



# Dimensionality Reduction by Feature Selection in Machine Learning

---

Author: Dunja Mladenič

J.Stefan Institute, Slovenia

Modified by Xiaoqian Wang

Presentator: Xiaoqian Wang

# Reasons for dimensionality reduction




---

Dimensionality reduction in machine learning is usually performed to:


- Improve the prediction performance
- Improve learning efficiency
- Reduce complexity of the learned results, enable better understanding of the underlying process

# Approaches to dimensionality reduction

Map the original features onto the reduced dimensionality space by:

- **selecting a subset** of the original features 
  - no feature transformation, just select a feature subset
- **constructing features** to replace the original features
  - using methods from statistics, such as, PCA
- using **background knowledge for constructing** new features to be used in addition/instead of the original features (can be followed by feature subset selection)
  - general background knowledge (sum or product of features,...)
  - domain specific background knowledge (parser for text data to get noun phrases, clustering of words, user-specified function,...)

# Example for the problem



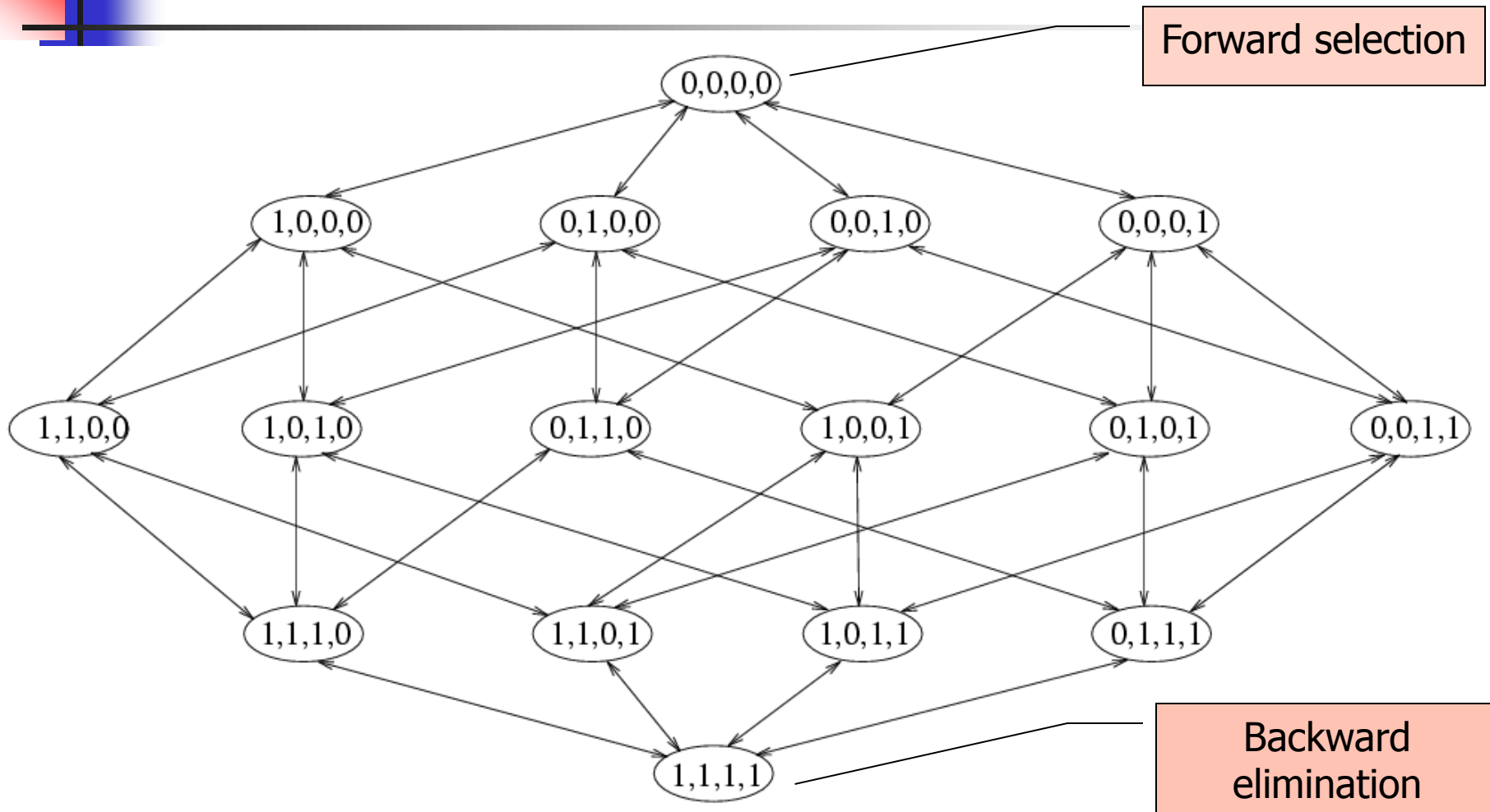
F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	F <sub>4</sub>	F <sub>5</sub>	C
0	0	1	0	1	0
0	1	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
0	0	1	1	0	0
0	1	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1

