Efficient features selection through coordinate descent with theoretical guarantees.

David Saltiel
A.I Square Connect and LISIC
35 boulevard d'Inkermann, 92200, Neuilly

Eric Benhamou
A.I Square Connect and LAMSADE
35 boulevard d'Inkermann, 92200, Neuilly

Abstract

Despite the advent of representation learning and end-to-end approaches, mainly through deep learning, features selection remains a key point in many machine learning scenarios. This paper introduces a new theoretically motivated method for features selection. The approach that fits within the family of embedded methods, casts the feature selection conundrum as a coordinate ascent optimization with variables dependencies materialized by block variables. Thanks to a limited number of iterations, it proves efficiency for gradient boosting methods, implemented as for instance the XGBoost algorithm. In case of convex and smooth functions, we are able to prove that the convergence rate is polynomial in terms of the dimension of the full features set. We provide comparisons with state of the art methods, Recursive Feature Elimination and Binary Coordinate Ascent and show that this method is competitive.

1 Introduction

Feature selection is also known as variable or attribute selection. This method concerns the selection of a subset of relevant attributes in our data that are most relevant to our predictive modeling problem. It has been an active and fruitful field of research and development for decades in statistical learning. It has proven to be effective and useful in both theory and practice for many reasons: enhanced learning efficiency and increasing predictive accuracy (see Mitra et al. (2002)), model simplification to ease its interpretation and improve performance (see Almuallim and Dietterich (1994), Koller and Sahami (1996) and Blum and Langley (1997)), shorter training time (see Mitra et al. (2002)), curse of dimensionality avoidance, enhanced generalization with reduced overfitting, and implied variance reduction. Both Hastie et al. (2009) and Guyon and Elisseeff (2003) are nice references to get an overview of various methods to tackle features selections. The approaches followed vary; briefly speaking, the methods can be sorted into three main categories: Filter method, Wrapper methods and Embedded methods. We develop these three categories in the following section.

1.1 Features selection methods

1.1.1 Filter methods

Filter type methods select variables despite the model. These methods suppress the least interesting variables using ranking techniques as a criteria to select the variables. Once the ranking is done, a threshold is determined in order to select features with rank above it. These methods are very effective in terms of computation time and robust to overfitting. One famous Filter method approach is the algorithm developed in Kira and Rendell (1992) in 1992, called *Relief*, for application to binary classification problems. By construction, Filter methods may select redundant variables as they do not consider the relationships between variables. One of the most used criteria for Filter methods is the Pearson correlation coefficient, which is simply the ratio between the covariance and the square root of the two variances: $\text{Cov}(x_i, y)/\sqrt{\text{Var}(x_i)\,\text{Var}(y)}$ with x_i the i^{th} feature in the model and y the label associated. It is well known that this correlation ranking can only detect linear dependencies between features ant the target label. The Filter method procedure is summarized in figure 1.

1.1.2 Wrapper methods

Wrapper methods allow detecting possible interactions between variables by evaluating subsets of them. In

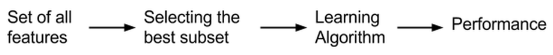

Figure 1: Filter method: it consists in 4 specific steps. Arrows emphasize that these steps are done in chronological order.

Wrapper methods, a model must be trained to test any subsequent feature subset. Consequently, these methods are iterative and computationally expensive. However, these methods can identify the best performing features set for that specific modeling algorithm. Theses methods are explained in the figure 2. Some known examples

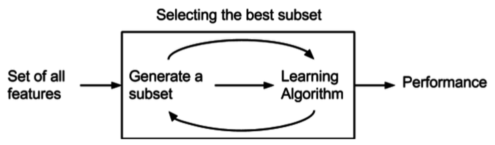

Figure 2: Wrapper method: like Filter method, it consists in 4 different steps but there are iterations between steps 2 and 3 emphasized by the rectangle until each feature subset combination is computed.

of Wrapper methods are forward and backward feature selection methods. These methods are presented in the work of Kohavi and John (1997). The backward elimination starts with all features and progressively remove them. At the opposite, the forward selection starts with an empty set and progressively add them. If we have n features, we need to train n classifiers for the first step, then $n-1$ classifiers for the second step and so on. We then have $n(n+1)/2$ training steps for both methods. However, forward selection starts with small features subsets so it can be computationally cheaper if the stopping condition is satisfied early. An adaptative forward-backward algorithm is proposed in Zhang (2009). One of the state of the art Wrapper method is Recursive Feature Elimination (RFE) (see for instance Mangal and Holm (2018) for more details). It first fits a model and removes features until a pre-determined number of features. Features are ranked through an external model that assigns weights to each features and RFE recursively eliminates features with the least weight at each iteration. One of the main limitation to RFE is that it requires the number of features to keep. This is hard to guess a priori and one may need to iterate much more than the desired number of feature to find an optimal feature set.

1.1.3 Embedded methods

Embedded methods perform features selection as a part of the modeling algorithms execution. Many hybrid methods are developed to combine the advantages of Wrapper and Filter methods. These methods combine learning and feature selection through a prior or a posterior regularization. The different methods of regularization are given in Gaudel and Sebag (2010). The use of L1 regularization is introduced in Tibshirani (1994) and a comparison between L1 and L2 regularization is explained in Ng (2004). Besides, an extension to non-linear spaces is given in Bach (2009), using Multiple Kernel Learning (MLKL). These methods are summarized in figure 3.

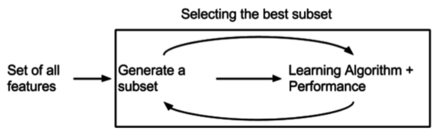

Figure 3: Embedded method: as opposed to Filter and Wrapper methods, there are only 3 steps as the learning and performance steps are combined into a single step. Like for the Wrapper method, we iterate between step 2 and 3 until we find the best features subset among all combinations.

1.2 Ensemble learning

Suppose that we have a training set consisting of a set of points $x_1, \ldots, x_n$ and real values y_i associated with each point x_i. We assume that there is a function with noise $y = f(x) + \varepsilon$, where the noise ε has zero mean and variance σ^2.

