Probabilistic Models for Classification: Generative Classification

Piyush Rai

Probabilistic Machine Learning (CS772A)

Aug 17, 2017

Recap

Parameter Estimation in Probabilistic Models

- Fully Bayesian inference (what we would ideally like to do)

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal likelihood}} \qquad \text{(easy when we have conjugacy!)}$$

Parameter Estimation in Probabilistic Models

- Fully Bayesian inference (what we would ideally like to do)

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal likelihood}} \qquad \text{(easy when we have conjugacy!)}$$

- A cheaper alternative: Maximum Likelihood (ML) Estimation

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \log p(\mathbf{X}|\theta) \left(= \arg\max_{\theta} \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\theta), \quad \text{if data } \mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \text{ is i.i.d.} \right)$$

Parameter Estimation in Probabilistic Models

- Fully Bayesian inference (what we would ideally like to do)

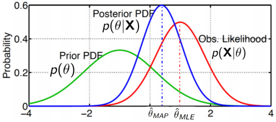
$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal likelihood}} \qquad \text{(easy when we have conjugacy!)}$$

- A cheaper alternative: Maximum Likelihood (ML) Estimation

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \log p(\mathbf{X}|\theta) \left(= \arg\max_{\theta} \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\theta), \quad \text{if data } \mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \text{ is i.i.d.} \right)$$

- Another cheaper (and better) alternative: Maximum-a-Posteriori (MAP) Estimation

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \log p(\theta|\mathbf{X}) = \arg\max_{\theta} \log p(\mathbf{X}|\theta) + \log p(\theta) \left(= \arg\max_{\theta} \left[\sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\theta) + \log p(\theta) \right], \quad \text{if data is i.i.d.} \right)$$



Parameter Estimation in Probabilistic Models

- Fully Bayesian inference (what we would ideally like to do)

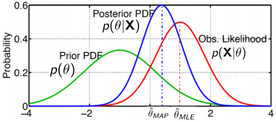
$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal likelihood}} \qquad \text{(easy when we have conjugacy!)}$$

- A cheaper alternative: Maximum Likelihood (ML) Estimation

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \log p(\mathbf{X}|\theta) \left(= \arg\max_{\theta} \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\theta), \quad \text{if data } \mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \text{ is i.i.d.} \right)$$

- Another cheaper (and better) alternative: Maximum-a-Posteriori (MAP) Estimation

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \log p(\theta|\mathbf{X}) = \arg\max_{\theta} \log p(\mathbf{X}|\theta) + \log p(\theta) \left(= \arg\max_{\theta} \left[\sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\theta) + \log p(\theta) \right], \quad \text{if data is i.i.d.} \right)$$



- In some cases, we may want (or may *have* to) to do a "hybrid" sort of inference (fully Bayesian inference for some parameters and MLE/MAP for other parameters)

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta, \mathcal{M})$, the marginal likelihood (or "evidence")

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

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta, \mathcal{M})$, the marginal likelihood (or "evidence")

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

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta,\mathcal{M})$, the marginal likelihood (or "evidence")

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X},\theta|\mathcal{M})d\theta = \int p(\mathbf{X}|\theta,\mathcal{M})p(\theta|\mathcal{M})d\theta = \mathbb{E}_{p(\theta|\mathcal{M})}[p(\mathbf{X}|\theta,\mathcal{M})]$$

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta, \mathcal{M})$, the marginal likelihood (or "evidence")

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X}, \theta|\mathcal{M})d\theta = \int p(\mathbf{X}|\theta, \mathcal{M})p(\theta|\mathcal{M})d\theta = \mathbb{E}_{p(\theta|\mathcal{M})}[p(\mathbf{X}|\theta, \mathcal{M})]$$

- Intuitively, marginal likelihood = The *average proability* that model $\mathcal{M}$ generated $\mathbf{X}$
    - It's the likelihood $p(\mathbf{X}|\theta, \mathcal{M})$ averaged or marginalized/integrated over the prior $p(\theta|\mathcal{M})$

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta,\mathcal{M})$, the marginal likelihood (or "evidence")

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X},\theta|\mathcal{M})d\theta = \int p(\mathbf{X}|\theta,\mathcal{M})p(\theta|\mathcal{M})d\theta = \mathbb{E}_{p(\theta|\mathcal{M})}[p(\mathbf{X}|\theta,\mathcal{M})]$$

- Intuitively, marginal likelihood = The *average proability* that model $\mathcal{M}$ generated $\mathbf{X}$
    - It's the likelihood $p(\mathbf{X}|\theta,\mathcal{M})$ averaged or marginalized/integrated over the prior $p(\theta|\mathcal{M})$
    - Marginal likelihood can also be seen as a measure of "goodness" of model $\mathcal{M}$ for data $\mathbf{X}$

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta,\mathcal{M})$, the marginal likelihood (or "evidence")

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X},\theta|\mathcal{M})d\theta = \int p(\mathbf{X}|\theta,\mathcal{M})p(\theta|\mathcal{M})d\theta = \mathbb{E}_{p(\theta|\mathcal{M})}[p(\mathbf{X}|\theta,\mathcal{M})]$$

- Intuitively, marginal likelihood $=$ The *average proability* that model $\mathcal{M}$ generated $\mathbf{X}$
  - It's the likelihood $p(\mathbf{X}|\theta,\mathcal{M})$ averaged or marginalized/integrated over the prior $p(\theta|\mathcal{M})$
  - Marginal likelihood can also be seen as a measure of "goodness" of model $\mathcal{M}$ for data $\mathbf{X}$
- Note: The word "model" is meant here in a rather broad sense

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta, \mathcal{M})$, the marginal likelihood (or "evidence")

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X}, \theta|\mathcal{M})d\theta = \int p(\mathbf{X}|\theta, \mathcal{M})p(\theta|\mathcal{M})d\theta = \mathbb{E}_{p(\theta|\mathcal{M})}[p(\mathbf{X}|\theta, \mathcal{M})]$$

