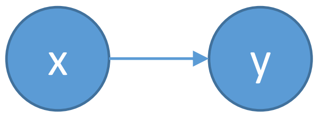
Conditional Models

Conditional Models

- Two node models - p($y|x$)



- Supervised learning
  - Regression (y is real)
  - Classification (y is discrete)

- y depends on x

Estimating Conditional Models

- Conditional models can be estimated using one of the following two ways
  1. Estimate the joint distribution $p(\boldsymbol{x}, y)$ and then use Bayes rule to get $p(y|\boldsymbol{x})$
  $$p(y|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, y|\theta)}{p(\boldsymbol{x}|\theta)}$$
  2. Estimate the conditional $p(y|\boldsymbol{x})$ directly (used when we don't care about modeling $\boldsymbol{x}$), e.g.
  $$p(y|\boldsymbol{x}) = \mathcal{N}(y|f_\mu(\boldsymbol{x}), f_{\sigma^2}(\boldsymbol{x})) \qquad \text{(params of } p(y|\boldsymbol{x}) \text{ will be functions of } \boldsymbol{x}\text{)}$$
- Approach 1 is called generative approach, approach 2 is called discriminative approach

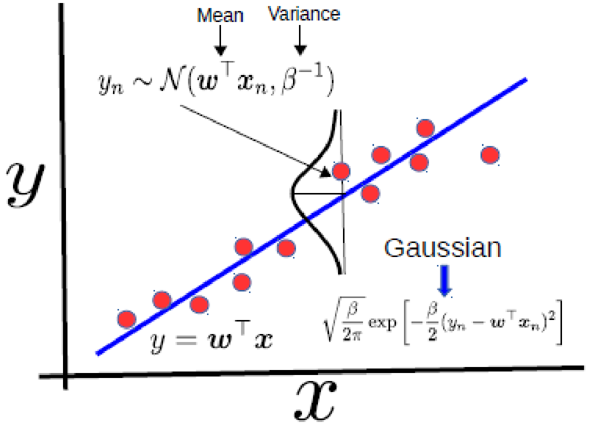
Linear Regression

- Given: N training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, features: $\boldsymbol{x}_n \in \mathbb{R}^D$, response $y_n \in \mathbb{R}$
- Assume a "noisy" linear model with regression weight vector $\boldsymbol{w} = [w_1, w_2, \ldots, w_D] \in \mathbb{R}^D$

$$y_n = \boldsymbol{w}^\top \boldsymbol{x}_n + \epsilon_n$$

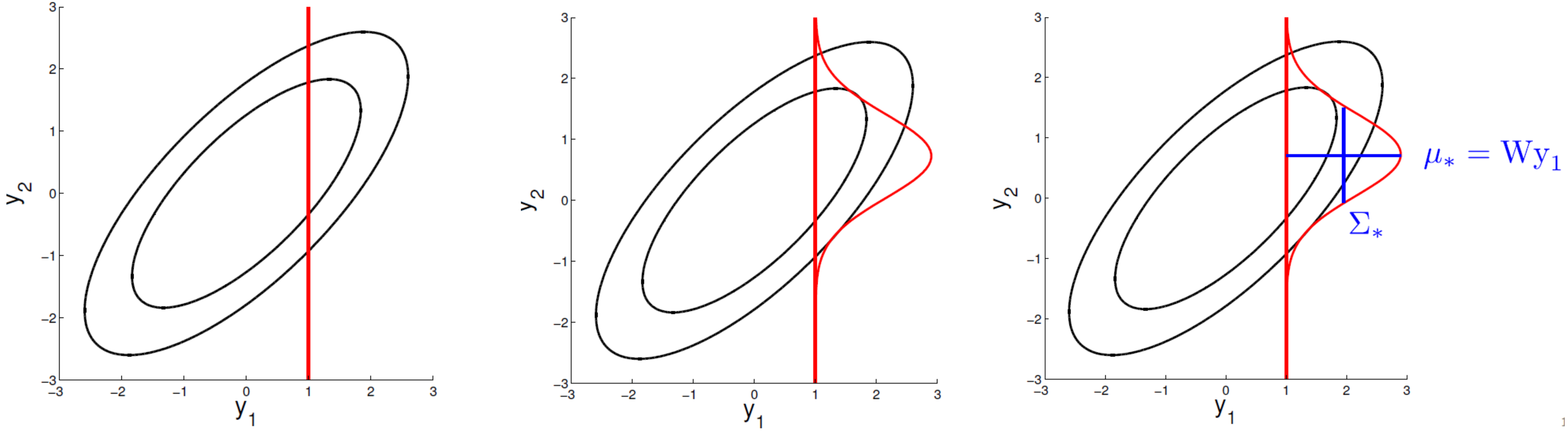
where $\epsilon_n \sim \mathcal{N}(0, \beta^{-1})$, β: precision (inverse variance) of Gaussian (assumed known)

- Therefore $p(y_n|\boldsymbol{x}_n, \boldsymbol{w}, \beta) = \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1})$

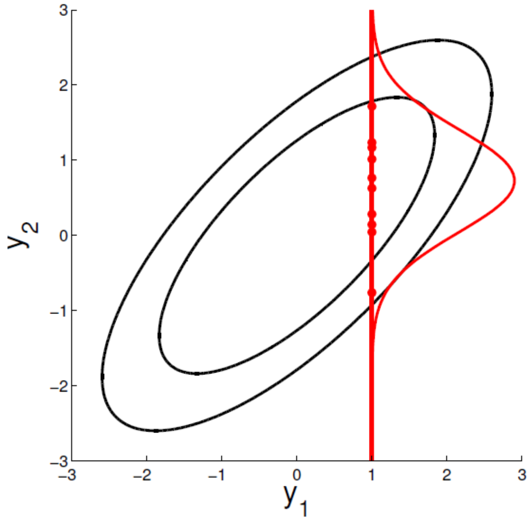


Conditional Distributions

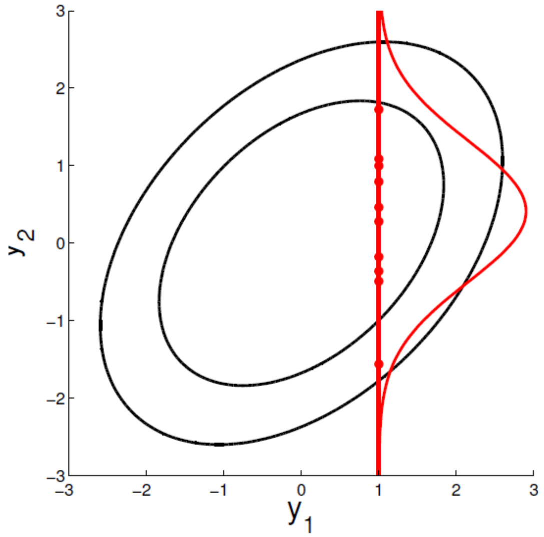
$$p(\mathrm{y}_2|\mathrm{y}_1, \Sigma) \propto \exp\left(-\tfrac{1}{2}(\mathrm{y}_2 - \mu_*)\Sigma_*{}^{-1}(\mathrm{y}_2 - \mu_*)\right)$$