We approximate the true function $f(x)$ by finding a function and measure which minimizing the mean squared error between y and $\hat{f}(x)$.

$$\begin{aligned} Err(x) &= E\left[\left(y - \hat{f}(x)\right)^2\right] \\ &= Var\left[\hat{f}(x)\right] + Biais\left[\hat{f}(x)\right]^2 + \sigma^2 \end{aligned}$$

With,

- $Var\left[\hat{f}(x)\right] = E\left[(\hat{f}(x) - E[\hat{f}(x)])^2\right]$
- $Biais\left[\hat{f}(x)\right] = E\left[\hat{f}(x) - f(x)\right]$

A model with a high biais is still missing important informations in the data and a model with a high variance is over-fitting on the training set and will perform badly on the test set. The biais-variance dilemma is summarized in figure 4.

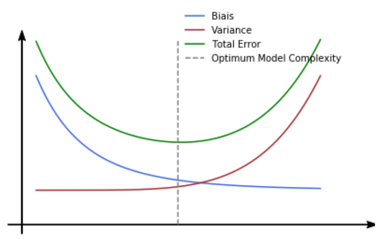

Figure 4: Balance between biais and variance : we can notice that when we decrease the biais (resp. the variance), we increase the variance (resp. the biais).

2 Framework

2.1 Block variables

Diakonikolas and Orecchia (2018) deals with the choice of the number of block to consider in a features selection problem. We propose an answer to the open problem discussed in Beck and Tetruashvili (2013) about solving an unconstrained minimization using a projection method in which each iteration consists of performing a gradient projection step with respect to a certain block taken in a cyclic order or using an alternating minimization method using only two blocks. Indeed, we decide to split the data into $n \geq 2$ blocks $\left([B^1] \ldots [B^n] \right)$ and instead of simply performing a gradient ascent method on a single block at each iteration, we first initialize our method by finding the best sub-block data set in term of prediction score for our supervised learning problem and then apply a gradient ascent method on each single block at every iteration (for more details, see section 3).

2.2 Result of convergence

In order to motivate our method that relies on coordinate ascent, we recall some theoretical results about the convergence of coordinate ascent optimization. The theory is well understood for the convex case (see Wright (2015)). The non convex case without gradient which is our example is however much harder as we have local minima issue and mathematical assumptions too weak to be able to prove convergence. However, convergence results under strong convex conditions provide some hint about the efficiency of this method and its convergence rate that is linear. Our proof provided in supplementary materials is inspired by Nesterov (2012) with a slight modification as we start by the critical point condition. We also provide the various building block lemma in supplementary materials to achieve this proof rapidly. In order to have some meaningful result, we need to make some necessary assumptions for our function f to be minimized. We assume that we have a real value n-dimensional function $f : \mathbb{R}^n \to \mathbb{R}$. We will impose some regularity and smoothness conditions given in 2.2.1 to be able to make some meaningful conclusions. Obviously, even if our final problem is a maximization, it is trivial to turn the minimization program into a maximization one by taking the opposite of the objective function. In this section, we stick to the traditional presentation and examine minimization to make proof reading easier. We examine the following optimization program:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1}$$

Assumption 2.2.1. *We assume our function f is twice differentiable and strongly convex with respect to the Euclidean norm:*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\sigma}{2} \|y - x\|_2^2 \tag{2}$$

for some $\sigma > 0$ and any $x, y \in \mathbb{R}^n$. We also assume that each gradient's coordinate is uniformly L_i Lipschitz, that is, there exists a constant L_i such that for any $x \in \mathbb{R}^n, t \in \mathbb{R}$

$$|[\nabla f(x + te_i)]_i - [\nabla f(x)]_i| \leq L_i |t| \tag{3}$$

With e_i the vector of the canonical basis.
We denote by $L_{\max}$ the maximum of these Lipschitz coefficients :

$$L_{max} = \underset{i=1\ldots n}{max} \; L_i \tag{4}$$

We assume that the minimum of f, denoted by $f^\star$, is attainable and that the left value of the epigraph with respect to our initial starting point x_0 is bounded, that is

$$\max_x \{\|x - x^\star\| : f(x) \leq f(x_0)\} \leq R_0 \tag{5}$$

Remark 2.2.1. *Strong convexity means that the function is between two parabolas. Condition* (3) *implies that the Gradient's growth is at most linear. Inequality* (5) *states that the function is increasing at infinity.*

Proposition 2.2.1. *Under assumption 2.2.1, coordinate ascent optimization (cf. Algorithm 2) converges to the global minimum f^* at a linear rate proportional to $2nL_{\max}R_0^2$, that is*

$$\mathbb{E}[f(x_k)] - f^\star \leq \frac{2nL_{\max}R_0^2}{k} \tag{6}$$

Proof. The proof is given in supplementary materials. □

Remark 2.2.2. *Proposition 2.2.1 gives us a theoretical control on the maximum number of iterations necessary to have an estimated error between the true optimum $f^\star$*

and our estimated optimum computed by $\mathbb{E}[f(x_k)]$. To achieve a precision of ε, we should have

$$\frac{2nL_{\max}R_0^2}{k} \leq \varepsilon$$

which leads to a practical minimum number of iterations $k_{\max,1}$ given by :

$$k_{\max,1} = \left\lceil \frac{2nL_{\max}R_0^2}{\varepsilon} \right\rceil \tag{7}$$

where $\lceil x \rceil$ is the ceiling function, i.e. the least integer greater than or equal to x.

We have an additional control of our convergence rate which is given by the following proposition.

Proposition 2.2.2. *Under assumption 2.2.1, and for $\sigma > 0$, coordinate ascent optimization (cf. Algorithm 2) error is controlled by the following inequality*

$$\mathbb{E}[f(x_k)] - f^\star \leq \left(1 - \frac{\sigma}{nL_{\max}}\right)^k (f(x_0) - f^\star) \tag{8}$$

Proof. The proof is given in supplementary materials. □

Remark 2.2.3. *The latter proposition is interesting as it gives a second theoretical number of iterations. If we want to achieve a precision of ε for our minimum estimation, we should have*

$$\left(1 - \frac{\sigma}{nL_{\max}}\right)^k (f(x_0) - f^\star) \leq \varepsilon$$

which leads to a practical maximum number of iterations $k_{\max,2}$ given by :

$$k_{\max,2} = \left\lceil \log\left(\frac{\varepsilon}{f(x_0) - f^\star}\right) / \log\left(1 - \frac{\sigma}{nL_{\max}}\right) \right\rceil \tag{9}$$

Remark 2.2.4. *Our function to be maximized is obviously not convex. However, a linear rate in the convex case is rather a good performance for the ascent optimization method. Provided the method generalizes which is still under research, this convergence rate is a good hint of the efficiency of this method.*

Remark 2.2.5. *Combining the results in (7) and (9), the optimal number of iterations must be the minimum of our two previous bounds : $\min(k_{\max,1}, k_{\max,2})$, that is*

$$\min\left(\left\lceil \frac{2nL_{\max}R_0^2}{\varepsilon} \right\rceil, \left\lceil \frac{\log\left(\frac{\varepsilon}{f(x_0) - f^\star}\right)}{\log\left(1 - \frac{\sigma}{nL_{\max}}\right)} \right\rceil\right)$$

The comparison between the two possible numbers of iteration is presented in figure 5. Interestingly, the two speeds of convergence play a different role. For small precision (large ε), the stringent boundary is the one from equation (7), while for high precison (small ε), the stringent boundary is the one from equation (9).