- Intuitively, marginal likelihood = The *average proability* that model $\mathcal{M}$ generated $\mathbf{X}$
  - It's the likelihood $p(\mathbf{X}|\theta, \mathcal{M})$ averaged or marginalized/integrated over the prior $p(\theta|\mathcal{M})$
  - Marginal likelihood can also be seen as a measure of "goodness" of model $\mathcal{M}$ for data $\mathbf{X}$
- Note: The word "model" is meant here in a rather broad sense
  - Could mean a model from a class of models (e.g., linear, quadratic, cubic regression)

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta, \mathcal{M})$, the marginal likelihood (or "evidence")

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X}, \theta|\mathcal{M})d\theta = \int p(\mathbf{X}|\theta, \mathcal{M})p(\theta|\mathcal{M})d\theta = \mathbb{E}_{p(\theta|\mathcal{M})}[p(\mathbf{X}|\theta, \mathcal{M})]$$

- Intuitively, marginal likelihood = The *average proability* that model $\mathcal{M}$ generated $\mathbf{X}$
  - It's the likelihood $p(\mathbf{X}|\theta, \mathcal{M})$ averaged or marginalized/integrated over the prior $p(\theta|\mathcal{M})$
  - Marginal likelihood can also be seen as a measure of "goodness" of model $\mathcal{M}$ for data $\mathbf{X}$
- Note: The word "model" is meant here in a rather broad sense
  - Could mean a model from a class of models (e.g., linear, quadratic, cubic regression)
  - Could mean a hyperparameter set to some value (each possible value will mean a different model)

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta, \mathcal{M})$, the marginal likelihood (or "evidence")

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X}, \theta|\mathcal{M})d\theta = \int p(\mathbf{X}|\theta, \mathcal{M})p(\theta|\mathcal{M})d\theta = \mathbb{E}_{p(\theta|\mathcal{M})}[p(\mathbf{X}|\theta, \mathcal{M})]$$

- Intuitively, marginal likelihood = The *average proability* that model $\mathcal{M}$ generated $\mathbf{X}$
  - It's the likelihood $p(\mathbf{X}|\theta, \mathcal{M})$ averaged or marginalized/integrated over the prior $p(\theta|\mathcal{M})$
  - Marginal likelihood can also be seen as a measure of "goodness" of model $\mathcal{M}$ for data $\mathbf{X}$
- Note: The word "model" is meant here in a rather broad sense
  - Could mean a model from a class of models (e.g., linear, quadratic, cubic regression)
  - Could mean a hyperparameter set to some value (each possible value will mean a different model)
- When clear from the context, we omit $\mathcal{M}$ and simply use $p(\mathbf{X})$ to denote the marginal likelihood

Marginal Likelihood

- Given data $\mathbf{X}$ from a model $\mathcal{M}$ with likelihood $p(\mathbf{X}|\theta,\mathcal{M})$, the marginal likelihood (or "evidence")

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X},\theta|\mathcal{M})d\theta = \int p(\mathbf{X}|\theta,\mathcal{M})p(\theta|\mathcal{M})d\theta = \mathbb{E}_{p(\theta|\mathcal{M})}[p(\mathbf{X}|\theta,\mathcal{M})]$$

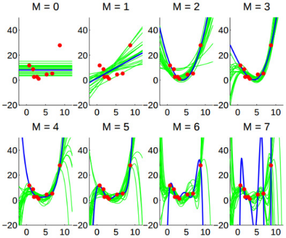
- Intuitively, marginal likelihood = The *average proability* that model $\mathcal{M}$ generated $\mathbf{X}$
  - It's the likelihood $p(\mathbf{X}|\theta,\mathcal{M})$ averaged or marginalized/integrated over the prior $p(\theta|\mathcal{M})$
  - Marginal likelihood can also be seen as a measure of "goodness" of model $\mathcal{M}$ for data $\mathbf{X}$
- Note: The word "model" is meant here in a rather broad sense
  - Could mean a model from a class of models (e.g., linear, quadratic, cubic regression)
  - Could mean a hyperparameter set to some value (each possible value will mean a different model)
- When clear from the context, we omit $\mathcal{M}$ and simply use $p(\mathbf{X})$ to denote the marginal likelihood
- Marginal likelihood can be hard to compute in general, but is a very useful quantity

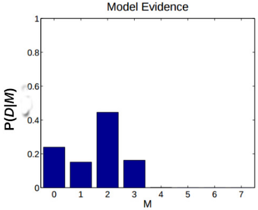
Model Selection via Marginal Likelihood

- Can use marginal likelihood to compare and choose from a set of models $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_K$
  - E.g., choose the model with the largest $p(\mathbf{X}|\mathcal{M})$ (or largest $p(\mathcal{M}|\mathbf{X})$)

Model Selection via Marginal Likelihood

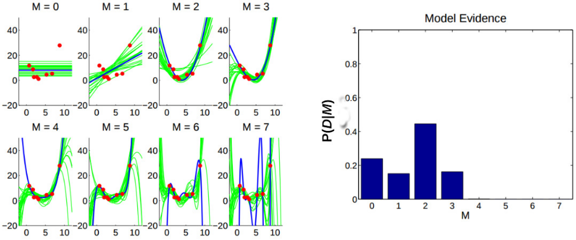
- Can use marginal likelihood to compare and choose from a set of models $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_K$
  - E.g., choose the model with the largest $p(\mathbf{X}|\mathcal{M})$ (or largest $p(\mathcal{M}|\mathbf{X})$)
- Intuition: For a "good" model, many random θ's from it (as opposed to just a select few!) would fit the data reasonably well (so, "on an average", we can say that this model fits the data well)





Model Selection via Marginal Likelihood

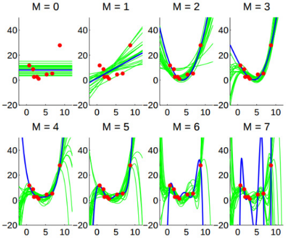
- Can use marginal likelihood to compare and choose from a set of models $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_K$
  - E.g., choose the model with the largest $p(\mathbf{X}|\mathcal{M})$ (or largest $p(\mathcal{M}|\mathbf{X})$)
- Intuition: For a "good" model, many random θ's from it (as opposed to just a select few!) would fit the data reasonably well (so, "on an average", we can say that this model fits the data well)

