Analytic Feature Extraction Methods

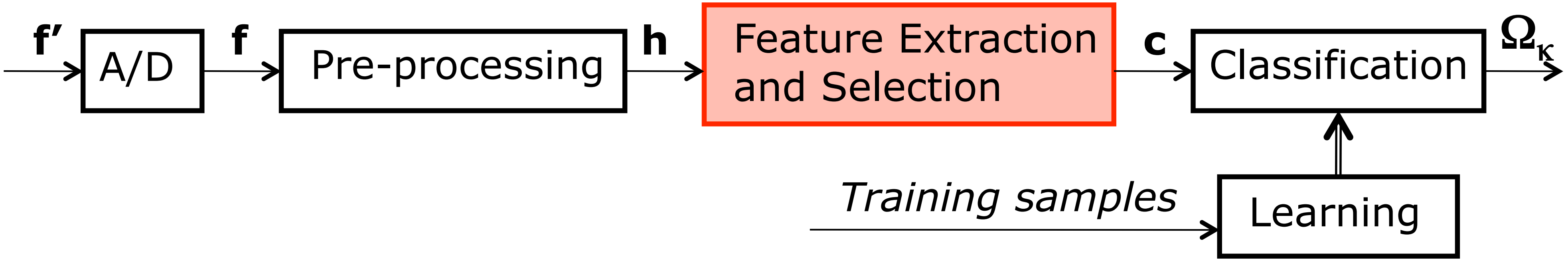
Optimal Feature Transform

Dr. Elli Angelopoulou

Lehrstuhl für Mustererkennung (Informatik 5)

Friedrich-Alexander-Universität Erlangen-Nürnberg

Pattern Recognition Pipeline



- Heuristic feature extraction methods
- Analytic feature extraction methods
  - Principal Component Analysis (PCA)
  - Minimal Intra-class Distance
  - Maximal Inter-class Distance
  - Linear Discriminant Analysis (LDA)
  - Optimal Feature Transform

Analytic Methods for Feature Computation

- Analytic feature extraction methods derive a linear transformation Φ that satisfies a specific optimality criterion.

$$\vec{c} = \Phi\vec{f}$$

- So far we have seen optimality criteria that are related to the postulates of pattern recognition:
  - Finding principal components that can explain the variability of the data.
  - Tight clusters for each class.
  - Distinct clusters for different classes.
- What about an optimality criterion that is directly related to the goal of pattern recognition itself: Good recognition (classification) rates

Optimal Feature Transform

- There exists an analytic feature extraction method whose goal is to minimize the number of misclassifications.
- Alternatively one can think of the dual problem which is maximizing the number of correct classifications.
- The resulting features are then optimal for the overall goal of pattern recognition.
- Thus, such a feature extraction method is called an **Optimal Feature Transform (OFT)**.

Optimality Criterion of OFT

- The goal of OFT is to derive a transformation matrix Φ that minimizes misclassifications.
- Expressing this goal mathematically requires us to precisely define misclassification.
- This implies that we have to set up the basics for describing classification itself.
- It is a long derivation, so keep in mind that at the end we want to derive an optimization function

$$s_6(\Phi) = \ldots$$

that describes misclassifications.

