Trade Selection with Supervised Learning and OCA

David Saltiel, Eric Benhamou

▶ **To cite this version:**

David Saltiel, Eric Benhamou. Trade Selection with Supervised Learning and OCA. ECML PKDD MIDAS 2021, Sep 2021, Bilbao (online), Spain. ⟨10.1007/978-3-030-66981-2
$_1$⟩.⟨*hal* − 02012476⟩

HAL Id: hal-02012476

https://hal.science/hal-02012476v1

Submitted on 8 Feb 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Trade Selection with Supervised Learning and OCA

David Saltiel [1 2] **Eric Benhamou** [1 3]

Abstract

In recent years, state-of-the-art methods for supervised learning have exploited increasingly gradient boosting techniques, with mainstream efficient implementations such as xgboost or lightgbm. One of the key points in generating proficient methods is Feature Selection (FS). It consists in selecting the right valuable effective features. When facing hundreds of these features, it becomes critical to select best features. While filter and wrappers methods have come to some maturity, embedded methods are truly necessary to find the best features set as they are hybrid methods combining features filtering and wrapping. In this work, we tackle the problem of finding through machine learning best a priori trades from an algorithmic strategy. We derive this new method using coordinate ascent optimization and using block variables. We compare our method to Recursive Feature Elimination (RFE) and Binary Coordinate Ascent (BCA). We show on a real life example the capacity of this method to select good trades a priori. Not only this method outperforms the initial trading strategy as it avoids taking loosing trades, it also surpasses other method, having the smallest feature set and the highest score at the same time. The interest of this method goes beyond this simple trade classification problem as it is a very general method to determine the optimal feature set using some information about features relationship as well as using coordinate ascent optimization.

1. Introduction: a motivating example

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). These algorithmic trading strategies typically relies on historical statistics. The main concept is to find some trading signals and information that identifies pattern or trend with a high probability of repetition. As desirable as it may be, the perfect algorithm is the one with the highest accuracy in terms of identifying the targeted pattern and with the smallest number of losing trades.

If we want to increase robustness and bring additional firewalls to the trading strategy, it makes senses to add supplementary logic with the use of supervised learning method. The question is to empirically validate whether a supervised machine learning method can a priori identify bad or good trade and hence select among the systematic trades spawned by our algorithmic trading strategy. This is a typical supervised learning classification problem, very similar to the boiler plate example of identifying spam in emails. The complexity in this challenge is to identify features that are relevant to assist the machine in being able to in advance determine the chance of success of a machine based trade.

This motivates for efficient method to select among a large set of features the ones that creates an efficient algorithm. This is precisely the subject of this paper. It is organized as follows. We first present the supervised learning classification problem. We then present the Optimal Cordinate Ascent algorithm that enables us selecting the Pareto optimal features set. The key contribution of this method is to exploit similarities between features and hence reduce the optimization search within categories as well as use coordinate ascent to transform the NP hard problem into a polynomial one. We then present results on a real life trading policy. We show that there is substantial improvement compared to the original strategy. We conclude on further work.

[1]A.I. SQUARE CONNECT, 35 Boulevard d'Inkermann, 92200 Neuilly sur Seine, France. [2]LISIC - Universite du Littoral - Cote d'Opale, France. [3]LAMSADE, Universite Paris Dauphine, 75016 Paris, France. Correspondence to: David Saltiel <david.saltiel@aisquareconnect.com>.

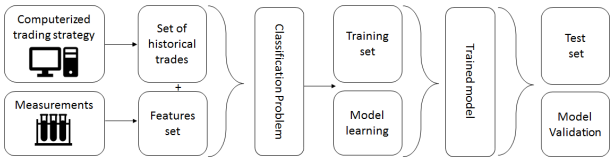

Figure 1. Learnig process for our trade selection challenge. We first use a proprietary trading strategy that generates some samples trades. We take various measures before the trades is executed to create a feature set. We combined these to create a supervised learning classification problem. Using 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.

2. Experience description

2.1. Challenge description

A trading strategy is usually defined with some signal that generates a trading entry. But once we are in position, then next question 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)). The profit target ensures that the strategy locks in real money the profit realized and is materialized by a limit order. The stop loss that is physically generated by a stop order safeguards the overall risk by limiting losses whenever the market backfires and contradicts the presumed pattern. To keep things simple we will hereby examine a trading strategy that has fixed profit target and stop loss. It generates about 1500 trades over a period of 10 years. For each of these trades, we make some measurements to get 135 features. The challenge is from these features to predict which trade is going to be successful. If we give brutally these features to a gradient boosting method like xgboost or lightgbm, the algorithm performs poorly as it is swamped by too many data that are noisy. The features that are provided are proprietary indicators whose identity and source are ignored by our machine learning algorithm. The challenge here is to find the optimal features set for our gradient boosting method. The learning process is summarized by figure 1.

