Statistical Classifiers

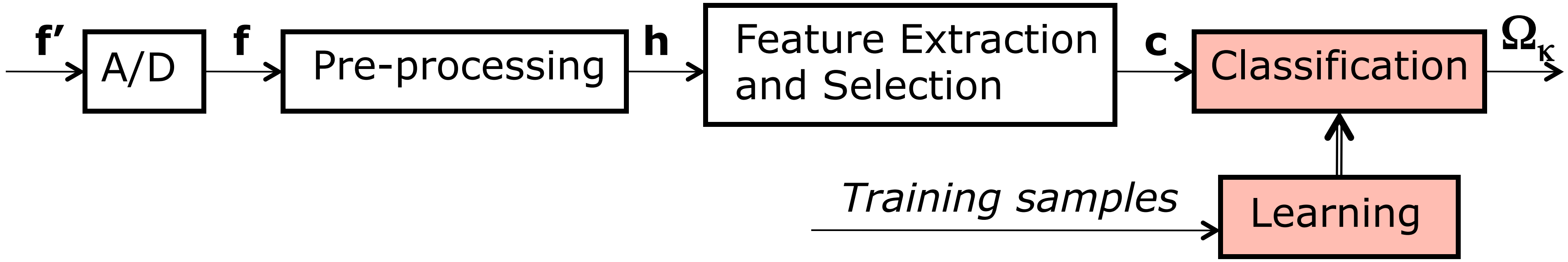
Bayesian Classifier

Dr. Elli Angelopoulou

Lehrstuhl für Mustererkennung (Informatik 5)

Friedrich-Alexander-Universität Erlangen-Nürnberg

Pattern Recognition Pipeline



- Feature Extraction and Selection
  - Heuristic feature extraction methods
  - Analytic feature extraction methods
  - Objective function for "goodness" of feature vector
  - Search method for exploring the feature space
- Classification
  - The step where the actual "recognition" takes place.
  - Assigns the transformed input signal to a class.
  - Labelled data can be critical in the recognition success.

Decision Function

- Goal: Map the computed feature vector $\vec{c}$ to a class Ω_κ.

$$\vec{c} \xrightarrow{\delta\left(\Omega_\kappa \middle| \vec{c}\right)} \Omega_\kappa$$

- The decision function $\delta()$ can be a probabilistic decision function.

$$\sum_{\kappa=1}^{K} \delta(\Omega_\kappa | \vec{c}) = 1$$

- Given a feature vector $\vec{c}$, there is a certain probability that we will decide that the observed signal belongs to a particular class.
- A probabilistic decision function expresses the fact that there is uncertainty in our decision making process.

Decision Function - continued

- Other times, the decision function is a binary function of the form:

$$\delta\left(\Omega_\kappa \middle| \vec{c}\right) = \begin{cases} 1 & \text{for } \Omega_\kappa \text{, if it is decided that } \vec{c} \in \Omega_\kappa \\ 0 & \text{for all other classes} \end{cases}$$

- In these cases the decision function can also be represented by a binary vector, with all zeroes, except the class to which the vector $\vec{c}$ belongs to.

Common Assumptions

- Very often during classification we make the following assumptions:

1. There exists a rejection class $\Omega_{0.}$
2. Each classification decision has individual costs associated with it. It is the cost of making a mistake.
3. After having classified a large number of samples, we are able to estimate the average costs, what we often refer to as the risk of the classification process.

Statistical Classifiers

- We have briefly seen classifiers that base their decision based on distances from a representative sample of each class (i.e. mean), or on decision boundaries.
- Statistical classifiers are based on the following idea:

1. Compute the risk associated with the classification of a pattern.
2. Compute the decision rule by minimizing the total risk.

- The final decision rule (that minimizes the risk) leads to the optimal classifier.

