Feature Selection as a One-Player Game

Romaric Gaudel ROMARIC.GAUDEL@LRI.FR
Michèle Sebag MICHELE.SEBAG@LRI.FR
LRI, CNRS UMR 8623 & INRIA-Saclay, Bât 490, Université Paris-Sud 11, 91405 Orsay Cedex FRANCE

Abstract

This paper formalizes Feature Selection as a Reinforcement Learning problem, leading to a provably optimal though intractable selection policy. As a second contribution, this paper presents an approximation thereof, based on a one-player game approach and relying on the Monte-Carlo tree search UCT (Upper Confidence Tree) proposed by Kocsis and Szepesvari (2006). The *Feature Uct SElection* (FUSE) algorithm extends UCT to deal with i) a finite *unknown* horizon (the target number of relevant features); ii) the huge branching factor of the search tree, reflecting the size of the feature set. Finally, a frugal reward function is proposed as a rough but unbiased estimate of the relevance of a feature subset. A proof of concept of FUSE is shown on benchmark data sets.

1. Introduction

Feature Selection (FS), one key issue in statistical learning, is a combinatorial optimization problem aimed at minimizing the generalization error of the learned hypothesis. FS is tackled using three main approaches: scoring, wrapping and embedded FS (more in section 2). Scoring approaches independently assess and rank the features w.r.t. the classification problem; in counterpart, they poorly account for the feature inter-dependencies. Wrapper methods basically tackle the whole combinatorial optimization problem, exploring the whole powerset of the feature set and computing for each candidate subset an estimate of the generalization error. Embedded approaches exploits the learned hypothesis and/or incorporate sparsity criteria during learning to achieve FS.

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

This paper formalizes Feature Selection as a Reinforcement Learning (RL) problem: find a good feature selection strategy, ultimately resulting in a minimal generalization error. While this formalization enables to define a provably optimal policy, such a policy is clearly intractable. An approximation thereof is obtained by casting the RL problem as a one-player game; the Upper Confidence Tree (UCT) framework (Kocsis & Szepesvári, 2006) yields a robust approach to optimization under uncertainty.

The paper is organized as follows: section 2 briefly reviews related work, without aiming at exhaustivity. The formalization of FS as a Reinforcement Learning problem is presented in section 3. For the sake of self-containedness, section 4 describes the UCT framework. Section 5 focuses on how to deal with the huge branching factor of the search tree, while section 6 presents other UCT extensions needed to handle the Feature Selection problem, and gives an overview of the FUSE (**F**eature **U**CT **SE**lection) algorithm. A proof of principle of the approach on some benchmark problems from the NIPS 2003 FS Challenge (Guyon et al., 2004) is discussed in section 7 and the paper concludes with some perspectives for further work.

2. State of the Art

Feature Selection mostly aims at reducing the generalization error of the learned hypothesis, through removing noise (filtering out irrelevant features) or considering simpler hypothesis spaces (filtering out redundant features). As already mentioned, three categories of FS algorithms are classically distinguished.

Filtering approaches proceed by independently ranking features using some score function, and selecting the top-ranked ones. The score basically measures the correlation between the feature and the class values, using e.g. *Student's t-test*, or its multi-class extension (aka. ANOVA, the *Analysis of variance*), or the *Fisher test*. The main limitation of the above score functions is to discard feature interdependencies. A

filtering approach accounting for feature correlations, RELIEF measures the feature impact on each example neighborhood and its purity (Kira & Rendell, 1992); the example neighborhood, defined from the Euclidean distance built from all features, might however be biased due to feature redundancy.

Wrapper approaches tackle the whole combinatorial optimization problem of FS, exploring the powerset of the feature set $\mathcal{F}$ and measuring (an estimate of) the generalization error of all considered subsets. The crux of wrapper approaches is how to navigate in the feature powerset lattice, and deal with the exploration versus exploitation (EvE) dilemma. On the one hand, exploration must be limited for the sake of scalability. On the other hand, exploitation aka. myopic search is bound to end up in a local optimum. Hill-climbing approaches (Hall, 2000), mixing forward and backward selection (Zhang, 2008a), using look-ahead or making large steps in the lattice (Margaritis, 2009), combining global and local search operators (Boullé, 2007), among many other heuristics, have been proposed to address the EvE dilemma.

