3 Feature Selection & Feature Extraction

Overview:

3.1 **Introduction**

3.2 **Feature Extraction**

3.3 **Feature Selection**

3.3.1 Max-Dependency, Max-Relevance, Min-Redundancy

3.3.2 Relevance Filter

3.3.3 Redundancy Detection Filter

3.4 **Case Study**

3.1 Introduction

- **Feature extraction**: reduce dimensionality by (linear or non-linear) projection of D-dimensional vector onto d-dimensional vector ($d < D$)

- **Feature selection**: reduce dimensionality by selecting *subset* of original variables

- **Motivation:**
  - reduce building/training complexity
    & generalization capability ↑ (*cf.*, curse of dimensionality)
  - faster training & testing
  - better models: more optimal bias/variance trade-off
    too large # inputs $\Rightarrow >$ # parameters $\Rightarrow >$ variance
    too small # inputs $\Rightarrow >$ bias
  - understanding complex models is more difficult
    (*cf.*, Occam's razor)
  - ranking informative variables ⇒ useful for interpretation

- Types of extraction & selection methods:
  1. **Unsupervised** methods (component analysis)
  2. **Supervised** methods (classification, regression)

3.2 Feature Extraction

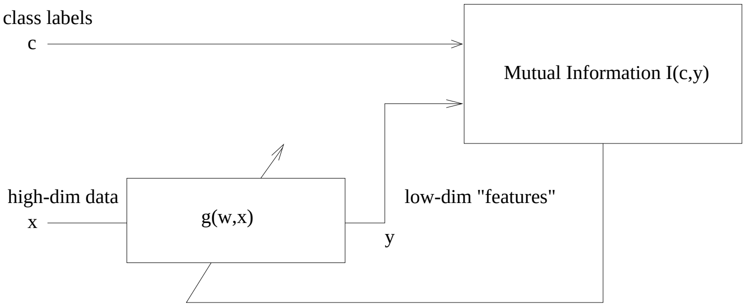
- **Unsupervised:**

  Can be linear or non-linear:

  - Principal Component Analysis (PCA)
    select PCs with largest eigenvalues as "features"
  - Independent Component Analysis (ICA)
    select ICs with largest kurtosis or largest negentropy
  - Multidimensional Scaling (MDS)
    select dimensions $\rightsquigarrow$ acceptable distortion of projected data
  - Topographic Maps (SOMs)
    dimensions of lattice space

- **Supervised:** (also called **Feature Construction**)

  - incorporate knowledge about classes
    - ∗ information used is class. performance = **wrapper** (see further)
    - ∗ information used is an alternative measure of discriminability between classes = called **filter**
      e.g., Linear Discriminant Analysis (LDA), Maximum Mutual Information between features and classes (Torkkola & Campbell, 2000)



  - also possible for *regression* by discretizing target variable into artificial classes (class-blind discretization)

3.3 Feature Selection

→ also for selecting **inputs** (IVS; Input Variable Selection)

- **Unsupervised:**
    1. By ranking input variables:
        - Retain inputs with largest variance
          logic: non-varying inputs cannot lead to changing outputs
        - Determine 1st PC and retain inputs with largest coefficients
          logic: largest coefficients code for largest data range along these dimensions ⇒ most likely lead to changing outputs
        - ...
    2. domain knowledge about which variables likely contribute
       e.g., for a mortgage: income & debt are important, not length of applicant
- **Supervised:** (using outputs, *e.g.*, class labels)
    1. by ranking variables
    2. by subset selection

3.3 Feature Selection – Cont'd

Supervised Feature Selection:

1. **By ranking variables:**

- **Correlation criteria** (Pearson correlation coefficient):

$$\mathcal{R} = \frac{cov(input\ variable\ i, output\ variable)}{\sqrt{var(input\ variable\ i)var(output\ variable)}}$$

  (assuming scalar output y)
  and use this for ranking all components i of input variable

  also usable for classification, *e.g.*, 2-class: $y \in \{-1, 1\}$
  → is related to Fisher's criterion and t-test

- **Single variable classifiers:**
  ranking according to predictive power of variables (classifiers) or goodness of fit (regression)
  predictive power of individual variable:

  – trade-off between false positive rate (fpr) and false negative rate (fnr) by varying threshold $\theta|(fpr = fnr)$ (break-even point)