Statistics Review

- Mean vector (expectation):
  - Continuous: $E\{X\} = \int_{-\infty}^{\infty} x p(x) dx$
  - Discrete: $\mu_X = \frac{1}{N} \sum_{i=1}^{N} x_i$
- Variance of scalar random variable
  - Continuous: $Var\{X\} = E\left\{\left(X - E\{X\}\right)^2\right\}$
  - Discrete: $\sigma_X^2 = \frac{1}{N} \sum_{i=1}^{N} \left(x_i - \mu_X\right)^2$
- Variance of vector data (a.k.a. covariance matrix, or variance-covariance matrix, or dispersion matrix)

$$Var\{\vec{X}\} = E\left\{\left(\vec{X} - E\{\vec{X}\}\right)\left(\vec{X} - E\{\vec{X}\}\right)^T\right\}$$

Parametric Densities

- Parametric density functions are densities that are completely defined by their parameters.
- For example, in a normal distribution, the pdf is completely described by the mean and the variance.
- In general, parametric density functions are of the form:

  $$\vec{c} \approx p(\vec{c} \mid \vec{\alpha})$$

  where $\vec{\alpha}$ is a parameter vector that has to be estimated.
- Example: Normally distributed feature vectors

  $$\vec{c} \approx \mathscr{N}(\vec{c}, \vec{\mu}, \Sigma)$$

  where the parameters $\vec{\mu}, \Sigma$ can be estimated via maximum likelihood estimation.

Classification Risk – a first look

- Recall that statistical classifiers are based on the following 2-step process:

1. Compute the risk associated with the classification of a pattern.
2. Compute the decision rule by minimizing the total risk.

- We need a way of quantifying the risk associated with a classifier.
- For that we need to first establish a cost for each classification decision.

Cost Function

- Let $r_{\lambda,\kappa} \in R$ denote the **cost** for classifying a pattern as belonging to class Ω_λ when it truly belongs to class Ω_κ.
- The **individual decision cost** $r_{\lambda,\kappa}$ has to be defined by the user of the classifier.
- A cost function (usually) should fulfill the following inequality:

$$0 \leq r_{\kappa,\kappa} \leq r_{\lambda,\kappa}$$

  where $r_{\kappa,\kappa}$ is the correct decision.
- In the presence of a rejection class Ω_0:

$$r_{\kappa,\kappa} \leq r_{0,\kappa} \leq r_{\lambda,\kappa}$$

Computing the Optimal Decision Rule

- In order to compute the optimal decision rule we need to perform the following steps:

1. Compute the probability of misclassification

$$p\left(\Omega_\lambda \middle| \Omega_\kappa\right)$$

2. Compute the **risk** *R*(δ) associated with using a **particular decision function** δ(), including correct decisions, as well as misclassifications:

$$R(\delta) = \sum_{\forall \kappa, \lambda} p(\Omega_\kappa) p\left(\Omega_\lambda \middle| \Omega_\kappa\right) r_{\lambda,\kappa}$$

3. Minimize the risk over all different decision rules

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

Computing the Prob. of Misclassification

- We want to compute the probability of misclassifying a signal as belonging to class Ω_λ when it truly belongs to class Ω_κ, $p\left(\Omega_\lambda \middle| \Omega_\kappa\right)$.
- By the definition of conditional probabilities:

$$p\left(A \middle| B\right) = p(A,B)/p(B)$$

- Given two jointly distributed random variables A and B, the marginal distribution of A is simply the probability distribution of A ignoring information about B. It is typically calculated by integrating the joint probability distribution over B:

$$\text{Marginal } p(A) = \int_B p(A,b)\,pb$$

Computing the Prob. of Misclassification (2)

- Given these facts, one can derive the probability of misclassification by starting with the conditional probability and doing a marginalization over $\vec{c}$.

$$p\left(\Omega_\lambda, \vec{c} \middle| \Omega_\kappa\right) = \frac{p(\Omega_\lambda, \vec{c}, \Omega_\kappa)}{p(\Omega_\kappa)}$$

$$= \frac{p(\Omega_\lambda, \vec{c}, \Omega_\kappa)}{p(\Omega_\kappa)} \frac{p(\vec{c}, \Omega_\kappa)}{p(\vec{c}, \Omega_\kappa)}$$

$$= \frac{p(\Omega_\lambda, \vec{c}, \Omega_\kappa)}{p(\vec{c}, \Omega_\kappa)} \frac{p(\vec{c}, \Omega_\kappa)}{p(\Omega_\kappa)}$$

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

Computing the Prob. of Misclassification (3)

- We have shown that $p(\Omega_\lambda, \vec{c} \mid \Omega_\kappa) = p(\Omega_\lambda \mid \vec{c}, \Omega_\kappa)\, p(\vec{c} \mid \Omega_\kappa)$
- However what we observe is just the feature vector $\vec{c}$ and not both $\vec{c}$ and Ω_κ.
- So we replace this term with a probabilistic decision for class Ω_λ, given that we have observed $\vec{c}$:

$$p(\Omega_\lambda, \vec{c} \mid \Omega_\kappa) = \delta(\Omega_\lambda \mid \vec{c})\, p(\vec{c} \mid \Omega_\kappa)$$