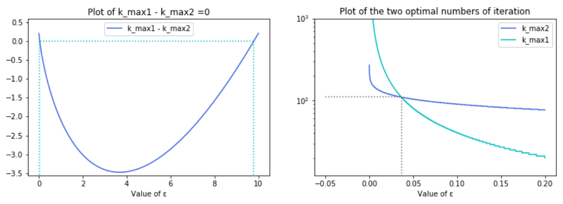

Figure 5: Comparison between the two possible numbers of iteration : we took as an example a 1-dimensional function to minimize with a starting point at 10 and we suppose the traditional convention that the minimum is at the origin. We then suppose the constants $L_{\max}$ and R_0 equal to one and the constant σ equal to 0.1. We first notice that we have two different cases when $k_{\max,1}$ is greater or smaller than $k_{\max,2}$ but it's obvious in our case to only study the case when the precision is not larger than 1. So, this reasoning leads to a choice of a precision less than 3.36% and to the choice of $k_{\max,2}$.

3 Method developed

In many applications, we can regroup features among families. We call these features block variables. Typical example is to regroup variables that are observations of some physical quantity but at a different time (like the speed of the wind measure at different hours for some energy prediction problem, the price of a stock in an algorithmic trading strategy for financial markets, the temperature or heart beat of a patient at different time, ...). We denote from now $\mathcal{M}_{(n,p)}$ the space of matrices of n rows and p columns. Supposing that we have J rows of data, we can formally regroup our variables into two sets:

- the first set encompasses $[B^1] \ldots [B^n]$. These are block variables of different length L_i. Mathematically, the sub-block variables are denoted by B_j^i with $(B_j^i)_{\substack{i\in 1\ldots n \\ j\in 1\ldots L_i}}$ taking value in $\mathbb{R}^J$.
- the second set is denoted $[S]$ and is a block of p single variables.

Graphically, our variables looks like that:

$$\left(\begin{array}{ccccccccccc} \multicolumn{3}{c}{\overbrace{\qquad\qquad}^{B^1}} & & \multicolumn{3}{c}{\overbrace{\qquad\qquad}^{B^n}} & & \multicolumn{3}{c}{\overbrace{\qquad\qquad}^{S}} \\ B_1^1 & \ldots\ldots & B_{L_1}^1 & & B_1^n & \ldots\ldots & B_{L_n}^n & & S_1 & \ldots\ldots & S_p \\ \hline \bullet & \ldots\ldots & \bullet & & \bullet & \ldots\ldots & \bullet & & \bullet & \ldots\ldots & \bullet \\ \vdots & & \vdots & \ldots\ldots & \vdots & & \vdots & & \vdots & & \vdots \\ \bullet & \ldots\ldots & \bullet & & \bullet & \ldots\ldots & \bullet & & \bullet & \ldots\ldots & \bullet \end{array}\right)$$

Thus, we have N variables split between block variables and single variables, hence $N = N_B + p$ with $N_B = \sum_{i=1}^{n} L_i$.

Our algorithm works as follows. We first fit our classification model to find a ranking of features importance. The performance is computed with the Gini index for each variable. We then keep the first k best ranked features for each blocks $B^1 \dots B^n$ in order to find the best initial guess for our coordinate ascent algorithm. Notice that the set of unique variables is not modified during the first step of the procedure. The objective function is the number of correctly classified samples at each iteration. It can be expressed as followed:

$$accuracy = \frac{TP + TN}{\text{number of examples}} \qquad (10)$$

with True Positives (TP) denoting the number of positive examples labeled as positive and True Negatives (TN) the number of negative examples labeled as negative.

Remark 3.0.1. *Notice that the number of examples (denoted by n in this remark) respects the following relation : $n = TP + TN + FP + FN$ with False Positives (FP) denoting the number of negative examples labeled as positive and False Negatives (FN) the number of positive examples labeled as negative.*

We then enter the main loop of the algorithm. Starting with the vector of $\left(k, \dots, k,\ \mathbb{1}_p^T\right)$ as the initial guess for our algorithm, we perform our coordinate ascent optimization in order to find the set with optimal score and the minimum number of features. The coordinate ascent loop stops whenever we either reach the maximum number of iterations or the current optimal solution has not moved so far between two steps.

Taking the previous notation, we start with the initial block data set $\left(\left[B_1^1 \dots B_{L_1}^1\right] \dots \left[B_1^n \dots B_{L_n}^n\right], [S]\right)$ and reduce it to the new data set $\left(\left[B^{1,\diamond}\right] \dots \left[B^{n,\diamond}\right], [S]\right)$, with $B^{i,\diamond} \in \mathcal{M}_{(J,k)} \quad \forall i \in 1 \dots n$. We can the firstly apply a coordinate ascent optimization on each single block and get a new data set structure without considering blocks. We secondly apply a binary coordinate ascent optimization on the remain data set in order to get the final data set $\left(\left[B^{1,\star}\right] \quad \dots \quad \left[B^{n,\star}\right], \quad \left[S^{\star}\right]\right)$. The procedure is detailed in figure 6. We summarize the algorithm in the pseudo code 1. To control early stop, we use precision variables denoted by ε_1 and ε_2, and two maximum iterations Iteration $\max_1$ and Iteration $\max_2$ that are initialized before starting the algorithm. We also denote Score$(k_1,\ \dots,\ k_n,\ \mathbb{1}_p)$ to be the accuracy score of our classifier with each B_i block of variables retaining k_i best variables and with single variable all retained.

