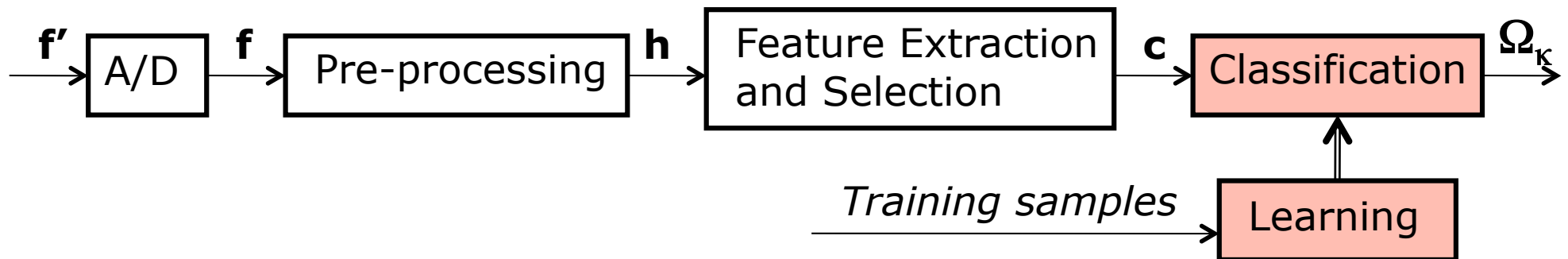# Polynomial Classifiers

Dr. Elli Angelopoulou

Lehrstuhl für Mustererkennung (Informatik 5)

Friedrich-Alexander-Universität Erlangen-Nürnberg

Pattern Recognition Pipeline

- **Classification**
  - Statistical classifiers
    - Bayesian classifier
    - Gaussian classifier
  - Polynomial classifiers

Key Concepts of Polynomial Classifiers

- Polynomial classifiers do not use a statistical model (make no assumptions) about the distribution of features (and the associated classes) in feature space.

- Their goal is to directly estimate an approximation to the ideal decision function by a polynomial.

- Typically, the designer of the classifier decides what degree of polynomial to use.

- Deriving a polynomial classifier becomes equivalent to computing the coefficients of these polynomials from a labeled training set (supervised training).

Discriminant Function

- Consider a two class problem, of the form a feature vector $\vec{c}$ either belongs to a class or not.

- Examples:
  - car/non-car
  - person/non-person
  - pass quality control/does not pass quality control.

- A discriminant function for class Ω_κ is a polynomial that evaluates to 1 if the feature vector $\vec{c}$ belongs to that class. Otherwise it evaluates to zero.

$$d_\kappa(\vec{c}) = \begin{cases} 1 & \text{if } \vec{c} \in \Omega_\kappa \\ 0 & \text{otherwise} \end{cases}$$

Assumption 1

1. Classification is done by using *K* (*K*= number of classes) discriminant functions (Trennfunktionen).

$$d_1(\vec{c}), d_2(\vec{c}), \ldots, d_K(\vec{c})$$

- We have as many discriminant functions as classes.

- Where in the statistical classifiers we had as many posterior probabilities as we had classes, we now have discriminant functions.

- We decide for the class Ω_λ that achieves the maximum discrimination/separation.

$$\lambda = \arg\max_\kappa d_\kappa(\vec{c})$$

Assumption 2

2. We assume that these K discriminant functions, $d_\kappa(\vec{c})$, belong to a parametric family of functions:

$$d_\kappa(\vec{c}) \in d(\vec{c}, \vec{a}_\kappa)$$

where $\vec{a}_\kappa$ are the coefficients of the polynomial $d_\kappa(\vec{c})$.

■ For example, if I have parabolas as discirminant functions, the functions are of the form:

$$d_\kappa(\vec{c}) = a\vec{c}^2 + b\vec{c} + e \quad \text{and} \quad \vec{a}_\kappa = (a, b, e)$$

■ Instead of a parametric family of pdfs as in the Gaussian classifier, we have a parametric family of functions.