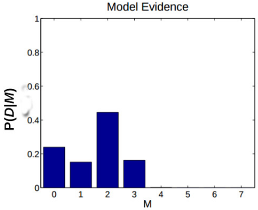


- Important: Note that here we aren't talking about a single θ fitting the data well (which could just be a case of overfitting the data) but many random θ's from the model $\mathcal{M}$

Model Selection via Marginal Likelihood

- Can use marginal likelihood to compare and choose from a set of models $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_K$
  - E.g., choose the model with the largest $p(\mathbf{X}|\mathcal{M})$ (or largest $p(\mathcal{M}|\mathbf{X})$)
- Intuition: For a "good" model, many random θ's from it (as opposed to just a select few!) would fit the data reasonably well (so, "on an average", we can say that this model fits the data well)





- Important: Note that here we aren't talking about a single θ fitting the data well (which could just be a case of overfitting the data) but many random θ's from the model $\mathcal{M}$
- Important: This approach doesn't require a separate held-out data (therefore especially useful when there is very little training data, or in unsupervised learning where cross-validation is impossible)

Predictions via Posterior Averaging

- Having learned a posterior $p(\theta|\mathbf{X})$, the posterior predictive distribution for a new observation $\boldsymbol{x}_*$

$$p(\boldsymbol{x}_*|\mathbf{X}) = \int p(\boldsymbol{x}_*, \theta|\mathbf{X})d\theta = \int p(\boldsymbol{x}_*|\theta)p(\theta|\mathbf{X})d\theta = \mathbb{E}_{p(\theta|\mathbf{X})}[p(\boldsymbol{x}_*|\theta)]$$

Predictions via Posterior Averaging

- Having learned a posterior $p(\theta|\mathbf{X})$, the posterior predictive distribution for a new observation $\boldsymbol{x}_*$

$$p(\boldsymbol{x}_*|\mathbf{X}) = \int p(\boldsymbol{x}_*, \theta|\mathbf{X})d\theta = \int p(\boldsymbol{x}_*|\theta)p(\theta|\mathbf{X})d\theta = \mathbb{E}_{p(\theta|\mathbf{X})}[p(\boldsymbol{x}_*|\theta)]$$

  which doesn't depend on a single θ but averages the prediction $p(\boldsymbol{x}_*|\theta)$ w.r.t. the posterior $p(\theta|\mathbf{X})$

Predictions via Posterior Averaging

- Having learned a posterior $p(\theta|\mathbf{X})$, the posterior predictive distribution for a new observation $\boldsymbol{x}_*$

$$p(\boldsymbol{x}_*|\mathbf{X}) = \int p(\boldsymbol{x}_*, \theta|\mathbf{X})d\theta = \int p(\boldsymbol{x}_*|\theta)p(\theta|\mathbf{X})d\theta = \mathbb{E}_{p(\theta|\mathbf{X})}[p(\boldsymbol{x}_*|\theta)]$$

  which doesn't depend on a single θ but averages the prediction $p(\boldsymbol{x}_*|\theta)$ w.r.t. the posterior $p(\theta|\mathbf{X})$

- Note: Computing the posterior predictive $p(\boldsymbol{x}_*|\mathbf{X})$ is tractable only in rare cases and may require an approximation in other cases (e.g., via sampling methods; we will look at these later)

Predictions via Posterior Averaging

- Having learned a posterior $p(\theta|\mathbf{X})$, the posterior predictive distribution for a new observation $\boldsymbol{x}_*$

$$p(\boldsymbol{x}_*|\mathbf{X}) = \int p(\boldsymbol{x}_*, \theta|\mathbf{X})d\theta = \int p(\boldsymbol{x}_*|\theta)p(\theta|\mathbf{X})d\theta = \mathbb{E}_{p(\theta|\mathbf{X})}[p(\boldsymbol{x}_*|\theta)]$$

  which doesn't depend on a single θ but averages the prediction $p(\boldsymbol{x}_*|\theta)$ w.r.t. the posterior $p(\theta|\mathbf{X})$
- Note: Computing the posterior predictive $p(\boldsymbol{x}_*|\mathbf{X})$ is tractable only in rare cases and may require an approximation in other cases (e.g., via sampling methods; we will look at these later)
- Note: If we only have a point estimate of θ, say θ_{MLE} or θ_{MAP}, then

$$p(\boldsymbol{x}_*|\mathbf{X}) = p(\boldsymbol{x}_*|\theta_{MLE}) \quad \text{or} \quad p(\boldsymbol{x}_*|\mathbf{X}) = p(\boldsymbol{x}_*|\theta_{MAP})$$

Predictions via Posterior Averaging

- Having learned a posterior $p(\theta|\mathbf{X})$, the posterior predictive distribution for a new observation $\boldsymbol{x}_*$

$$p(\boldsymbol{x}_*|\mathbf{X}) = \int p(\boldsymbol{x}_*, \theta|\mathbf{X})d\theta = \int p(\boldsymbol{x}_*|\theta)p(\theta|\mathbf{X})d\theta = \mathbb{E}_{p(\theta|\mathbf{X})}[p(\boldsymbol{x}_*|\theta)]$$

  which doesn't depend on a single θ but averages the prediction $p(\boldsymbol{x}_*|\theta)$ w.r.t. the posterior $p(\theta|\mathbf{X})$

- Note: Computing the posterior predictive $p(\boldsymbol{x}_*|\mathbf{X})$ is tractable only in rare cases and may require an approximation in other cases (e.g., via sampling methods; we will look at these later)