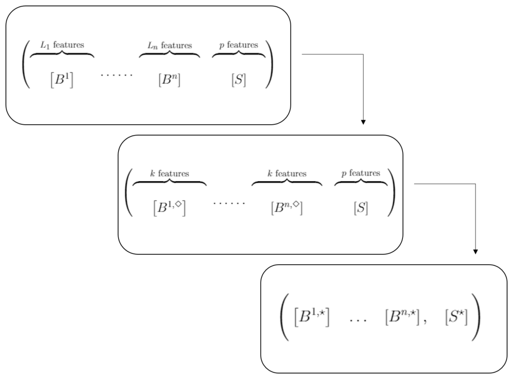

Figure 6: Summary of our method : we fisrt compute the best value for k in order to have all blocks (except the last block of single features) of k features maximizing the score function. We then apply a coordinate ascent method on each block to maximize the score function. We finally apply a binary coordinate ascent on the remaining features. We will denote by Optimal Coordinate Ascent (OCA) our method developed for simplicity.

Remark 3.0.2. *The originality of this coordinate ascent optimization is to regroup variable by blocks, hence it reduces the number of iterations compared to Binary Coordinate Ascent (BCA) as presented in Zarshenas and Suzuki (2016). The stopping condition can be changed to accommodate for other stopping conditions.*

Remark 3.0.3. *There are many variants to this algorithm. It can be modified by using a randomized coordinate ascent. In this case, we choose the index randomly at each step instead of using the provided order. The pseudo code 2 is listed below :*

Algorithm 2 Randomized Coordinate ascent :

```
  Initialization
  Start with $x_0 \in \mathbb{R}^n$
  Set $k = 0$
  while stop criteria not satisfied do
      Choose index $i_k$ uniformly distributed in $\{1, \dots, n\}$ independently from prior iteration
      Set $x_{k+1} = x_k - \alpha_k \left[\nabla f(x_k)\right]_{i_k} e_{i_k}$ for some $\alpha_k > 0$
      Set $k = k + 1$
  end while
```

Remark 3.0.4. *The specificity of our method is to keep the j best representative features for each features class, as opposed to other methods that only select one representative feature from each group, ignoring the strong similarities between each feature of a given variable block. This takes in particular the opposite view of feature Selection with Ensembles, Artificial Variables,*

```
Algorithm 1 OCA algorithm :
  J Best optimization
  We retrieve features importance from a fitted model
  We find the index $k^\star$ that gives the best score for variables block of same size $k$:
  $k^\star \in \underset{k \in \mathbb{R}^{L_{\min}}}{\text{argmax}} \text{ Score}\left(k,\ \ldots,\ k,\ \mathbb{1}_p\right)$
  Initial guess : $x^0 = (k^\star, \ldots, k^\star, \mathbb{1}_p)$
  while $|\text{Score}(x^i) - \text{Score}(x^{i-1})| \geq \varepsilon_1$ and $i \leq$ Iteration $\max_1$ do
    $x_1^i \in \underset{j \in \mathbb{R}^{L_1}}{\text{argmax}} \text{ Score}\left(j,\ x_2^{i-1},\ x_3^{i-1},\ \ldots\ ,\ x_n^{i-1}, \mathbb{1}_p\right)$
    ...
    $x_n^i \in \underset{j \in \mathbb{R}^{L_n}}{\text{argmax}} \text{ Score}\left(x_1^i,\ x_2^i,\ x_3^i,\ \ldots\ , j, \mathbb{1}_p\right)$
    i += 1
  end while

  Full coordinate ascent optimization
  Use previous solutions: $X^* = (x_1^i, \ldots, x_n^i, \mathbb{1}_p)$
  $Y^* = \text{Score}\left(X^*\right)$
  while $|Y - Y^*| \geq \varepsilon_2$ and iteration $\leq$ Iteration $\max_2$ do
    for i=1 ... N do
      $X = X^*$
      $X_i = \text{not}(X_i^*)$
      if Score $(X) \geq$ Score $(X^*)$ then
        $X^* = X$
      end if
    end for
    $Y = \text{Score}\left(X^*\right)$
    iteration += 1
  end while
  Return $X^*, Y^*$
```

and Redundancy Elimination as developed in Tuv et al. (2009).

3.1 Stopping criteria

To design an efficient optimization algorithm, stopping conditions are critical as they will ensure that we do not do useless iterations and stop as soon as possible. We will use the precision and the number of iteration discussed in section 3.

4 Numerical Results

4.1 Data set

We carry out our experiment on real finance data set. In financial markets, algorithmic trading has become more and more standard over the last few years. The rise of the machine has been particularly significant in liquid and electronic markets such as foreign exchange and futures markets reaching between 60 to 80 percent of total traded volume (see for instance Chan (2013), Goldstein et al. (2014) or Chaboud et al. (2015) for more details on the various markets). These strategies are even more concentrated whenever there are very fast market moves as reported in Kirilenko et al. (2017). A trading strategy is usually defined with some signal that generates a trading entry. But once we are in position, then main following point is the trading exit strategy. There are multiple method to handle efficient exits, ranging from fixed target and stop loss, to dynamic target and stop loss. Indeed, to enforce success and crystallize gain or limit loss, a common practice is to associate to the strategy a profit target and stop loss as described in various papers (Labadie and Lehalle (2010), Giuseppe Di Graziano (2014), Fung (2017), or Vezeris et al. (2018)). We use our algorithm to do a supervised classification according to some a priori features. We are given 1500 trades over a ten years history with 135 features that can be classified into five blocks of twenty variables, one block of thirty variables and five single variables. We know for each trade whether it is a ‘good’ or ‘bad’ trade (thanks to the gain and loss, denoted by PnL, engendered by the trade). The idea is to use the minimum number of features to classify a priori this data set. We use cross validation with 70% for the training set and 30% for the test sets. The procedure is summarized in figure 7. For full reproducibility, full data set and corresponding python code for this algorithm is available publicly on github. We consider a gradient boosting framework by using XGBoost library, initially developed in Chen (2016), in the implementation of our algorithm. It is illuminating to

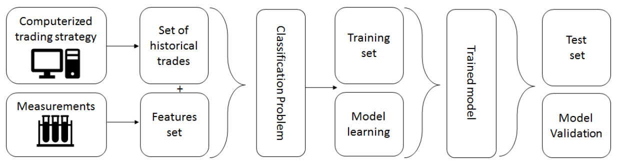

Figure 7: Learning process for our trade selection challenge. We first use a proprietary trading strategy that generates some samples trades. We take various measures before the trades is executed to create a feature set. We combined these to create a supervised learning classification problem. Using xgboost method and OCA, we learn model parameters on a train set. We monitor overall performance of the trading strategy on a separate test set to validate scarce overfitting.

look at the histogram of gain and losses of our trades over our 10 years of history. We can observed two peaks corresponding to the profit target and stop loss level as shown in figure 8. It is much better to use the gain and loss curve in the native currency of the underlying instrument than to look at the consolidated currency of our trading strategies to avoid foreign exchange noise. We will consider two data sets of a 'bad' strategy, called strategy 1, and a 'good' one, called strategy 2; each strategy engendering trades among a fixed period of time.