Optimal Decision Function

- Ideally, an optimal decision function should map a vector $\vec{c}$ to class Ω_κ if it truly belongs to Ω_κ:

$$\delta(\vec{c}) = \begin{cases} 1 & \text{if } \vec{c} \in \Omega_\kappa \\ 0 & \text{for all other classes} \end{cases}$$

- Since we have a binary decision function, we can build a binary K-dimensional decision vector with 0s for all the wrong classes and 1 only in the correct class Ω_κ.

$$\vec{\delta}(\vec{c}) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Linear Discriminant Function

- Key question: How do we estimate the parameters of the discriminant function?

- Consider a **linear** discriminant function:

$$d_\lambda(\vec{c}) = (a_{\lambda,0}, a_{\lambda,1}, \ldots, a_{\lambda,M}) \begin{bmatrix} 1 \\ c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix}$$

$$d_\lambda(\vec{c}) = \vec{a}_\lambda \vec{c}'^T$$

  where M is the dimensionality of the feature vector $\vec{c} = (c_1, c_2, \ldots, c_M)$, $\vec{c}' = (1, c_1, c_2, \ldots, c_M)$ and $M+1$ is the number of coefficients.

- We want to derive the values of $a_{\lambda,i}$ for $i = 0, \ldots, M$ and $\lambda = 1, \ldots, K$ from the training set.

Training Set

- We have a training set T composed of N pairs of feature vectors and their assigned class:

$$T = \left\{ \left(\vec{c}_l, \Omega_{\kappa(l)} \right), l = 1, 2, \ldots, N \right\}$$

where $\Omega_{\kappa(l)}$ is the class of feature vector $\vec{c}_l$.

- How can we use this training set?

- An ideal discriminant function $d_\lambda(\vec{c})$ would assign a sample $\vec{c}_l$ to its correct class $\Omega_{\kappa(l)}$.

- In other words, if $v = \kappa(l)$ then in an ideal separating function $d_v(\vec{c}) = 1$, while for $v \neq \kappa(l)$ one should get $d_v(\vec{c}) = 0$.

Multiple Equations

■ So given a collection of feature vectors $\vec{c}_l$, with known class membership we can write $N>M$ equations for each class Ω_λ. The equations are equal to 1 or zero, depending on whether the feature vector $\vec{c}_l$ belongs to class Ω_λ.

$$d_\lambda(\vec{c}_1) = \vec{a}_\lambda \vec{c}_1'^T = 0$$

$$d_\lambda(\vec{c}_2) = \vec{a}_\lambda \vec{c}_2'^T = 1$$

$$\vdots$$

$$d_\lambda(\vec{c}_l) = \vec{a}_\lambda \vec{c}_l'^T = 1$$

$$\vdots$$

$$d_\lambda(\vec{c}_N) = \vec{a}_\lambda \vec{c}_N'^T = 0$$

Linear System of Equations

- We can write this system of linear equations in matrix form.

$$
\begin{bmatrix}
1 & c_{11} & \dots & c_{1j} & \dots & c_{1M} \\
1 & c_{21} & \dots & c_{2j} & \dots & c_{2M} \\
1 & c_{31} & \dots & c_{3j} & \dots & c_{3M} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & c_{i1} & \dots & c_{ij} & \dots & c_{iM} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & c_{(N-1)1} & \dots & c_{(N-1)j} & \dots & c_{(N-1)M} \\
1 & c_{N1} & \dots & c_{Nj} & \dots & c_{NM}
\end{bmatrix}
\begin{bmatrix}
a_{\lambda,0} \\
a_{\lambda,1} \\
\vdots \\
a_{\lambda,i} \\
\vdots \\
a_{\lambda,M}
\end{bmatrix}
=
\begin{bmatrix}
0 \\
1 \\
\vdots \\
1 \\
\vdots \\
0
\end{bmatrix}
\Rightarrow A\vec{a}_\lambda^T = \vec{b} \Rightarrow \vec{a}_\lambda^T = A^+\vec{b}
$$

- Then we can solve for the coefficient vector $\vec{a}_\lambda^T$ of a discriminant function $d_\lambda(\vec{c})$ using the pseudoinverse.

