Mach Learn (2012) 89:87–122
DOI 10.1007/s10994-012-5306-7

Sequential approaches for learning datum-wise sparse representations

Gabriel Dulac-Arnold · Ludovic Denoyer · Philippe Preux · Patrick Gallinari

Received: 4 November 2011 / Accepted: 13 June 2012 / Published online: 10 August 2012

Abstract In supervised classification, data representation is usually considered at the dataset level: one looks for the "best" representation of data assuming it to be the same for all the data in the data space. We propose a different approach where the representations used for classification are tailored to each datum in the data space. One immediate goal is to obtain *sparse datum-wise representations*: our approach learns to build a representation specific to each datum that contains only a small subset of the features, thus allowing classification to be fast and efficient. This representation is obtained by way of a sequential decision process that sequentially chooses which features to acquire before classifying a particular point; this process is learned through algorithms based on Reinforcement Learning.

The proposed method performs well on an ensemble of medium-sized sparse classification problems. It offers an alternative to global sparsity approaches, and is a natural framework for sequential classification problems. The method extends easily to a whole family of sparsity-related problem which would otherwise require developing specific solutions. This is the case in particular for cost-sensitive and limited-budget classification, where feature acquisition is costly and is often performed sequentially. Finally, our approach can handle non-differentiable loss functions or combinatorial optimization encountered in more complex feature selection problems.

Editors: Dimitrios Gunopulos, Donato Malerba, and Michalis Vazirgiannis.

G. Dulac-Arnold (✉) · L. Denoyer · P. Gallinari
UPMC, LIP6, Université Pierre et Marie Curie, Case 169, 4 Place Jussieu, 75005 Paris, France
e-mail: gabriel.dulac-arnold@lip6.fr

L. Denoyer
e-mail: ludovic.denoyer@lip6.fr

P. Gallinari
e-mail: patrick.gallinari@lip6.fr

P. Preux
LIFL (UMR CNRS) & INRIA Lille Nord-Europe, Université de Lille, Villeneuve d'Ascq, France
e-mail: philippe.preux@inria.fr

Keywords Classification · Features selection · Sparsity · Sequential models · Reinforcement learning

1 Introduction

Feature Selection is one of the main contemporary issues in Machine Learning and has been approached from many directions. One modern approach to feature selection in linear models consists in minimizing an L_0 regularized empirical risk. This particular risk encourages the model to have a good balance between a low classification error and a high sparsity (where only a few features are used for classification). As the L_0 regularized problem is combinatorial, many approaches such as the LASSO (Tibshirani 1994) try to address the combinatorial problem by using more practical norms such as L_1. These classical approaches to sparsity aim at finding a sparse representation of the feature space that is global to the entire dataset.

We propose a new approach to sparsity where the goal is to limit the number of features **per datapoint** classified, thus *datum-wise sparse classification model* (DWSM). Our approach allows the choice of features used for classification to vary relative to each datapoint; data points that are easy to classify can be inferred on with only a few features, and others which might require more information can be classified using more features. The underlying motivation is that, while classical approaches balance between accuracy and sparsity at the dataset level, our approach optimizes this balance at the individual datum level.

This approach differs from the usual feature selection paradigm which operates at a global level on the dataset. We believe that several problems could benefit from our approach. For many applications, a decision could be taken by observing only a few features for most items, whereas other items would require closer examination. In these situations, datum-wise approaches can achieve higher overall sparsity than classical methods. In our opinion, however, this is not the only important aspect of this model, as there are two primary domains where this alternative approach to sparsity can also be beneficial. First, this is a natural framework for sequential classification tasks, where a decision regarding an item class is taken as soon as enough information has been gathered on this item (Louradour and Kermorvant 2011; Dulac-Arnold et al. 2011). Second, the proposed framework naturally adapts to a variety of sparsity or feature selection problems that would require new specific solutions with classical approaches. It also allows for the handling of situations where the loss function is not continuous—situations that are difficult to cope with through classical optimization. It can be easily adapted, for example, to limited budget or cost sensitive problems where the acquisition of features is costly, as it is often the case in domains such as diagnosis, medicine, biology or even for information extraction problems (Kanani and McCallum 2012). DWSM also allows handling easily complex problems where features admit a certain inherent structure.

In order to solve the combinatorial feature selection problem, we propose to model feature selection and classification as a single sequential Markov Decision Process (MDP). A scoring function associated to the MDP policy will return at any time the best possible action. Each action consists either in choosing a new feature for a given datum or in deciding on the class of x if enough information is available for deciding so. During inference, this score function will allow us to greedily choose which features to use for classifying each particular datum. Learning the policy is performed using an algorithm inspired by Reinforcement Learning (Sutton and Barto 1998). In this sequential decision process, datum-wise sparsity is obtained by introducing a penalizing reward when the agent chooses to incorporate an additional feature into the decision process. We show that an optimal policy in this MDP corresponds to an optimal classifier w.r.t. the datum wise loss function which incorporates a sparsity inducing term.

The contributions of the paper are as follows:

1. We formally introduce the concept of datum-wise sparse classification, and propose a new classifier model whose goal is two-fold: maximize the classification accuracy, while using as few features as possible to represent each datum. The model considers classification as a sequential process, where the model's choice of features is dependent on the current datum's values.
2. We formalize this sequential model using a Markov Decision Process. This allows us to use an MDP, we show the equivalence between learning to maximize a reward function and minimizing a datum-wise loss function. This new approach results in a model that obtains good performance in terms of classification while maximizing datum-wise sparsity, i.e. the mean number of features used for classifying the whole dataset. Our model also naturally handles multi-class classification problems, solving them by using as few features as possible for all classes combined.
3. We propose a series of extensions to our base model that allow us to deal with variants of the feature selection problem, such as: hard budget classification, group features, cost-sensitive classification, and relational features. These extensions aim at showing that the proposed sequential model is more than a classical sparse classifier, but can be easily adapted to many different classification tasks where the features selection/acquisition process is complex, all while maintaining good classification accuracy.
4. We perform a series of experiments on many different corpora: 13 for the base model and additional corpora for the extensions. Then we compare the model with those obtained by optimizing the LASSO problem (Efron et al. 2004), an L_1-regularized SVM, and CART decision trees. This provides a qualitative study of the behavior of our algorithm. Additionally, we perform a series of experiments to demonstrate the potential of the various extensions proposed to our model.

The paper is organized as follows: First, we define the notion of **datum-wise sparse classifiers** and explain the interest of such models in Sect. 2. We then describe our sequential approach to classification and detail the learning algorithm in Sect. 3. We then proceed to explain how the model described in Sect. 3 can be extended to handle more complex problems such as cost-sensitive classification in Sect. 4. e discuss the algorithm complexity in Sect. 5 and its scalability to large datasets. We detail experiments and also give a qualitative analysis of the behavior of this base model and the extensions in Sect. 6. Finally, a state-of-the-art in sparsity, as well as an overview of related work is presented in Sect. 7.

2 Datum-wise sparse classifiers

We consider the problem of supervised multi-class classification[1] where one wants to learn a classification function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$. The function f_θ associates one category $y \in \mathcal{Y}$ to a vector $\mathbf{x} \in \mathcal{X}$, with $\mathcal{X} = \mathbb{R}^n$, n being the dimension of the input vectors. θ is the set of parameters learned from a training set composed of input/output pairs $\mathcal{T}_{train} = \{(\mathbf{x_i}, y_i)\}_{i\in[1..N]}$.

[1] Note that this includes the binary supervised classification problem as a special case.

These parameters are commonly found by minimizing the empirical risk defined by:

$$\theta^* = \operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Delta\big(f_\theta(\mathbf{x_i}), y_i\big), \tag{1}$$

where Δ is the loss associated to a prediction error, defined as:

$$\Delta(x, y) = \begin{cases} 0 & \text{if } x = y, \\ 1 & \text{if } x \neq y. \end{cases}$$

This empirical risk minimization problem does not consider any *prior assumption* or constraint concerning the form of the solution and can result in overfitting models. Moreover, when facing a very large number of features, obtained solutions usually need to perform computations on all the features for classifying each datum, thus negatively impacting the model's classification speed. We propose a different risk minimization problem where we add a penalization term that encourages the obtained classifier to classify using **on average** as few features as possible. In comparison to classical L_0 or L_1 regularized approaches where the goal is to constrain the number of features used at the dataset level, our approach performs sparsity at the datum level, allowing the classifier to use different features when classifying different inputs. This results in a **datum-wise sparse classifier** that, when possible, only uses a few features for classifying easy inputs, and more features for classifying difficult or ambiguous ones.

We consider a classifier function that, in addition to predicting a label y given an input $\mathbf{x}$, also provides information about which features have been used for classification. Let us denote $\mathcal{Z} = \{0; 1\}^n$. We define a **datum-wise classification function** f of parameters θ as:

$$f_\theta : \begin{cases} \mathcal{X} \rightarrow \mathcal{Y} \times \mathcal{Z}, \\ f_\theta(\mathbf{x}) = (y, \mathbf{z}), \end{cases}$$

where y is the predicted output and $\mathbf{z}$ is a n-dimensional vector $\mathbf{z} = (z^1, \dots, z^n)$, such that $z^i = 1$ implies that feature i has been taken into consideration for computing label y on datum $\mathbf{x}$. By convention, we denote the predicted label as $y_\theta(\mathbf{x})$ and the corresponding $\mathbf{z}$-vector as $z_\theta(\mathbf{x})$. Thus, if $z_\theta^i(\mathbf{x}) = 1$, feature i has been used for classifying $\mathbf{x}$ into category $y_\theta(\mathbf{x})$.

This definition of data-wise classifiers has two main advantages: First, as we will see in the next section, because f_θ can explain its use of features with $z_\theta(\mathbf{x})$, we can add constraints on the features used for classification. This allows us to encourage datum-wise sparsity which we define below. Second, experimental analysis of $z_\theta(\mathbf{x})$ gives a qualitative explanation of how the classification decision has been made, which we discuss in Sect. 6. Note that the way we define datum-wise classification is an extension to the usual definition of a classifier.

2.1 Datum-wise sparsity

Datum-wise sparsity is obtained by adding a penalization term to the empirical loss defined in Eq. (1) that limits the average number of features used for classifying:

$$\theta^* = \operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Delta\big(y_\theta(\mathbf{x_i}), y_i\big) + \lambda \frac{1}{N} \sum_{i=1}^{N} \big\| z_\theta(\mathbf{x_i}) \big\|_0. \tag{2}$$

The term $\|z_\theta(\mathbf{x_i})\|_0$ is the L_0 norm[2] of $z_\theta(\mathbf{x_i})$, i.e. the number of features selected for classifying $\mathbf{x_i}$. In the general case, the minimization of this new risk results in a classifier that *on average* selects only a few features for classifying, but may use a different set of features w.r.t. to the input being classified. *We consider this to be the crux of the DWSM model*: the classifier takes each datum into consideration differently during the inference process.

Note that the optimization of the loss defined in Eq. (2) is a combinatorial problem that becomes quickly intractable. In the next section, we propose an original way to deal with this problem, based on a Markov Decision Process.

3 Datum-wise sparse sequential classification

In the next couple of pages, we begin by proposing a model that allows us to solve the problem posed in Eq. (2). Then we explain how we can use this model to classify in a sequential manner that allows for datum-wise sparsity.

3.1 Markov decision process

We consider a Markov Decision Process (MDP, Puterman 1994)[3] to classify an input $\mathbf{x} \in \mathbb{R}^n$. At first, we have no information about $\mathbf{x}$, that is, we have no attribute/feature values. Then, step-by-step, we can choose to either acquire a particular feature of $\mathbf{x}$ or to classify $\mathbf{x}$. The act of classifying $\mathbf{x}$ in the category y ends an "episode" of the sequential process. This process is illustrated with an example MDP in Fig. 1. The classification process is a deterministic process defined by:

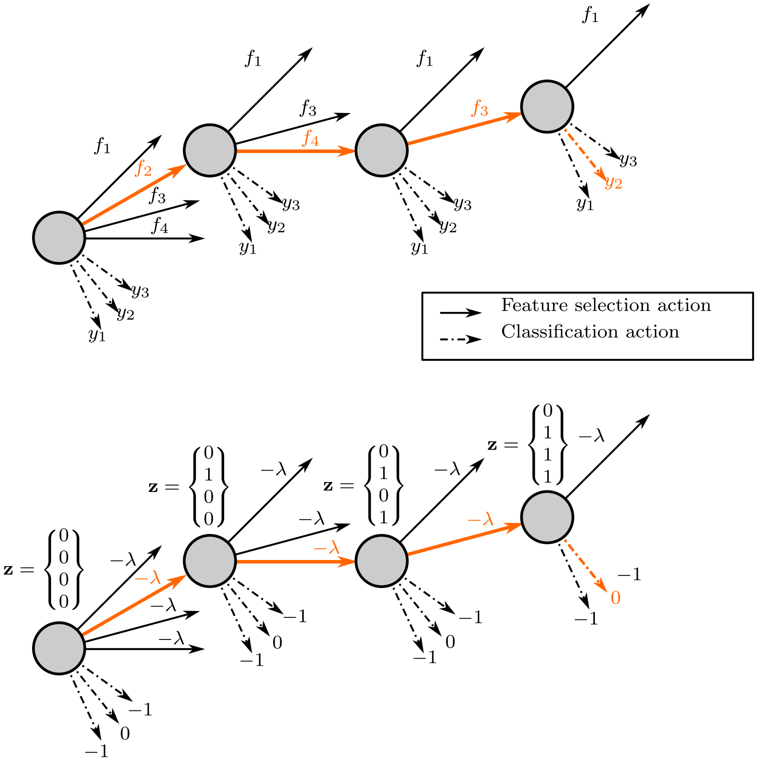
- A set of states $\mathcal{X} \times \mathcal{Z}$, where the tuple $(\mathbf{x}, \mathbf{z})$ corresponds to the state where the agent is considering datum $\mathbf{x}$ and has selected features specified by $\mathbf{z}$. The number of currently selected features is thus $\|\mathbf{z}\|_0$.
- A set of actions $\mathcal{A}$ where $\mathcal{A}(\mathbf{x}, \mathbf{z})$ denotes the set of possible actions in state $(\mathbf{x}, \mathbf{z})$. We consider two types of actions:
  - $\mathcal{A}_f$ is the set of feature selection actions such that, for $a \in \mathcal{A}_f$, choosing action $a = a_j$ corresponds to choosing feature f_j. Note that the set of possible feature selection actions on state $(\mathbf{x}, \mathbf{z})$, denoted $\mathcal{A}_f(\mathbf{x}, \mathbf{z})$, is equal to the subset of currently unselected features, i.e. $\mathcal{A}_f(\mathbf{x}, \mathbf{z}) = \{a_j \in \mathcal{A}_f, \text{ s.t. } \mathbf{z}_j = 0\}$.
  - $\mathcal{A}_y$ is the set of classification actions—one action for each possible class—that correspond to assigning a label to the current datum. Classification actions stop the sequential decision process.
- A transition function defined only for feature selection actions (since classification actions are terminal):

$$\mathcal{T} : \begin{cases} \mathcal{X} \times \mathcal{Z} \times \mathcal{A}_f \rightarrow \mathcal{X} \times \mathcal{Z}, \\ \mathcal{T}((\mathbf{x}, \mathbf{z}), a) = (\mathbf{x}, \mathbf{z}') \end{cases}$$

  where $\mathbf{z}'$ is an updated version of $\mathbf{z}$ such that for $a = a_j$, $\mathbf{z}' = \mathbf{z} + \mathbf{e_j}$ with $\mathbf{e_j}$ as the vector whose value is 1 for component j and 0 for the other components.