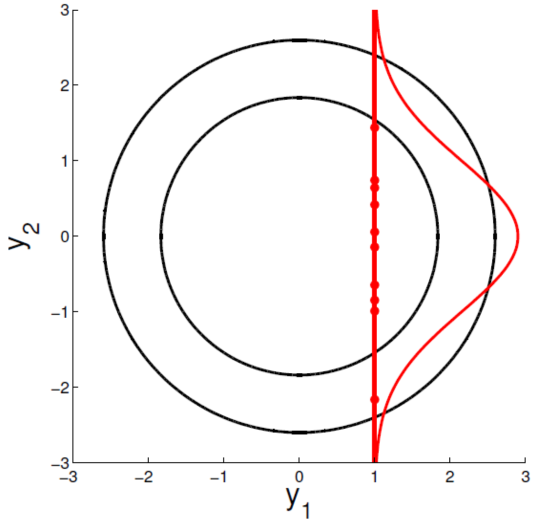
Conditional Distributions



$$\Sigma = \begin{bmatrix} 1 & .7 \\ .7 & 1 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 1 & .4 \\ .4 & 1 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Maximum Likelihood Estimate

- Notation: $\mathbf{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_N]^\top$: $N \times D$ feature matrix, $\boldsymbol{y} = [y_1 \dots y_N]^\top$: $N \times 1$ response vector
- Assuming independent observations, the likelihood model

$$
\begin{aligned}
p(\boldsymbol{y}|\boldsymbol{w}, \mathbf{X}, \beta) = \prod_{n=1}^{N} p(y_n|\boldsymbol{w}, \boldsymbol{x}_n, \beta) &= \prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) \\
&= \prod_{n=1}^{N} \sqrt{\frac{\beta}{2\pi}} \exp\left[-\frac{\beta}{2}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2\right] \\
&= \left(\frac{\beta}{2\pi}\right)^{\frac{N}{2}} \exp\left[-\frac{\beta}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2\right]
\end{aligned}
$$

can also write the likelihood $p(\boldsymbol{y}|\boldsymbol{w}, \mathbf{X})$ as an N-dim multivariate Gaussian

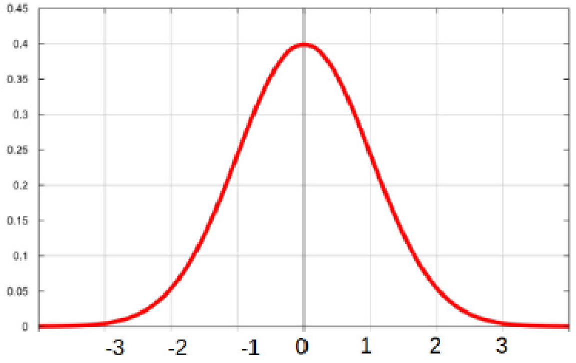
$$
p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}, \beta) = \mathcal{N}(\boldsymbol{y}|\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N) = \left(\frac{\beta}{2\pi}\right)^{\frac{N}{2}} \exp\left[-\frac{\beta}{2}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})\right]
$$

Prior

- Assume the entries in $\boldsymbol{w}$ are i.i.d. with zero mean Gaussian priors. Therefore

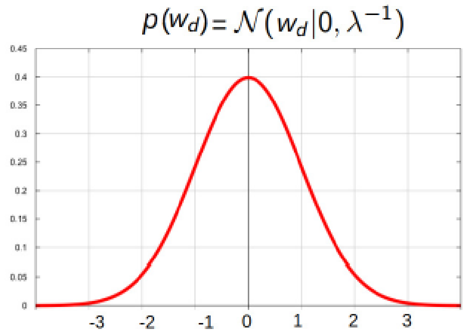
$$p(\boldsymbol{w}) = \prod_{d=1}^{D} p(w_d) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \mathcal{N}(\boldsymbol{w}|\mathbf{0}, \lambda^{-1}\mathbf{I}_D) = \left(\frac{\lambda}{2\pi}\right)^{\frac{D}{2}} \exp\left[-\frac{\lambda}{2}\boldsymbol{w}^\top \boldsymbol{w}\right]$$

$p(w_d) = \mathcal{N}(w_d|0, \lambda^{-1})$



- This prior promotes the entries in $\boldsymbol{w}$ to be small (close to zero)

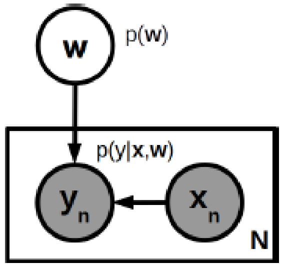
Prior Hyperparameters



- The role of the precision hyperparam λ in the prior is important
- Large values of λ would more aggressively encourage w_d to be close to zero
- Can think of λ as the regularization hyperparam for the weights
- Can even have different λ for each w_d, i.e., $p(\boldsymbol{w}|\{\lambda_d\}_{d=1}^{D}) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda_d^{-1})$

Sparse regression

Bayesian Linear Regression



(Hyperparameters λ,β not shown as they are fixed/known)

- Want to infer the posterior distribution over $\boldsymbol{w}$ (for now, assume β and λ to be known)

$$p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}, \beta, \lambda) = \frac{p(\boldsymbol{w}|\lambda)p(\boldsymbol{y}|\boldsymbol{w}, \mathbf{X}, \beta)}{p(\boldsymbol{y}|\mathbf{X}, \beta, \lambda)}$$

- Want to infer the posterior predictive distribution

$$p(y_*|\boldsymbol{x}_*, \mathbf{X}, \boldsymbol{y}, \beta, \lambda) = \int p(y_*|\boldsymbol{w}, \boldsymbol{x}_*, \beta)p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y}, \beta, \lambda)d\boldsymbol{w}$$

If Likelihood and Prior are assumed to be Gaussians, posterior is simple to compute

