Design of Linear Discriminant Functions

Classification

Perceptrons

Fisher's linear discriminant analysis

(Figures: adapted from Duda, Hart, Stork (2001) Pattern Classification)

November 1, 2019

The Problem of Statistical Decisions

n objects should be grouped in the classes $1, \ldots, k$, $\mathcal{D}$, $\mathcal{O}$

- $\mathcal{D}$: **doubt class** ($\rightarrow$ new measurements required)
- $\mathcal{O}$: **outlier class**, definitively none of the classes $1, 2, \ldots, k$

Objects are characterized by feature vectors $X \in \mathcal{X}$ with $\mathcal{X}$ being the feature space.

Structure of the feature space

- $\mathcal{X} \subset \mathbb{R}^d$
- $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_d$ with $\mathcal{X}_i \subseteq \mathbb{R}$ or $\mathcal{X}_i$ finite.

Remark: in most situations we can define the feature space as subsets of $\mathbb{R}^d$ or as tuples of real, categorial or ordinal numbers; sometimes we have more complicated data spaces composed of lists, trees or graphs.

Motivation: The Problem of Classification

Given: labeled data of type

$$\begin{aligned} Z : \mathcal{O} &\rightarrow \mathbb{R}^d \times [1, \ldots, k], \\ o &\mapsto (X_o, Y_o) \end{aligned}$$

$$\mathcal{Z} = \{z_i = (x_i, y_i) : 1 \leq i \leq n\}$$

Questions:

1. What are the class boundaries?
2. What are the class conditional (specific) densities $p_y(x)$?
3. How many modes or parameters do we need to model $p_y(x)$?
4. ...

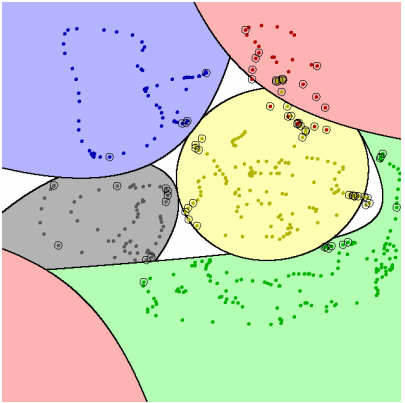


Figure: quadratic SVM classifier for five classes. White areas are ambiguous regions.

Classification error $\hat{c}(x) \neq y$

Classifier mistakes: The indicator function $\mathbb{I}_{\{.\}}$ counts errors

$$\hat{\mathcal{R}}(\hat{c}|\mathcal{Z}) = \sum_{x \in \mathcal{X}} \mathbb{I}_{\{\hat{c}(x) \neq y\}} = \sum_{(x_i, y_i) \in \mathcal{Z}} \mathbb{I}_{\{\hat{c}(x_i) \neq y_i\}}$$

Note that this error count is a random variable!

Expected errors are also called the **expected risk** and they define the quality of a classifier

$$\mathcal{R}(\hat{c}) = \sum_{y \leq k} \mathbf{P}(y) \mathbb{E}_{\mathbf{P}(x|y)} \left[\mathbb{I}_{\{\hat{c}(x) \neq y\}} | Y = y \right] + \text{terms from } \mathcal{D}$$

Remark: The rational behind this choice comes from gambling. If we bet on a particular outcome of our experiment and our gain is measured by how often we assign the measurements to the correct class then classifier with minimal expected risk will win on average against any other classification rule ("Dutch books")!

The Loss Function

Weighted mistakes ...

... are introduced when classification errors are not equally costly; e.g. in medical diagnosis, some disease classes might be harmless and others might be lethal despite of similar symptoms.

Loss function $L(y, z)$

$\Rightarrow$ We introduce a loss function $L(y, z)$ which denotes the loss for the decision z if class y is correct.

• Weighted losses or classification costs $L(y, z) \in \mathbb{R}^+$ are frequently used, e.g. in medicine;
classification costs can also be asymmetric, that means $L(y, z) \neq L(z, y)$ $\quad ((z, y) \sim$ (pancreas cancer, gastritis).

0-1 Loss and Conditional Risk

• All classes are treated the same! We assume constant costs for misclassifications!

$$L^{0-1}(y,z) = \begin{cases} 0 & \text{if } z = y \text{ (correct decision)} \\ 1 & \text{if } z \neq y \text{ and } z \neq \mathcal{D} \text{ (wrong decision)} \\ d & \text{if } z = \mathcal{D} \text{ (no decision)} \end{cases}$$

Conditional risk function of the classifier ...
... is the expected loss of class y

$$\mathcal{R}(\hat{c}, y) = \mathbb{E}_X \left[L^{0-1}(y, \hat{c}(x)) | Y = y \right]$$

$$= \underbrace{\mathbf{P}\{\hat{c}(x) \neq y \wedge \hat{c}(x) \neq \mathcal{D} | Y = y\}}_{\text{pmc}(y) \text{ probability of misclassification}} + \underbrace{d \cdot \mathbf{P}\{\hat{c}(x) = \mathcal{D} | Y = y\}}_{\text{pd}(y) \text{ probability of doubt}}$$

Surrogate loss functions

Often, the direct optimization of 0-1 loss is not feasible (i.e. non-differentiable, non-convex, etc.). Alternatively, the following surrogate loss functions are considered. Let's assume $k = 2$ classes and $y, z \in \{0, 1\}$.

exponential loss takes the distance to the decision surface into account.

$$L^{\text{exp}}(y, z) = \exp\big(-(2y - 1)(2z - 1)\big)$$

logistic loss: $L^{\text{log}}(y, z) = \ln\big(1 + \exp(-(2y - 1)(2z - 1))\big)$

hinge loss favors sparsity and it is used in SVMs

$$L^{\text{hinge}}(y, z) = \max\{0, 1 - (2y - 1)(2z - 1)\}$$

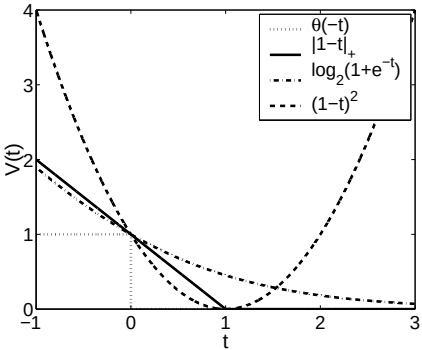
ϵ insensitive loss favors sparsity and it is used in SVMs

$$L^{\epsilon}(y, z) = \max\{0, |y - z| - \epsilon\} =: |y - z|_{\epsilon}$$

Graphs of loss functions

Various loss functions used in classification

Parameter $t = (2y - 1)(2z - 1)$.



L. Rosasco et al., (2004) *Are loss functions all the same*, Neural Computation **15**, 1063-1076

Generative Modeling vs. Discriminative Modeling

Discriminative modeling:
For data generated by (Y, X), find a function $f : x \mapsto y$.

Examples:

- Linear Regression.
- Perceptrons.
- Support Vector Machines (SVMs).
- ...

Generative modeling:

Find a joint model $\mathbf{P}\{Y = y, X = x\}$.

