Heteroscedastic Discriminant Analysis and its integration into mlR package for uniform machine learning

Prof. dr hab. Katarzyna Stąpor
Institute of Computer Science, Silesian Technical University
Akademicka 16, 44-100 Gliwice, Poland
katarzyna.stapor@polsl.pl
www: zti.polsl.pl/stapor

A G E N D A

- Two approaches two **Discriminant Analysis**: **Bayesian** and **Fisher**
- **Heteroscedastic extension of Fisher LDA – Chernoff Classifier**
- **General methodology** for creating machine learning method
  in a unified way using **mlR package**
- **Integration** of new learner (Chernoff classifier) **into mlR package**
- **Application** of Chernoff classifier for building **credit scoring model**

PATTERN RECOGNITION

"The assignment of a phycical object or event to one of several pre-specified categories" *–Duda and Hart*

"Given some examples of complex signals and the correct decisions for them, make decisions automatically for a stream of future examples" *–Ripley*

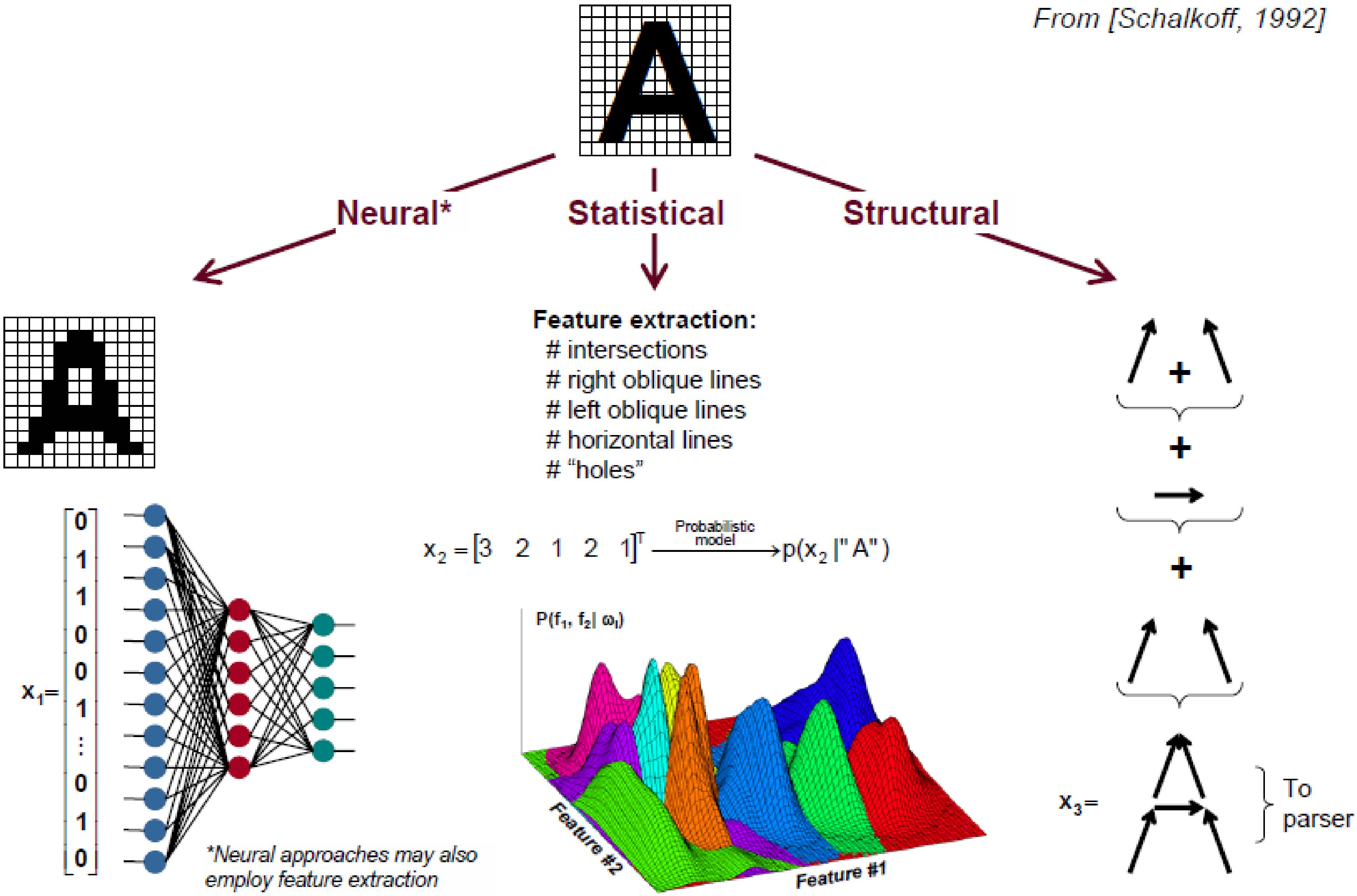
It involves the following tasks:

Classification the problem of assigning an object to a class
f.ex. in ***credit scoring program*** *classifying a candidate as good or bad (i.e. repays or not his loan*)

Clustering discovering groups and structures in the data that are „similar"