Posterior Distribution

- The posterior over $\boldsymbol{w}$ (for now, assume hyperparams β and λ to be known)

$$p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}, \beta, \lambda) = \frac{p(\boldsymbol{w}|\lambda)p(\boldsymbol{y}|\boldsymbol{w}, \mathbf{X}, \beta)}{p(\boldsymbol{y}|\mathbf{X}, \beta, \lambda)} \propto p(\boldsymbol{w}|\lambda)p(\boldsymbol{y}|\boldsymbol{w}, \mathbf{X}, \beta)$$

- Computing $p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y}, \beta, \lambda)$

$$p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}, \beta, \lambda) \propto \mathcal{N}(\boldsymbol{w}|\mathbf{0}, \lambda^{-1}\mathbf{I}_D) \times \mathcal{N}(\boldsymbol{y}|\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N)$$

- Using the "completing the squares" trick (or directly using Gaussian conditioning formula)

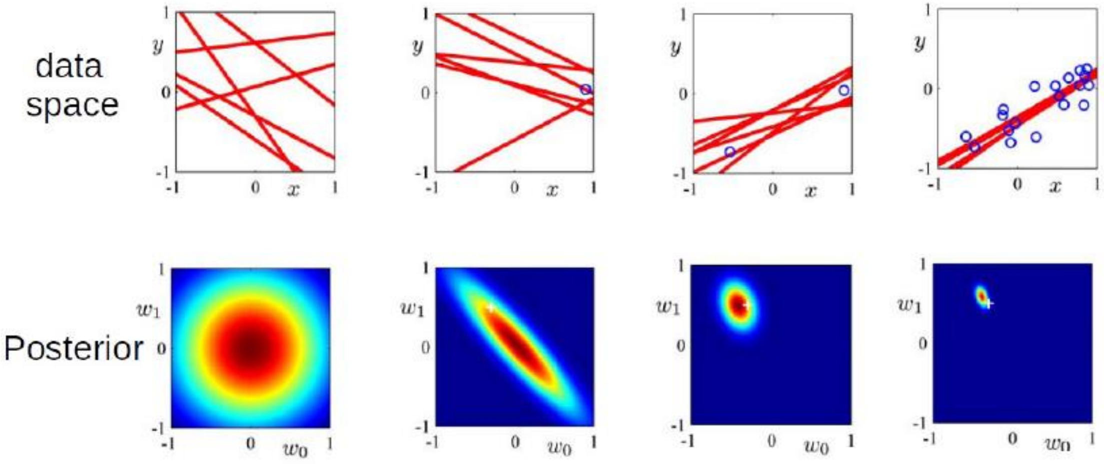
$$p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}, \beta, \lambda) = \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$

$$\text{where} \quad \boldsymbol{\Sigma}_N = (\beta \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top + \lambda \mathbf{I}_D)^{-1} = (\beta \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \quad \text{(posterior's covariance matrix)}$$

$$\boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N \left[\beta \sum_{n=1}^{N} y_n \boldsymbol{x}_n\right] = \boldsymbol{\Sigma}_N \left[\beta \mathbf{X}^\top \boldsymbol{y}\right] = (\mathbf{X}^\top \mathbf{X} + \frac{\lambda}{\beta}\mathbf{I}_D)^{-1}\mathbf{X}^\top \boldsymbol{y}$$

Visualizing the Posterior

- Assume a linear regression problem with ground truth $\boldsymbol{w} = [w_0, w_1]$ with $w_0 = -0.3, w_1 = 0.5$
- Assume data generated by a linear regression model $y = w_0 + w_1 x +$ "noise"
  - Note: It's actually 1-D regression (w_0 is just a bias term), or 2-D reg. with feature $[1, x]$
- Figures below show the "data space" and posterior of $\boldsymbol{w}$ for different number of observations (note: with no observations, the posterior = prior)



Posterior Predictive Distribution

- Given the posterior $p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}, \beta, \lambda) = \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$, how to make prediction y_* for a new input $\boldsymbol{x}_*$?
- The posterior predictive distribution will be

$$p(y_*|\boldsymbol{x}_*, \mathbf{X}, \boldsymbol{y}, \beta, \lambda) = \int p(y_*|\boldsymbol{x}_*, \boldsymbol{w}, \beta) p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y}, \beta, \lambda) d\boldsymbol{w}$$

- Using Gaussian predictive/marginal formula, the posterior predictive will be another Gaussian

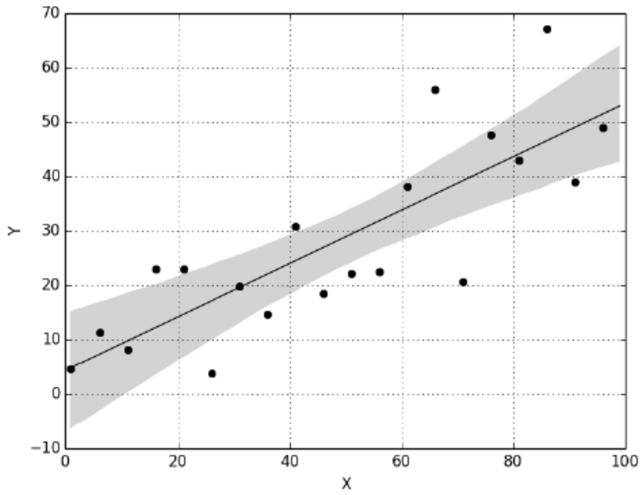
$$\boxed{p(y_*|\boldsymbol{x}_*, \mathbf{X}, \boldsymbol{y}, \beta, \lambda) = \mathcal{N}(\boldsymbol{\mu}_N^\top \boldsymbol{x}_*, \beta^{-1} + \boldsymbol{x}_*^\top \boldsymbol{\Sigma}_N \boldsymbol{x}_*)}$$

- So we get a predictive mean $\boldsymbol{\mu}_N^\top \boldsymbol{x}_*$ and an input-specific predictive variance $\beta^{-1} + \boldsymbol{x}_*^\top \boldsymbol{\Sigma}_N \boldsymbol{x}_*$
- In contrast, MLE and MAP make "plug-in" predictions (using the point estimate of $\boldsymbol{w}$)

$$\begin{aligned} p(y_*|\boldsymbol{x}_*, \boldsymbol{w}_{MLE}) &= \mathcal{N}(\boldsymbol{w}_{MLE}^\top \boldsymbol{x}_*, \beta^{-1}) && \text{- MLE prediction} \\ p(y_*|\boldsymbol{x}_*, \boldsymbol{w}_{MAP}) &= \mathcal{N}(\boldsymbol{w}_{MAP}^\top \boldsymbol{x}_*, \beta^{-1}) && \text{- MAP prediction} \end{aligned}$$