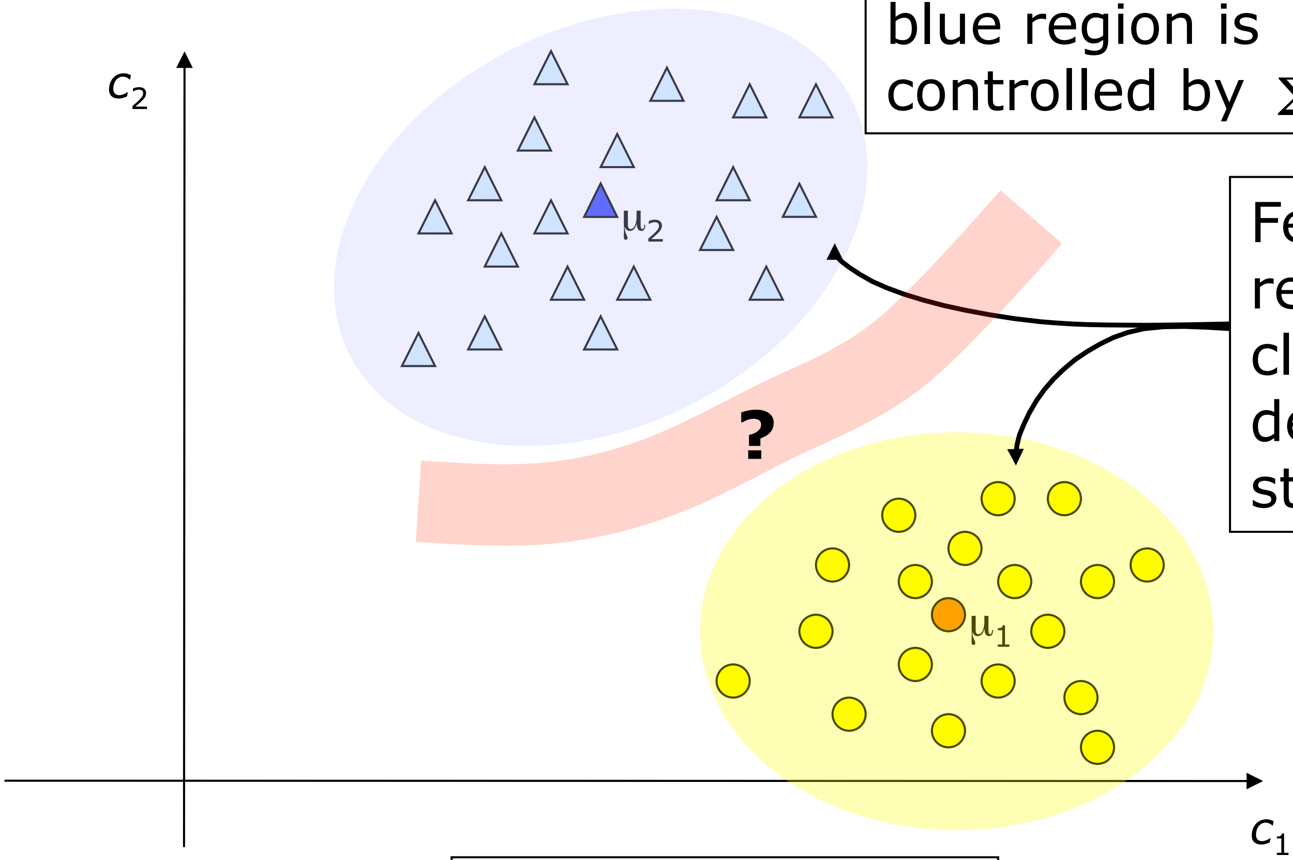
Gaussian Distributed Features

- We can **not** design a feature transform that will be optimal for **any possible** input signal.
- Rather we design optimal feature transformations **for particular cases.**
- So, let's look at one such particular case.
- <u>Special case:</u> Features are normally distributed, i.e. the probability density function of $\vec{c}$ is a Gaussian

$$\vec{c} \approx \mathcal{N}(\vec{c}, \vec{\mu}_\kappa, \Sigma_\kappa) = \frac{1}{\sqrt{2\pi |\Sigma_\kappa|}} e^{-(\vec{c}-\vec{\mu}_\kappa)^T \Sigma_\kappa^{-1} (\vec{c}-\vec{\mu}_\kappa)}$$

where $\mathcal{N}$ is a Gaussian distribution with amplitude $\vec{c}$, mean $\vec{\mu}_\kappa$ and variance Σ_κ.

Different Decision Regions



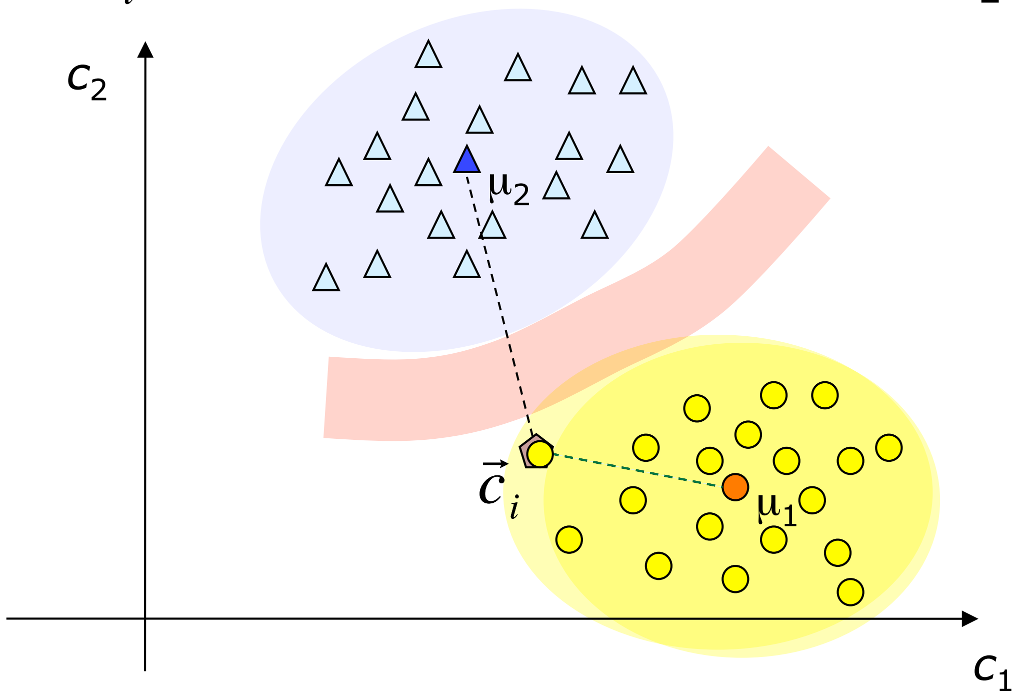
The shape of the blue region is controlled by Σ_2

