

Statistical Classifiers

Gaussian Classifier



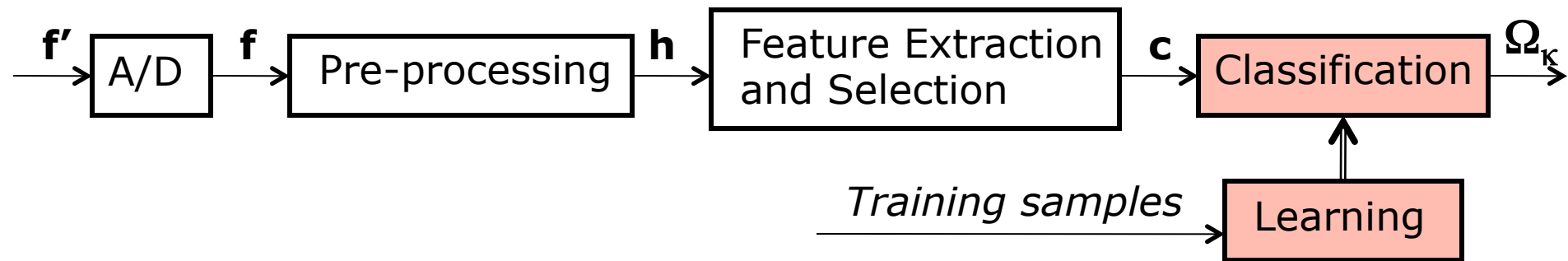
Dr. Elli Angelopoulou

Lehrstuhl für Mustererkennung (Informatik 5)

Friedrich-Alexander-Universität Erlangen-Nürnberg



Pattern Recognition Pipeline



■ Classification

- Statistical classifiers
 - Bayesian classifier
 - Gaussian classifier



Framework of Statistical Classifiers

- A statistical classifier typically uses a probabilistic decision function:

$$\delta(\Omega_\lambda | \vec{c}) : \vec{c} \rightarrow \Omega_\lambda$$

- Each decision function $\delta()$ has a risk associated with it:

$$R(\delta) = \int_{R\vec{c}} \sum_{\lambda=0}^K u_\lambda(\vec{c}) \delta(\Omega_\lambda | \vec{c}) d\vec{c}$$

where

$$u_\lambda(\vec{c}) = \sum_{\kappa=1}^K r_{\lambda,\kappa} p(\Omega_\kappa) p(\vec{c} | \Omega_\kappa)$$

Framework of Statistical Classifiers - cont



- The optimal classifier is the one that uses the decision function δ that minimizes the risk $R(\delta)$:

$$\hat{\delta} = \underset{\delta}{\operatorname{arg\,min}} R(\delta)$$

which occurs when the decision function “votes” for the minimal $u_{\lambda}(\vec{c})$ value.

- Any classifier that maximizes posterior probabilities is a Bayesian classifier:

$$\delta(\Omega_{\lambda} | \vec{c}) = \begin{cases} 1 & \text{if } \lambda = \underset{\kappa}{\operatorname{arg\,max}} p(\Omega_{\kappa} | \vec{c}) \\ 0 & \text{otherwise} \end{cases}$$



Gaussian Classifier

- It is a Bayesian classifier where we have **normally distributed class-conditional feature vectors** $p(\vec{c}|\Omega_k)$.
- Example: 2-class problem, Ω_1, Ω_2 .
- The training data includes N sample feature vectors from class Ω_1 and M sample features vectors from Ω_2 .
- The feature vectors within each class are normally distributed.

$$\Omega_1 : \{ {}^1\vec{c}_1, {}^1\vec{c}_2, \dots, {}^1\vec{c}_N \} \in \mathcal{N}(\vec{c}, \vec{\mu}_1, \Sigma_1)$$

$$\Omega_2 : \{ {}^2\vec{c}_1, {}^2\vec{c}_2, \dots, {}^2\vec{c}_M \} \in \mathcal{N}(\vec{c}, \vec{\mu}_2, \Sigma_2)$$



Gaussian Classifier Example

- Since the Gaussian classifier is a Bayesian classifier we have to decide based on the maximal posterior probability $p(\Omega_\lambda|\vec{c})$.
- How can we compute, $p(\Omega_1|\vec{c})$ and $p(\Omega_2|\vec{c})$?
- Use the Bayesian rule.
- Use Maximum Likelihood Estimation (MLE).

Review: It is a statistical method that can be used when we have a fixed data set and an underlying probability model to estimate the most likely values of the parameters of the underlying probability model. For normal distributions MLE gives a unique solution.



