Statistical Classifiers

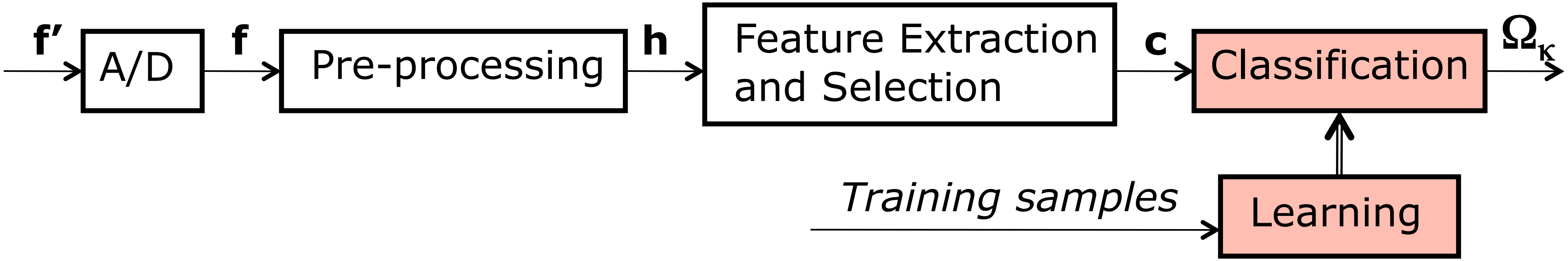
Gaussian Classifier

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 δ() 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 delta that minimizes the risk $R(\delta)$:

  $$\hat{\delta} = \arg\min_{\delta} R(\delta)$$

  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:

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

Gaussian Classifier

- It is a Bayesian classifier where we have normally distributed class-conditional feature vectors $p(\vec{c}\,|\,\Omega_\kappa)$.
- 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.

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

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

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

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

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:

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

where

$$\lambda = \underset{\kappa}{\arg\max}\, p(\Omega_\kappa)p(\vec{c}|\Omega_\kappa)$$

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

Decision Rule of a Gaussian Classifier - cont

Keep on expanding…

$$\lambda = \operatorname*{argmax}_{\kappa} p(\Omega_\kappa)\mathcal{N}(\vec{c}, \vec{\mu}_\kappa, \Sigma_\kappa)$$

$$\lambda = \operatorname*{argmax}_{\kappa} \log\left(p(\Omega_\kappa)\mathcal{N}(\vec{c}, \vec{\mu}_\kappa, \Sigma_\kappa)\right)$$

$$\lambda = \operatorname*{argmax}_{\kappa}\left(\log\left(p(\Omega_\kappa)\right) + \log\left(\mathcal{N}(\vec{c}, \vec{\mu}_\kappa, \Sigma_\kappa)\right)\right)$$

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

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

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

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:

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

- In that case:

Term independent of maximizing parameter

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

Further Simplification

- The 3rd term also becomes simpler…

$$(\vec{c}-\vec{\mu}_\kappa)^T\Sigma^{-1}(\vec{c}-\vec{\mu}_\kappa) = \vec{c}^T\Sigma^{-1}\vec{c}$$

Term independent of maximizing parameter.

$$+\ \vec{\mu}_\kappa^T\Sigma^{-1}\vec{\mu}_\kappa$$

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

$$-\ \vec{\mu}_\kappa^T\Sigma^{-1}\vec{c} - \vec{c}^T\Sigma^{-1}\vec{\mu}_\kappa$$

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

$$\lambda = \underset{\kappa}{\arg\max}\ p(\Omega_\kappa)\,p(\vec{c}\,|\,\Omega_\kappa)$$

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

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:

$$\lambda = \arg\max_{\kappa} \vec{a}_{\kappa}^{T} \vec{c} + b_{\kappa}$$

- Why compute the mean and variance and not directly recover the coefficients $\vec{a}_{\kappa}$ and b_{κ} 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

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

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

$$(\vec{c}-\vec{\mu}_\kappa)^T\Sigma_\kappa^{-1}(\vec{c}-\vec{\mu}_\kappa) = \vec{c}^{\,T}\Sigma_\kappa^{-1}\vec{c} + \vec{\mu}_\kappa^T\Sigma_\kappa^{-1}\vec{\mu}_\kappa - \vec{\mu}_\kappa^T\Sigma_\kappa^{-1}\vec{c} - \vec{c}^{\,T}\Sigma_\kappa^{-1}\vec{\mu}_\kappa$$

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

Summary

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

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

- 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