Feature space region where classification decision is straightforward

The shape of the yellow region is controlled by Σ_1

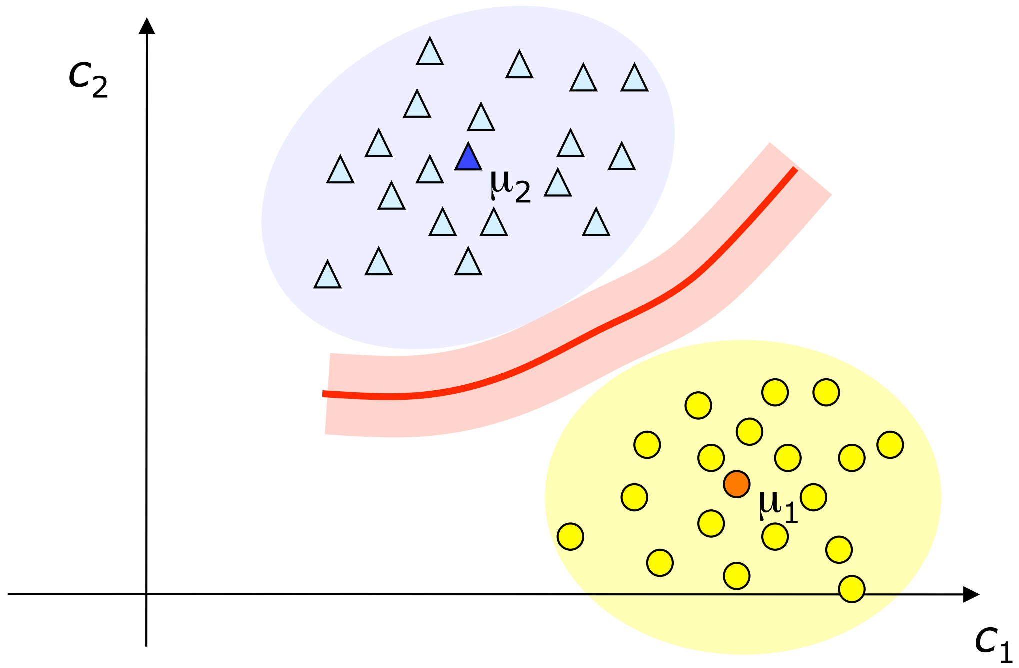
Distance Function

- Consider a function u() which is a measure of how far a point in feature space is from the center of a cluster.
  - $u_1()$ is a distance measure to the center of cluster 1.
  - $u_2()$ is a distance measure to the center of cluster 2.
- If for a specific feature vector $\vec{c}_i$, $u_1(\vec{c}_i) < u_2(\vec{c}_i)$ then we classify $\vec{c}_i$ as belonging to class Ω_1.



Decision Boundary

- There is a region, where it is ambiguous whether the data belongs to class 1, Ω_1, or class 2, Ω_2.
- This region is called the *decision boundary*.
- It is the area where $u_1() = u_2()$.
- It is the where we are most probable to have misclassifications for both classes.



OFT and Decision Boundary

- Recall that the goal of OFT is to derive a transformation matrix Φ that minimizes misclassifications.
- We also know that the misclassifications will most probably occur at the decision boundary ($u_1() = u_2()$).
- So we have to focus our derivation of the optimization function for the computation of Φ on the decision boundary and the distance functions.
- Assuming that the feature vectors within each class are normally distributed, an appropriate distance function is:

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

Mahalanobis distance

Decision Boundary Manifold

- The decision boundaries are the manifolds where the points belonging to them are equidistant to different class centers:

$$H_{\kappa\lambda} = \left\{ \vec{c} \,\middle|\, u_\kappa\left(\vec{c}\right) = u_\lambda\left(\vec{c}\right) \right\}$$

  where $H_{\kappa\lambda}$ is the decision boundary between classes Ω_κ and Ω_λ.
- What does the shape of $H_{\kappa\lambda}$ look like?
  - Straight line?
  - Section of a Circle?
  - Section of an Ellipse?
  - …
- To answer that we must look at the distance function.