[2] The L_0 'norm' is not a proper norm, but we will refer to it as the L_0 norm in this paper, as is common in the sparsity community.

[3] The MDP is deterministic in our case.



f_1
f_2
f_3
f_4
y_1
y_2
y_3
Feature selection action
Classification action
$\mathbf{z} = \begin{Bmatrix}0\\0\\0\\0\end{Bmatrix}$
$\mathbf{z} = \begin{Bmatrix}0\\1\\0\\0\end{Bmatrix}$
$\mathbf{z} = \begin{Bmatrix}0\\1\\0\\1\end{Bmatrix}$
$\mathbf{z} = \begin{Bmatrix}0\\1\\1\\1\end{Bmatrix}$
$-\lambda$
-1
0

Fig. 1 The sequential process for a problem with 4 features $(f_1, \dots, f_4)$ and 3 possible categories $(y_1, \dots, y_3)$. *Top*: The leftmost circle is the initial state for one particular input $\mathbf{x}$. Classification actions are terminal, and therefore end the decision process. In this example, the classification has been made by sequentially choosing to acquire feature 2, feature 4, feature 3 and then classifying $\mathbf{x}$ in category y_2. The bold (orange) arrows correspond to the trajectory made by the current policy. *Bottom*: The values of $z(\mathbf{x})$ for the different states are illustrated. The value on the arrows corresponds to the immediate reward received by the agent assuming that $\mathbf{x}$ belongs to category y_2. At the end of the process, the agent has received a total reward of -3λ

3.1.1 Policy

We define a parameterized policy π_θ, which, for each state $(\mathbf{x}, \mathbf{z})$, returns the best action as defined by a scoring function $s_\theta(\mathbf{x}, \mathbf{z}, a)$:

$$\pi_\theta : \mathcal{X} \times \mathcal{Z} \rightarrow \mathcal{A} \quad \text{and} \quad \pi_\theta(\mathbf{x}, \mathbf{z}) = \operatorname*{argmax}_a s_\theta(\mathbf{x}, \mathbf{z}, a).$$

The policy π_θ decides which action to take by applying the scoring function to every action possible from state $(\mathbf{x}, \mathbf{z})$ and greedily taking the highest scoring action. The scoring function reflects the *overall* quality of taking action a in state $(\mathbf{x}, \mathbf{z})$, which corresponds to the total reward obtained by taking action a in $(\mathbf{x}, \mathbf{z})$ and thereafter following policy π_θ:[4]

$$s_\theta(\mathbf{x}, \mathbf{z}, a) = r(\mathbf{x}, \mathbf{z}, a) + \sum_{t=1}^{T} r_\theta^t \big| \big((\mathbf{x}, \mathbf{z}), a\big). \tag{3}$$

Here, $\sum_{t=1}^{T} r_\theta^t|((\mathbf{x}, \mathbf{z}), a)$ corresponds to the cumulative reward when starting in state $(\mathbf{x}, \mathbf{z})$, choosing action a, and following policy π_θ until classification.

Here $r_\theta^t \mid ((\mathbf{x}, \mathbf{z}), a)$ corresponds to the reward obtained at step t. Taking the sum of these rewards gives us the total reward from state $(\mathbf{x}, \mathbf{z})$ until the end of the episode. Since the policy is deterministic, we may refer to a parameterized policy using only θ. Note that the optimal parameterization θ^* obtained after learning (see Sect. 3.4) is the parameterization that maximizes the expected reward over *all* possible state-action pairs.

In practice, the initial state of such a process for an input $\mathbf{x}$ corresponds to an empty $\mathbf{z}$ vector where no feature has been selected. The policy θ sequentially picks, one by one, a set of features pertinent to the classification task, and then chooses to classify once enough features have been considered.

3.1.2 Reward

The reward function reflects the *immediate* quality of taking action a in state $(\mathbf{x}, \mathbf{z})$ relative to the problem at hand. We define a reward function $\mathcal{R} : \mathcal{X} \times \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ for $(\mathbf{x_i}, y_i) \in \mathcal{T}_{train}$:

- If a corresponds to a feature selection action i.e. $a \in \mathcal{A}_f$:

$$r(\mathbf{x_i}, \mathbf{z}, a) = -\lambda.$$

- If a corresponds to a classification action i.e. $a \in \mathcal{A}_y$:

$$r(\mathbf{x_i}, \mathbf{z}, a) = \begin{cases} 0 & \text{if } a = y_i, \\ -1 & \text{if } a \neq y_i. \end{cases}$$

With the reward function defined in this way, correctly classifying an input $\mathbf{x_i}$ by acquiring f features results in obtaining a reward of $-f \cdot \lambda$ while an incorrect classification results in a reward of $-f \cdot \lambda - 1$. A good rule of thumb is to keep $\lambda < 1/n$ in order to avoid situations where classifying incorrectly is a better decision than choosing sufficient features. Of course depending on the particular application and desired sparsity, on can choose larger values for λ.

3.2 Reward maximization and loss minimization

As explained in Sect. 2, our ultimate goal is to find the parameterization θ^* that minimizes the datum-wise empirical loss defined in Eq. (2). Let us therefore show that maximizing the expected reward is equivalent to minimizing the datum-wise empirical loss.

[4] This corresponds to the classical Q-function in Reinforcement Learning. We have however removed the expectation since our MDP is deterministic.

$$\begin{aligned}
\theta^* &= \operatorname*{argmax}_\theta \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_\theta(\mathbf{x_i})+1} r\big(\mathbf{x_i}, z_\theta^{(t)}(\mathbf{x_i}), \pi_\theta\big(\mathbf{x_i}, z_\theta^{(t)}\big)\big)\\
&= \operatorname*{argmax}_\theta \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 0-\lambda\cdot\|z_\theta(\mathbf{x_i})\|_0 & \text{if } y=y_i\\ -1-\lambda\cdot\|z_\theta(\mathbf{x_i})\|_0 & \text{if } y\neq y_i\end{cases}\\
&= \operatorname*{argmax}_\theta \frac{1}{N}\sum_{i=1}^{N}\big(-\Delta\big(y_\theta(\mathbf{x_i}), y_i\big)-\lambda\big\|z_\theta(\mathbf{x_i})\big\|_0\big)\\
&= \operatorname*{argmin}_\theta \frac{1}{N}\sum_{i=1}^{N}\big(\Delta\big(y_\theta(\mathbf{x_i}), y_i\big)+\lambda\big\|z_\theta(\mathbf{x_i})\big\|_0\big)\\
&= \operatorname*{argmin}_\theta \frac{1}{N}\sum_{i=1}^{N}\Delta\big(y_\theta(\mathbf{x_i}), y_i\big)+\lambda\frac{1}{N}\sum_{i=1}^{N}\big\|z_\theta(\mathbf{x_i})\big\|_0. \qquad (4)
\end{aligned}$$

Here, $\pi_\theta(\mathbf{x_i}, \mathbf{z}_\theta^{(t)})$ is the action taken at time t by the policy π_θ for the training example $\mathbf{x_i}$ and $T_\theta(\mathbf{x_i})$ is the number of features acquired by π_θ before classifying $\mathbf{x_i}$.

Such an equivalence between risk minimization and reward maximization shows that the optimal classifier θ^* corresponds to the optimal policy in the MDP defined previously. This equivalence allows us to use classical MDP resolution algorithms (Sutton and Barto 1998) in order to find the best classifier. We detail the learning procedure in Sect. 3.4.

3.3 Inference and approximated decision processes

Due to the infinite number of possible inputs $\mathbf{x}$, the number of states is also infinite. Moreover, the reward function $r(\mathbf{x}, \mathbf{z}, a)$ is only known for the values of $\mathbf{x}$ that are in the training set and cannot be computed for any other datum. For these two reasons, it is not possible to compute the score function for all state-action pairs in a tabular manner, and this function has to be approximated.

The scoring function that underlies the policy $s_\theta(\mathbf{x}, \mathbf{z}, a)$ is approximated with a linear model:

$$s(\mathbf{x}, \mathbf{z}, a) = \big\langle \Phi(\mathbf{x}, \mathbf{z}, a); \theta\big\rangle.$$

The policy defined by such a function consists in taking in state $(\mathbf{x}, \mathbf{z})$ the action a' that maximizes the scoring function i.e. $a' = \operatorname{argmax}_{a\in\mathcal{A}}\langle\Phi(\mathbf{x}, \mathbf{z}, a); \theta\rangle$. The scoring function $s(\mathbf{x}, \mathbf{z}, a)$ is thus the function which chooses which action to take in any state of the decision process: it decides which features to acquire and when to classify a particular input. In our experiments we have chosen to restrict ourselves to a linear scoring function. The choice of a linear function is natural to be able to compare ourselves with the classification performance of L_1-regularized linear models. Our work could easily be extended to non-linear cases by changing the regression machine's nature. Note that, as explained in Sect. 5, using linear machines allows one to have a very low inference complexity.

The state-action pairs are represented in a feature space. We note $\Phi(\mathbf{x}, \mathbf{z}, a)$ the featurized representation of the $((\mathbf{x}, \mathbf{z}), a)$ state-action pair. Many definitions may be used for this feature representation, but we propose a simple projection. To begin with, let us restrict the representation of $\mathbf{x}$ to only the selected features. Let $\mu(\mathbf{x}, \mathbf{z})$ be the restriction of $\mathbf{x}$ according to $\mathbf{z}$:

$$\mu(\mathbf{x},\mathbf{z})^i = \begin{cases} x^i & \text{if } z^i = 1, \\ 0 & \text{elsewhere.} \end{cases}$$

To be able to differentiate between an attribute of $\mathbf{x}$ that is not yet known, and an attribute of x that is simply equal to 0, we must keep this information present in $\mathbf{z}$. Let $\phi(\mathbf{x},\mathbf{z}) = (\mathbf{z}, \mu(\mathbf{x},\mathbf{z}))$ be the intermediate representation that corresponds to the *concatenation* of $\mathbf{z}$ with $\mu(\mathbf{x},\mathbf{z})$. Now we simply need to keep the information present in a such that each action can be easily distinguished by a linear classifier. To do this, we use block-vectors (Har-Peled et al. 2002). This consists in projecting $\phi(\mathbf{x},\mathbf{z})$ into a higher dimensional space, such that the position of $\phi(\mathbf{x},\mathbf{z})$ inside the global vector $\Phi(\mathbf{x},\mathbf{z},a)$ is dependent on action a:

$$\Phi(\mathbf{x},\mathbf{z},a) = \big(\mathbf{0},\dots,\mathbf{0},\phi(\mathbf{x},\mathbf{z}),\mathbf{0},\dots,\mathbf{0}\big).$$

In $\Phi(\mathbf{x},\mathbf{z},a)$, the block $\phi(\mathbf{x},\mathbf{z})$ is at position $i_a \cdot |\phi(\mathbf{x},\mathbf{z})|$ where i_a is the index of action a in the set of all the possible actions. Thus, $\phi(\mathbf{x},\mathbf{z})$ is offset by an amount dependent on the action a. This equates to having a different linear classifier for each possible action in the MDP.

3.4 Learning

The goal of the learning phase is to find an optimal policy parameterization θ^* which maximizes the expected reward, thus minimizing the datum-wise regularized loss defined in Eq. (2). The combinatorial space consisting of all possible feature subsets for each individual datum in the training set is extremely large. Therefore, we cannot exhaustively explore the state space during training, and thus use a Monte-Carlo approach to sample example states from the learning space.

In order to find the optimal policy parameterization θ^*—which maximizes the expected reward, thus minimizing the regularized empirical loss defined in Eq. (2)—we propose to use an Approximate Policy Iteration learning approach based on Rollouts Classification Policy Iteration (RCPI) (Lagoudakis and Parr 2003). Sampling state-action pairs according to a previous policy $\pi_{\theta^{(t-1)}}$, RCPI consists in iteratively learning a better policy $\pi_{\theta^{(t)}}$ by iteratively improving estimations of s_θ as defined in Eq. (3). The RCPI algorithm is composed of three main steps that are iteratively repeated:

1. The algorithm begins by sampling a set of random states: the $\mathbf{x}$ vector is sampled from a uniform distribution in the training set, and $\mathbf{z}$ is also sampled using a uniform binomial distribution.
2. For each sampled state, the policy $\pi_{\theta^{(t-1)}}$ is used to compute the expected reward of choosing each possible action from that state. We now have a feature vector $\Phi(\mathbf{x},\mathbf{z},a)$ for each state-action pair in the sampled set, and the corresponding expected reward denoted $R_{\theta^{(t-1)}}(\mathbf{x},\mathbf{z},a)$.
3. The parameters $\theta^{(t)}$ of the new policy are then computed using classical linear regression on the set of feature vectors—$\Phi(\mathbf{x},\mathbf{z},a)$—and corresponding expected rewards—$R_{\theta^{(t-1)}}(\mathbf{x},\mathbf{z},a)$—as regression targets. This classifier gives an estimated score to state-action pairs even if we have never seen them previously.

