Merging Classifiers of Different Classification Approaches

Incremental Classification, Concept Drift and Novelty Detection Workshop

Antonina Danylenko[1] and Welf Löwe[1]

antonina.danylenko@lnu.se

14 December, 2014

[1]Linnaeus University, Sweden

Agenda

- Introduction;
- Problem, Motivation, Approach;
- Decision Algebra;
- Merge as an Operation of Decision Algebra;
- Merging Classifiers;
- Experiments;
- Conclusions.

Introduction

- Classification is a common problem that arises in different fields of Computer Science (data mining, information storage and retrieval, knowledge management);
- Classification approaches are often tightly coupled to:
  - learning strategies: different algorithms are used;
  - data structures: represent information in different ways;
  - how common problems are addressed: workarounds;
- It is not that easy to select an appropriate classification model for classification problem (be aware of accuracy, robustness, scalability);

Problem and Motivation

- Simple combining of classifiers learned over different data sets of the same problem is not straightforward;
- Current work is done in aggregation and meta-learning:
  - combine different classifiers learned over same data set;
  - construct single classifier learned on the different variations of the same classification problem;
  - as a result - do not take into account that the context can differ.
- Combining classifiers with partly- or completely- disjoint contexts use one single classification approach for base-level classifiers;
- Generality gets lost: incomparable, difficult benchmarking, hard to propagate advances between domains;

Proposed Approach

- Use Decision Algebra that defines classifiers as re-usable black-boxes in terms of so-called decision functions;
- Define a general *merge* operation over these decisions functions which allows for symbolic computations with classification information captured;
- Show an example of merging classifiers of different classification approaches;
- Show that the merger of classifiers tendentiously becomes more accurate;

Classification Information

- **Classification information** is a set of decision tuples:

$$CI = \{(\vec{a}_1, c_1), \ldots (\vec{a}_n, c_n)\}$$

- It is complete if: $\forall \vec{a} \in \vec{A} : (\vec{a}, c) \in CI$;
- It is non-contradictive if: $\forall (\vec{a}_i, c_i), (\vec{a}_j, c_j) \in CI : \vec{a}_i = \vec{a}_j \Rightarrow c_i = c_j$;
- **Problem domain** $(\mathcal{A}, \mathcal{C})$ of CI is a superset of $\vec{A} \times C$, that defines the actual classification problem, where $\vec{A} \in \mathcal{A}$;

Decision Function

- **Decision Function** is a representation of complete and possibly contradictive decision information:

$$df : \vec{A} \to D(C)$$

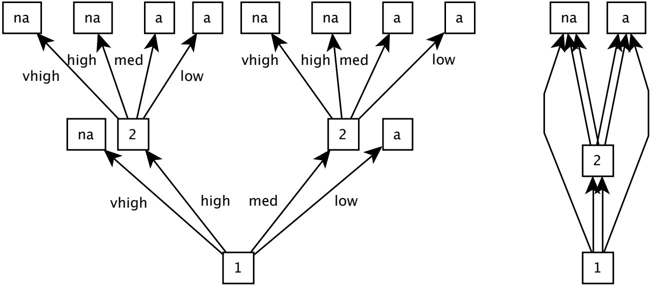
  maps actual context $\vec{a} \in \vec{A}$ to a (probability) distribution $D(C)$;
- It is a higher order (or curried) function:
  $df^n : A_n \to (A_{n-1} \to (\ldots(A_1 \to (\to D(C)))))$;
- Can be easily represented as a decision tree or decision graph:

$$df^n = x^1(df_1^{n-1}, \ldots, df_{|\Lambda_1|}^{n-1})$$

  where Λ_i is a domain of attribute A_1

Graph Representation of Decision Function

- Decision function $df^2 = x^1(na, x^2(na, na, a, a), x^2(na, na, a, a), a)$



Figur: A tree (left) and graph (right) representation of df^2. Each node labeled with n represents a decision term with a selection operator x^n; each square leaf node labled with c corresponds to a probability distribution over classes C with c the most probable class.