Examples:

- Gaussian Naive Bayes Classifier
- Gaussian Bayes Classifier
- ...

with resulting **discriminant functions**:

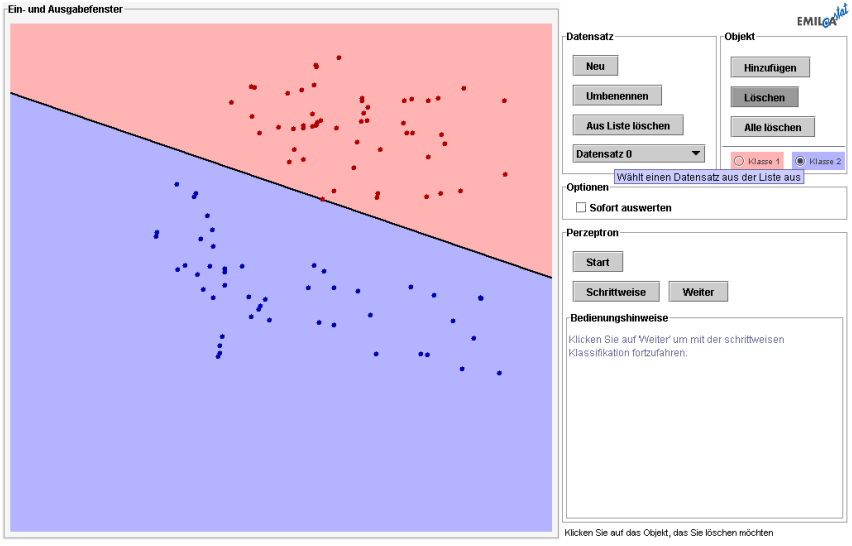
- Linear Discriminant Analysis (LDA).
- Quadratic Discriminant Analysis (QDA).
- ...

Note on generative modeling:

- Harder problem than modeling only $\mathbf{P}\{Y = y|X = x\}$.
- Prone to errors in estimating $\mathbf{P}\{X = x\}$.
- Advantage: outlier detection possible.
- Potentially more powerful.

$\Rightarrow$ Let's start with **linear discriminant functions**.

Linear Classification



Why Linear Classifiers?

Optimality for Gaussians with equal covariances: The Bayes optimal classifier with the maximum a posteriori rule yields linear classification rules for Gaussian class conditional densities when the variances are the same.

Statistical simplicity: We often cannot estimate more complex classifiers than linear discriminant functions due to lack of data and high dimensionality of the feature space.

Computational efficiency: Linear classifiers have efficient learning algorithms and are efficient inference machines.

Historical: Perceptrons were invented by Rosenblatt (1957) and rigorously analyzed by Novikov (1962) who proved convergence of the perceptron learning algorithm.

Generalized Linear Discriminant Functions

Linear Discriminant Functions can be written as

$$g(x) \;=\; w_0 + \sum_{j \le d} w_j x_j \;=\; (w_0, w)(1, x)^T \;=:\; a^T \tilde{x}\,.$$

with generalized coordinates $\tilde{x} = (1, x)^T,\ a = (w_0, w)^T$.

Note that all generalized separating hyperplanes contain the origin of the $\tilde{x}$-space!

Quadratic Discriminant Functions have the form

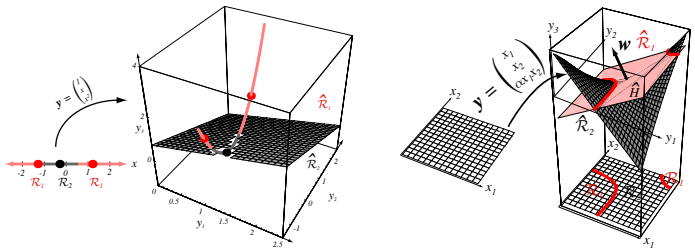
$$g(x) = w_0 + \sum_{j \le d} w_j x_j + \sum_{j \le d} \sum_{l \le d} w_{jl} x_j x_l\,.$$

and they represent the optimal solution for Gaussian class conditional densities.

Linear discriminant with nonlinear feature transformation

How to learn nonlinear discriminant functions?

Use a linear classifier in a high dimensional feature space, generated by a non-linear transformation of feature vectors!



Note that the quadratic transformation $\tilde{x} = (1, x, x^2)^T$ transforms a line in a parabola in three dimensions. A planar split in $\tilde{x}$-space corresponds to a partitioning in x-space which is not simply connected anymore.

Advantage of Homogeneous Coordinates

Homogeneous Representation and Weight Vector:

$$\tilde{x} = (1, x)^T \qquad a = (w_0, w)^T$$

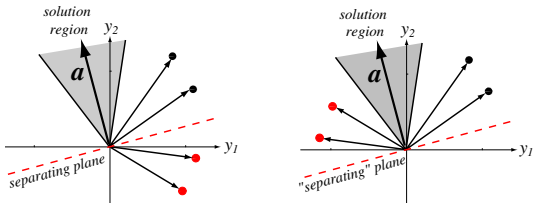
- The transformation is trivial from a mathematical point of view but it simplifies the notation and the analysis substantially.

Dichotomies: Class 1 if $a^T \tilde{x}_i > 0$; class 2 if $a^T \tilde{x}_i < 0$.

$\Longrightarrow$ Normalization: transform $\tilde{x}_i \rightarrow -\tilde{x}_i$ iff the object belongs to class 2.
Now, we have derived a uniform criterion $a^T \tilde{x}_i > 0,$ for all samples i and we do not require a distinction between the two classes.

Linear Separable Two Class Case

Linear Separability $\exists a$ with $\begin{cases} a^T \tilde{x}_i > 0 & \text{for } y_i = 1 \\ a^T \tilde{x}_i < 0 & \text{for } y_i = 2 \end{cases}$



Problem: The solution vector is not unique!
Idea: Introduce a *margin* b to classify data with a "safe distance" from the decision boundary, i.e., $a^T \tilde{x}_i \geq b > 0$.
Regularization of classifier by margin b!

Algorithms for Learning the Weight Vector

Gradient Descent Procedure

$$a(k+1) = a(k) - \eta(k)\nabla J(a(k))$$

$J(a(k))$ is a general cost function for weight vector a. It measures how well a hyperplane orthogonal to a classifies the data.

$\eta(k)$ denotes the step size or learning rate at iteration k.

Gradient Descent Algorithm

INPUT: $a(0), J(\cdot)$

OUTPUT: $a(k^{\text{final}})$

Require: initialize a, threshold ϵ, $\eta(\cdot)$, $k \leftarrow 0$

1: **repeat**
2: $\quad a \leftarrow a - \eta(k)\nabla J(a)$
3: $\quad k \leftarrow k + 1$
4: **until** $|\eta(k)\nabla J(a)| < \epsilon$

What is the Optimal Learning Rate $\eta(\cdot)$?

