Apprentissage séquentiel pour la classification de données complexes

Ludovic Denoyer - LIP6
Gabriel Arnold Dulac
Francis Maes

June 11, 2014

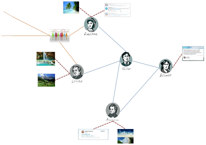
Context

State-of-the-art

- ML models are very good for some generic problems
    - Text Classification, Object Recognition, Speech Recognition

Big Data

"Big Data" is not a quantiative problem that concerns power/parallelization, but mainly a qualitative problem: it changes **the nature** and **the way we access information** and encourages the development of new ML paradigms.



Context

Usual assumptions

- When computing $f_\theta(x)$, we consider that x is known
- x has a known topology
- x has a pre-computed representation (i.e $x \in \mathbb{R}^n$)
- When learning, we consider a known (acquired) training set.

Theses assumptions are not appropriated for dealing with actual systems $\Rightarrow$

New Needs

- Learning to deal with heterogeneous/complex data
- **Learning to acquire data**
- **Learning to deal with operationnal constraints**
- Decentralized Learning
- Never-Ending Learning
- ...

Context

Datum

A datum $\mathbf{x} \in \mathbb{R}^n$ is made up of features $x_{i \in [0,n]}$:

$$\mathbf{x} = (x_1, \cdots, x_n)$$

Empirical Risk Minimization

$$\text{Find } f_{\theta^*}(\mathbf{x}) = y \text{ such that}$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}}[L(\theta)] = \underset{\theta}{\operatorname{argmin}}[\frac{1}{N}\sum_{i=1}^{N} \Delta(f_\theta(\mathbf{x_i}), y_i)]$$

Remarks

- Classification procedure is *atomic* and same for each datum.
- What matters is the result, but not *how* it was obtained.

Drawbacks of Atomic Classifiers

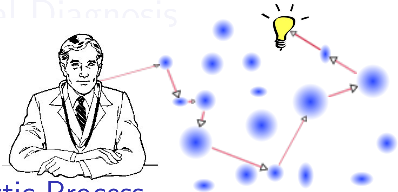
Feature Acquisition

- Entire datum must be available upfront.
- Cannot adapt feature choice to each datum.

Costly Features

- Costs associated to features cannot be taken into account.
- Any trade-offs between cost and accuracy occur on a dataset level.

Example: Medical Diagnosis



General Diagnostic Process

- Ask general questions to get context.
- Ask increasingly specific questions based on previous answers.
- During entire process, gauge cost-benefit trade-off for expensive or risky diagnostic procedures.

Two aspects:

- Adaptive information querying.
- Cost-benefit trade-offs.

This is a general procedure for domain experts when analyzing a problem.

Outline

Sequential Learning Models

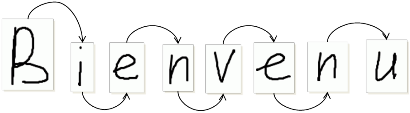
- Models that learn to acquire information **during inference**
  - and that can deal with complex topologies
- Models able to handle operationnal constraints **during inference**

Complex Data as a Graph of Content

Complex Data

A complex datum can be represented as a graph of content nodes

A sequence...

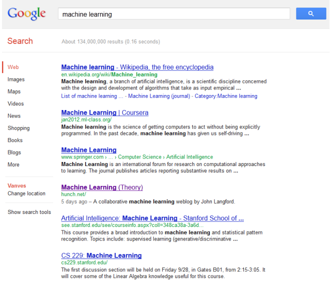


...as a multi-relationnal graph

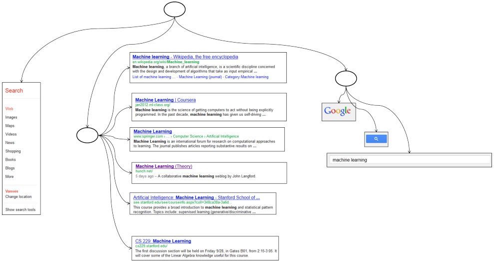
Complex Data as a Graph of Content

Complex Data

A complex datum can be represented as a graph of content nodes



A Structured document...



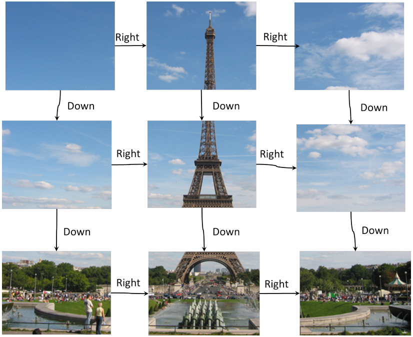
...as a multi-relationnal graph

Complex Data as a Graph of Content

Complex Data

A complex datum can be represented as a graph of content nodes

An Image...

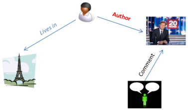


...as a multi-relationnal graph

Complex Data as a Graph of Content

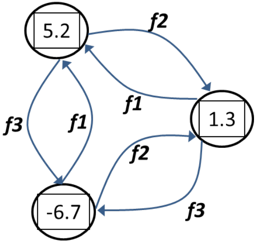
Complex Data

A complex datum can be represented as a graph of content nodes



A social network... ...as a multi-relational graph

$$\begin{pmatrix} 5.2 \\ 1.3 \\ -6.7 \end{pmatrix} \Longrightarrow$$



A vector as a multi-relational graph

A Sequential Approach to Classification

Sequential Classification

- Classification is modeled as a *sequential* process.
- Learning considers *how* the information is acquired.

Advantages of Sequential Classification

- Previously acquired information can guide further queries.
- The existence of a classification *process* allows it to be constrained to make cost/accuracy trade-offs.

We use the term *datum-wise* classifier, as each datum is classified differently.

