

Feature selection with optimal coordinate ascent (OCA)

David Saltiel^{a,b}, Eric Benhamou^{a,c}

^a*A.I. SQUARE CONNECT, 35 Boulevard d'Inkermann, 92200 Neuilly sur Seine, France*

^b*LISIC - Université du Littoral - Cote d'Opale, France*

^c*LAMSADE, Université Paris Dauphine,
Place du Maréchal de Lattre de Tassigny, 75016 Paris, France*

Abstract

In machine learning, Feature Selection (FS) is a major part of efficient algorithm. It fuels the algorithm and is the starting block for our prediction. In this paper, we present 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). 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 explode making a grid search unfeasible. It reduces considerably the number of iterations changing this NP hard problem into a polynomial search one. OCA brings substantial differences and improvements compared to previous coordinate ascent feature selection method: we group variables into block and individual variables instead of a binary selection. Our initial guess is based on the k-best group variables making our initial point more robust. We also introduced new stopping criteria making our optimization faster. We compare these two methods on our data set. We found that our method outperforms the initial one. We also compare our method to the Recursive Feature Elimination (RFE) method and find that OCA leads to the minimum feature set with the highest score. This is a nice byproduct of our method as it provides empirically the most compact data set with optimal performance.

Keywords: feature selection, coordinate ascent, gradient boosting method

2010 MSC: 68T01, 68T05

1 Introduction

Feature selection is also known as variable or attribute selection. It is the selection of a subset of relevant attributes in our data that are most relevant to our predictive modeling problem. It has been

*Fully documented templates are available in the elsarticle package on CTAN.

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, 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 varies. Briefly speaking, the methods can be sorted into three main categories: Filter method, Wrapper methods and Embedded methods. We developed these three categories in the following section.

1.1 Features selection methods

1.1.1 Filter methods

Filter type methods select variables regardless of the model. These methods suppress the least interesting variables by using ranking techniques as a criteria to select the variables. Once the ranking is done, a threshold is determined in order to select features above it. These methods are very effective in terms of computation time and robust to overfitting. By construction, filter methods may select redundant variables as they do not consider the relationships between variables. To stress this last point, we can present one of the most known criteria, 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 and the target label.

1.1.2 Wrappers methods

Wrapper methods evaluate subsets of variables. They thus allow detecting possible interactions between variables. In 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. Some known examples of wrapper methods are forward and backward feature selection methods.

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 $\frac{n(n+1)}{2}$ 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. One of the State of the art wrappers 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 method perform feature selection as a part of the modeling algorithm’s execution. Many hybrid methods are developed to combine the advantages of wrappers and filters methods

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 appendix section 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 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. 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:

$$\min_x f(x) \tag{1}$$

We denote by e_i the traditional vector with 0 for any coordinate except 1 for coordinate i . It is the vector of the canonical basis.

Assumption 2.1. *We assume our function f is twice differentiable and strongly convex with respect to the Euclidean norm:*

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\sigma}{2} \|y - x\|_2^2 \text{ for some } \sigma > 0 \text{ and any } x, y \in \mathbb{R}^n \tag{2}$$

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 $x \in \mathbb{R}^n, t \in \mathbb{R}$

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

We denote by L_{\max} the maximum of these Lipschitz coefficients :

$$L_{\max} = \max_{i=1 \dots n} L_i \quad (4)$$

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

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

Remark 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 2.1. Under assumption 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

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

In addition, for $\sigma > 0$, we have

$$\mathbb{E}[f(x_k)] - f^* \leq \left(1 - \frac{\sigma}{nL_{\max}}\right)^k (f(x_0) - f^*) \quad (7)$$

Proof. The proof is quite simple and given in Appendix A.1. \square

Remark 2.2. Our function to be maximize 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.

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, like the price of a stock in an algorithmic trading strategy for financial markets, like the temperature or heart beat of a patient at different time, etc ...). Formally, we can regroup our variables into two sets:

- the first set encompasses $B_1 \dots B_n$. These are called block variables of different length L_i . Mathematically, the Block variables are denoted by B_i with B_i taking value in \mathbb{R}^{L_i} , $\forall i \in 1 \dots n$
- the second set is denoted S and is a block of p single variables.

