Datum-Wise Classification:
A Sequential Approach to Sparsity

Gabriel Dulac-Arnold[1], **Ludovic Denoyer**[1], Philippe Preux[2], Patrick Gallinari[1]

LIP6 - University Pierre et Marie Curie - Paris

LIFL & INRIA

September 7, 2011

SORBONNE UNIVERSITÉS

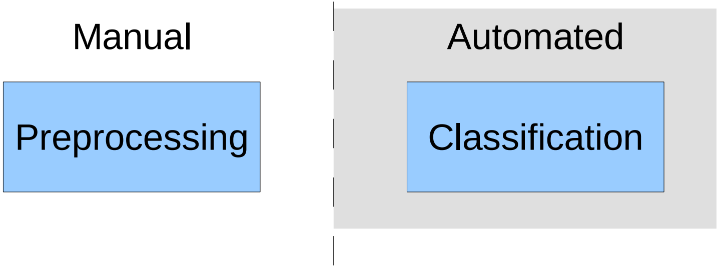
Outline

- Motivation
- Classical Features Selection Methods
- Datum-wise classifiers
- Sparsity as a Sequential Process
- Learning
- Experiments
- Conclusion and Perspectives

Motivation

General Motivation

Is it possible to include the classical **preprocessing** step into the learning process (for classification) ?



Motivation

General Motivation

Is it possible to include the classical **preprocessing** step into the learning process (for classification) ?

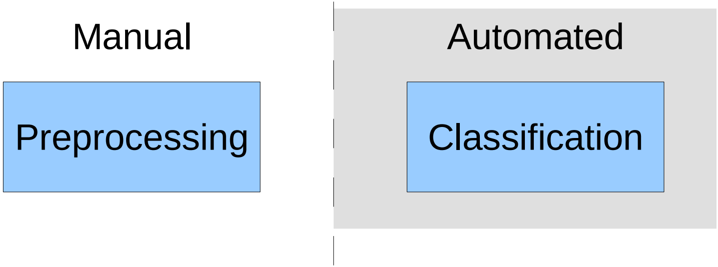
Applications:

- Text: Building dictionary, mapping documents to vectors.
- Image: applying image transformation operators, building visual dictionary,...
- Numerical Data : Features selection, Features acquisition, features construction, etc...

Motivation

General Motivation

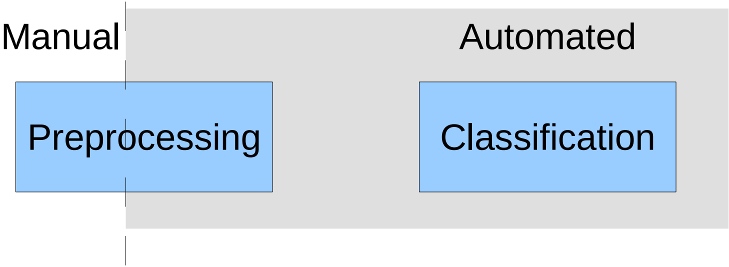
Is it possible to include the classical **preprocessing** step into the learning process (for classification) ?



Motivation

General Motivation

Is it possible to include the classical **preprocessing** step into the learning process (for classification) ?



Motivation

General Motivation

Is it possible to include the classical **preprocessing** step into the learning process (for classification) ?

Proposed Solution

- Consider the whole process as a sequential process:
  - Start with some preprocessing steps...
  - ...then apply a classification step
- Use Sequential Learning Methods (Reinforcement Learning,...)

Here: we focus on the problem of selecting as few features as possible for classification (Sparse classification)

Main Approaches to Sparsity

Three main types of approaches to Sparsity/Features Selection for classification:

- **Wrapper Approaches :** Exhaustive Search of Features-Space
- **Filter Approaches :** Independent Ranking of Features
- **Embedded Approaches :** Minimization of a regularized loss function

Main Approaches to Sparsity

Three main types of approaches to Sparsity/Features Selection for classification:

- **Wrapper Approaches :** Exhaustive Search of Features-Space
- **Filter Approaches :** Independent Ranking of Features
- **Embedded Approaches :** Minimization of a regularized loss function

Some drawbacks

Main Approaches to Sparsity

Three main types of approaches to Sparsity/Features Selection for classification:

- **Wrapper Approaches :** Exhaustive Search of Features-Space
  - Searches are poorly directed and quickly intractable.
- **Filter Approaches :** Independent Ranking of Features
- **Embedded Approaches :** Minimization of a regularized loss function

Some drawbacks

Main Approaches to Sparsity

Three main types of approaches to Sparsity/Features Selection for classification:

- **Wrapper Approaches :** Exhaustive Search of Features-Space
  - Searches are poorly directed and quickly intractable.
- **Filter Approaches :** Independent Ranking of Features
  - Feature inter-dependencies are ignored, metrics are heuristic.
- **Embedded Approaches :** Minimization of a regularized loss function

Some drawbacks

Main Approaches to Sparsity

Three main types of approaches to Sparsity/Features Selection for classification:

- **Wrapper Approaches :** Exhaustive Search of Features-Space
  - Searches are poorly directed and quickly intractable.
- **Filter Approaches :** Independent Ranking of Features
  - Feature inter-dependencies are ignored, metrics are heuristic.
- **Embedded Approaches :** Minimization of a regularized loss function
  - Kernel choice must be made in terms of problem, feature inter-dependencies are ignored. Usually restricted to convex loss-functions.

Some drawbacks

Global Methods

Most feature-selection approaches try to find the subset of features, $\mathcal{F}_s$, that best represents the **entire dataset**.

There are two main drawbacks:

- $\mathcal{F}_s$ is the same for the entire dataset, even if different generating distributions are present.
- All of of the features in $\mathcal{F}_s$ are used for every new datapoint, even if some points are easily classified with only one or two features.

General Idea

Learn a classifier able to select the best subset of features to use for classifying each new input. The subset **depends on** the input to classify.

Loss Function

Classical L_1 regularized loss minimization problem

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta(f_\theta(\mathbf{x_i}), y_i) + \lambda |\mathbf{w}|_1. \quad (1)$$

Ideally, the L_0 norm would be used, but that makes for an non-continuous non-derivable risk function.

Loss Function

Classical L_1 regularized loss minimization problem

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta(f_\theta(\mathbf{x_i}), y_i) + \lambda |\mathbf{w}|_1. \quad (1)$$

Proposed problem

We define a new type of classifier, that provides both the label of the datum and the features considered:

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

The vector $\mathbf{z}$ is the set of features used to infer that point $\mathbf{x}$ should be labeled as y.

Loss Function

Classical L_1 regularized loss minimization problem

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta(f_\theta(\mathbf{x_i}), y_i) + \lambda |\mathbf{w}|_1. \quad (1)$$

Proposed problem

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

The obtained loss is:

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

Loss Function

Classical L_1 regularized loss minimization problem

$$\theta^* = \operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \Delta(f_\theta(\mathbf{x_i}), y_i) + \lambda |\mathbf{w}|_1. \quad (1)$$

Proposed problem

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

$\sum_{i=1}^{N} \|z_\theta(\mathbf{x_i})\|_0$ attemps to reduce the *average* number of features used over the entire dataset.

Sequential Decision Process

Minimization Problem

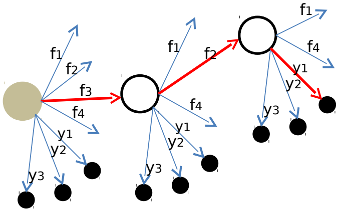
$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \Delta(y_\theta(\mathbf{x_i}), y_i) + \lambda \frac{1}{N} \sum_{i=1}^{N} \|z_\theta(\mathbf{x_i})\|_0$$

The optimization problem is a discrete optimization problem which is hard to solve:

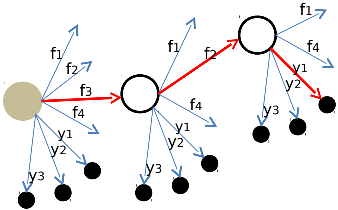
MDP

- We propose to model the classifier as a sequential decision process...
- ...the **Optimal Policy** is the solution of the loss minimization problem

Illustration



Illustration



- In a particular state (x, z), the agent is currently classifying a specific datum $\mathbf{x}$, with the features specified by $\mathbf{z}$ having been selected in the past.
- Two types of possible actions:
  - Get a new feature (in the set of unknown features)
    - New state is (x, z') where $\mathbf{z}' = \mathbf{z} + \mathbf{f_j}$.
    - Reward received is $-\lambda$
  - Classify (and stop the process
    - Reward is -1 is the chosen category is a bad one, 0 elsewhere.