- Note: If we only have a point estimate of θ, say θ_{MLE} or θ_{MAP}, then

$$p(\boldsymbol{x}_*|\mathbf{X}) = p(\boldsymbol{x}_*|\theta_{MLE}) \quad \text{or} \quad p(\boldsymbol{x}_*|\mathbf{X}) = p(\boldsymbol{x}_*|\theta_{MAP})$$

  which is still a distribution but only depends on a single value of θ

Predictions via Posterior Averaging

- Having learned a posterior $p(\theta|\mathbf{X})$, the posterior predictive distribution for a new observation $\boldsymbol{x}_*$

$$p(\boldsymbol{x}_*|\mathbf{X}) = \int p(\boldsymbol{x}_*, \theta|\mathbf{X})d\theta = \int p(\boldsymbol{x}_*|\theta)p(\theta|\mathbf{X})d\theta = \mathbb{E}_{p(\theta|\mathbf{X})}[p(\boldsymbol{x}_*|\theta)]$$

  which doesn't depend on a single θ but averages the prediction $p(\boldsymbol{x}_*|\theta)$ w.r.t. the posterior $p(\theta|\mathbf{X})$

- Note: Computing the posterior predictive $p(\boldsymbol{x}_*|\mathbf{X})$ is tractable only in rare cases and may require an approximation in other cases (e.g., via sampling methods; we will look at these later)

- Note: If we only have a point estimate of θ, say θ_{MLE} or θ_{MAP}, then

$$p(\boldsymbol{x}_*|\mathbf{X}) = p(\boldsymbol{x}_*|\theta_{MLE}) \quad \text{or} \quad p(\boldsymbol{x}_*|\mathbf{X}) = p(\boldsymbol{x}_*|\theta_{MAP})$$

  which is still a distribution but only depends on a single value of θ

- Note: The posterior predictive distribution $p(\boldsymbol{x}_*|\mathbf{X})$ integrates out θ but may still depend on other hyperparameters if those are hard to integrate out (or if we know "good" values for those).

Probabilistic Models for Classification

$$p(y = k|x) = ?, \quad k = 1, 2, \ldots, K$$

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$
  - Note: We can use MLE/MAP or a fully Bayesian procedure to estimate $p(\boldsymbol{x}|y)$ and $p(y)$

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$
  - Note: We can use MLE/MAP or a fully Bayesian procedure to estimate $p(\boldsymbol{x}|y)$ and $p(y)$
  - Finally, use these distributions to compute $p(y = k|\boldsymbol{x})$ for an input $\boldsymbol{x}$ by applying Bayes rule

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

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$
  - Note: We can use MLE/MAP or a fully Bayesian procedure to estimate $p(\boldsymbol{x}|y)$ and $p(y)$
  - Finally, use these distributions to compute $p(y = k|\boldsymbol{x})$ for an input $\boldsymbol{x}$ by applying Bayes rule

$$p(y = k|\boldsymbol{x}) = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})} = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{\sum_{k=1}^{K} p(y = k)p(\boldsymbol{x}|y = k)}$$

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$
  - Note: We can use MLE/MAP or a fully Bayesian procedure to estimate $p(\boldsymbol{x}|y)$ and $p(y)$
  - Finally, use these distributions to compute $p(y = k|\boldsymbol{x})$ for an input $\boldsymbol{x}$ by applying Bayes rule

$$p(y = k|\boldsymbol{x}) = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})} = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{\sum_{k=1}^{K} p(y = k)p(\boldsymbol{x}|y = k)}$$

  - Some examples: Naïve Bayes, Linear/Quadratic *Discriminant* Analysis (name **a misnomer**), etc.

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$
  - Note: We can use MLE/MAP or a fully Bayesian procedure to estimate $p(\boldsymbol{x}|y)$ and $p(y)$
  - Finally, use these distributions to compute $p(y = k|\boldsymbol{x})$ for an input $\boldsymbol{x}$ by applying Bayes rule

$$p(y = k|\boldsymbol{x}) = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})} = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{\sum_{k=1}^{K} p(y = k)p(\boldsymbol{x}|y = k)}$$

  - Some examples: Naïve Bayes, Linear/Quadratic *Discriminant* Analysis (name **a misnomer**), etc.
- Discriminative classification

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$
  - Note: We can use MLE/MAP or a fully Bayesian procedure to estimate $p(\boldsymbol{x}|y)$ and $p(y)$
  - Finally, use these distributions to compute $p(y = k|\boldsymbol{x})$ for an input $\boldsymbol{x}$ by applying Bayes rule

$$p(y = k|\boldsymbol{x}) = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})} = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{\sum_{k=1}^{K} p(y = k)p(\boldsymbol{x}|y = k)}$$

  - Some examples: Naïve Bayes, Linear/Quadratic *Discriminant* Analysis (name **a misnomer**), etc.
- Discriminative classification
  - Based on directly learning a probability distribution $p(y = k|\boldsymbol{x})$ of class y given the input $\boldsymbol{x}$

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$
  - Note: We can use MLE/MAP or a fully Bayesian procedure to estimate $p(\boldsymbol{x}|y)$ and $p(y)$
  - Finally, use these distributions to compute $p(y = k|\boldsymbol{x})$ for an input $\boldsymbol{x}$ by applying Bayes rule

$$p(y = k|\boldsymbol{x}) = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})} = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{\sum_{k=1}^{K} p(y = k)p(\boldsymbol{x}|y = k)}$$

  - Some examples: Naïve Bayes, Linear/Quadratic *Discriminant* Analysis (name **a misnomer**), etc.
- Discriminative classification
  - Based on directly learning a probability distribution $p(y = k|\boldsymbol{x})$ of class y given the input $\boldsymbol{x}$
  - Some examples: Logistic regression, softmax classification, Gaussian Process classification, etc.