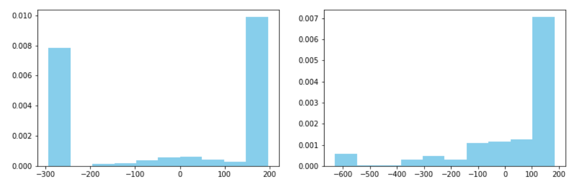

Figure 8: Histogram of the PnL of the the two strategies in native currency. The left histogram corresponds to the strategy 1 and the right one to the second one. As the algorithm is performed on an underlying asset listed on an American financial market, the resulting strategy is denominated in USD. The strategy uses fixed stop loss and profit target level. This leads to two major columns corresponding for the left peak to trades that end in a loss when they hit the stop loss level and for the right peak to trades that exit with a profit as they reach the profit target.

The graph of the evolution of the gain and loss of the two strategies is given in figure 9.

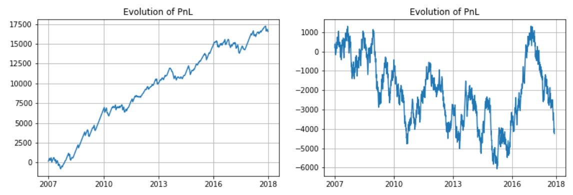

Figure 9: Evolution of the gain and loss of the two strategies.

Remark 4.1.1. *The data set of the strategy 1 is quite balanced in terms of label (PnL engendered by the strategy) whereas it is not the case for the second one, we can't thus apply the definition of the accuracy given in* (10) *because it would perform badly and it is recommended to use the definition of the balance accuracy, that is (taking the previous notation introduced in* (10) *and remark 3.0.1:*

$$\textit{Balance accuracy} = \frac{TPR + TNR}{2}$$

with the True Positive Rate (TPR) and the True Negative Rate (TNR) defined as :

$$TPR = \frac{TP}{TP + FN}, \quad TNR = \frac{TN}{TN + FP}$$

True Positive Rate (TPR) concentrates on getting accurate results on the positive labels, regardless of false results. On the opposite, True Negative Rate (TNR) focuses on avoiding to incorrectly classify as a true label something that is a negative label. This is similar to type I and type II error in statistical tests.

4.2 Comparison

We first analyze the results given by the strategy 1. We compare our method to two other methods that are supposed to be state of the art for feature selections, namely RFE and BCA. Our new method achieves a score of 62.80 % with 16% of features used, to be compared to RFE that achieves 62.80 % with 19% of features used. BCA performs poorly with a highest score given by 62.19 % with 27.08% of features used. If we take in terms of efficiency criterium, the highest score with the less feature, our method is the most efficient among these three methods. In comparison, with the same number of features, namely 16.6%, RFE gets a score of 62.39 %. All these figures are summarized in the table 2.

Table 1: Method Comparison for the strategy 1: for each row, we provide in red the best(s) (hotest) method(s) and in blue the worst (coldest) method, while intermediate methods are in orange. We can notice that OCA achieves the higher score with the minimum feature sets. For the same cardinal of features set, RFE performs worst or equally, if we want the same performance for RFE, we need to have a larger feature set. BCA is the worst method both in terms of score and minimum feature set.

Method	**% of features**	**Score (in %)**
OCA using 24 features	16.6	62.8
RFE using 24 features	16.6	62.39
BCA using 39 features	27.08	62.19
RFE using 28 features	19.4	62.8

Table 2: Method Comparison for the strategy 2:

Method	**% of features**	**Score (in %)**
OCA using 10 features	6.75	74.28
RFE using 10 features	6.75	50.47
BCA using 52 features	35.13	77.14

5 Discussion

Compared to BCA our method reduces the number of iterations as it uses the fact that variables can be regrouped into categories or classes. The number of iterations for both OCA and BCA methods is provided below in figure 10. Our method requires only 350 iterations steps ton converge as opposed to BCA that needs up to 700 iterations steps as it considers blindly variables ignoring similarities between them.

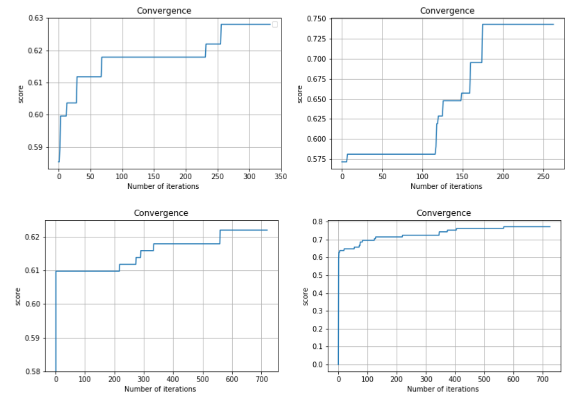

Figure 10: Iterations steps up to convergence for OCA and BCA. OCA method is above while BCA is below. The figures on the left represent the strategy 1 and the strategy 2 is represented on the right. We see that OCA requires around 250 or 350 iteration steps to converge whereas BCA requires the double (around 700 iteration) steps to converge.

Graphically, we can determine the best candidates for the three methods listed in table 2 in figures 11, 12 and 13 for both strategies. We have taken the following color code. The hottest (or best performing) method is plotted in red, while the worst in blue. Average performing methods are plotted in orange. In order to compare finely OCA and RFE, we have plotted in figure 12 the result of RFE for used features set percentage from 10 to 30 percent. We can notice that for the same feature set as OCA, RFE has a lower score and equally that to get the same score as OCA, RFE needs a large features set.