After a certain number of iterations, the parameterized policy converges to a final policy $\pi_{\hat{\theta}}$ which is used for inference. Convergence is not guaranteed by Approximate Policy Iteration algorithms, but in practice occurs after only a few iterations. Termination of the learning algorithm happens once performance of a new policy no longer significantly improves over the previous iteration's policy.

RCPI is based on two different hyper-parameters that have to be tuned manually: the number of states used for Monte-Carlo Simulation and the number of rollout trajectories sampled for each state-action pair. These parameters have a direct influence over the performances of the algorithm and the time spent for learning. As explained in Lazaric et al. (2010), a good choice consists in choosing a high value of sampled state with only a few sampling trajectories. This is the choice we have made for the experiments.

4 Model extensions

So far, we have introduced the concept of datum-wise sparsity, and showed how it can be modeled as a Sequential Decision Process. Let us now show how DWSM can be extended to tackle other types of feature selection problems. This section aims to show that the proposed DWSM model is very general and can easily be adapted for many new feature selection problems, while keeping its datum-wise properties. We show how we can address the following classification tasks: **hard budget feature selection**, **cost-sensitive feature acquisition**, **group feature sparsity**, and **relational feature sparsity**. All of these problems have been derived from real-world applications and have been explored separately in different publications where problem is solved by a particular approach. We show that our model allows us to address all these tasks by making only a few changes to the original formulation.

We begin by providing an informal description of each of these tasks and describing the corresponding losses that are to be minimized on a training set. Note that these loss minimization problems are **datum-wise** variants inspired by losses found in the literature, and are therefore slightly different. We then describe how these losses can be solved by making simple modifications to the structure of the decision process described in Sect. 3. Experimental results are presented in Sect. 6.3.

4.1 Definitions

Here are descriptions of the different problems we address, and the corresponding *datum-wise* loss minimization problems:

- **Hard budget feature selection** (Kapoor and Greiner 2005) considers that there is a fixed budget on feature acquisition, be it during the training, the inference, per-datum, or globally. We choose to put in place this constraint as a per-datum hard budget during inference. The goal is to maximize classification accuracy while respecting this strict limit on the feature budget. The corresponding loss minimization problem can be written as:

$$\begin{aligned}\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} \Delta\big(y_\theta(\mathbf{x_i}), y_i\big) + \lambda \frac{1}{N}\sum_{i=1}^{N} \big\| z_\theta(\mathbf{x_i})\big\|_0, \\ \text{subject to } \big\| z_\theta(\mathbf{x_i})\big\|_0 \leq M. \end{aligned} \tag{5}$$

- **Cost-sensitive feature acquisition and classification** (Turney 1995; Greiner 2002; Ji and Carin 2007) is an important domain in both feature selection and active learning. The problem is defined by assigning a fixed cost to each feature the classifier can consider. Moreover, the cost of misclassification errors depends on the error made. For example, false positive errors will have a different cost than false negatives. The goal is thus to minimize the overall cost which is composed of both the misclassification cost and also the sum of the cost of all the features acquired for classifying. This task is well-suited for some medical applications where there is a cost associated with each medical procedure (blood test, x-ray, etc.) and a cost depending on the quality of the final diagnosis.

Let us denote ξ the vector that contains the cost of each of the possible features, the datum-wise minimization problem can be written as:

$$\theta^* = \operatorname*{argmin}_\theta \frac{1}{N}\sum_{i=1}^{N} \Delta_{cost}\big(y_\theta(\mathbf{x_i}), y_i\big) + \frac{1}{N}\sum_{i=1}^{N}\big\langle \xi ; z_\theta(\mathbf{x_i})\big\rangle, \tag{6}$$

where Δ_{cost} is a cost-sensitive error loss. Let C be a classification cost matrix such that $C_{i,j}$ is the cost of classifying a datum as i when its real class is j. This cost is generally positive[5] for $i \neq j$, and negative or zero for $i = j$. We can thus define Δ_{cost} as:

$$\Delta_{cost}(i, j) = C_{i,j}. \tag{7}$$

The matrix C is defined *a priori* by the problems one wants to solve.

- **Group feature selection** has been previously considered in the context of the Group Lasso (Yuan and Lin 2006). In this problem, feature selection is considered in the context of groups of features; the classifier can choose to use a certain number of groups of features, but cannot select individual features. Many feature selection tasks present a certain organization in the feature space. For example, a subset of features f_s may all be somehow correlated, and need to be selected together. For example, f_s may represent a discretized real variable, or an ensemble of values that correspond to a single physical test. These groups can either be defined relative to a certain structure already present in the data (Jenatton et al. 2011), or can be used to reduce the dimensionality of the problem. Let us consider the set of n features denoted $\mathcal{F}$ and a set of g groups of features denoted $\mathcal{F}_1 \dots \mathcal{F}_g$ such that $\bigcup_{i=1}^{g} \mathcal{F}_i = \mathcal{F}$. Let us define the set of selected features for a particular datum $\mathbf{x_i}$ as $\mathcal{Z}_\theta(\mathbf{x_i}) = \{j \in \mathcal{F} \text{ s.t. } z_\theta^j(\mathbf{x_i}) = 1\}$. The corresponding datum-wise loss, inspired by the Group Lasso, can be now be written as:

$$\theta^* = \operatorname*{argmin}_\theta \frac{1}{N}\sum_{i=1}^{N} \Delta\big(y_\theta(\mathbf{x_i}), y_i\big) + \lambda\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{g} \mathbb{1}\big(\mathcal{F}_t \subset \mathcal{Z}_\theta(\mathbf{x_i})\big). \tag{8}$$

This loss tries to minimize the number of $\mathcal{F}_t$ groups present in the actual set of selected features. We use $\mathbb{1}(\cdot)$ as a truth function:

$$\mathbb{1}(P) = \begin{cases} 1 & \text{if } P \textit{ True}, \\ 0 & \text{otherwise.} \end{cases}$$

This allows us to quantify the number of groups that have been chosen in $\mathcal{Z}_\theta(\mathbf{x_i})$, so that we may minimize their number.

- **Relational feature selection:** Finally, we consider a more complex problem where features are organized in a complex structure. This problem, which we call **relational feature selection**, is inspired by *structured sparsity* (Huang et al. 2009). We imagine a couple of problems that can fall into this category:

[5] Note that a positive cost implies a penalty in the minimization sense.

Table 1 Proposed tasks and corresponding learning problems

Task	Loss minimization problem
Hard budget	$\theta^* = \operatorname{argmin}_\theta \frac{1}{N}\sum_{i=1}^N \Delta(y_\theta(\mathbf{x_i}), y_i) + \lambda \frac{1}{N}\sum_{i=1}^N \|z_\theta(\mathbf{x_i})\|_0$ subject to $\|z_\theta(\mathbf{x_i})\|_0 \le M$
Cost-sensitive	$\theta^* = \operatorname{argmin}_\theta \frac{1}{N}\sum_{i=1}^N \Delta(y_\theta(\mathbf{x_i}), y_i) + \frac{1}{N}\sum_{i=1}^N \langle \xi; z_\theta(\mathbf{x_i})\rangle$
Grouped features	$\theta^* = \operatorname{argmin}_\theta \frac{1}{N}\sum_{i=1}^N \Delta(y_\theta(\mathbf{x_i}), y_i) + \lambda \frac{1}{N}\sum_{i=1}^N \sum_{t=1}^g \mathbb{1}(\mathcal{F}_t \subset \mathcal{Z}_\theta(\mathbf{x_i}))$
Relational features	$\theta^* = \operatorname{argmin}_\theta \frac{1}{N}\sum_{i=1}^N \Delta(y_\theta(\mathbf{x_i}), y_i)$ $+ \frac{1}{N}\sum_{i=1}^N \sum_{f,f' \in \mathcal{Z}_\theta(x_i)} \mathit{Related}(f, f')(\lambda - \gamma) + \gamma$

- Conditional features, where one or a subset of features can only be selected depending on the previously acquired features.
- Constrained features, where the cost of acquiring a particular feature depends on the previously acquired features. For example, in computer vision, one can constrain a system to acquire values of pixels that are close in an image—see Sect. 6.3.3.

Let us define a boolean function that tells us if two features are related:

$$\mathit{Related}: \quad \begin{cases} \mathcal{A}_f \times \mathcal{A}_f \to \{1, 0\}, \\ \mathit{Related}(f, f') = 1 \quad \text{if related, else 0.} \end{cases} \tag{9}$$

In the case of this function, the relation can be any imaginable constraint that can be calculated simply by considering f and f'.

The underlying idea is that acquiring features that are somehow related is less expensive than acquiring features that do not share a relation. The corresponding loss can be written as:

$$\theta^* = \operatorname*{argmin}_\theta \frac{1}{N}\sum_{i=1}^N \Delta\big(y_\theta(\mathbf{x_i}), y_i\big) + \frac{1}{N}\sum_{i=1}^N \sum_{f,f' \in \mathcal{Z}_\theta(x_i)} \mathit{Related}\big(f, f'\big)(\lambda - \gamma) + \gamma. \tag{10}$$

Here, the term $\mathit{Related}(f, f')(\lambda - \gamma) + \gamma$ equals λ if f and f' are related, and γ otherwise. In that definition, the cost of acquiring non related features is γ while the cost of related features is λ. Therefore, to encourage the use of related features, one simply needs to set $\gamma > \lambda$.

The different problems are summarized in Table 1.

4.2 Adapting the datum-wise classifier

In the rest of this section, we will show how these different tasks can be easily solved by making slight modifications to the model proposed in Sect. 3. We will first cover the general idea underlying these modifications, and then detail for each of the previously described tasks how they can be handled by the sequential process. This section aims at showing that our approach is not only a novel way to compute sparse models, but also an original

Table 2 Summary of the modifications made for incorporating the different variants in the original model

Task	Decision process modification	Commentary
Hard budget	$\mathcal{A}(\mathbf{x},\mathbf{z}) = \begin{cases} \mathcal{A}_f(\mathbf{x},\mathbf{z}) \cup \mathcal{A}_y(\mathbf{x},\mathbf{z}) \text{ if } \Vert\mathbf{z}\Vert_0 < M \\ \mathcal{A}_y(\mathbf{x},\mathbf{z}) \text{ if } \Vert\mathbf{z}\Vert_0 = M \end{cases}$	Allows users to choose a minimum level of sparsity. Reduces training complexity.
Cost-sensitive	$r(\mathbf{x_i},\mathbf{z},a) = \begin{cases} -\xi_i \text{ if } a \in \mathcal{A}_f \\ -C_{a,y_i} \text{ if } a \in \mathcal{A}_y \end{cases}$	Well-suited for features with variable costs.
Grouped features	$\mathcal{A}_f = \mathcal{A}_{group}$ $\mathcal{T}((\mathbf{x},\mathbf{z}),a_j) = \left(\mathbf{x}, \mathbf{z} + \sum_{i \in \mathcal{F}_j} \mathbf{e_i}\right)$	Well adapted to features presenting a grouped nature. Complexity is reduced.
Relational features	$r(\mathbf{x},\mathbf{z},a_j) = \begin{cases} -\lambda \text{ if } \forall f \in \mathcal{Z}(\mathbf{x}), Related(f_j,f) = 1 \\ -\gamma \text{ otherwise} \end{cases}$	Naturally suited for complex feature inter-dependencies.

and flexible approach that allows one to easily imagine many different models for solving complex classification problems.

4.2.1 General idea

Our model is based on the idea[6] of proposing a sequential decision process where the long term reward obtained by a policy is equal to the negative loss obtained by the corresponding classifier. With such an equivalence, the optimal policy obtained through learning is thus equivalent to an optimal classifier for the particular loss considered. In order to deal with the previously proposed classification problems, we simply modify DWSM's MDP in order to correspond to the new loss function.

The main advantage of making only small changes to the structure of the decision process is that we do not need to change the principles of the learning and inference algorithms, thus resulting in new classifiers that are very simple to specify and implement. We believe that this approach is well suited for solving real-world classification tasks that often corresponds to complex loss functions.

In order to deal with the four different proposed tasks, we have to make modifications to the MDP by changing: the reward function $r(\cdot,\cdot,\cdot)$, the action set $\mathcal{A}(\cdot,\cdot)$, and/or the transition function $\mathcal{T}(\cdot,\cdot)$.

To be able to adapt our model to these new problems, we do not need to modify either the feature projector, $\Phi(\cdot,\cdot)$, the actual definition of the state space, the learning algorithm, or the inference algorithm.

In the next part, we detail which modifications we have made on the original decision process for each of the four new tasks. We do not detail the derivation for the equivalence between the long term reward and the optimized loss function. However, these derivations are easy to achieve following the steps presented in Sect. 3.2. A summary of the modifications is made in Table 2.

[6] See Sect. 3.

4.2.2 *Hard budget*

In order to integrate the hard budget constraint into our model, we modify the set of possible actions $\mathcal{A}(\mathbf{x},\mathbf{z})$ such that:

$$\mathcal{A}(\mathbf{x},\mathbf{z}) = \begin{cases} \mathcal{A}_f(\mathbf{x},\mathbf{z}) \cup \mathcal{A}_y(\mathbf{x},\mathbf{z}) & \text{if } \|\mathbf{z}\|_0 < M, \\ \mathcal{A}_y(\mathbf{x},\mathbf{z}) & \text{if } \|\mathbf{z}\|_0 = M. \end{cases}$$

This new action set function allows the model to either choose a new feature, or classify if the number of selected features $\|\mathbf{z}\|_0$ is inferior to M. When $M-1$ features have been selected, only classification actions may be performed by the classifier.

One advantage of this constraint is to reduce the complexity of the training algorithm, since the maximum size of a trajectory in the decision process is now M. This has the effect of limiting the length of each rollout, thus making the training simulation much faster to compute.

4.2.3 *Cost-sensitive*

We express variable feature costs in our model by modifying the reward function. While DWSM uses a $-\lambda$ reward for all feature actions, we use a variable reward depending on the feature being selected. Additionally, we modify the reward obtained when classifying to make it dependent on the cost matrix C. The reward function can now be written as:

$$r(\mathbf{x_i},\mathbf{z},a) = \begin{cases} -\xi_i & \text{if } a \in \mathcal{A}_f, \\ -C_{a,y_i} & \text{if } a \in \mathcal{A}_y, \end{cases}$$

where ξ_i is the cost of acquiring feature i, as used in Eq. (6), and C is the cost-sensitive error loss defined in Eq. (7).

