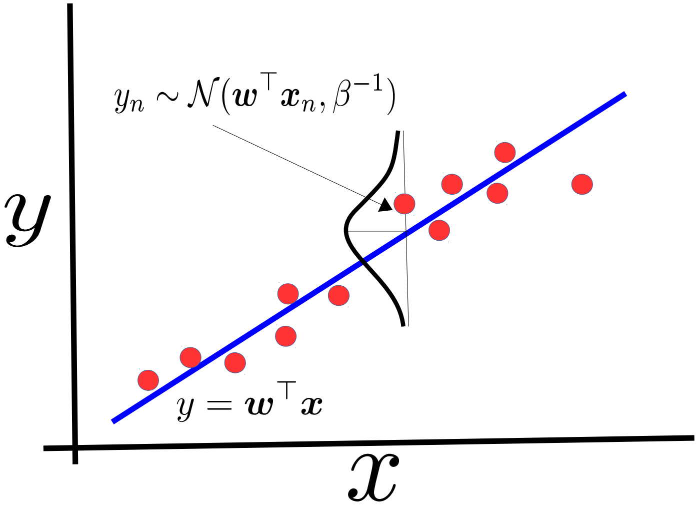
Probabilistic Models for Supervised Learning (Contd.)

Piyush Rai

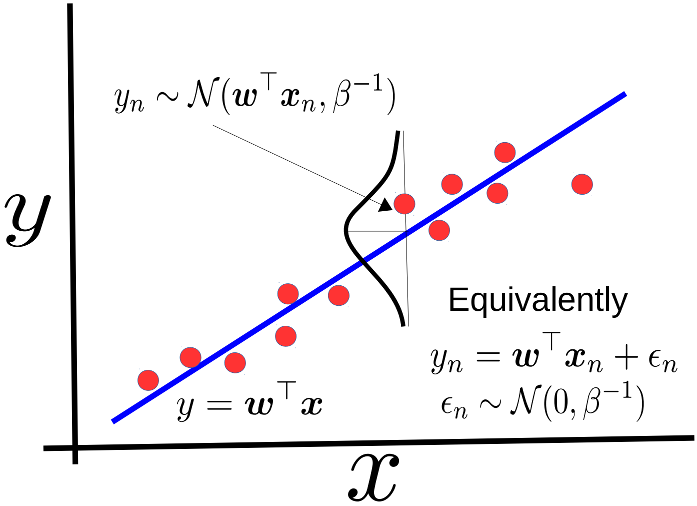
Introduction to Machine Learning (CS771A)

August 21, 2018

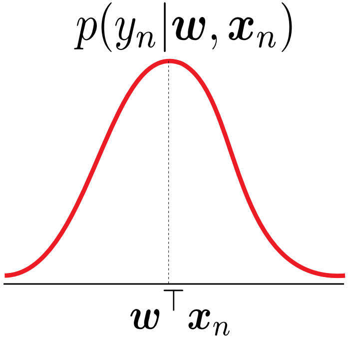
Recap: Probabilistic Linear Regression



Recap: Probabilistic Linear Regression



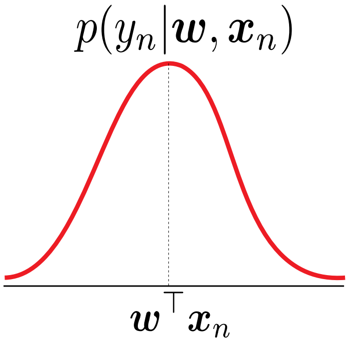
Recap: Probabilistic Linear Regression



The Likelihood

$$p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} \exp\left[-\frac{\beta}{2}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2\right]$$

Recap: Probabilistic Linear Regression

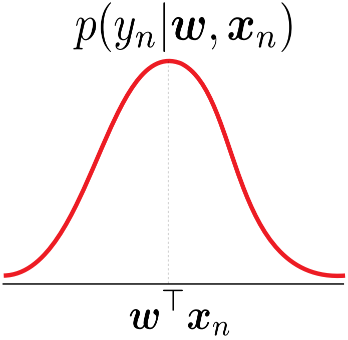


The Likelihood

$$p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} \exp\left[-\frac{\beta}{2}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2\right]$$

$$p(\boldsymbol{y}|\boldsymbol{w}, \mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) = \underbrace{\mathcal{N}(\boldsymbol{y}|\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N)}_{N\text{-dim pdf}}$$

Recap: Probabilistic Linear Regression



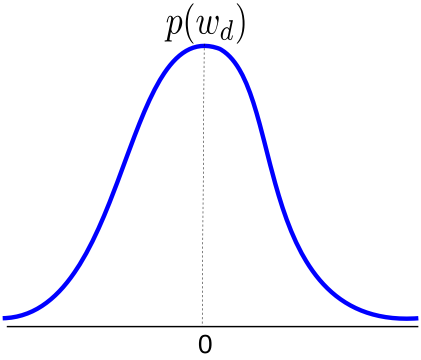
The Likelihood

$$p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} \exp\left[-\frac{\beta}{2}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2\right]$$

$$p(\boldsymbol{y}|\boldsymbol{w}, \mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1}) = \underbrace{\mathcal{N}(\boldsymbol{y}|\mathbf{X}\boldsymbol{w}, \beta^{-1}\mathbf{I}_N)}_{N\text{-dim pdf}}$$

$$\left(\frac{\beta}{2\pi}\right)^{N/2} \exp\left[-\frac{\beta}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2\right] = \frac{1}{\sqrt{\det(2\pi\beta^{-1}\mathbf{I}_N)}} \exp\left[-\frac{\beta}{2}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})\right]$$

Recap: Probabilistic Linear Regression

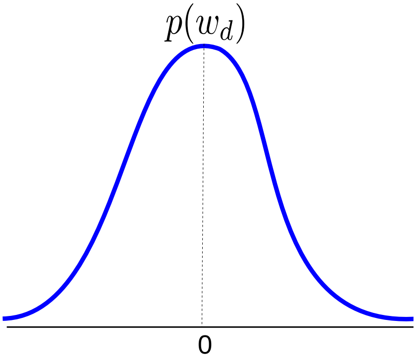


The Prior

$$p(w_d) = \mathcal{N}(w_d|0, \lambda^{-1}) = \sqrt{\frac{\lambda}{2\pi}} \exp\left[-\frac{\lambda}{2} w_d^2\right]$$

Zero-mean Gaussian prior encourages weights to be small. Precision λ controls how strong this prior is.

Recap: Probabilistic Linear Regression



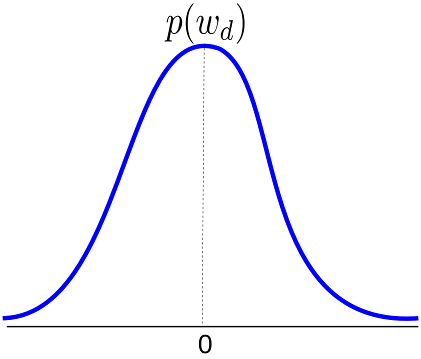
The Prior

$$p(w_d) = \mathcal{N}(w_d|0, \lambda^{-1}) = \sqrt{\frac{\lambda}{2\pi}} \exp\left[-\frac{\lambda}{2} w_d^2\right]$$

$$p(\boldsymbol{w}) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \underbrace{\mathcal{N}(\boldsymbol{w}|\mathbf{0}, \lambda^{-1}\mathbf{I}_D)}_{D\text{-dim pdf}}$$

Zero-mean Gaussian prior encourages weights to be small. Precision λ controls how strong this prior is.

Recap: Probabilistic Linear Regression



The Prior

$$p(w_d) = \mathcal{N}(w_d|0, \lambda^{-1}) = \sqrt{\frac{\lambda}{2\pi}} \exp\left[-\frac{\lambda}{2} w_d^2\right]$$

$$p(\boldsymbol{w}) = \prod_{d=1}^{D} \mathcal{N}(w_d|0, \lambda^{-1}) = \underbrace{\mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \lambda^{-1}\mathbf{I}_D)}_{D\text{-dim pdf}}$$

$$\left(\frac{\lambda}{2\pi}\right)^{D/2} \exp\left[-\frac{\lambda}{2}\sum_{d=1}^{D} w_d^2\right] = \frac{1}{\sqrt{\det(2\pi\lambda^{-1}\mathbf{I}_D)}} \exp\left[-\frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w}\right]$$

Zero-mean Gaussian prior encourages weights to be small. Precision λ controls how strong this prior is.