  – ROC curve ("hit" rate (1-fpr) vs. false alarm fnr) (criterion= max. area under curve)

- **Information-theoretic criteria:**

$$MI(x_j, C) = \sum_C \int_{x_j} p(x_j, C) \log_2 \frac{p(x_j, C)}{p(x_j)P(C)} dx_j$$

  with C class label (Torkkola, 2003) (regression: $C \to y$)
  difficulty = estimating densities!
  discrete case = easier: integrals → sums

  ↪ perform ranking based in MI

3.3 Feature Selection – Cont'd

Supervised Feature Selection – Cont'd:

1. **By ranking variables**

2. **By subset selection** (also called **Feature Construction**):

   It can easily be shown that ranking variables according to their individual predictive power is less useful than selecting subsets of features according to their joint predictive power

   The latter can be tackled with **filters** and **wrappers**:

   - Selection independent from chosen predictor (regressor/classifier) = **filter**:
     - relevance filter (see next)
     - redundancy filter (see next)
   - Use classification (or regression) performance = **wrapper**

     One needs to choose:

     (a) classification (or regression) model
     e.g., Bayesian classifier, MLP, SVM,. . .

     (b) search procedure, *e.g.*, exhaustive search, branch & bound, genetic algorithms (see further)
     e.g., *exhaustive search:*

       - choose inputs to use
       - optimize model parameters
       - quantify model performance
       - change set of inputs
       - repeat procedure,. . .
       - select inputs that yield best performance

3.3 Feature Selection – Cont'd

3.3.1 Max-Dependency, Max-Relevance, Min-Redundancy

- Optimal classification means **minimal classification error**

- In an unsupervised situation (*i.e.*, not using classifiers), minimal error usually requires maximal statistical dependency of the target class on the distribution of space spanned by feature subset = **maximal dependency**

- in Mutual Information terms, maximal dependency:

  $$\max MI(F_1, \ldots, F_k, C)$$

  with $F_1, \ldots, F_k$ features

- When $k = 1$ then feature that maximizes $MI(F_1, C)$

- When $k \neq 1$: assume that we already have $k-1$ features, kth feature is one that leads to largest increase in MI:

  $$MI(F_1, \ldots, F_k, C) = \sum_C \int_{f_1, \ldots, f_k} p(f_1, \ldots, f_k, C) \log \frac{p(f_1, \ldots, f_k, C)}{p(f_1, \ldots, f_k) p(C)}$$

  with f_j projections of data points onto feature F_j

  - Hard to get estimates for multivariate densities $p(f_1, \ldots, f_k, C)$, $p(f_1, \ldots, f_k)$
  - Is computationally slow
  - Even for discrete/categorical case: # joint states quickly ↑

3.3 Feature Selection – Cont'd

3.3.1 Max-Dependency, Max-Relevance, Min-Redundancy – Cont'd

- Approximate Maximum Dependency by **Maximal Relevance**: select k best features with (individually) highest relevance to target class C (= sequential features search)
(relevance defined as correlation or mutual information)

- But k best features $\neq$ best k features

- There could be dependency between k features (**redundancy**): correlation or mutual information between pairs of features could be high

- Removing 1 of 2 mutually redundant features does not change classification error
(& in practice: less parameters $\rightsquigarrow$ better classifier)

- Hence: principle of **minimum Redundancy, maximum Relevance** (mRMR) feature selection (Peng *et al.*, 2005, but exists since Battiti, 1994)

- mRMR is independent from the type of classifier, hence, it can be combined with wrapper (hence, a 2-stage algorithm)

- mRMR software: `http://research.janelia.org/peng/proj/mRMR/`

3.3 Feature Selection – Cont'd

3.3.2 Relevance Filter

- Commonly, several features do not contain any information about target variable (not "relevant"), but which ones?