Embedded Approaches combine learning and feature selection through prior or posterior regularization. (Tibshirani, 1994) pioneered the use of L_1 regularization, amenable to Linear Programming-based learning (Fung & Mangasarian, 2000) and paving the way for the so-called *least absolute shrinkage and selection operator* (Lasso) approach (Efron et al., 2004). Quite a few other regularization approaches, mostly considering linear hypothesis spaces, have been considered along the same lines (Guyon et al., 2002; Chan et al., 2007; Zhang, 2008b; Helleputte & Dupont, 2009). An extension of the above to non-linear spaces is given by Multiple Kernel Learning (MLKL) (Bach, 2008), achieving a layer-wise exploration of the "kernel lattice" as opposed to the feature lattice. While the kernel lattice indeed offers a more compact and hence more tractable search space than the feature powerset, the approach clearly cannot be extended beyond certain limits. Other approaches use non-linear hypotheses (Shen et al., 2008; Varma & Babu, 2009) or random forests (Breiman et al., 1984; Rogers & Gunn, 2005), to compute feature scores. The main weakness of these scores in a FS perspective is to be hindered by feature redundancy.

In summary, filtering and embedded approaches alike consider each feature in a fixed context: either standalone (and they do not capture their correlations) or together with all other features (and they are hindered by feature redundancy). Only wrapper approaches, considering feature subsets, have a chance to see features at their best; but the price to pay is that of considering sufficiently many subsets. Regarding wrapper approaches, the delicate issue is to control the Exploration versus Exploitation (EvE) tradeoff. It thus comes naturally to wonder whether: i/ a feature subset can be provided with a cheap, unbiased, assessment; ii/ these assessments can be used to control the EvE tradeoff in an efficient way, and gradually focus the search on the most promising regions of the feature lattice. Both questions are at the root of the proposed Multi-Armed Bandit-based approach to feature selection.

3. Feature Selection as Reinforcement Learning

The first contribution of the presented work is to formalize FS as a Markov Decision Process (MDP) in the feature set lattice, as follows. Let $\mathcal{F}$ denote the finite set of features, plus an additional *stopping* feature noted f_s. The state space $\mathcal{S}$ of the MDP is the powerset of $\mathcal{F}$. Final states are all states $F \subseteq \mathcal{F}$ containing f_s. The action space, likewise, is the set of features. Within any non final state F, a possible action is to select a feature in $\mathcal{F} \setminus F$: letting $p : \mathcal{S} \times \mathcal{F} \times \mathcal{S} \to I\!R^+$ be the transition function, $p(F, f, F')$ is non zero if $F' = F \cup \{f\}$.

As the main goal of FS is to yield a minimal generalization error at the end of the learning process, it comes naturally that the reward function associated to a final state F is the generalization error of the learned hypothesis $\mathcal{A}(F \setminus \{f_s\})$ noted $\mathbf{Err}(\mathcal{A}(F))$. Let π be an MDP policy associating to each state an action. Let F_π denote the final state (the subset of features) built by following π starting from the empty state. Clearly the optimal policy $\pi^\star$ is the one minimizing $\mathbf{Err}(\mathcal{A}(F_\pi))$:

$$\pi^\star = \operatorname*{argmin}_{\pi} \mathbf{Err}\left(\mathcal{A}\left(F_\pi\right)\right) \tag{1}$$

Following Bellman's optimality principle (Bellman, 1957) and defining the optimal value function $V^\star$ as:

$$V^\star(F) = \begin{cases} \mathbf{Err}(\mathcal{A}(F)) & \text{if } F \text{ is final} \\ \min\limits_{f \in \mathcal{F} \setminus F} V^\star(F \cup \{f\}) & \text{otherwise} \end{cases} \tag{2}$$

the optimal policy $\pi^\star$ is defined from $V^\star$ as follows:

$$\pi^\star(F) = \operatorname*{argmin}_{f \in \mathcal{F} \setminus F} V^\star(F \cup \{f\}) \tag{3}$$

While the above $\pi^\star$ policy is provably optimal, it does not lead to a tractable algorithm, as the state space

is exponential in the number of features. The second contribution of the paper is to provide a tractable approximation of the above optimal policy, taking inspiration from (Rolet et al., 2009). The proposed policy approximation relies on the Upper Confidence Tree (UCT) framework (Kocsis & Szepesvári, 2006) which extends the optimal exploration vs. exploitation tradeoff of the Upper Confidence Bound algorithm (UCB) (Auer et al., 2002) to the case of sequential decision making.