- We can now do a marginalization over $\vec{c}$:

$$p(\Omega_\lambda \mid \Omega_\kappa) = \int_{R_{\vec{c}}} \delta(\Omega_\lambda \mid \vec{c})\, p(\vec{c} \mid \Omega_\kappa)\, d\vec{c}$$

Computing the Optimal Decision Rule - revisit

- In order to compute the optimal decision rule we need to perform the following steps:

1. Compute the probability of misclassification

$$p(\Omega_\lambda | \Omega_\kappa) = \int_{R_{\vec{c}}} \delta(\Omega_\lambda | \vec{c})\, p(\vec{c} | \Omega_\kappa)\, d\vec{c}$$

2. Compute the **risk** *R*(δ) associated with using a **particular decision function** δ(), including correct decisions, as well as misclassifications:

$$R(\delta) = \sum_{\forall \kappa,\lambda} p(\Omega_\kappa)\, p\left(\Omega_\lambda | \Omega_\kappa\right) r_{\lambda,\kappa}$$

3. Minimize the risk over all different decision rules

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

Computing the Risk of a Decision Function

- The risk $R(\delta)$ associated with using a particular decision function $\delta()$ for a specific class Ω_κ is:

$$R(\delta|\Omega_\kappa) = \sum_{\lambda=0}^{K} p\left(\Omega_\lambda|\Omega_\kappa\right) r_{\lambda,\kappa}$$

$$= \sum_{\lambda=0}^{K} \int_{R_{\vec{c}}} \delta(\Omega_\lambda|\vec{c})\, p(\vec{c}|\Omega_\kappa)\, d\vec{c}\, r_{\lambda,\kappa}$$

- For the overall risk, we have to sum over all the classes, taking under consideration the probability of occurrence of each class.

$$R(\delta) = \sum_{\kappa=1}^{K} R(\delta|\Omega_\kappa) p(\Omega_\kappa) = \int_{R_{\vec{c}}} \sum_{\lambda=0}^{K} \underbrace{\boxed{\sum_{\kappa=1}^{K} r_{\lambda,\kappa} p(\Omega_\kappa) p(\vec{c}|\Omega_\kappa)}}_{u_\lambda(\vec{c})} \delta(\Omega_\lambda|\vec{c})\, d\vec{c}$$

$u_\lambda(\vec{c}) \leftarrow$ measurement value

Objective Function

- The overall risk *R*(δ) can then be written more compactly as:

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

- Goal: Derive an optimal decision rule which minimizes overall risk:

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

- Conclusion: The optimal classifier will decide for the class that leads to the smallest measurement value $u_\lambda(\vec{c})$.

Optimal Decision Rule

- Let $u_{min}(\vec{c})$ be the smallest possible measurement value among all possible classes.

$$u_{min}(\vec{c}) = \min_{\lambda} u_{\lambda}(\vec{c})$$

- Then, the optimal decision rule is:

$$\boxed{\delta(\Omega_{\lambda}|\vec{c}) = \begin{cases} 1 & \text{if } u_{\lambda}(\vec{c}) = u_{min}(\vec{c}) \\ 0 & \text{otherwise} \end{cases}}$$

A Remark on the Measurement Value

- The computation of $u_\lambda(\vec{c})$ can be done by a vector product calculation:

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

$$= \underbrace{[r_{\lambda,1}, r_{\lambda,2}, \ldots, r_{\lambda,K}]}_{\boxed{\text{independent of } \vec{c}}} \begin{bmatrix} p(\Omega_1)\, p(\vec{c}\,|\,\Omega_1) \\ p(\Omega_2)\, p(\vec{c}\,|\,\Omega_2) \\ \vdots \\ p(\Omega_K)\, p(\vec{c}\,|\,\Omega_K) \end{bmatrix}$$