5.1 Reduced overfitting

We look at the final goal which is to compare the trading strategy with and without machine learning. A standard way in machine learning is to split our data set between a randomized training and test set. We keep one third of our data for testing to spot any potential overfitting. If we use the standard and somehow naive way to take randomly one third of the data for our test set, we break

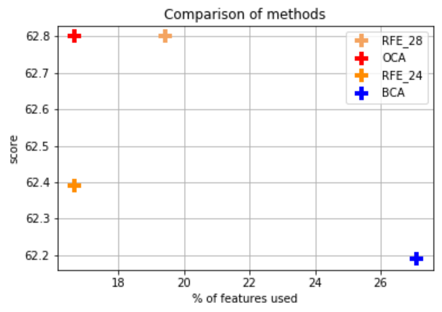

Figure 11: Comparison between the 3 methods for strategy 1. To qualify the best method, it should be in the upper left corner. The desirable feature is to have as little features as possible and the highest score. We can see that the red cross that represents OCA is the best. The color code has been designed to ease readability. Red is the best, orange is a slightly lower performance while blue is the worst.

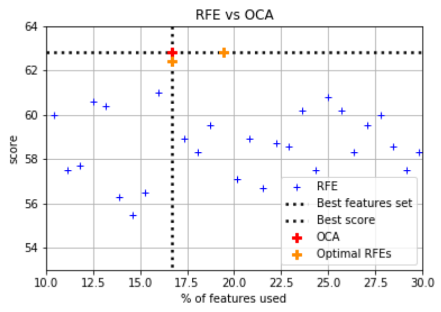

Figure 12: Comparison between OCA and RFE for strategy 1: For RFE, we provide the score for various features set in blue. The two best RFE performers points are the orange cross marker points that are precisely the one listed in table 2. The red cross marker point represents OCA. It achieves the best efficiency as it has the highest score and the smallest feature set for this score.

the time dependency of our data. This has two consequences. We use in our training set some data that are after our test sets which is not realistic compared to real life. We also neglect any regime change in our data by mixing data that are not from the same period of time. However, we can do the test on this mainstream approach and compare the trading strategy with and without machine learning filtering. This is provided in figure 14. The orange (respectively blue) curve represents our algorithmic trading strategy without any machine learning filtering (respectively with filtering done by the xgboost method trained with OCA). Since the blue curve is above the orange one, we experimentally validate that using machine learning enhances the overall profitability of our

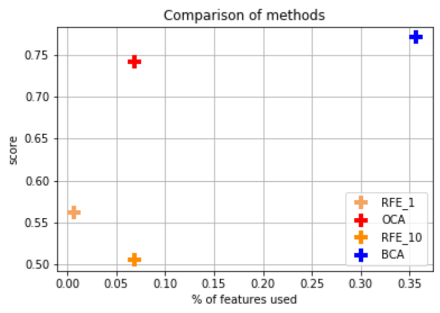

Figure 13: Comparison between the 3 methods for strategy 2. We keep the same color code as before. The best performer point for the RFE method has a less percentage of features used than the OCA method but the score of the last one is much better. Besides, the BCA method lead to a higher score than the two other methods but with a larger features set.

trading strategy by avoiding the bad trades.

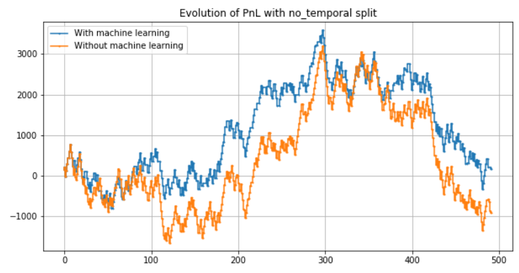

Figure 14: Evolution of the PnL for strategy 1 with a randomized test set. The orange curve represents our algorithmic trading strategy without any machine learning filtering while the blue line is the result of the combination of our algorithmic trading strategy and the oca method to train our xgboost method.

If instead we split our set into two sets that are continuous in time, meaning we use as a training test the first two third of the data when there are sorted in time and as a test set the last third of the data, we get better result as the divergence between the blue and orange curve is larger. An explanation of this better efficiency may come from the fact that the non randomization of the training set makes the learning for our model easier and leads to less overfitting overall. This method of splitting the two sets is illustrated in figures 15 and 16. Since the blue curve is above the orange one for both strategies, we experimentally validate that using machine learning enhances the overall profitability of our trading strategy by avoiding the bad trades.

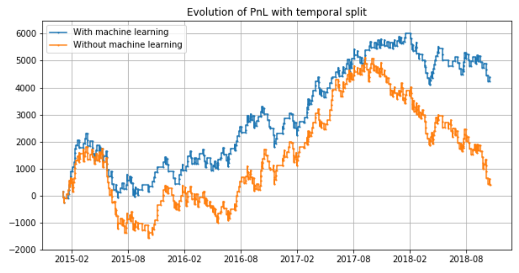

Figure 15: Evolution of the PnL for strategy 1 with a test set given by the last third of the data to take into account temporality in our data set. The orange curve represents our algorithmic trading strategy without any machine learning filtering while the blue line is the result of the combination of our algorithmic trading strategy and the OCA method to train our xgboost method.

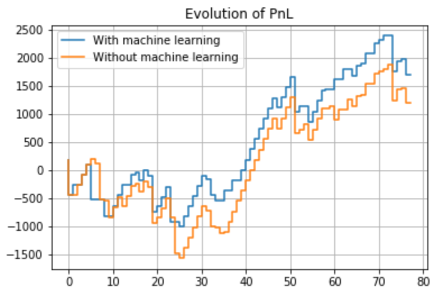

Figure 16: Evolution of the PnL for strategy 2 with temporal split. We keep the same color code as before.

6 Conclusion

In this paper, we have presented a new method, called Optimal Coordinate Ascent (OCA) that allows us selecting features among block and individual features. OCA relies on coordinate ascent to find an optimal solution for gradient boosting methods score (number of correctly classified samples for our data). OCA takes into account the notion of dependencies between variables forming blocks in our optimization. The coordinate ascent optimization solves the issue of the NP hard original problem where the number of combinations rapidly explodes making a grid search unfeasible. It transforms the NP hard problem of finding the best features into a polynomial search one. Comparing result with two other methods, Binary Coordinate Ascent (BCA) and Recursive Feature Elimination (RFE), we find that OCA leads to the minimum feature set with the highest score. Hence, OCA provides empirically the most compact data set with optimal performance.

References

Almuallim, H., Dietterich, T.G., 1994. Learning boolean concepts in the presence of many irrelevant features. Artificial Intelligence 69, 279–305.