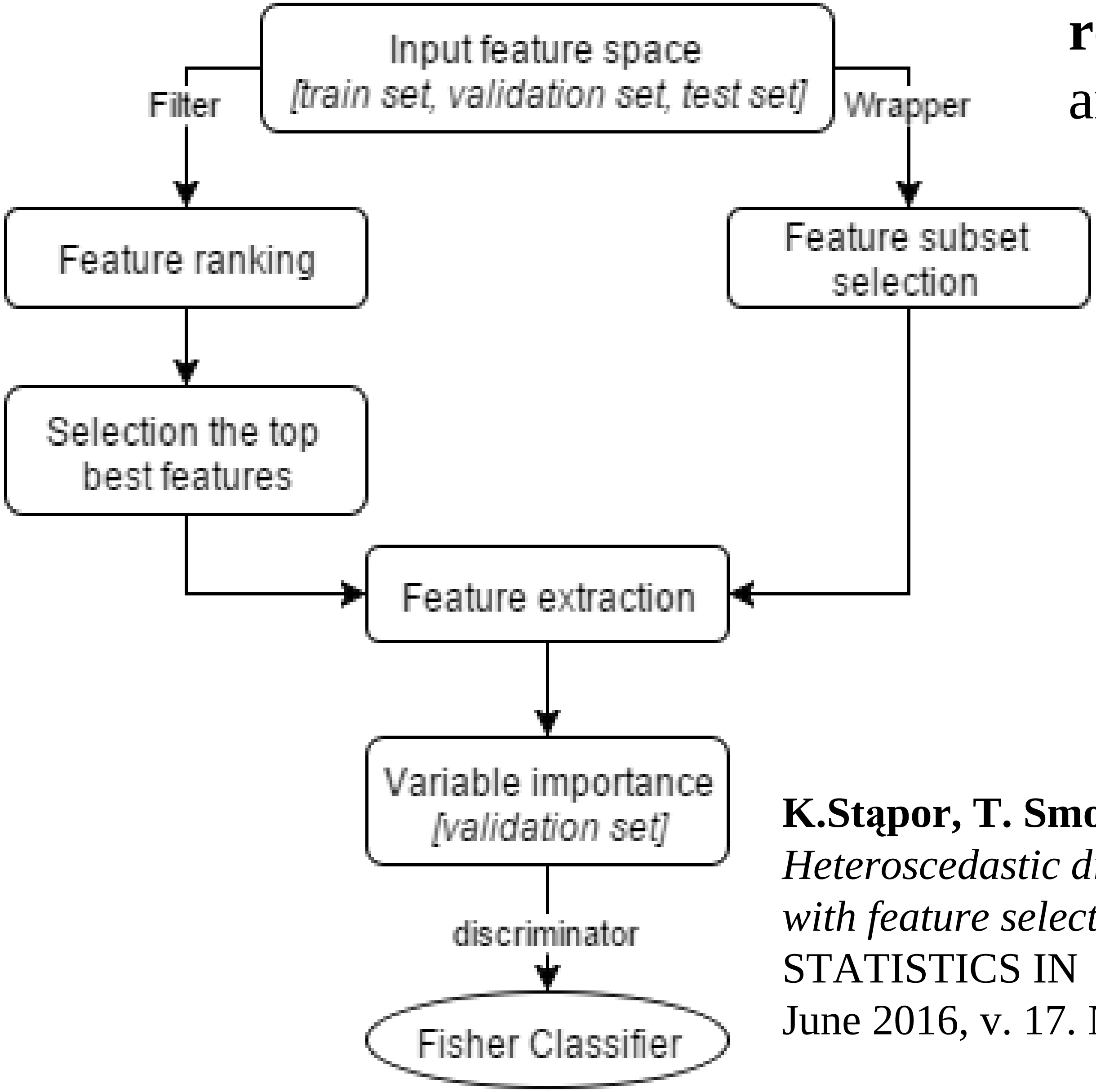
Regression attempts to find a function which models the data with the least error

Pattern recognition approaches



Pattern recognition system for credit scoring

recognition of good and **bad clients**



K.Stąpor, T. Smolarczyk, P. Fabian: *Heteroscedastic discriminant analysis combined with feature selection for credit scoring*. STATISTICS IN TRANSITION New Series, June 2016, v. 17. Nr 2, pp. 1-16.

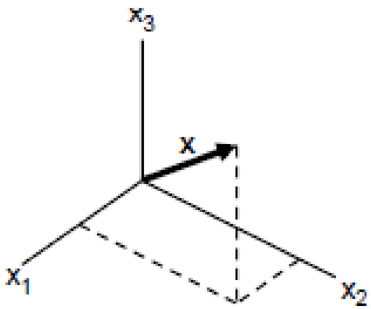
Features and patterns

■ Feature

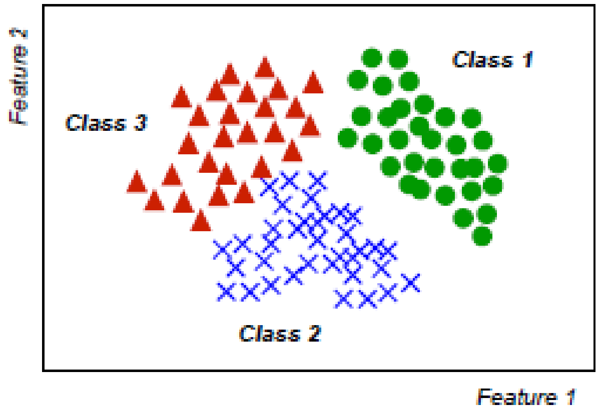
- Feature is any distinctive aspect, quality or characteristic
  - Features may be symbolic (i.e., color) or numeric (i.e., height)
- Definitions
  - The combination of d features is represented as a d-dimensional column vector called a **feature vector**
  - The d-dimensional space defined by the feature vector is called the **feature space**
  - Objects are represented as points in feature space. This representation is called a **scatter plot**

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Feature vector



Feature space (3D)



Scatter plot (2D)

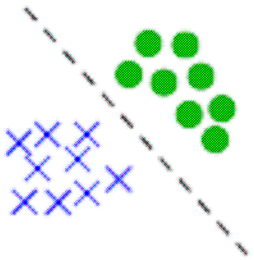
■ Pattern

- Pattern is a composite of traits or features characteristic of an individual
- In classification tasks, a pattern is a pair of variables $\{x, \omega\}$ where
  - $\boldsymbol{x}$ is a collection of observations or features (feature vector)
  - $\boldsymbol{\omega}$ is the concept behind the observation (label)

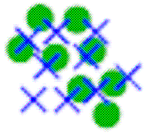
Features and patterns

What makes a "good" feature vector?