Recap: MLE, MAP, and Bayesian Inference for Prob. Lin. Reg.

- For MLE, we maximize the log-likelihood. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MLE} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w})$$

Recap: MLE, MAP, and Bayesian Inference for Prob. Lin. Reg.

- For MLE, we maximize the log-likelihood. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MLE} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 \right]$$

Recap: MLE, MAP, and Bayesian Inference for Prob. Lin. Reg.

- For MLE, we maximize the log-likelihood. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MLE} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 \right]$$

- For MAP, we maximize the log-posterior. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MAP} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X})$$

Recap: MLE, MAP, and Bayesian Inference for Prob. Lin. Reg.

- For MLE, we maximize the log-likelihood. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MLE} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 \right]$$

- For MAP, we maximize the log-posterior. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MAP} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{w} \right]$$

Recap: MLE, MAP, and Bayesian Inference for Prob. Lin. Reg.

- For MLE, we maximize the log-likelihood. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MLE} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 \right]$$

- For MAP, we maximize the log-posterior. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MAP} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{w} \right]$$

- For Bayesian inference, we compute the full posterior. Easily computable (thanks to conjugacy)

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

Recap: MLE, MAP, and Bayesian Inference for Prob. Lin. Reg.

- For MLE, we maximize the log-likelihood. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MLE} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 \right]$$

- For MAP, we maximize the log-posterior. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MAP} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\lambda}{2} \boldsymbol{w}^\top \boldsymbol{w} \right]$$

- For Bayesian inference, we compute the full posterior. Easily computable (thanks to conjugacy)

$$\begin{aligned} p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}) &= \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N) \\ \boldsymbol{\Sigma}_N &= (\beta \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \end{aligned}$$

Recap: MLE, MAP, and Bayesian Inference for Prob. Lin. Reg.

- For MLE, we maximize the log-likelihood. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MLE} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2\right]$$

- For MAP, we maximize the log-posterior. Ignoring constants w.r.t. $\boldsymbol{w}$, we have

$$\hat{\boldsymbol{w}}_{MAP} = \arg\max_{\boldsymbol{w}} \log p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}) = \arg\min_{\boldsymbol{w}} \left[\frac{\beta}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \frac{\lambda}{2}\boldsymbol{w}^\top \boldsymbol{w}\right]$$

- For Bayesian inference, we compute the full posterior. Easily computable (thanks to conjugacy)

$$\begin{aligned} p(\boldsymbol{w}|\boldsymbol{y}, \mathbf{X}) &= \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N) \\ \boldsymbol{\Sigma}_N &= (\beta \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_D)^{-1} \\ \boldsymbol{\mu}_N &= (\mathbf{X}^\top \mathbf{X} + \frac{\lambda}{\beta}\mathbf{I}_D)^{-1}\mathbf{X}^\top \boldsymbol{y} \end{aligned}$$

Recap: Predictive Distribution for Prob. Lin. Reg.

- When using MLE/MAP estimate of $\boldsymbol{w}$, we compute the "plug-in" predictive distribution

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

Recap: Predictive Distribution for Prob. Lin. Reg.

- When using MLE/MAP estimate of $\boldsymbol{w}$, we compute the "plug-in" predictive distribution

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

- For MLE approach, mean of predicted output is $\boldsymbol{w}_{MLE}^\top \boldsymbol{x}_*$, variance is β^{-1}

Recap: Predictive Distribution for Prob. Lin. Reg.

- When using MLE/MAP estimate of $\boldsymbol{w}$, we compute the "plug-in" predictive distribution

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

- For MLE approach, mean of predicted output is $\boldsymbol{w}_{MLE}^\top \boldsymbol{x}_*$, variance is β^{-1}
- For MAP approach, mean of predicted output is $\boldsymbol{w}_{MAP}^\top \boldsymbol{x}_*$, variance is β^{-1}

Recap: Predictive Distribution for Prob. Lin. Reg.

- When using MLE/MAP estimate of $\boldsymbol{w}$, we compute the "plug-in" predictive distribution

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

- For MLE approach, mean of predicted output is $\boldsymbol{w}_{MLE}^\top \boldsymbol{x}_*$, variance is β^{-1}
- For MAP approach, mean of predicted output is $\boldsymbol{w}_{MAP}^\top \boldsymbol{x}_*$, variance is β^{-1}
- When using the fully posterior, we can compute the posterior predictive distribution

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

Recap: Predictive Distribution for Prob. Lin. Reg.

- When using MLE/MAP estimate of $\boldsymbol{w}$, we compute the "plug-in" predictive distribution

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

- For MLE approach, mean of predicted output is $\boldsymbol{w}_{MLE}^\top \boldsymbol{x}_*$, variance is β^{-1}
- For MAP approach, mean of predicted output is $\boldsymbol{w}_{MAP}^\top \boldsymbol{x}_*$, variance is β^{-1}
- When using the fully posterior, we can compute the posterior predictive distribution

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

- For Bayesian approach, mean of predicted output is $\boldsymbol{w}_N^\top \boldsymbol{x}_*$, variance is $\beta^{-1} + \boldsymbol{x}_*^\top \boldsymbol{\Sigma}_N \boldsymbol{x}_*$

Recap: Predictive Distribution for Prob. Lin. Reg.

- When using MLE/MAP estimate of $\boldsymbol{w}$, we compute the "plug-in" predictive distribution

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

- For MLE approach, mean of predicted output is $\boldsymbol{w}_{MLE}^\top \boldsymbol{x}_*$, variance is β^{-1}
- For MAP approach, mean of predicted output is $\boldsymbol{w}_{MAP}^\top \boldsymbol{x}_*$, variance is β^{-1}
- When using the fully posterior, we can compute the posterior predictive distribution

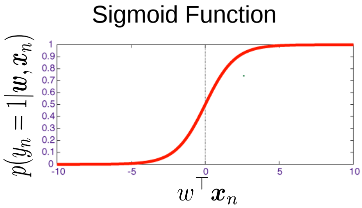
$$
p(y_*|\boldsymbol{x}_*, \mathbf{X}, \boldsymbol{y}) = \int p(y_*|\boldsymbol{x}_*, \boldsymbol{w}) p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y}) d\boldsymbol{w} = \mathcal{N}(\boldsymbol{\mu}_N^\top \boldsymbol{x}_*, \beta^{-1} + \boldsymbol{x}_*^\top \boldsymbol{\Sigma}_N \boldsymbol{x}_*)
$$

- For Bayesian approach, mean of predicted output is $\boldsymbol{w}_N^\top \boldsymbol{x}_*$, variance is $\beta^{-1} + \boldsymbol{x}_*^\top \boldsymbol{\Sigma}_N \boldsymbol{x}_*$(note the different variance for each test input, unlike MLE/MAP prediction)

Recap: Logistic Regression

- Logistic Regression models $p(y_n = 1|\boldsymbol{w}, \boldsymbol{x}_n)$ using the sigmoid function

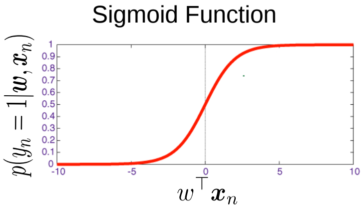
$$p(y_n = 1|\boldsymbol{w}, \boldsymbol{x}_n) = \mu_n = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x}_n)} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$$



Recap: Logistic Regression

- Logistic Regression models $p(y_n = 1|\boldsymbol{w}, \boldsymbol{x}_n)$ using the sigmoid function

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

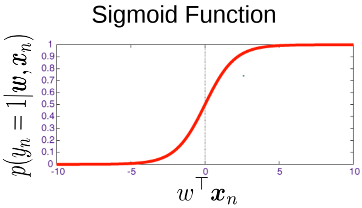


- Thus each likelihood $p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \text{Bernoulli}(y_n|\mu_n) = \mu_n^{y_n}(1 - \mu_n)^{1-y_n}$

Recap: Logistic Regression

- Logistic Regression models $p(y_n = 1|\boldsymbol{w}, \boldsymbol{x}_n)$ using the sigmoid function