4.2.4 *Group features*

In the context of our model, implementing group features corresponds only to a slight modification of the action space. We simply define a mapping that allows a specific feature selection action to correspond to the addition of a whole subset of actual datum attributes. To do this, let us define a new set of features actions $\mathcal{A}_{group}$ which will replace $\mathcal{A}_f$:

$$\mathcal{A}_{group} = \{a'_1, \dots, a'_{n'}\},$$

where each a'_j corresponds to acquiring the jth *group* of features—instead of the jth feature. To allow this simultaneous acquisition of features, we can modify the transition function $\mathcal{T}$, and define a new transition function for grouped features:

$$\mathcal{T}\big((\mathbf{x},\mathbf{z}),a_j\big) = \left(\mathbf{x}, \mathbf{z} + \sum_{i \in \mathcal{F}_j} \mathbf{e_i}\right),$$

where $\mathcal{F}_j$ is the jth group of features as defined previously in Sect. 4.1. The new set of possible actions is then the union between $\mathcal{A}_{group}$ and $\mathcal{A}_y$, allowing the classification process to choose, at each step, to either classify or acquire simultaneously a whole group of features.

Because the size of $\mathcal{A}_{group}$ is inferior to the size of $\mathcal{A}_f$, the new MDP corresponds to a learning and inference complexity which is greatly reduced relative to he original problem. This aspect allows us to deal both with datasets where features are "naturally" grouped, but also, to deal with datasets with large number of features, by artificially grouping the features into groups. We consider both of these approaches experimentally in Sect. 6.3.3.

4.2.5 *Relational features*

Contrary to group features where the classifier can choose amongst a set of predefined groups of features, constrained features allow the classifier to discover coherent groups of features in the feature space. This type of constraint is more interesting for particular datasets that represent a spatial or relational representation. For every action $a_j \in \mathcal{A}_f$,[7] let us define the reward as:

$$r(\mathbf{x}, \mathbf{z}, a_j) = \begin{cases} -\lambda & \text{if } \forall f \in \mathcal{Z}(\mathbf{x}),\ \mathit{Related}(f_j, f) = 1, \\ -\gamma & \text{otherwise.} \end{cases}$$

Thus, the classifier is encouraged to choose features that are related to the features previously chosen. Nevertheless, for a certain penalty $\gamma > \lambda$, the classifier can choose to start a new, unrelated feature group. One can use a very high value for γ in order to force the classifier to choose only related features. In that case, the relational features can be easily implemented by reducing the set of the possible *features acquisition* actions, reducing the complexity of the system. This variant is not described in this paper.

4.3 Overview of model extensions

As we have seen in this section, our model is easily adaptable to a large set of sparsity-inspired problems. What we believe to be of particular interest is not only that our model can function under complex constraints, but that adapting it for these constraints is relatively straightforward. Indeed, one can imagine many more sparsity-inspired problems requiring constraints that are not easily dealt with traditional classification approaches, yet that could be easily expressed with our model. For this reason, we strongly believe that our model's adaptability to more complex problems is one of its strong points.

5 Complexity analysis and scaling

Let us focus on the analysis of the complexity of the proposed model. We detail the complexity concerning the initial datum-wise sparse model proposed in Sect. 2, and then detail the complexity of each proposed extension. We discuss the ability of our approach to deal with datasets with a large number of features and propose different possible extensions for reducing the complexity of the approach.

5.1 Complexity of the datum-wise sparse classifier

Inference complexity Inference on an input $\mathbf{x}$ consists in sequentially choosing features, and then classifying $\mathbf{x}$. Having acquired t features, on a dataset with n features and c categories in total, one has to perform $(n - t) + c$ linear computations through the s_θ function in order to choose the best action at each state. The inference complexity is thus $O(N_f \cdot n)$, where N_f is the mean number of features chosen by the system before classifying. In fact, due to the shape of the Φ function presented in Sect. 3.3 and the linear nature of s_θ, the score of the actions can be efficiently incrementally computed at each step of the process by only adding the contribution of the newly added feature to each action's score. This complexity makes the model able to quickly classify very large datasets with many examples. The inference complexity is the same for all the proposed variants.

[7] This corresponds to choosing feature f_j.

Table 3 Learning complexity of the different variants of the Datum-Wise Model. n is the number of features, c the number of classes, N_s is the number of states used for rollouts

Variant	Learning complexity	Remarks
Sparse model	$\mathcal{O}(N_s \cdot T \cdot (n+c)^2)$	Limited to hundreds of features, $T \approx n$.
Hard budget	$\mathcal{O}(N_s \cdot \bar{T} \cdot (n+c)^2)$	Same complexity as the base model, shorter learning time with the budget $\bar{T} \ll n$.
Grouped features	$\mathcal{O}(N_s \cdot T \cdot (\bar{n}+c)^2)$	Same complexity as the base model, much shorter learning time with the number of groups $\bar{n} \ll n$, and $T \approx \bar{n}$.
Relational features	$\mathcal{O}(N_s \cdot T \cdot (n+c)^2)$ to $\mathcal{O}(N_s \cdot T \cdot c^2)$	The complexity depends on the structure of the features. In the extreme case, where features have to be acquired in a fixed order, the complexity[a] is $O(N_s \cdot T \cdot c^2)$ and allows the model to scale linearly relative to features.

[a] Since there is no choice in the features, the number of actions is only $1+c$

Learning complexity As detailed in Sect. 3.4, the learning method is based on Rollouts Classification Policy Iteration. The computational cost of one iteration of RCPI is composed of a simulation cost which corresponds to the time spent making Monte Carlo Simulation using the current policy. This cost takes the following general form:

$$Cost(RCPI) = N_s A \times TA.$$

$A = (n+c)$ is the total number of actions in the MDP (feature selection plus classification actions), TA is therefore the cost of evaluating one trajectory of length T, where each of the T state transition requires querying the value of each of the A actions from the scoring function s_θ. $N_s A \times TA$ is the overall cost of executing the Monte Carlo simulation by evaluating a trajectory for each of the A actions of each of the N_s states.

The complexity of each iteration is therefore $\mathcal{O}(N_s \cdot T \cdot (n+c)^2)$, with $T \approx n$. This implies a learning method which is quasi-cubic w.r.t. the number of features; the base variant of our method is limited to problems with features in the hundreds. The extensions proposed have different learning complexities presented in Table 3; this allows some of them to be used with datasets containing thousands of features.

5.2 Scalability

If the learning complexity of our model is higher than baseline global linear methods, inference is linear. In practice, during training, most of the baseline methods select a subset of variables in a couple seconds to a couple minutes, whereas our method is an order of magnitude slower. The problem encountered is the increase in training time relative to the number of features. Inference, however, is indeed performed at the same speed as baseline methods, which is in our opinion the important factor.

The proposed approach is clearly more suitable for dealing with classification where the problem is complex, involving for example cost-sensitive features, relational features or features that naturally present some structure. There are nevertheless a couple ways to handle datasets with many features using our method:

- The complexity of the hard budget variant is clearly an order of magnitude lower than the complexity of the initial model. This variant is useful when one wants to obtain a very high sparsity by limiting the maximum number of used features to ten or twenty. In this case, this model is faster than the DWSM model and can be learned on larger datasets

- The grouped-features model has a complexity which depends on the number of groups of features. So one possible solution when dealing with many features, is to group these features in a hundred of packets. These groups can be formed randomly, or by hand if the features are naturally organized in a complex structure. Such a use of the model on a large dataset is illustrated in Table 7.
- When dealing with sparse datasets, the learning algorithm can be easily adapted for reducing its complexity. This acquisition of features can thus be restricted (during learning) to the subset of non-null values, strongly reducing the number of possible actions in the MDP to the number of non-null features.
- At last, the use of faster Reinforcement Learning techniques can be a possible solution to fasten the learning phase. Recent techniques have been developed (Dulac-Arnold et al. 2012) allowing to reduce the complexity of our model from $O(N_s \cdot (n+c)^3)$ to $O(N_s \cdot \log(n+c))$ at the price of a final sub-optimal classification policy. These methods will be tested on this task in a future work

6 Experiments

The experimental section is organized as follows: First, we present the results obtained by basic DWSM on 8 binary classification datasets in Sect. 6.1. We analyze the performance and behavior of this algorithm and compare with state-of-the-art methods. We then present results for multiclass classification with this base model in Sect. 6.2. After that, we describe experiments performed with the four extensions to the base model proposed in Sect. 4 on the binary datasets and additional corpora. For brevity, we present only representative results in the core article, while providing results obtained on all binary dataset with the DWSM and its variants in the Appendix.

6.1 Sparse binary classification

The first set of experiments focuses on binary classification. Experiments were run on 8 different UCI (Frank and Asuncion 2010) datasets obtained from the LibSVM Website.[8] The datasets are described in Table 4. For each dataset, we randomly sampled different training sets by taking 10 %, 20 %, 50 % and 90 % of the examples as training examples, with the remaining examples being kept for testing. This sampling was performed 30 times, thus generating 30 train/test pairs for each split of the dataset. We did not use k-fold cross-validation as it imposes the number of possible train/test splits for a particular percentage split, and is therefore not applicable in our situation.

We performed experiments with four different models:

- **LARS** was used as a baseline linear model with L_2 loss and L_1 regularization.
- L_1**-SVM** is an SVM classifier with L_1 regularization, effectively providing LASSO-like sparsity.[9]
- **CART** (Breiman et al. 1984) was used as a baseline decision tree model.
- **Datum-Wise Sequential Model (DWSM)** is the Datum-Wise Sparse model presented above.

[8] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

[9] Using LIBLINEAR (Fan et al. 2008).

Table 4 Binary classification datasets characteristics

Name	Number of features	Number of examples	Number of classes	Task
Australian	14	690	2	Binary
Breast Cancer	10	683	2	Binary
Diabetes	8	768	2	Binary
Heart	13	270	2	Binary
Ionosphere	34	351	2	Binary
Liver Disorders	6	345	2	Binary
Sonar	60	208	2	Binary
Splice	60	1,000	2	Binary

For evaluation, we used a classical accuracy measure which corresponds to 1-*error rate* on the test set of each dataset. The sparsity has been measured as the proportion of features *not* used for the LARS and SVM-L_1 models, and the *mean* proportion of features not used to classify **testing examples** in DWSM and CART. Each model was run with many different hyper-parameters values—the C and ϵ values for LARS and SVM-L_1, the pruning value for CART and λ for DWSM.

Concerning our method, the number of rollout states (step 1 of the learning algorithm) is set to ten states for each learning example and the number of policy iterations is set to ten.[10] Note that experiments with more rollout states and/or more iterations give similar results. Experiments were made using an α*-mixture policy*[11] with $\alpha = 0.7$ to ensure the stability of the learning process—a lower α-value involves less stability while a higher value makes more learning iterations necessary for convergence. The following figures present accuracy/sparsity curves averaged over the 30 runs and also show the variance obtained over the 30 runs.

Figures within this experimental section present average accuracies over the 30 different splits. In the case of L_1-SVM and LARS, models have a fixed number of features, however, in the case of DWSM or decision trees, where the number of features is variable, results are actually representative of multiple sparsities within a same experiment. Horizontal error bars are therefore presented for our models.

By looking at the different curves presented in Appendix,[12] one can see that DWSM outperforms SVM-L_1 and LARS at equivalent sparsity on 7 of the 8 binary datasets. Figure 2 illustrates two sparsity/accuracy curves comparing L_1 approaches with DWSM while Fig. 3 also compare results obtained with CART to L_1 approaches and DWSM.

- On Fig. 2 (left)—corresponding to experiments on the *breast cancer* dataset—one can see that at a level of sparsity of 70 %, we obtain 96 % accuracy while the two baselines obtain about 88 % to 90 %. The same observation also holds for other datasets as shown on Fig. 2 (right). Looking at all the curves given in the Appendix, one can see that our model tends to outperform L_1 models—LARS and SVM-L_1—on seven of eight datasets.[13] In these cases, at similar levels of global sparsity, our approach tends to maintain higher

[10] Stability of the learned policy is usually obtained after 4 to 5 iterations

[11] θ^t is chosen with probability $1 - \alpha$, otherwise the previous α-mixture policy is used.

[12] As said before, only representative results are proposed in the core article.

[13] Those datasets are: Australian, breast-cancer, diabetes, heart, ionosphere, sonar and liver.

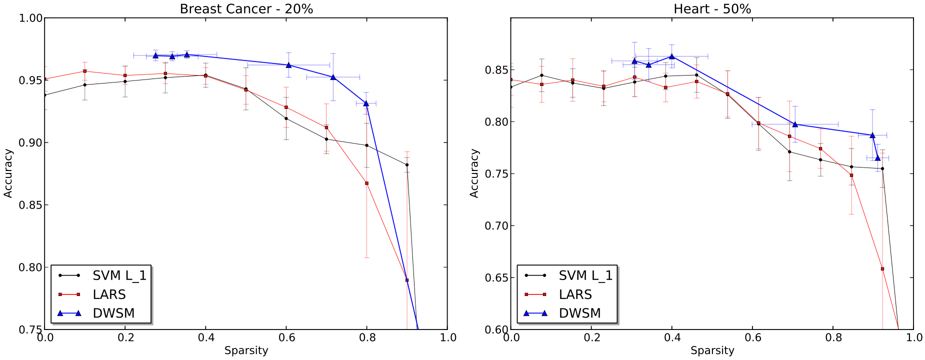


Fig. 2 DWSM vs L1 models: accuracy w.r.t. to sparsity. In both plots, the left side on the x-axis corresponds to a low sparsity, while the right side corresponds to a high sparsity. The performance of the models is usually decreasing when the sparsity increases, except in case of overfitting

accuracy.[14] These results show that DWSM can be competitive w.r.t. L_1 methods, and can outperform these baseline methods on a certain number of datasets.