- The quality of a feature vector is related to its ability to discriminate examples from different classes
  - Examples from the same class should have similar feature values
  - Examples from different classes have different feature values



"Good" features



"Bad" features

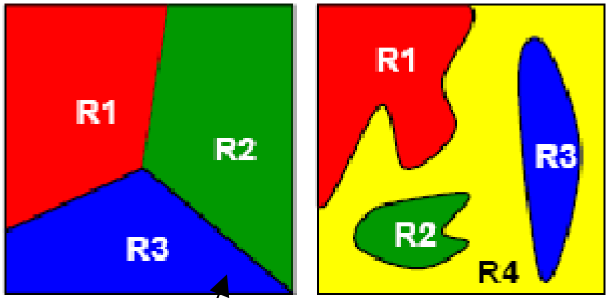
Classification

Prediction of class label for a given observation *x*

Classifier Ψ **- function** that **assigns** to each feature vector $\boldsymbol{x}$ from feature space $\boldsymbol{E}$ class label $\boldsymbol{k}$ (decision) from a decision set $\boldsymbol{I}$

$$\Psi: \quad E \to I = \{1, 2, \ldots, c\}$$

$$\Psi(x) = k \qquad \textbf{c} \text{ – number of classes}$$



feature space $\boldsymbol{E}$ with **decision regions**

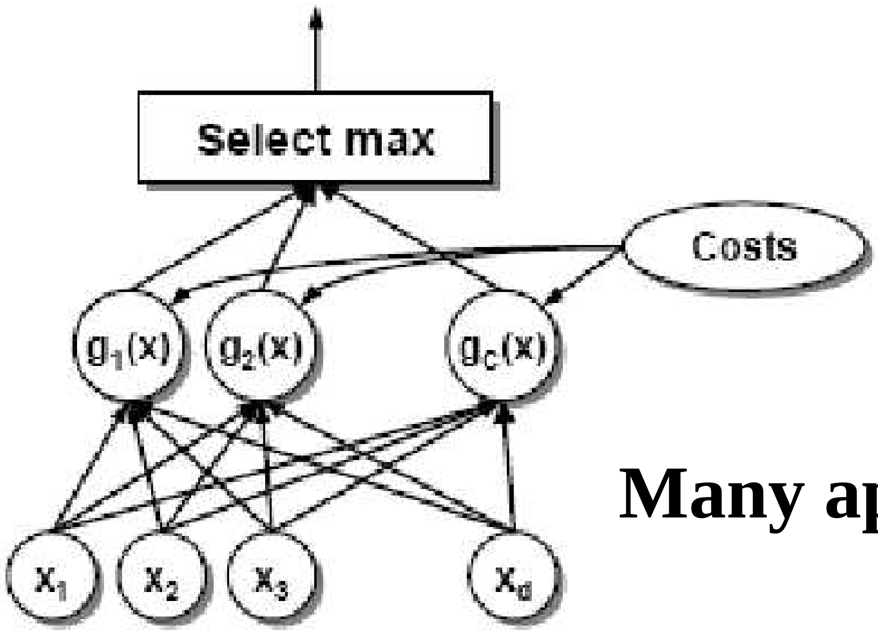
Classifier Ψ partitiones feature space into **decision regions**

Classifiers - discriminant functions

A classifier Ψ can be **represented** as a set of **discriminant functions** g_j

$$\underset{\substack{x \in O_k \\ j=1,\dots,c,\; j \neq k}}{\forall} \quad g_k(x) > g_j(x)$$

Class label



$$\Psi(x) = k \quad \text{if} \quad \underset{\substack{j=1,\dots,c \\ j \neq k}}{\forall} g_k(x) > g_j(x)$$

Many approaches to classification !

Classifier construction process

- **Training** determining it's discriminant functions based on training dataset
- **Validation** determining unknown parameters based on validation set
- **Testing** determining performance of classifier based on testing dataset

Two approaches to discriminant analysis

1) **Bayesian approach**: to apply the **decision theory framework** <u>assuming a parametric form</u> of the population distribution and a prior probability for each class, then derive **Bayesian decision rule** (**Bayes classifier**) for classification.

If the assumed population distribution for each class is **multivariate normal** and the **covariances are common** across different classes, the resulting **decision rule** is based on a **linear function** of the input data

2) **Fisher approach**: looking for a “sensible” rule to discriminate the classes <u>without assuming any particular parametric form</u> for the distribution of the populations.

Bayes classifier Ψ_B

probabilistic classifier

$$\Psi_B(x) = \arg\max_i P(Y = i \mid X = x) = \arg\max_i P(i)\, f_i(x)$$

a posteriori probability

assigns observation x **to** the **most probable class**

$$\boxed{g_i(x) = P(i) \cdot f_i(x)}$$

Bayes discriminant function for i-th class

$f_i(x)$ conditional density function (class i)

$P(i)$ a priori probability of class i

Bayes classifier for normal distribution

conditional density function for *i*-th class:

$$f(x \mid i) = f_i(x) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i) \right]$$

Σ_i covariance matrix

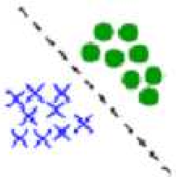
μ_i mean vector

discriminant function for *i*-th class:

$$g_i(x) = \ln f_i(x) + \ln P(i)$$

$$g_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i) - \frac{1}{2}\ln|\Sigma_i| + \ln P(i)$$ **quadratic**

equal covariance matrices: $\Sigma_i = \Sigma \quad i = 1, \ldots, c$



$$g_i(x) = x^T \Sigma^{-1} \mu_i - \frac{1}{2}\mu_i^T \Sigma^{-1} \mu_i + \ln P(i)$$ **linear**

Empirical gaussian Bayes classifier

Training means **estimation** of **unknown parameters**:

mean vector, covariance matrix, a priori probability

Estimators of parameters

$$\hat{\Sigma} \quad \hat{\mu}_i \quad \hat{P}(i)$$

$$\downarrow$$

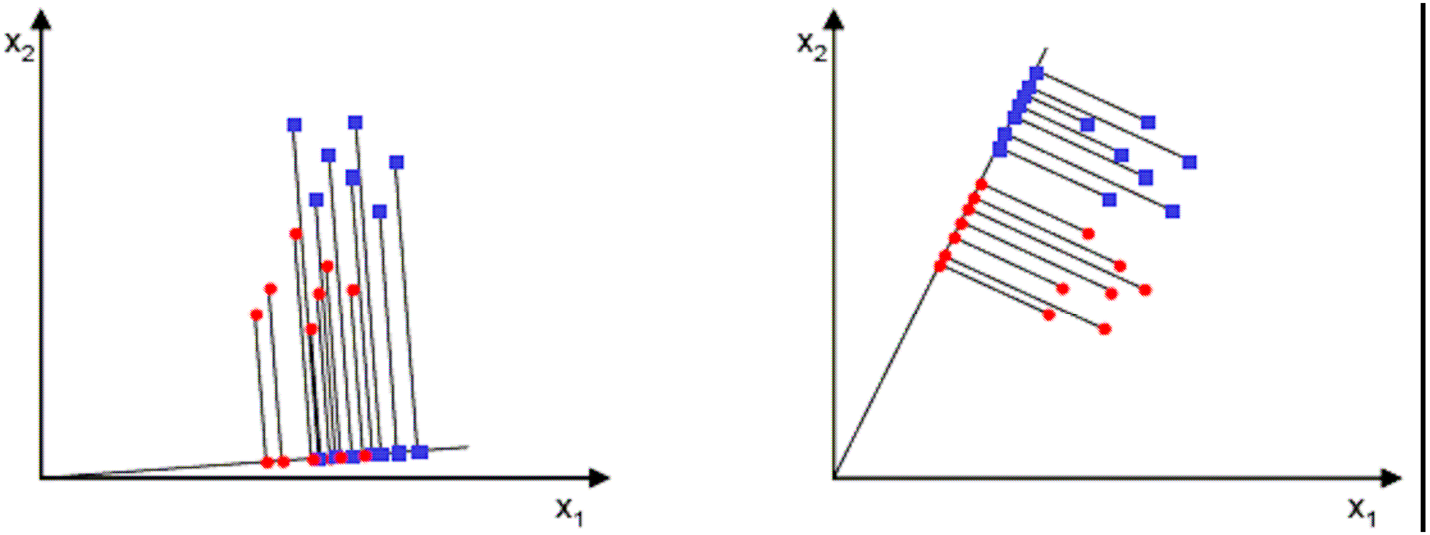
$$\hat{g}_i(x) = x^T \hat{\Sigma}^{-1} \hat{\mu}_i - \frac{1}{2} \hat{\mu}_i^T \hat{\Sigma}^{-1} \hat{\mu}_i + \ln \hat{P}(i)$$

discriminant function for *i*-th class

Fisher's Linear Discriminant Analysis (LDA)

The objective of LDA is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible

$$y = a^T x$$



Ilustration of the idea of LDA for 2-dimensions, 2 populations

Fisher's Linear Discriminant Analysis (LDA)

Given: $X = (X_1, ..., X_d)^T$ multivariate random variable

$G_1, \ldots, G_c$ coming from *c* populations

$\mu_1, ..., \mu_c$ $\Sigma_1, ..., \Sigma_c$ population means and covariance matrices

<u>Assumption</u>: covariance matrices are equal and of full rank

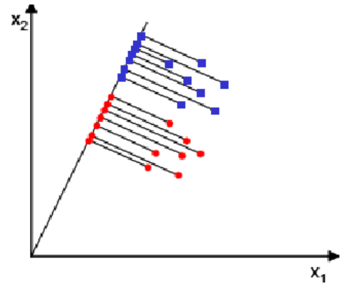
homoscedasticity

$$\Sigma_1 = \Sigma_2 = \ldots = \Sigma_c = \Sigma$$

Fisher's Linear Discriminant Analysis (LDA)

linear combination (a **projection** onto direction ***a*** **)**

$$Y = a^T X$$



The **goodness** of **discrimination**

$$\frac{a^T B a}{a^T \Sigma a}$$

measures the variability **between** the groups of *Y*-values relative **to** the common variability **within** groups

select ***a*** to maximize the above ratio

$$B = \sum_{i=1}^{c} (\mu_i - \bar{\mu})(\mu_i - \bar{\mu})^T$$ **Between covariance matrix** (variation between groups)

$$\bar{\mu} = \frac{1}{c}\sum_{i=1}^{c} \mu_i$$ overall mean

μ_i mean of *i*-th class

Fisher's Linear Discriminant Analysis (LDA)

The vector of coefficients ***a*** that maximizes the ratio:

$$\frac{a^T B a}{a^T \Sigma a}$$

is given by $a_k = e_k \quad k = 1, \dots, s$

$e_1 > \dots > e_k > \dots > e_s$ nonzero **eigenvectors of** $\Sigma^{-1} B$

$\lambda_1 > \dots > \lambda_k > \dots > \lambda_s > 0$ corresponding nonzero **eigenvalues**

$s \leq \min\,(c-1,\; d)$ **dimensionality reduction** $\boxed{\textbf{s = 1} \text{ for c = 2}}$

$a_1 \dots a_k \dots a_s \longrightarrow$ define **new discriminant space**

the linear combination $a_k X$ ***k*-th discriminant variable**

the components of the new discriminant space are **uncorrelated**

Fisher classifier

The new observation x is assigned to the class G_k if:

$$D_k(x) = \min_{j=1,\ldots,c} D_j(x)$$

$$D_j^2(x) = \sum_{i=1}^{s} (y_i - \mu_{jY_i})^2 - 2\log P(j)$$ **Fisher discriminant score**

measures the **Euclidean distance** of the observation ***x***
to the ***j*-th group center** in the **new discriminant space**

Minimum distance classifier in the new discriminant space

Sampled LDA

Σ, μ_i, B unavailable ---- **estimation of the ratio necessary** !

sample between groups matrix:

$$B = \sum_{i=1}^{c} n_i (\overline{x}_i - \overline{x})(\overline{x}_i - \overline{x})^T$$

sample within group matrix:

$$W = \sum_{i=1}^{c} (n_i - 1) S_i$$

$$S_i = \frac{1}{n_i - 1} \sum_{i=1}^{c} \sum_{j=1}^{n_i} \left(x_{ij} - \overline{x}_i \right) \left(x_{ij} - \overline{x}_i \right)^T$$ estimate of Σ_i

Limitation of Fisher LDA

* it **only** tries **to separate class means** as good as possible
* it does not take the discriminatory information present in the **difference of the covariance matrices** (**heteroscedasticity**) into account

Two-class heteroscedastic discriminant analysis

Distance Directed Matrices (DDM) (1)

If there is discriminatory information due to heteroscedasticity
it should be apparent in DDM !