Similar Approaches

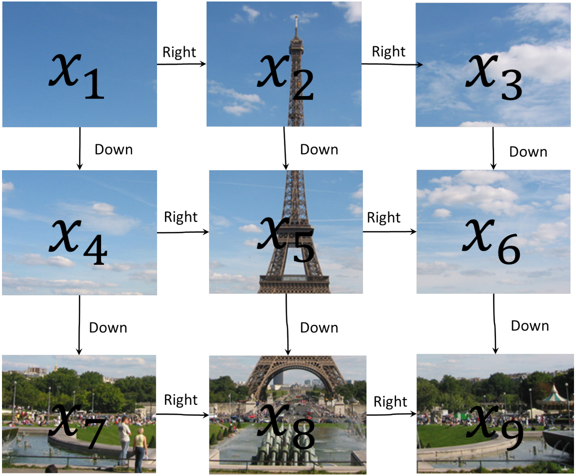
Adaptive Feature Order

- Reinforcement learning approaches.
- Decision trees.

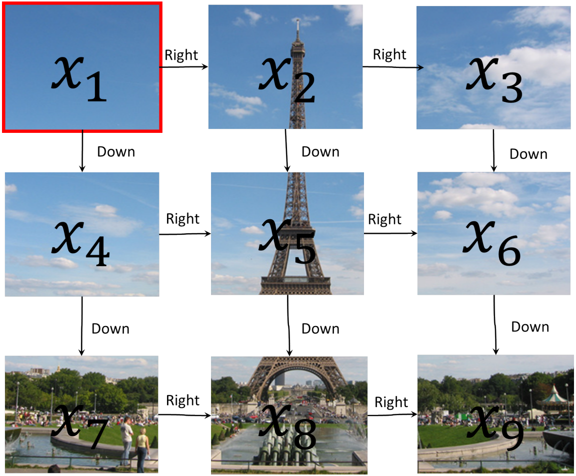
Fixed Feature Order

- Cascade classifiers.
- Early-stopping algorithms.

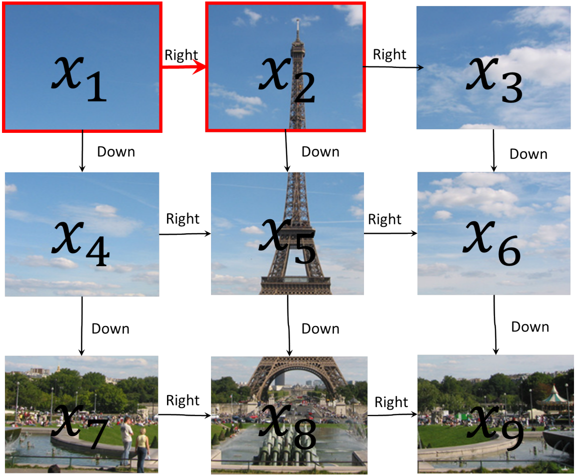
Sequential Models for Classification



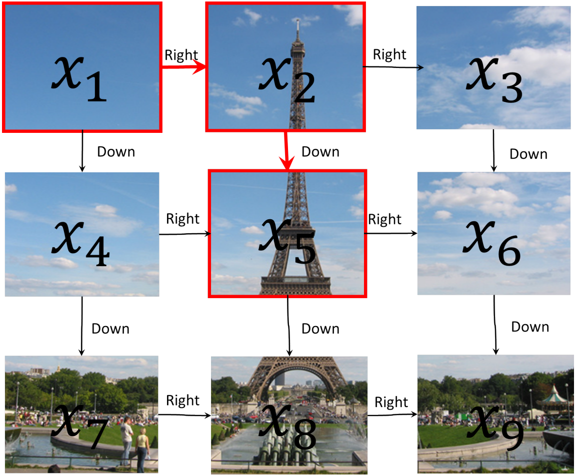
Sequential Models for Classification



Sequential Models for Classification

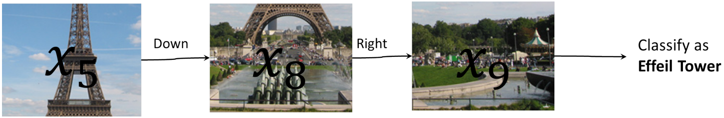


Sequential Models for Classification



Eiffel Tower

Illustration



Sequential Process for Classification

- an initial block $x_{(0)}$ is sampled
- the classifier sequentially chooses a relation $r_{(t)}$ and acquires the information $x_{(t+1)}$
- At last, the classifier chooses which category to assign to x

Classification as a Sequential Process

Two conceptual elements to the sequential classification process:

- Look at information (select features).
  - Look at a first element of information.
  - Given this new information and previously acquired information...
  - ...choose a new element of information.
- Once enough information considered, emit a decision (classify).

These two aspects can be learned jointly as a Markov decision process!

Benefits

Benefits

- It can classify **any type** of block data.
- It can classify data are are **only partially known** (e.g. streams).
- Acquired part can be processed **on the fly**
  - speed-up of the process when processing is expensive.
  - able to learn/predict with data that are not fully known
- The process is able to focus on relevant parts avoiding **noise** and **misleading** information.

Drawback

- Learning is not really fast.... (but can be parallelized)

Markov Decision Process

Classification as an MDP

Let an MDP $\mathcal{M}$ be defined by a 4-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$.

- $\mathcal{S}$ is the set of states, representing all information acquired thus far for a specific datum.
- $\mathcal{A}$ is the set of possible actions, either information acquisition actions $a \in \mathcal{A}_y$ or classification actions $a \in \mathcal{A}_f$.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the MDP's *transition function*, returning the requested information.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the *reward function*, returning the reward for taking action a in state s and ending up in state s'.
  - Responsible for defining the agent's ultimate goal for the task.

Classifier as a Policy

Policy

Decisions in an MDP are taken by a *policy* π_θ.

- The classifier *is* a policy: the final output of π_θ is our class label.
- The policy is a parameterized function $\pi_\theta(s) = a$.
- The goal of the policy π_θ is to maximize the overall reward:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T_\theta(\mathbf{x}_i)} r(\mathbf{x}_i, \pi_\theta^t(\mathbf{x}_i)).$$

- Defining a proper reward function is key.