Visualizing PPD

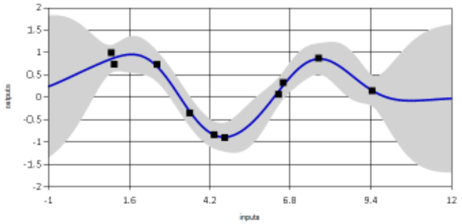
Black dots are training examples



Width of the shaded region at any x denotes the predictive uncertainty at that x ($+/-$ one std-dev)

Regions with more training examples have smaller predictive variance

Nonlinear Regression



- Can extend the linear regression model to handle nonlinear regression problems
- One way is to replace the feature vectors $\boldsymbol{x}$ by a nonlinear mapping $\phi(\boldsymbol{x})$

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}^\top \phi(\boldsymbol{x}), \beta^{-1})$$

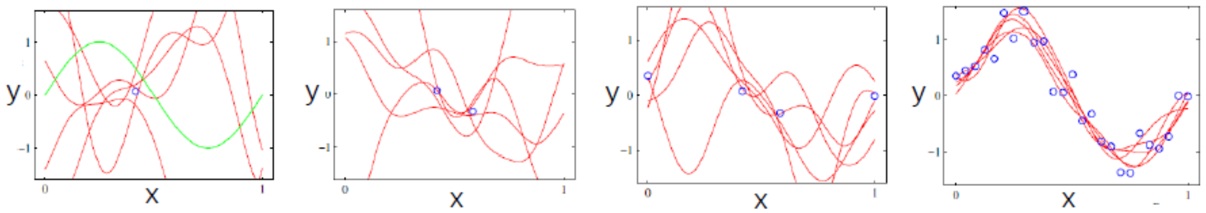
- The nonlinear mapping can be defined directly, e.g., for a one-dimensional feature x

$$\phi(x) = [1, x, x^2]$$

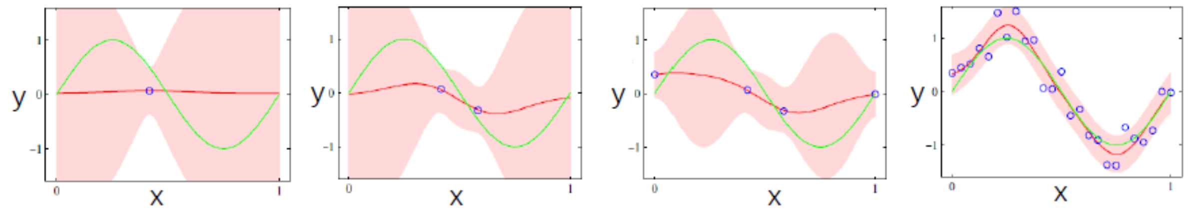
- Alternatively, a kernel function can be used to implicitly define the nonlinear mapping

Visualizations

- We can similarly visualize a Bayesian nonlinear regression model
- Figures below: Green curve is the true function and blue circles are observations (x_n, y_n)
- Posterior of the nonlinear regression model: Some curves drawn from the posterior

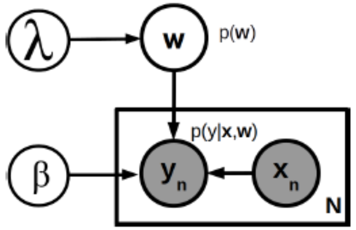


- Posterior predictive: Red curve is predictive mean, shaded region denotes predictive uncertainty



Learning Hyperparameters

- Can treat hyperparams as just a bunch of additional unknowns
- Can be learned using a suitable inference algorithm (point estimation or fully Bayesian)
- Example: For the linear regression model, the full set of parameters would be $(\boldsymbol{w}, \lambda, \beta)$



- Can assume priors on all these parameters and infer their "joint" posterior distribution

$$p(\boldsymbol{w}, \beta, \lambda|\mathbf{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}, \beta, \lambda)p(\boldsymbol{w}, \lambda, \beta)}{p(\boldsymbol{y}|\mathbf{X})} = \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}, \beta, \lambda)p(\boldsymbol{w}|\lambda)p(\beta)p(\lambda)}{\int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}, \beta)p(\boldsymbol{w}|\lambda)p(\beta)p(\lambda)\, d\boldsymbol{w}\, d\lambda d\beta}$$

- Infering the above is usually intractable (rare to have conjugacy). Requires approximations.

MLE on Hyperparameters

- One popular way to estimate hyperparameters is by maximizing the marginal likelihood
- For our linear regression model, this quantity (a function of the hyperparams) will be

$$p(\boldsymbol{y}|\mathbf{X}, \beta, \lambda) = \int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}, \beta)p(\boldsymbol{w}|\lambda)d\boldsymbol{w}$$

- The "optimal" hyperparameters in this case can be then found by

$$\hat{\beta}, \hat{\lambda} = \arg\max_{\beta,\lambda} \log p(\boldsymbol{y}|\mathbf{X}, \beta, \lambda)$$

- This is called MLE-II or (log) evidence maximization

Sparse Regression

- Many probabilistic models consist of weights that are given zero-mean Gaussian priors, e.g.,

$$\mu(\boldsymbol{x}) = \sum_{d=1}^{D} w_d x_d \qquad \text{(mean of a prob. lin reg model)}$$

$$\mu(\boldsymbol{x}) = \sum_{n=1}^{N} w_n k(\boldsymbol{x}_n, \boldsymbol{x}) \qquad \text{(mean of a prob. kernel based nonlin reg model)}$$

- A zero-mean prior is of the form $p(w_d) = \mathcal{N}(0, \lambda^{-1})$ or $p(w_d) = \mathcal{N}(0, \lambda_d^{-1})$
- Precision λ or λ_d specifies our belief about how close to zero w_d is (like regularization hyperparam)
- However, such a prior usually gives small weights but not very strong sparsity
- Putting a gamma prior on precision can give sparsity (will soon see why)

  - Sparsity of weights will be a very useful thing to have in many models, e.g.,
    - For linear model, this helps learn relevance of each feature x_d

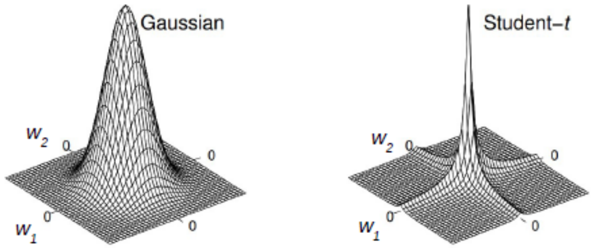
Hierarchical Priors

- Consider linear regression with prior $p(w_d|\lambda_d) = \mathcal{N}(0, \lambda_d^{-1})$ on each weight
- Let's treat precision λ_d as unknown and use a gamma (shape $= a$, rate $= b$) prior on it

$$p(\lambda_d) = \text{Gamma}(a, b) = \frac{b^a}{\Gamma(a)} \lambda_d^{a-1} \exp(-b\lambda_d)$$