- We have also compared DWSM with CART. The latter shares some similarities with our sequential approach in the sense that, for both algorithms only some of the features will be considered before classifying a data point. Aside from this point, the two methods have strong differences; in particular, CART builds a global tree with a fixed number of possible paths whereas, DWSM adaptively decides for each pattern which features to use. Note that CART does not incorporate a specific sparsity mechanism and has never been fully compared to L_1 based methods in term of accuracy and sparsity. Figure 3 gives two illustrative results obtained on two datasets. On Fig. 3 (left), one can see that our method outperforms decision trees in term of accuracy at the same level of sparsity. Moreover, DWSM allows one to easily obtain different models at different levels of sparsity, while this is not the case for CART where sparsity could only be controlled indirectly by the pruning mechanism. For some datasets, CART outperforms both DWSM and L_1 based models. An example is given in Fig. 3 (right), where at the 0.9 level of sparsity, CART achieves about 90 % in term of accuracy, while DWSM achieves similar performance to the baseline methods. CART's advantage is most certainly linked to its ability to create a highly non-linear decision boundary.

Qualitative results Figure 4 gives qualitative results on the *breast-cancer* dataset obtained by two models (DWSM and LARS) for a sparsity level of 50 %. This figure is representative of the behavior of the methods on most datasets. The left histogram gives the proportion of testing examples classified by DWSM and L_1 models using a particular feature. The right part of the figure represents the proportion of testing examples that have been classified using a given number of features. From the left histogram, one can see that, whereas some features are used for all data with DWSM (as they are for the global LARS), many others are evenly distributed among the samples. This shows a clear difference in the behavior of global and sequential methods. This would also suggest that global methods do not make the best use of the features. Note that the features that are used for every datum by DWSM

[14] DWSM outperforms L1-norm based approaches when blue curves are above red and black curves.

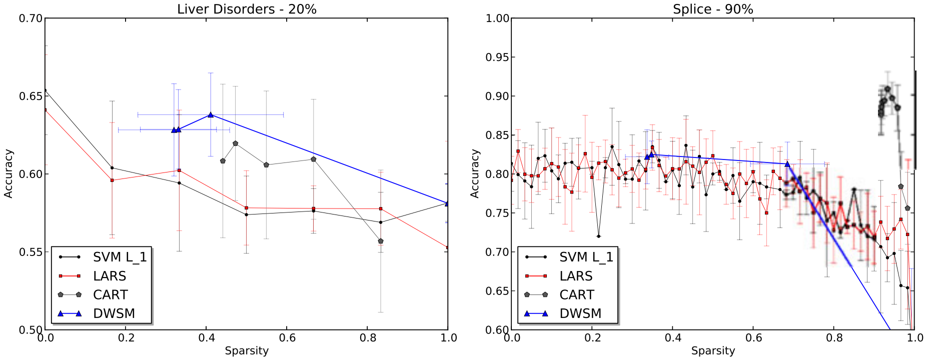
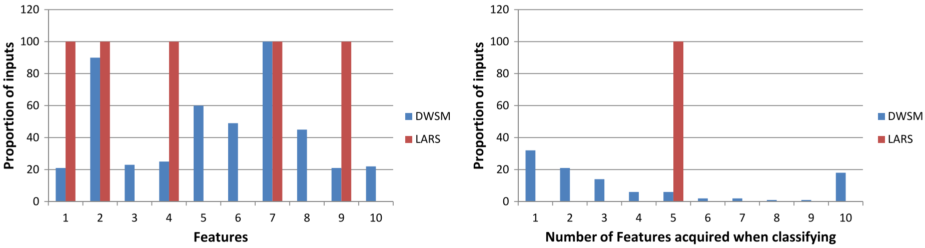


Fig. 3 DWSM and decision trees: while CART can sometimes obtain a better accuracy, control over sparsity is difficult to achieve



Breast-Cancer, Training Size = 20%

Breast-Cancer, Training Size = 20%

Fig. 4 Qualitative results: *Left*: The distribution of use of each feature for label prediction. For example, LARS uses feature 2 for classifying 100 % of the test examples while DWSM uses feature 2 for only 88 % of the test examples. *Right*: The mean proportion of features used for classifying. For example DWSM classifies 20 % of the examples using exactly 2 features while LARS uses 5 features for all the test examples

are also features used by L_1 based approaches. For this example, the sparsity gain w.r.t. the baseline model is obtained through features 1 and 9 that are used only for about 20 % of the DWSM decisions, while they are used in 100 % of the decisions of the LARS model.

The histogram in Fig. 4 (Right) describes the average number of testing examples classified with a particular amount of acquired features. For example, DWSM classifies around 60 % of the examples using no more than 3 features. DWSM mainly uses 1, 2, 3 or 10 features, meaning that it identifies two “levels of difficulty”; some easy examples can be classified using less than 3 features, while for the hard to classify examples, all the features have to be used. A deeper analysis of this behavior shows that almost all the classification mistakes have been made after looking at all 10 features.

Note that DWSM is able to attain better accuracy than the LARS for equivalent sparsity levels. This is due to the fact that DWSM is not bound to a strict set of features for classification, whereas the LARS is. Therefore, it can request more features than the LARS has available for a particularly ambiguous datum. This allows DWSM to have more information in difficult regions of the decision space, while maintaining sparsity for a “simple” datum.

We have analyzed the correlation between the value of λ and the obtained sparsity—Fig. 5. The higher λ is, the higher the sparsity. It is interesting to note that, when the value of λ is too high, the model chooses to select no features and associates each input to the larger category.

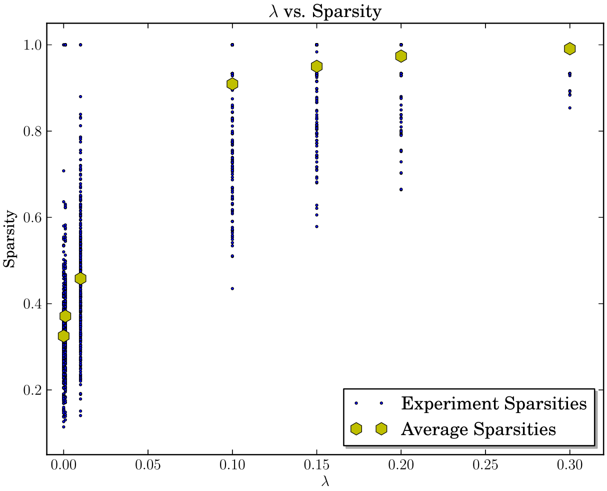


Fig. 5 This plot shows the value of λ for all the experiments run with DWSM, and their corresponding sparsity. The average sparsities are calculated for each λ; they are biased towards a sparsity of 1 by the large number of experiments with no selected features, especially for large values of λ. There are an equal amount of experiments for each value of λ. $\lambda \in \{0, 0.001, 0.01, 0.1, 0.15, 0.2, 0.3\}$. The *two yellow hexagons* on the left actually represent 2 different values of λ: 0, and 0.001

Table 5 Multiclass classification datasets

Name	Number of features	Number of examples	Number of classes
Segment	19	2, 310	7
Vehicle	18	846	4
Vowel	10	1, 000	11
Wine	13	178	3

6.2 Sparse multiclass classification

Multiclass experiments were performed with the same experimental protocal as in Sect. 6.1. The baseline method is a One-vs-All L_1-regularized SVM, and experiments were performed on four different datasets described in Table 5. The experiments show the same general behavior as for the binary datasets. The full results on the four datasets is presented in the Annex. Figure 6 shows example performance for the Wine and Vowel datasets. Sparsity for the SVM-L_1 model is calculated using the total number of features considered while classifying a datum.

The sequential model is particularly interesting with low sparsities, as it is able to maintain good accuracy even with high sparsity. We can see this in Fig. 6 (left), with DWSM able to maintain ∼90 % accuracy while at sparsity of 0.8, whereas the SVM-L_1 model has already sharply decreased in performance. Additionally, extending the model to the multiclass case is completely natural and requires nothing more than adding additional classification actions to the MDP.

6.3 Model extensions

In this section we will present a series of results for the model extensions described in Sect. 4. Contrary to what was done for the base model, we did not attempt to perform extensive comparisons with baseline models but rather wanted to show—using some of the datasets used previously—that the extensions were indeed sound and efficient. The reason

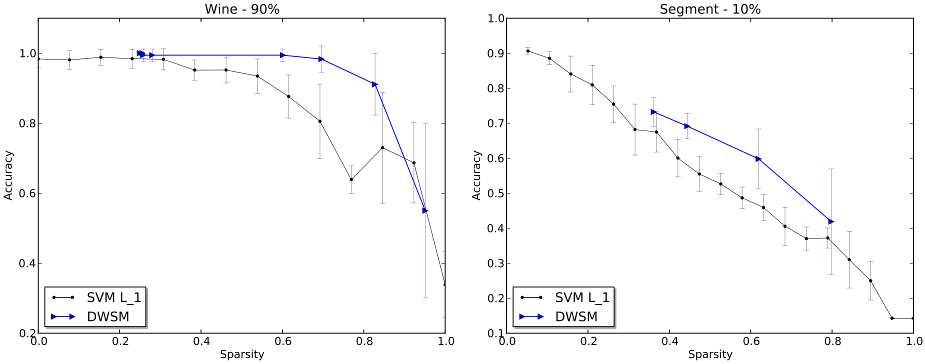


Fig. 6 DWSM vs L_1 models in multiclass: accuracy w.r.t. to sparsity. We can see that as sparsity decreases, performance converges. However, for intermediate sparsities, performance decreases more rapidly with SVM-L_1 models

for this is twofold: First, there is no baseline able to cope with all the extensions, so comparison would require for each problem a specific algorithm for each extension. Second, for some of the extensions, not all the datasets are pertinent.

6.3.1 *Hard budget*

To demonstrate the effects of a hard feature budget, we have set hard budget limits on the datasets described in Table 4. All experiments were performed with $\lambda = 0$, since sparsity in this extension is controlled through the value of M. We have run experiments for $M \in \{2, 5, 10, 20\}$. Figure 7 presents the results obtained on two particular datasets—all other results are provided in the Appendix. In many cases, the addition of a hard constraint allows us to obtain an equivalent or better accuracy relative to the original model, particularly for high levels of sparsity. For example, on Fig. 7 (left), with $M = 5$ and a level of sparsity of 0.8, the hard budget model achieves 77 % in term of accuracy while LARS is about 72 % and L_1-SVM 65 %. The same effect can be seen on Fig. 7 (right). The Hard Budget extension allows for much finer-grained control of sparsity within the model, and this is indeed what is expressed in these figures. The Hard Budget model allows us to increase the accuracy of the DWSM while reducing its learning and inference complexity: this model is many times faster to learn and to infer with—its complexity is $O(M \cdot (n + c))$ instead of $O(N_f \cdot (n + c))$ where M is the budget, n the input dimension, c the number of classes and N_f the mean number of features used by the base DWSM model. Hard budget can be suitable for datasets with many features when one wants to obtain a high sparsity.

6.3.2 *Cost-sensitive feature acquisition*

For the cost sensitive extension, we perform experiments on the *diabetes* dataset, as it has already been used by different authors for cost sensitive classification tasks. This dataset defines one feature extraction cost and one misclassification cost—see Sect. 4.2.3. Experiments were performed with the cost table used by Ji and Carin (2007), originally established by Turney (1995).[15] We used 5-fold cross-validation, and produced the results in Table 6.

[15] We did not use Turney's actual costs as they are difficult to compare to with our metrics.

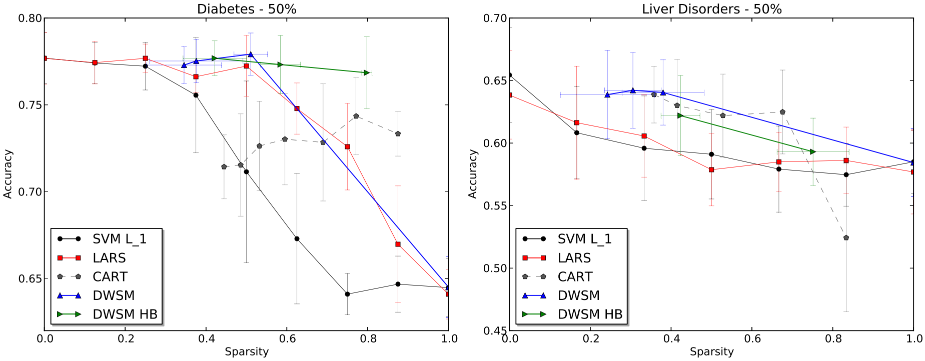


Fig. 7 Hard Budget: Hard Budget models corresponding to different feature budgets M, with $\lambda = 0$. Hard Budget models with a low value of M outperform DWSM and LARS models at high level of sparsity while reducing the learning and inference complexity. Hard-Budget models also allow a more specific tuning of the model's sparsity

Table 6 This table presents results obtained on the cost-sensitive Pima Diabetes dataset. As a reference, results from one of the cost-parameterization of Li and Carin is presented as well

Classifier	Error penalty	Average cost	Accuracy
DWSM	800	181	0.75
DWSM	400	74	0.76
Li and Carin	800	180[a]	0.75[a]
Li and Carin	400	75[a]	0.75[a]

[a] Numerical values from Li and Carin are estimated graphically from their paper

Experiments were run with two different misclassification costs[16] of 400 and 800. We refer to this cost as "Error Penalty" in the results table. We use "Average Cost" to refer to the average per-datum cost, where per-datum cost corresponds to the sum of incurred feature acquisition costs plus the cost of classification. Finally, "Accuracy" is the standard measure of accuracy.[17] In this situation, a smaller cost is better.

The reference results in Table 6 were extracted from page 18 of the article (Ji and Carin 2007). We can see that our cost-sensitive model obtains, in each of the two cases[18] the same average cost as the one obtained by Li and Carin. Accuracy is also equivalent for both models, showing that DWSM is competitive w.r.t. to a cost-sensitive-specific algorithm, while only needing a slight modification to its MDP.

6.3.3 Grouped and relational features

In order to test the ability of our approach to deal with grouped and relational features, we have performed three sets of experiments:

[16] Misclassification cost corresponds to C_{a,y_i} for $a \neq y_i$—see Sect. 4.2.3.

[17] Note that the classifier's goal is to optimize per-datum cost as in Eq. (6), and not accuracy.

[18] Misclassification cost is 800 or 400.