2nd Order Algorithm: Taylor expansion at $a(k)$ ($H = \frac{\partial^2 J}{\partial a_i \partial a_j}$)

We expand the cost function $J(a)$ at $a(k)$ up to second order (Hessian $H = \frac{\partial^2 J}{\partial a_i \partial a_j}$) .

$$J(a) \approxeq J(a(k)) + \nabla J^T (a - a(k)) + \frac{1}{2}(a - a(k))^T H (a - a(k))$$

Study costs $J(a)$ for weight vector $a(k+1)$ and insert gradient descent rule $a(k+1) - a(k) = -\eta(k)\nabla J(a(k))$ into 2nd order Taylor expansion of $J(a)$. $\Longrightarrow$

$$\begin{aligned} J(a(k+1)) &\approxeq J(a(k)) - \eta(k)\nabla J^T \nabla J + \frac{1}{2}\eta^2(k)\nabla J^T H \nabla J \\ \frac{\partial}{\partial \eta} J(a(k+1)) &= 0 \quad \Rightarrow \quad -\|\nabla J\|^2 + \eta \nabla J^T H \nabla J = 0 \\ \eta^{\text{opt}} &= \frac{\|\nabla J\|^2}{\nabla J^T H \nabla J} \quad \text{optimal learning rate} \end{aligned}$$

Newton's Algorithm

Optimality Condition: Choose $a(k+1)$ optimally to minimize the 2^{nd} order Taylor expansion of $J(a(k+1))$,

$$J(a(k+1)) \approxeq J(a(k)) + \nabla J^T \cdot \big(a(k+1) - a(k)\big) + \frac{1}{2}\big(a(k+1) - a(k)\big)^T \cdot H \cdot \big(a(k+1) - a(k)\big)$$

Choose the optimal weight vector $a(k+1)$!

$$\frac{\partial}{\partial a(k+1)} J(a(k+1)) = \nabla J + H\big(a(k+1) - a(k)\big) = 0$$

New update direction: $a(k+1) - a(k) = -H^{-1}\nabla J$

Newton's Rule: $\boxed{a(k+1) = a(k) - H^{-1}\nabla J}$

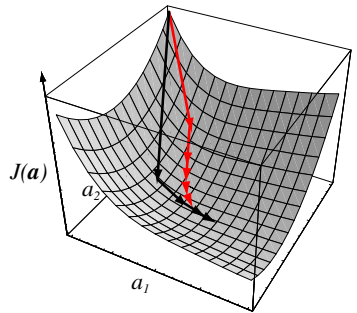
Newton's algorithm and its convergence by gradient descent optimization

Newton's rule changes the update direction in weight space, i.e. we move in the direction $H^{-1}\nabla J(a)$ rather than in the direction $\nabla J(a)$ as suggested by the optimal learning rate.

Newton's algorithm for learning

Require: initialize a,
threshold θ,
learning rate $\eta(\cdot)$,
counter $k \leftarrow 0$

1: **repeat**
2: $a \leftarrow a - H^{-1}\nabla J(a)$
3: $k \leftarrow k + 1$
4: **until** $|H^{-1}\nabla J(a)| < \theta$



black: Newton's rule,
red: naive gradient descent

The Perceptron Criterion

Problem: Solve the inequalities $a^T \tilde{x}_i > 0, \forall i$

Criterion Functions: $J(a; \tilde{x}_1, \ldots, \tilde{x}_n) = \ldots$

- $\ldots$ number of misclassified samples: poor choice since J is piecewise constant! No gradient!
- $\ldots$ sum of violating projections.

Perceptron Criterion: $$J_p(a) = \sum_{\tilde{x} \in \tilde{\mathcal{X}}^{\text{mc}}} (-a^T \tilde{x})$$

$\tilde{\mathcal{X}}^{\text{mc}}$ is the set of misclassified samples.

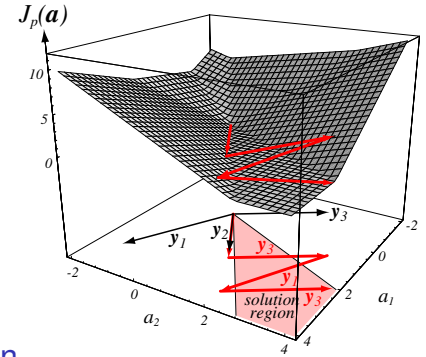
Perceptron Rule: $\Rightarrow \; a(k+1) = a(k) + \eta(k) \sum_{\tilde{x} \in \tilde{\mathcal{X}}^{\text{mc}}} \tilde{x}$

Perceptron Algorithm

Batch Version with Variable Learning Rate

Require: initialize $a, \theta, \eta(\cdot)$

1: $k \leftarrow 0$
2: **repeat**
3: $\quad a \leftarrow a + \sum_{\tilde{x} \in \tilde{\mathcal{X}}^{\text{mc}}} \eta(k) \tilde{x}$
4: $\quad k \leftarrow k + 1$
5: **until** $|\eta(k) \sum_{\tilde{x} \in \tilde{\mathcal{X}}^{\text{mc}}} \tilde{x}| < \theta$

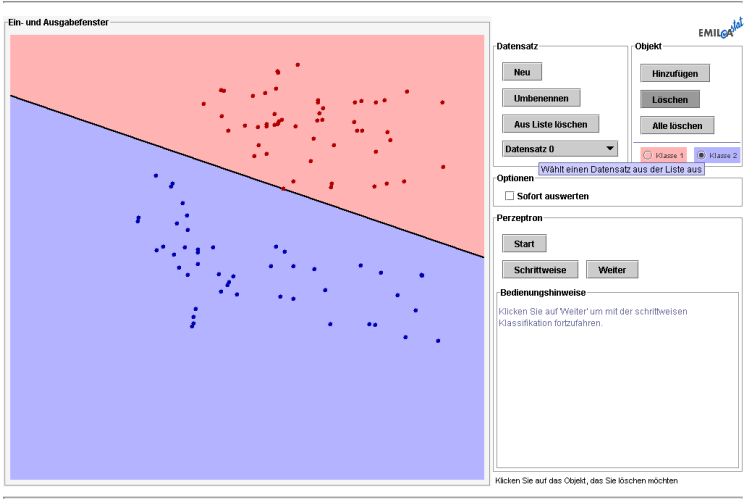


Fixed-Increment Single Sample Perceptron

Require: initialize $a, k \leftarrow 0$

1: **repeat**
2: $\quad k \leftarrow (k + 1) \bmod n$
3: $\quad$ **if** $\tilde{x}^k$ is misclassified by a **then**
4: $\quad\quad a \leftarrow a + \tilde{x}^k$
5: $\quad$ **end if**
6: **until** all patterns are correctly classified

Demonstration of Perceptron Algorithm



Perceptron Convergence

Theorem (Novikov 1962)

If the training samples are linearly separable, then the sequence of weight vectors $a \leftarrow a + \tilde{x}^k$ will terminate at a solution vector.

Proof: Let $\hat{a}$ be a solution vector, i.e., $\hat{a}^T \tilde{x}^i > 0, \ \forall i$ and let $\alpha > 0$ be a scaling factor. Then it holds:

$$\begin{aligned} a(k+1) - \alpha\hat{a} &= a(k) - \alpha\hat{a} + \tilde{x}^k \\ \Rightarrow \ \|a(k+1) - \alpha\hat{a}\|^2 &= \|a(k) - \alpha\hat{a}\|^2 + 2\left(a(k) - \alpha\hat{a}\right)^T \tilde{x}^k + \|\tilde{x}^k\|^2 \end{aligned}$$

Since $\tilde{x}^k$ was misclassified the inequality $a^T(k)\tilde{x}^k \leq 0$ holds.

$$\Rightarrow \ \|a(k+1) - \alpha\hat{a}\|^2 \leq \|a(k) - \alpha\hat{a}\|^2 - 2\underbrace{\alpha\hat{a}^T\tilde{x}^k}_{>0} + \|\tilde{x}^k\|^2$$

$\hat{a}^T\tilde{x}^k$ dominates $\|\tilde{x}^k\|^2$ for sufficiently large α.