Cost Functions

- So far we have considered the user-defined cost function $r_{\lambda,\kappa}$, where $\lambda = 0,1,2,\ldots,K$ and $\kappa = 1,2,\ldots,K$ and where *K* is the number of classes. So the user must specify (*K*+1)*K* different cost values.
- A simpler cost setup involves just 3 distinct cost functions:

$$r_{\kappa,\kappa} = r_c \ \forall\kappa \quad \text{(correct classification)}$$

$$r_{0,\kappa} = r_r \ \forall\kappa \quad \text{(reject)}$$

$$r_{\lambda,\kappa} = r_f \ \forall\kappa \neq \lambda \quad \text{(false classification)}$$

- So one can also think of the total cost of a decision function as:

$$R(\delta) = p_c r_c + p_f r_f + p_r r_r$$

(0,1)-Cost Function

- A special case of cost function is the (0,1)-cost function which:
  - uses no rejection class
  - has an $r_{\kappa,\kappa} = r_c = 0 \; \forall \kappa$ correct decision cost
  - has an $r_{\lambda,\kappa} = r_f = 1 \; \forall \kappa \neq \lambda$ false decision cost
- The risk function for the (0,1) cost function is a simplified version of the general $R(\delta)$:

$$R(\delta) = p_c r_c + p_f r_f + p_r r_r = p_f$$

- Thus, a classifier that minimizes the risk for a (0,1)-cost function is equivalent to the classifier that minimizes the error probability.

Decision rule of a (0,1)-Cost Function

- Using a (0,1)-cost function simplifies the measurement value:

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

- Recall that the optimal decision rule is:

$$\delta(\Omega_\lambda|\vec{c}) = \begin{cases} 1 & \text{if } u_\lambda(\vec{c}) = u_{min}(\vec{c}) = \min\limits_{\kappa} u_\kappa(\vec{c}) \\ 0 & \text{otherwise} \end{cases}$$

- Notice that $u_\lambda(\vec{c})$ is minimal when the largest summand is left out, i.e. when the class Ω_κ with the largest $p(\Omega_\kappa)\, p(\vec{c}\,|\,\Omega_\kappa)$ product is not included in the summation.

Measurement Value of a (0,1)-Cost Function

- More specifically, minimizing $u_\lambda(\vec{c})$ for a (0,1)-cost function involves:

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

- But the sum is minimal when the largest summand is left out. The largest term of the sum is realized for the class with the largest $p(\Omega_\kappa)p(\vec{c}\,|\,\Omega_\kappa)$ term.
- How can we exclude this from the sum?
- Assign $\vec{c}$ to the class with the largest $p(\Omega_\kappa)p(\vec{c}\,|\,\Omega_\kappa)$ term. Then through the $\kappa \neq \lambda$ condition the term is excluded from the sum.

Largest Summand Example

- Here is a simple example that demonstrates why selecting the parameter (class) that minimizes the sum is equivalent to selecting the parameter (class) that gives the largest summand.
- We want to find

$$u_{\min}(\vec{c}) = \min_{\lambda} u_{\lambda}(\vec{c}) = \min_{\lambda} \sum_{\substack{\kappa=1 \\ \kappa\neq\lambda}}^{K} p(\Omega_{\kappa}) p(\vec{c}\,|\,\Omega_{\kappa}) = \min_{\lambda} \sum_{\substack{\kappa=1 \\ \kappa\neq\lambda}}^{K} f_k$$

where $f_k = p(\Omega_{\kappa}) p(\vec{c}\,|\,\Omega_{\kappa})$ is used for a more compact presentation.

- Consider the example where $K = 5$ and $f_1 = 0.15$

$f_2 = 0.67$ $f_3 = 0.04$ $f_4 = 0.12$ $f_5 = 0.02$

Largest Summand Example - continued

- Our goal is to find the λ that minimizes

$$u_{\min}(\vec{c}) = \min_{\lambda} u_{\lambda}(\vec{c}) = \min_{\lambda} \sum_{\substack{\kappa=1 \\ \kappa\neq\lambda}}^{K} f_k$$

- Recall: $f_1 = 0.15\,, f_2 = 0.67, f_3 = 0.04\,, f_4 = 0.12\,, f_5 = 0.02$
- For $\lambda = 1$: $u_1(\vec{c}) = \sum_{\substack{\kappa=1 \\ \kappa\neq 1}}^{K} f_k = 0.67 + 0.04 + 0.12 + 0.02 = 0.85$
- For $\lambda = 2$: $u_2(\vec{c}) = \sum_{\substack{\kappa=1 \\ \kappa\neq 2}}^{K} f_k = 0.15 + 0.04 + 0.12 + 0.02 = 0.33$
- For $\lambda = 3$: $u_3(\vec{c}) = \sum_{\substack{\kappa=1 \\ \kappa\neq 3}}^{K} f_k = 0.15 + 0.67 + 0.12 + 0.02 = 0.96$