Shape of the Decision Boundary

- At the decision boundary $u_\kappa(\vec{c}) = u_\lambda(\vec{c})$
- Using the Mahalanobis distance metric

$$u_\kappa(\vec{c}) = u_\lambda(\vec{c}) \Leftrightarrow (\vec{c} - \vec{\mu}_\kappa)^T \Sigma_\kappa^{-1} (\vec{c} - \vec{\mu}_\kappa) = (\vec{c} - \vec{\mu}_\lambda)^T \Sigma_\lambda^{-1} (\vec{c} - \vec{\mu}_\lambda)$$

where $\vec{\mu}_i$ and Σ_i are constants for each class Ω_i.

- This equation shows that, for classes whose features follow a Gaussian distribution, $H_{\kappa\lambda}$ is quadratic in the components of the vector $\vec{c}$.
- This means that in a 2D feature space $H_{\kappa\lambda}$ will look like a parabola.

On the Mahalanobis Distance

- Consider the case where all the feature vectors that belong to class Ω_κ are equidistant from the mean value of that class, $\vec{\mu}_\kappa$:

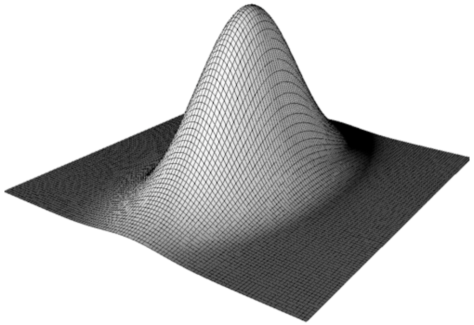
$$u_\kappa(\vec{c}) = \alpha, \quad \forall \vec{c} \in \Omega_\kappa$$

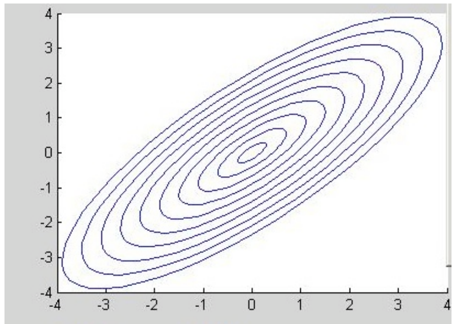
where α is a constant.

- Plot such a distribution.
- If $u_\kappa()$ is the Euclidean distance, then we get a circle of radius α which is centered around $\vec{\mu}_\kappa$.
- Looking at the definition of the Mahalanobis distance, $u_\kappa(\vec{c}) = (\vec{c} - \vec{\mu}_\kappa)^T \Sigma_\kappa^{-1} (\vec{c} - \vec{\mu}_\kappa)$, we get a circle only when the variance matrix is the identity $\Sigma_\kappa = I$.

On the Mahalanobis Distance – cont.

- In general, the (co-)variance matrix is not the identity matrix I, $\Sigma_\kappa \neq I$.
- In 2D think of a Gaussian with independent standard deviations in each of the two axes, $\sigma_x \neq \sigma_y$. What one gets is an oblong 3D bell shape.