- Data set
  - Five Boolean features
  - $C = F_1 \vee F_2$
  - $F_3 = \neg F_2$ ,  $F_5 = \neg F_4$
  - Optimal subset:  
 $\{F_1, F_2\}$  or  $\{F_1, F_3\}$
- optimization in space of all feature subsets ( $2^F$  possibilities)

(tutorial on genomics [Yu 2004])

# Search for feature subset

■ An example of search space (*John & Kohavi 1997*)





# Feature subset selection

---

- commonly used search strategies:
  - **forward selection**
    - $F_{\text{Subset}} = \{\}$ ; greedily add features one at a time
  - **forward stepwise selection**
    - $F_{\text{Subset}} = \{\}$ ; greedily add or remove features one at a time
  - **backward elimination**
    - $F_{\text{Subset}} = \text{AllFeatures}$ ; greedily remove features one at a time
  - **backward stepwise elimination**
    - $F_{\text{Subset}} = \text{AllFeatures}$ ; greedily add or remove features one at a time
  - **random mutation**
    - $F_{\text{Subset}} = \text{RandomFeatures}$ ;
    - greedily add or remove randomly selected feature one at a time
    - stop after a given number of iterations



# Approaches to feature subset selection

---

- **Filters** - evaluation function independent of the learning algorithm
- **Wrappers** - evaluation using model selection based on the machine learning algorithm
- **Embedded approaches** - feature selection during learning
- **Simple Filters** - assume feature independence (used for problems with large number of features, eg. text classification)



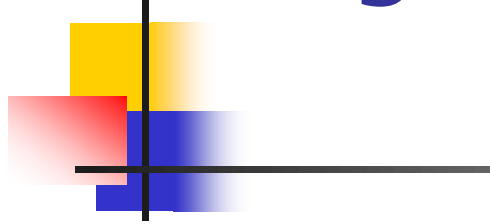
# Approaches to feature subset selection