4. Upper Confidence Tree for Optimal Policy Approximation

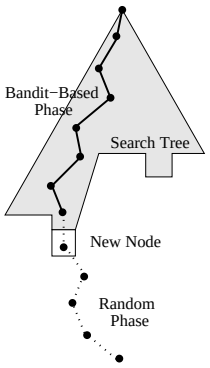


Figure 1. One iteration of FUSE (UCT approach for FS).

For the sake of self-containedness this section briefly summarizes UCT, a Monte-Carlo tree search algorithm asymmetrically growing the search tree to explore the most promising regions thereof (Fig. 1). UCT proceeds by successive iterations where each iteration executes a sequence of actions divided into two phases: a bandit-based phase and a random phase. Within the bandit phase, in each node F, one selects the action maximizing the Upper Confidence Bound (UCB) criterion (Auer et al., 2002):

$$a^\star = \underset{a \in A}{\text{argmax}} \left\{ \hat{\mu}_{F,a} + \sqrt{\frac{c_e \ln(T_F)}{t_{F,a}}} \right\} \qquad (4)$$

where T_F stands for the number of times node F has been visited, $t_{F,a}$ denotes the number of times a has been selected in node F, and $\hat{\mu}_{F,a}$ is the average reward collected when selecting action a from node F. Parameter c_e (set to 2 in the original UCB formula) was introduced in UCT to control the exploration strength (Kocsis & Szepesvári, 2006).

At some point, one arrives at a node F which does not belong to the search tree. This node is added to the tree and the iteration switches to the random phase: new actions are selected either randomly or following ad-hoc rules until arriving at a final node. At this point, the instant reward is computed and the T_F, $t_{F,a}$ and $\hat{\mu}_{F,a}$ indicators attached to each node and action of the visited branch are updated accordingly.

5. The Many-armed Issue

A general limitation of Multi-Arm Bandit algorithms, when dealing with a large number of actions compared to the number of allowed iterations, is to be biased toward exploration (Wang et al., 2008). This limitation is even more severe for UCT, which must resist over-exploration at each level of the tree. This section recalls several heuristics proposed to control exploration and details how these are combined in the proposed approach.

5.1. Restricting the Number of Arms

A first modification of the original UCT algorithm is to limit the strength of the exploration term, taking into account the empirical variance $\hat{\sigma}^2_{F,a}$ of the rewards collected when selecting action a from node F. Formally, Eq. (4) is replaced by the so-called UCB1-tuned equation (Auer et al., 2002):

$$\hat{\mu}_{F,a} + \sqrt{\frac{c_e \ln(T_F)}{t_{F,a}} \min\left(\frac{1}{4}, \hat{\sigma}^2_{F,a} + \sqrt{\frac{2\ln(T_F)}{t_{F,a}}}\right)} \qquad (5)$$

Two heuristics, a discrete and a continuous one, have been considered to further control the exploration. The continuous one proposed by (Gelly & Silver, 2007) sets constant c_e to a very small value; the discrete one proposed by (Coulom, 2006) restricts the number of considered child nodes depending on the number of visits T_F to the parent node. Formally, a new child node is added whenever the integer part of T_F^b is incremented by 1, where $b < 1$ is a parameter of the algorithm.

5.2. Rapid Action Value Estimation

As shown in earlier work, the selection of new nodes can benefit from any prior knowledge (Rolet et al., 2009) or any knowledge gained within the search. Taking inspiration from MoGo (Gelly & Silver, 2007), knowledge gradually acquired along the search is summarized through the so-called *Rapid Action Value Estimation* vector (RAVE), associating to each feature f the average reward of all final nodes F_t containing f:

$$\text{g-RAVE}_f = average\{V(F_t), f \in F_t\} \qquad (6)$$

While g-RAVE provides a global indication on feature relevance, it also makes sense to consider the feature conditionally to those selected within the current node,

yielding the ℓ-RAVE vector:

$$\ell\text{-RAVE}_{F,f} = average\{V(F_t), F \rightsquigarrow F_t, f \in F_t\} \quad (7)$$

where the average is taken over iterations t visiting node F and ending up in final node F_t, noted $F \rightsquigarrow F_t$.