Ideal Discriminant Function

■ As previously stated an ideal discriminant function should lead to correct classification decision.

■ So the ideal separating function is:

$$d_\lambda(\vec{c}) = \begin{cases} 1 & \text{if } \lambda = \kappa(1) \\ 0 & \text{if } \lambda \neq \kappa(1) \end{cases}$$

■ In practice, we can not expect to get exactly zero and exactly 1, so we use the following approximations:

$$d_\lambda(\vec{c}) = \begin{cases} \left(d_\lambda(\vec{c}) - 1\right)^2 = \min & \text{if } \lambda = \kappa(1) \\ \left(d_\lambda(\vec{c})\right)^2 = \min & \text{if } \lambda \neq \kappa(1) \end{cases}$$

Ideal Discriminant Function – cont.

- We want our polynomial separating functions to approximate as closely as possible the ideal decision function.

- The ideal decision function is $\delta_\kappa(\ )$ and the linear separating function is $d_\kappa(\ )$, where $\vec{\delta} = (\delta_1, \delta_2, \ldots, \delta_\kappa, \ldots, \delta_K)$

- So when computing the discriminant functions, the error we want to minimize is:

$$\varepsilon = \sum_{\kappa=1}^{K} \sum_{l=1}^{N} \left(\delta_\kappa(\vec{c}_l) - d_\kappa(\vec{c}_l) \right)^2$$

Ideal Separating Function – cont.

- The goal of a polynomial classifier is then to derive the polynomial coefficients that minimize the deviation from the ideal decision function:

$$\varepsilon = \sum_{\kappa=1}^{K} \sum_{l=1}^{N} \left(\delta_\kappa(\vec{c}_l) - d_\kappa(\vec{c}_l) \right)^2$$

- In other words find $\vec{a}_\lambda$ such that:

$$\vec{a}_\lambda = \arg\min_{\vec{a}_k} \varepsilon$$

Minimizing ε

- For each $a_{\lambda,i}, \; i = 0,1,\ldots,M$ we do the following.

$$\frac{\partial \varepsilon}{\partial a_{\lambda,i}} = 0 \Rightarrow \left. \partial \sum_{l=1}^{N} \sum_{\kappa=1}^{K} \left(\delta_\kappa(\vec{c}_l) - d_\kappa(\vec{c}_l) \right)^2 \middle/ \partial a_{\lambda,i} \right. = 0$$

$$\frac{\partial \sum_{l=1}^{N} \sum_{\kappa=1}^{K} \left(\delta_\kappa(\vec{c}_l) - (a_{\kappa,0}, a_{\kappa,1}, \ldots, a_{\kappa,M}) \begin{bmatrix} 1 \\ c_{l,1} \\ c_{l,2} \\ \vdots \\ c_{l,M} \end{bmatrix} \right)^2}{\partial a_{\lambda,i}} = 0$$

Minimizing ε - cont

$$-2\sum_{l=1}^{N}\sum_{\kappa=1}^{K}\left(\delta_{\kappa}(\vec{c}_l)-(a_{\kappa,0},a_{\kappa,1},\ldots,a_{\kappa,M})\begin{bmatrix}1\\c_{l,1}\\c_{l,2}\\\vdots\\c_{l,M}\end{bmatrix}\right)c_{l,i}=0$$

$$-2\sum_{l=1}^{N}\sum_{\kappa=1}^{K}\left(\delta_{\kappa}(\vec{c}_l)-\vec{a}_{\kappa}\vec{c}_l^{\prime T}\right)c_{l,i}=0$$

- Note that this equation is linear in $(a_{\kappa,0},a_{\kappa,1},\ldots,a_{\kappa,M})$