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



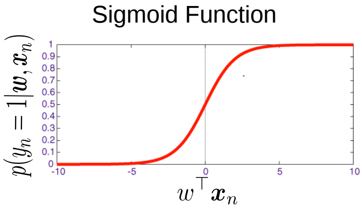
- Thus each likelihood $p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \text{Bernoulli}(y_n|\mu_n) = \mu_n^{y_n}(1-\mu_n)^{1-y_n}$
- Assuming i.i.d. labels, likelihood is product of Bernoullis

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

Recap: Logistic Regression

- Logistic Regression models $p(y_n = 1|\boldsymbol{w}, \boldsymbol{x}_n)$ using the sigmoid function

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



- Thus each likelihood $p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \text{Bernoulli}(y_n|\mu_n) = \mu_n^{y_n}(1 - \mu_n)^{1-y_n}$

- Assuming i.i.d. labels, likelihood is product of Bernoullis

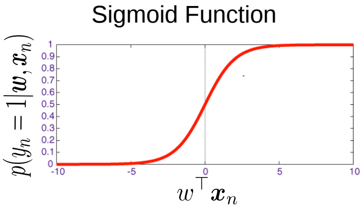
$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \prod_{n=1}^{N} \mu_n^{y_n}(1 - \mu_n)^{1-y_n}$$

- Can also use a Gaussian prior $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\mathbf{0}, \lambda^{-1}\mathbf{I}_D)$ just like in probabilistic linear regression

Recap: Logistic Regression

- Logistic Regression models $p(y_n = 1|\boldsymbol{w}, \boldsymbol{x}_n)$ using the sigmoid function

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



- Thus each likelihood $p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \text{Bernoulli}(y_n|\mu_n) = \mu_n^{y_n}(1 - \mu_n)^{1-y_n}$
- Assuming i.i.d. labels, likelihood is product of Bernoullis

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

- Can also use a Gaussian prior $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\mathbf{0}, \lambda^{-1}\mathbf{I}_D)$ just like in probabilistic linear regression
- Can estimate $\boldsymbol{w}$ via MLE, MAP, or (a somewhat hard to do) fully Bayesian inference

Recap: Logistic Regression

- Logistic regression can be extended to more than 2 classes

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^K \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk}$$

Recap: Logistic Regression

- Logistic regression can be extended to more than 2 classes

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^{K} \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk} \quad \text{and} \quad \sum_{\ell=1}^{K} \mu_{n\ell} = 1$$

Recap: Logistic Regression

- Logistic regression can be extended to more than 2 classes

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^K \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk} \quad \text{and} \quad \sum_{\ell=1}^K \mu_{n\ell} = 1$$

- MLE/MAP for logistic/softmax does not have closed form solution (unlike linear regression case)

Recap: Logistic Regression

- Logistic regression can be extended to more than 2 classes

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^K \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk} \quad \text{and} \quad \sum_{\ell=1}^K \mu_{n\ell} = 1$$

- MLE/MAP for logistic/softmax does not have closed form solution (unlike linear regression case)
- Computing full posterior is intractable (since Bernoulli/multinoulli and Gaussian are not conjugate)

Recap: Logistic Regression

- Logistic regression can be extended to more than 2 classes

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^{K} \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk} \quad \text{and} \quad \sum_{\ell=1}^{K} \mu_{n\ell} = 1$$

- MLE/MAP for logistic/softmax does not have closed form solution (unlike linear regression case)
- Computing full posterior is intractable (since Bernoulli/multinoulli and Gaussian are not conjugate)
  - Laplace (Gaussian) approximation is one way to get an approximate posterior

Recap: Logistic Regression

- Logistic regression can be extended to more than 2 classes

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^{K} \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk} \quad \text{and} \quad \sum_{\ell=1}^{K} \mu_{n\ell} = 1$$

- MLE/MAP for logistic/softmax does not have closed form solution (unlike linear regression case)
- Computing full posterior is intractable (since Bernoulli/multinoulli and Gaussian are not conjugate)
  - Laplace (Gaussian) approximation is one way to get an approximate posterior
- Predictive distribution is straightforward when using MLE/MAP

Recap: Logistic Regression

- Logistic regression can be extended to more than 2 classes

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^K \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk} \quad \text{and} \quad \sum_{\ell=1}^K \mu_{n\ell} = 1$$

- MLE/MAP for logistic/softmax does not have closed form solution (unlike linear regression case)
- Computing full posterior is intractable (since Bernoulli/multinoulli and Gaussian are not conjugate)
  - Laplace (Gaussian) approximation is one way to get an approximate posterior
- Predictive distribution is straightforward when using MLE/MAP
- Predictive distribution is intractable when using full posterior

Generative Models for Supervised Learning

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

Here, we will model both inputs and outputs!

Generative Classification

- Consider a classification problem with $K \geq 2$ classes

Generative Classification

- Consider a classification problem with $K \geq 2$ classes
- Assuming θ to collectively denote all the params, the generative classification model is

$$p(y = k|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, y = k|\theta)}{p(\boldsymbol{x}|\theta)}, \quad k = 1, \ldots, K$$

Generative Classification

- Consider a classification problem with $K \geq 2$ classes
- Assuming θ to collectively denote all the params, the generative classification model is

$$p(y = k|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, y = k|\theta)}{p(\boldsymbol{x}|\theta)}, \quad k = 1, \ldots, K$$

- Note that the denominator $p(\boldsymbol{x}|\theta) = \sum_{k=1}^{K} p(\boldsymbol{x}, y = k|\theta)$, using sum rule of probability

Generative Classification

- Consider a classification problem with $K \geq 2$ classes
- Assuming θ to collectively denote all the params, the generative classification model is

$$p(y = k|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, y = k|\theta)}{p(\boldsymbol{x}|\theta)}, \quad k = 1, \ldots, K$$

- Note that the denominator $p(\boldsymbol{x}|\theta) = \sum_{k=1}^{K} p(\boldsymbol{x}, y = k|\theta)$, using sum rule of probability
- Can use the chain rule to re-express the above as

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\color{red}{p(y = k|\theta)}\color{blue}{p(\boldsymbol{x}|y = k, \theta)}}{p(\boldsymbol{x}|\theta)}$$

Generative Classification

- Consider a classification problem with $K \geq 2$ classes
- Assuming θ to collectively denote all the params, the generative classification model is

$$p(y = k|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, y = k|\theta)}{p(\boldsymbol{x}|\theta)}, \quad k = 1, \ldots, K$$

- Note that the denominator $p(\boldsymbol{x}|\theta) = \sum_{k=1}^{K} p(\boldsymbol{x}, y = k|\theta)$, using sum rule of probability
- Can use the chain rule to re-express the above as

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\color{red}{p(y = k|\theta)}\color{blue}{p(\boldsymbol{x}|y = k, \theta)}}{p(\boldsymbol{x}|\theta)}$$

- This depends on two quantities

Generative Classification

- Consider a classification problem with $K \geq 2$ classes
- Assuming θ to collectively denote all the params, the generative classification model is

$$p(y = k|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, y = k|\theta)}{p(\boldsymbol{x}|\theta)}, \quad k = 1, \ldots, K$$

- Note that the denominator $p(\boldsymbol{x}|\theta) = \sum_{k=1}^{K} p(\boldsymbol{x}, y = k|\theta)$, using sum rule of probability
- Can use the chain rule to re-express the above as

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

- This depends on two quantities
  - $p(y = k|\theta)$: The class-marginal distribution (also called "class prior")

Generative Classification

- Consider a classification problem with $K \geq 2$ classes
- Assuming θ to collectively denote all the params, the generative classification model is

$$p(y = k|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, y = k|\theta)}{p(\boldsymbol{x}|\theta)}, \quad k = 1, \ldots, K$$

- Note that the denominator $p(\boldsymbol{x}|\theta) = \sum_{k=1}^{K} p(\boldsymbol{x}, y = k|\theta)$, using sum rule of probability
- Can use the chain rule to re-express the above as

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

- This depends on two quantities
  - $p(y = k|\theta)$: The class-marginal distribution (also called "class prior")
  - $p(\boldsymbol{x}|y = k, \theta)$: The class-conditional distribution of the inputs