This extra distance is, in general, **in different directions** than
the eigenvectors of $\Sigma^{-1}B$, which separates the means in **Fisher LDA**
and so **DDM** should have **more nonzero eigenvalues**

(1) **Loog M., Duin R.** (2002). *Non-iterative heteroscedastic linear dimension reduction for two-class data: from Fisher to Chernoff*. Proc. 4th Int. Workshop S+SSPR, 508-517

Two-class heteroscedastic disriminant analysis

Distance Directed Matrices (DDM)

Chernoff distance between two probability density functions

$$d_1, d_2$$

$$\partial_C = -\log \int d_1^{\alpha}(x) d_2^{1-\alpha}(x) dx \qquad \alpha \in (0,1)$$

Two-class heteroscedastic disriminant analysis

Distance Directed Matrices (DDM) (1)

For two normally distributed densities, the **DDM** is a positive semi-definite matrix **C**:

$$C = S^{-\frac{1}{2}}\left(m_1 - m_2\right)\left(m_1 - m_2\right)^T S^{-\frac{1}{2}} + \frac{1}{p_1 p_2}\left(\log S - p_1 \log S_1 - p_2 \log S_2\right)$$

$$\alpha = p_1,\ S = p_1 S_1 + p_2 S_2$$

p_i a priori probability of class i

S_i within-class covariance matrix of class i

(1) Loog M., Duin R. (2002). *Non-iterative heteroscedastic linear dimension reduction for two-class data: from Fisher to Chernoff*. Proc. 4th Int. Workshop S+SSPR, 508-517

Two-class heteroscedastic disriminant analysis

DDM

$$C = S^{-\frac{1}{2}}(m_1 - m_2)(m_1 - m_2)^T S^{-\frac{1}{2}} + \frac{1}{p_1 p_2}\left(\log S - p_1 \log S_1 - p_2 \log S_2\right)$$

The **trace of matrix C** is the **Chernoff distance** between two densities

replace matrix **B** by **C** in transformation $\Sigma^{-1}B$

This criterion allows for **preserving as much of the Chernoff distance** in the lower dimensional space **as possible** !

Two-class heteroscedastic disriminant analysis
Chernoff classifier

replace matrix **B** by **C** in transformation

$$\Sigma^{-1}B \longrightarrow \Sigma^{-1}C$$

Solution: eigenvectors corresponding to the largest eigenvalues of $\Sigma^{-1}C$

Chernoff classifier = Fisher classifier
in Chernoff-based discriminant space

mlR package: ***m*achine *l*earning in *R***

A **framework** that offers a **unified interface**
to access various machine learning algorithms in R

Bernd Bischl, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter,
Zachary Jones and Giuseppe Casalicchio (**2016)**

https://CRAN.R-project.org/package=mlr

URLs https://github.com/mlr-org/mlR

mlR: ***m**achine **l**earning in **R***

1) **R** does not define a standardized interface for all its machine learning algorithms!

2) You need to write lengthy, tedious and error-prone wrappers to call the different algorithms and unify their respective outputs.

3) **mlR** provides clear **S3 interface** to **R classification, regression, clustering** and **survival analysis** methods

4) You can **extend it yourself** through **S3 inheritance**

mlR: ***m**achine **l**earning in **R***

A general **methodology**
for building a learner using mlR

1. Task creation
2. Constructing a learner
3. Training a learner
4. Evaluating learner performance
5. Predicting outcomes for new data

mlR: Learning Tasks

ClassifTask	classification problems
RegrTask	regression problems
SurvTask	survival analysis
ClusterTask	cluster analysis
MultilabelTask	multilabel classification
CostSensTask	general cost-sensitive classification

mlR: Task creation (step 1)

make <TaskType>

Example: Fisher Linear Discriminant Analysis on iris dataset:

classif.task <- makeClassifTask (id = "tutorial",
data = iris, **target =** "Species**")**

id id string for object
data a data frame containing the features and target variable(s)
target name(s) of the target variable(s)

mlR: Constructing a learner (step 2)

For classification:

makeLearner (“classif.<method_name>”)

Example: Fisher Linear Discriminant Analysis:

classif.lrn <- makeLearner("classif.lda")

Moreover, you can:

- set hyperparameter (for ex. tuning via grid search)
- control the output for later prediction (e.g. class labels or probabilities)

.

mlR: Training a learner (step 3)

By calling the function **train** on a learner and a suitable Task:

classif.model <- train (classif.lrn, classif.task)

mlR: **evaluating learner performance (step 4)**

1) a large number of performance measures
(**mean misclassification error**, **accuracy** or measures based on **ROC analysis)**

2) **resampling strategies** for evaluation performance
(via the function **makeResampleDesc)**:

- Cross-validation ("CV"),
- Leave-one-out cross-validation ("LOO""),
- Repeated cross-validation ("RepCV"),
- Out-of-bag bootstrap and other variants ("Bootstrap"),
- Subsampling, also called Monte-Carlo cross-validaton ("Subsample"),
- Holdout (training/test) ("Holdout").

mlR: predicting outcomes for new data (step 5)

task.pred <- predict (classif.model,
classif.task, subset = test.set)

returns a named list of **class Prediction**

$data a data frame that contains columns with the true values of the target variable, and the predictions.

getPredictionTruth to access the true and predicted values of the target variable

mlR: Integrating new learner

Interface code to the R function must be written:

1. **Definition of the learner**
2. **Creating the training function of the learner**
3. **Creating the prediction function of the learner**

mlR: Integrating new learner

Definition of the learner (step 1)

All new learners should inherit from **RLearner.classif**

Example: **Chernoff classifier** (HDA) (*)

```
makeRLearner.classif.chernoff <- function() {
    makeRLearnerClassif ( cl = "classif.chernoff",
                          package = "base",
                          par.set = makeParamSet (
                          makeIntegerLearnerParam (id = "directions",
                          default = 1, lower = 1) ),
                         properties = c("twoclass", "numerics", "multiclass"),
                         name = "Chernoff extension to LDA"
                        ) }
```

(*) **K.Stąpor, T. Smolarczyk, P. Fabian**. *Heteroscedastic discriminant analysis combined with feature selection for credit scoring*.
STATISTICS IN TRANSITION New Series, June 2016, v. 17. Nr 2, pp. 1-16

mlR: **Integrating new learner**

Creating the training function of the Learner (step 2)

function (.learner, .task, .subset, .weights = NULL, …)
{.......}

Any special code the learner may need can be encapsulated here !!!

This function must fit a model on the data of the task **.task**
with regard to the subset defined in the integer vector **.subset**
and the parameters passed in the arguments **…** .

Example:

trainLearner.classif.chernoff <- function (.learner, .task, .subset,
.weights, ...) {......... }

mlR: Integrating new learner

Creating the prediction function of the Learner (step 3)

function (.learner, .model, .newdata, …)
{ ... }

Example
predictLearner.classif.chernoff <- function(.learner, .model, .newdata, ...) { ...}

It must predict for the new observations in the **data.frame .newdata**
with the wrapped model **.model**, which is returned from the training function.