Defs.: $\beta^2 := \max_i \|\tilde{x}_{i \in \tilde{\mathcal{X}}^{\text{mc}}}\|^2$, $\gamma := \min_{i \in \tilde{\mathcal{X}}^{\text{mc}}} (\hat{a}^T \tilde{x}_i) > 0$

$$\begin{aligned} \Rightarrow \|a(k+1) - \alpha\hat{a}\|^2 &\leq \|a(k) - \alpha\hat{a}\|^2 - 2\alpha\gamma + \beta^2 \\ &= \|a(k) - \alpha\hat{a}\|^2 - \beta^2 \quad \text{for} \quad \alpha = \beta^2/\gamma \end{aligned}$$

The algorithm converges since $\|a(k+1) - \alpha\hat{a}\|^2$ decreases at least by the constant β^2 and every error will be corrected. □

Bound on the Number of Update Steps

$$\|a(k+1) - \alpha\hat{a}\|^2 \leq \|a(1) - \alpha\hat{a}\|^2 - k\beta^2 = 0$$

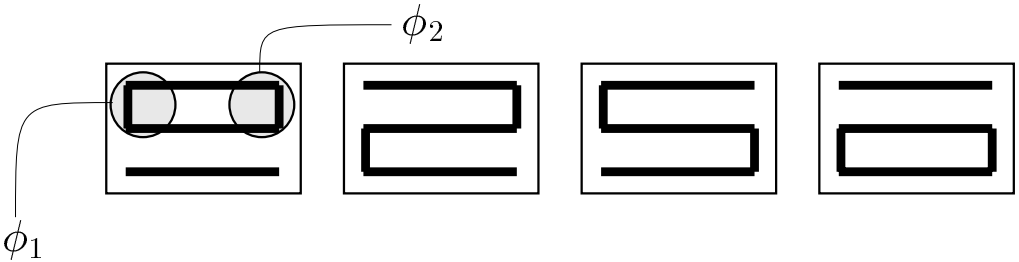
$$\Rightarrow k_0 = \frac{\|a(1) - \alpha\hat{a}\|^2}{\beta^2}$$

Choose $a(1) = 0$

$$\Rightarrow k_0 = \frac{\alpha^2 \|\hat{a}\|^2}{\beta^2} = \frac{\beta^2 \|\hat{a}\|^2}{\gamma^2} = \frac{\max_{i \in \tilde{\mathcal{X}}^{\text{mc}}} \|\tilde{x}_i\|^2 \|\hat{a}\|^2}{\left(\min_{i \in \tilde{\mathcal{X}}^{\text{mc}}} (\hat{a}^T \tilde{x}_i)\right)^2}$$

Remark: Examples $\tilde{x}_i$ that are almost orthogonal to solution vector $\hat{a}$ ($(\hat{a}^T \tilde{x}_i)^2 < \epsilon$, ϵ small) are difficult to learn!

Limitations of Single-Layer Perceptrons



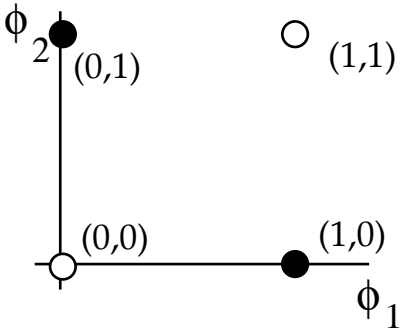
Theorem: A size limited perceptron cannot decide in all cases if parts of a figure are connected or separate.

Proof: The problem is reduced to the XOR problem which is not linearly separable.
ϕ_1, ϕ_2 are detectors which recognize vertical bars in the upper left and right corner.

Limitations of Single-Layer Perceptrons

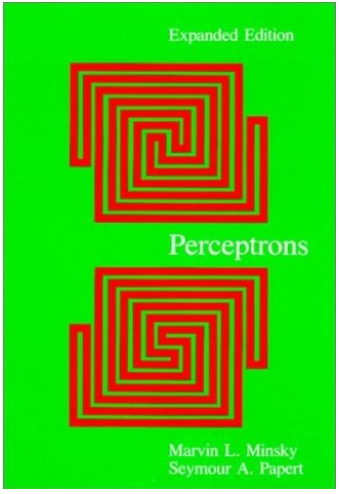
Truth Table for the connectivity problem

ϕ_1	ϕ_2	y
1	1	0
0	1	1
1	0	1
0	0	0



Simple (single layer) perceptrons can be trained efficiently since classification errors can be "blamed" to components of the weight vector a in a direct way. Credit Assignment!

Historical Remarks on Perceptrons



Minsky and Pappert's criticism of the limitations of perceptrons disencouraged funding in this line of research and it stimulated the growth of Artificial Intelligence. The idea that intelligence can be primarily understood as a question of logic irrespective of implementation (e.g. in brains, robots, ...) was pursued in the 70's and 80's with success in expert systems and theorem proving (formal methods).
The big questions of AI like vision, speech, motor control and planning under uncertainty remained open!
In the mid eighties, the Parallel Distributed Processing (PDP) group at UC San Diego revived the field of neural networks with the idea of multilayer perceptrons and the backpropagation algorithm for their training.

Generalizations of the Perceptron Algorithm

Variable-Increment Perceptron with Margin

Require: initialize a, threshold θ, margin $b, \eta(\cdot), k \leftarrow 0$

1: **repeat**
2: $\quad k \leftarrow k + 1$
3: $\quad$ **if** $a^T \tilde{x}^k \leq b$ **then**
4: $\quad\quad a \leftarrow a + \eta(k)\tilde{x}^k$
5: $\quad$ **end if**
6: **until** $a^T \tilde{x}^k > b, \ \forall k$

All the various perceptron algorithms can be formulated as batch algorithms where all the data are processed together and as online algorithms for sequential data processing.
The batch versions are invariant to permutations of the data indices whereas the online versions depend on the sequence. Online variants tend to be more robust when repeated updates are employed for the estimate of the weights. Any idea why?

The WINNOW Algorithm

Idea: exponential update of weights

Consider a two class learning problem with many irrelevant Boolean features $\tilde{x}_i \in \{0,1\}^d$ (normalization $\tilde{x}_i \leftarrow -\tilde{x}_i$ is not performed).

The weights are separated in positive and negative weight vectors a^+, a^-, each associated with one of the two categories. Corrections on a^+ (a^-) are made iff samples of class 1 (2) are misclassified.

Notation: label $y_k = \begin{cases} 1 & \text{if } \tilde{x}_k \text{ belongs to class 1} \\ -1 & \text{if } \tilde{x}_k \text{ belongs to class 2} \end{cases}$

α is a scaling factor for exponential update, i.e., by scaling with a factor $\alpha^{\pm \tilde{x}_{ki}}$.