- Consider **relevance** expressed as mutual information (MI) between feature F_j and class labels C:

$$MI(F_j, C) = \sum_C \int_{f_j} p(f_j, C) \log_2 \frac{p(f_j, C)}{p(f_j)P(C)} \, df_j$$

  with f_j projections of data points onto feature F_j

- Approach = supported by **Data Inequality theorem**:

$$MI(\mathbf{x}, C) \geq MI(F_j(\mathbf{x}), C)$$

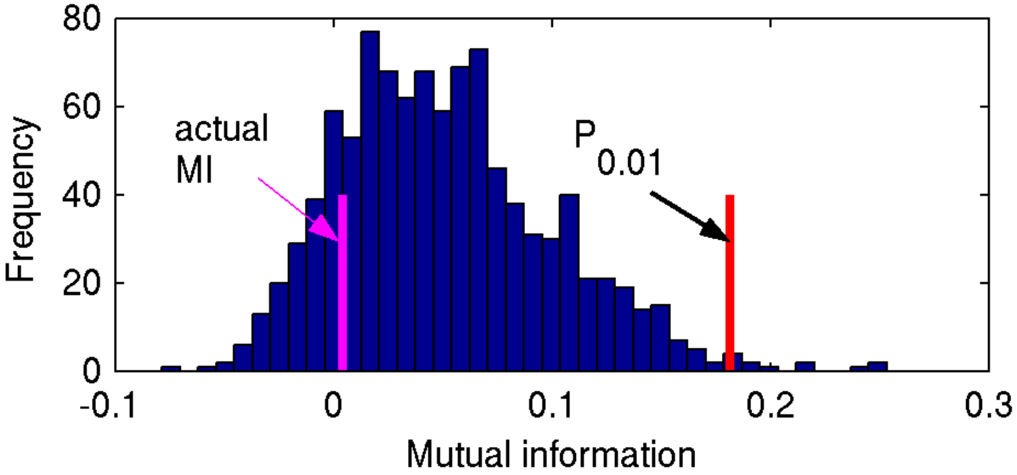
  with $\mathbf{x}$ data point
  Purpose: obtain features that maximize MI after transformation (since transformation can only decrease MI)

3.3 Feature Selection – Cont'd

3.3.2 Relevance Filter – Cont'd:

- **Hence**, one needs to estimate MI:
  MI only estimated from finite sample (real density = unknown)
  but: MI depends on sample size and distribution
  ⇒ based on MI feature relevance = difficult to decide

- **Solution:** *permutation test:*
  compute MI under random permutations of class labels relative to features & repeat random permutation N times
  ⇒ MI distribution for null hypothesis: F_j = irrelevant



Histogram of mutual information obtained from 1000 random permutations. It shows that actual MI $< P_{0.01}$ threshold. Hence, feature is not relevant

3.3 Feature Selection – Cont'd

3.3.3 Redundancy Detection Filter

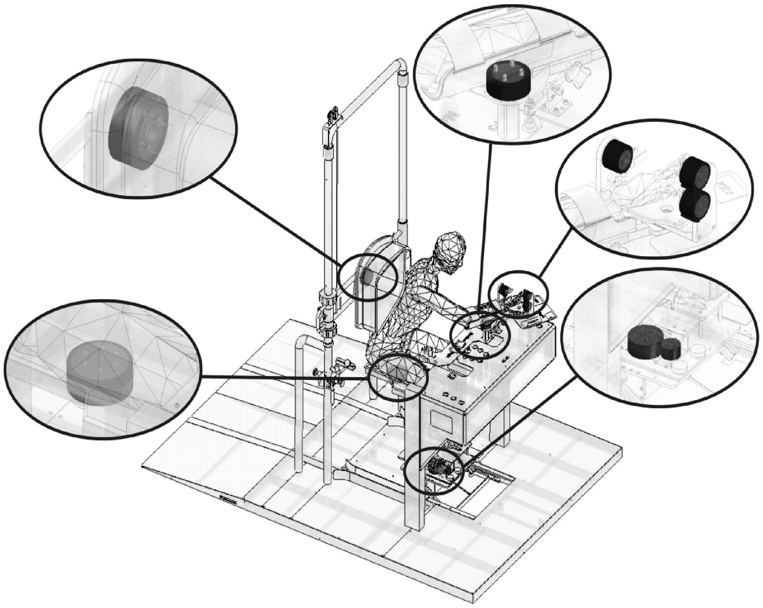
- several **relevant** features may still carry **same** information
- select those for which "distance" is larger than threshold
- distance between features $d(F_i, F_j) = 1$ - normalized MI:

  $$nMI(F_i, F_j) = \frac{2MI(F_i, F_j)}{H(F_i) + H(F_j)} = \frac{2(H(F_i) + H(F_j) - H(F_i, F_j))}{H(F_i) + H(F_j)}$$

  with $H(.)$ (differential) entropy, $0 \leq nMI(F_i, F_j) \leq 1$
  if $nMI(F_i, F_j) = 0 \Rightarrow$ completely *independent* features
  if $nMI(F_i, F_j) = 1 \Rightarrow$ completely *dependent* features
  (note: normalization does not appear in Peng's algorithm)