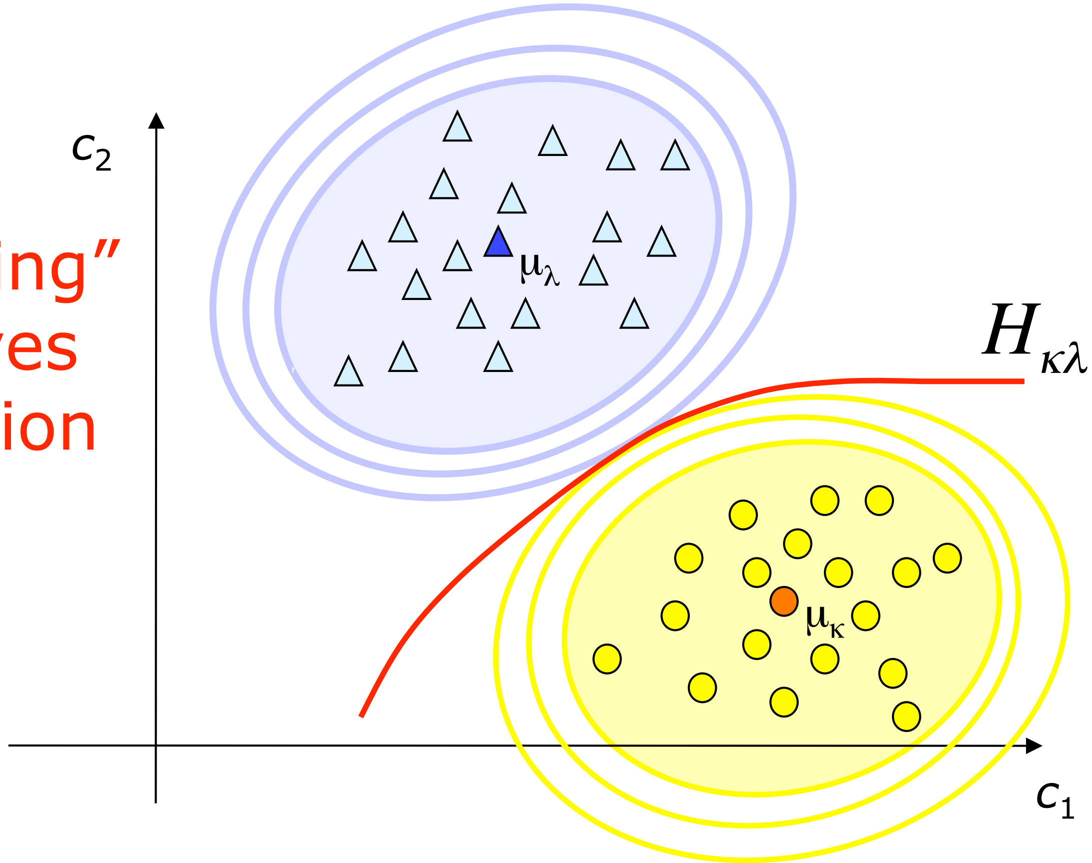


- If we consider a set of feature points $\vec{c}$ that are equidistant to the class mean $\vec{\mu}_\kappa$, i.e. $\mathrm{u}_\kappa(\vec{c}) = \alpha$, For this general case, we get an ellipsoid.
- Thus $H_{\kappa\lambda}$ is an ellipsoid.

Ellipsoids and Classification

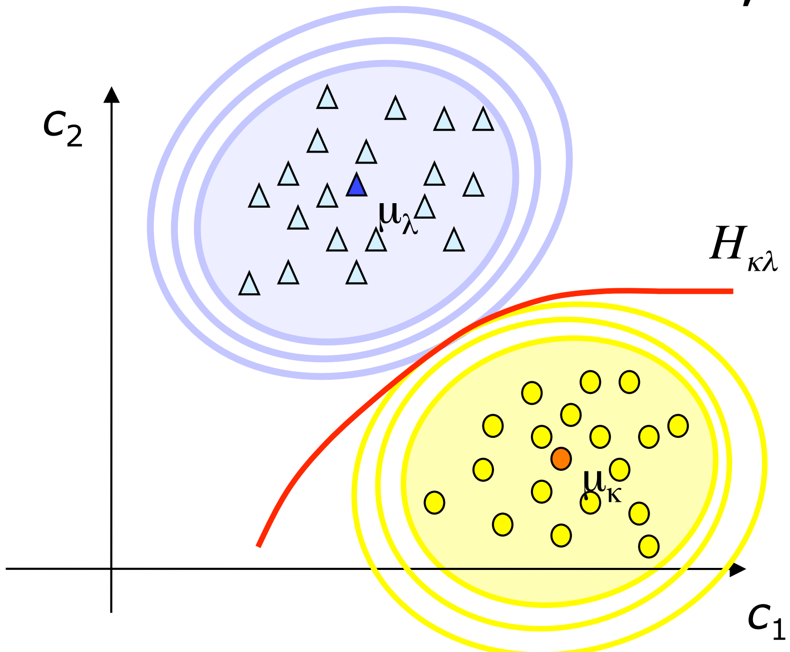
- There is an ellipsoid in class Ω_κ that just touches the decision boundary $H_{\kappa\lambda}$. There is an ellipsoid in class Ω_λ that just touches the decision boundary $H_{\kappa\lambda}$.

This "touching" ellipsoid gives a classification guarantee.



Ellipsoids and Classification - continued

- Consider the maximal ellipsoid for class Ω_κ that still completely lies on the Ω_κ side of the decision boundary $H_{\kappa\lambda}$.
- For all the points inside that ellipsoid $\mathrm{u}_\kappa(\vec{c}) < \mathrm{u}_\lambda(\vec{c})$.
- So as long as we stay within the ellipsoid, there is no ambiguity about our classification decision, there is no misclassification.