Generative Classification

- Consider a classification problem with $K \geq 2$ classes
- Assuming θ to collectively denote all the params, the generative classification model is

$$p(y = k|\boldsymbol{x}, \theta) = \frac{p(\boldsymbol{x}, y = k|\theta)}{p(\boldsymbol{x}|\theta)}, \quad k = 1, \ldots, K$$

- Note that the denominator $p(\boldsymbol{x}|\theta) = \sum_{k=1}^{K} p(\boldsymbol{x}, y = k|\theta)$, using sum rule of probability
- Can use the chain rule to re-express the above as

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

- This depends on two quantities
  - $p(y = k|\theta)$: The class-marginal distribution (also called "class prior")
  - $p(\boldsymbol{x}|y = k, \theta)$: The class-conditional distribution of the inputs
- Generative classification requires first estimating the parameters θ of these two distributions

Generative Classification: Estimating Class-Marginal Distribution

- Estimating the class-marginal is usually straightforward in generative classification

Generative Classification: Estimating Class-Marginal Distribution

- Estimating the class-marginal is usually straightforward in generative classification
- The class marginal distribution is (has to be!) a discrete distribution (multinoulli)

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

Generative Classification: Estimating Class-Marginal Distribution

- Estimating the class-marginal is usually straightforward in generative classification
- The class marginal distribution is (has to be!) a discrete distribution (multinoulli)

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

where multinoilli parameters $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$, $\sum_{k=1}^{K} \pi_k = 1$

Generative Classification: Estimating Class-Marginal Distribution

- Estimating the class-marginal is usually straightforward in generative classification
- The class marginal distribution is (has to be!) a discrete distribution (multinoulli)

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

where multinoilli parameters $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$, $\sum_{k=1}^{K} \pi_k = 1$, and $\pi_k = p(y = k)$

Generative Classification: Estimating Class-Marginal Distribution

- Estimating the class-marginal is usually straightforward in generative classification
- The class marginal distribution is (has to be!) a discrete distribution (multinoulli)

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

where multinoilli parameters $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$, $\sum_{k=1}^{K} \pi_k = 1$, and $\pi_k = p(y = k)$

- Given N labeled training examples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, MLE for $\boldsymbol{\pi}$ (won't depend on $\boldsymbol{x}_n$'s) will be

$$\boldsymbol{\pi}_{MLE} = \arg\max_{\boldsymbol{\pi}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\pi})$$

Generative Classification: Estimating Class-Marginal Distribution

- Estimating the class-marginal is usually straightforward in generative classification
- The class marginal distribution is (has to be!) a discrete distribution (multinoulli)

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

where multinoilli parameters $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$, $\sum_{k=1}^{K} \pi_k = 1$, and $\pi_k = p(y = k)$

- Given N labeled training examples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, MLE for $\boldsymbol{\pi}$ (won't depend on $\boldsymbol{x}_n$'s) will be

$$\boldsymbol{\pi}_{MLE} = \arg\max_{\boldsymbol{\pi}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\pi})$$

.. which gives $\pi_k = N_k/N$ (exercise: verify) where $N_k = \sum_{n=1}^{N} \mathbb{I}[y_n = k]$

Generative Classification: Estimating Class-Marginal Distribution

- Estimating the class-marginal is usually straightforward in generative classification
- The class marginal distribution is (has to be!) a discrete distribution (multinoulli)

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

where multinoilli parameters $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$, $\sum_{k=1}^{K} \pi_k = 1$, and $\pi_k = p(y = k)$

- Given N labeled training examples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, MLE for $\boldsymbol{\pi}$ (won't depend on $\boldsymbol{x}_n$'s) will be

$$\boldsymbol{\pi}_{MLE} = \arg\max_{\boldsymbol{\pi}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\pi})$$

.. which gives $\pi_k = N_k/N$ (exercise: verify) where $N_k = \sum_{n=1}^{N} \mathbb{I}[y_n = k]$

- Note: If MAP (or full posterior) is needed, we can use a Dirichlet prior distribution on $\boldsymbol{\pi}$

Generative Classification: Estimating Class-Marginal Distribution

- Estimating the class-marginal is usually straightforward in generative classification
- The class marginal distribution is (has to be!) a discrete distribution (multinoulli)

$$p(y|\boldsymbol{\pi}) = \text{multinoulli}(y|\pi_1, \ldots, \pi_K) = \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y=k]}$$

where multinoilli parameters $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_K]$, $\sum_{k=1}^{K} \pi_k = 1$, and $\pi_k = p(y = k)$

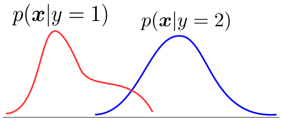
- Given N labeled training examples $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, MLE for $\boldsymbol{\pi}$ (won't depend on $\boldsymbol{x}_n$'s) will be

$$\boldsymbol{\pi}_{MLE} = \arg\max_{\boldsymbol{\pi}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{\pi})$$

.. which gives $\pi_k = N_k/N$ (exercise: verify) where $N_k = \sum_{n=1}^{N} \mathbb{I}[y_n = k]$

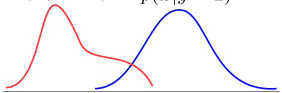
- Note: If MAP (or full posterior) is needed, we can use a Dirichlet prior distribution on $\boldsymbol{\pi}$
  - Another exercise: Try to derive the MAP estimate of $\boldsymbol{\pi}$ and also the full posterior (good news: multinoulli and Dirichlet are conjugate to each other, so full posterior is easy)

Generative Classification: Estimating Class-Conditional Distr.



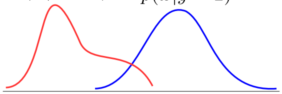
- We usually assume an appropriate class-conditional $p(\boldsymbol{x}|y=k,\theta)$ for the inputs, e.g.,
  - If $\boldsymbol{x} \in \mathbb{R}^D$, then a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ may be appropriate (here $\theta = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$)
  - If $\boldsymbol{x} \in \{0,1\}^D$, then a D-dim Bernoulli may be appropriate

Generative Classification: Estimating Class-Conditional Distr.



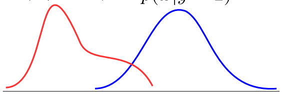
- We usually assume an appropriate class-conditional $p(\boldsymbol{x}|y=k,\theta)$ for the inputs, e.g.,
  - If $\boldsymbol{x} \in \mathbb{R}^D$, then a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ may be appropriate (here $\theta = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$)
  - If $\boldsymbol{x} \in \{0,1\}^D$, then a D-dim Bernoulli may be appropriate
  - Can choose more flexible distributions as well (any density estimation model for that matter)

Generative Classification: Estimating Class-Conditional Distr.



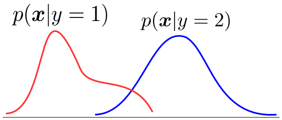
- We usually assume an appropriate class-conditional $p(\boldsymbol{x}|y=k,\theta)$ for the inputs, e.g.,
  - If $\boldsymbol{x} \in \mathbb{R}^D$, then a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ may be appropriate (here $\theta = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$)
  - If $\boldsymbol{x} \in \{0,1\}^D$, then a D-dim Bernoulli may be appropriate
  - Can choose more flexible distributions as well (any density estimation model for that matter)
- For the assumed class-conditional, we can do MLE/MAP estimation or learn full posterior for θ

Generative Classification: Estimating Class-Conditional Distr.



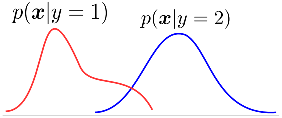
- We usually assume an appropriate class-conditional $p(\boldsymbol{x}|y=k,\theta)$ for the inputs, e.g.,
  - If $\boldsymbol{x} \in \mathbb{R}^D$, then a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ may be appropriate (here $\theta = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$)
  - If $\boldsymbol{x} \in \{0,1\}^D$, then a D-dim Bernoulli may be appropriate
  - Can choose more flexible distributions as well (any density estimation model for that matter)
- For the assumed class-conditional, we can do MLE/MAP estimation or learn full posterior for θ
- An issue: When D is large, we may need to estimate a huge number of parameters

Generative Classification: Estimating Class-Conditional Distr.