Two Approaches to Probabilistic Classification

- Generative classification
  - Use training data to learn class-conditional distribution $p(\boldsymbol{x}|y)$ of inputs from each class $y = 1, \ldots, K$
  - Use training data to learn class prior probability distribution $p(y)$ for each class $y = 1, \ldots, K$
  - Note: We can use MLE/MAP or a fully Bayesian procedure to estimate $p(\boldsymbol{x}|y)$ and $p(y)$
  - Finally, use these distributions to compute $p(y = k|\boldsymbol{x})$ for an input $\boldsymbol{x}$ by applying Bayes rule

$$p(y = k|\boldsymbol{x}) = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})} = \frac{p(y = k)p(\boldsymbol{x}|y = k)}{\sum_{k=1}^{K} p(y = k)p(\boldsymbol{x}|y = k)}$$

  - Some examples: Naïve Bayes, Linear/Quadratic *Discriminant* Analysis (name **a misnomer**), etc.
- Discriminative classification
  - Based on directly learning a probability distribution $p(y = k|\boldsymbol{x})$ of class y given the input $\boldsymbol{x}$
  - Some examples: Logistic regression, softmax classification, Gaussian Process classification, etc.
- Important: Generative approach models the inputs $\boldsymbol{x}$. Discriminative approach treats $\boldsymbol{x}$ as fixed

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y = k|\boldsymbol{x})$$

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y = k|\boldsymbol{x}) = \arg\max_k \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})}$$

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y=k|\boldsymbol{x}) = \arg\max_k \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})} = \arg\max_k p(y=k)p(\boldsymbol{x}|y=k)$$

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y = k|\boldsymbol{x}) = \arg\max_k \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})} = \arg\max_k p(y = k)p(\boldsymbol{x}|y = k)$$

- If we know true $p(\boldsymbol{x}|y)$ and $p(y)$, this approach provably has smallest possible classification error

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y = k|\boldsymbol{x}) = \arg\max_k \frac{p(y = k)p(\boldsymbol{x}|y = k)}{p(\boldsymbol{x})} = \arg\max_k p(y = k)p(\boldsymbol{x}|y = k)$$

- If we know true $p(\boldsymbol{x}|y)$ and $p(y)$, this approach provably has smallest possible classification error
  - Also known as the Bayes optimal classifier (minimizes the misclassification probability)

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y=k|\boldsymbol{x}) = \arg\max_k \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})} = \arg\max_k p(y=k)p(\boldsymbol{x}|y=k)$$

- If we know true $p(\boldsymbol{x}|y)$ and $p(y)$, this approach provably has smallest possible classification error
  - Also known as the Bayes optimal classifier (minimizes the misclassification probability)
- However, the generative classification also suffers from some issues

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y=k|\boldsymbol{x}) = \arg\max_k \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})} = \arg\max_k p(y=k)p(\boldsymbol{x}|y=k)$$

- If we know true $p(\boldsymbol{x}|y)$ and $p(y)$, this approach provably has smallest possible classification error
  - Also known as the Bayes optimal classifier (minimizes the misclassification probability)
- However, the generative classification also suffers from some issues, e.g.,
  - May require too many parameters to estimate (especially for class-conditional distributions $p(\boldsymbol{x}|y)$)

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y=k|\boldsymbol{x}) = \arg\max_k \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})} = \arg\max_k p(y=k)p(\boldsymbol{x}|y=k)$$

- If we know true $p(\boldsymbol{x}|y)$ and $p(y)$, this approach provably has smallest possible classification error
  - Also known as the Bayes optimal classifier (minimizes the misclassification probability)
- However, the generative classification also suffers from some issues, e.g.,
  - May require too many parameters to estimate (especially for class-conditional distributions $p(\boldsymbol{x}|y)$)
  - May do badly if our assumed $p(\boldsymbol{x}|y)$ (say a Gaussian) is very different from the true class-conditional

Generative Classification

- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y=k|\boldsymbol{x}) = \arg\max_k \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})} = \arg\max_k p(y=k)p(\boldsymbol{x}|y=k)$$

- If we know true $p(\boldsymbol{x}|y)$ and $p(y)$, this approach provably has smallest possible classification error
  - Also known as the Bayes optimal classifier (minimizes the misclassification probability)
- However, the generative classification also suffers from some issues, e.g.,
  - May require too many parameters to estimate (especially for class-conditional distributions $p(\boldsymbol{x}|y)$)
  - May do badly if our assumed $p(\boldsymbol{x}|y)$ (say a Gaussian) is very different from the true class-conditional
- Nevertheless, a very powerful approach, especially if we have a good method to estimate the class-conditional densities.

Generative Classification

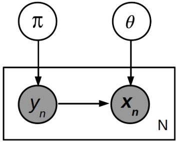
- Assume we've learned $p(\boldsymbol{x}|y)$ and $p(y)$. The classification rule will be

$$\hat{y} = \arg\max_k p(y=k|\boldsymbol{x}) = \arg\max_k \frac{p(y=k)p(\boldsymbol{x}|y=k)}{p(\boldsymbol{x})} = \arg\max_k p(y=k)p(\boldsymbol{x}|y=k)$$

- If we know true $p(\boldsymbol{x}|y)$ and $p(y)$, this approach provably has smallest possible classification error
  - Also known as the Bayes optimal classifier (minimizes the misclassification probability)
- However, the generative classification also suffers from some issues, e.g.,
  - May require too many parameters to estimate (especially for class-conditional distributions $p(\boldsymbol{x}|y)$)
  - May do badly if our assumed $p(\boldsymbol{x}|y)$ (say a Gaussian) is very different from the true class-conditional
- Nevertheless, a very powerful approach, especially if we have a good method to estimate the class-conditional densities. Also enables doing semi-supervised learning (more on this later)

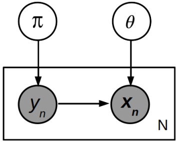
Generative Classification: A Generative Story of Labeled Data

- Basic idea: Each input $\boldsymbol{x}_n$ is generated conditioned on the value of the corresponding label y_n



Generative Classification: A Generative Story of Labeled Data

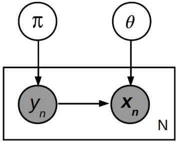
- Basic idea: Each input $\boldsymbol{x}_n$ is generated conditioned on the value of the corresponding label y_n