---

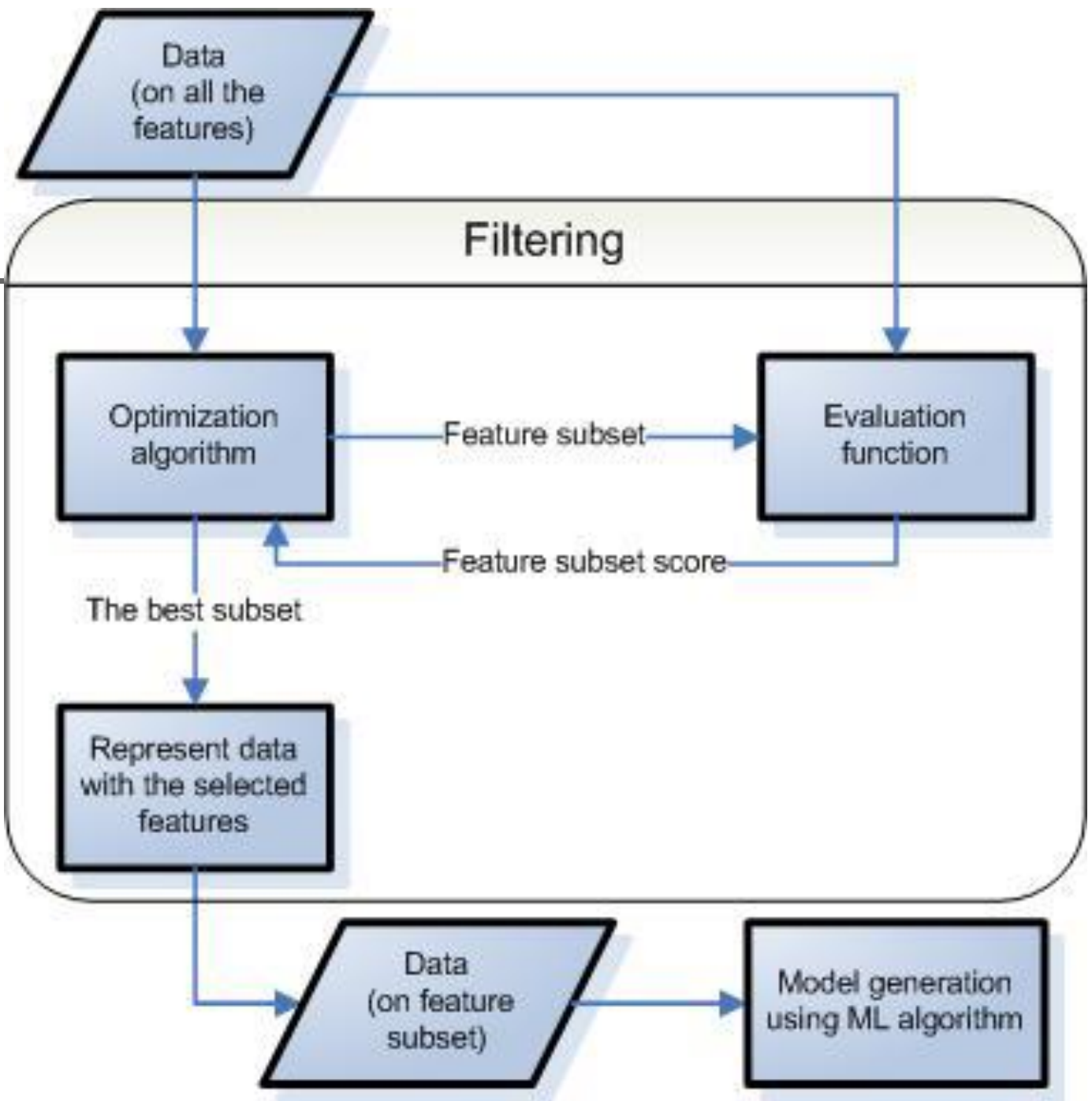
- **Filters** - evaluation function independent of the learning algorithm
- **Wrappers** - evaluation using model selection based on the machine learning algorithm
- **Embedded approaches** - feature selection during learning
- **Simple Filters** - assume feature independence (used for problems with large number of features, eg. text classification)



# Filtering



Evaluation independent of ML algorithm





# Filters: Distribution-based

[Koller & Sahami 1996]

---

Idea: select a minimal subset of features that keeps class probability distribution close to the original distribution  $P(C|\text{FeatureSet})$  is close to  $P(C|\text{AllFeatures})$

1. start with all the features
  2. use backward elimination to eliminate a predefined number of features
- evaluation: the next feature to be deleted is obtained using Cross-entropy measure



# Filters: Relief [Kira & Rendell 1992]

---

## Evaluation of a feature subset

1. represent examples using the feature subset
2. on a random subset of examples calculate average difference in distance from
  - the nearest example of the same class and the nearest example of the different class

$$\delta(\mathbf{Ex}_1, \mathbf{Ex}_2) = \sum_{i=1}^{|\mathbf{F}_{\text{subset}}|} \text{diff}(Ex_1(F_i), Ex_2(F_i))$$