3.4 Case Study

- Stroke patients have a reduced ability to perform *Activity of Daily Living* (ADL) tasks
- Here 6 ADL tasks: drinking a glass of water, turning a key, picking up spoon, lifting a bag, reaching for a bottle, lifting and carrying a bottle
- To quantify the patient's performance in these tasks, several force and torque sensors are applied to the patient's body.
- Measurements performed in a mechatronic platform

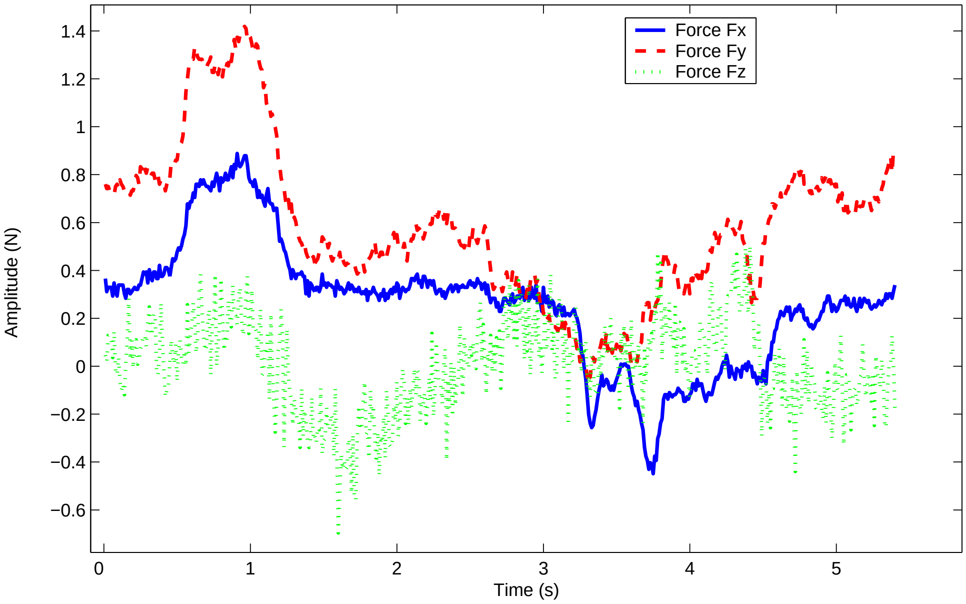


Mechatronic platform. The positions of the different force and torque sensors are shown.

(for more information: Van Dijck, Van Vaerenbergh, & Van Hulle, *Artificial Intelligence in Medicine*, 2009.)

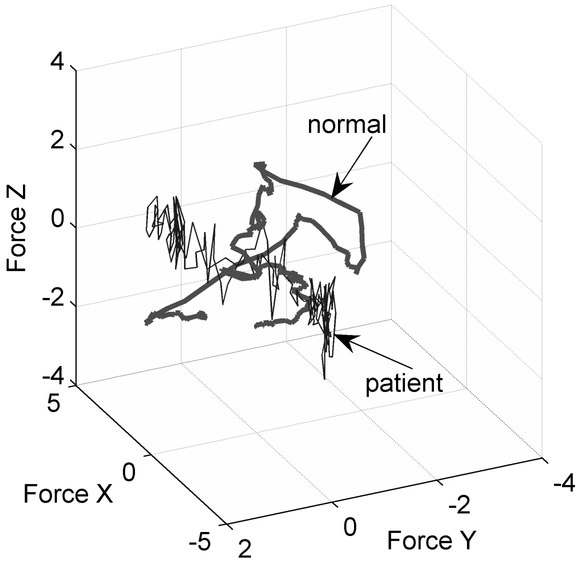
3.4 Case Study – Cont'd

Example: Force trajectories over time for the "drinking a glass"



Forces exerted on a glass when drinking. The forces in X, Y, and Z direction are shown for the thumb. After approximately 0.5 s the patient tries to grasp the glass, as shown by the increased force amplitudes in the X and Y direction.