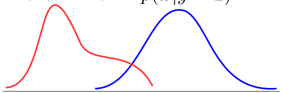
- We usually assume an appropriate class-conditional $p(\boldsymbol{x}|y=k,\theta)$ for the inputs, e.g.,
  - If $\boldsymbol{x} \in \mathbb{R}^D$, then a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ may be appropriate (here $\theta = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$)
  - If $\boldsymbol{x} \in \{0,1\}^D$, then a D-dim Bernoulli may be appropriate
  - Can choose more flexible distributions as well (any density estimation model for that matter)
- For the assumed class-conditional, we can do MLE/MAP estimation or learn full posterior for θ
- An issue: When D is large, we may need to estimate a huge number of parameters, e.g.,
  - A D-dim Gaussian will have D params for mean and $O(D^2)$ params for covariance matrix

Generative Classification: Estimating Class-Conditional Distr.



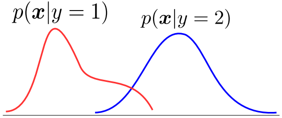
- We usually assume an appropriate class-conditional $p(\boldsymbol{x}|y=k,\theta)$ for the inputs, e.g.,
  - If $\boldsymbol{x} \in \mathbb{R}^D$, then a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ may be appropriate (here $\theta = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$)
  - If $\boldsymbol{x} \in \{0,1\}^D$, then a D-dim Bernoulli may be appropriate
  - Can choose more flexible distributions as well (any density estimation model for that matter)
- For the assumed class-conditional, we can do MLE/MAP estimation or learn full posterior for θ
- An issue: When D is large, we may need to estimate a huge number of parameters, e.g.,
  - A D-dim Gaussian will have D params for mean and $O(D^2)$ params for covariance matrix
  - Some workarounds: Regularize well

Generative Classification: Estimating Class-Conditional Distr.



- We usually assume an appropriate class-conditional $p(\boldsymbol{x}|y=k,\theta)$ for the inputs, e.g.,
  - If $\boldsymbol{x} \in \mathbb{R}^D$, then a D-dim Gaussian $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ may be appropriate (here $\theta = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$)
  - If $\boldsymbol{x} \in \{0,1\}^D$, then a D-dim Bernoulli may be appropriate
  - Can choose more flexible distributions as well (any density estimation model for that matter)
- For the assumed class-conditional, we can do MLE/MAP estimation or learn full posterior for θ
- An issue: When D is large, we may need to estimate a huge number of parameters, e.g.,
  - A D-dim Gaussian will have D params for mean and $O(D^2)$ params for covariance matrix
  - Some workarounds: Regularize well; assume diagonal (or same) covariance for all classes, which means that the features are independent given the class

Generative Classification: Estimating Class-Conditional Distr.



- We usually assume an appropriate class-conditional $p(\boldsymbol{x}|y=k,\theta)$ for the inputs, e.g.,
  - If $\boldsymbol{x} \in \mathbb{R}^D$, then a D-dim Gaussian $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ may be appropriate (here $\theta = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$)
  - If $\boldsymbol{x} \in \{0,1\}^D$, then a D-dim Bernoulli may be appropriate
  - Can choose more flexible distributions as well (any density estimation model for that matter)
- For the assumed class-conditional, we can do MLE/MAP estimation or learn full posterior for θ
- An issue: When D is large, we may need to estimate a huge number of parameters, e.g.,
  - A D-dim Gaussian will have D params for mean and $O(D^2)$ params for covariance matrix
  - Some workarounds: Regularize well; assume diagonal (or same) covariance for all classes, which means that the features are independent given the class (used in "naïve" Bayes models)

Generative Classification: The Prediction Rule

- Suppose we've estimated the parameters of $p(y = k|\theta)$ and $p(\boldsymbol{x}|y = k, \theta)$ (assuming MLE/MAP)

Generative Classification: The Prediction Rule

- Suppose we've estimated the parameters of $p(y = k|\theta)$ and $p(\boldsymbol{x}|y = k, \theta)$ (assuming MLE/MAP)
- The "most likely" class for a test input $\boldsymbol{x}_*$ will be (skipping θ from the notation)

$$y_* = \arg\max_k p(y_* = k|\boldsymbol{x}_*)$$

Generative Classification: The Prediction Rule

- Suppose we've estimated the parameters of $p(y = k|\theta)$ and $p(\boldsymbol{x}|y = k, \theta)$ (assuming MLE/MAP)
- The "most likely" class for a test input $\boldsymbol{x}_*$ will be (skipping θ from the notation)

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

Generative Classification: The Prediction Rule

- Suppose we've estimated the parameters of $p(y = k|\theta)$ and $p(\boldsymbol{x}|y = k, \theta)$ (assuming MLE/MAP)
- The "most likely" class for a test input $\boldsymbol{x}_*$ will be (skipping θ from the notation)

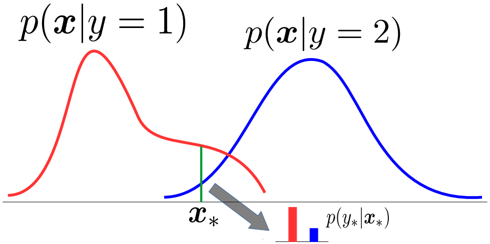
$$y_* = \arg\max_k p(y_* = k|\boldsymbol{x}_*) = \arg\max_k \frac{p(y_* = k)p(\boldsymbol{x}_*|y_* = k)}{p(\boldsymbol{x}_*)} = \arg\max_k p(y_* = k)p(\boldsymbol{x}_*|y_* = k)$$

Generative Classification: The Prediction Rule

- Suppose we've estimated the parameters of $p(y=k|\theta)$ and $p(\boldsymbol{x}|y=k,\theta)$ (assuming MLE/MAP)
- The "most likely" class for a test input $\boldsymbol{x}_*$ will be (skipping θ from the notation)

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

- If $p(y=k)$ is the same for all the classes then, we simple compare $p(\boldsymbol{x}|y=k)$



Generative Classification using Gaussian Class-conditionals

- Recall our generative classification model $p(y = k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$

Generative Classification using Gaussian Class-conditionals

- Recall our generative classification model $p(y = k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$
- Assume each class-conditional to be a Gaussian

$$p(\boldsymbol{x}|y = k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma_k}|}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

Generative Classification using Gaussian Class-conditionals

- Recall our generative classification model $p(y=k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$

- Assume each class-conditional to be a Gaussian

$$p(\boldsymbol{x}|y=k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_{\boldsymbol{k}}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_{\boldsymbol{k}}^{\mathbf{-1}}(\mathbf{x}-\boldsymbol{\mu}_k)\right]$$

- Class-marginal is multinoulli (already saw): $p(y=k) = \pi_k \in (0,1)$, s.t.. $\sum_{k=1}^{K} \pi_k = 1$

Generative Classification using Gaussian Class-conditionals

- Recall our generative classification model $p(y=k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$
- Assume each class-conditional to be a Gaussian

$$p(\boldsymbol{x}|y=k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma_k}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x}-\boldsymbol{\mu}_k)\right]$$

- Class-marginal is multinoulli (already saw): $p(y=k) = \pi_k \in (0,1)$, s.t.. $\sum_{k=1}^K \pi_k = 1$
- Parameters $\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ can be estimated using MLE/MAP/Bayesian approach

Generative Classification using Gaussian Class-conditionals

- Recall our generative classification model $p(y = k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$
- Assume each class-conditional to be a Gaussian

$$p(\boldsymbol{x}|y = k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma_k}|}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_{\boldsymbol{k}}^{-\mathbf{1}}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

- Class-marginal is multinoulli (already saw): $p(y = k) = \pi_k \in (0, 1)$, s.t.. $\sum_{k=1}^{K} \pi_k = 1$
- Parameters $\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ can be estimated using MLE/MAP/Bayesian approach
    - We also saw estimation of π_k's

Generative Classification using Gaussian Class-conditionals

- Recall our generative classification model $p(y=k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$
- Assume each class-conditional to be a Gaussian

$$p(\boldsymbol{x}|y=k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma_k}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]$$

- Class-marginal is multinoulli (already saw): $p(y=k) = \pi_k \in (0,1)$, s.t.. $\sum_{k=1}^K \pi_k = 1$
- Parameters $\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ can be estimated using MLE/MAP/Bayesian approach
  - We also saw estimation of π_k's. $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ can be found via Gaussian parameter estimation