Reward & 0/1 Loss

Learning Goal

$$\theta^* = \operatorname*{argmin}_\theta \frac{1}{N}\sum_{i=1}^{N} \Delta_{0-1}(f_\theta(\mathbf{x_i}), y_i) \qquad \text{minimize loss}$$

$$\Leftrightarrow$$

$$\theta^* = \operatorname*{argmax}_\theta \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T_\theta(s_i)+1} r(s_i, \pi_\theta^t(s_i)) \qquad \text{maximize reward}$$

Reward Definition

Reward designed to correspond to a 0/1 classification loss:

$$r(s,a) = \begin{cases} -1 \text{ if } a \in \mathcal{A}_y \wedge a \neq y_i \\ 0 \text{ otherwise} \end{cases}$$

Policy

Defining the Policy

The policy is a linear function approximator parameterized by θ, π_θ:

- $\pi_\theta(s) = \operatorname{argmax}_{a \in \mathcal{A}} \theta^\intercal \Phi(s, a)$
- $\Phi(s, a)$ represents acquired features, their values, and the given action.
- $\mathbf{z}$ is and indicator vector of selected features.

Masked representation:

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

State representation:

$$\phi(s) = (\mu(\mathbf{x}, \mathbf{z}), \mathbf{z})$$

State-Action tuple:

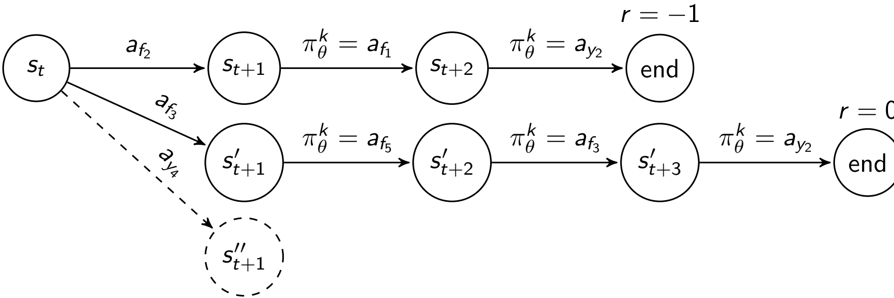
$$\Phi(s, a) = (0, \cdots, \phi(s)_a, \cdots, 0)$$

Training

Rollouts Classification Policy Iteration

Monte-Carlo policy iteration:

- Sample a series of random states: random datum, random set of features.
- Estimate best action a^* for state s_t by sampling:



- Train parameterized policy with best action for $\Phi(s_t, a^*)$.

RCPI Training

Algorithm 1 RCPI

```
procedure TRAIN–RCPI(S_R, M, π_0, K)
    π = π_0
    repeat
        S_T = ∅
        for s ∈ S_R do
            for a ∈ A do
                Q̃_π(s, a) ← Rollout(M, s, a, K, π)
            A* = argmax_{a∈A} Q̃_π(s, a)
            S_T ← S_T ∪ {(s, a*) ∀a* ∈ A*}
        f_θ = Train(S_T)                              ▷ f_θ is a multiclass classifier
        π' from f_θ as defined in Eq. (??)
        π_t = α(π', π_{t−1})
    until π_t ∼ π_{t−1}
    return π_t
```

RCPI Complexity

- Inference Complexity is $\mathcal{I}(A)$ at each step
  - $\mathcal{I}(A)$ is the cost of choosing the action to do
  - $\mathcal{I}(A) = |\mathcal{A}|$ when using One Against All classifiers.
- **Learning Complexity**: $SAK \times T\mathcal{I}(A) + \mathcal{C}(S, A)$.
  - $T\mathcal{I}(A)$ is the cost of sampling one trajectory of size T
  - SAK is the number trajectories necessary during simulation.
  - $\mathcal{C}(S, A)$ is the cost of learning the corresponding classifier.

When using OVA classifiers, RCPI learning complexity is $\mathcal{O}(\mathcal{A}^2)$!.

Later

This complexity can be reduced to $\mathcal{O}(log(A))$

Section Summary

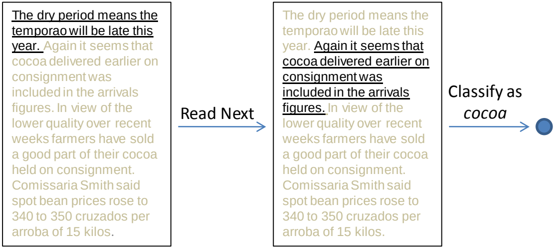
Sequential Classification & Reinforcement Learning

Two concepts introduced:

- Classification as a sequential process.
- Markov decision process & reinforcement learning to find a good policy.

Now, some applications.

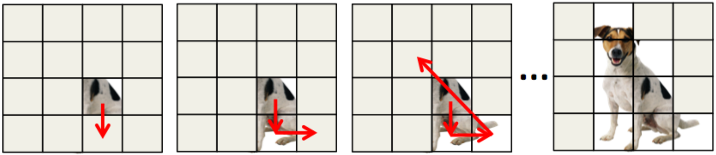
Sequential Classification Tasks



Sentence-Based Text Classification

- Each document is read sentence-by-sentence, with the classifier allowed to classify at any point.
- A Bag-of-Words representation is used for each sentence.

Sequential Classification Tasks



Region-Based Image Classification

- Each image is acquired region-by-region, with the classifier deciding which region to acquire next or which class to classify into.
- Each region is represented by its local SIFT features.