Gaussian Classifier Example - continued

- More specifically: $\max_{\lambda} p(\Omega_{\lambda} | \vec{c}) = \max_{\lambda} p(\Omega_{\lambda}) p(\vec{c} | \Omega_{\lambda})$
- Assuming that our training data is a fair representation of the true population, the class probability can be estimated from the samples as follows:

$$p(\Omega_1) = \frac{N}{N + M}$$

- The class conditional probability is normally distributed

$$p(\vec{c} | \Omega_1) \approx \mathcal{N}(\vec{c}, \vec{\mu}_1, \Sigma_1)$$

- Via MLE $\vec{\mu}_1 = \frac{1}{N} \sum_{i=1}^N {}^1\vec{c}_i$ and $\Sigma_1 = \frac{1}{N} \sum_{i=1}^N ({}^1\vec{c}_i - \vec{\mu}_1)({}^1\vec{c}_i - \vec{\mu}_1)^T$



Gaussian Classifier Example - continued

- Similarly we replace $p(\Omega_2|\vec{c}) = p(\Omega_2)p(\vec{c}|\Omega_2)$
- Since the two classes represent the entire population we can simply use $p(\Omega_1)$ in estimating the class probability for Ω_2 :

$$p(\Omega_2) = 1 - p(\Omega_1)$$

- The class conditional probability for the 2nd class Ω_2 is also normally distributed:

$$p(\vec{c}|\Omega_2) \approx \mathcal{N}(\vec{c}, \vec{\mu}_2, \Sigma_2)$$

- Via MLE $\vec{\mu}_2 = \frac{1}{M} \sum_{i=1}^M {}^2\vec{c}_i$ and $\Sigma_2 = \frac{1}{M} \sum_{i=1}^M ({}^2\vec{c}_i - \vec{\mu}_2)({}^2\vec{c}_i - \vec{\mu}_2)^T$



Decision Rule of a Gaussian Classifier

- Once we have estimated $p(\Omega_1|\vec{c})$ and $p(\Omega_2|\vec{c})$ we can assign the feature vector \vec{c} to the class that gives the largest posterior probability.
- The decision rule for a Gaussian classifiers is:

$$\delta(\Omega_\lambda|\vec{c}) = \begin{cases} 1 & \text{if } \lambda = \arg \max_{\kappa} p(\Omega_\kappa) p(\vec{c}|\Omega_\kappa) \\ 0 & \text{otherwise} \end{cases}$$

where

$$\lambda = \arg \max_{\kappa} p(\Omega_\kappa) p(\vec{c}|\Omega_\kappa)$$

$$\lambda = \arg \max_{\kappa} p(\Omega_\kappa) \mathcal{N}(\vec{c}, \vec{\mu}_\kappa, \Sigma_\kappa)$$



Decision Rule of a Gaussian Classifier - cont

Keep on expanding...

$$\lambda = \arg \max_{\kappa} p(\Omega_{\kappa}) \mathcal{N}(\vec{c}, \vec{\mu}_{\kappa}, \Sigma_{\kappa})$$

$$\lambda = \arg \max_{\kappa} \log(p(\Omega_{\kappa}) \mathcal{N}(\vec{c}, \vec{\mu}_{\kappa}, \Sigma_{\kappa}))$$

$$\lambda = \arg \max_{\kappa} (\log(p(\Omega_{\kappa})) + \log(\mathcal{N}(\vec{c}, \vec{\mu}_{\kappa}, \Sigma_{\kappa})))$$

$$\lambda = \arg \max_{\kappa} \left(\log(p(\Omega_{\kappa})) + \log \left(\frac{1}{\sqrt{2\pi\Sigma_{\kappa}}} e^{-\frac{1}{2}(\vec{c} - \vec{\mu}_{\kappa})^T \Sigma_{\kappa}^{-1} (\vec{c} - \vec{\mu}_{\kappa})} \right) \right)$$

$$\lambda = \arg \max_{\kappa} \left(\log(p(\Omega_{\kappa})) + \log \left(\frac{1}{\sqrt{2\pi\Sigma_{\kappa}}} \right) - \frac{1}{2} (\vec{c} - \vec{\mu}_{\kappa})^T \Sigma_{\kappa}^{-1} (\vec{c} - \vec{\mu}_{\kappa}) \right)$$

$$\lambda = \arg \max_{\kappa} \left(\log(p(\Omega_{\kappa})) - \frac{1}{2} \log(|2\pi\Sigma_{\kappa}|) - \frac{1}{2} (\vec{c} - \vec{\mu}_{\kappa})^T \Sigma_{\kappa}^{-1} (\vec{c} - \vec{\mu}_{\kappa}) \right)$$