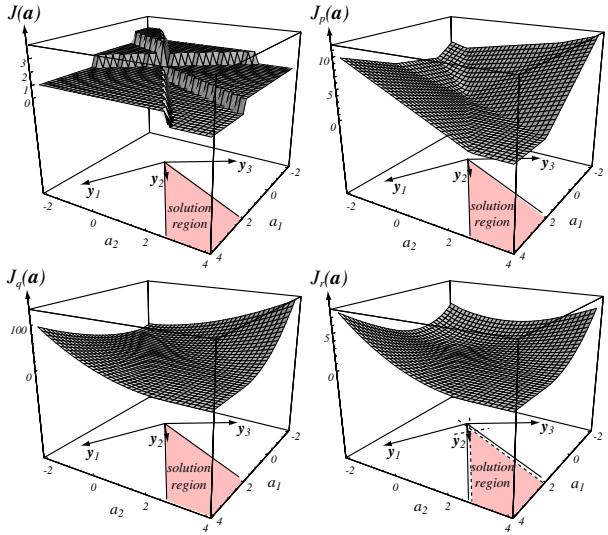
Nick Littlestone, (1988) *Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm, Machine Learning 2: 285-318*

Balanced WINNOW

Require: initialize $a^+, a^-, \eta(\cdot), k \leftarrow 0, \alpha > 1$

repeat
  $k \leftarrow (k+1) \bmod n$
  if $\text{sgn}[(a^{+T} - a^{-T})\tilde{x}_k] \neq y_k$ **then** // pattern misclassified
    if $y_k = +1$ **then** // class 1 error
      for all $i \leq d$ **do**
        $a_i^+ \leftarrow \alpha^{+\tilde{x}_{ki}} a_i^+;\ a_i^- \leftarrow \alpha^{-\tilde{x}_{ki}} a_i^-$ // exponentiated update
      end for
    end if
    if $y_k = -1$ **then** // class 2 error
      for all $i \leq d$ **do**
        $a_i^+ \leftarrow \alpha^{-\tilde{x}_{ki}} a_i^+;\ a_i^- \leftarrow \alpha^{+\tilde{x}_{ki}} a_i^-$ // exponentiated update
      end for
    end if
  end if
until convergence

Different Cost Function



Generative modeling approach

Bayesian view: Both feature vector X and class label Y of an object O are random variables!

Notation
- π_y: fraction of the samples out of class $Y = y$ (π_y is known)
- $p_y(x) := \mathbf{P}\{X = x | Y = y\}$: probability distribution / density of the samples out of class $Y = y$ (class conditional density)
  - $p_y(x)$ is unknown $\Rightarrow$ density estimation
  - $p_y(x)$ is known up to a parameter $\Rightarrow$ parameter estimation

Parametric Statistics: estimate the parameters of the class densities

Non-Parametric Statistics: sample based estimation of the densities

Statistical Learning Theory: minimize the empirical risk

Bayes Rule for Known Densities and Parameters

Assume that we know how the features are distributed for the different classes, i.e., the class conditional densities $p_y(x)$ and their parameters are known. What is the best classification strategy in this situation?

Classifier:

$$\hat{c} : \mathcal{X} \to \mathcal{Y} := \{1, \ldots, k, \mathcal{D}\}$$
$$X \mapsto \hat{c}(X)$$

The assignment function $\hat{c}$ maps the feature space $\mathcal{X}$ to the set of classes $\{1, \ldots, k, \mathcal{D}\}$. (Outliers are neglected)

Quality of a classifier: Whenever a classifier returns a label which differs from the correct class $Y = y$ then it has made an error. Minimize errors to improve quality!

Posterior class probability

Posterior: Let

$$p(y|x) \equiv \mathbf{P}\{Y = y|X = x\} = \frac{\pi_y p_y(x)}{\sum_z \pi_z p_z(x)}$$

be the posterior of the class y given $X = x$.

(The 'Partition of One" $\pi_y p_y(x) / \sum_z \pi_z p_z(x)$ results from the normalization $\sum_z p(z|x) = 1.$)

Likelihood: The class conditional density $p(x|y) = p_y(x)$ is the probability of observing data $X = x$ given class $Y = y$.

Prior: $\pi_y := \mathbf{P}(Y = y)$ is the probability of class $Y = y$.

Bayes Optimal Classifier

Theorem 1 *The classification rule which minimizes the total risk for* $0-1$ *loss is*

$$c(x) = \begin{cases} y & \text{if} \quad p(y|x) = \max_{z \leq k} p(z|x) > 1 - d, \\ \mathcal{D} & \text{if} \quad p(y|x) \leq 1 - d \quad \forall y. \end{cases}$$

Generalization to arbitrary loss functions

$$c(x) = \begin{cases} y & \text{if} \quad \sum_z L(z,y) p(z|x) = \min_{\rho \leq k} \sum_z L(z,\rho) p(z|x) \leq d, \\ \mathcal{D} & \text{else}. \end{cases}$$

Bayes classifier: Select the class y^{MAP} with highest $\pi_y p_y(x)$ value if the costs for not making a decision exceed the loss of this class y^{MAP}, i.e., $\pi_y p_y(x) > (1-d)p(x)$.

Proof: Calculate the total expected loss $\mathcal{R}(\hat{c})$

$$\begin{aligned}\mathcal{R}(\hat{c}) &= \mathbb{E}_X\left[\mathbb{E}_Y\left[L^{0-1}(Y,\hat{c}(x))|X=x\right]\right] \\ &= \int_{\mathcal{X}} \mathbb{E}_Y\left[L^{0-1}(Y,\hat{c}(x))|X=x\right] p(x)dx \text{ with } p(x)=\sum_{z\leq k}\pi_z p_z(x)\end{aligned}$$

Minimize the conditional expectation value since it depends only on $\hat{c}$.

$$\begin{aligned}\hat{c}(x) &= \text{argmin}_{\tilde{c}\in\{1,\ldots,k,\mathcal{D}\}}\mathbb{E}_Y\left[L^{0-1}(Y,\tilde{c})|X=x\right] \\ &= \text{argmin}_{\tilde{c}\in\{1,\ldots,k,\mathcal{D}\}}\sum_{z\leq k} L^{0-1}(z,\tilde{c})p(z|x) \\ &= \begin{cases}\text{argmin}_{\tilde{c}\in\{1,\ldots,k\}}\left(1-p(\tilde{c}|x)\right) & \text{if } d > \min_c(1-p(c|x)) \\ \mathcal{D} & \text{else}\end{cases} \\ &= \begin{cases}\text{argmax}_{\tilde{c}\in\{1,\ldots,k\}}p(\tilde{c}|x) & \text{if } 1-d < \max_c p(c|x) \\ \mathcal{D} & \text{else}\end{cases}\end{aligned}$$

□

Outliers

- Modeling by an outlier class $\pi_{\mathcal{O}}$ with $p_{\mathcal{O}}(x)$
- **"Novelty Detection"**: Classify a measurement as an outlier if

$$\pi_{\mathcal{O}} p_{\mathcal{O}}(x) \geq \max \left\{ (1-d)p(x), \max_z \pi_z p_z(x) \right\}$$

- The outlier concept causes conceptual problems and it does not fit to the statistical decision theory since outliers indicate an erroneous or incomplete specification of the statistical model!
- The outlier class is often modeled by a uniform distribution.
  Attention: Normalization of uniform distribution does not exist in many feature spaces!