Regarding the stopping feature f_s, one RAVE score is associated to each size of feature set. Specifically, let $f_s^{(d)}$ be the stopping feature at level d; g-RAVE$_{f_s^{(d)}}$ is defined as the average reward of all final nodes F_t of size $d+1$:

$$\text{g-RAVE}_{f_s^{(d)}} = average\{V(F_t), |F_t| = d+1\} \quad (8)$$

Global and local scores provide an estimate of the feature relevance (conditionally to feature subset F in the case of ℓ-RAVE and $\hat{\mu}_{F,f}$) achieving different bias vs. variance trade-offs: g-RAVE is computed from more trials than ℓ-RAVE, which in turn is computed from more trials than $\hat{\mu}_{F,f}$; but $\hat{\mu}_{F,f}$ (respectively ℓ-RAVE) more precisely reflects the impact of f conditionally to F than ℓ-RAVE (resp. g-RAVE).

5.3. Selection of New Nodes

RAVE scores are used to focus the search and avoid a (hopeless) uniform exploration of the feature space. How to use RAVE depends on whether the exploration heuristics (section 5.1) is discrete or continuous. The discrete one selects the top ranked feature after RAVE whenever the integer part of T_F^b is incremented. The continuous one replaces the UCB formula (Eq. (5)) by a weighted sum $(1-\alpha)\cdot \text{UCB} + \alpha \cdot \ell\text{-RAVE}$ where $\alpha = \frac{c}{c+t_{F,f}}$ controls the impact of ℓ-RAVE, the weight of which becomes small as soon as f has been sufficiently selected from node F ($t_{F,f} > c$) (Gelly & Silver, 2007). Along the same lines, we consider both g-RAVE and ℓ-RAVE to select the feature f maximizing:

$$(1-\alpha)\cdot\hat{\mu}_{F,f} + \alpha\left((1-\beta)\cdot \ell\text{-RAVE}_{F,f} + \beta \cdot \text{g-RAVE}_f\right) + \sqrt{\frac{c_e \ln(T_F)}{t_{F,f}} \min\left(\frac{1}{4}, \hat{\sigma}^2_{F,f} + \sqrt{\frac{2\ln(T_F)}{t_{F,f}}}\right)} \quad (9)$$

where $\beta = \frac{c_l}{c_l + t_l}$ and t_l denotes the number of iterations involved in ℓ-RAVE computation.

It must be noted that the FUSE tree and the RAVE score are tightly coupled. While the RAVE score guides the FUSE exploration, FUSE can inversely be viewed as a sophisticated way of building the RAVE score, providing an educated guess about the feature relevance.

6. UCT-based Feature Selection: FUSE

This section finally gives an overview of the FUSE algorithm, first detailing the reward function and the random phase policy which are specific to the Feature Selection task.

6.1. Instant Reward Function

A primary requirement for Monte-Carlo tree search is to set up a computationally cheap reward to be computed in each iteration. The FUSE reward function, aimed at estimating the generalization error, proceeds by learning from few examples, for the sake of tractability, while accounting for non-linear concepts for the sake of efficiency. Specifically, the reward attached to a feature subset F proceeds as follows. Let d_F denote the Euclidean distance based on features in F. Let $\mathcal{L}$ denote the training set and $\mathcal{V}$ an aggressive subsample of $\mathcal{L}$. For each labeled example $z = (x, y)$ in $\mathcal{V}$, let $\mathcal{N}_{F,k}(x)$ denote the set of the k nearest neighbors of x in $\mathcal{L}$ after d_F; define $s_F(z)$ the number of positive examples among these neighbors:

$$s_F(z) = |\{z' \in \mathcal{N}_{F,k}(x),\ y' > 0\}| \quad (10)$$

A reasonable tradeoff between stable and computationally frugal reward is given by the Area Under the ROC curve (AUC, aka. Mann Whitney Wilcoxon criterion), defined[1] as:

$$V(F) = \frac{|\{(z,z') \in \mathcal{V}^2,\ s_F(x) < s_F(x'),\ y < y'\}|}{|\{(z,z') \in \mathcal{V}^2,\ y < y'\}|} \quad (11)$$

Upon computing $V(F)$, the score of all nodes in the current path is updated. Although FUSE actually explores a graph (the lattice of the feature set), it does not update the reward of all nodes containing F due to the high branching factor of the graph.

Computational complexity: $V(F)$ is computed with linear complexity in the size n of $\mathcal{L}$ up to a

[1] Algorithmically, for each positive example $z \in \mathcal{V}$ let $v(z)$ (respectively $w(z)$) denote the number of negative examples $z' \in \mathcal{V}$ s.t. $s_F(z') < s_F(z)$ (resp. $s_F(z') = s_F(z)$), and let $S(F)$ be the sum of $z \in \mathcal{V}$ of $v(z) + \frac{1}{2}w(z)$ over all positive examples. Score $V(F)$ is set to $S(F)$ divided by the product of the number of positive and negative examples in $\mathcal{V}$.