Bach, F.R., 2009. Exploring large feature spaces with hierarchical multiple kernel learning, in: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (Eds.), Advances in Neural Information Processing Systems 21. Curran Associates, Inc., pp. 105–112.

Beck, A., Tetruashvili, L., 2013. On the convergence of block coordinate descent type methods. SIAM Journal on Optimization 23, 2037–2060.

Blum, A.L., Langley, P., 1997. Selection of relevant features and examples in machine learning. Artif. Intell. 97, 245–271.

Chaboud, Alain p.and Chiquoine, B., Hjalmarsson, E., Vega, C., 2015. Rise of the machines: Algorithmic trading in the foreign exchange market. The Journal of Finance 69, 2045–2084.

Chan, E., 2013. Algorithmic Trading: Winning Strategies and Their Rationale. 1st ed., Wiley Publishing.

Chen, 2016. Xgboost documentation. URL: https://xgboost.readthedocs.io/en/latest/index.html.

Diakonikolas, J., Orecchia, L., 2018. Alternating randomized block coordinate descent, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, Stockholmsmssan, Stockholm Sweden.

Fung, S.P.Y., 2017. Optimal online two-way trading with bounded number of transactions. CoRR .

Gaudel, R., Sebag, M., 2010. Feature Selection as a One-Player Game, in: International Conference on Machine Learning, Haifa, Israel. pp. 359–366. URL: https://hal.inria.fr/inria-00484049.

Giuseppe Di Graziano, D.B.A., 2014. Optimal trading stops and algorithmic trading. SSRN URL: https://ssrn.com/abstract=2381830.

Goldstein, M., Viljoen, T., Westerholm, P.J., Zheng, H., 2014. Algorithmic trading, liquidity, and price discovery: An intraday analysis of the spi 200 futures. The Financial Review 49, 245–270.

Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182.

Hastie, T., Tibshirani, R., Friedman, J.H., 2009. The elements of statistical learning: data mining, inference, and prediction, 2nd Edition. Springer series in statistics, Springer.

Kira, K., Rendell, L.A., 1992. A practical approach to feature selection, in: Proceedings of the Ninth International Workshop on Machine Learning, San Francisco, CA, USA.

Kirilenko, A., Kyle, A.S., Samadi, M., Tuzun, T., 2017. The flash crash: High-frequency trading in an electronic market. Journal of Finance 72, 967–998.

Kohavi, R., John, G.H., 1997. Wrappers for feature subset selection. ARTIFICIAL INTELLIGENCE 97, 273–324.

Koller, D., Sahami, M., 1996. Toward optimal feature selection, in: Proceedings of the Thirteenth International Conference on International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 284–292.

Labadie, M., Lehalle, C.A., 2010. Optimal algorithmic trading and market microstructure. Working Papers. HAL.

Mangal, A., Holm, E.A., 2018. A comparative study of feature selection methods for stress hotspot classification in materials. ArXiv e-prints .

Mitra, P., Murthy, C.A., Pal, S.K., 2002. Unsupervised feature selection using feature similarity. IEEE Trans. Pattern Anal. Mach. Intell. 24, 301–312.

Nesterov, Y., 2012. Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM Journal on Optimization 22, 341–362.

Ng, A.Y., 2004. Feature selection, l1 vs. l2 regularization, and rotational invariance, in: ICML '04.

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

Tuv, E., Borisov, A., Runger, G., Torkkola, K., 2009. Feature selection with ensembles, artificial variables, and redundancy elimination. J. Mach. Learn. Res. 10, 1341–1366.

Vezeris, D., Kyrgos, T., Schinas, C.T.P., Loss, S., 2018. Trading strategies comparison in combination with an macd trading system. J. Risk Financial Manag 11, 56.

Wright, 2015. Coordinate descent algorithms. Math. Program. 151, 3–34.

Zarshenas, A., Suzuki, K., 2016. Binary coordinate ascent: An efficient optimization technique for feature subset selection for machine learning. Knowledge-Based Systems 110, 191 – 201.

Zhang, T., 2009. Adaptive forward-backward greedy algorithm for sparse learning with linear models, in: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (Eds.), Advances in Neural Information Processing Systems 21. Curran Associates, Inc., pp. 1921–1928.

Supplementary Material for the article : Efficient features selection through coordinate descent with theoretical guarantees.

1 Proofs

1.1 Proof of proposition 2.2.1

In order to prove result, we first start by a simple lemma.

Lemma 1.1. *If $(u_n)_{n\in\mathbf{N}}$ is non increasing such that $u_n - u_{n+1} \geq a u_n^2$ with $a > 0$, then $u_n \leq \frac{1}{na}$.*

Proof. We remark that $u_{n+1} \leq u_n - a u_n^2$ or equivalently $u_{n+1} \leq u_n (1 - a u_n)$, which says that u_{n+1} is bounded by a lower parabola. Let $f : x \rightarrow x\,(1-ax)$ be this parabola. f's variation are easy to study with its derivative given by $f'(x) = 1 - 2ax$. It is an increasing function up to $1/2a$ with its maximum given by $1/4a$ and decreasing afterwards.

We can now trivially prove our result by induction. The initialization step is obvious for $k \leq 4$ as $u_k \leq \frac{1}{ka}$ since the global maximum of our parabola is $1/4a$ which is less than $1/ka$ for $k \leq 4$. If the result holds for $k \geq 2$, we know that the maximum of the parabola ($f(U_k)$) is attained in $\frac{1}{ka}$ since $\frac{1}{ka} \leq \frac{1}{2a}$. This implies that

$$u_{k+1} \leq \frac{1}{ka} - a\frac{1}{(ka)^2} \quad \text{or} \quad u_{k+1} \leq \frac{k-1}{k^2 a}$$

We can trivially conclude as $\frac{k-1}{k^2} \leq \frac{k-1}{k^2-1} = \frac{1}{k+1}$ □

Proof. We can now prove the result of proposition 2.2.1. By assumptions, we do a gradient descent according to one coordinate: $x_{k+1} = x_k - \alpha_k \left[\nabla f(x_k)\right]_{i_k} e_{i_k}$ for some $\alpha_k \geq 0$. A Taylor-Lagrange expansion for $f(x_{k+1})$'s gradient gives us:

$$\begin{aligned} \nabla f(x_{k+1}) &\triangleq \nabla f\left(x_k - \alpha_k \left[\nabla f(x_k)\right]_{i_k} e_{i_k}\right) \\ &= \nabla f(x_k) - \langle \alpha_k \left[\nabla f(x_k)\right]_{i_k} e_{i_k} \,, \\ &\quad \mathcal{H} f\left(\theta_k x_k + (1-\theta_k)\, x_{k+1}\right)\rangle \\ &\quad \text{for } \theta_k \in \left]0, 1\right[\end{aligned}$$