Table 7 This table describes group-feature results for the Gisette and Adult datasets

Classifier	Dataset	# groups	λ	Sparsity	# of features	Accuracy
DWSM	Gisette	10	0.000	0.255	7.4 groups	0.932
			0.005	0.308	6.9 groups	0.937
			0.010	0.310	6.9 groups	0.926
			0.100	0.549	4.5 groups	0.898
LASSO	Gisette	***	***	0.98	100 features	0.962
DWSM	Adult	14	0.000	0.41	8.75 groups	0.82
			0.005	0.677	4.83 groups	0.79
			0.010	1.0	0 groups	0.76[a]
LASSO	Adult	***	***	0.32	95 features	0.83

[a] The final accuracy for the Adult dataset represents the default accuracy of putting all test elements into the biggest class

Artificial random groups The first set consists in considering artificial groups of features on binary classification datasets. These groups of features have been generated randomly and we aim to show that our model is able to learn with grouped features. Although not detailed here, results are generally lower than that of the base models. For example, the best accuracy obtained by Grouped DWSM on the *Sonar* set is of 75 % with 3 groups, which is 5 points lower than both the baseline and standard DWSM. This is not a surprise since groups of features have been chosen randomly and relevant features can be present in different groups; the model may need to acquire all the groups in order to acquire the necessary features, thus decreasing the sparsity of the obtained model. The advantage of grouped features is to decrease the complexity of the model which now only depends on the number of groups, instead of the number of features, thus allowing the model to be trained on datasets with many features. The resulting complexity is $O(N_g \cdot (N_g + c))$ in comparison to $O(n \cdot (n + c))$, where N_g is the number of groups. All these experiments only involve a small number of features. In order to test the ability of the Grouped model to handle many features, we performed experiments using the Gisette dataset created by Isabelle Guyon for the NIPS 2003 Feature Selection Challenge (Guyon et al. 2005). This is a 2-class dataset with 5000 features. We used 10 feature groups—with each group therefore containing 500 features—to train DWSM on Gisette. Results for varying values of λ can be found in Table 7. Note that we do not aim at comparing ourselves with the state of the art on this dataset (which can be found on the challenge website[19] or in the results analysis Guyon et al. 2005) as the best techniques are obtained by complex kernelized methods. From Table 7, we can see that our model obtains good performance ranging from 90 % to 93.2 % accuracy while using on average 4.5 to 7.4 groups of features. Note that Gisette contains many probe features that do not carry any information about the category. This explains why a classical LASSO model is able to obtain a very low sparsity while the grouped model—which learns to select groups that contain relevant features—has a lower sparsity: when a group contains one or more relevant features, it also contains many probe features.

Grouped features based on the structure of the features One advantage of the Grouped Features model is that it can consider datasets where features are naturally organized into

[19] http://www.nipsfsc.ecs.soton.ac.uk/results/?ds=gisette.

groups. This is for example the case for the Adult[20] dataset from UCI, which has 14 attributes, some of which are categorical. These categorical attributes have been expanded to continuous features using discretized quantiles—each quantile is represented by a binary feature. The set of features that corresponds to the quantile of a particular categorical attribute naturally corresponds to a group of features. The continuous dataset is composed of 123 real features grouped in 14 groups. We have created a mapping from the expanded dataset back to the original set of features, and run experiments using this feature grouping. We use the LASSO as a baseline. Table 7 presents the results obtained by our model at three different levels of sparsity and show that our method achieves a 79 % accuracy while selecting on average 4.83 of the 14 groups of features. Results for the LASSO are also presented, although the LASSO's results are *not* constrained to respect the group structure, and furthermore its sparsity corresponds to the sparsity obtained on the expanded dataset, not the sparsity over the initial dataset.

Relational features Finally, we present an experiment performed on image data which allows us to consider both group and relational constraints. Experiments have been performed on the MNIST (LeCun et al. 1998) dataset with the relational model described in Sect. 4.2.5. We have used the following two constraints: (i) First, we have put in place a group mapping on the pixels that corresponds to a 4 × 4 grid of blocks. (ii) Secondly, we have forced the model to focus on contiguous blocks of pixels i.e. the cost of acquiring a block of pixels touching a previously acquired block is lower than the cost of acquiring a block which is

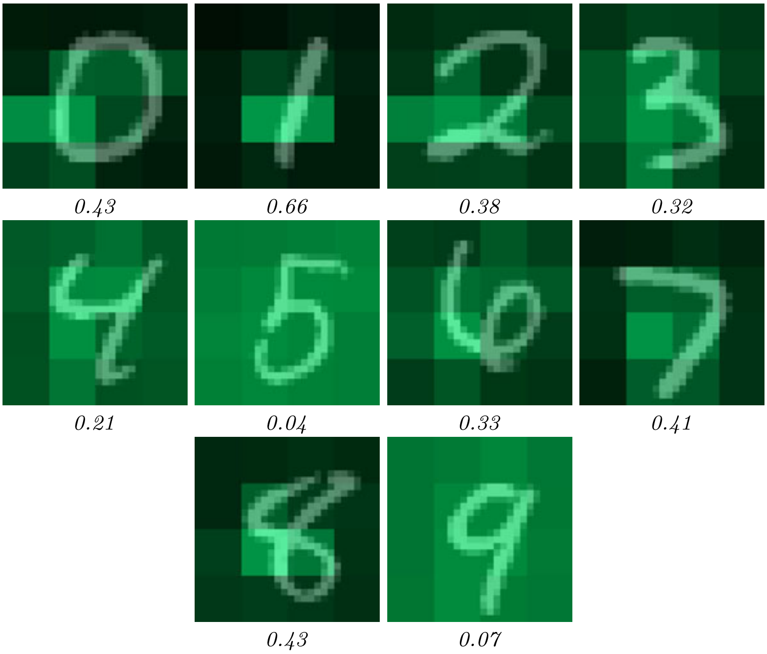


Fig. 8 Average utilization of each feature block for correctly classified data, grouped by class. The corresponding sparsity is below the image. Lighter squares represent more frequent usage

[20] http://archive.ics.uci.edu/ml/datasets/Adult.

further away. We then make use of spatial relations between groups of pixels. Referring to the relational model in Sect. 4.2.5, the *Related*$(\cdot, \cdot)$ function tells us whether two feature blocks are contiguous in the image.

A visualization of the feature acquisition frequencies is presented in Fig. 8. For a given class (a digit 0–9) we compute the mean selection frequency of each block in the image for all correctly identified images. We then represent this frequency overlaid onto a sample image. For a given sample, the brighter a block of pixels, the more it has been used for classifying this digit. Thus, we can see what information the classifier is attempting to access when it is correctly classifying a datum into a particular class. For example: The number 1 is quickly identified by just looking at the two central blocks of features, while the system tends to explore all the possible blocks of features when classifying a 5, which is a more complex digit.

7 Related work

We begin by providing an overview of feature selection techniques that correspond that are close to the datum-wise sparse model presented in Sect. 2. Then, for each of the proposed extensions, we describe the works that address similar problems. Note that none of the following citations correspond to a model that can deal with all these classification problem in an unified way.

Features selection and sparsity Datum-Wise Feature Selection positions itself in the field of feature selection, a field that has seen a good share of approaches (Guyon and Elisseefi 2003). Our approach positions itself between two main veins in feature selection: embedded approaches and wrapper approaches.

Embedded approaches include feature selection as part of the learning machine. These include algorithms solving the LASSO problem (Tibshirani 1994), and other linear models involving a regularizer with a sparsity-inducing norm ($L_{p\in[0;1]}$-norms such as Elastic Net, Zou and Hastie 2005 and group LASSO, Yuan and Lin 2006). These methods are very different from our proposed method, and rely on the direct minimization of a regularized empirical risk. These approaches are very effective at these specific tasks, but are difficult to extend to more complex risks that are neither continuous nor differentiable. Nevertheless, some interesting work has been done along the lines of finding surrogate losses for more structured forms of sparsity (Jenatton et al. 2011). We believe our method is nevertheless more naturally expressive for these types of problems, as its optimization criteria is not subject to any constraints on continuity or derivability.

Wrapper approaches aim at searching the feature space for an optimal subset of features that maximizes the classifier's performance. Searching the entire feature space is very quickly intractable and therefore various recent approaches have been proposed to restrict the search using genetic programming (Girgin and Preux 2008) or UCT-based algorithms (Gaudel and Sebag 2010). We were encouraged by these works, as the use of a learning approach to direct the search for feature subsets in the feature graph is very similar in spirit to our approach. We differentiate our approach from these through its datum-wise nature, which is not considered by either of the aforementioned articles.

Regarding the datum-wise nature of our algorithm, the classical model that shares some similarities in terms of inference process is the Decision Tree (Quinlan 1993). During inference with a Decision Tree, feature usage is in effect datum-dependent. In contrast to our method, Decision Trees are highly non-linear and as far as we know, have never been studied in terms of sparsity. Moreover, the learning algorithm is very different to the one proposed in this paper, and Decision Trees are not easily generalizable to more complex problems described in Sect. 4. Nevertheless, Decision Trees prove to be perform very well in situations where strong sparsity is imposed.

Cost sensitive classification The particular extensions presented in Sect. 4 have been inspired by various recent works. Cost-sensitive classification problems have been studied by Turney (1995) and Greiner (2002). The model proposed by Turney (1995) is an extension of decision trees to cost-sensitive problems, using a certain heuristic to discourage the use of costly features. Greiner (2002) models this task as a sequential problem. However, the formalism used by Greiner is different from the one we propose, and restricted to cost-sensitive problems.

Hard budget classification Hard Budget classification has been considered before, Kapoor and Greiner (2005), in the context of Active Learning. Modelization as an MDP is suggested in the article, but is not performed. Hard Budget classification is primarily motivated by its more fine-grained ability to tune the sparsity, as well as the inherent speedups it provides in the complexity of the learning phase.

Grouped features Many datasets inherently provide some form of group or relational structure, and grouped features have been recently proposed as an extension to the LASSO problem called Group-LASSO (Yuan and Lin 2006). Relational features have also been studied in different papers about structured sparsity (Huang et al. 2009; Jenatton et al. 2011), which also base themselves on LASSO-derived resolution algorithms. Additionally, these papers consider global sparsity and are not datum-wise. They are based on a continuous convex formulation of the sparse L_1 regularized loss and are thus very different from our approach. DWSM provides a much richer expressivity relative to these methods, at the cost of a more complex resolution algorithm.

All the different approaches to our extensions have not been previously brought together under one framework as far as we can tell, additionally many more extensions can be imagined, with the ability to adapt to the fine-grained constraints of real-world problems.

Classification as a sequential problem At last, the idea of using sequential models for classical machine learning tasks has recently seen a surge of interest. For example, there have been sequential models proposed for structured classification (Daumé and Marcu 2005; Maes et al. 2009). These methods leverage Reinforcement Learning approaches to solved more 'traditional' Structured Prediction tasks. Although they are specialized in the prediction of structured data, and do not concentrate on aspects of sparsity or feature selection, the general idea of applying RL to ML tasks is in the same vein of work as DWSM.

The authors have previously presented an original sequential model for text classification (Dulac-Arnold et al. 2011), and there has been similar work using Reinforcement Learning techniques for self-terminating anytime classification (Póczos et al. 2009). These approaches can be considered as more constrained versions of the problem proposed in this paper, since the only criteria being learned is *when* to stop asking for more information, but not *what* information to ask for. Nevertheless, these approaches provide the base intuition for datum-wise approaches.

The most similar Reinforcement Learning works are the paper by Ji and Carin (2007) and the (still unpublished) paper by Rückstieß et al. (2011) which proposes MDP models for cost-sensitive classification. Both of these papers have formalizations that are similar to ours, yet concentrate on cost-sensitive problems. We compare ourselves to experiments performed by Ji and Carin in Sect. 6.3.2.

8 Conclusion

In this article we have introduced the concept of datum-wise classification, where we learn simultaneously a classifier and a sparse representation of the data that adapts to each new datum being classified. For solving the combinatorial feature selection problem, we have proposed a sequential approach where we have modeled the selection—classification problem as a MDP. Learning this MDP is performed using an algorithm inspired by Reinforcement Learning. Solving the MDP is shown to be equivalent to minimizing a L_0 datum wise regularized loss for the classification problem.

This base model has then been extended to different families of feature selection problems: cost-sensitive, grouped features, hard budget and structured features. The proposed formalism can be easily adapted to any of these problems and thus provides a fairly general framework for datum wise sparsity.

Experimental results on 12 datasets have shown that the base model is indeed able to learn data dependent sparse classifiers while maintaining a good classification accuracy. The potential of the 4 extensions to the base model, has been demonstrated on different datasets. All of them solve a specific sparsity problem while requiring only slight changes to the initial model. We believe that this model might be easily adapted to other complex classification problems while requiring only slight changes to the MDP.

For inference, the model complexity is similar to a classical—non datum dependent—sparse classifier. Training the MDP remains however more costly than for global classification approaches. A couple of directions for future work are being considered: the first one consists in using more efficient RL-inspired algorithm such as Fitted Q-Learning, which could greatly reduce the time spent during training. Another possible extension is to remove—during learning—features that are generally judged as irrelevant by the system i.e. features that are never or rarely used for classifying data. In that case, the system only keeps in memory a subset of the possible features and thus reduces the dimensionality of the training space. Finally, a more prospective research direction is to consider a sequential process that is also able to create new features—by combining existing features—opening the way to **feature *construction***.

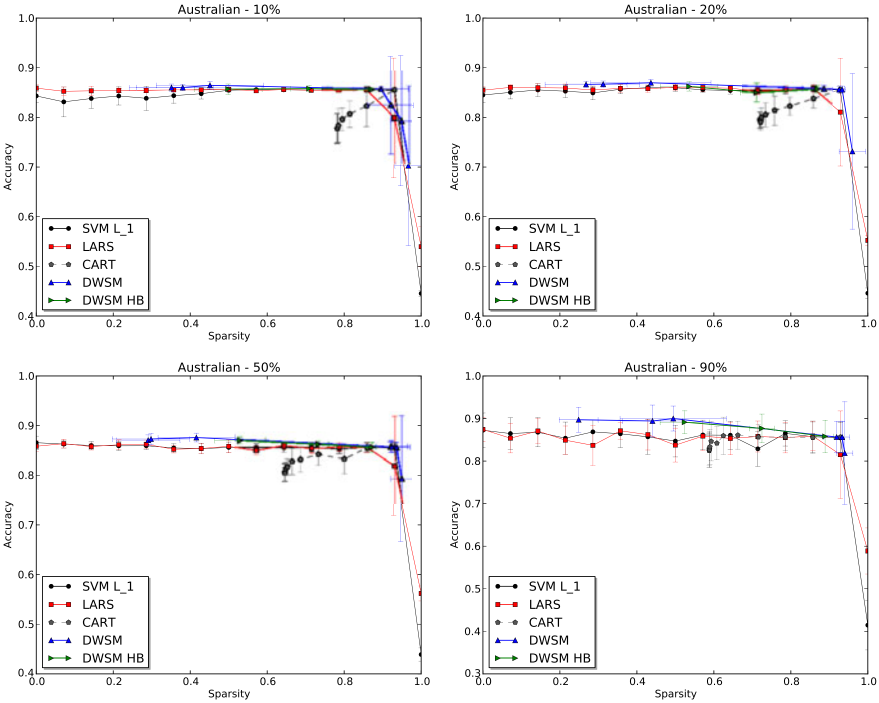
Acknowledgements This work was partially supported by the French National Agency of Research (Lampada ANR-09-EMER-007). The authors gratefully acknowledge the many discussions with Dr. Francis Maes.