Solving the Minimization

- We need to repeat this process for each $a_{\lambda,i},\ i = 0,1,\ldots,M$

- We get a system of linear equations:

$$-2\sum_{l=1}^{N}\sum_{\kappa=1}^{K}\left(\delta_{\kappa}(\vec{c}_{l}) - \vec{a}_{\kappa}\vec{c}_{l}^{\prime T}\right) = 0$$

$$-2\sum_{l=1}^{N}\sum_{\kappa=1}^{K}\left(\delta_{\kappa}(\vec{c}_{l}) - \vec{a}_{\kappa}\vec{c}_{l}^{\prime T}\right)c_{l,1} = 0$$

$$-2\sum_{l=1}^{N}\sum_{\kappa=1}^{K}\left(\delta_{\kappa}(\vec{c}_{l}) - \vec{a}_{\kappa}\vec{c}_{l}^{\prime T}\right)c_{l,2} = 0$$

$$\vdots$$

$$-2\sum_{l=1}^{N}\sum_{\kappa=1}^{K}\left(\delta_{\kappa}(\vec{c}_{l}) - \vec{a}_{\kappa}\vec{c}_{l}^{\prime T}\right)c_{l,M} = 0$$

$$Q\begin{bmatrix} a_{\kappa,0} \\ a_{\kappa,1} \\ a_{\kappa,2} \\ \vdots \\ a_{\kappa,M} \end{bmatrix} = \vec{b} \Rightarrow \begin{bmatrix} a_{\kappa,0} \\ a_{\kappa,1} \\ a_{\kappa,2} \\ \vdots \\ a_{\kappa,M} \end{bmatrix} = Q^{+}\vec{b}$$

Linear Classifier and Gaussian Classifier

- Recall that a linear classifier is equivalent to a Gaussian classifier where the covariance matrix is independent of the class Ω_κ.

- Given a classification problem, one can test quickly how well a linear classifier works. If we get good results, then we most probably have normally distributed features with same covariances in all classes.

- We can then choose to explicitly use a Gaussian classifier, or otherwise exploit the normal distribution of the features.

- A similar process can be applied for quadratic separating functions and normally distributed features with distinct covariances among the different classes.

Higher Order Polynomials

- In higher order polynomials we take powers of the components of the feature vector $\vec{c}$.

- The general form of higher order polynomials is:

$$d_\lambda(\vec{c}) = \sum_{\substack{n=0 \\ l_1 + l_2 + l_3 + \ldots + l_M = n}}^{P} a_{\lambda,n} c_1^{l_1} c_2^{l_2} \cdots c_M^{l_M}$$

  where P is the degree of the polynomial

- For example, for $P=2$

$$d_\lambda(\vec{c}) = a_{\lambda,0} + a_{\lambda,1} c_1 + a_{\lambda,1} c_2 + \ldots + a_{\lambda,1} c_M +$$

$$+ a_{\lambda,2} c_1^2 + a_{\lambda,2} c_2^2 \ldots + a_{\lambda,2} c_M^2 + a_{\lambda,2} c_1 c_2 + a_{\lambda,2} c_1 c_3 + \cdots$$

Estimation of the Coefficients

- Note that in the higher order polynomials, the discriminant functions are still linear in the $a_{\lambda,i}$s but not in the components of the vector $\vec{c}$.

- This means that estimation of the coefficients $a_{\lambda,i}$ can be done as before.

- We want to get as closely as possible to the ideal decision function, so we use a similar error function.

- To minimize it we take the partial derivative for each $a_{\lambda,i}$.