3.4 Case Study – Cont'd



Representation in 3D. The trajectory is obtained by linking consecutive end-points of the force vectors. The patient trajectory and normal control trajectory can be distinguished by their relative degrees of smoothness: the normal control force trajectory seems smoother, while the patient's trajectory is less smooth.

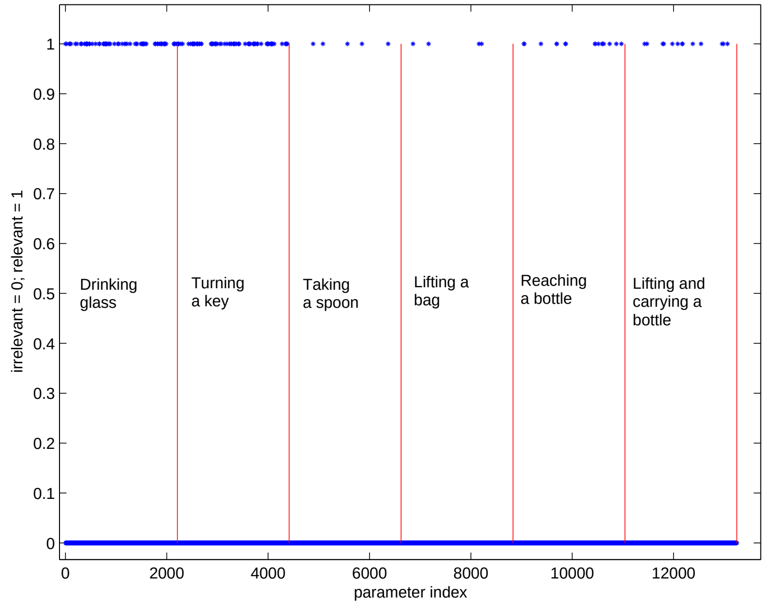
3.4 Case Study – Cont'd

Example of relevance analysis

An exhaustive list of features can be designed based on the time series generated by the sensors (based on trajectory planning, continuity in effort, velocity components, synchronization between sensors, time delay between sensors). This leads to 13248 features! (= huge)

But: which ones characterize difference between patients & normals?

⇒ relevance analysis of features using MI ($P_{0.01}$)



Relevance analysis of features. The feature relevance is set to 1 when it is relevant, otherwise it is set to 0. Only a small subset of 202 of the 13248 original features is relevant. Most of the relevant features can be contributed to "drinking a glass" and "turning a key" tasks.

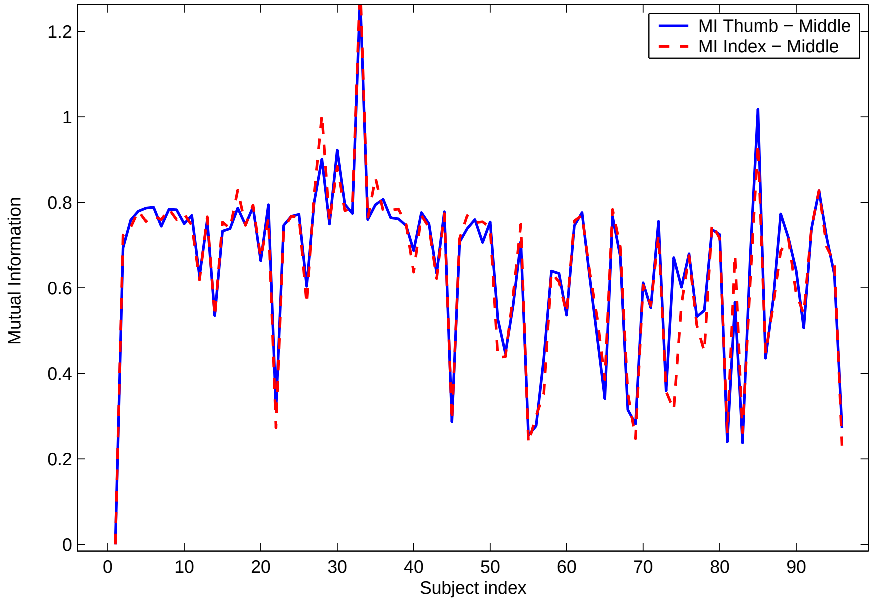
3.4 Case Study – Cont'd

Redundancy Detection Filter

Distance between features $d(F_i, F_j) = 1$ - normalized MI:

$$nMI(F_i, F_j) = \frac{2MI(F_i, F_j)}{H(F_i) + H(F_j)} = \frac{2(H(F_i) + H(F_j) - H(F_i, F_j))}{H(F_i) + H(F_j)}$$

with $H(.)$ (differential) entropy, $0 \leq nMI(F_i, F_j) \leq 1$

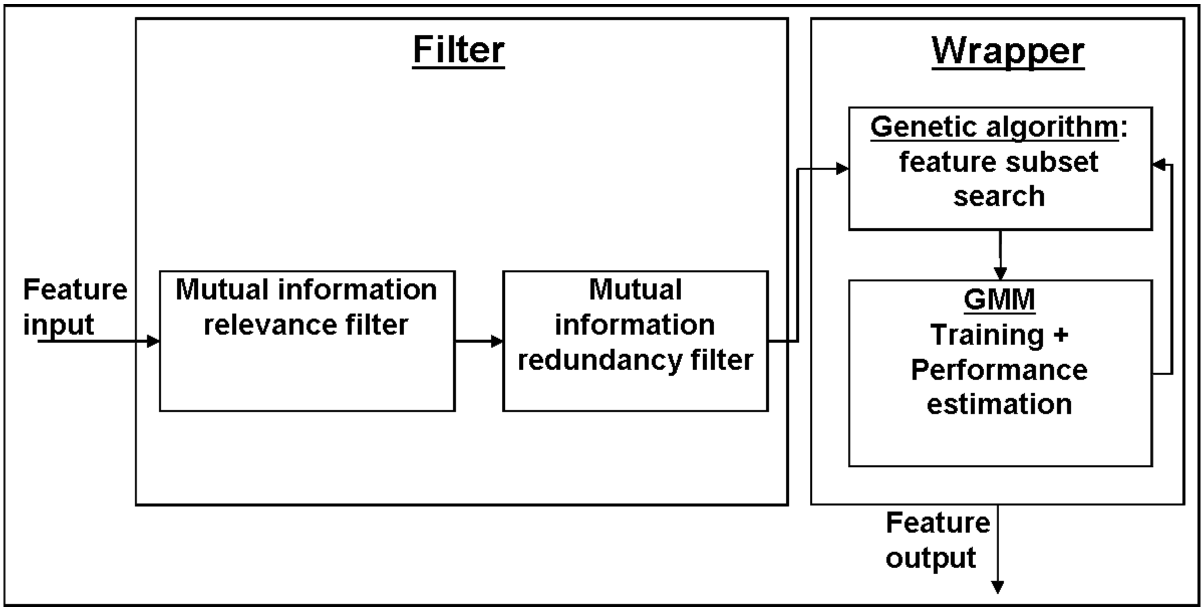


Strongly dependent features. MI between Thumb and Middle finger sensors and Index and Middle finger sensors are the same for different subjects. $nMI(F_i, F_j)$ *between both features is 0.729 (*$d(F_i, F_j) = 0.271$*).*

3.4 Case Study – Cont'd

Example of wrapper

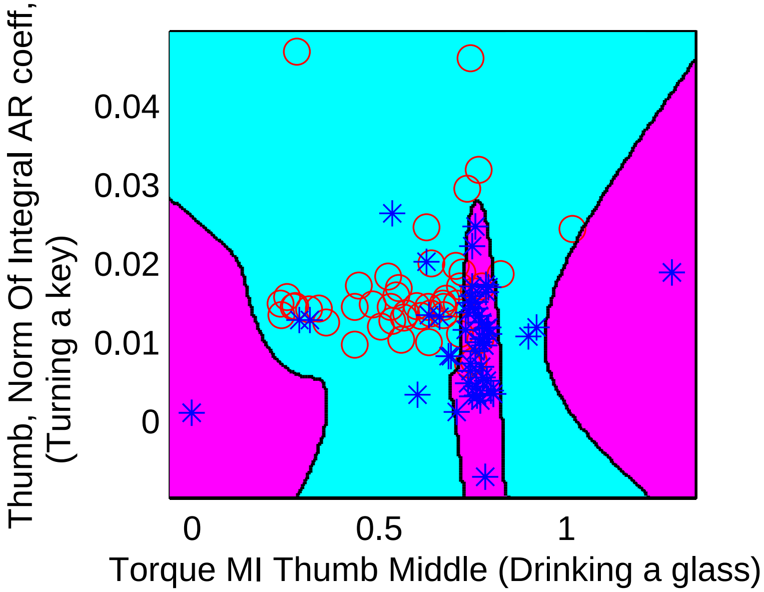
- roulette wheel genetic algorithm
- Bayesian classifier (based on Gaussian Mixture Model)