Appendix

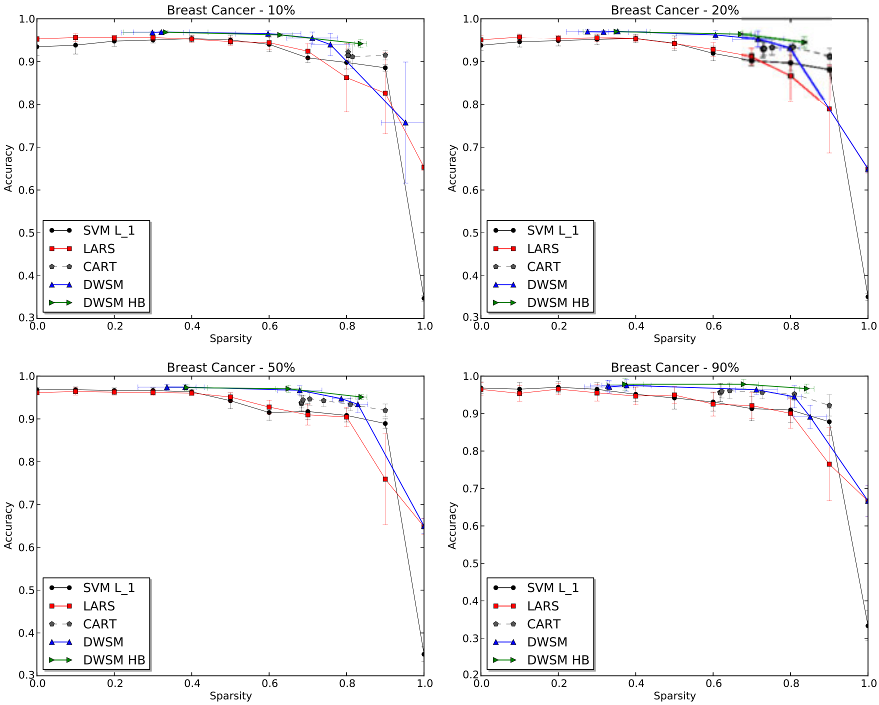
This appendix provide all the accuracy/sparsity curves obtained for the 8 binary datasets, on 4 different training sizes. The models presented in the curves are:

- The two L_1-based models: LARS and SVM-L_1.
- The base sparse sequential model: DWSM.
- The CART decision trees.
- The Hard Budget model HB with budget values in $\{2, 5, 10, 20\}$. For these models, we only report the values obtained with $\lambda = 0$.
- For the multiclass datasets, the DWSM model has been run with $\lambda \in \{0, 0.01, 0.05, 0.1\}$ over the 30 runs.

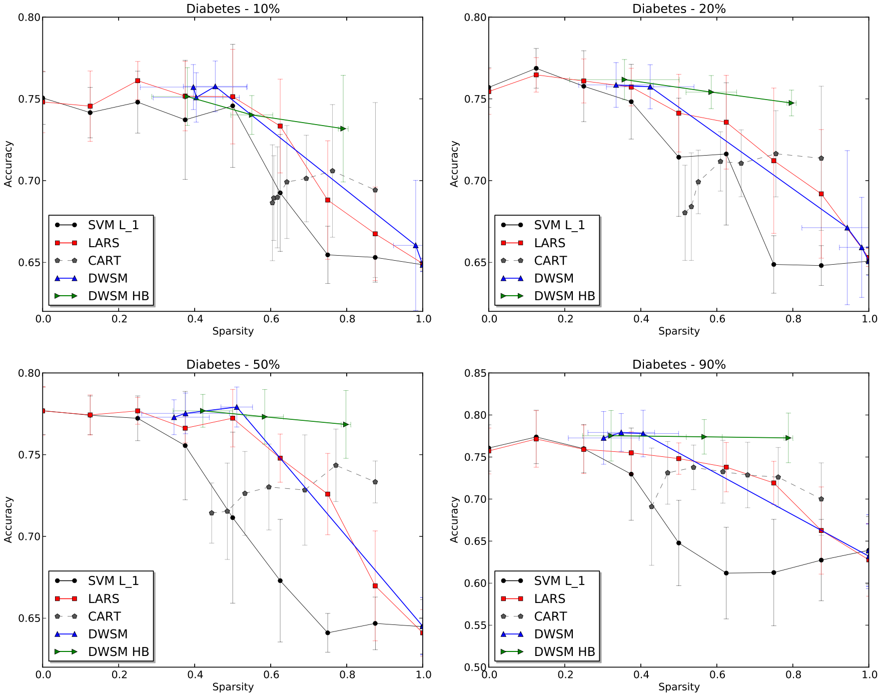
In short, when the blue curves are above the red and black curves, our datum-wise model outperforms classical L_1 based models. When the gray curve is above the other curves, it means that CART is the best sparse algorithm. For each dataset, we present 4 figures corresponding to 4 different training sizes.