Algorithm 1 FUSE

FUSE
Input: number of iterations T and many-armed behavior MA
Output: search tree $\mathcal{T}$ and g-RAVE score
Initialize $\mathcal{T} \leftarrow \emptyset$, $\forall f$, g-RAVE(f) $= 0$
for $t = 1$ **to** T **do**
  Iterate($\mathcal{T}$, g-RAVE, $\emptyset$)
end for

Iterate
Input: search tree $\mathcal{T}$, score g-RAVE, subset F
Output: reward V
if F final **then**
  $V \leftarrow V(F \setminus \{f_s\})$; Update g-RAVE
else
  if $t(F) \neq 0$ **then**
    if MA = progressive widening **then**
      $f^\star \leftarrow \underset{f \in AllowedFeatures(F)}{\text{argmax}}$ UCB1-tuned(F, f)
    else
      $f^\star \leftarrow \underset{f \in \mathcal{F} \setminus F}{\text{argmax}}$ tradeoff UCB/RAVE(F, f)
    end if
    $V \leftarrow iterate(\mathcal{T}$, g-RAVE, $F \cup \{f^\star\})$
  else
    $V \leftarrow iterate\_random(\mathcal{T}$, g-RAVE, $F)$
  end if
  Update T_F, t_f, $\hat{\mu}_{F,f}$, $\hat{\sigma}^2_{F,f}$ and ℓ-RAVE$_{F,\cdot}$
end if

Iterate_random
Input: search tree $\mathcal{T}$, score g-RAVE, subset F
Output: reward V
while $rand() < q^{|F|}$ **do**
  $f^\star \leftarrow$ uniformly selected feature in $\mathcal{F} \setminus (F \cup \{f_s\})$
  $F \leftarrow F \cup \{f^\star\}$
end while
$V \leftarrow V(F)$; Update g-RAVE

logarithmic term. More precisely the complexity is $\tilde{O}(mnd)$ where m denotes the size of $\mathcal{V}$ and d the size of the feature subset.

6.2. Random Phase

A key specificity of the Feature Selection task, viewed as Reinforcement Learning problem, is that it deals with a finite unknown horizon (the target number of features). This specificity is accounted for in the random UCT phase, by enforcing the selection of the stopping feature at level d with probability $1 - q^d$, where $q < 1$ is a parameter of the algorithm.

6.3. Overview of FUSE

The FUSE algorithm (algorithm 1) constructs both a search tree and a RAVE score, yielding two different FS strategies. The first one, noted FUSE, retains the features in the most often visited path in the search tree (to be preferred to the path with maximal average reward after the common UCT practice). The second one denoted FUSE^R uses the RAVE score to rank the features, and proceeds as a standard filtering FS algorithm.

In the following, the prefix C or D will be used to indicate whether FUSE and FUSE^R are used with the continuous or discrete heuristics (section 5.1).

7. Experimental Validation

This section aims at providing a proof of principle of the approach on standard benchmark data sets from the NIPS 2003 FS Challenge (Guyon et al., 2004). Due to space limitations, only binary classification problems and numerical features are considered.

7.1. Experimental Setting

The goal of the experiments is to answer three main questions. The first one is related to the overall **FS performance** of the proposed approach depending on the complexity of the underlying target concept and feature correlations. The second one regards the **comparative strengths and weaknesses** of FUSE and FUSE^R, and the merits of the discrete and continuous heuristics used to control exploration. The third one regards the **convergence speed** of the approach depending on the computational effort.

Three benchmark data sets (Table 1) have been considered to answer the first question. The Madelon and Arcene data sets involve concepts with correlated features. Madelon is an artificial 500-feature dataset, where the target concept is set to the XOR of five relevant features. The other 495 features involve 15 features, built as linear combinations of the relevant ones; the remaining features are irrelevant. Arcene is the concatenation of several datasets relevant to related though different concepts (each dataset being related to a particular type of cancer); expectedly, the target concept in Arcene involves the disjunction of overlapping sub concepts. The third dataset, Colon, is a microarray dataset introduced by (Alon et al., 1999), which is considered to be "simple". After (Guyon et al., 2007; Shen et al., 2008), Madelon and Colon are centered and normalized (based on the training set). For the sake of tractability, FUSE and CFS only consider the top 2000 features of the Arcene

Table 1. Datasets characteristics.