MDP Elements

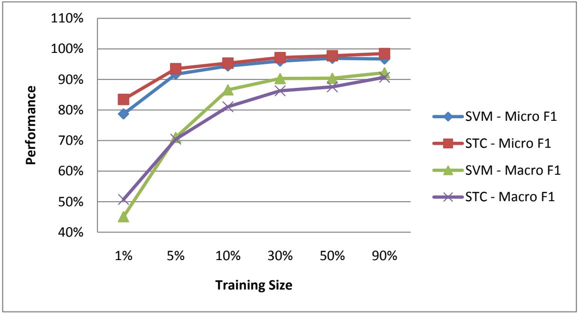
- States composed of images or text information acquired thus far.
- Actions allow for reading next sentence or acquiring particular region.

Experiments

Experimental Protocol

- Standard datasets such as Reuters 8-class for text (3 actions) or PPMI (18 actions) for image.
- Datasets split into train & test (varrying splits).
- Classification policy trained on training set.
- Performance calculated on test set with final learned policy.

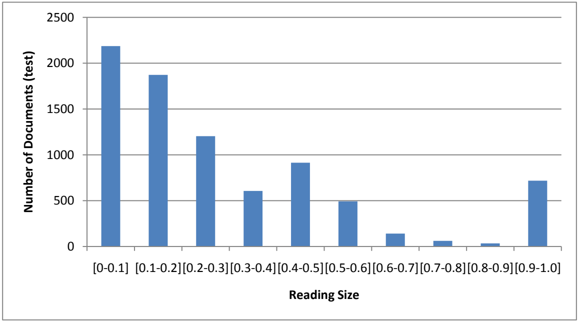
Sequential Text Classification: Experimental Results



Reuters-8 Performance

Performance is comparable to state-of-the-art approaches (BoW SVM).

Sequential Text Classification: Behavior



Reuters-8 Behavior

- Number of sentences used for classifying each document.
- We see that most documents are barely read.

Sequentiual Image Classification: Performance

Instrument	Sequential Image	SVM 16
Bassoon	0.77 @ 0.04	0.77
Cello	0.76 @ 0.03	0.75
Clarinet	0.69 @ 0.02	0.69
Erhu	0.82 @ 0.08	0.80
Flute	0.90 @ 0.05	0.89
French Horn	0.85 @ 0.03	0.84
Guitar	0.85 @ 0.03	0.85
Harp	0.81 @ 0.06	0.81
Recorder	0.69 @ 0.03	0.68
Saxophone	0.80 @ 0.06	0.79
Trumpet	0.72 @ 0.05	0.68
Violin	0.86 @ 0.06	0.85
Average	**0.8** @ 0.08	0.78

Sequential Image Classification on PPMI

Performance on the PPMI dataset is comparable to baseline SVM, but sparsity is not appearing natively (number after the @)...

A Lack of Constraints

Conclusions:

- Sequential text classification offers good performance and naturally uses very little information.
- Sequential image classification also performs well ... but does not constrain its use of features.

Solutions

- Find a way to encourage the sequential classifier to be efficient with its information usage.

Presentation Structure

Outline

Three main parts to this presentation:

1. Introduce sequential classification and some example tasks.
2. Constrain the classification task to encourage sparsity.
3. Show that our sparse model can easily be extended to many cost-sensitive tasks.

Sparsity and Classification

An existing approach to encouraging efficiency is to consider a sparsity constraint:

Regularized Empirical Loss

$$R(\theta) = \frac{1}{N}\sum_{i=1}^{N}\Delta(f_\theta(\mathbf{x}_i), y_i) + \underbrace{\lambda||\theta||_0}_{L_0 \text{ regularization term}}$$

- The L_0 norm penalizes feature usage by the model on a *dataset* level.
- For a sequential, *datum-wise* classifier, we need a *datum-wise* penalization.

Datum-Wise Sparsity

Sequential Classifier Definition

Let $y^{\mathbf{x}_i}$ be a class label, $\mathbf{z}^{\mathbf{x}_i}$ be an indicator vector of selected features.

$$f_\theta(\mathbf{x}_i) = (y_\theta^{\mathbf{x}_i}, \mathbf{z}_\theta^{\mathbf{x}_i})$$

Datum-Wise Loss

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta(y_\theta^{\mathbf{x}_i}, y_i) + \underbrace{\lambda \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{z}_\theta^{\mathbf{x}_i}\|_0}_{\text{datum-wise regularization term}}$$

Datum-Wise Sparse Model

Corresponding MDP

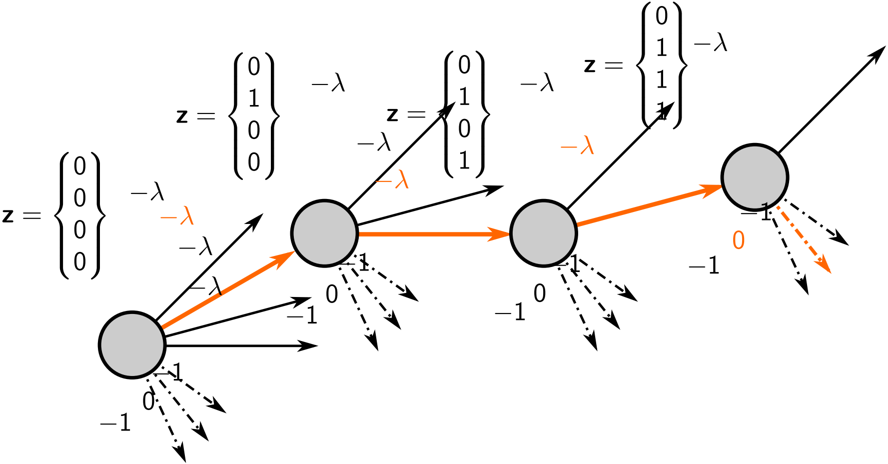
As mentioned previously, the reward function must be equivalent to the defined loss.

Reward

Reward must correspond to a 0/1 classification loss with a datum-wise regularization penalty:

$$r(s,a) = \begin{cases} -\lambda \text{ if } a \in \mathcal{A}_f \text{ (instead of 0)} \\ -1 \text{ if } a \in \mathcal{A}_y \wedge a \neq y_i \\ 0 \text{ otherwise} \end{cases}$$

Datum-Wise Sparse Classification (contd.)