Decision Algebra

- (DA) is a theoretical framework that is defined as a parameterized specification, with $\vec{A}$ and $D(C)$ as parameters. It provides a general representation of classification information as an abstract classifier;

Operations Over Decision Functions

- *Constructor* x^n:

$$x^n : \underbrace{\Lambda_1 \times DF[\vec{A}', D] \times \cdots \times \Lambda_1 \times DF[\vec{A}', D]}_{|\Lambda_1| \text{ times}} \rightarrow DF[\vec{A}, D]$$

- *Bind* binds attribute A_i to an attribute value $a \in \Lambda_i$:

$$\begin{aligned} bind_{A_i} \quad &: \quad DF[\vec{A}, D] \times \Lambda_i \rightarrow DF[\vec{A}', D] \\ bind_{A_1} \quad & \quad (x^n(a_1, df_1, \cdots, a_{|\Lambda_1|}, df_{|\Lambda_1|}), a) \equiv df_i, \text{if } a = a_i \\ bind_{A_1} \quad & \quad (df^2, \text{high}) = x^2(na, na, a, a) \end{aligned}$$

- *Evert* changes the order of attributes in the decision function:

$$\begin{aligned} evert_{A_i} \quad &: \quad DF[\vec{A}, D] \rightarrow DF[\vec{A}', D] \\ evert_{A_i}(df) \quad &:= \quad x(a_1, bind_{A_i}(df, a_1), \ldots, \\ & \qquad a_{|\Lambda_i|}, bind_{A_i}(df, a_{|\Lambda_i|})) \\ evert_{A_2}(df^2) \quad &= \quad x^2(x^1(na, na, na, a), x^1(na, na, na, a), \end{aligned}$$

Merge Operation over Decision Functions

- Merge operator $\sqcup_D$ over class distribution $D(C)$;

$$\begin{aligned} \sqcup_D \quad &: \quad D(C) \times D(C) \rightarrow D(C) \\ d(C) \sqcup_D d'(C) = & \quad \{(c, p + p') | (c, p) \in d(C), (c, p') \in d'(C)\} \end{aligned}$$

- **General merge operation over decision functions**:

$$\sqcup : DF_1[\vec{A}, D] \times DF_2[\vec{A}, D] \rightarrow DF'[\vec{A}, D]$$

- Merge over constant decision functions $df_1{}^0, df_2{}^0 \in DF_\emptyset[\{\vec{0}\}, D]$:

$$\sqcup(df_1^0, df_2^0) \quad := \quad x^0(\sqcup_D(df_1^0, df_2^0))$$

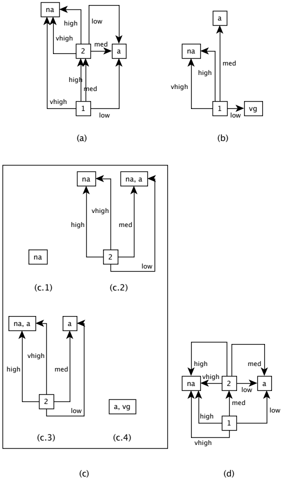
Scenario One: Same Formal Context

- **Prerequisite**: The decision functions $df_1 \in DF_1[\vec{A}, D]$ and $df_2 \in DF_2[\vec{A}', D]$ are constructed over different samples of the same problem domain and $\vec{A} = \vec{A}' = \Lambda_1 \times \ldots \times \Lambda_n$;

$$\begin{aligned} \sqcup(df_1, df_2) := x^n(\quad & a_1, \sqcup(bind_{A_1}(df_1, a_1), bind_{A_1}(df_2, a_1)), \\ & \ldots, \\ & a_k, \sqcup(bind_{A_1}(df_1, a_k), bind_{A_1}(df_2, a_k))) \end{aligned}$$

Scenario One: Cont'd

if $df_1 \in DF_\emptyset[\{\vec{0}\}, D] \wedge df_2 \in DF_\emptyset[\{\vec{0}\}, D]$ **then**
  return $x(\sqcup_D(df_1, df_2))$
end if
for all $a \in \Lambda_1$ **do**
  $df_a = \sqcup(bind_1(df_1, a), bind_1(df_2, a))$
end for
return $x(a_1, df_{a_1}, \ldots, a_{|\Lambda_1|}, df_{a_{|\Lambda_1|}})$



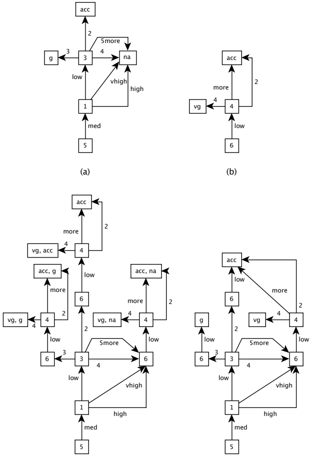
Scenario Two: Disjoint Formal Contexts

- **Prerequisite**: The decision functions $df_1 \in DF_1[\vec{A}, D]$ and $df_2 \in DF_2[\vec{A}', D]$ are constructed over samples with disjoint formal contexts of the same problem domain: $\vec{A} = \Lambda_1 \times \ldots \times \Lambda_n$ and $\vec{A}' = \Lambda'_1 \times \ldots \times \Lambda'_m$ and attributes $\{A_1, \ldots, A_n\} \cap \{A'_1, \ldots, A'_m\} = \emptyset$;