Data set	Samples	Features	Properties
Madelon	2600	500	XOR-like
Arcene	200	10,000	Disjunctive
Colon	62	2,000	"Easy"

dataset, ranked after their ANOVA score.

All reported results are averaged on ten independent runs. Each run considers a 5-fold stratified cross-validation (CV). For each fold, FUSE is launched with 200,000 Monte-Carlo iterations. The runtime is respectively 45 minutes, 5 minutes and 4 minutes on Madelon, Arcene and Colon on an Intel Core 2×2.6GHz CPU with 2GB memory (only considering FS on the training set, i.e. 80% of the dataset).

FUSE parameters include k (used in the reward function, section 6.1) set to 5; q (used to control the average depth in the random phase, section 6.2), set to $1-10^i$ for $i=-1,-3,-5$. The exploration control heuristics involves parameter c_e set to 10^i for $i=-4,-2,0,2$; the discrete heuristics (section 5.1) involves parameter b set to 2 in all experiments; the continuous heuristics involves c and c_l (section 5.3) both set to 10^i for $i=-\infty,2,4$.

FUSE performance is assessed comparatively to three baseline approaches: Correlation-based Feature Selection (CFS) (Hall, 2000), the Random-Forest based Gini score (Gini-RF) (Breiman et al., 1984) and Lasso (Tibshirani, 1994)[2]. FUSE^R was further compared to its naive variant RAND^R, using the average feature score built from uniformly selected 20-feature subsets.

For the sake of a fair comparison, all FS algorithms were combined with the same end learner, a Gaussian SVM (Collobert et al., 2002); its hyper-parameters are optimized using 5 fold CV on the training set unless otherwise specified.

7.2. Experimental Results

Some answers to the question of the performance of FUSE algorithms are given on Fig. 2, reporting their generalization error comparatively to that of Gini-RF, CFS and Lasso. The differences among the datasets are visible as Lasso is outperformed by other approaches on Arcene and Madelon (which are not linear) whereas all algorithms similarly perform on the Colon dataset. On Arcene and Madelon, the comparison between FUSE, CFS and Gini-RF suggests that the FUSE algorithms take the best of both worlds. On the one hand, they detect features which are only relevant when combined with other features, like Gini-RF, and they improve on Gini-RF when few features are considered. On the other hand, they can filter out redundant features, like CFS, and they statistically significantly improve on CFS when considering many features. This good behavior is explained from the reward used in FUSE, which allows for a global assessment of feature subsets without making assumptions about the complexity of the underlying target concept. In the meanwhile, the assessment of many feature subsets is made possible as only a tiny subsample of the instance space is considered in each iteration.

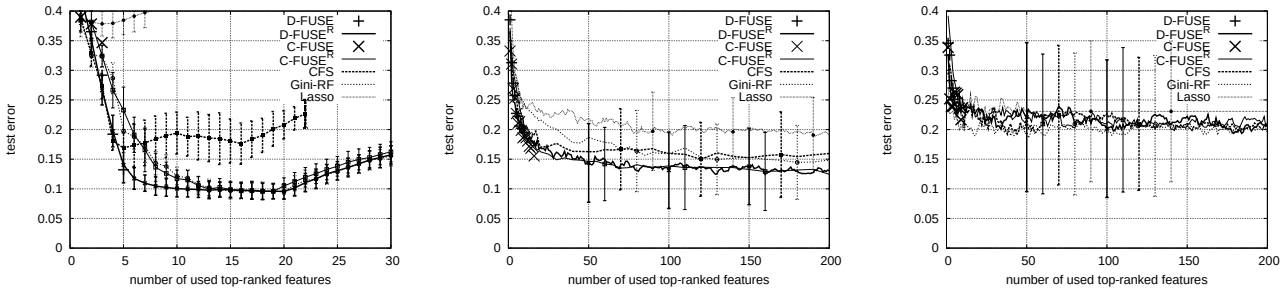
Fig. 2 also illustrates the difference between FUSE and FUSE^R, and the comparative merits of the discrete (progressive-widening-based) and continuous (Mogo-like) heuristics used to control the exploration width. A first remark is that FUSE does not control the depth of the search tree in an efficient way; the best FUSE paths involve from 5 to 15 features. The filtering approach based on the RAVE score built by FUSE, FUSE^R, thus appears to be significantly more efficient than FUSE (although both algorithms almost always select the same top features, and thus coincide at the beginning of the curve reporting the performance vs. the number of features).