Generative Classification using Gaussian Class-conditionals

- Recall our generative classification model $p(y=k|\boldsymbol{x}) = \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})}$
- Assume each class-conditional to be a Gaussian

$$p(\boldsymbol{x}|y=k) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma_k}|}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x}-\boldsymbol{\mu}_k)\right]$$

- Class-marginal is multinoulli (already saw): $p(y=k) = \pi_k \in (0,1)$, s.t.. $\sum_{k=1}^K \pi_k = 1$
- Parameters $\theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ can be estimated using MLE/MAP/Bayesian approach
  - We also saw estimation of π_k's. $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ can be found via Gaussian parameter estimation
- If using MLE/MAP estimate of θ, the predictive distribution will be

$$p(y_* = k|\boldsymbol{x}_*, \theta) = \frac{\pi_k|\boldsymbol{\Sigma_k}|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}_* - \boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma_k^{-1}}(\mathbf{x}_* - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^K \pi_k|\boldsymbol{\Sigma_k}|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}_* - \boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma_k^{-1}}(\mathbf{x}_* - \boldsymbol{\mu}_k)\right]}$$

Decision Boundaries

- The generative classification prediction rule we saw had

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

Decision Boundaries

- The generative classification prediction rule we saw had

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

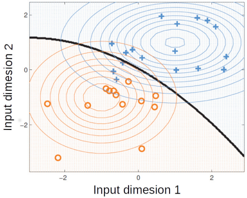
- The decision boundary between any pair of classes will be..

Decision Boundaries

- The generative classification prediction rule we saw had

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

- The decision boundary between any pair of classes will be.. a quadratic curve

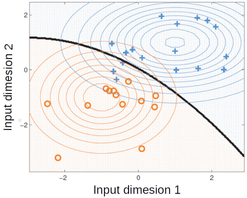


Decision Boundaries

- The generative classification prediction rule we saw had

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

- The decision boundary between any pair of classes will be.. a quadratic curve



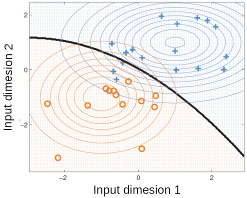
- Reason: For any two classes k and k', at the decision boundary $p(y = k|\boldsymbol{x}) = p(y = k'|\boldsymbol{x})$.

Decision Boundaries

- The generative classification prediction rule we saw had

$$p(y=k|\boldsymbol{x},\theta) = \frac{\pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}$$

- The decision boundary between any pair of classes will be.. a quadratic curve



- Reason: For any two classes k and k', at the decision boundary $p(y=k|\boldsymbol{x}) = p(y=k'|\boldsymbol{x})$.Thus

$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma_{k'}}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{k'}) = 0 \qquad \text{(ignoring terms that don't depend on } \boldsymbol{x})$$

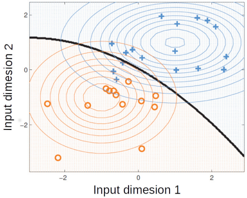
.. defines the decision boundary

Decision Boundaries

- The generative classification prediction rule we saw had

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

- The decision boundary between any pair of classes will be.. a quadratic curve



- Reason: For any two classes k and k', at the decision boundary $p(y = k|\boldsymbol{x}) = p(y = k'|\boldsymbol{x})$.Thus

$$(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - (\mathbf{x} - \boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}_{k'}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{k'}) = 0 \qquad \text{(ignoring terms that don't depend on } \boldsymbol{x}\text{)}$$

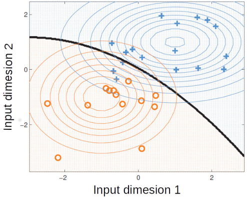
.. defines the decision boundary, which is a quadratic function of $\boldsymbol{x}$

Decision Boundaries

- The generative classification prediction rule we saw had

$$p(y=k|\boldsymbol{x},\theta) = \frac{\pi_k|\boldsymbol{\Sigma_k}|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K}\pi_k|\boldsymbol{\Sigma_k}|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}$$

- The decision boundary between any pair of classes will be.. a quadratic curve



- Reason: For any two classes k and k', at the decision boundary $p(y=k|\boldsymbol{x}) = p(y=k'|\boldsymbol{x})$.Thus

$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma_k}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_{k'})^\top\boldsymbol{\Sigma_{k'}}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{k'}) = 0 \qquad \text{(ignoring terms that don't depend on } \boldsymbol{x}\text{)}$$

.. defines the decision boundary, which is a quadratic function of $\boldsymbol{x}$ (this model is popularly known as Quadratic Discriminant Analysis)

Decision Boundaries

- Let's again consider the generative classification prediction rule with Gaussian class-conditionals

$$p(y=k|\boldsymbol{x},\theta) = \frac{\pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}$$

Decision Boundaries

- Let's again consider the generative classification prediction rule with Gaussian class-conditionals

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

- Let's assume all classes to have the **same covariance** (i.e., same shape/size), i.e., $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$

Decision Boundaries

- Let's again consider the generative classification prediction rule with Gaussian class-conditionals

$$p(y=k|\boldsymbol{x},\theta) = \frac{\pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^K \pi_k |\boldsymbol{\Sigma_k}|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma_k^{-1}}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}$$

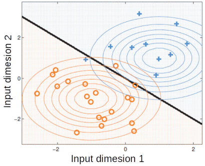
- Let's assume all classes to have the **same covariance** (i.e., same shape/size), i.e., $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$
- Now the decision boundary between any pair of classes will be..

Decision Boundaries

- Let's again consider the generative classification prediction rule with Gaussian class-conditionals

$$p(y = k|\mathbf{x}, \theta) = \frac{\pi_k |\mathbf{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^{K} \pi_k |\mathbf{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

- Let's assume all classes to have the **same covariance** (i.e., same shape/size), i.e., $\mathbf{\Sigma}_k = \mathbf{\Sigma}, \forall k$
- Now the decision boundary between any pair of classes will be.. linear



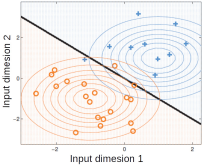
- Reason: For any two classes k and k', at the decision boundary $p(y = k|\boldsymbol{x}) = p(y = k'|\boldsymbol{x})$.

Decision Boundaries

- Let's again consider the generative classification prediction rule with Gaussian class-conditionals

$$p(y=k|\boldsymbol{x},\theta) = \frac{\pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}{\sum_{k=1}^K \pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right]}$$

- Let's assume all classes to have the **same covariance** (i.e., same shape/size), i.e., $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$
- Now the decision boundary between any pair of classes will be.. linear



- Reason: For any two classes k and k', at the decision boundary $p(y=k|\boldsymbol{x}) = p(y=k'|\boldsymbol{x})$. Thus

$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{k'}) = 0 \quad \text{(ignoring terms that don't depend on } \boldsymbol{x}\text{)}$$

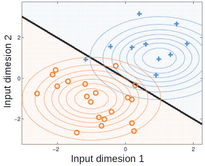
.. terms quadratic in $\boldsymbol{x}$ cancel out in this case and we get a linear function of $\boldsymbol{x}$

Decision Boundaries

- Let's again consider the generative classification prediction rule with Gaussian class-conditionals

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}{\sum_{k=1}^K \pi_k |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]}$$

- Let's assume all classes to have the **same covariance** (i.e., same shape/size), i.e., $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$
- Now the decision boundary between any pair of classes will be.. linear



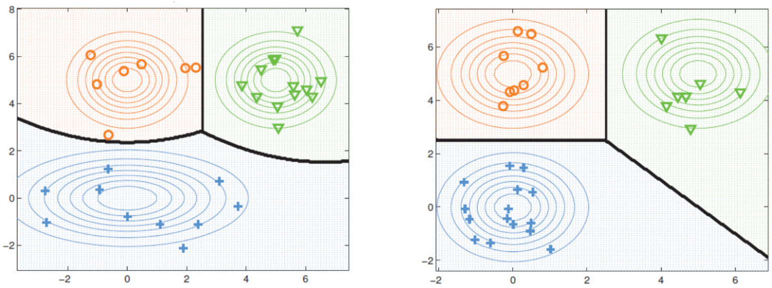
- Reason: For any two classes k and k', at the decision boundary $p(y = k|\boldsymbol{x}) = p(y = k'|\boldsymbol{x})$. Thus

$$(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - (\mathbf{x} - \boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{k'}) = 0 \qquad \text{(ignoring terms that don't depend on } \boldsymbol{x}\text{)}$$