$$
\begin{aligned}
\sqcup(df_1, df_2) := x^n(\quad & a_1, \sqcup(bind_{A_1}(df_1, a_1), bind_{A_1}(df_2, a_1)), \\
& \ldots, \\
& a_k, \sqcup(bind_{A_1}(df_1, a_k), bind_{A_1}(df_2, a_k))) \\
\sqcup(df_1^0, df_2) := \sqcup(\quad & df_2, df_1^0)
\end{aligned}
$$

Scenario Two: Cont'd

if $df_1 \in DF_\emptyset[\{\vec{0}\}, D] \wedge df_2 \in DF_\emptyset[\{\vec{0}\}, D]$ **then**
  return $x(\sqcup_D(df_1, df_2))$
end if
if $df_1 \in DF_\emptyset[\{\vec{0}\}, D]$ **then**
  return $\sqcup(df_2, df_1)$
end if
for all $a \in \Lambda_1$ **do**
  $df_a = \sqcup(bind_1(df_1, a), bind_1(df_2, a))$
end for
return $x(a_1, df_{a_1}, \ldots, a_{|\Lambda_1|}, df_{a_{|\Lambda_1|}})$



(a) (b) (c) (d)

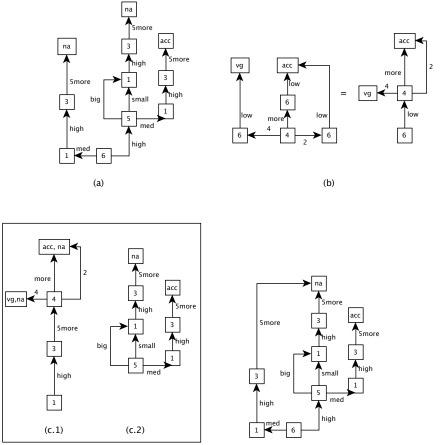
Scenario Three: General Case

- **Prerequisite**: For this general case, scenarios one and two are just special cases. The decision functions $df_1 \in DF_1[\vec{A}, D]$ and $df_2 \in DF_2[\vec{A}', D]$ are constructed over samples with arbitrary formal contexts of the same problem domain: $\vec{A} = \Lambda_1 \times \ldots \times \Lambda_n$ and $\vec{A}' = \Lambda'_1 \times \ldots \times \Lambda'_m$;

$$
\begin{aligned}
\sqcup(df_1, df_2) := x^n(\quad & a_1, \sqcup(bind_{A_1}(df_1, a_1), bind_{A_1}(df_2, a_1)),\\
& \ldots,\\
& a_k, \sqcup(bind_{A_1}(df_1, a_k), bind_{A_1}(df_2, a_k)))\\
\sqcup(df_1^0, df_2) := \sqcup(\quad & df_2, df_1^0)\\
\sqcup(df_1, df_2) := \sqcup(\quad & df_1, evert_{A_1}(df_2)) \text{ iff } A_1 \in \{A'_2, \ldots, A'_m\}
\end{aligned}
$$

Scenario Three: Cont'd

if $df_1 \in DF_\emptyset[\{\vec{0}\}, D] \wedge df_2 \in DF_\emptyset[\{\vec{0}\}, D]$ **then**
  return $x(\sqcup_D(df_1, df_2))$
end if
if $df_1 \in DF_\emptyset[\{\vec{0}\}, D]$ **then**
  return $\sqcup(df_2, df_1))$
end if
if $A_1 \neq A_1' \wedge A_1 \in \{A_2', \ldots, A_m'\}$ **then**
  return $\sqcup(df_1, evert_{A_1}(df_2))$
end if
for all $a \in \Lambda_1$ **do**
  $df_a = \sqcup(bind_1(df_1, a), bind_1(df_2, a))$
end for
return $x(a_1, df_{a_1}, \ldots, a_{|\Lambda_1|}, df_{a_{|\Lambda_1|}})$



(a) (b) (c.1) (c.2) (c) (d)