Largest Summand Example - continued

- Our goal is to find the λ that minimizes

$$u_{\min}(\vec{c}) = \min_{\lambda} u_{\lambda}(\vec{c}) = \min_{\lambda} \sum_{\substack{\kappa=1 \\ \kappa\neq\lambda}}^{K} f_k$$

- Recall: $f_1 = 0.15$, $f_2 = 0.67$, $f_3 = 0.04$, $f_4 = 0.12$, $f_5 = 0.02$
- For $\lambda = 4$: $u_4(\vec{c}) = \sum_{\substack{\kappa=1 \\ \kappa\neq 4}}^{K} f_k = 0.15 + 0.67 + 0.04 + 0.02 = 0.88$
- For $\lambda = 5$: $u_5(\vec{c}) = \sum_{\substack{\kappa=1 \\ \kappa\neq 5}}^{K} f_k = 0.15 + 0.67 + 0.04 + 0.12 = 0.98$
- Thus: $\min_{\lambda} u_{\lambda}(\vec{c}) = \min_{\lambda}\left(0.85, 0.33, 0.96, 0.88, 0.98\right) = 0.33, \text{ for } \lambda = 2$ because $\lambda = 2$ gave the smallest sum by excluding the biggest summand $f_2 = 0.67$.

Minimizing the Measurement Value – (0,1) cost

- The class that minimizes measurement value then is:

$$\eta = \arg\min_{\lambda} u_\lambda(\vec{c}) = \arg\min_{\lambda} \sum_{\substack{\kappa=1 \\ \kappa\neq\lambda}}^{K} p(\Omega_\kappa) p(\vec{c}\,|\,\Omega_\kappa)$$

Selecting the largest summand

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

Dividing with a term independent of the maximizing argument λ

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

Using the Bayesian rule

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

a-posteriori probability

Optimal Decision Rule Revisited

- So, given a feature vector $\vec{c}$ we compute for each class the a-posteriori probability and decide for the class with the largest probability.
- Lemma: The classifier that minimizes the probability for misclassification (minimizes p_f) applies the following decision rule:

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

Bayesian decision rule

Bayesian Decision Rule

- The Bayes decision rule is a very important result in pattern recognition.
- It states that if we want to have a classification scheme that minimizes the probability of misclassifications, then the only thing one needs to do is to:
  a. Compute the posterior probabilities $p(\Omega_\kappa | \vec{c})$
  b. Decide for the class that give the maximum posterior probability.
- Simple concept:

  Finding the optimal classifier requires finding the posterior probabilities.

Bayesian Classifier

- Definition: A classifier whose decision rule is based on the maximization of posterior probabilities is called a *Bayesian classifier*.
- So pattern recognition is then done/solved in terms of classification.
- All we need to do is given some training data to compute the posterior probability $p(\Omega_\kappa | \vec{c})$.
- A simple task. Or is it?

Bayesian Classifier

- Obtaining accurate estimates of the posterior probabilities from training data can be challenging.
- One of the topics of Pattern Recognition is to find good methodologies for approximating the posterior probabilities.
- So in theory, there is no other classifier that can achieve a lower error probability than a **(0,1)-Bayesian classifier**. Let us denote the error probability of a Bayesian classifier as p_B.
- In general, this error probability p_B will act as a lower bound when discussing the error probabilities of other classifiers.

Remarks

1. Many classifiers try to approximate the Bayesian classifier.

   Caution: a (0,1)-cost function must make sense, do not force a (0,1)-cost function if it doesn't fit the application.

2. The Bayesian classifier requires complete knowledge about $p(\Omega_\kappa | \vec{c})$.

   How do we get enough training data?

   Is the training data appropriate? In other words are our samples good examples of the real population?

Remarks - continued

3. Modeling of $p(\Omega_\kappa | \vec{c})$ is a key issue.

   For instance:

   In speech recognition we don't classify based on a single feature but rather on a sequence of features. How do we handle feature sequences in the posterior probability computation?

   How do we deal with the fact that the image data we get is a projection from 3D to 2D, i.e. we already have information loss?