.. terms quadratic in $\boldsymbol{x}$ cancel out in this case and we get a linear function of $\boldsymbol{x}$ (this model is popularly known as Linear or "Fisher" Discriminant Analysis)

Decision Boundaries

- Depending on the form of the covariance matrices, the boundaries can be quadratic/linear



A Closer Look at the Linear Case

- For the linear case (when $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$), we have

$$p(y = k|\boldsymbol{x}, \theta) \propto \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

A Closer Look at the Linear Case

- For the linear case (when $\mathbf{\Sigma}_k = \mathbf{\Sigma}$), we have

$$p(y = k|\boldsymbol{x}, \theta) \propto \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

- Expanding further, we can write the above as

$$p(y = k|\boldsymbol{x}, \theta) \propto \exp\left[\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k\right] \exp\left[\boldsymbol{x}^\top \Sigma^{-1}\boldsymbol{x}\right]$$

A Closer Look at the Linear Case

- For the linear case (when $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$), we have

$$p(y = k|\mathbf{x}, \theta) \propto \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

- Expanding further, we can write the above as

$$p(y = k|\boldsymbol{x}, \theta) \propto \exp\left[\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k\right] \exp\left[\boldsymbol{x}^\top \Sigma^{-1}\boldsymbol{x}\right]$$

- Therefore, the above posterior class probability can be written as

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}{\sum_{k=1}^{K} \exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}$$

where $\boldsymbol{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k$

A Closer Look at the Linear Case

- For the linear case (when $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$), we have

$$p(y = k|\mathbf{x}, \theta) \propto \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

- Expanding further, we can write the above as

$$p(y = k|\boldsymbol{x}, \theta) \propto \exp\left[\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k\right] \exp\left[\boldsymbol{x}^\top \Sigma^{-1}\boldsymbol{x}\right]$$

- Therefore, the above posterior class probability can be written as

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}{\sum_{k=1}^{K} \exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}$$

where $\boldsymbol{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k$ and $b_k = -\frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k$

A Closer Look at the Linear Case

- For the linear case (when $\mathbf{\Sigma}_k = \mathbf{\Sigma}$), we have

$$p(y = k|\mathbf{x}, \theta) \propto \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

- Expanding further, we can write the above as

$$p(y = k|\boldsymbol{x}, \theta) \propto \exp\left[\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k\right] \exp\left[\boldsymbol{x}^\top \Sigma^{-1}\boldsymbol{x}\right]$$

- Therefore, the above posterior class probability can be written as

$$p(y = k|\boldsymbol{x}, \theta) = \frac{\exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}{\sum_{k=1}^{K} \exp\left[\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right]}$$

  where $\boldsymbol{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k$ and $b_k = -\frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k$

- Interestingly, this has exactly the same form as the softmax classification model (saw it in last class), which is a discriminative model, as opposed to a generative model.

A Very Special Case: Prototype based Classification

- We can get a non-probabilistic analogy for the Gaussian generative classification model

A Very Special Case: Prototype based Classification

- We can get a <u>non-probabilistic analogy</u> for the Gaussian generative classification model
- Note the decision rule when $\mathbf{\Sigma}_k = \mathbf{\Sigma}$

$$\hat{y} = \arg\max_k p(y = k|\boldsymbol{x}) \quad = \quad \arg\max_k \quad \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

A Very Special Case: Prototype based Classification

- We can get a non-probabilistic analogy for the Gaussian generative classification model
- Note the decision rule when $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$

$$\hat{y} = \arg\max_k p(y = k|\mathbf{x}) = \arg\max_k \quad \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

$$= \arg\max_k \quad \log \pi_k - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

A Very Special Case: Prototype based Classification

- We can get a non-probabilistic analogy for the Gaussian generative classification model
- Note the decision rule when $\mathbf{\Sigma}_k = \mathbf{\Sigma}$

$$\hat{y} = \arg\max_k p(y = k|\mathbf{x}) = \arg\max_k \quad \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

$$= \arg\max_k \quad \log \pi_k - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

- Further, let's assume the classes to be of equal size, i.e., $\pi_k = 1/K$. Then we will have

$$\hat{y} = \arg\min_k \quad (\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

A Very Special Case: Prototype based Classification

- We can get a non-probabilistic analogy for the Gaussian generative classification model
- Note the decision rule when $\mathbf{\Sigma}_k = \mathbf{\Sigma}$

$$
\begin{aligned}
\hat{y} = \arg\max_k p(y = k|\boldsymbol{x}) &= \arg\max_k \quad \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right] \\
&= \arg\max_k \quad \log \pi_k - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)
\end{aligned}
$$

- Further, let's assume the classes to be of equal size, i.e., $\pi_k = 1/K$. Then we will have

$$
\hat{y} = \arg\min_k \quad (\mathbf{x} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)
$$

- This is equivalent to assigning $\boldsymbol{x}$ to the "closest" class in terms of a Mahalanobis distance

A Very Special Case: Prototype based Classification

- We can get a non-probabilistic analogy for the Gaussian generative classification model
- Note the decision rule when $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$

$$\hat{y} = \arg\max_k p(y = k|\mathbf{x}) = \arg\max_k \; \pi_k \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]$$

$$= \arg\max_k \; \log \pi_k - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

- Further, let's assume the classes to be of equal size, i.e., $\pi_k = 1/K$. Then we will have

$$\hat{y} = \arg\min_k \; (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)$$

- This is equivalent to assigning $\boldsymbol{x}$ to the "closest" class in terms of a Mahalanobis distance
  - The covariance matrix "modulates" how the distances are computed

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple
  - E.g., Gaussian with diagonal covariances $\Rightarrow$ Gaussian naïve Bayes

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple
  - E.g., Gaussian with diagonal covariances $\Rightarrow$ Gaussian naïve Bayes
  - Another popular model is multinomial naïve Bayes (widely used for document classification)

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple
  - E.g., Gaussian with diagonal covariances $\Rightarrow$ Gaussian naïve Bayes
  - Another popular model is multinomial naïve Bayes (widely used for document classification)
  - The so-called "naïve" models assume features to be independent conditioned on y, i.e.,

$$p(\boldsymbol{x}|\theta_y) = \prod_{d=1}^{D} p(x_d|\theta_y) \quad \text{(significantly reduces the number of parameters to be estimated)}$$

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple
  - E.g., Gaussian with diagonal covariances $\Rightarrow$ Gaussian naïve Bayes
  - Another popular model is multinomial naïve Bayes (widely used for document classification)
  - The so-called "naïve" models assume features to be independent conditioned on y, i.e.,

$$p(\boldsymbol{x}|\theta_y) = \prod_{d=1}^{D} p(x_d|\theta_y) \quad \text{(significantly reduces the number of parameters to be estimated)}$$

- Generative classification models work seamlessly for any number of classes

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple
  - E.g., Gaussian with diagonal covariances $\Rightarrow$ Gaussian naïve Bayes
  - Another popular model is multinomial naïve Bayes (widely used for document classification)
  - The so-called "naïve" models assume features to be independent conditioned on y, i.e.,

$$p(\boldsymbol{x}|\theta_y) = \prod_{d=1}^{D} p(x_d|\theta_y) \quad \text{(significantly reduces the number of parameters to be estimated)}$$

- Generative classification models work seamlessly for any number of classes
- Can choose the form of class-conditionals $p(\boldsymbol{x}|y)$ based on the type of inputs $\boldsymbol{x}$

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple
  - E.g., Gaussian with diagonal covariances $\Rightarrow$ Gaussian naïve Bayes
  - Another popular model is multinomial naïve Bayes (widely used for document classification)
  - The so-called "naïve" models assume features to be independent conditioned on y, i.e.,

$$p(\boldsymbol{x}|\theta_y) = \prod_{d=1}^{D} p(x_d|\theta_y) \quad \text{(significantly reduces the number of parameters to be estimated)}$$

- Generative classification models work seamlessly for any number of classes
- Can choose the form of class-conditionals $p(\boldsymbol{x}|y)$ based on the type of inputs $\boldsymbol{x}$
- Can handle missing data (e.g., if some part of the input $\boldsymbol{x}$ is missing) or missing labels