$\Longrightarrow$ Limit the support of the measurement space or put a (Gaussian) measure on it!

Class Conditional Densities and Posteriors for 2 Classes

Class-conditional probability density function

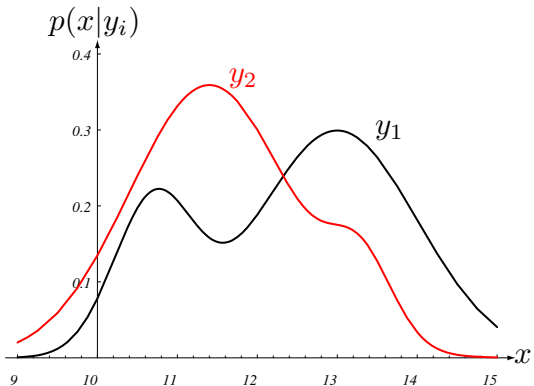


Figure 2.1: Duda, Hart, Stork (2000), *Pattern Classification*, Wiley, 2nd ed.

Posterior probabilities for priors $\mathbf{P}(y_1) = \frac{2}{3}, \mathbf{P}(y_2) = \frac{1}{3}$.

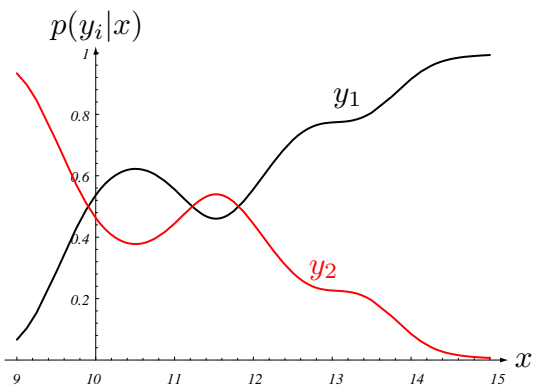
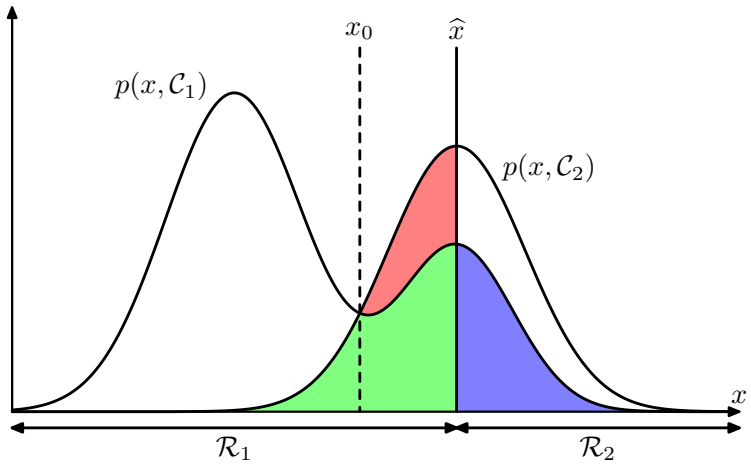


Figure 2.2: Duda, Hart, Stork (2000), *Pattern Classification*, Wiley, 2nd ed.

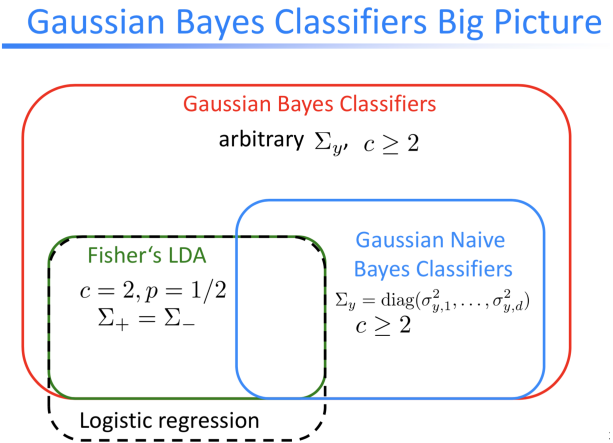
Errors in Dichotomies



The red, blue and green areas are errors for decisions $\hat{x}$ and x_0. The decision x_0 is better than $\hat{x}$.

Gaussian Bayes classifier

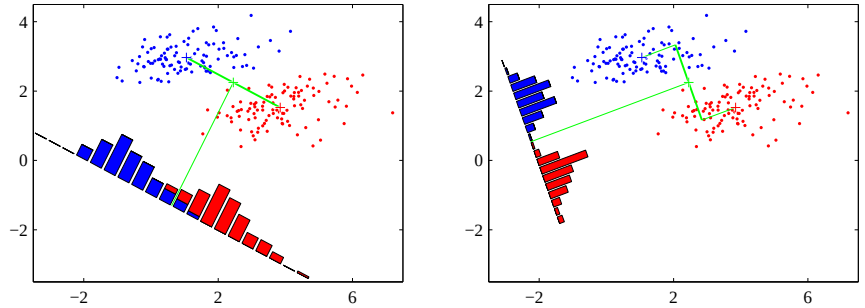
Assume $p_y(x) \sim \mathcal{N}(\mu_y, \Sigma_y)$.



Source: Andreas Krause (2018), Introduction to Machine Learning.

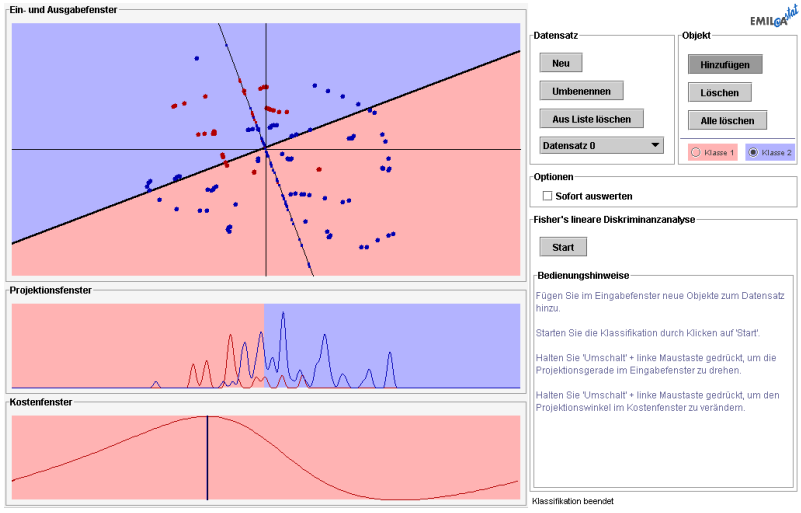
A different view on Fisher's Linear Discriminant Analysis

Idea: Project high-dimensional data of two classes to a one-dimensional linear subspace such that the classes are optimally separated.



Question: How should we select the projection vector w, which optimally discriminates between the different classes?