Accuracy of the Merged Decision Functions

- Decision function df_1 is more accurate than a decision function df_2 iff it more often gives the "right" classification based on some ground truth (which is usually not known);
- $oracle_{\vec{a}} : C \to \mathbb{R}$ is the accurate classification probability distribution;
- $oracle : \vec{A} \to D(C)$ is an accurate decision function with $\forall \vec{a} \in \vec{A} : oracle(\vec{a}) = oracle_{\vec{a}}$;
- $df : \vec{A} \to D(C)$ is **probably accurate** with respect to *oracle* iff $\forall \vec{a} \in \vec{A} : df(\vec{a})$ is a random sample of $oracle_{\vec{a}}$;
- **Theorem:** Let $df_1, \ldots, df_n$ be a series of independently learned decision functions $df : \vec{A} \to D(C)$ that are probably accurate with respect to an accurate decision function $oracle : \vec{A} \to D(C)$. For large n, the merged decision function $df_1 \sqcup \ldots \sqcup df_n$ converges in probability to the *oracle*.

Naïve Bayesian Classifiers

- **Constructor:**

$$nb^n \quad : \quad D(C) \times \underbrace{PD_1^1 \times \cdots \times PD_n^1 \times \ldots \times PD_1^k \times \cdots \times PD_n^k}_{n \times k \text{ conditional probability distributions}} \rightarrow NB[\vec{A}, D].$$

- Probability distribution functions : $PD_i^j \widehat{=} PD(\Lambda_i | C = c_j)$;
- **Bind** operation: $bind_{A_i} : NB[\vec{A}, D] \times \Lambda_i \rightarrow NB[\vec{A}', D]$

$$\begin{aligned}
bind_{A_i}(df^n, a) \quad &:= \quad nb^{n-1}(cons_D((c_1, prob_{c_1,i}(df^n, a)), \ldots (c_k, prob_{c_k,i}(df^n, a))), \\
&\quad\quad pd_1(df^n, c_1), \ldots, pd_{i-1}(df^n, c_1), pd_{i+1}(df^n, c_1), \ldots, pd_n(df^n, c_1), \\
&\quad\quad \ldots \\
&\quad\quad pd_1(df^n, c_k), \ldots, pd_{i-1}(df^n, c_k), pd_{i+1}(df^n, c_k), \ldots, df_n^n(df^n, c_k)) \\
prob_{c,i}(df, a) \quad &= \quad prob(dist(df), c) \cdot prob(pd_{A_i}(df, c), a)
\end{aligned}$$

Bind: Example

Class: Car Acceptability

Accept	Not accept
0.69	0.31

Buying price

	high	low
Accept	0.31	0.69
Not accept	0.14	0.86

Maintenance price

	high	low
Accept	0.26	0.74
Not accept	0.68	0.32

⇓

bind $_{Bying}$ (NB, high)

Class: Car Acceptability

Accept	Not accept
0.69 * 0.31	0.31 * 0,14

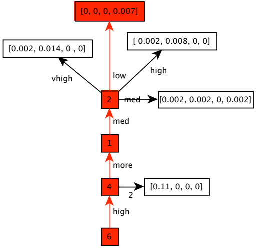
Maintenance price

	high	low
Accept	0.26	0.74
Not accept	0.68	0.32

Merging of Naïve Bayesian Classifiers

$$
\begin{array}{rcl}
\sqcup_{NB} & : & NB[\vec{A}, D] \times NB[\vec{A}', D] \rightarrow NB[\vec{A}'', D] \\
\sqcup_{NB}(df, df') & := & nb^n(\sqcup_D(dist(df), dist(df')), \\
& & pd_{A_1}(df, c_1), \ldots, pd_{A_i}(df, c_1)), pd_{A_1'}(df', c_1), \ldots, pd_{A_j'}(df', c_1)) \\
& & \sqcup_D(pd_{A_1''}(df, c_1), pd_{A_1''}(df', c_1)), \ldots, \sqcup_D(pd_{A_l''}(df, c_1), pd_{A_l''}(df', c_1)), \\
& & \ldots \\
& & pd_{A_1}(df, c_k), \ldots, pd_{A_i}(df, c_k)), pd_{A_1'}(df', c_k), \ldots, pd_{A_j'}(df', c_k)) \\
& & \sqcup_D(pd_{A_1''}(df, c_k), pd_{A_1''}(df', c_k)), \ldots, \sqcup_D(pd_{A_l''}(df, c_k), pd_{A_l''}(df', c_k)) \\
pd_{A_i} & : & NB[\vec{A}, D] \times C \rightarrow D(A_i)
\end{array}
$$