- Marginalizing the precision leads to a Student-t prior on each w_d

$$p(w_d) = \int p(w_d|\lambda_d)p(\lambda_d)d\lambda_d = \frac{b^a\Gamma(a+1/2)}{\sqrt{2\pi}\Gamma(a)}(b + w_d^2/2)^{-(a+1/2)}$$



Bayesian Logistic Regression

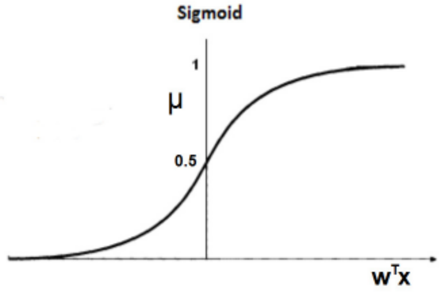
- The goal is to learn $p(y|\boldsymbol{x})$. Here $p(y|\boldsymbol{x})$ will be a discrete distribution (e.g., Bernoulli, multinoulli)
- Usually two approaches to learn $p(y|\boldsymbol{x})$: <u>Discriminative</u> Classification and <u>Generative</u> Classification
- Discriminative Classification: Model and learn $p(y|\boldsymbol{x})$ <u>directly</u>
  - This approach does not model the distribution of the inputs $\boldsymbol{x}$
- Generative Classification: Model and learn $p(y|\boldsymbol{x})$ "indirectly" as $p(y|\boldsymbol{x}) = \frac{p(y)p(\boldsymbol{x}|y)}{p(\boldsymbol{x})}$
  - Called generative because, via $p(\boldsymbol{x}|y)$, we model how the inputs $\boldsymbol{x}$ of each class are generated
  - The approach requires first learning class-marginal $p(y)$ and class-conditional distributions $p(\boldsymbol{x}|y)$
  - Usually harder to learn than discriminative but also has some advantages (more on this later)

Classification by Logistic Regression

- **Logistic Regression** (LR) is an example of discriminative binary classification, i.e., $y \in \{0, 1\}$
- Logistic Regression models $\boldsymbol{x}$ to y relationship using the sigmoid function

$$p(y = 1|\boldsymbol{x}, \boldsymbol{w}) = \mu = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x})} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x})}$$

  where $\boldsymbol{w} \in \mathbb{R}^D$ is the weight vector. Also note that $p(y = 0|\boldsymbol{x}, \boldsymbol{w}) = 1 - \mu$



- A large positive (negative) "score" $\boldsymbol{w}^\top \boldsymbol{x}$ means large probability of label being 1 (0)

Classification Rules

- The LR classification rule is

$$\begin{aligned} p(y=1|\boldsymbol{x},\boldsymbol{w}) &= \mu = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x})} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})} \\ p(y=0|\boldsymbol{x},\boldsymbol{w}) &= 1-\mu = 1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})} \end{aligned}$$

- This implies a Bernoulli likelihood model for the labels

$$p(y|\boldsymbol{x},\boldsymbol{w}) = \text{Bernoulli}(\sigma(\boldsymbol{w}^\top \boldsymbol{x})) = \left[\frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})}\right]^y \left[\frac{1}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})}\right]^{(1-y)}$$

- Given N observations $(\mathbf{X}, \boldsymbol{y}) = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$, we can do point estimation for $\boldsymbol{w}$ by maximizing the log-likelihood (or minimizing the negative log-likelihood). This is basically MLE.

$$\boldsymbol{w}_{MLE} = \arg\max_{\boldsymbol{w}} \sum_{n=1}^N \log p(y_n|\boldsymbol{x}_n,\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} -\sum_{n=1}^N \log p(y_n|\boldsymbol{x}_n,\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} NLL(\boldsymbol{w})$$

Bayesian Logistic Regression

- Recall that the likelihood model is Bernoulli

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \text{Bernoulli}(\sigma(\boldsymbol{w}^\top \boldsymbol{x})) = \left[\frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x})}\right]^y \left[\frac{1}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x})}\right]^{(1-y)}$$

- Just like the Bayesian linear regression case, let's use a Gausian prior on $\boldsymbol{w}$

$$p(\boldsymbol{w}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I}_D) \propto \exp(-\frac{\lambda}{2}\boldsymbol{w}^\top \boldsymbol{w})$$

- Given N observations $(\mathbf{X}, \boldsymbol{y}) = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$, where $\mathbf{X}$ is $N \times D$ and $\boldsymbol{y}$ is $N \times 1$, the posterior over $\boldsymbol{w}$

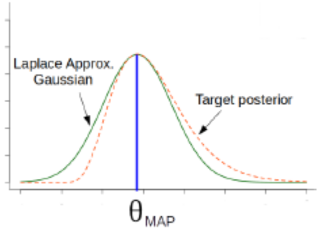
$$p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}} = \frac{\prod_{n=1}^N p(y_n|\boldsymbol{x}_n, \boldsymbol{w})p(\boldsymbol{w})}{\int \prod_{n=1}^N p(y_n|\boldsymbol{x}_n, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}}$$

- The denominator is intractable in general (logistic-Bernoulli and Gaussian are not conjugate)

Laplace Approximation of Posterior Distrib.

- Approximate the posterior distribution $p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D},\theta)}{p(\mathcal{D})}$ by the following Gaussian

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_{MAP}, \mathbf{H}^{-1})$$



- Note: θ_{MAP} is the maximum-a-posteriori (MAP) estimate of θ, i.e.,

$$\theta_{MAP} = \arg\max_{\theta} p(\theta|\mathcal{D}) = \arg\max_{\theta} p(\mathcal{D},\theta) = \arg\max_{\theta} p(\mathcal{D}|\theta)p(\theta) = \arg\max_{\theta}[\log p(\mathcal{D}|\theta) + \log p(\theta)]$$

- Usually θ_{MAP} can be easily solved for (e.g., using first/second order iterative methods)
- $\mathbf{H}$ is the Hessian matrix of the negative log-posterior (or negative log-joint-prob) at θ_{MAP}

$$\mathbf{H} = -\nabla^2 \log p(\theta|\mathcal{D})\big|_{\theta=\theta_{MAP}} = -\nabla^2 \log p(\mathcal{D},\theta)\big|_{\theta=\theta_{MAP}}$$