- Assume a class probability param. vector $\pi = \{\pi_1, \ldots, \pi_K\}$, s.t., $\sum_{k=1}^{K} \pi_k = 1$: Defines $p(y = k)$

Generative Classification: A Generative Story of Labeled Data

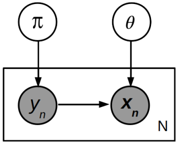
- Basic idea: Each input $\boldsymbol{x}_n$ is generated conditioned on the value of the corresponding label y_n



- Assume a class probability param. vector $\pi = \{\pi_1, \ldots, \pi_K\}$, s.t., $\sum_{k=1}^{K} \pi_k = 1$: Defines $p(y = k)$
- Assume $\theta = \{\theta_1, \ldots, \theta_k\}$ to be parameters distributions generating the data: Defines $p(\boldsymbol{x}|y = k)$

Generative Classification: A Generative Story of Labeled Data

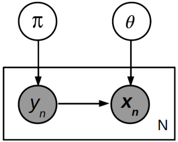
- Basic idea: Each input $\boldsymbol{x}_n$ is generated conditioned on the value of the corresponding label y_n



- Assume a class probability param. vector $\pi = \{\pi_1, \ldots, \pi_K\}$, s.t., $\sum_{k=1}^{K} \pi_k = 1$: Defines $p(y = k)$
- Assume $\theta = \{\theta_1, \ldots, \theta_k\}$ to be parameters distributions generating the data: Defines $p(\boldsymbol{x}|y = k)$
- Again note that there is no directly defined $p(y = k|\boldsymbol{x})$ in this case

Generative Classification: A Generative Story of Labeled Data

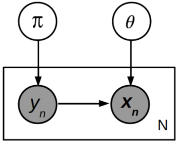
- Basic idea: Each input $\boldsymbol{x}_n$ is generated conditioned on the value of the corresponding label y_n



- Assume a class probability param. vector $\pi = \{\pi_1, \ldots, \pi_K\}$, s.t., $\sum_{k=1}^{K} \pi_k = 1$: Defines $p(y = k)$
- Assume $\theta = \{\theta_1, \ldots, \theta_k\}$ to be parameters distributions generating the data: Defines $p(\boldsymbol{x}|y = k)$
- Again note that there is no directly defined $p(y = k|\boldsymbol{x})$ in this case
- The associated generative story for each example $(\boldsymbol{x}_n, y_n)$ is as follows

Generative Classification: A Generative Story of Labeled Data

- Basic idea: Each input $\boldsymbol{x}_n$ is generated conditioned on the value of the corresponding label y_n

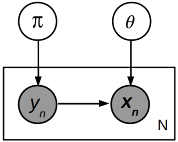


- Assume a class probability param. vector $\pi = \{\pi_1, \ldots, \pi_K\}$, s.t., $\sum_{k=1}^{K} \pi_k = 1$: Defines $p(y = k)$
- Assume $\theta = \{\theta_1, \ldots, \theta_k\}$ to be parameters distributions generating the data: Defines $p(\boldsymbol{x}|y = k)$
- Again note that there is no directly defined $p(y = k|\boldsymbol{x})$ in this case
- The associated generative story for each example $(\boldsymbol{x}_n, y_n)$ is as follows
  - First choose a class/label $y_n \in \{1, 2, \ldots, K\}$

$$y_n \sim \text{multinoulli}(\pi)$$

Generative Classification: A Generative Story of Labeled Data

- Basic idea: Each input $\boldsymbol{x}_n$ is generated conditioned on the value of the corresponding label y_n



- Assume a class probability param. vector $\pi = \{\pi_1, \ldots, \pi_K\}$, s.t., $\sum_{k=1}^{K} \pi_k = 1$: Defines $p(y = k)$
- Assume $\theta = \{\theta_1, \ldots, \theta_k\}$ to be parameters distributions generating the data: Defines $p(\boldsymbol{x}|y = k)$
- Again note that there is no directly defined $p(y = k|\boldsymbol{x})$ in this case
- The associated generative story for each example $(\boldsymbol{x}_n, y_n)$ is as follows
  - First choose a class/label $y_n \in \{1, 2, \ldots, K\}$

$$y_n \sim \text{multinoulli}(\pi)$$

  - Now draw ("generate") the input $\boldsymbol{x}_n$ from a distribution specific to the value y_n takes

$$\boldsymbol{x}_n | y_n \sim p(\boldsymbol{x}|\theta_{y_n})$$

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

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

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

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

- Assume $p(y=k) = \pi_k \in (0,1)$, s.t.. $\sum_{k=1}^K \pi_k = 1$ (equivalent to saying that $p(y)$ is multinoulli)

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

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

- Assume $p(y = k) = \pi_k \in (0, 1)$, s.t.. $\sum_{k=1}^K \pi_k = 1$ (equivalent to saying that $p(y)$ is multinoulli)
- Parameters $\pi = \{\pi_k\}_{k=1}^K$, $\theta = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ 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

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

- Assume $p(y = k) = \pi_k \in (0, 1)$, s.t.. $\sum_{k=1}^{K} \pi_k = 1$ (equivalent to saying that $p(y)$ is multinoulli)
- Parameters $\pi = \{\pi_k\}_{k=1}^{K}$, $\theta = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{K}$ can be estimated using MLE/MAP/Bayesian approach
- After estimating the parameters,the "plug-in" prediction rule will be

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

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

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

- Assume $p(y=k) = \pi_k \in (0,1)$, s.t.. $\sum_{k=1}^K \pi_k = 1$ (equivalent to saying that $p(y)$ is multinoulli)
- Parameters $\pi = \{\pi_k\}_{k=1}^K$, $\theta = \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ can be estimated using MLE/MAP/Bayesian approach
- After estimating the parameters,the "plug-in" prediction rule will be

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

- An Important Note: The above rule is not truly Bayesian (it WILL be if we inferred the full posterior distribution of the parameters and averaged $p(y=k|\boldsymbol{x},\theta,\pi)$ over that posterior distribution)