Applet HTML Page



Fisher's Design Considerations for Linear Discriminants

Classification by dimension reduction

1. Solve the classification problem in a $k-1$ dimensional linear subspace, i.e., a dichotomy is solved by projecting on a line.
2. Represent the decision point for a dichotomy by the projected class conditional means.
3. Separate classes as good as possible relative to their covariances

Problem with this strategy: It is not clear that such a parameterization of the discriminant function actually minimizes the classification risk?

Separability Measured by Sample Means

Samples: $\mathcal{Z} = \{(x_1, y_1), \ldots, (x_n, y_n\} =: \mathcal{X} \times \mathcal{Y}$

- Partition the sample set in "class specific" subsets $\mathcal{Z}_\alpha := \mathcal{X}_\alpha \times \mathcal{Y}_\alpha,\ 1 \leq \alpha \leq k$ with $y_i = \alpha\ \forall x_i \in \mathcal{X}_\alpha$.
- Projected samples: $x_i' = w^T x_i,\ 1 \leq i \leq n$.

Measure for the discrimination of the projected points?

- Sample averages $m_\alpha = \frac{1}{n_\alpha} \sum_{x \in \mathcal{X}_\alpha} x$ with $n_\alpha = |\mathcal{X}_\alpha|$.
- Sample averages of projected points:

$$\tilde{m}_\alpha = \frac{1}{n_\alpha} \sum_{x \in \mathcal{X}_\alpha} w^T x = w^T m_\alpha$$

- Distance of sample averages for 2 classes: $\boxed{w^T(m_1 - m_2)}$

Scaling of class distance

Scatter Matrices

- Scatter of class α: $\Sigma_\alpha = \sum_{x \in \mathcal{X}_\alpha} (x - m_\alpha)(x - m_\alpha)^T$
- "within scatter": $\Sigma_W = \Sigma_1 + \Sigma_2$
- Scatter of projected data: average of class specific variance!

$$\begin{aligned} \sum_{x \in \mathcal{X}_\alpha} (w^T x - \tilde{m}_\alpha)(w^T x - \tilde{m}_\alpha)^T &= \sum_{x \in \mathcal{X}_\alpha} w^T (x - m_\alpha)(x - m_\alpha)^T w \\ &= w^T \Sigma_\alpha w =: \tilde{\Sigma}_\alpha \\ \overset{\text{2 classes}}{\Longrightarrow} \quad \tilde{\Sigma}_1 + \tilde{\Sigma}_2 &= w^T \Sigma_W w \end{aligned}$$

Remark: To separate the data of different classes as good as possible, the class centroids in the projected space should be as far away as possible and, simultaneously, the variance in the projection space of the class specific data should be as small as possible.

Fisher's Separation Criterion

$$J(w) = \frac{\|\tilde{m}_1 - \tilde{m}_2\|^2}{\tilde{\Sigma}_1 + \tilde{\Sigma}_2} = \frac{w^T \overbrace{(m_1 - m_2)(m_1 - m_2)^T}^{=:\Sigma_B} w}{w^T \Sigma_W w}$$

Idea: separate classes as good as possible relative to their covariances

Fisher's plausibility reasoning:
"Optimal" w can be found by maximizing $J(w)$!

$$\begin{aligned} \frac{d}{dw} J(w) &= \frac{d}{dw} \left((w^T \Sigma_W w)^{-1} \, w^T \Sigma_B w \right) \\ &= -2\Sigma_W w (w^T \Sigma_W w)^{-2} \, w^T \Sigma_B w \\ &\quad +2(w^T \Sigma_W w)^{-1} \Sigma_B w = 0 \qquad \Big| \cdot w^T \Sigma_W w / 2 \end{aligned}$$

$$\Sigma_B w = \Sigma_W \, w \, \frac{w^T \Sigma_B w}{w^T \Sigma_W w}$$

Equation for weight vector

Generalized Eigenvalue Problem

Eigenvalue problem for weight vector

Let Σ_W be non singular

$$\Sigma_W^{-1} \underbrace{\Sigma_B w}_{\sim(m_1 - m_2)} = \lambda w \quad \text{with} \quad \lambda = \frac{w^T \Sigma_B w}{w^T \Sigma_W w}$$

$\Sigma_B = (m_1 - m_2)(m_1 - m_2)^T$ is an unscaled projector. The product of Σ_B with any vector w yields a vector parallel to $(m_1 - m_2)$, i.e.,
$\Sigma_B w = (m_1 - m_2)\left((m_1 - m_2)^T w\right) \;||\; (m_1 - m_2)$

Unscaled Solution

$$\boxed{\hat{w} = \Sigma_W^{-1}(m_1 - m_2)} \quad *$$

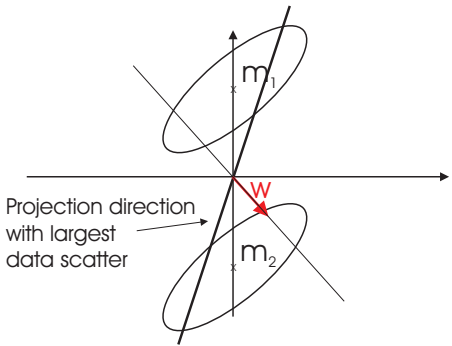
Computational Complexity: The computation of the optimal $\hat{w}$ is determined by the calculation of the intra-class scatter and its inverse, which requires the $\mathcal{O}(d^2 n)$ steps.

Example: Let $m_1 = -m_2 = (0,1)$, $\Sigma_1 = \Sigma_2 = \begin{pmatrix} \lambda & 1 \\ 1 & \lambda \end{pmatrix}$, $\lambda \gtrapprox 1$.

Then it follows:

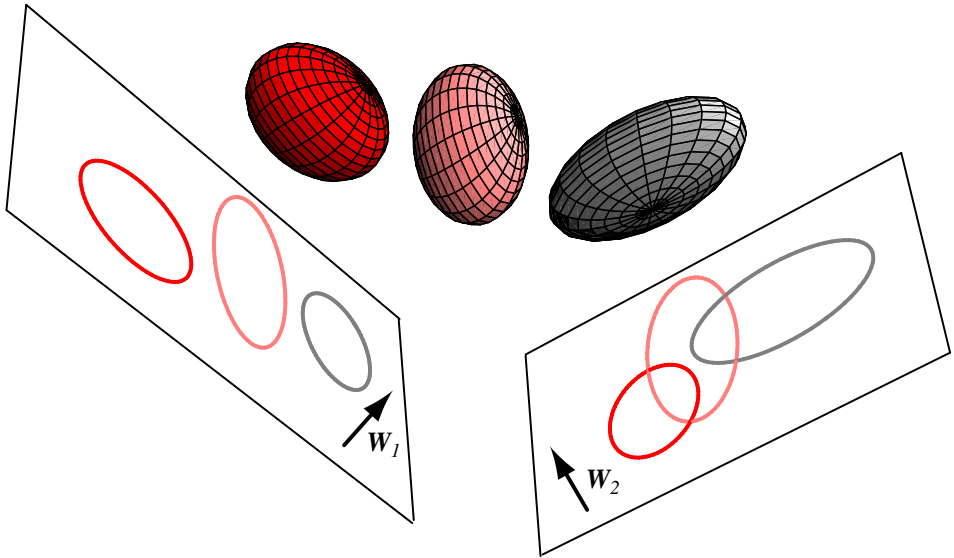
$$\Sigma_W = 2\begin{pmatrix} \lambda & 1 \\ 1 & \lambda \end{pmatrix} \qquad \Sigma_W^{-1} = \frac{1}{2(\lambda^2-1)}\begin{pmatrix} \lambda & -1 \\ -1 & \lambda \end{pmatrix}$$

$$w = \frac{1}{2(\lambda^2-1)}\begin{pmatrix} \lambda & -1 \\ -1 & \lambda \end{pmatrix}\begin{pmatrix} 0 \\ 2 \end{pmatrix} = \frac{1}{\lambda^2-1}\begin{pmatrix} -1 \\ \lambda \end{pmatrix}$$



The weight vector w is orthogonal to the discriminant function.