OFT and Ellipsoids

- The goal of OFT is to derive a transformation matrix Φ that minimizes misclassifications.
- Find a Φ that transforms the input signal $\vec{f}$ to a feature vector $\vec{c}$ so that the radius of the "touching" ellipsoid (this "guarantee" ellipsoid) is maximal.
- In that way we will have the largest possible region in the feature space where we will be getting correct classifications.
- Still missing: A mathematical definition of the touching ellipsoid.
- Keep in mind that there may be more than 2 classes.

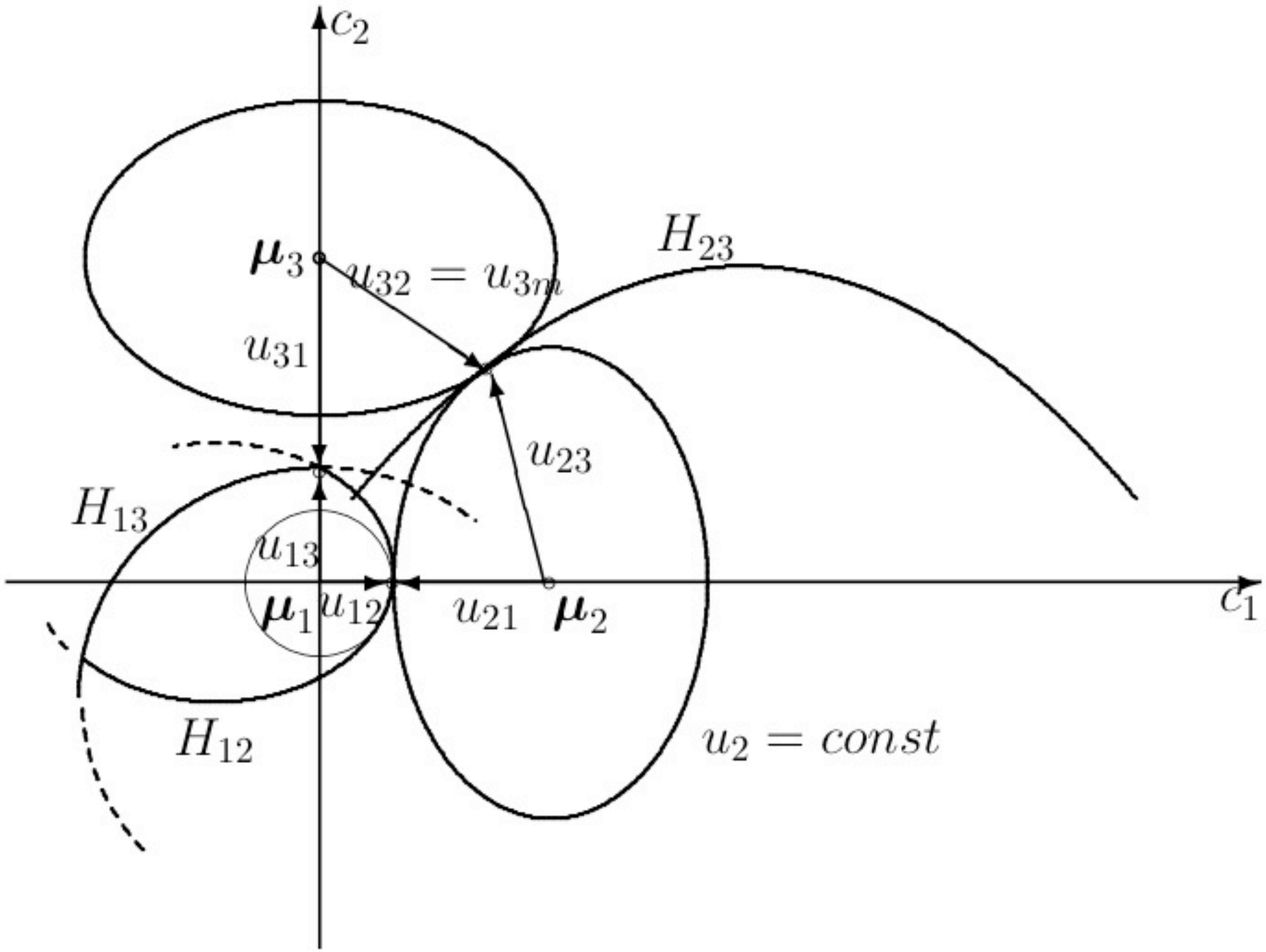
Guarantee Ellipsoid and Decision Boundary