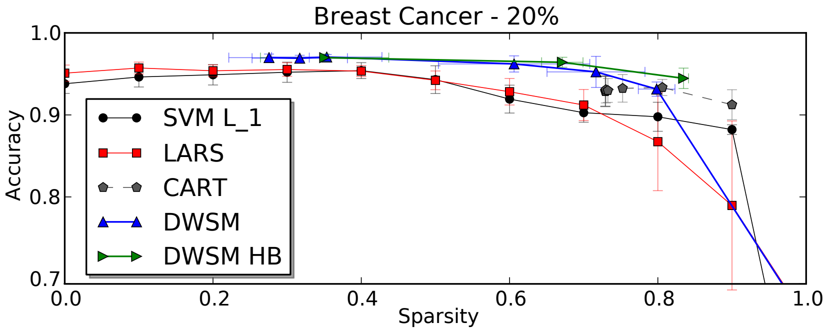
In this illustrated example, the final reward is -3λ.

Experiments

Experimental Protocol

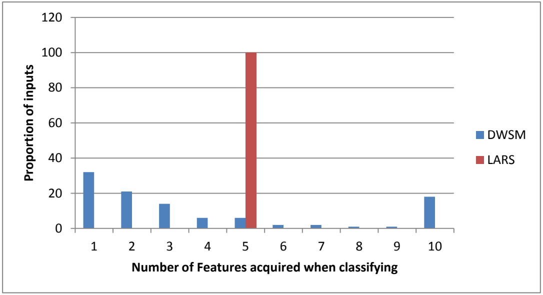
- Experiments are run on 8 single-class and 4 multi-class vectorial datasets, with varying train/test splits.
- Datasets have between 6 to 60 features, and 200-1000 elements.
- For each classifier, the sparsity parameter is varied from one extreme (no features selected) to the other (all features selected).
- We compare DWSM's performance to an L_1 regularized linear SVM and LARS as well as a C4.5 decision tree (CART).

Datum-Wise Sparsity: Experiments



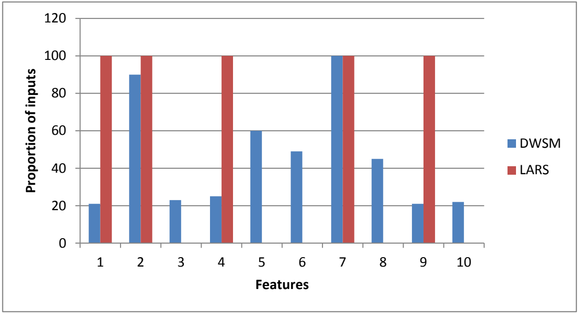
- This accuracy vs. sparsity plot is typical of results with DWSM.
- DWSM performs is able to keep accuracy high even under high levels of sparsity.

Experimental Behavior



This graphic shows the average number of features used when classifying data on the Breast Cancer dataset. Feature usage is constant for LARS but varies for DWSM.

Experimental Behavior



This graphic shows how much each feature was used to classify data on the Breast Cancer dataset. We see that some features chosen by LARS are also used heavily by DWSM.

A wider field of application...

Conclusions

- DWSM is able to compete with state-of-the-art sparsity constraints.
- DWSM is able to find a good policy for the complex datum-wise L_0 loss.
- The learning algorithm does not require a uniform cost.
- ... which can allow us to consider more complex constrained classification problems.

Presentation Structure

Outline

Three main parts to this presentation:

1. Introduce sequential classification and some example tasks.
2. Constrain the classification task to encourage sparsity.
3. Show that our sparse model can easily be extended to many cost-sensitive tasks.

Budgeted classifiers

Budgeted classifiers

Standard classifier:
$f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$

Budgeted classifier:
$f_\theta : \mathcal{X} \rightarrow \mathcal{Y} \times \mathcal{Z}$

- $\mathcal{Z}$ corresponds to all the ways the classifier can classify.
- $y_\theta(x)$ is the prediction, $z_\theta(x)$ is the classification process.

Advantage

We can define a risk with a datum process loss:

$$\mathcal{R}(\theta) = \int_{x,y} (\Delta(y_\theta(x), y) + C(z_\theta(x))P(x, y)dx.dy$$

Budgeted classifiers

Empirical Risk minimization

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{\ell} \sum_{i=1}^{\ell} (\Delta(y_\theta(x^i), y^i) + C(z_\theta(x^i))$$

- C is a loss (or constraint) over the way inputs are processed
- C is defined at the datum level
- **Note that:** if C only depends on θ, it corresponds to the classical regularization term of usual classifiers

$C =$

Price of the acquisition of a learning example
Time spent to classify
Interpretability of the resulting model
...

Budgeted Classifier

This generic loss can be optimized through Sequential Learning methods (RL) by modifying the shape of the previously presented MDP.

- **No constraint**: The classifier can stop when it wants
- **Sparse Classification**: minimization of acquisition
- **Cost-Sensitive Classification**: Each feature/block has a cost
- **Budgeted Classification**: Limited or Maximum Budget
- **Relationnal Features**: The cost of acquiring a part depends on the previouly acquired parts
- **Structured Output**: Outputs are structures

Datum-Wise Constrained Losses

An example datum-wise risk for cost-sensitive classification:

Cost-Sensitive Datum-Wise Empirical Risk

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta_{cost}(y_\theta^{\mathbf{x}_i}, y_i) + \frac{1}{N} \sum_{i=1}^{N} \langle \xi, \mathbf{z}_\theta^{\mathbf{x}_i} \rangle$$

Example

$$\begin{aligned} \xi &= (8, 1, 123, 1, 1, 40) \\ \mathbf{z}_\theta^{\mathbf{x}_i} &= (1, 0, 0, 1, 0, 1) \\ \text{Cost} &= \langle \xi, \mathbf{z}_\theta^{\mathbf{x}_i} \rangle = 49 \end{aligned}$$

...and their corresponding MDP definitions.