Derivation of Laplace Aprroximation

- Let's write the Bayes rule as

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D},\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D},\theta)}{\int p(\mathcal{D},\theta)d\theta} = \frac{e^{\log p(\mathcal{D},\theta)}}{\int e^{\log p(\mathcal{D},\theta)}d\theta}$$

- Suppose $\log p(\mathcal{D},\theta) = f(\theta)$. Let's approximate $f(\theta)$ using its 2nd order Taylor expansion

$$f(\theta) \approx f(\theta_0) + (\theta - \theta_0)^\top \nabla f(\theta_0) + \frac{1}{2}(\theta - \theta_0)^\top \nabla^2 f(\theta_0)(\theta - \theta_0)$$

where θ_0 is some arbitrarily chosen point in the domain of f

- Let's choose $\theta_0 = \theta_{MAP}$. Note that $\nabla f(\theta_{MAP}) = \nabla \log p(\mathcal{D},\theta_{MAP}) = 0$. Therefore

$$\log p(\mathcal{D},\theta) \approx \log p(\mathcal{D},\theta_{MAP}) + \frac{1}{2}(\theta - \theta_{MAP})^\top \nabla^2 \log p(\mathcal{D},\theta_{MAP})(\theta - \theta_{MAP})$$

Contd..

- Plugging in this 2nd order Taylor approximation for $\log p(\mathcal{D},\theta)$, we have

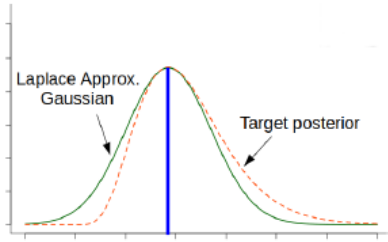
$$p(\theta|\mathcal{D}) = \frac{e^{\log p(\mathcal{D},\theta)}}{\int e^{\log p(\mathcal{D},\theta)}d\theta} \approx \frac{e^{\log p(\mathcal{D},\theta_{MAP})+\frac{1}{2}(\theta-\theta_{MAP})^\top \nabla^2 \log p(\mathcal{D},\theta_{MAP})(\theta-\theta_{MAP})}}{\int e^{\log p(\mathcal{D},\theta_{MAP})+\frac{1}{2}(\theta-\theta_{MAP})^\top \nabla^2 \log p(\mathcal{D},\theta_{MAP})(\theta-\theta_{MAP})}d\theta}$$

- Further simplifying, we have

$$p(\theta|\mathcal{D}) \approx \frac{e^{-\frac{1}{2}(\theta-\theta_{MAP})^\top \{-\nabla^2 \log p(\mathcal{D},\theta_{MAP})\}(\theta-\theta_{MAP})}}{\int e^{-\frac{1}{2}(\theta-\theta_{MAP})^\top \{-\nabla^2 \log p(\mathcal{D},\theta_{MAP})\}(\theta-\theta_{MAP})}d\theta}$$

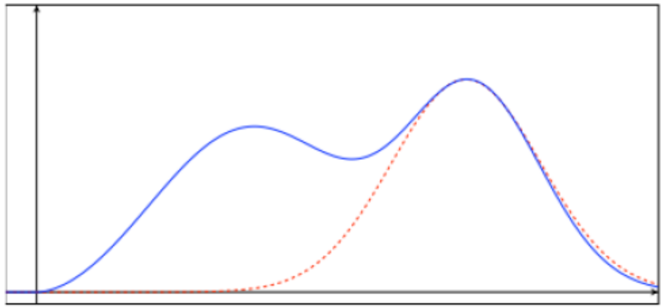
- Therefore the Laplace approximation of the posterior $p(\theta|\mathcal{D})$ is a Gaussian and is given by

$$\boxed{p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta|\theta_{MAP},\mathbf{H}^{-1})} \qquad \text{where} \quad \mathbf{H} = -\nabla^2 \log p(\mathcal{D},\theta_{MAP})$$



Properties of Laplace Aprroximation

- Usually straightforward if derivatives (first and second) can be computed easily
- Expensive if the number of parameters is very large (due to Hessian computation and inversion)
- Can do badly if the (true) posterior is multimodal



- Can actually apply it when working with any regularized loss function (not just probabilistic models) to get a Gaussian posterior distribution over the parameters
  - negative log-likelihood (NLL) = loss function, negative log-prior = regularizer

Laplace Approximation for Logistic Regression

- Data $\mathcal{D} = (\mathbf{X}, \boldsymbol{y})$ and parameter $\theta = \boldsymbol{w}$. The Laplace approximation of posterior will be

$$p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y}) \approx \mathcal{N}(\boldsymbol{w}_{MAP}, \mathbf{H}^{-1})$$

- The required quantities are defined as

$$\begin{aligned} \boldsymbol{w}_{MAP} &= \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}) = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{y}, \boldsymbol{w}|\mathbf{X}) = \arg\min_{\boldsymbol{w}} [-\log p(\boldsymbol{y}, \boldsymbol{w}|\mathbf{X})] \\ \mathbf{H} &= \nabla^2[-\log p(\boldsymbol{y}, \boldsymbol{w}|\mathbf{X})]\Big|_{\boldsymbol{w}=\boldsymbol{w}_{MAP}} \end{aligned}$$

- We can compute $\boldsymbol{w}_{MAP}$ using iterative methods (gradient descent):
  - First-order (gradient) methods: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}_t$. Requires gradient $\boldsymbol{g}$ of $-\log p(\boldsymbol{y}, \boldsymbol{w}|\mathbf{X})$

$$\boldsymbol{g} = \nabla[-\log p(\boldsymbol{y}, \boldsymbol{w}|\mathbf{X})]$$

  - Second-order methods. $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \mathbf{H}_t^{-1}\boldsymbol{g}_t$. Requires both gradient and Hessian (defined above)

PPD for Logistic Regression

- When using MLE, the predictive distribution will be

$$\begin{aligned} p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{w}_{MLE}) &= \sigma(\boldsymbol{w}_{MLE}^\top \boldsymbol{x}_*) \\ p(y_*|\boldsymbol{x}_*, \boldsymbol{w}_{MLE}) &= \text{Bernoulli}(\sigma(\boldsymbol{w}_{MLE}^\top \boldsymbol{x}_*)) \end{aligned}$$

- When using MAP, the predictive distribution will be

$$\begin{aligned} p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{w}_{MAP}) &= \sigma(\boldsymbol{w}_{MAP}^\top \boldsymbol{x}_*) \\ p(y_*|\boldsymbol{x}_*, \boldsymbol{w}_{MAP}) &= \text{Bernoulli}(\sigma(\boldsymbol{w}_{MAP}^\top \boldsymbol{x}_*)) \end{aligned}$$