Parameter Tying

- Often, when we have too many parameters, we tie together one degree of freedom to simplify the problem at hand. This is called *parameter tying*.
- In the case of the decision function of a Gaussian classifier we tie together the covariance of the different classes.
- In other words, we assume all classes have the same covariance:

$$\Sigma = \Sigma_{\kappa}, \text{ for } \kappa = 1, 2, \dots, K$$

- In that case:

$$\lambda = \arg \max_{\kappa} \left(\log(p(\Omega_{\kappa})) - \frac{1}{2} \log(|2\pi\Sigma|) - \frac{1}{2} (\vec{c} - \vec{\mu}_{\kappa})^T \Sigma^{-1} (\vec{c} - \vec{\mu}_{\kappa}) \right)$$

Term independent of maximizing parameter



Further Simplification

- The 3rd term also becomes simpler...

$$\begin{aligned}
 (\vec{c} - \vec{\mu}_k)^T \Sigma^{-1} (\vec{c} - \vec{\mu}_k) &= \cancel{\vec{c}^T \Sigma^{-1} \vec{c}} + \vec{\mu}_k^T \Sigma^{-1} \vec{\mu}_k - \vec{\mu}_k^T \Sigma^{-1} \vec{c} - \vec{c}^T \Sigma^{-1} \vec{\mu}_k
 \end{aligned}$$

Term independent of maximizing parameter.

Term constant for each class and independent of the input feature vector.

These two terms are linear in \vec{c} .

- If we look at the function we are trying to maximize, i.e. the a-posteriori probability

$$\begin{aligned}
 \lambda &= \arg \max_{\kappa} p(\Omega_{\kappa}) p(\vec{c} | \Omega_{\kappa}) \\
 &= \arg \max_{\kappa} \left(\log(p(\Omega_{\kappa})) - \frac{1}{2} (\vec{c} - \vec{\mu}_{\kappa})^T \Sigma^{-1} (\vec{c} - \vec{\mu}_{\kappa}) \right)
 \end{aligned}$$

we notice that it is linear in the components of \vec{c} .



General Form of Gaussian Decision Rule

- Thus, when the class conditional probabilities are normally distributed and have the same covariance Σ then the decision rule is of the form of a **linear** equation:

$$\lambda = \operatorname{argmax}_K \vec{a}_K^T \vec{c} + b_K$$

- Why compute the mean and variance and not directly recover the coefficients \vec{a}_K and b_K of the linear equation?
- One could do that, and then we have the more general case of directly computing linear decision boundaries.



Distinct Covariances

- If the covariance matrix of the class-conditional probabilities varies among classes

$$\Sigma_{\lambda} \neq \Sigma_{\kappa}, \text{ for } \kappa = 1, 2, \dots, K, \lambda = 1, 2, \dots, K \text{ and } \lambda \neq \kappa$$

then the term $\vec{c}^T \Sigma_{\kappa}^{-1} \vec{c}$ can not be ignored in the maximizing function:

$$\begin{aligned} (\vec{c} - \vec{\mu}_{\kappa})^T \Sigma_{\kappa}^{-1} (\vec{c} - \vec{\mu}_{\kappa}) &= \vec{c}^T \Sigma_{\kappa}^{-1} \vec{c} + \vec{\mu}_{\kappa}^T \Sigma_{\kappa}^{-1} \vec{\mu}_{\kappa} \\ &\quad - \vec{\mu}_{\kappa}^T \Sigma_{\kappa}^{-1} \vec{c} - \vec{c}^T \Sigma_{\kappa}^{-1} \vec{\mu}_{\kappa} \end{aligned}$$

- This means that the decision boundary is given by a **quadratic** function.



Summary

- Using a Gaussian classifier involves computing, for each class Ω_k : $p(\Omega_k)$, $\vec{\mu}_k$ and Σ_k .
- All these quantities can be estimated from the training data.
- These values are then used in the decision function:

$$\lambda = \arg \max_k \left(\log(p(\Omega_k)) - \frac{1}{2} \log(|2\pi\Sigma_k|) - \frac{1}{2} (\vec{c} - \vec{\mu}_k)^T \Sigma_k^{-1} (\vec{c} - \vec{\mu}_k) \right)$$

- The class that maximizes this function is then picked as the class of the feature vector \vec{c} .



Remarks

- There are two important issues that one should keep in mind.
 1. How good is my training data?
 2. Are the feature vectors within each class truly normally distributed?
 - Classify assuming the normal distribution assumption holds. If the system works, then the assumption was valid.
 - Apply statistical tests that verify whether the normal distribution assumption holds.
 - Select feature extraction methods like PCA that generate normally distributed features