Cost-Sensitive Reward Function

Once again we define a reward function that is equivalent to the loss:

$$r(s, a_i) = \begin{cases} -\xi_i \text{ if } a_i \in \mathcal{A}_f \text{ (instead of } -\lambda) \\ -C_{a_i,y} \text{ if } a_i \in \mathcal{A}_y \text{ (instead of -1)} \\ 0 \text{ otherwise} \end{cases}$$

Task	Regularized Empirical Loss
Hard Budget	$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta(y_\theta^{\mathbf{x}_i}, y_i) + \lambda \frac{1}{N} \sum_{i=1}^{N} \lVert \mathbf{z}_\theta^{\mathbf{x}_i} \rVert_0$ subject to $\lVert z_\theta^{\mathbf{x}_i} \rVert_0 \leq M.$
Cost-Sensitive	$\theta^* = \operatorname{argmin}_\theta \frac{1}{N} \sum_{i=1}^{N} \Delta_{cost}(y_\theta^{\mathbf{x}_i}, y_i) + \frac{1}{N} \sum_{i=1}^{N} \langle \xi, \mathbf{z}_\theta^{\mathbf{x}_i} \rangle$
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	$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta(y_\theta^{\mathbf{x}_i}), y_i) + \frac{1}{N} \sum_{i=1}^{N} \sum_{f, f' \in \mathcal{Z}_\theta^{\mathbf{x}_i}} \text{Related}(f, f')(\lambda - \gamma) + \gamma$

Proposed tasks and corresponding learning problems.

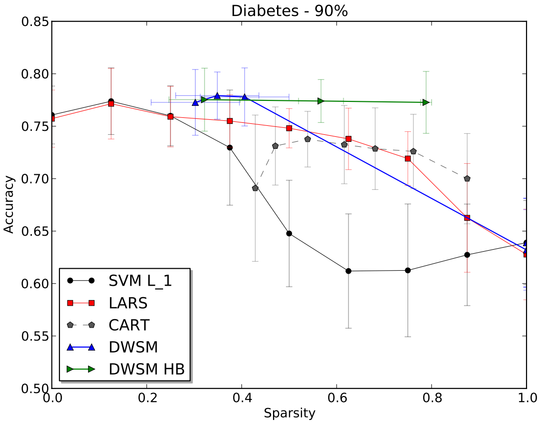
Task	Decision Process Modification	Commentary
Hard Budget	$\mathcal{A}(s) = \begin{cases} \mathcal{A}_f(s) \bigcup \mathcal{A}_y(s) \text{ if } \lVert\mathbf{z}\rVert_0 < M \\ \mathcal{A}_y(s) \text{ if } \lVert\mathbf{z}\rVert_0 = M \end{cases}$	Allows users to choose minimum level of spars Reduces training complex
Cost-Sensitive	$r(s_i, a) = \begin{cases} -\xi_i \text{ if } a \in \mathcal{A}_f \\ -C_{a,y_i} \text{ if } a \in \mathcal{A}_y \end{cases}$	Well-suited for features w variable costs.
Grouped Features	$\mathcal{A}_f = \mathcal{A}_{group}$ $\mathcal{T}(s, a_j) = (\mathbf{x}, \mathbf{z} + \sum_{i \in \mathcal{F}_j} \mathbf{e_i})$	Well adapted to featu presenting a grouped natu Complexity is reduced.
Relational Features	$r(s_i, a_j) = \begin{cases} -\lambda \text{ if } (\ \forall f \in \mathcal{Z}(\mathbf{x}), \\ \qquad \text{Related}(f_j, f) = 1\ ) \\ -\gamma \text{ otherwise} \end{cases}$	Naturally suited for comp feature inter-dependencie

Medical Diagnosis: Results

Classifier	Error Penalty	Average Cost	Accuracy
DWSM	800	181	0.75
DWSM	400	74	0.76
Li & Carin	800	180	0.75
Li & Carin	400	75	0.75

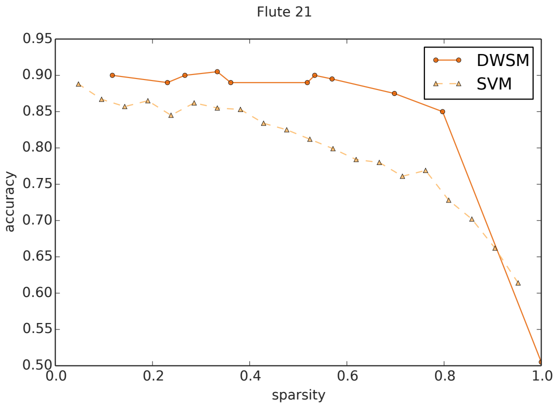
The modified DWSM MDP is able to compete with other state-of-the-art cost-sensitive classifiers on the Pima Diabetes dataset.

Maximum Cost



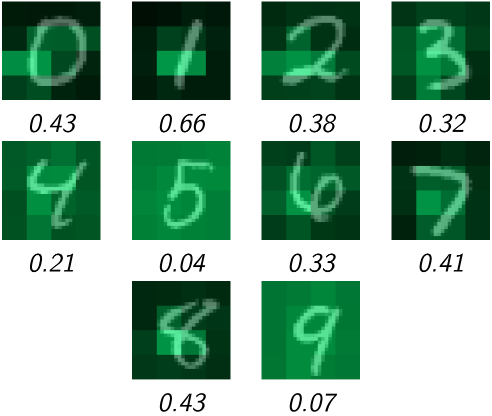
A hard limit on features makes attaining specific levels of sparsity easier.

Grouped Features for Image Classification



With a penalty for each region, DWSM is able to maintain accuracy with much less information consumption relative to a baseline method.