- Let $u_{\kappa\lambda}$ be the minimum distance of a feature vector $\vec{c}$ on the decision boundary, $\vec{c} \in H_{\kappa\lambda}$, to the mean value of class Ω_κ :

$$u_{\kappa\lambda} = \min_{\vec{c} \in H_{\kappa\lambda}} u_\kappa(\vec{c})$$

- In other words, we walk on the decision boundary. We compute $u_\kappa(\vec{c})$ for each point on the decision boundary $H_{\kappa\lambda}$. For one such point $u_\kappa(\vec{c})$ will be minimal. This "minimal" point is where the "guarantee" ellipse of class Ω_κ touches the boundary.
- We can have more than 2 classes. So we get a decision boundary $H_{\alpha\beta}$ for every pair of classes Ω_α and Ω_β. For each $H_{\alpha\beta}$ we get a $u_{\alpha\beta}$.

Multiclass Decision Boundaries



Using the Guarantee Ellipsoids

- As long as we are inside a “guarantee” ellipse, we have ideally no misclassifications.
- In a multiclass setup, we will possibly end up with intersecting ellipses.
- In order to preserve the “no misclassification property” of the guarantee ellipse, we must avoid intersections that result from the different decision boundaries.
- Thus, we must be conservative. For each particular class Ω_κ we must examine each decision boundary with that class, $H_{\kappa\alpha}, H_{\kappa\beta}, H_{\kappa\gamma}, \ldots$, and pick the ellipse that is closest to the mean of the cluster.

Using the Guarantee Ellipsoids - continued

- For each particular class Ω_κ we must examine each decision boundary with that class, $H_{\kappa\alpha}, H_{\kappa\beta}, H_{\kappa\gamma}, \ldots$, and pick the ellipse that is closest to the mean of the cluster.
- We can use the minimal distance to find such an ellipse: $u_{\kappa_m} = \min_{\kappa \neq \lambda} u_{\kappa\lambda}$
- A pattern will be correctly classified if the feature vector $\vec{c}$ lies inside the ellipsoid with radius u_{κ_m} .
- For each class Ω_κ we get a radius that ensures correct separation of the classes Ω_κ and Ω_λ. To be able to separate **all** classes, we take the smallest radius among all classes Ω_λ .

Probability of Misclassification

- What happens outside the ellipse?
- There may still be points outside the conservative ellipse that belong to class Ω_κ but get mistakenly classified as belonging to another class.
- What is the probability of my making this mistake?

$$p_{f_\kappa}(\vec{c}) \leq p\left(u_{\kappa_m} < u_\kappa(\vec{c})\right)$$

- So for the overall error probability, for all the classes is the sum weighted by the probability of the class occurring:

$$p_{err} = \sum_{\kappa=1}^{K} p(\Omega_\kappa) p_{f_\kappa}(\vec{c}) \leq \sum_{\kappa=1}^{K} p(\Omega_\kappa) p\left(u_{\kappa_m} < u_\kappa(\vec{c})\right)$$

Probability of Misclassification- continued

- So for the overall error probability, for all the classes is the sum weighted by the probability of the class occurring:

$$p_{err} = \sum_{\kappa=1}^{K} p(\Omega_\kappa) p_{f_\kappa}(\vec{c}) \leq \sum_{\kappa=1}^{K} p(\Omega_\kappa) p\left(u_{\kappa_m} < u_\kappa(\vec{c})\right)$$

- Use Chebyshev's inequality:

$$p\left(u_{\kappa_m} < u_\kappa(\vec{c})\right) < \frac{M}{u_{\kappa_m}} \quad , \quad \text{where } M = \dim(\vec{c})$$

- The objective function for the OFT becomes:

$$s_6(\Phi) = p_{err} = \sum_{\kappa=1}^{K} p(\Omega_\kappa) \frac{M}{u_{\kappa_m}}$$

Linear Transformations in Feature Space

- What happens if we apply a linear transformation to the feature vector $\vec{c}$?
- Consider for example the case, where $\vec{c}\,'$ is related to vector $\vec{c}$ by an invertible linear transformation *B*:

$$\vec{c}\,' = B\vec{c}$$

- Are the mean values of vectors $\vec{c}$ and $\vec{c}\,'$ related?

$$\vec{\mu}_\kappa = \mathrm{E}\{\vec{c}\}$$
$$\vec{\mu}\,'_\kappa = \mathrm{E}\{B\vec{c}\} = B\,\mathrm{E}\{\vec{c}\} = B\vec{\mu}_\kappa$$

- So the new expected value is just the original expected value transformed by *B*.