The discrete and continuous heuristics yield same results at their best, i.e. when the c and c_l parameters of the continuous heuristics are well chosen. For this reason, the discrete heuristics is preferred as more robust: the single b parameter used to control the branching factor depending on the number of visits of the node was set to 2 in all experiments.

These results are comparable to the state of the art along the NIPS FS challenge setting (using an additional separate test set): the best performance on Madelon (reached by (Shen et al., 2008)) is 6.22% vs 6.5 for FUSE^R.

Regarding the scalability question, some answers are graphically provided by Fig. 3, reporting the average test error versus the number of iterations in log scale on the Madelon dataset; the performances reported for D-FUSE^R, C-FUSE^R and RAND^R correspond to the best 20 features for the sake of comparison with D-FUSE and C-FUSE. FUSE reaches its optimum after 10,000 iterations and thereafter stagnates, basically because the exploration does not allow for an in-depth exploration of the search tree and the size of the best path remains limited. Interestingly, FUSE^R converges more slowly than FUSE at the beginning; but it soon

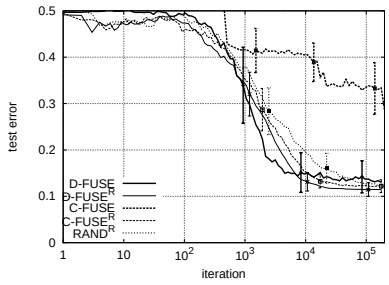
[2]Using the Weka implementation of CFS; the R implementation of Gini-RF (with 1,000 trees); the R (glmnet) implementation of Lasso.



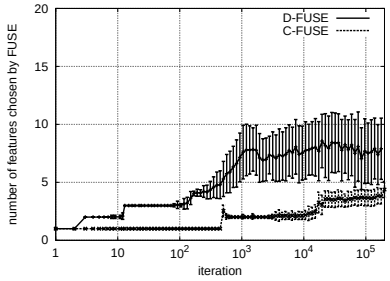
(a) Madelon (b) Arcene (c) Colon

Figure 2. Performance versus the number of selected features. The performance measures the average error on 50 splits of Madelon (left), Arcene (center) and Colon (right). D-FUSE and C-FUSE are only visible at the beginning of the curve as they select from 5 to 15 features.

catches up and improves on FUSE after 10,000 iterations. In the meanwhile, FUSE^R is faster by an order of magnitude than RAND^R, i.e. reaches the same level of performances with about 10 times less iterations.



(a) Test error



(b) Size of FUSE feature subset

Figure 3. Average test error (a) and size of D-FUSE and C-FUSE feature subset (b) on Madelon vs. number N of iterations in logscale. D-FUSE^R and C-FUSE^R use the top 20 features after the RAVE score built from N iterations; RAND^R use the top-20 features after the average score out of N uniformly selected 20-feature subsets.

8. Discussion and Perspectives

The main merits of the presented work are i/ to formalize Feature Selection as a Reinforcement Learning problem aimed at minimizing the generalization error; and ii/ to use this formalization to revisit wrapper approaches, for these uniquely simultaneously account for feature interdependencies and filter out useless redundancies. Algorithmically, the FS combinatorial optimization problem is tackled by combining Monte-Carlo Tree Search and a fast approximation of the generalization error; several extensions of the UCT framework have been proposed to handle the specificities of the FS search space, resulting in the FUSE algorithms. While a proof-of-principle experiment demonstrates the comparative performances of the approach, it raises several questions for further investigation. A main question is whether the proposed approach should be viewed as a wrapper, or a filtering FS approach. On the one hand, the RAVE vector summarizing the trials is a way of supporting a better-informed FUSE exploration, and thus a wrapper FS approach; on the other hand, FUSE can be viewed as a sophisticated way of building the RAVE score, supporting a filtering FS approach.

Further research will extend the approach to multi-class problems and mixed (continuous and discrete) search spaces, combine FUSE with other end learners, and reconsider the instant reward criterion. A longer term perspective is to extend the presented approach to Feature Construction, by extending FUSE to explore a regular language as in (de Mesmay et al., 2009).

Acknowledgments

We thank Olivier Teytaud and Philippe Rolet for many fruitful discussions, and the anonymous reviewers for their suggestions and comments. We acknowledge the support of the PASCAL2 Network of Excellence, IST-2007-216886.