Filter
Wrapper
Genetic algorithm: feature subset search
Feature input
Mutual information relevance filter
Mutual information redundancy filter
GMM Training + Performance estimation
Feature output

Schematic overview of overall feature subset selection strategy for classification.

3.4 Case Study – Cont'd

Example of wrapper – Cont'd



Decision boundaries computed from a Bayesian classifier. The stroke patients are indicated by "o", the normals by "". Correctness of prediction of patient/normal classification= 85 %.*

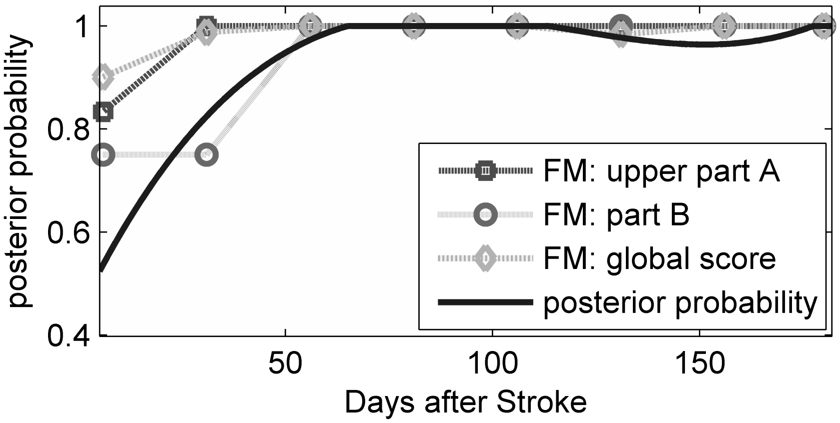
3.4 Case Study – Cont'd

Patient recovery

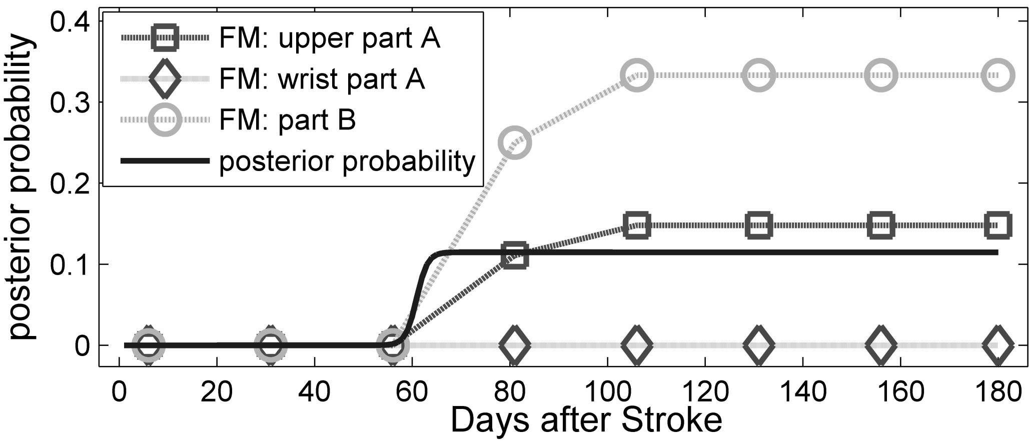
- Based on reduced set of features, patients- and normals densities can be estimated
- Based on those, posterior probabilities of given case to belong to the normals class can be computed
- Recovery of patient can be plotted over time against posterior probability
- Can be compared with Fugl-Meyer (sub)scores
  (Fugl-Meyer = assessment in motor recovery done by experts)

3.4 Case Study – Cont'd

Patient recovery – Cont'd



Class posterior probability profile and subscores of Fugl-Meyer assessment for a subject with fast recovery.



Class posterior probability profile and subscores of Fugl-Meyer assessment for a subject with poor recovery.