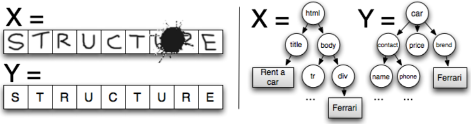
Regionally-Constrained Image Regions

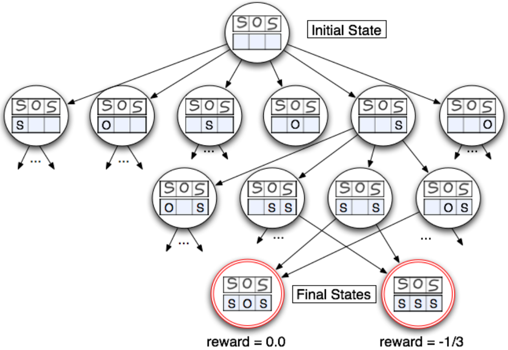


Most utilized regions on the regionally-constrained MNIST task.

Structured Output Classification

Different tasks and Complex Transformations scheme:





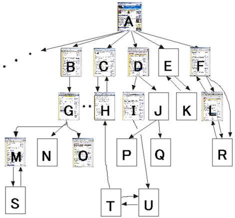
Future Workd

- Online Budgeted Learning
  - Budgeted constraints **during learning**
  - Use of bandit-based methods or transfer learning techniques
- Classification under Realistic Budgets
  - Time (in seconds)
  - Electric Power Consumption
  - ...
- Interaction in ML

Learning on the Web

Sequential Search Engines

- Find a relevant document for a particular query in 1 minute
- Find me a good hotel in Orsay in less than 3 days
- Organize my trip to Barcelona



Sequential Recommender Systems

Able to ask questions to a new user (cold start)

Learning on the real world

Robots

- Tell me where you are in 5 seconds
- Find a target in 3 seconds

Initial state

Movement computed by the robot to acquire relevant information

Final state

Visual Reinforcement Learning

Decide based on cameras

Sensors Networks

Choose which sensor to ask for collecting information

Learning on the Web and the real world

Web Social Robots

- Learning to talk
- *fill my fridge*

Not clear difference between robots and ML models.....

RCPI Complexity

- Inference Complexity is $\mathcal{I}(A)$ at each step
  - $\mathcal{I}(A)$ is the cost of choosing the action to do
  - $\mathcal{I}(A) = |\mathcal{A}|$ when using One Against All classifiers.
- **Learning Complexity**: $SAK \times T\mathcal{I}(A) + \mathcal{C}(S, A)$.
  - $T\mathcal{I}(A)$ is the cost of sampling one trajectory of size T
  - SAK is the number trajectories necessary during simulation.
  - $\mathcal{C}(S, A)$ is the cost of learning the corresponding classifier.

When using OVA classifiers, RCPI learning complexity is $\mathcal{O}(\mathcal{A}^2)$!.

Error-Correcting Output Code (ECOC)

Error-Correcting Output Codes (ECOCs) have been used for multiclass classification for a while (Dietterich and Bakiri (1995)). Each of the 5 labels is associated to a binary code of length 3.

	b_1	b_2	b_3
y_1	$+$	$+$	$-$
y_2	$-$	$+$	$-$
y_3	$+$	$-$	$+$
y_4	$-$	$+$	$+$
y_5	$+$	$-$	$-$

- Minimum code length: $C = \log_2(|\mathcal{Y}|)$.
- In general: $C = \gamma \log(|\mathcal{Y}|)$ with $\gamma \approx 15$.

ECOC Inference

$$\left.\begin{array}{l} f_1(x) = + \\ f_2(x) = - \\ f_3(x) = + \end{array}\right\} \longrightarrow \text{`}+-+\text{'}$$

	b_1	b_2	b_3	$dist_H(+-+)$
y_1	$+$	$+$	$-$	2
y_2	$-$	$+$	$-$	3
y_3	$+$	$-$	$+$	0
y_4	$-$	$+$	$+$	2

$\boxed{f(x) = y_3}$

- $f(x) = \text{argmin}_{y \in \mathcal{Y}}\, d_{\text{Hamming}}(y, (f_1(x), f_2(x), f_3(x)))$
- Only $\mathcal{O}(\log(n))$ classifiers.

ECOC–Training

Question

How to learn the $\gamma \log(n)$ classifiers $f_{i \in [1,C]}$?

- Original labels are mapped to C binary label spaces:

$$y(x) = y_3$$
$$c(y(x)) = (+, -, +)$$
$$y^1(x) = +$$

- One binary classifier learned for each space

$$Y^1 = \{y^1(x) \ \forall x \in X\}$$
$$f_1 = Train(X, Y^1)$$

ECOC for MDPs

ECOC w/ RCPI

RCPI allows for the use of any multiclass classifier to define a policy.

In this case we use an ECOC classifier, where each action is given a binary label of length $C = \gamma \log(|\mathcal{A}|)$.

	b_1	b_2	b_3
a_1	$+$	$+$	$-$
a_2	$-$	$+$	$-$
a_3	$+$	$-$	$+$
a_4	$-$	$+$	$+$
a_5	$+$	$-$	$-$

General Idea (cont.)

We effectively define C new sub-policies, each one associated to a particular binary mapping of the MDP.

$$\begin{aligned}
&\pi_i : s \to \{+,-\} \\
&\pi_i(s) = f_i(s) \\
&\pi(s) = \operatorname*{argmin}_{a \in \mathcal{A}} d_{\text{Hamming}}(\mathbf{M}^c_{[a,*]}, (\pi_1(s), \cdots, \pi_C(s)).
\end{aligned}$$

ERCPI

- The sub-policies are learned conjointly using the RCPI algorithm.
- Training is performed as for a standard ECOC multiclass classifier.