- When using Bayesian inference, the posterior predictive distribution, based on posterior averaging

$$p(y_* = 1|\boldsymbol{x}_*, \mathbf{X}, \boldsymbol{y}) = \int p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{w})p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})d\boldsymbol{w} = \int \sigma(\boldsymbol{w}^\top \boldsymbol{x}_*)p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})d\boldsymbol{w}$$

- Above is hard in general. :-(If using the Laplace approximation for $p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y})$, it will be

$$p(y_* = 1|\boldsymbol{x}_*, \mathbf{X}, \boldsymbol{y}) \approx \int \sigma(\boldsymbol{w}^\top \boldsymbol{x}_*)\mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_{MAP}, \mathbf{H}^{-1})d\boldsymbol{w}$$

- Its multiclass extension is softmax regression (which again can be treated in a Bayesian manner)

Bayesian Generative Classification

- Consider N labeled examples $\{(\boldsymbol{x}_i, y_i)\}_{n=1}^{N}$. Assume binary labels, i.e., $y_i \in \{0, 1\}$
- Goal: Classify a new example $\boldsymbol{x}$ by assigning a label $y \in \{0, 1\}$ to it
- We will assume a Generative Model for both labels y and and features $\boldsymbol{x}$
  - What it means: We will have (probabilistic) observation models for both y as well as $\boldsymbol{x}$
  - In contrast, in Bayesian linear regression model (and Bayesian logistic regression model), we didn't model $\boldsymbol{x}$ (there, we simply conditioned y on $\boldsymbol{x}$, treating $\boldsymbol{x}$ as "fixed")
  - When we don't model $\boldsymbol{x}$ and simply model y as a function of $\boldsymbol{x}$: Discriminative Model
- Generative classification models have many benefits. E.g.,
  - Can also utilize unlabeled examples (semi-supervised learning)
  - Can handle missing/corrupted features in $\boldsymbol{x}$
  - Can properly handle cases when features in $\boldsymbol{x}$ could be of mixed type (e.g., real, binary, count)

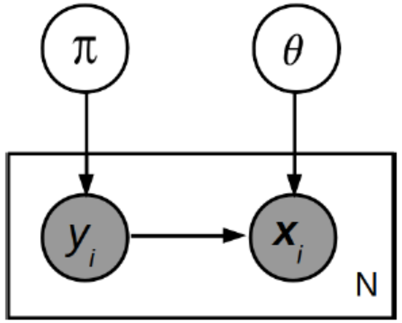
Generative Classification

- Basic idea: Each $\boldsymbol{x}_i$ is assumed generated conditioned on the value of corresponding label y_i
- The associated generative story is as follows
  - First draw ("generate") a binary label $y_i \in \{0, 1\}$

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

  - Now draw ("generate") the feature vector $\boldsymbol{x}$ from a distribution specific to the value y_i takes

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



Shaded is observed

Generative Classification

- Our generative model for classification is

$$y_i \sim \text{Bernoulli}(\pi), \qquad \boldsymbol{x}_i | y_i \sim p(\boldsymbol{x}|\theta_{y_i})$$

- Note: We have two distributions $p(\boldsymbol{x}|\theta_0)$ and $p(\boldsymbol{x}|\theta_1)$ for feature vector $\boldsymbol{x}$ (depending on its label)
- These distributions are also known as "class-conditional distributions"
- For now, we will not assume any specific form for the distriutions $p(\boldsymbol{x}|\theta_0)$ and $p(\boldsymbol{x}|\theta_1)$
  - Depends on nature of $\boldsymbol{x}$ (real-valued vectors? binary vectors? count vectors?)
- Model parameters to be learned here: $(\pi, \theta_0, \theta_1)$
- Note: Can extend to more than 2 classes (e.g., by replacing the Bernoulli on y by multinoulli)

Predicting Labels

- Note: The generative model only defines $p(y|\pi)$ and $p(\boldsymbol{x}|\theta_y)$. Doesn't define $p(y|\boldsymbol{x})$
- We combine these using Bayes rule to get $p(y|\boldsymbol{x})$

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

- Parameters of distributions $p(y|\pi)$ and $p(\boldsymbol{x}|\theta_y)$ are estimated from training data using point estimation methods (MLE or MAP) or using fully Bayesian inference (discussed today)
- Once these parameters π and θ_y are estimated (point estimates, or full posterior if doing Bayesian inference), the above Bayes rule can be applied to a new input $\hat{\boldsymbol{x}}$ to compute $p(\hat{y}|\hat{\boldsymbol{x}})$

Priors

- Let us focus on the supervised, binary classification setting for now
- In this case, we have three parameters to be learned: π, θ_0, and θ_1
  - Probability $\pi \in (0, 1)$ of the Bernoulli. Can assume the following Beta prior

$$\pi \sim \text{Beta}(a, b)$$

  - Parameters θ_0, and θ_1 of the class-conditional distributions. Will assume the same prior on both

$$\theta_0, \theta_1 \sim p(\theta)$$

  - We will jointly denote the prior on π, θ_0, and θ_1 as $p(\pi, \theta_0, \theta_1) = p(\pi)p(\theta_0)p(\theta_1)$

Likelihood

- Denote the $N \times D$ feature matrix by X and the $N \times 1$ label vector by $\boldsymbol{y}$
- Since both X and $\boldsymbol{y}$ are being modeled here, the likelihood function will be

$$
\begin{aligned}
p(X, \vec{y} | \pi, \theta_1, \theta_0) &= \prod_{i=1}^{N} p(x_i, y_i | \pi, \theta_1, \theta_0) \\
&= \prod_{i=1}^{N} p(x_i | y_i, \pi, \theta_1, \theta_0) p(y_i | \pi, \theta_1, \theta_0) \\
&= \prod_{i=1}^{N} p(x_i | \theta_{y_i}) p(y_i | \pi)
\end{aligned}
$$

Posterior

- We need to infer the following posterior distribution

$$p(\pi, \theta_1, \theta_0 | \vec{y}, X) = \frac{p(X, \vec{y} | \pi, \theta_1, \theta_0) p(\pi, \theta_1, \theta_0)}{\int_{\Omega_\theta} \int_{\Omega_\theta} \int_0^1 p(X, \vec{y} | \pi, \theta_1, \theta_0) p(\pi, \theta_1, \theta_0) d\pi d\theta_1 d\theta_0}$$