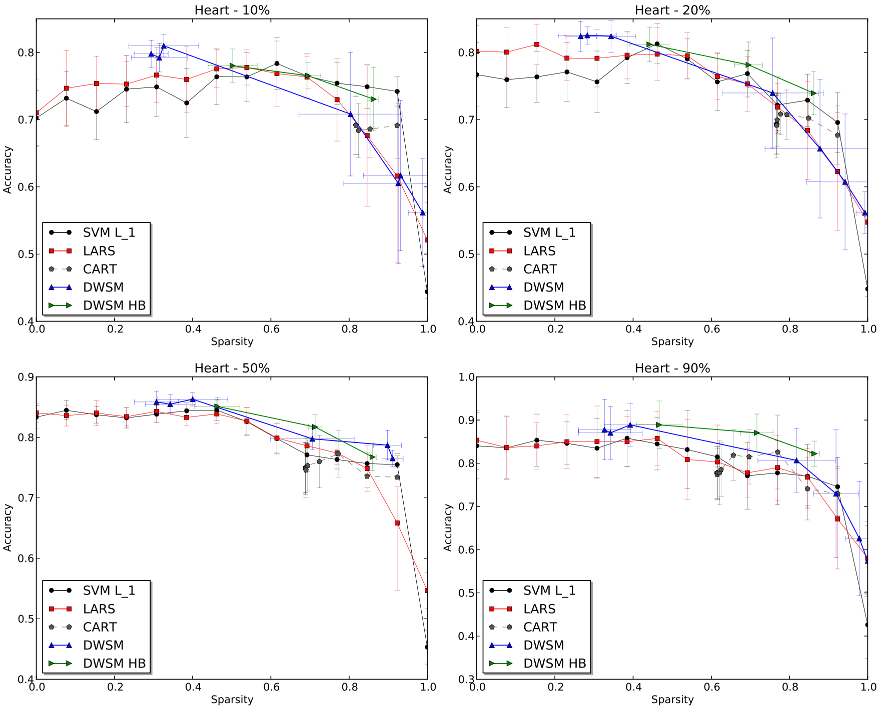
Australian



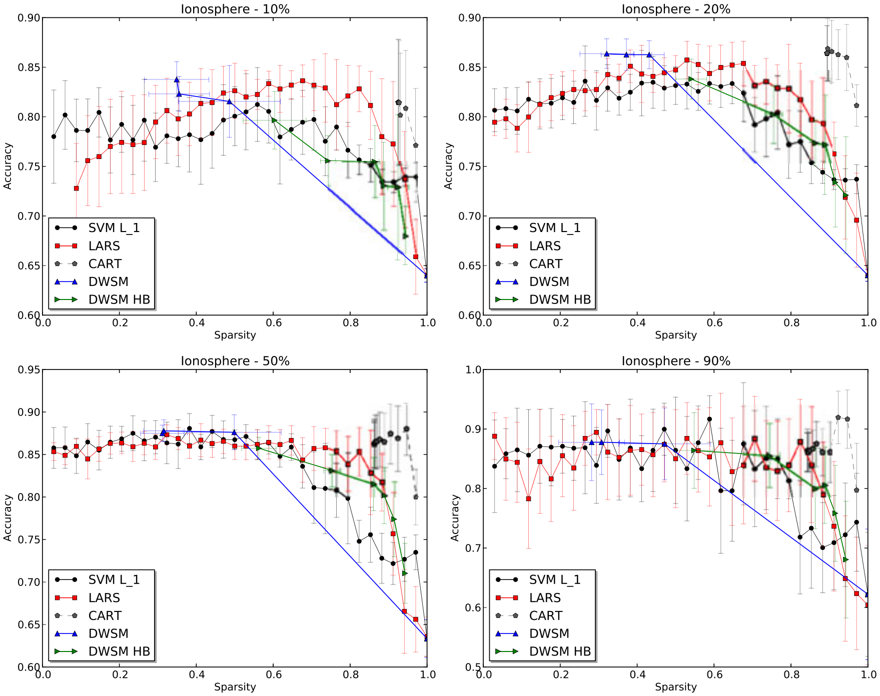
Breast Cancer



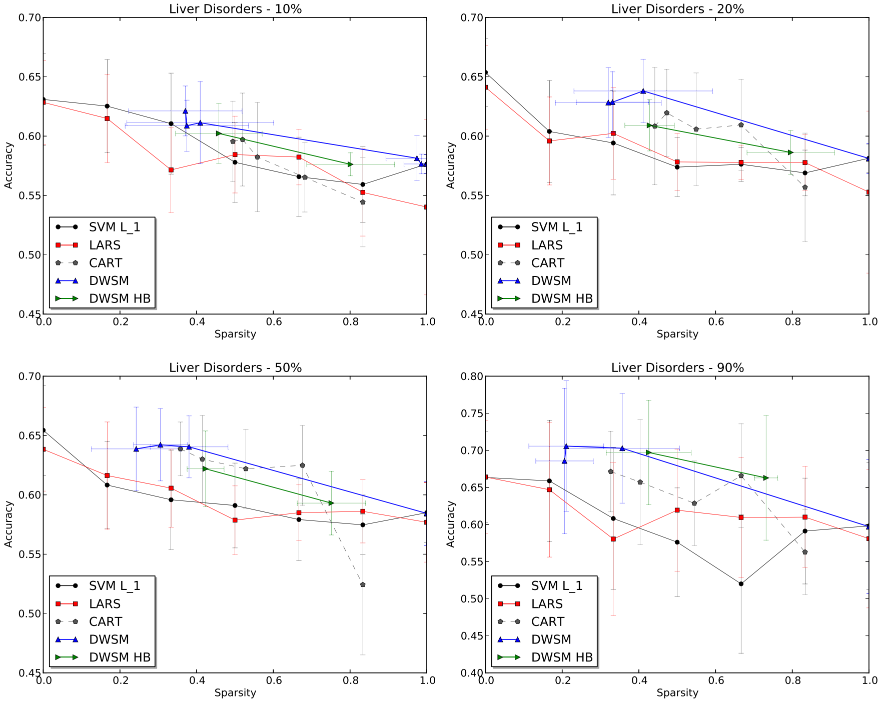
Diabetes



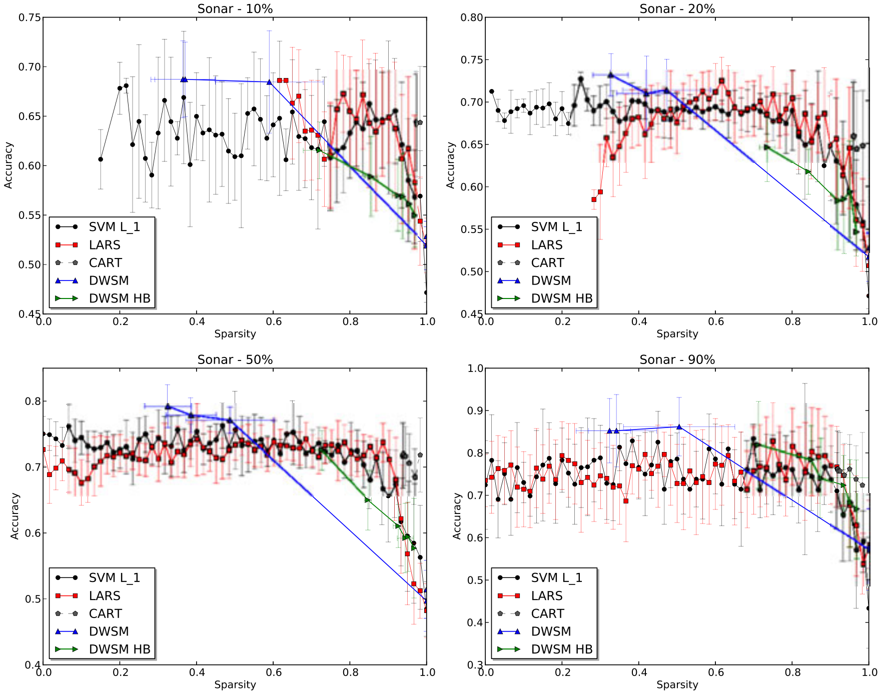
Heart



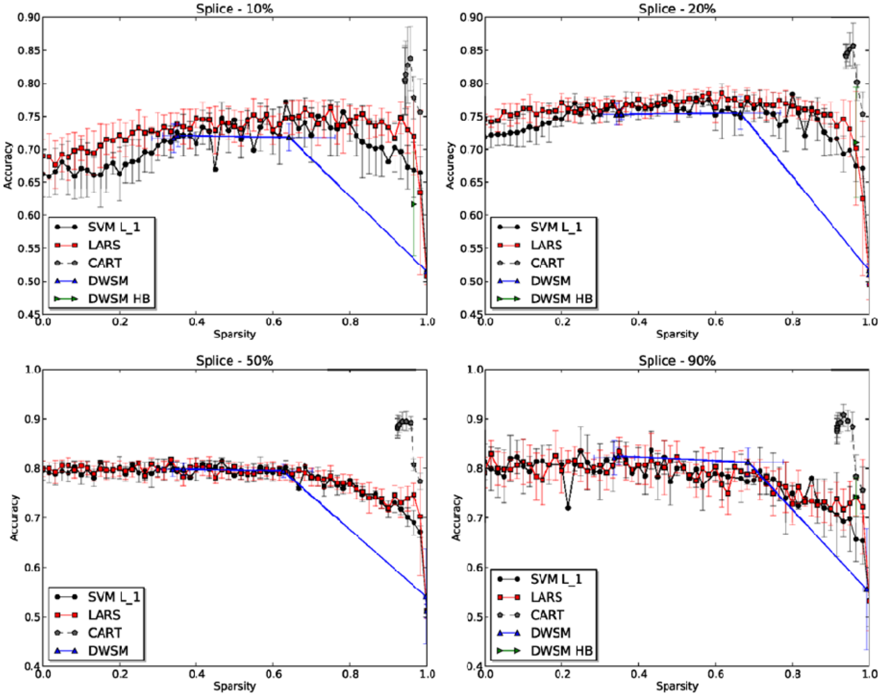
Ionosphere



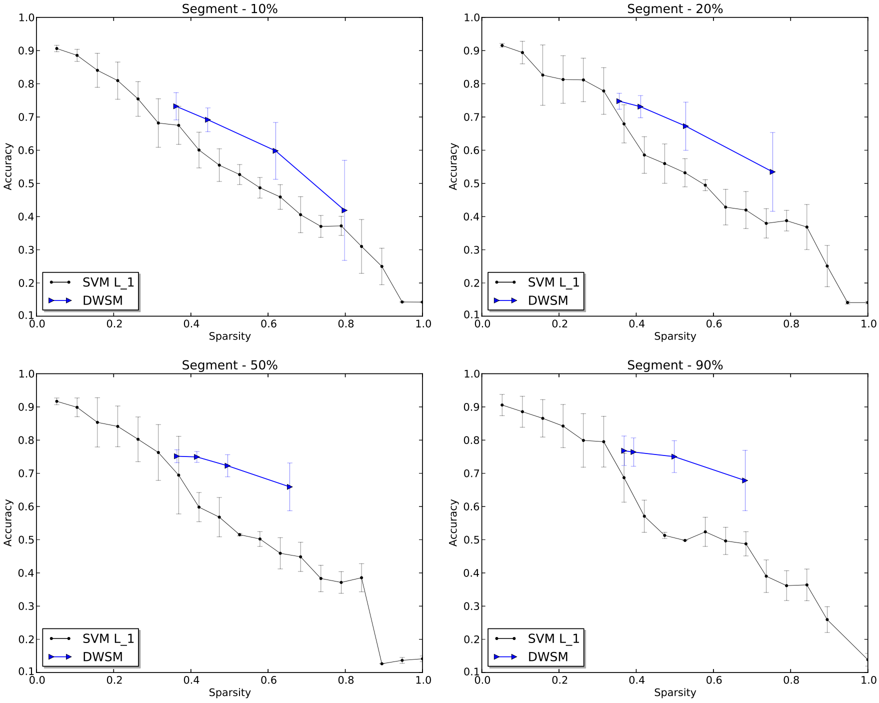
Liver Disorders



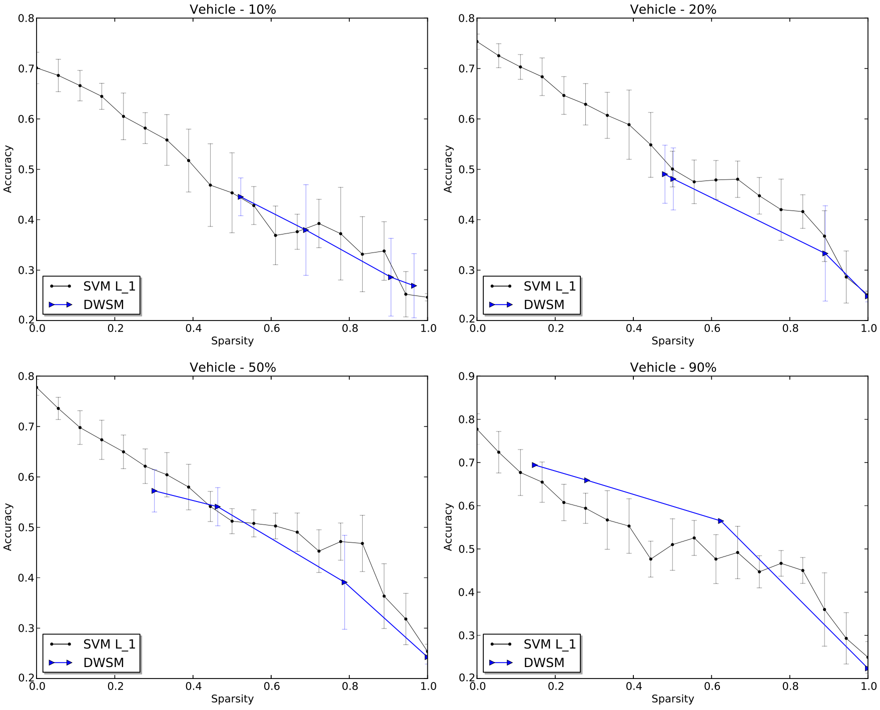
Sonar



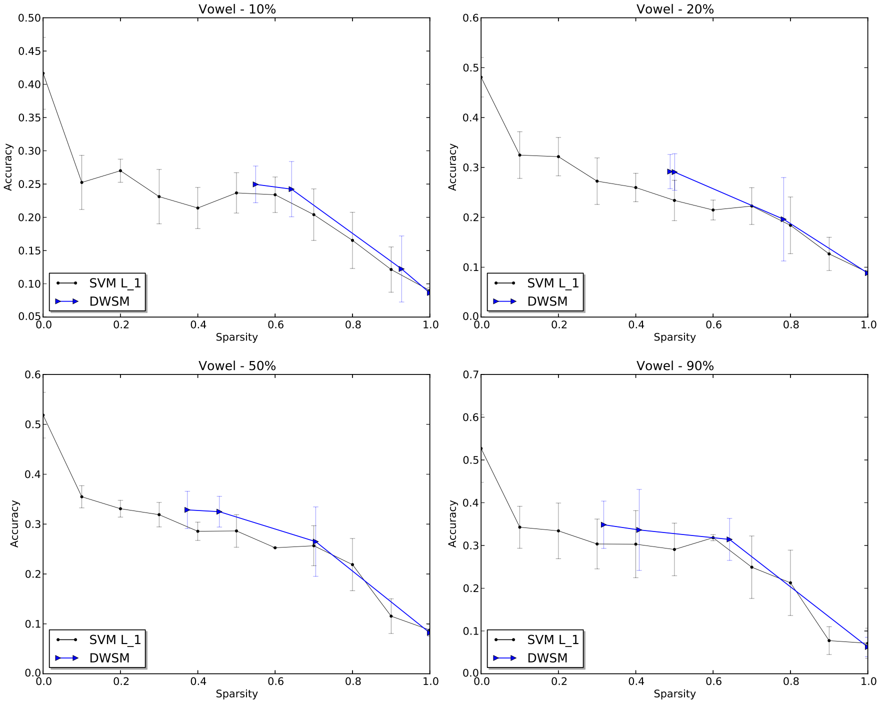
Splice



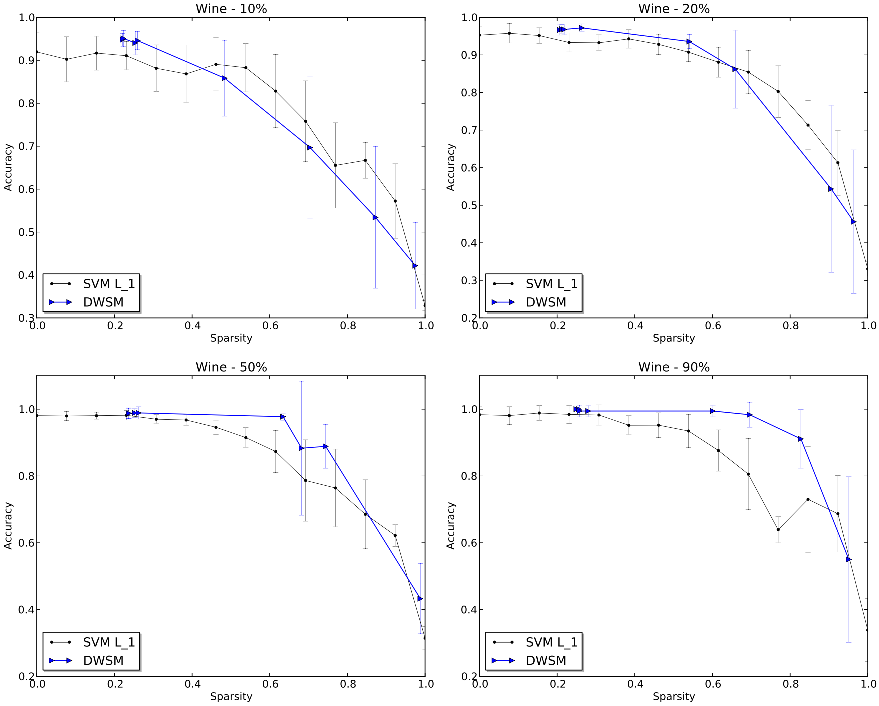
Segment (Multiclass)



Vehicle (Multiclass)



Vowel (Multiclass)



Wine (Multiclass)

References

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Belmont: Wadsworth.

Daumé, H. III, & Marcu, D. (2005). Learning as search optimization: approximate large margin methods for structured prediction. In *Proceedings of ICML* (pp. 169–176). New York: ACM.

Dulac-Arnold, G., Denoyer, L., & Gallinari, P. (2011). Text classification: a sequential reading approach. In *Lecture notes in computer science: Vol. 6611. Proceedings of ECIR* (pp. 411–423). Berlin: Springer.

Dulac-Arnold, G., Denoyer, L., Preux, P., & Gallinari, P. (2012). Fast reinforcement learning with large action sets using error-correcting output codes for MDP factorization. In *Proc. of ECML*.

Efron, B., Hastie, T., & Johnstone, I. (2004). Least angle regression. *Annals of Statistics*, *52*(4), 1390–1400. doi:10.1016/j.neuroimage.2010.05.017.

Fan, R., Chang, K., Hsieh, C., Wang, X., & Lin, C. (2008). LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, *9*, 1871–1874.

Frank, A., & Asuncion, A. (2010). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml.

Gaudel, R., & Sebag, M. (2010). Feature selection as a one-player game. In *Proceedings of ICML*.

Girgin, S., & Preux, P. (2008). Feature discovery in reinforcement learning using genetic programming. In *Proceedings of European conference on genetic programming*, November.

Greiner, R. (2002). Learning cost-sensitive active classifiers. *Artificial Intelligence*, *139*(2), 137–174. doi:10.1016/S0004-3702(02)00209-6.

Guyon, I., & Elisseefi, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*(7–8), 1157–1182. doi:10.1162/153244303322753616.

Guyon, I., Gunn, S., & Ben-Hur, A. (2005). Result analysis of the NIPS 2003 feature selection challenge. In *Proceedings of NIPS*.

Har-Peled, S., Roth, D., & Zimak, D. (2002). Constraint classification: a new approach to multiclass classification. In *Proceedings of NIPS*.

Huang, J., Zhang, T., & Metaxas, D. (2009). Learning with structured sparsity. In *Proceedings of ICML*.

Jenatton, R., Audibert, J. Y., & Bach, F. (2011). Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, *12*, 2777–2824.

Ji, S., & Carin, L. (2007). Cost-sensitive feature acquisition and classification. *Pattern Recognition*, *40*(5), 1474–1485. doi:10.1016/j.patcog.2006.11.008.

Kanani, P. H., & McCallum, A. K. (2012). Selecting actions for resource-bounded information extraction using reinforcement learning. In *Proceedings of ACM international conference on web search and data mining, WSDM'12* (pp. 253–262). New York: ACM.

Kapoor, A., & Greiner, R. (2005). Learning and classifying under hard budgets. In *Proceedings ECML* (pp. 170–181). Berlin: Springer.

Lagoudakis, M. G., & Parr, R. (2003). Reinforcement learning as classification: leveraging modern classifiers. In *Proceedings of ICML*.

Lazaric, A., Ghavamzadeh, M., & Munos, R. (2010). Analysis of a classification-based policy iteration algorithm. In *Proceedings of ICML* (pp. 607–614).

LeCun, Y., Bottou, L., & Bengio, Y. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Louradour, J., & Kermorvant, C. (2011). Sample-dependent feature selection for faster document image categorization. In *Proceedings of ICDAR* (pp. 309–313).

Maes, F., Denoyer, L., & Gallinari, P. (2009). Structured prediction with reinforcement learning. *Machine Learning Journal*, *77*(2–3), 271–301.

Póczos, B., Abbasi-Yadkori, Y., Szepesvári, C., Greiner, R., & Sturtevant, N. (2009). Learning when to stop thinking and do something! In *Proceedings of ICML*, NYC, USA. doi:10.1145/1553374.1553480.

Puterman, M. L. (1994). *Markov decision processes: discrete stochastic dynamic programming*. New York: Wiley-Interscience.

Quinlan, J. (1993). *C4.5: programs for machine learning*. San Mateo: Morgan Kaufmann.

Rückstieß, T., Osendorfer, C., & van der Smagt, P. (2011). Sequential feature selection for classification. In *Australasian conference on artificial intelligence*.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: an introduction*. Cambridge: MIT Press.

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

Turney, P. (1995). Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. *The Journal of Artificial Intelligence Research*, *2*, 369–409.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society. Series B*, *68*(1095), 49–67. doi:10.1111/j.1467-9868.2005.00532.x.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *67*(2), 301–320. doi:10.1111/j.1467-9868.2005.00503.x.