Parameter Estimation via MLE

- Given data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, \ldots, K\}$, the log likelihood will be

$$\log p(\mathcal{D}|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n|\Theta)$$

Parameter Estimation via MLE

- Given data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, \ldots, K\}$, the log likelihood will be

$$\log p(\mathcal{D}|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n|\theta_{y_n})p(y_n|\pi)$$

Parameter Estimation via MLE

- Given data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, \ldots, K\}$, the log likelihood will be

$$\log p(\mathcal{D}|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n|\theta_{y_n})p(y_n|\pi) = \sum_{n=1}^{N} [\log p(\boldsymbol{x}_n|\theta_{y_n}) + \log p(y_n|\pi)]$$

Parameter Estimation via MLE

- Given data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, \ldots, K\}$, the log likelihood will be

$$\log p(\mathcal{D}|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n|\theta_{y_n}) p(y_n|\pi) = \sum_{n=1}^{N} [\log p(\boldsymbol{x}_n|\theta_{y_n}) + \log p(y_n|\pi)]$$

where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ denotes all the unknown parameters of the model

Parameter Estimation via MLE

- Given data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, \dots, K\}$, the log likelihood will be

$$\log p(\mathcal{D}|\Theta) = \log \prod_{n=1}^N p(\boldsymbol{x}_n, y_n|\Theta) = \log \prod_{n=1}^N p(\boldsymbol{x}_n|\theta_{y_n})p(y_n|\pi) = \sum_{n=1}^N [\log p(\boldsymbol{x}_n|\theta_{y_n}) + \log p(y_n|\pi)]$$

  where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ denotes all the unknown parameters of the model

- Here $\log p(\boldsymbol{x}_n|\theta_{y_n}) = \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{y_n}, \boldsymbol{\Sigma}_{y_n})$

Parameter Estimation via MLE

- Given data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, \ldots, K\}$, the log likelihood will be

$$\log p(\mathcal{D}|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n|\theta_{y_n})p(y_n|\pi) = \sum_{n=1}^{N} [\log p(\boldsymbol{x}_n|\theta_{y_n}) + \log p(y_n|\pi)]$$

  where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ denotes all the unknown parameters of the model

- Here $\log p(\boldsymbol{x}_n|\theta_{y_n}) = \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{y_n}, \boldsymbol{\Sigma}_{y_n})$

- Here $\log p(y_n|\pi) = \log \text{multinoulli}(y_n|\pi) = \log \prod_{k=1}^{K} \pi_k^{\mathbb{I}[y_n=k]}$

Parameter Estimation via MLE

- Given data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, \dots, K\}$, the log likelihood will be

$$\log p(\mathcal{D}|\Theta) = \log \prod_{n=1}^N p(\boldsymbol{x}_n, y_n|\Theta) = \log \prod_{n=1}^N p(\boldsymbol{x}_n|\theta_{y_n}) p(y_n|\pi) = \sum_{n=1}^N [\log p(\boldsymbol{x}_n|\theta_{y_n}) + \log p(y_n|\pi)]$$

  where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ denotes all the unknown parameters of the model

- Here $\log p(\boldsymbol{x}_n|\theta_{y_n}) = \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{y_n}, \boldsymbol{\Sigma}_{y_n})$

- Here $\log p(y_n|\pi) = \log \text{multinoulli}(y_n|\pi) = \log \prod_{k=1}^K \pi_k^{\mathbb{I}[y_n=k]}$

- Substituting these, we can write the likelihood as

$$\log p(\mathcal{D}|\Theta) = \sum_{k=1}^K \sum_{n:y_n=k} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{n=1}^N \sum_{k=1}^K \mathbb{I}[y_n = k] \log \pi_k \quad \text{(Exercise: Verify this!)}$$

Parameter Estimation via MLE

- Given data $\mathcal{D} = \{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{1, \ldots, K\}$, the log likelihood will be

$$\log p(\mathcal{D}|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n, y_n|\Theta) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n|\theta_{y_n}) p(y_n|\pi) = \sum_{n=1}^{N} [\log p(\boldsymbol{x}_n|\theta_{y_n}) + \log p(y_n|\pi)]$$

  where $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ denotes all the unknown parameters of the model

- Here $\log p(\boldsymbol{x}_n|\theta_{y_n}) = \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{y_n}, \boldsymbol{\Sigma}_{y_n})$
- Here $\log p(y_n|\pi) = \log \text{multinoulli}(y_n|\pi) = \log \prod_{k=1}^K \pi_k^{\mathbb{I}[y_n=k]}$
- Substituting these, we can write the likelihood as

$$\log p(\mathcal{D}|\Theta) = \sum_{k=1}^{K} \sum_{n:y_n=k} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[y_n = k] \log \pi_k \quad \text{(Exercise: Verify this!)}$$

- We can now do MLE for the parameters $\Theta = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

Parameter Estimation via MLE

- The log likelihood is

$$\log p(\mathcal{D}|\Theta) = \sum_{k=1}^{K} \sum_{n:y_n=k} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[y_n = k] \log \pi_k$$

Parameter Estimation via MLE

- The log likelihood is

$$\log p(\mathcal{D}|\Theta) = \sum_{k=1}^{K} \sum_{n:y_n=k} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[y_n = k] \log \pi_k$$

- MLE for π_k is straightforward: $\pi_k = \frac{N_k}{N}$ (makes sense intuitively, but verify as an exercise)

Parameter Estimation via MLE

- The log likelihood is

$$\log p(\mathcal{D}|\Theta) = \sum_{k=1}^{K} \sum_{n:y_n=k} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[y_n = k] \log \pi_k$$

- MLE for π_k is straightforward: $\pi_k = \frac{N_k}{N}$ (makes sense intuitively, but verify as an exercise)
- MLE for $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ is a Gaussian parameter estimation problem given data from class k. Solution is:

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{n:y_n=k} \boldsymbol{x}_n, \qquad \hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{n:y_n=k} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^\top \quad \text{(Exercise: Verify this!)}$$