Multiple Discriminant Analysis



Multiple Discriminant Analysis

k-Class Problem: Find a $k-1$ dimensional linear subspace, which separates the classes in an optimal fashion ($d \geq k$).

Within-Class Scatter: Let $m_\alpha = \frac{1}{n_\alpha} \sum_{x \in \mathcal{X}_\alpha} x$ be the class centroids.

$$\Sigma_W = \sum_{\alpha \leq k} \Sigma_\alpha = \sum_{\alpha \leq k} \sum_{x \in \mathcal{X}_\alpha} (x - m_\alpha)(x - m_\alpha)^T$$

Total Mean Vector: $m = \frac{1}{n} \sum_x x = \frac{1}{n} \sum_\alpha n_\alpha m_\alpha$.

Total Scatter Matrix: $\Sigma_T = \sum_x (x - m)(x - m)^T =$

$$\begin{aligned}
&= \sum_\alpha \sum_{x \in \mathcal{X}_\alpha} (x - m_\alpha + m_\alpha - m)(x - m_\alpha + m_\alpha - m)^T \\
&= \sum_\alpha \sum_{x \in \mathcal{X}_\alpha} (x - m_\alpha)(x - m_\alpha)^T + \sum_\alpha \sum_{x \in \mathcal{X}_\alpha} (m_\alpha - m)(m_\alpha - m)^T \\
&= \Sigma_W + \sum_\alpha n_\alpha (m_\alpha - m)(m_\alpha - m)^T =: \Sigma_W + \Sigma_B
\end{aligned}$$

Σ_B is defined as generalized between-class scatter matrix.

Projection from a d-dimensional feature space to a $(k-1)$-dimensional subspace is achieved by $(k-1)$ discriminant functions $y_i = w_i^T x$, $1 \leq i \leq k-1$ (Matrix notation: $\vec{y} = W^t \vec{x}$).

Fisher's Criterion:

$$J(W) = \frac{|W^T \Sigma_B W|}{|W^T \Sigma_W W|}$$

Solution of this equation:

- Solve the generalized eigenvalue problem $\Sigma_B w_i = \lambda_i \Sigma_W w_i$ for the $(k-1)$ largest eigenvalues by Gram-Schmidt orthonormalization.
- Σ_B is a sum of k rank one (or less) matrices; at most $(k-1)$ matrices are independent (center of mass constraint) $\Longrightarrow$ $|\Sigma_B| \leq (k-1)$.
- Isotropic case: space is spanned by the $(k-1)$ vectors $m_i - m$.

Linear Discriminants and the Multicategory Case

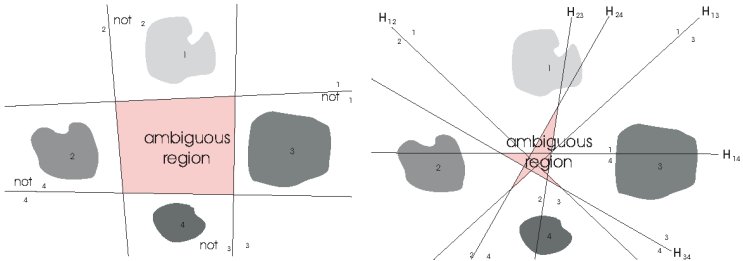
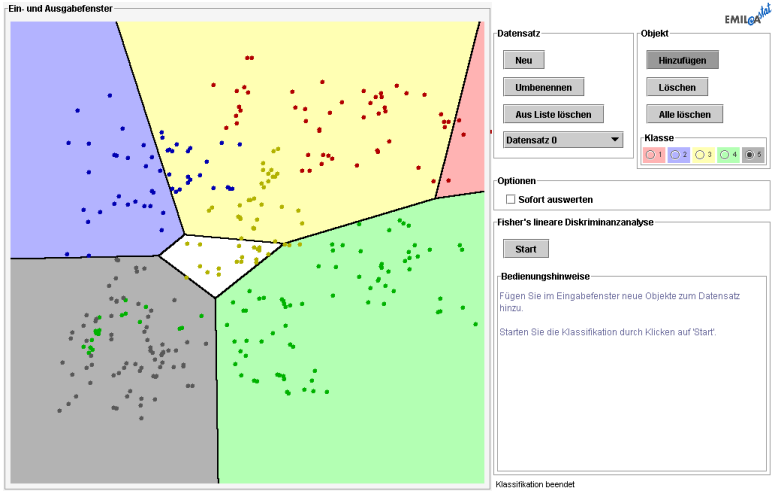


FIGURE 5.3. Linear decision boundaries for a four-class problem. The Top figure shows ω_i/not ω_i dichotomies while the bottom figure shows ω_i/ω_j dichotomies and the corresponding decision boundaries H_{ij}. The pink regions have ambiguous category assignments. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*.

Idea: it is often preferable to reformulate the multiclass problem as $(k-1)$ "*class* α – *not class* α" dichotomies or $k(k-1)/2$ "*class* α *or* β" dichotomies.

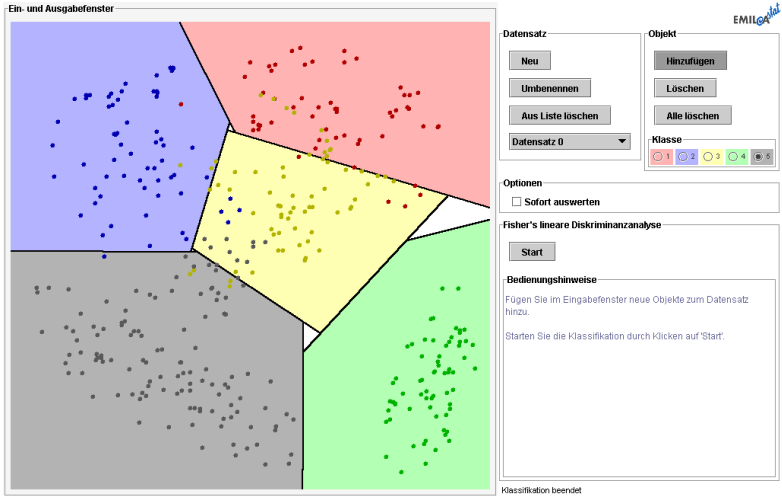
Problem: some areas in feature space are ambiguously classified.

Applet HTML Page



Generated by NetBeans IDE

Applet HTML Page



Generated by NetBeans IDE