$$\begin{array}{l} \blacksquare \text{ F discrete} \quad \left\{ \begin{array}{l} 0; Ex_1(F) = Ex_2(F) \\ 1; otherwise \end{array} \right\} \quad \text{F cont.} \quad \frac{|Ex_1(F) - Ex_2(F)|}{\max(F) - \min(F)} \end{array}$$

- some extensions, empirical and theoretical analysis in [Robnik-Sikonja & Kononenko 2003]



# Filters: FOCUS [Almallim & Dietterich 1991]

---

## Evaluation of a feature subset

1. represent examples using the feature subset
  2. count conflicts in class value (two examples with the same feature values and different class value)
- Search: all the (promising) subsets of the same (increasing) size are evaluated until a sufficient (no conflicts) subset is found
  - assumes existence of a small sufficient subset --> not appropriate for tasks with many features
  - some extensions of the algorithm use heuristic search to avoid evaluating all the subsets of the same size



# Illustration of FOCUS

F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	F <sub>4</sub>	F <sub>5</sub>	C
0	0	1	0	1	0
0	1	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
0	0	1	1	0	0
0	1	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1



F <sub>4</sub>	F <sub>5</sub>	C
0	1	0
0	1	1
0	1	1
0	1	1
1	0	0
1	0	1
1	0	1
1	0	1

← Conflict!