Graphically, our variables looks like that:

$$\left(\begin{array}{c|c|c} \overbrace{B_1} & & \overbrace{B_n} & & \overbrace{S} \\ \hline B_{1,1} & \dots & B_{1,n} & & B_{n,1} & \dots & B_{n,s} & & S_1 & \dots & S_p \\ \hline \bullet & \dots & \bullet & & \bullet & \dots & \bullet & & \bullet & \dots & \bullet \\ \vdots & & \vdots & \dots & \vdots & & \vdots & & \vdots & & \vdots \\ \bullet & \dots & \bullet & & \bullet & \dots & \bullet & & \bullet & \dots & \bullet \end{array} \right)$$

In addition, 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. We then enter the main loop of the algorithm. Starting with the vector of $(k, \dots, k, \mathbb{1}_p^T)$ 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 between two steps.

We summarize the algorithm in the pseudo code 1. We denote by ε the tolerance for the convergence stopping condition. To control early stop, we use a precision variable denoted by $\varepsilon_1, \varepsilon_2$ and two iteration maximum Iteration \max_1 and Iteration \max_2 that are initialized before starting the algorithm. We also denote $\text{Score}(k_1, \dots, k_n, \mathbb{1}_p)$ 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.

Algorithm 1 OCA algorithm

J Best optimization

We retrieve features importance from a fitted model

We find the index k^* that gives the best score for variables block of same size k :

$$k^* \in \operatorname{argmax}_{k \in \mathbb{R}^{L_{\min}}} \operatorname{Score}(k, \dots, k, \mathbb{1}_p) \quad \triangleright L_{\min} = \min_{i \in \mathbb{R}^n} L_i$$

Initial guess : $x^0 = (k^*, \dots, k^*, \mathbb{1}_p)$

while $|\operatorname{Score}(x^i) - \operatorname{Score}(x^{i-1})| \geq \varepsilon_1$ and $i \leq \text{Iteration max}_1$ **do**

$$x_1^i \in \operatorname{argmax}_{j \in \mathbb{R}^{L_1}} \operatorname{Score}(j, x_2^{i-1}, x_3^{i-1}, \dots, x_n^{i-1}, \mathbb{1}_p)$$

...

$$x_n^i \in \operatorname{argmax}_{j \in \mathbb{R}^{L_n}} \operatorname{Score}(x_1^i, x_2^i, x_3^i, \dots, j, \mathbb{1}_p)$$

$i += 1$

end while

Full coordinate ascent optimization

Use previous solutions: $X^* = (x_1^i, \dots, x_n^i, \mathbb{1}_p)$ $\triangleright i$ is the last index in previous while loop

$$Y^* = \operatorname{Score}(X^*)$$

while $|Y - Y^*| \geq \varepsilon_2$ and iteration $\leq \text{Iteration max}_2$ **do**

for $i=1 \dots N$ **do**

$$X = X^*$$

$$X_i = \text{not}(X_i^*)$$

$$\triangleright \text{not}(0) = 1 \text{ and } \text{not}(1) = 0$$

if $\operatorname{Score}(X) \geq \operatorname{Score}(X^*)$ **then**

$$X^* = X$$

end if

end for

$$Y = \operatorname{Score}(X^*)$$

iteration $+= 1$

end while

Return X^*, Y^*

Remark 3.1. *The originality of this coordinate ascent optimization is to regroup variable by block, 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 3.2. *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 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 $x_{k+1} = x_k - \alpha_k [\nabla f(x_k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$

 Set $k = k + 1$

end while

Remark 3.3. *The specificity of our method is to keep the j best representative features for each feature 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, and Redundancy Elimination as developed in Tuv et al. (2009).*

4 Numerical Results

4.1 Data set

We use this algorithm to do a supervised classification of a data set obtained from financial markets trades that we want to classify according to some a priori features. We are given 1500 trades with 135 features that can be classified into 5 block of 20 variables, 1 block of 30 variables and 5 single variables. We know for each trade whether it is a good or bad 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. For full reproducibility, full data set and corresponding python code for this algorithm is available publicly on github.. The authors may further update the code to reflect improvements or typos if required. This code is provided as it is. The authors do not grant any warranty nor assume any liability for the content thereof.

4.2 Comparison

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% 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%, RFE gets a score of 62.40 %. All these figures are summarized in the table 1.

Table 1: Method Comparison: for each row, we provide in red the best(s) (hottest) 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 feature 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	OCA using 24 features	RFE using 24 features	BCA using 39 features	RFE using 28 features
% of features	16.6	16.6	27.08	19.4
Score (in %)	62.8	62.39	62.19	62.8

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. Below is provided the number of iterations for OCA and BCA in figure 1. Our method requires only 350 iterations steps to converge as opposed to BCA that needs up to 700 iterations steps as it computes blindly variables ignoring similarities between the different variables.