- We have a system of equations from our training data which we could solve via SVD.
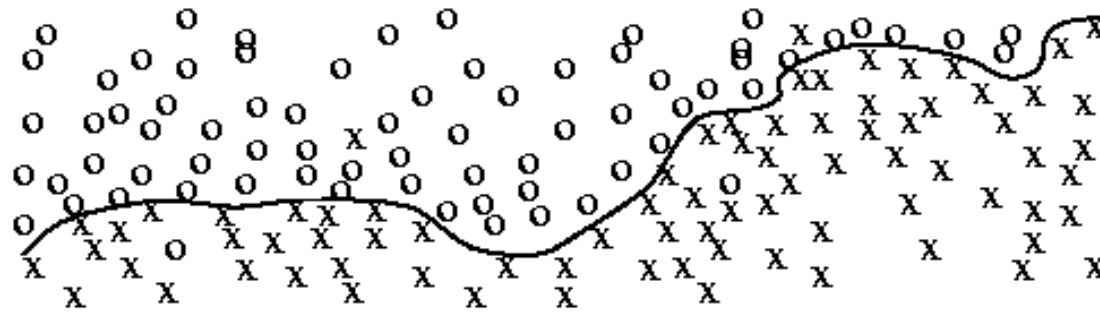
Remarks

- When designing a polynomial classifier one needs:

1. A labeled training set

2. Decide on the degree of the polynomial

- Be careful: from polynomial approximation we know that high order polynomials can perfectly fit the training data, but it may lead to an overfitting problem.

- Data Overfitting: The classifier (or more generally the model) responds to very specific attributes of the data (even noise) that do not generalize to the overall population.
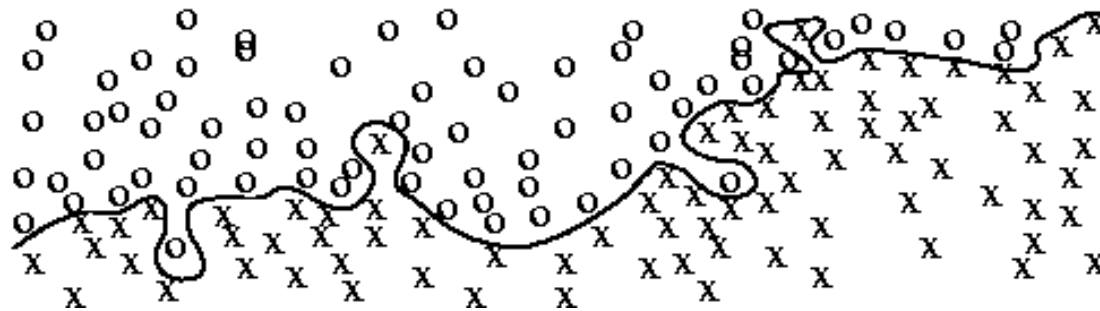
Overfitting Example
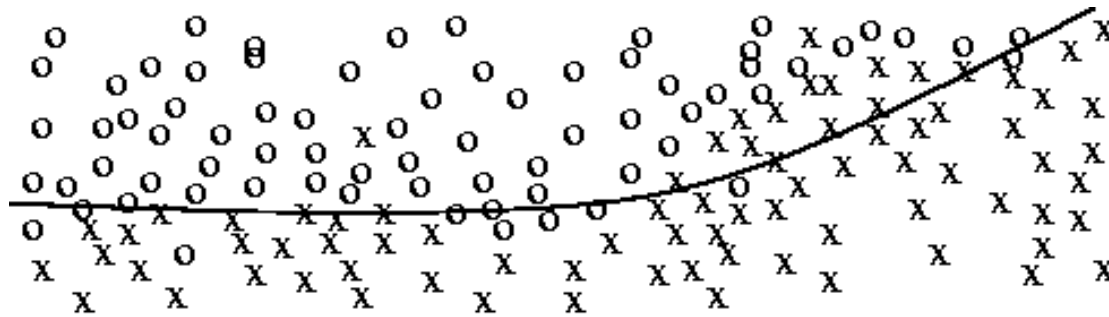
Under-Trained

Well-Trained

Overfitted

Plot courtesy of A. Schmidt http://www.teco.edu/~albrecht/neuro/html/node10.html

More Remarks

- Training is equivalent more or less to solving linear equations.

- If we do not restrict $d_\lambda(\vec{c})$ to a parametric family of functions, and we use a (0,1) cost function with no rejection class, then we will end up with.

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

- In general, because of the so-called Weierstrass principle, polynomial classifiers are considered universal approximations to the Bayesian classifier.