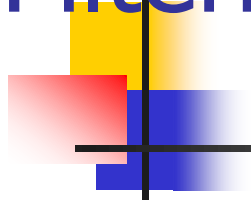
← Conflict!

# Filters: Random [Liu & Setiono 1996]

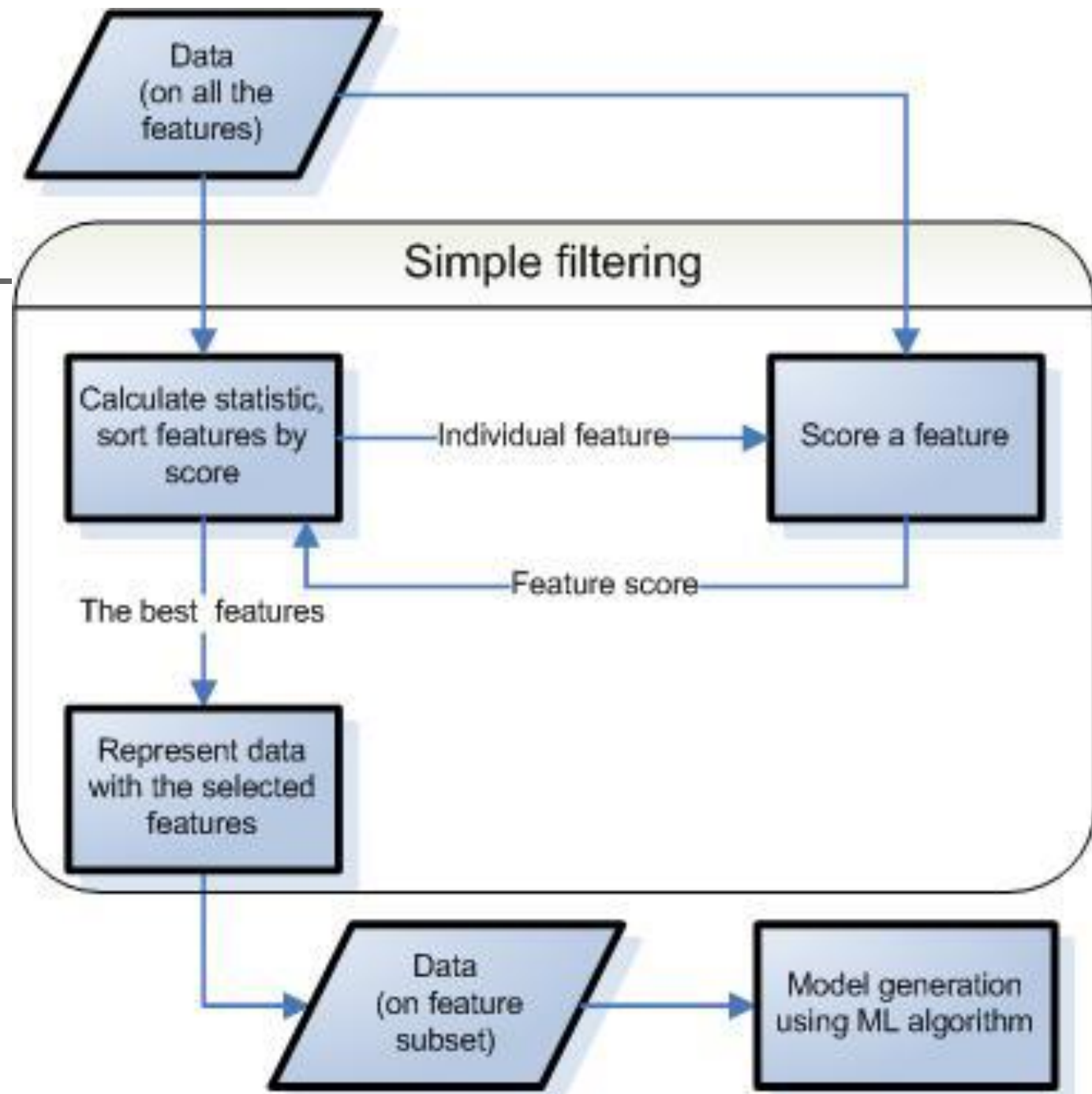
## Evaluation of a feature subset

1. represent examples using the feature subset
  2. calculate the inconsistency rate  
(the average difference between the number of examples with equal feature values and the number of examples among them with the locally, most frequent class value)
  3. select the smallest subset with inconsistency rate below the given threshold
- Search: random sampling to search the space of feature subsets
  - evaluate the predetermined number of subsets
  - noise handling by setting the threshold  $> 0$
  - if threshold = 0, then the same evaluation as in FOCUS