- Note: Ω_θ denotes the domain of θ

- Recall the prior $p(\pi, \theta_0, \theta_1) = p(\pi)p(\theta_0)p(\theta_1)$. The likelihood also factorized over data points, i.e.,

$$p(X, \mathbf{y} | \pi, \theta_1, \theta_0) = \prod_{i=1}^{N} p(x_i | \theta_{y_i}) p(y_i | \pi)$$

Posterior:

$$p(\pi, \theta_1, \theta_0 | \vec{y}, X) \propto \left[\prod_{i:y_i=1} p(x_i | \theta_1) p(\theta_1) \right] \left[\prod_{i:y_i=0} p(x_i | \theta_0) p(\theta_0) \right] \left[\prod_{i=1}^{N} p(y_i | \pi) p(\pi) \right]$$

Posterior

- Luckily, in this case, the same factorization structure simplies the denominator as well

$$p(\pi, \theta_1, \theta_0 | \vec{y}, X) = \frac{\prod_{i:y_i=1} p(x_i|\theta_1)p(\theta_1)}{\int \prod_{i:y_i=1} p(x_i|\theta_1)p(\theta_1)d\theta_1} \cdot \frac{\prod_{i:y_i=0} p(x_i|\theta_0)p(\theta_0)}{\int \prod_{i:y_i=0} p(x_i|\theta_0)p(\theta_0)d\theta_0} \cdot \frac{\prod_{i=1}^{N} p(y_i|\pi)p(\pi)}{\int \prod_{i=1}^{N} p(y_i|\pi)p(\pi)d\pi}$$

- The above is just a product of three posterior distributions !

$$p(\pi, \theta_1, \theta_0 | \vec{y}, X) = p(\theta_1 | \{x_i : y_i = 1\}) p(\theta_0 | \{x_i : y_i = 0\}) p(\pi | \vec{y})$$

- We also know what $p(\pi|\boldsymbol{y})$ will be (recall the coin-toss example)

$$p(\pi|\vec{y}) \propto \prod_{i=1}^{N} p(y_i|\pi)p(\pi) \quad \longrightarrow \quad p(\pi|\vec{y}) = \text{Beta}(a + \textstyle\sum_i y_i, b + N - \sum_i y_i)$$

- Form of posteriors on θ_1 and θ_2 will depend on $p(\boldsymbol{x}|\theta_1)$ and $p(\theta_1)$, and $p(\boldsymbol{x}|\theta_0)$ and $p(\theta_0)$, resp.

PPD

- Original goal is classification. We thus also want the predictive posterior for label of a new input, i.e., $p(\hat{y}|\hat{\mathbf{x}})$, for which the more "complete" notation in this Bayesian setting would be $p(\hat{y}|\hat{\mathbf{x}}, X, \mathbf{y})$

$$p(\hat{y}|\hat{x}, X, \vec{y}) = \int_{\Omega_\theta} \int_{\Omega_\theta} \int_0^1 p(\hat{y}|\hat{x}, \theta_1, \theta_0, \pi) p(\theta_1, \theta_0, \pi|X, \vec{y}) d\pi d\theta_1 d\theta_0$$

- Luckily, in this case, this too has a rather simple form. Using Bayes rule, we have

$$\begin{aligned} p(\hat{y}|\hat{x}, X, \vec{y}) &= \frac{p(\hat{x}|\hat{y}, X, \vec{y}) p(\hat{y}|X, \vec{y})}{p(\hat{x}|\hat{y}=1, X, \vec{y}) p(\hat{y}=1|X, \vec{y}) + p(\hat{x}|\hat{y}=0, X, \vec{y}) p(\hat{y}=0|X, \vec{y})} \\ &= \frac{p(\hat{x}|\hat{y}, X, \vec{y}) p(\hat{y}|\vec{y})}{p(\hat{x}|\hat{y}=1, X, \vec{y}) p(\hat{y}=1|\vec{y}) + p(\hat{x}|\hat{y}=0, X, \vec{y}) p(\hat{y}=0|\vec{y})} \end{aligned}$$

  - In order to compute this, we need $p(\hat{x}|\hat{y}, X, \mathbf{y})$ and $p(\hat{y}|\mathbf{y})$
    - $p(\hat{x}|\hat{y}, X, \mathbf{y})$: Marginal class-conditional distribution of the new input vector $\hat{x}$
    - $p(\hat{y}|\mathbf{y})$: Marginal probability of its label $\hat{y}$ given the labels of training data

Contd..

- Predictive posterior requires computing $p(\hat{x}|\hat{y}, X, \mathbf{y})$ and $p(\hat{y}|\mathbf{y})$
- The marginal likelihood $p(\hat{x}|\hat{y}, X, \mathbf{y})$ of $\hat{x}$ can be computed as

$$\begin{aligned} p(\hat{x}|\hat{y}, X, \vec{y}) &= \int_{\Omega_\theta}\int_{\Omega_\theta} p(\hat{x}|\hat{y}, \theta_1, \theta_0)p(\theta_1, \theta_0|X, \vec{y})d\theta_1 d\theta_0 \\ &= \int_{\Omega_\theta} p(\hat{x}|\theta_{\hat{y}})p(\theta_{\hat{y}}|\{x_i : y_i = \hat{y}\})d\theta_{\hat{y}} \end{aligned}$$

- The above is simply the posterior predictive distribution of class $\hat{y}$. The final expression will depend on the forms of $p(\hat{x}|\theta_{\hat{y}})$ and $p(\theta_{\hat{y}}|.)$. If exp-family, we will have closed form expression!

Naïve Bayes Classifier

- Usually the most critical choice in generative classification is that of class conditional $p(\boldsymbol{x}|\theta_y)$
- Very complex $p(\boldsymbol{x}|\theta_y)$ with lots of parameters may make estimation difficult
- Often however we can choose simple forms of $p(\boldsymbol{x}|\theta_y)$ to make estimation easier
- The naïve Bayes assumption: The conditional distribution $p(\boldsymbol{x}|\theta_y)$ factorizes over individual features (or over groups of features)
  - Suppose the features of $\hat{x}$ can be partitioned into v groups $\hat{x} = \{\hat{x}(j)\}_{j=1}^{v}$
  - Can also assume a similar partitioning for the parameters $\theta_{\hat{y}}$
  - This further simplifies calculation of marginal likelihood $p(\hat{x}|\hat{y}, X, \boldsymbol{y})$

$$p(\hat{x}|\hat{y}, X, \vec{y}) = \int_{\Omega_\theta} \prod_{j=1}^{v} p(\hat{x}(j)|\theta_{\hat{y}}(j))p(\theta_{\hat{y}}(j)|\{x_i(j) : y_i = \hat{y}\})d\theta_{\hat{y}}$$