The actual model is stored in the **$learner.model** member
and can be accessed through **.model$learner.model**.

mlR: Using the new Learner „classif.chernoff”

task <- **makeClassifTask** (data = …., target = ……)

lrn <- **makeLearner** ("**classif.chernoff**", directions = …)

mod <- mlr::**train** (lrn, task, subset = train.set)

task.pred <- **predict** (mod, task = task, subset = test.set)

Credit Scoring (CS) problem

Problem of discrimination between good and bad clients

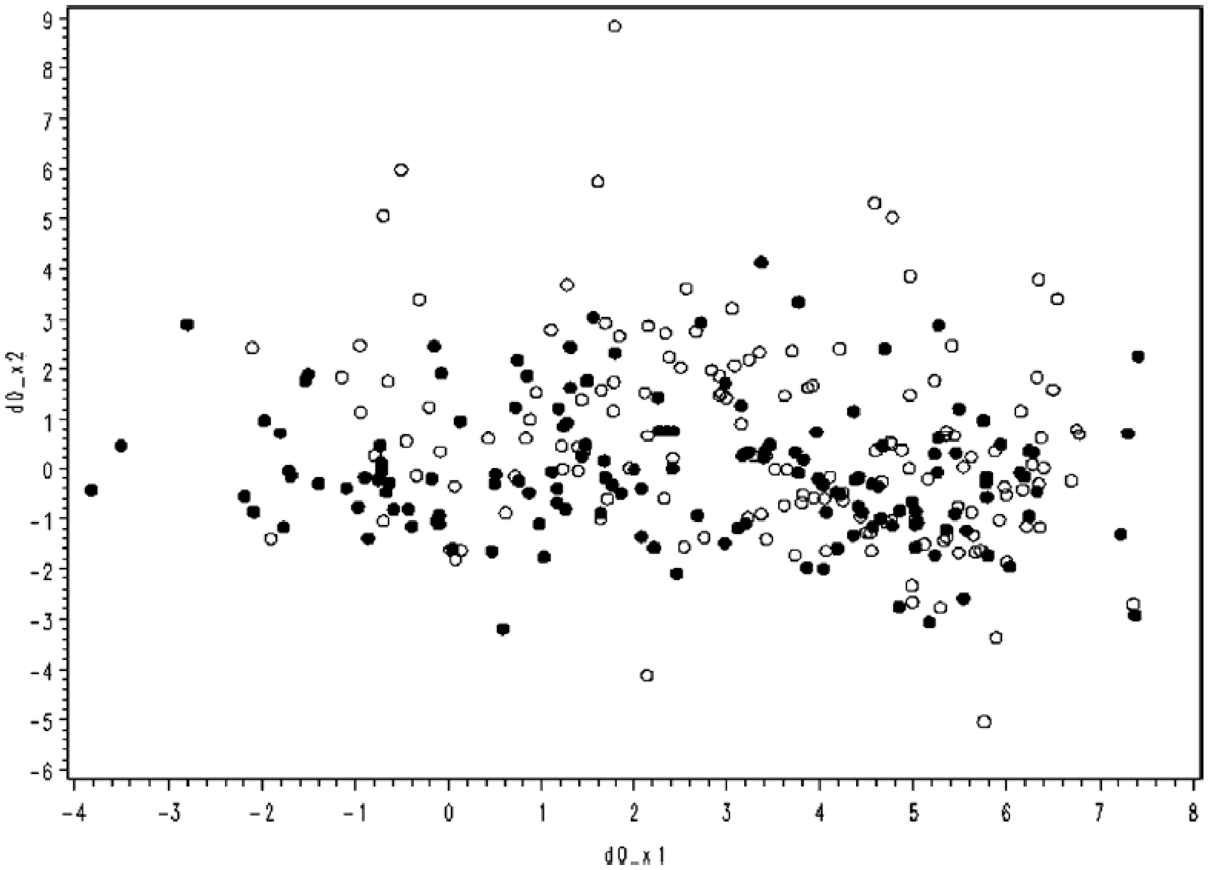


Fig. 1. Illustration of the poor separability of the credit data. Partial least squares was used to transform data for 100 good cases (black) and 100 bad cases (white) selected randomly from the data into two factors given as the *x* and *y* axis of the graph.

German credit dataset

Attribute	Description	Values
1.	Status of existing checking account (qualitative)	A11 : ... < 0 DM A12 : 0 <= ... < 200 DM A13 : ... >= 200 DM /salary assignments for at least 1 year A14 : no checking account
2.	Duration in month (numerical)	
3.	Credit history (qualitative)	A30 : no credits granted/all credits paid back duly A31 : all credits at this bank paid back duly A32 : existing credits paid back duly until now A33 : delay in paying off in the past A34 : critical account/other credits existing (not at this bank)
4.	Purpose (qualitative)	A40 : car (new) A41 : car (used) A42 : furniture/equipment A43 : radio/television A44 : domestic appliances A45 : repairs A46 : education A47 : (vacation - does not exist?) A48 : retraining A49 : business A410 : others

Most important attributes:

1. Duration in month
2. Credit history
3. Installment rate of disposable income
4. Present residence since
5. Present employment since

German credit dataset

5.	Credit amount (numerical)	
6.	Savings account/bonds (qualitative)	A61 : ... < 100 DM A62 : 100 <= ... < 500 DM A63 : 500 <= ... < 1000 DM A64 : .. >= 1000 DM A65 : unknown/ no savings account
7.	Present employment since (qualitative)	A71 : unemployed A72 : ... < 1 year A73 : 1 <= ... < 4 years A74 : 4 <= ... < 7 years A75 : .. >= 7 years
8.	Instalment rate in percentage of disposable income (numerical)	
9.	Personal status and sex (qualitative)	A91 : male : divorced/separated A92 : female : divorced/separated/married A93 : male : single A94 : male : married/widowed A95 : female : single
10.	Other debtors / guarantors (qualitative)	A101 : none A102 : co-applicant A103 : guarantor
11.	Present residence since (numerical)	