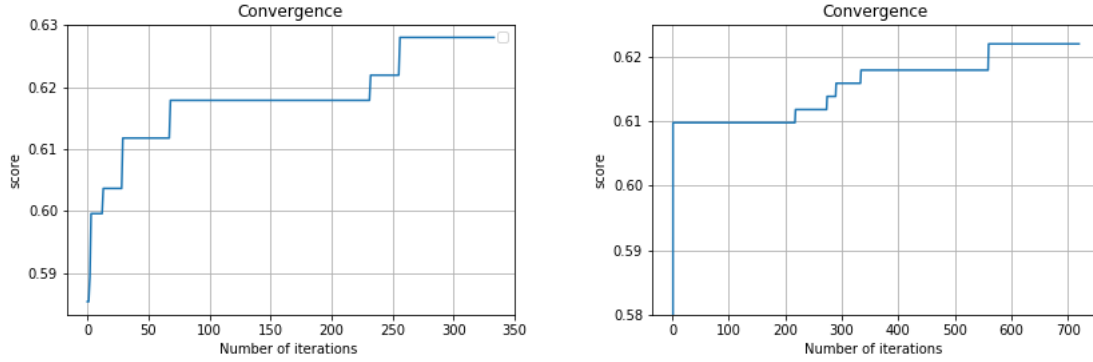


Figure 1: Iterations steps up to convergence for OCA and BCA. OCA method is on the left while BCA is on the right. We see that OCA requires around 350 iteration steps to converge while BCA requires the double around 700 iteration steps to converge

Graphically, we can compute the best candidates for the four methods listed in table 1 in figure 2 and 3. 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 3 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.

Figure 2: Comparison between the 4 methods. 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.

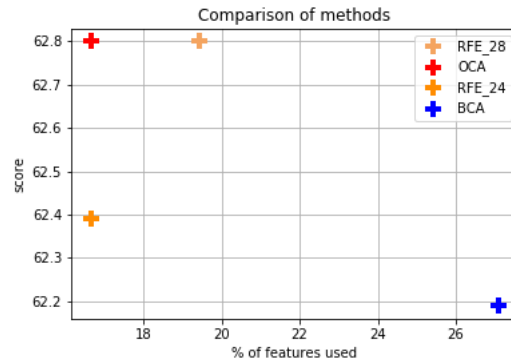
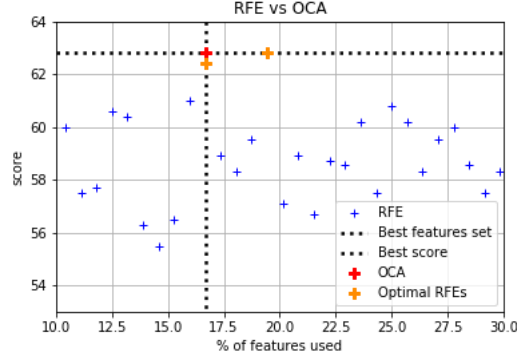


Figure 3: Comparison between OCA and RFE. Zoom on the methods. For RFE, we provide the score for various features set in blue. The two best RFE performers points are the orange cross marker points that are precisely the one listed in table 1. The red cross marker point represents OCA. It achieves the best efficiency as it has the highest score and the smallest feature set for this score.



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). 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 explode 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. OCA provides empirically the most compact data set with optimal performance. Obtaining a reduced features set compared to other method is highly desirable for at least two reasons: First, a lower feature set should have a stronger generalization power as it has less noise created by too many variables (in a similar way in a sense as the Lasso method that eliminates variables in regression). Second, fewer features leads to smaller memory size model and faster computation. Possible extension is to parallelize and potentially use GPU acceleration for this algorithm to leverage its strong decoupling when examining candidate solutions.

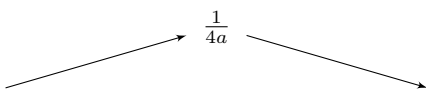
Appendix A Proofs

Appendix A.1 Proof of proposition 2.1

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

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

Proof. We remark that $u_{n+1} \leq u_n - au_n^2$ or equivalently $u_{n+1} \leq u_n (1 - au_n)$, 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 and given below (with $f'(x) = 1 - 2ax$):

x	$-\infty$	$\frac{1}{2a}$	$+\infty$
$f'(x)$	$+$	0	$-$
$f(x)$			

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 our main result. By assumptions, we do a gradient descent according to one coordinate: $x_{k+1} = x_k - \alpha_k [\nabla f(x_k)]_{i_k} e_{i_k}$ for some $\alpha_k \geq 0$. A Taylor-Lagrange expansion for $f(x_{k+1})$'s gradient gives us:

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

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 (A.2), we have the following equality:

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

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

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

Bounding the Hessian from (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:

$$x_{k+1} = x_k - \frac{1}{L_{\max}} [\nabla f(x_k)]_{i_k} e_{i_k} \quad (\text{A.3})$$

Using the fact that $\text{Var}(\|\nabla f(x_k)\|) \geq 0$, we can conclude that

$$\mathbb{E} \left[\|\nabla f(x_k)\|^2 \right] \geq \mathbb{E} [\|\nabla f(x_k)\|]^2 \quad (\text{A.4})$$

By convexity of f , we have

$$f(x_k) - f(x^*) \leq \langle \nabla f(x_k), x_k - x^* \rangle \quad (\text{A.5})$$

$$\leq \|\nabla f(x_k)\| \|x_k - x^*\| \quad (\text{By Cauchy Schwartz}) \quad (\text{A.6})$$

$$\leq \|\nabla f(x_k)\| R_0 \quad (\text{By 5}) \quad (\text{A.7})$$

We denote $U_k = \mathbb{E}[f(x_k)] - f(x^*)$. Taking the expectation over all the random index i_k in A.7, we obtain :

$$U_k \leq \mathbb{E} [\|\nabla f(x_k)\|] R_0 \quad (\text{A.8})$$

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

$$U_k^2 \leq \mathbb{E} [\|\nabla f(x_k)\|]^2 R_0^2 \quad (\text{A.9})$$

$$\leq \mathbb{E} [\|\nabla f(x_k)\|^2] R_0^2 \quad \text{by A.4} \quad (\text{A.10})$$

Using the relation in A.3, we apply Taylor-Lagrange to our objective function f at step k+1 :

$$f(x_{k+1}) = f(x_k) - \frac{1}{L_{\max}} \langle [\nabla f(x_k)]_{i_k} e_{i_k}, \nabla f(x_k) \rangle \quad (\text{A.11})$$

$$+ \frac{1}{2} \left(\frac{1}{L_{\max}} \right)^2 ([\nabla f(x_k)]_{i_k} e_{i_k})^T \mathcal{H}f(d_k) ([\nabla f(x_k)]_{i_k} e_{i_k}) \quad (\text{A.12})$$

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

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{L_{\max}} \left\| [\nabla f(x_k)]_{i_k} \right\|^2 \quad (\text{A.13})$$

$$+ \frac{1}{2} \frac{1}{L_{\max}^2} \left\| [\nabla f(x_k)]_{i_k} \right\|^2 L_{\max} \quad (\text{A.14})$$

$$\leq f(x_k) - \frac{1}{2L_{\max}} \left\| [\nabla f(x_k)]_{i_k} \right\|^2 \quad (\text{A.15})$$

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

$$\mathbb{E} [\mathbb{E}_{i_k} [f(x_{k+1})]] \leq \mathbb{E} \left[\mathbb{E}_{i_k} \left[f(x_k) - \frac{1}{2L_{\max}} \frac{1}{n} \sum_{i=1}^n [\nabla f(x_k)]_i^2 \right] \right] \quad (\text{A.16})$$

$$\mathbb{E} [f(x_{k+1})] \leq \mathbb{E} [f(x_k)] - \frac{1}{2nL_{\max}} \mathbb{E} [\|\nabla f(x_k)\|^2] \quad (\text{A.17})$$

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

$$U_{k+1} \leq U_k - \frac{1}{2nL_{\max}} \mathbb{E} [\|\nabla f(x_k)\|^2] \quad (\text{A.18})$$

Using the inequality A.10, we ensure that :

$$U_{k+1} \leq U_k - \frac{1}{2nL_{\max}} \frac{1}{R_0^2} U_k^2 \quad (\text{A.19})$$

We can conclude using lemma Appendix A.1 to get $U_k \leq \frac{2nL_{\max}}{k}$ so that the first result of proposition 6 holds.

The second result to prove 7 is also easy. Taking the minimum of both sides of 2 leads to $f^* \geq f(x_k) - \frac{1}{2\sigma} |\nabla f(x_k)|^2$ or equivalently $|\nabla f(x_k)|^2 \geq 2\sigma U_k$. Then using A.18, 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. \square

References

- Almuallim, H., Dietterich, T.G., 1994. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence* 69, 279–305.
- Blum, A.L., Langley, P., 1997. Selection of relevant features and examples in machine learning. *Artif. Intell.* 97, 245–271.
- 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.
- 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.
- 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.
- 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.
- Wright, S.J., 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.