# Simple Filtering



Evaluation independent of ML algorithm





# Feature subset selection on text data – commonly used methods

---

- Simple filtering using some scoring measure to evaluate individual feature
  - supervised measures:
    - information gain, cross entropy for text (information gain on only one feature value), mutual information for text
  - supervised measures for binary class
    - odds ratio (target class vs. the rest), bi-normal separation
  - unsupervised measures:
    - term frequency, document frequency





# Scoring individual feature

---

- InformationGain:  $\sum_{F=W, \bar{W}} P(F) \sum_{C=pos, neg} P(C|F) \log P(C|F)$
- CrossEntropyTxt:  $P(W) \sum_{C=pos, neg} P(C|W) \log P(C|W)$
- OddsRatio:  $\log \frac{P(W | pos) \times (1 - P(W | neg))}{(1 - P(W | pos)) \times P(W | neg)}$
- Frequency:  $Freq(W)$



# Example

From the tutorial of  
<https://www.youtube.com/watch?v=eKD5gxPPeY0>

<b>Day</b>	<b>Outlook</b>	<b>Humidity</b>	<b>Wind</b>	<b>Play</b>
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No



# Approaches to feature subset selection

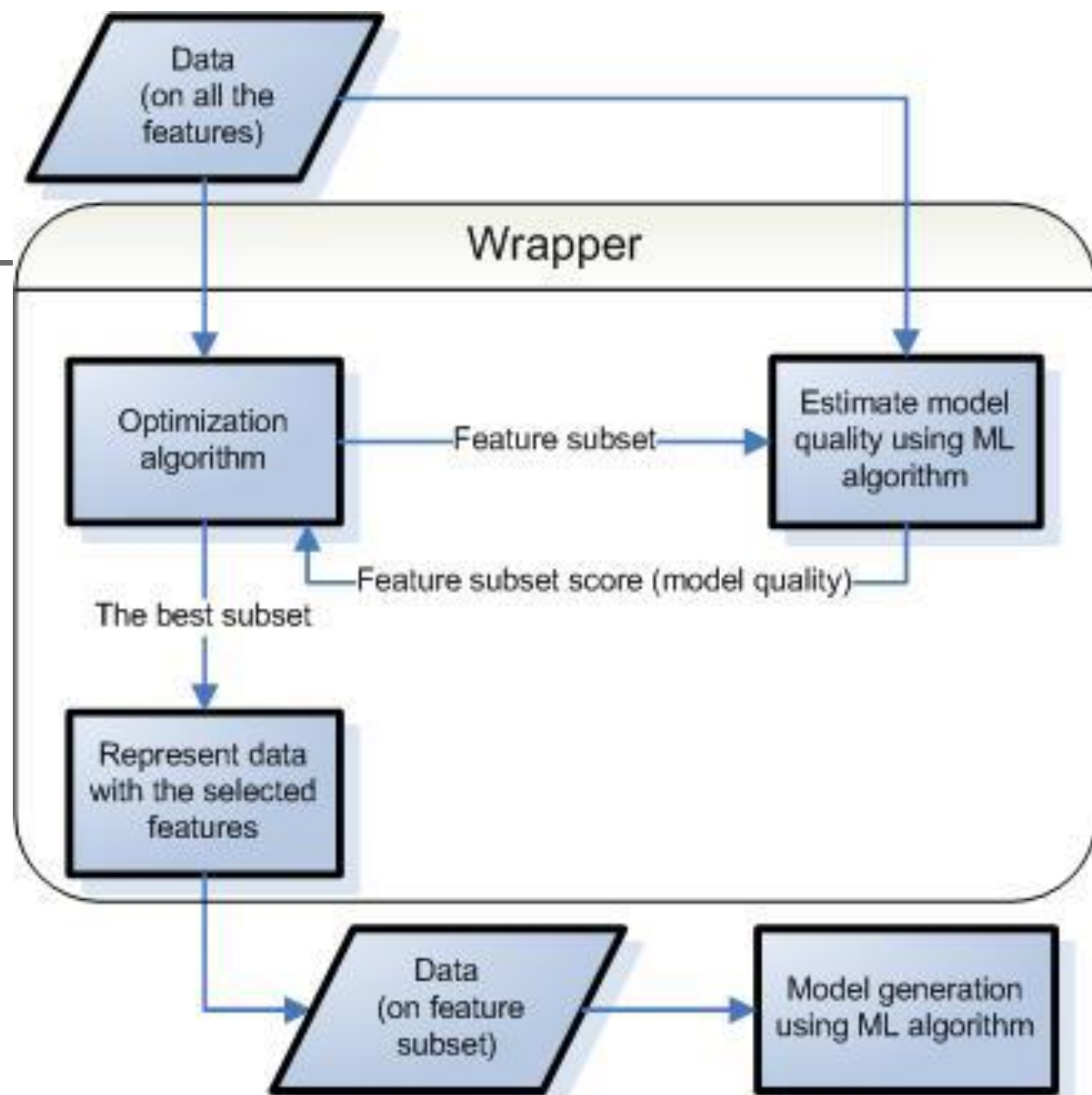
---

- **Filters** - evaluation function independent of the learning algorithm
- **Wrappers** - evaluation using model selection based on the machine learning algorithm
- **Embedded approaches** - feature selection during learning
- **Simple Filters** - assume feature independence (used for problems with large number of features, eg. text classification)

# Wrapper



Evaluation uses the same ML algorithm that is used after the feature selection





# Wrappers: Instance-based learning

---

## Evaluation using instance-based learning

- represent examples using the feature subset
- estimate model quality using cross-validation
  
- Search [Aha & Bankert 1994]
  - start with random feature subset
  - use beam search with backward elimination
- Search [Skalak 1994]
  - start with random feature subset
  - use random mutation



# Wrappers: Decision tree induction

---

Evaluation using decision tree induction

- represent examples using the feature subset
- estimate model quality using cross-validation
- Search [Caruana & Freitag 1994]
  - adding and removing features (backward stepwise elimination)
  - additionally, at each step removes all the features that were not used in the decision tree induced for the evaluation of the current feature subset