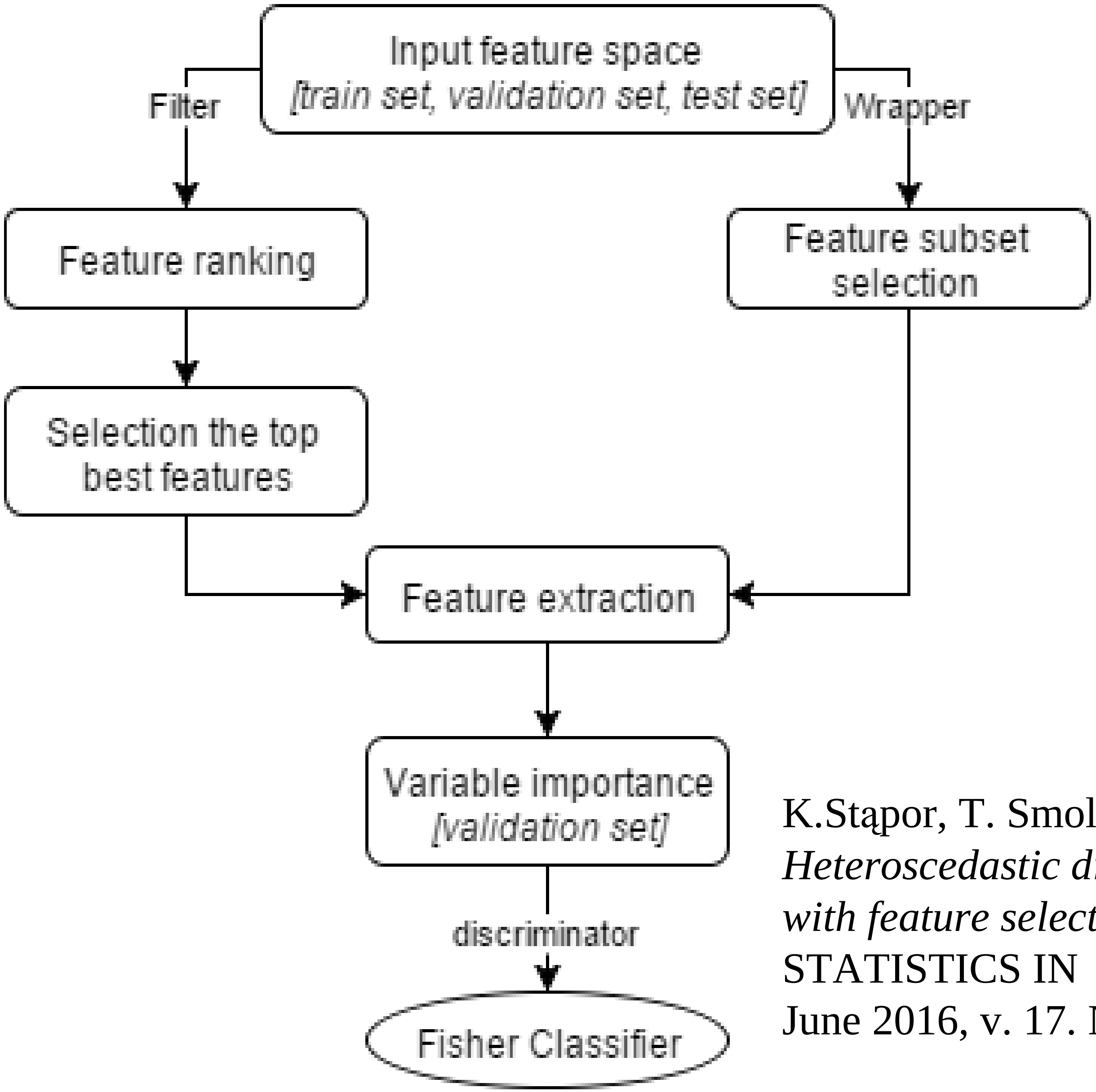
Nominal features were replaced by a number of binary features representing every possibility.

German credit dataset

12.	Property (qualitative)	A121 : real estate A122 : if not A121 : building society savings agreement/life insurance A123 : if not A121/A122 : car or other, not in attribute 6 A124 : unknown / no property
13.	Age in years (numerical)	
14.	Other instalment plans (qualitative)	A141 : bank A142 : stores A143 : none
15.	Housing (qualitative)	A151 : rent A152 : own A153 : for free
16.	Number of existing credits at this bank (numerical)	
17.	Job (qualitative)	A171 : unemployed/ unskilled - non-resident A172 : unskilled - resident A173 : skilled employee / official A174 : management/ self-employed/highly qualified employee/ officer
18.	Number of people being liable to provide maintenance (numerical)	
19.	Telephone (qualitative)	A191 : none A192 : yes, registered under the customer's name
20.	Foreign worker (qualitative)	A201 : yes A202 : no

UCI Machine Learning Repository

Our CS model architecture



K.Stąpor, T. Smolarczyk, P. Fabian: *Heteroscedastic discriminant analysis combined with feature selection for credit scoring*. STATISTICS IN TRANSITION New Series, June 2016, v. 17. Nr 2, pp. 1-16.

Feature Subset Selection
Search strategy and objective function

Feature Subset Selection requires

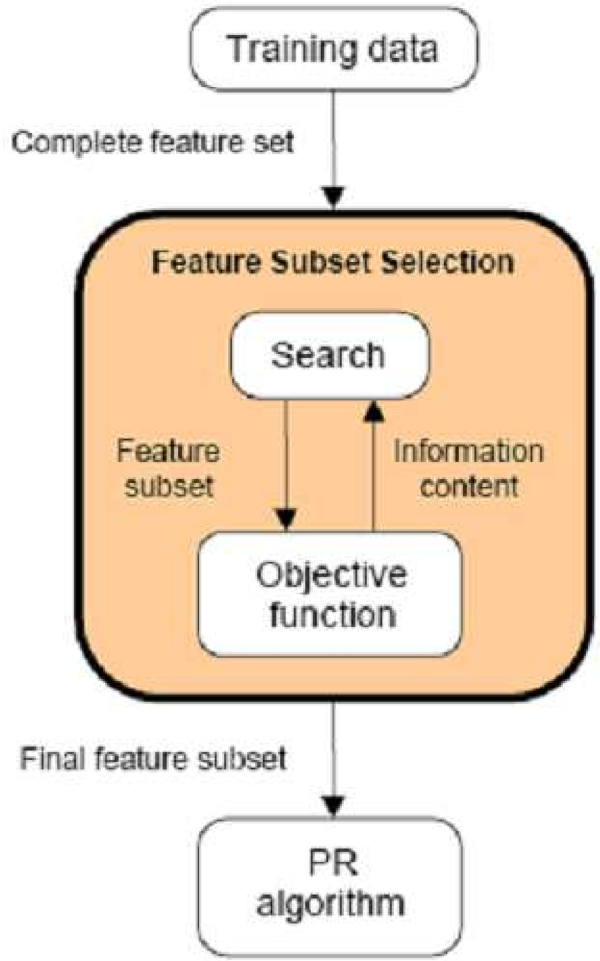
- A search strategy to select candidate subsets
- An objective function to evaluate these candidates