Generative Classification: Some Comments

- A simple but powerful approach to probabilistic classification
- Especially easy to learn if class-conditionals are simple
  - E.g., Gaussian with diagonal covariances $\Rightarrow$ Gaussian naïve Bayes
  - Another popular model is multinomial naïve Bayes (widely used for document classification)
  - The so-called "naïve" models assume features to be independent conditioned on y, i.e.,

$$p(\boldsymbol{x}|\theta_y) = \prod_{d=1}^{D} p(x_d|\theta_y) \quad \text{(significantly reduces the number of parameters to be estimated)}$$

- Generative classification models work seamlessly for any number of classes
- Can choose the form of class-conditionals $p(\boldsymbol{x}|y)$ based on the type of inputs $\boldsymbol{x}$
- Can handle missing data (e.g., if some part of the input $\boldsymbol{x}$ is missing) or missing labels
- Generative models are also useful for unsupervised and semi-supervised learning (will look at later)

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important
- In general, the class-conditional $p(\boldsymbol{x}|y)$ may have too many parameter to be estimated (e.g., if we use full covariance Gaussians when the class-conditionals are Gaussians)

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important
- In general, the class-conditional $p(\boldsymbol{x}|y)$ may have too many parameter to be estimated (e.g., if we use full covariance Gaussians when the class-conditionals are Gaussians)
  - Can be difficult if we don't have enough data for each class

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important
- In general, the class-conditional $p(\boldsymbol{x}|y)$ may have too many parameter to be estimated (e.g., if we use full covariance Gaussians when the class-conditionals are Gaussians)
  - Can be difficult if we don't have enough data for each class
- Assuming shared and/or diagonal covariance for each Gaussian can reduce the number of params

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important
- In general, the class-conditional $p(\boldsymbol{x}|y)$ may have too many parameter to be estimated (e.g., if we use full covariance Gaussians when the class-conditionals are Gaussians)
  - Can be difficult if we don't have enough data for each class
- Assuming shared and/or diagonal covariance for each Gaussian can reduce the number of params
  - .. or the "naïve" assumptions

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important
- In general, the class-conditional $p(\boldsymbol{x}|y)$ may have too many parameter to be estimated (e.g., if we use full covariance Gaussians when the class-conditionals are Gaussians)
  - Can be difficult if we don't have enough data for each class
- Assuming shared and/or diagonal covariance for each Gaussian can reduce the number of params
  - .. or the "naïve" assumptions
- MLE for parameter estimation in these models can be prone to overfitting

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important
- In general, the class-conditional $p(\boldsymbol{x}|y)$ may have too many parameter to be estimated (e.g., if we use full covariance Gaussians when the class-conditionals are Gaussians)
  - Can be difficult if we don't have enough data for each class
- Assuming shared and/or diagonal covariance for each Gaussian can reduce the number of params
  - .. or the "naïve" assumptions
- MLE for parameter estimation in these models can be prone to overfitting
  - Need to regularize the model properly to prevent that

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important
- In general, the class-conditional $p(\boldsymbol{x}|y)$ may have too many parameter to be estimated (e.g., if we use full covariance Gaussians when the class-conditionals are Gaussians)
  - Can be difficult if we don't have enough data for each class
- Assuming shared and/or diagonal covariance for each Gaussian can reduce the number of params
  - .. or the "naïve" assumptions
- MLE for parameter estimation in these models can be prone to overfitting
  - Need to regularize the model properly to prevent that
  - A MAP or fully Bayesian approach can help (will need prior on the parameters)

Generative Classification: Some Comments

- Estimating the class-conditional distributions $p(\boldsymbol{x}|y)$ reliably is important
- In general, the class-conditional $p(\boldsymbol{x}|y)$ may have too many parameter to be estimated (e.g., if we use full covariance Gaussians when the class-conditionals are Gaussians)
  - Can be difficult if we don't have enough data for each class
- Assuming shared and/or diagonal covariance for each Gaussian can reduce the number of params
  - .. or the "naïve" assumptions
- MLE for parameter estimation in these models can be prone to overfitting
  - Need to regularize the model properly to prevent that
  - A MAP or fully Bayesian approach can help (will need prior on the parameters)
- A good density estimation model is necessary for generative classification model to work well

Probabilistic Models for Supervised Learning: Wrapping Up

- Both discriminative and generative models estimate the conditional distribution $p(y|\boldsymbol{x})$

Probabilistic Models for Supervised Learning: Wrapping Up

- Both discriminative and generative models estimate the conditional distribution $p(y|\boldsymbol{x})$
- Note that both are basically doing density estimation for learning $p(y|\boldsymbol{x})$ but in different ways

Probabilistic Models for Supervised Learning: Wrapping Up

- Both discriminative and generative models estimate the conditional distribution $p(y|\boldsymbol{x})$
- Note that both are basically doing density estimation for learning $p(y|\boldsymbol{x})$ but in different ways
- Discriminative models directly model $p(y|\boldsymbol{x})$ via some parameters (say $\boldsymbol{w}$ if using linear model)

$$
\begin{aligned}
p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) && \text{(prob. linear regression)} \\
p(y|\boldsymbol{w}, \boldsymbol{x}) &= \text{Bernoulli}[\sigma(\boldsymbol{w}^\top \boldsymbol{x})] && \text{(prob. linear binary classification)}
\end{aligned}
$$

Probabilistic Models for Supervised Learning: Wrapping Up

- Both discriminative and generative models estimate the conditional distribution $p(y|\boldsymbol{x})$
- Note that both are basically doing density estimation for learning $p(y|\boldsymbol{x})$ but in different ways
- Discriminative models directly model $p(y|\boldsymbol{x})$ via some parameters (say $\boldsymbol{w}$ if using linear model)

$$
\begin{aligned}
p(y|\boldsymbol{w},\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) && \text{(prob. linear regression)} \\
p(y|\boldsymbol{w},\boldsymbol{x}) &= \text{Bernoulli}[\sigma(\boldsymbol{w}^\top \boldsymbol{x})] && \text{(prob. linear binary classification)}
\end{aligned}
$$

- These parameters can then be estimated via MLE, MAP, or fully Bayesian inference

Probabilistic Models for Supervised Learning: Wrapping Up

- Both discriminative and generative models estimate the conditional distribution $p(y|\boldsymbol{x})$
- Note that both are basically doing density estimation for learning $p(y|\boldsymbol{x})$ but in different ways
- Discriminative models directly model $p(y|\boldsymbol{x})$ via some parameters (say $\boldsymbol{w}$ if using linear model)

$$
\begin{aligned}
p(y|\boldsymbol{w},\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{w}^\top\boldsymbol{x},\beta^{-1}) && \text{(prob. linear regression)} \\
p(y|\boldsymbol{w},\boldsymbol{x}) &= \text{Bernoulli}[\sigma(\boldsymbol{w}^\top\boldsymbol{x})] && \text{(prob. linear binary classification)}
\end{aligned}
$$

- These parameters can then be estimated via MLE, MAP, or fully Bayesian inference
- Generative classification models define $p(y|\boldsymbol{x})$ as

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

and estimate the parameters of the class-marginal and class-conditional distributions

Probabilistic Models for Supervised Learning: Wrapping Up

- Both discriminative and generative models estimate the conditional distribution $p(y|\boldsymbol{x})$
- Note that both are basically doing density estimation for learning $p(y|\boldsymbol{x})$ but in different ways
- Discriminative models directly model $p(y|\boldsymbol{x})$ via some parameters (say $\boldsymbol{w}$ if using linear model)

$$
\begin{aligned}
p(y|\boldsymbol{w}, \boldsymbol{x}) &= \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1}) && \text{(prob. linear regression)} \\
p(y|\boldsymbol{w}, \boldsymbol{x}) &= \text{Bernoulli}[\sigma(\boldsymbol{w}^\top \boldsymbol{x})] && \text{(prob. linear binary classification)}
\end{aligned}
$$

- These parameters can then be estimated via MLE, MAP, or fully Bayesian inference
- Generative classification models define $p(y|\boldsymbol{x})$ as

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

  and estimate the parameters of the class-marginal and class-conditional distributions
- Note: Can use generative models for doing regression as well (will be an exercise)