# Wrappers: SVM RFE

---

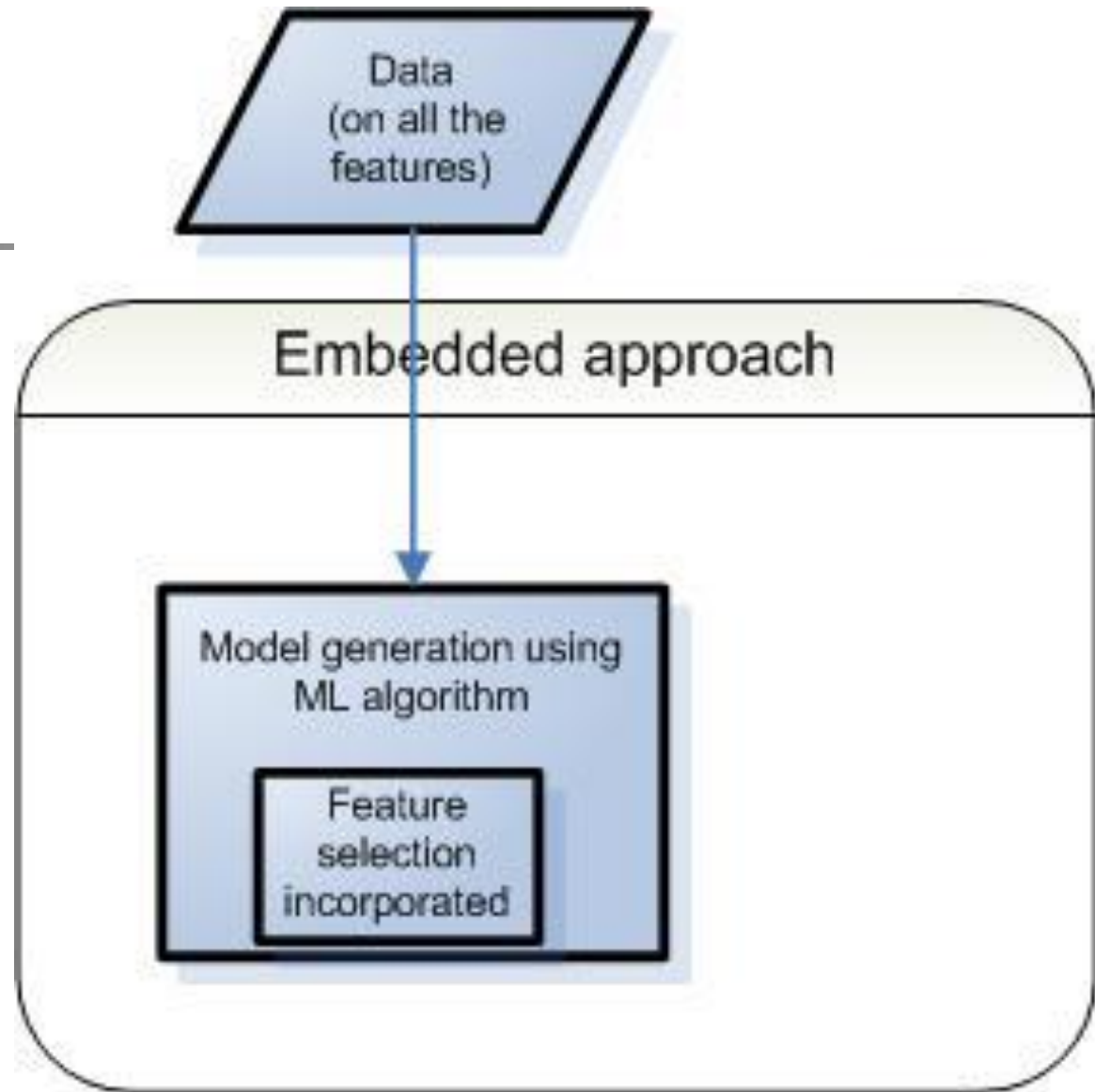
## Evaluation using SVM

- represent examples using the feature subset
- estimate model quality using cross-validation
- SVM Recursive Feature Elimination [Guyon & Weston & Barnhill & Vapnik 2002]
  - remove features according to feature weight in the  $W$  matrix learned by SVM
  - backward elimination: stop until no feature left

# Embedded



Feature selection  
as integral part of  
model generation







# Embedded: using decision tree

---

- Use embedded feature selection as pre-processing [Cardie 1993]
- evaluation and search using the process embedded in decision tree induction
- the final feature subset contains only the features that appear in the induced decision tree
- used for learning using Nearest Neighbor algorithm



# Embedded: using sparsity regularization

---

Select features across all data points with joint sparsity. [Nie & Huang & Cai & Ding 2010]

- The features should have either small scores for all data points or large scores over all data.
- Feature Selection with Joint  $L_{2,1}$ -norm Regularization



# Influence of feature selection on the classification performance

---

- Some ML algorithms are more sensitive to the feature subset than other
  - Naïve Bayes on document categorization sensitive to the feature subset
  - Linear SVM has embedded weighting of features that partially compensates for feature selection

# Discussion

## Using discarded features can help

---

- The features that harm performance if used as input were found to improve performance if used as additional output
- obtain additional information by introducing mapping from the selected features to the discarded features (the multitask learning setting [Caruana de Sa 2003])
  - experiments on synthetic regression and classification problems and real-world medical data have shown improvements in performance

Intuition: transfer of information occurs inside the model, when in addition to the class value it models also additional output consisting of the discarded features



# Discussion

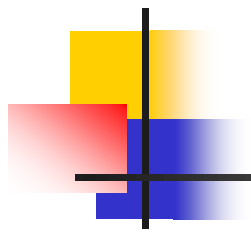
---

## Feature subset selection as pre-processing

- ignore interaction with the target learning algorithm
  - **Simple Filters** – work for large number of features
    - assume feature independence, limited results
    - the size of feature subset to be determined
  - **Filters** – search space of size  $2^F$ , can not handle many features
    - rely on general data characteristics (consistency, distance, class distribution)
- use the target learning algorithm for evaluation
  - **Wrappers** – high accuracy, computationally expensive
    - use model selection with cross-validation of the target algorithm

## Feature subset selection during learning

- use the target learning algorithm during feature selection
  - **Embedded**



---

Thanks!