Search Strategy

- Exhaustive evaluation of feature subsets involves $\binom{N}{M}$ combinations for a fixed value of M, and 2^N combinations if M must be optimized as well
  - This number of combinations is unfeasible, even for moderate values of M and N, so a search procedure must be used in practice
  - For example, exhaustive evaluation of 10 out of 20 features involves 184,756 feature subsets; exhaustive evaluation of 10 out of 100 involves more than 10^{13} feature subsets [Devijver and Kittler, 1982]
- A search strategy is therefore needed to direct the FSS process as it explores the space of all possible combination of features

Objective Function

- The objective function evaluates candidate subsets and returns a measure of their "goodness", a feedback signal used by the search strategy to select new candidates



Feature ranking using Fisher Score

- n_i is the number of instances of class
- μ_i and σ_i is the mean and variance of class *i*, corresponding to the *r*-th feature
- μ and σ are the mean and variance of the whole dataset respectively

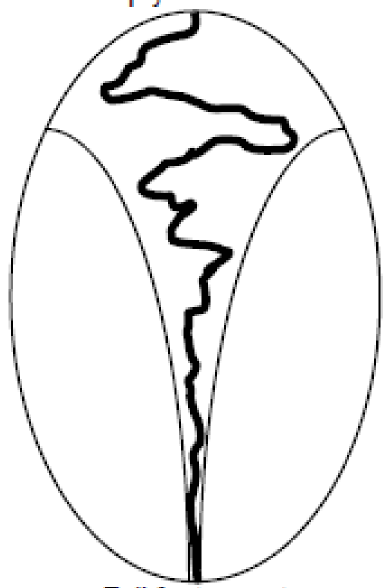
$$F_r = \frac{\sum_{i=1}^{c} n_i (\mu_i - \mu)^2}{\sum_{i=1}^{c} n_i \sigma_i^2}$$

Fisher Score (F-Score) algorithm is designed to find subset of features that will maximize the distance between instances from different classes and, at the same time, minimize the distances within the same class

The larger the F-score is, the more likely this feature is more discriminative

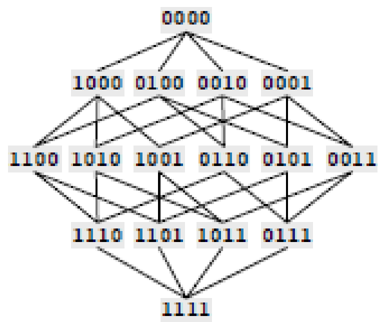
Sequential Forward Selection (SFS)

- **Sequential Forward Selection is the simplest greedy search algorithm**
  - Starting from the empty set, sequentially add the feature x^+ that results in the highest objective function $J(Y_k+x^+)$ when combined with the features Y_k that have already been selected
- **Algorithm**



1. Start with the empty set $Y_0=\{\varnothing\}$
2. Select the next best feature $x^+ = \underset{x \notin Y_k}{\operatorname{argmax}}\left[J(Y_k + x)\right]$
3. Update $Y_{k+1}=Y_k+x^+$; $k=k+1$
4. Go to 2

- **Notes**
  - SFS performs best when the optimal subset has a small number of features
    - When the search is near the empty set, a large number of states can be potentially evaluated
    - Towards the full set, the region examined by SFS is narrower since most of the features have already been selected
  - The search space is drawn like an ellipse to emphasize the fact that there are fewer states towards the full or empty sets
    - As an example, the state space for 4 features is shown. Notice that the number of states is larger in the middle of the search tree
    - The main disadvantage of SFS is that it is unable to remove features that become obsolete after the addition of other features



Sequential Floating Forward Selection (SFFS)

The extension of **SFS** with **flexible backtracking mechanisms**

* it starts from the empty set
* after each forward step, SFFS performs backwards steps as long as the objective function increases

1. Start with the empty set $Y=\{\varnothing\}$
2. Select the best feature

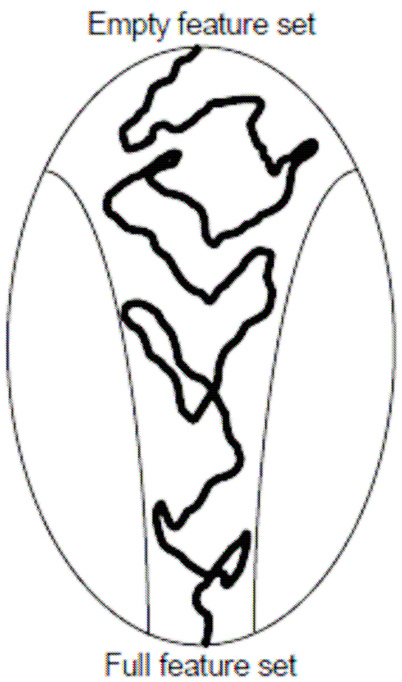
$$x^+ = \underset{x \notin Y_k}{\operatorname{argmax}} \left[J(Y_k + x) \right]$$

$$Y_k = Y_k + x^+; \quad k = k+1$$

3. Select the worst feature*

$$x^- = \underset{x \in Y_k}{\operatorname{argmax}} \left[J(Y_k - x) \right]$$

4. If $J(Y_k - x^-) > J(Y_k)$ then
   $Y_{k+1} = Y_k - x$; $k=k+1$
   go to Step 3
   else
   go to Step 2



Experimental results

Algorithm \ data set	FDA_Cher			
	Accuracy rate (%)		Number of selected features	Number of directions
	Avg.	Std.	Median	
All features	59.70%	11.18%	59	3
CFS	73.90%	4.95%	28	2
FS	75.10%	3.38%	18	3
SFFS	67.00%	6.04%	6	3
GRASP	67.90%	5.43%	17	3
MA	65.80%	8.68%	28	3

German credit dataset

UCI Machine Learning Repository