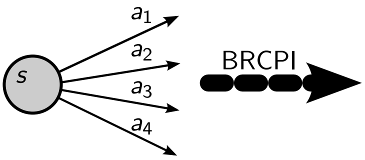
BRCPI

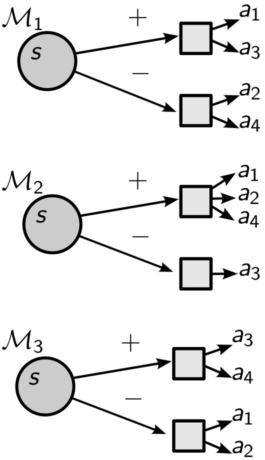
- ERCPI still requires the full policy π_{t-1} to be available for simulation, and must still run a simulation for every action in every state.
- ERCPI Learning complexity is $\mathcal{O}(\mathcal{A} \log \mathcal{A})$

Second Idea

To reduce the complexity of this algorithm, we propose learning the C binary sub-policies — $\pi_{i \in [1,C]}$ — **independently**, transforming our initial MDP into C sub-MDPs, each one corresponding to the environment in which a particular π_i is acting.

BRCPI Example





	b_1	b_2	b_3
a_1	$+$	$+$	$-$
a_2	$-$	$+$	$-$
a_3	$+$	$-$	$+$
a_4	$-$	$+$	$+$

BRCPI (cont.)

We can now define C new binary MDPs that we name sub-MDPs, and denote $\mathcal{M}_{i\in[1,C]}$.

- $\mathcal{S}_i = \mathcal{S}$, the same state-set as the original MDP.
- $\mathcal{A}_i$ is the binary action set: $\{+,-\}$.
- $\mathcal{T}_i = \mathcal{T}(s', s, a)P(a|a_i) = P(s'|s, a)P(a|a_i)$.
  - $P(a|a_i)$ is the probability of choosing action $a \in \mathcal{A}^{a_i}$, knowing that the sub-action applied on the sub-MDP $\mathcal{M}_i$ is $a_i \in \{+,-\}$.
  - We consider $P(a|+)$ to be uniform for $a \in \mathcal{A}^+$ and null for $a \in \mathcal{A}^-$, and vice versa. $P(s'|s, a)$ is the original MDP's transition probability.
- $r_i(s, a_i) = \sum\limits_{a\in\mathcal{A}_i^{a_i}} P(a|a_i)r(s, a)$.

BRCPI - Algorithm

Algorithm 2 BRCPI

procedure TRAIN–BRCPI($\mathcal{S}_R$, $\mathcal{M}$, π_0, K, T)
    for $i \in C$ **do**
        $\pi_i = \pi_0$
        repeat
            $\mathcal{S}_{\mathrm{T}} = \emptyset$
            for $s \in \mathcal{S}_R$ **do**
                for $a \in \{+, -\}$ **do**
                    $\tilde{Q}_\pi(s, a) \leftarrow$ **Rollout**($\mathcal{M}_i$, s, a, K, π_i)
                $\mathcal{A}^* = \mathrm{argmax}_{a \in \mathcal{A}} \tilde{Q}_\pi(s, a)$
                for $a^* \in \mathcal{A}^*$ **do**
                    $\mathcal{S}_{\mathrm{T}} = \mathcal{S}_{\mathrm{T}} \cup \{(s, a^*)\}$
            $f_{\theta_i} =$ **Train**($\mathcal{S}_{\mathrm{T}}$)
            π_i' from f_{θ_i} as defined in Eq. (**??**)
            $\pi_i = \alpha(\pi_i, \pi_i')$
        until $\pi_i \sim \pi_i'$
    return $\pi(s) = \mathrm{argmin}_{a \in \mathcal{A}}\, d_{\mathrm{Hamming}}(\mathbf{M}^C_{[a,*]}, (\pi_1(s), \cdots, \pi_C(s)).$

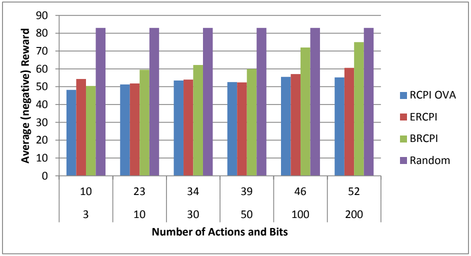
Complexities

Complexities

Algorithm	Simulation Cost	Learning Cost
ORCPI	$\mathcal{O}(A^2)$	$\mathcal{O}(A)$
ERCPI	$\mathcal{O}(A\log(A))$	$\mathcal{O}(\log(A))$
BRCPI	$\mathcal{O}(\log(A))$	$\mathcal{O}(\log(A))$

Mountain Car

- Mountain Car problem
- Actions (between -1 and 1) have been discretized



Smaller is Better

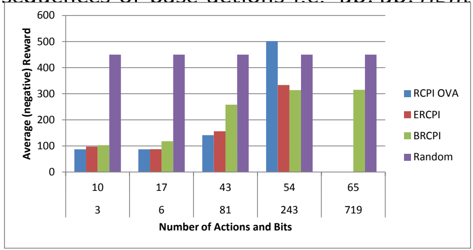
Results

ERCPI gives similar results to ORCPI

BRCPI allows one to obtain a non-trivial sub-optimal policy

Maze

- Negative reward associated to some cells
- Base actions are *up*,*down*, and *right*
- We define sequences of base actions i.e. *up, up, right*



Results

ERCPI outperforms ORCPI when the number of actions is large
BRCPI allows one to obtain a non-trivial sub-optimal policy **when other methods fail**

Speedup

Tradeoff between training speed and policy quality.

Mountain Car - 100 Actions - 46 bits				
	Sim.	Learning	Total	Speedup
OVA	4312	380	4698	$\times 1.0$
ERCPI	3188	190	3378	$\times 1.4 (\times 1.35)$
BRCPI	184	190	374	$\times 12.5 (\times 23.5)$

Questions

- Sequential approaches for learning datum-wise sparse representations – G Dulac-Arnold, L Denoyer, P Preux, P Gallinari *in Machine learning 2012*
- Structured prediction with reinforcement learning – F Maes, L Denoyer, P Gallinari, *in Machine learning 2009*