Scenario One: Same Formal Context



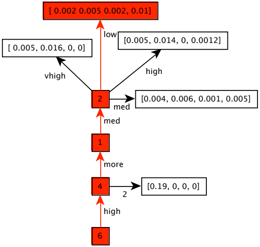
Class: [0.69, 0.23, 0.04, 0.03]

safety (6) = high: [0.24, 0.57, 0.44, 1]

per cap (4) = more: [0.29, 0.47, 0.5, 0.53]

buying (1) = med: [0.21, 0.28, 0.38, 0.6]

maint (2) = low: [0.22, 0.29, 0.69, 0.53]

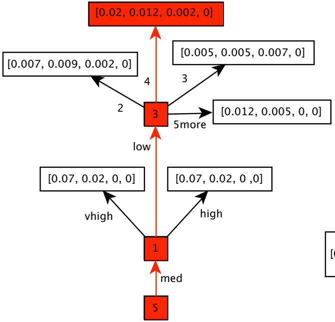


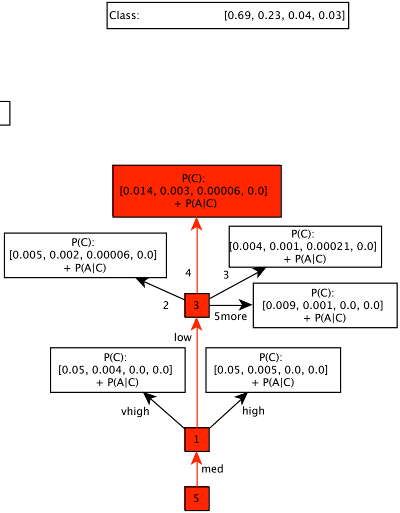
Scenario Two: Disjoint Contexts

Replace $\sqcup(df^0_1, df_2) := \sqcup(df_2, df^0_1)$ with:

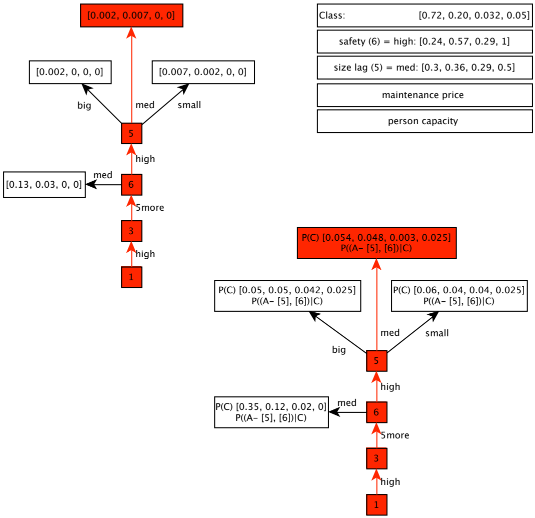
$$
\begin{aligned}
\sqcup_H \quad &: \quad DF^0[\{\vec{0}\}, D] \times NB[\vec{A}, D] \rightarrow NB[\vec{A}, D] \\
df^0_{dg} \sqcup_H df_{nb} \quad &:= \quad nb(dist(df^0_{dg}) \sqcup_D dist(df_{nb}), \\
&\qquad pd_{A_1}(df, c_1), \ldots, pd_{A_n}(df, c_1) \ldots, pd_{A_1}(df, c_k), \ldots, pd_{A_n}(df, c_k))
\end{aligned}
$$

Scenario Two: Cont'd

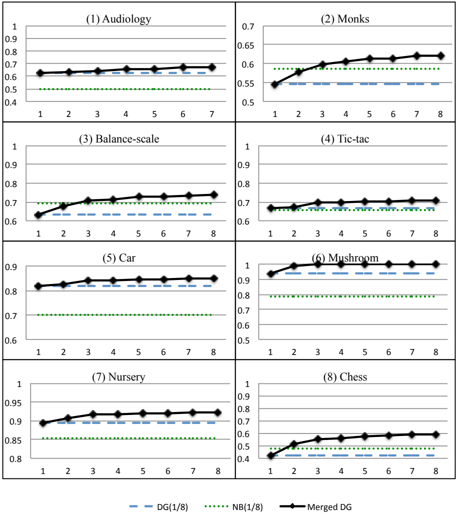




Scenario Three: General



Experiments



Conclusions

- *Merge* $\sqcup$ operation over decision functions is a general way to combine classifiers;
- Decision Algebra allows applying *merge* implementing:
  - a single core-operation *bind* over classifiers defined as decision functions;
  - a $\sqcup_D$ over co-domain of decision functions (usually represented as distributions);
- We showed that merging of a series of probably accurate decision functions results an more accurate decision function (experiments: 2.7% - 17%).

Thank you for your attention. Questions?