References

Alon, U., Barkai, N., Notterman, D. A., Gishdagger, K., Ybarradagger, S., Mackdagger, D., and Levine, A. J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. In *Proc. of the Nat. Ac. of Sc. of the USA*, pp. 6745–6750, June 1999.

Auer, P., Cesa-Bianchi, N., and Fischer, P. Finite-time analysis of the Multiarmed Bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002.

Bach, F. Exploring large feature spaces with Hierarchical Multiple Kernel Learning. In *NIPS'08*, pp. 105–112, 2008.

Bellman, R. *Dynamic Programming*. Princeton Univ. Press, 1957.

Boullé, M. Compression-based averaging of selective Naive Bayes classifiers. *J. Mach. Learn. Res.*, 8:1659–1685, 2007.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R.A. *Classification and Regression Trees*. Taylor & Francis, Inc., 1984.

Chan, A. B., Vasconcelos, N., and Lanckriet, G. R. G. Direct convex relaxations of sparse SVM. In *ICML'07*, pp. 145–153, 2007.

Collobert, R., Bengio, S., and Marithoz, J. Torch: A modular machine learning software library. Technical report, IDIAP, 2002.

Coulom, R. Efficient selectivity and backup operators in Monte-Carlo tree search. In *Computers and Games*, pp. 72–83, 2006.

de Mesmay, F., Rimmel, A., Voronenko, Y., and Püschel, M. Bandit-based optimization on graphs with application to library performance tuning. In *ICML'09*, pp. 729–736, 2009.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

Fung, G. and Mangasarian, O. L. Data Selection for Support Vector Machine Classifiers. In *KDD'00*, pp. 64–70, 2000.

Gelly, S. and Silver, D. Combining online and offline knowledge in UCT. In *ICML'07*, pp. 273–280, 2007.

Guyon, I., Weston, J., Barnhill, S., and Vapnik, V. Gene selection for cancer classification using Support Vector Machines. *Mach. Learn.*, 46(1-3):389–422, 2002.

Guyon, I., Gunn, S. R., Ben-Hur, A., and Dror, G. Result analysis of the NIPS 2003 Feature Selection challenge. In *NIPS'04*, pp. 545–552, 2004.

Guyon, I., Li, J., Mader, T., Pletscher, P. A., Schneider, G., and Uhr, M. Competitive baseline methods set new standards for the NIPS 2003 feature selection benchmark. *Pattern Recogn. Lett.*, 28(12):1438–1444, 2007.

Hall, M. A. Correlation-based Feature Selection for discrete and numeric class Machine Learning. In *ICML'00*, pp. 359–366, 2000.

Helleputte, T. and Dupont, P. Partially supervised Feature Selection with regularized linear models. In *ICML'09*, pp. 409–416, 2009.

Kira, K. and Rendell, L. A. A practical approach to feature selection. In *ML'92*, pp. 249–256, 1992.

Kocsis, L. and Szepesvári, C. Bandit based Monte-Carlo planning. In *ECML'06*, pp. 282–293, 2006.

Margaritis, D. Toward provably correct Feature Selection in arbitrary domains. In *NIPS'09*, pp. 1240–1248, 2009.

Rogers, J. and Gunn, S. R. Identifying feature relevance using a Random Forest. In *SLSFS*, pp. 173–184, 2005.

Rolet, P., Sebag, M., and Teytaud, O. Boosting Active Learning to optimality: a tractable Monte-Carlo, Billiard-based algorithm. In *ECML'09*, pp. 302–317, 2009.

Shen, K. Q., Ong, C. J., Li, X. P., and Wilder-Smith, E. P. V. Feature selection via sensitivity analysis of SVM probabilistic outputs. *Mach. Learn.*, 70(1):1–20, 2008.

Tibshirani, R. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

Varma, M. and Babu, B. R. More generality in efficient Multiple Kernel Learning. In *ICML'09*, pp. 1065–1072, 2009.

Wang, Y., Audibert, J.Y., and Munos, R. Algorithms for infinitely Many-Armed Bandits. In *NIPS08*, pp. 1729–1736, 2008.

Zhang, T. Adaptive Forward-Backward Greedy Algorithm for Sparse Learning with Linear Models. In *NIPS'08*, pp. 1921–1928, 2008a.

Zhang, T. Multi-stage Convex Relaxation for Learning with Sparse Regularization. In *NIPS'08*, pp. 1929–1936, 2008b.