)

Policy

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

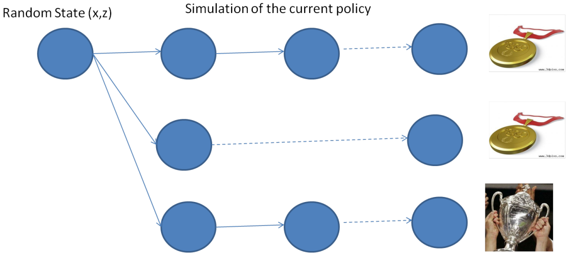
$$\pi_\theta : \mathcal{X} \times \mathcal{Z} \rightarrow \mathcal{A} \text{ and } \pi_\theta(\mathbf{x}, \mathbf{z}) = \underset{a}{\operatorname{argmax}} \langle \Phi(\mathbf{x}, \mathbf{z}, a); \theta \rangle$$

where $\Phi(\mathbf{x}, \mathbf{z}, a)$ contains information about:

- Which features have been previously acquired
- The value of these features

The optimal policy is found by using **Monte Carlo techniques** (Rollout)

Rollout



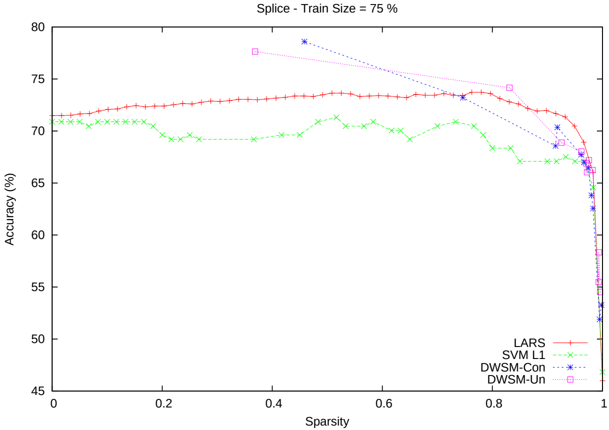
We obtain a set of learning examples $\{(\Phi(s,a), \mathit{reward})\}$ used for learning new policy (regression/classification).

The learning complexity is quite high - able to process datasets with hundred of features

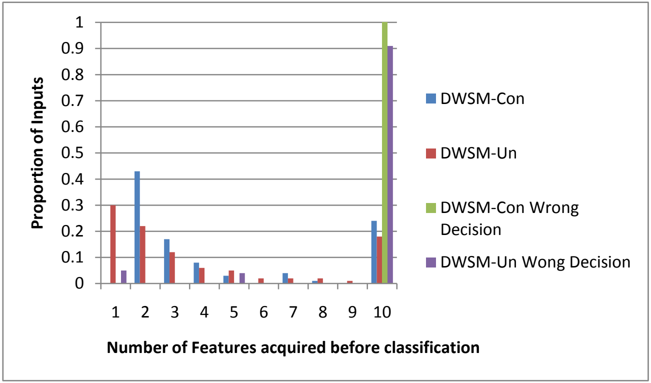
Inference is as fast as linear classification

Sparsity vs. Accuracy

- Made on 14 (binary/multiclass) UCI datasets
- Comparison with $SVM - L_1$ and $LARS$



Feature Use w/ Breast Cancer Dataset



Conclusion

- We have proposed a new type of classifier...
- ...that is able to decice which features to use for classifying a particular input
- ...which can learn to use *on average* as few features as possible (sparse classifier)

- It has a high learning complexity but a low inference complexity
- It is able to outperforms classical L_1 methdods *at the same level of sparsity*

It is a first step to develop sequential classifiers which learn how to preprocess data for maximizing classification accuracy.

- We have to reduce the learning complexity
- We are applying this idea to more compelx problems like image classification, face recognition and problems where you have an underlying structure between group of features.

Questions?

Questions?