Linear Transformations in Feature Space 2

- Are the covariances of vectors $\vec{c}$ and $\vec{c}\,'$ related?

$$
\begin{aligned}
\Sigma_\kappa &= \mathrm{E}\left\{\left(\vec{c}-\vec{\mu}_\kappa\right)\left(\vec{c}-\vec{\mu}_\kappa\right)^T\right\} \\
\Sigma'_\kappa &= \mathrm{E}\left\{\left(\vec{c}\,'-\vec{\mu}'_\kappa\right)\left(\vec{c}\,'-\vec{\mu}'_\kappa\right)^T\right\} \\
&= \mathrm{E}\left\{\left(B\vec{c}-B\vec{\mu}_\kappa\right)\left(B\vec{c}-B\vec{\mu}_\kappa\right)^T\right\} \\
&= \mathrm{E}\left\{B\left(\vec{c}-\vec{\mu}_\kappa\right)\left(\vec{c}-\vec{\mu}_\kappa\right)^T B^T\right\} \\
&= B\,\mathrm{E}\left\{\left(\vec{c}-\vec{\mu}_\kappa\right)\left(\vec{c}-\vec{\mu}_\kappa\right)^T\right\}B^T \\
&= B\Sigma_\kappa B^T
\end{aligned}
$$

- The covariance of the linearly transformed vector is linearly related to the covariance of the original vector.

Invariance of the Mahalanobis Distance

- How is the Mahalanobis distance of the transformed vector $\vec{c}\,'$ affected?

$$
\begin{aligned}
u'_\kappa(\vec{c}\,') &= \left(\vec{c}\,' - \vec{\mu}'_\kappa\right)^T \Sigma_\kappa'^{-1} \left(\vec{c}\,' - \vec{\mu}'_\kappa\right) \\
&= \left(B\vec{c} - B\vec{\mu}_\kappa\right)^T \left(B\Sigma_\kappa B^T\right)^{-1} \left(B\vec{c} - B\vec{\mu}_\kappa\right) \\
&= \left(\vec{c} - \vec{\mu}_\kappa\right)^T B^T \left(B^T\right)^{-1} \Sigma_\kappa^{-1} B^{-1} B\left(\vec{c} - \vec{\mu}_\kappa\right) \\
&= \left(\vec{c} - \vec{\mu}_\kappa\right)^T \Sigma_\kappa^{-1} \left(\vec{c} - \vec{\mu}_\kappa\right) \\
&= u_\kappa\left(\vec{c}\right)
\end{aligned}
$$

- Conclusion: The Mahalanobis distance metric $u_\kappa()$ is independent of regular (aka invertible) linear transformations.

Impact of the Mahalanobis Invariance

- Can we use this invariance property to simplify the optimization problem of computing the transformation matrix for the Optimal Feature Transform?

$$\hat{\Phi} = \underset{\Phi}{\operatorname{argmin}}\, s_6(\Phi) = \underset{\Phi}{\operatorname{argmin}} \sum_{\kappa=1}^{K} p(\Omega_\kappa) \frac{M}{u_{\kappa_m}}$$

- $\Phi \in R^{(M \times N)}$ with *MN* unknowns.
- Can we reduce the *MN* search space for an optimal solution by using the invariance property of $u_\kappa()$?
- Recall that: $\vec{c} = \Phi \vec{f}$
- What happens when we apply to the feature vector $\vec{c}$ a regular linear transformation?

Impact of the Mahalanobis Invariance – cont

- When we apply a regular linear transformation *B* to $\vec{c}$:

$$\vec{c}\,' = B\vec{c} = B\Phi\vec{f} = \Phi'\vec{f} \quad \text{, where } \Phi' = B\Phi$$

- Due to the invariance of the Mahalanobis distance to regular linear transformations, $\vec{c}\,'$ has the same $u_\kappa()$ and therefore the *same optimal solution* to $s_6(\Phi)$.
- Thus, Φ' is also an optimal feature transformation matrix.
- Can we select a regular linear transformation *B* so that deriving the elements of the transformation matrix Φ' involves a smaller search space?

Impact of the Mahalanobis Invariance – cont

- *B* must be an MxM invertible matrix.
- Let us choose a *B* so that Φ' has the following form:

$$\Phi' = \begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{matrix} \Phi''$$

where Φ'' is multiplied to the left with an MxM identity matrix.

- Why should Φ' have this form?
- Because the search space is reduced from MN dimensions to MN-M^2.

Remarks on Computing Φ

- We reduced the search space, but we still have to estimate Φ'.

$$\hat{\Phi}' = \arg\min_{\Phi'} s_6(\Phi')$$

- Deriving the elements of Φ is not trivial.
- Keep trying to simplify the problem as much as possible.
- For example, we saw how one can exploit the invariance of $u_\kappa()$ to invertible linear transformations in order to reduce the very large search space.