Parameter Estimation via MLE

- The log likelihood is

$$\log p(\mathcal{D}|\Theta) = \sum_{k=1}^{K} \sum_{n:y_n=k} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[y_n = k] \log \pi_k$$

- MLE for π_k is straightforward: $\pi_k = \frac{N_k}{N}$ (makes sense intuitively, but verify as an exercise)
- MLE for $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ is a Gaussian parameter estimation problem given data from class k. Solution is:

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{n:y_n=k} \boldsymbol{x}_n, \qquad \hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{n:y_n=k} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^\top \quad \text{(Exercise: Verify this!)}$$

.. which is simply the empirical mean and empirical covariance of the inputs from class k

Parameter Estimation via MLE

- The log likelihood is

$$\log p(\mathcal{D}|\Theta) = \sum_{k=1}^{K} \sum_{n:y_n=k} \log \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[y_n = k] \log \pi_k$$

- MLE for π_k is straightforward: $\pi_k = \frac{N_k}{N}$ (makes sense intuitively, but verify as an exercise)
- MLE for $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ is a Gaussian parameter estimation problem given data from class k. Solution is:

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{N_k} \sum_{n:y_n=k} \boldsymbol{x}_n, \qquad \hat{\boldsymbol{\Sigma}}_k = \frac{1}{N_k} \sum_{n:y_n=k} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^\top \quad \text{(Exercise: Verify this!)}$$

.. which is simply the empirical mean and empirical covariance of the inputs from class k

- Note: Can also do MAP or fully Bayesian inference for these parameters

Decision Boundaries

- The generative classification prediction rule was

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

Decision Boundaries

- The generative classification prediction rule was

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

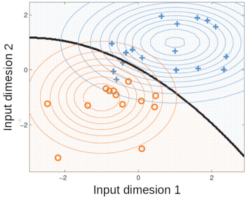
- The decision boundary between any pair of classes will be..

Decision Boundaries

- The generative classification prediction rule was

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

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

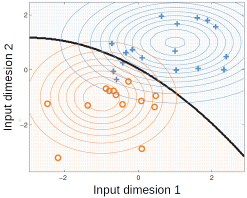


Decision Boundaries

- The generative classification prediction rule was

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

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



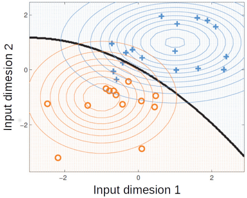
- Reason: For any two classes k and k', at the decision boundary $p(y = k|\boldsymbol{x}) = p(y = k'|\boldsymbol{x})$.

Decision Boundaries

- The generative classification prediction rule was

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

- 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

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

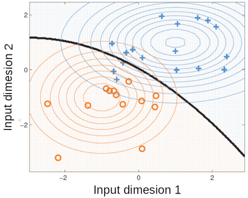
.. defines the decision boundary

Decision Boundaries

- The generative classification prediction rule was

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

- 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

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

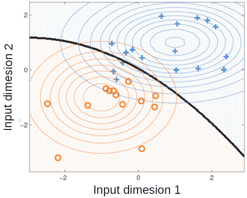
.. defines the decision boundary, which is a quadratic function of $\boldsymbol{x}$

Decision Boundaries

- The generative classification prediction rule was

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

- 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

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

Decision Boundaries

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

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

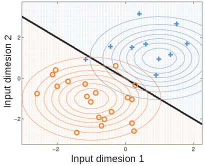
- Let's assume all classes to have the **same covariance** (i.e., same shape/size), i.e., $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}, \forall k$
- Now the decision boundary between any pair of classes will be..

Decision Boundaries

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

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

- 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



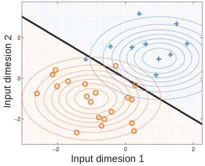
- Reason: For any two classes k and k', at the decision boundary $p(y=k|\boldsymbol{x}) = p(y=k'|\boldsymbol{x})$.

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

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

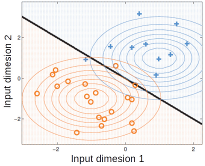
.. terms quadratic in $\boldsymbol{x}$ cancel out in this case and we get a linear function of $\boldsymbol{x}$

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



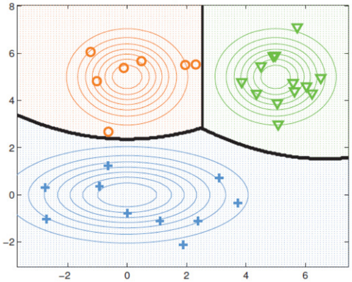
- Reason: For any two classes k and k', at the decision boundary $p(y = k|\boldsymbol{x}) = p(y = k'|\boldsymbol{x})$. Thus

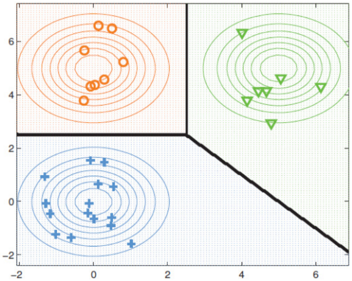
$$(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - (\mathbf{x} - \boldsymbol{\mu}_{k'})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{k'}) = 0 \qquad \text{(ignoring terms that don't depend on } \boldsymbol{x}\text{)}$$

.. 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 $\mathbf{\Sigma}_k = \mathbf{\Sigma}$), we have

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

A Closer Look at the Linear Case

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

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

- 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

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

- 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 $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$), we have

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

- 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, which is a discriminative model (will look at these later), as opposed to a generative model.

Analogy: Classifying by Computing Distances from Classes

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

Analogy: Classifying by Computing Distances from Classes

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

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

Analogy: Classifying by Computing Distances from Classes

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

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

Analogy: Classifying by Computing Distances from Classes

- 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

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

Analogy: Classifying by Computing Distances from Classes

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

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

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

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

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

Analogy: Classifying by Computing Distances from Classes

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

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

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

- 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
  - The covariance matrix "modulate" 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 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