We denote $C_k = \theta_k x_k + (1-\theta_k)\, x_{k+1}$. We want α_k such that $\nabla f(x_{k+1}) = 0$. Combined with (1), we have the following equality:

$$\alpha_k \langle \left[\nabla f(x_k)\right]_{i_k} e_{i_k} \,,\ \mathcal{H} f(C_k)\rangle = \nabla f(x_k)$$

Taking the norm and using the Cauchy Schwarz inequality, we have that :

$$\|\nabla f(x_k)\| \leq |\alpha_k| \left\|\left[\nabla f(x_k)\right]_{i_k} e_{i_k}\right\| \, \|\mathcal{H} f(C_k)\|$$

Bounding the Hessian from equation (3) and using equation (4), we have: $\|\nabla f(x_k)\| \leq |\alpha_k| \, \|\nabla f(x_k)\| \, L_{\max}$

Assuming that the objective function minimum is not attained at step k, $\|\nabla f(x_k)\| \neq 0$, we have: $\frac{1}{L_{\max}} \leq \alpha_k$. We precisely take this critical value $1/L_{\max}$ for α_k at each step k in order to avoid a step too large to prevent oscillation phenomena. Our recursive relationship is now:

$$x_{k+1} = x_k - \frac{1}{L_{\max}} \left[\nabla f(x_k)\right]_{i_k} e_{i_k} \tag{1}$$

Using the fact that $Var\left(\|\nabla f(x_k)\|\right) \geq 0$, we can conclude that

$$\mathbb{E}\left[\|\nabla f(x_k)\|^2\right] \geq \mathbb{E}\left[\|\nabla f(x_k)\|\right]^2 \tag{2}$$

By convexity of f, we have

$$\begin{aligned} f(x_k) - f(x^\star) &\leq \langle \nabla f(x_k) \,,\ x_k - x^\star\rangle && (3) \\ &\leq \|\nabla f(x_k)\| \, \|x_k - x^\star\| && (4) \\ &\qquad \text{(by Cauchy Schwartz)} \\ &\leq \|\nabla f(x_k)\| \, R_0 && (5) \\ &\qquad \text{(by assumption (5))} \end{aligned}$$

We denote $U_k = \mathbb{E}\left[f(x_k)\right] - f(x^\star)$. Taking the expectation over all the random index i_k in equation (6), we obtain :

$$U_k \leq \mathbb{E}\left[\|\nabla f(x_k)\|\right] R_0 \tag{6}$$

$U_k \geq 0$ for any k by definition of $x^\star$ which is the minima of f. So, taking the square value on both side of the inequality, we have

$$\begin{aligned} U_k^2 &\leq \mathbb{E}\left[\|\nabla f(x_k)\|\right]^2 R_0^2 & (7)\\ &\leq \mathbb{E}\left[\|\nabla f(x_k)\|^2\right] R_0^2 & (8)\\ &\text{by inequality (2)} \end{aligned}$$

Using equation (1), we apply Taylor-Lagrange to our objective function f at step k+1 :

$$\begin{aligned} f(x_{k+1}) &= f(x_k) - \frac{1}{L_{\max}} \langle [\nabla f(x_k)]_{i_k} e_{i_k}, \nabla f(x_k) \rangle \\ &+ \frac{1}{2}\left(\frac{1}{L_{\max}}\right)^2 \left([\nabla f(x_k)]_{i_k} e_{i_k}\right)^T \\ &\quad \mathcal{H}f(d_k)\left([\nabla f(x_k)]_{i_k} e_{i_k}\right) & (9) \end{aligned}$$

with $d_k \in]x_k, x_{k+1}[$. Therefore:

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \frac{1}{L_{\max}} \left\|[\nabla f(x_k)]_{i_k}\right\|^2 \\ &\quad + \frac{1}{2}\frac{1}{L_{\max}^2} \left\|[\nabla f(x_k)]_{i_k}\right\|^2 L_{\max} & (10)\\ &\leq f(x_k) - \frac{1}{2L_{\max}} \left\|[\nabla f(x_k)]_{i_k}\right\|^2 & (11) \end{aligned}$$

Taking the expectation over all the random indexes i_k, we have :

$$\begin{aligned} \mathbb{E}\left[\mathbb{E}_{i_k}[f(x_{k+1})]\right] &\leq \mathbb{E}\Bigg[\mathbb{E}_{i_k}\Bigg[f(x_k) \\ &\quad - \frac{1}{2L_{\max}} \frac{1}{n} \sum_{i=1}^{n} [\nabla f(x_k)]_i^2 \Bigg]\Bigg] & (12)\\ \mathbb{E}[f(x_{k+1})] &\leq \mathbb{E}[f(x_k)] \\ &\quad - \frac{1}{2nL_{\max}} \mathbb{E}\left[\|[\nabla f(x_k)]\|^2\right] & (13) \end{aligned}$$

Subtracting $f(x^\star)$ on both side, we obtain the main recursive relation between U_k and U_{k+1} :

$$U_{k+1} \leq U_k - \frac{1}{2nL_{\max}} \mathbb{E}\left[\|[\nabla f(x_k)]\|^2\right] \qquad (14)$$

Using the inequality (9), we ensure that :

$$U_{k+1} \leq U_k - \frac{1}{2nL_{\max}} \frac{1}{R_0^2} U_k^2 \qquad (15)$$

We can conclude using lemma 1.1 to get $U_k \leq \frac{2nL_{\max}}{k}$ so that the first result of proposition 2.2.1 holds. □

1.2 Proof of proposition 2.2.2

Proof. The result of the proposition, given by inequality (8) is also easy to prove. Taking the minimum of both sides of inequality (2) leads to $f^\star \geq f(x_k) - \frac{1}{2\sigma}|\nabla f(x_k)|^2$ or equivalently $|\nabla f(x_k)|^2 \geq 2\sigma U_k$. Then using inequality (14), we get

$$U_{k+1} \leq U_k - \frac{\sigma}{nL_{\max}} U_k = (1 - \frac{\sigma}{nL_{\max}})U_k.$$

The result is trivially obtained by applying this formula recursively. □