = p(y|x, z) \cdot p(x, z)$$

Approach 1 (McCullum et al, 2006) – modeling the joint $p(x, y|\theta)$ using a multi-conditional likelihood

Approach 2 (Lasserre et al, 2006) – Coupled parameters between discriminative and generative models

Approach 3 (Kuleshov and Ermon, 2017) – Coupling discriminative and generative models via a latent variable z (see "Deep Hybrid Models: Bridging Discriminative and Generative Approaches", UAI 2017)