2.2. Feature selection

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 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.

However, these methods do not exploit some particularities of our features set. We are able to 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 ...).

3. OCA Method

The approach adopted here is the method referred to as the Optimal Cordinate Ascent (OCA) method that is described in (Saltiel and Benhamou, 2018). 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:

$$\left(\overbrace{\begin{array}{ccc} B_{1,1} & \dots & B_{1,n} \\ \bullet & \dots & \bullet \\ \vdots & & \vdots \\ \bullet & \dots & \bullet \end{array}}^{B_1} \quad \dots\dots \quad \overbrace{\begin{array}{ccc} B_{1,1} & \dots & B_{1,n} \\ \bullet & \dots & \bullet \\ \vdots & & \vdots \\ \bullet & \dots & \bullet \end{array}}^{B_n} \quad \overbrace{\begin{array}{ccc} S_1 & \dots & S_p \\ \bullet & \dots & \bullet \\ \vdots & & \vdots \\ \bullet & \dots & \bullet \end{array}}^{S} \right)$$

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

3.1. Algorithm description

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 $\left(k, \dots, k,\ \mathbb{1}_p^T\right)$ 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 between two steps.

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:
$k^\star \in \underset{k \in \mathbb{R}^{L_{\min}}}{\text{argmax}}\ \text{Score}\,(k,\ \dots,\ k,\ \mathbb{1}_p)\ \{L_{\min} = \min_{i \in \mathbb{R}^n} L_i\}$
Initial guess : $x^0 = (k^\star, \dots, k^\star, \mathbb{1}_p)$
while $|\text{Score}(x^i) - \text{Score}(x^{i-1})| \geq \varepsilon_1$ and $i \leq$ Iter $\max_1$ **do**
  $x_1^i \in \underset{j \in \mathbb{R}^{L_1}}{\text{argmax}}\ \text{Score}\left(j,\ x_2^{i-1},\ x_3^{i-1},\ \dots\ ,\ x_n^{i-1}, \mathbb{1}_p\right)$
  ...
  $x_n^i \in \underset{j \in \mathbb{R}^{L_n}}{\text{argmax}}\ \text{Score}\left(x_1^i,\ x_2^i,\ x_3^i,\ \dots\ , j, \mathbb{1}_p\right)$
  i += 1
end while

Full coordinate ascent optimization
Use previous solutions: $X^* = (x_1^i, \dots, x_n^i, \mathbb{1}_p)$ {i is the last index in previous while loop}
$Y^* = \text{Score}\,(X^*)$
while $|Y - Y^*| \geq \varepsilon_2$ and iteration $\leq$ Iter $\max_2$ **do**
  for i=1 ... N **do**
    $X = X^*$
    $X_i = \text{not}(X_i^*)$ {not(0) = 1 and not(1) = 0}
    if Score $(X) \geq$ Score (X^*) **then**
      $X^* = X$
    end if
  end for
  $Y = \text{Score}\,(X^*)$
  iteration += 1
end while
Return X^*, Y^*

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 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.

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. *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. Theoretical convergence speed

Although it may be hard to determine the convergence speed for a real life example, under some weak conditions, we can prove that the convergence speed in linear. Hence we changed dramatically the nature of the problem as this method converts an NP hard problem into a polynomial one, making it feasible in a couple of minutes to train our model.

To formalize the concept, let us assume we examine the following optimization program: $\min_x f(x)$. 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 4.1. *We assume our function f is twice differentiable and strongly convex with respect to the Euclidean norm:*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\sigma}{2} \|y - x\|_2^2 \qquad (4.1)$$

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

$$|[\nabla f(x + te_i)]_i - [\nabla f(x)]_i| \leq L_i |t| \qquad (4.2)$$

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

$$L_{max} = \max_{i=1 \dots n} L_i \qquad (4.3)$$

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

$$\max_x \{\|x - x^\star\| : f(x) \leq f(x_0)\} \leq R_0 \qquad (4.4)$$

Remark 4.1. *Strong convexity means that the function is between two parabolas. Condition 4.2 implies that the Gradient's growth is at most linear. Inequality 4.4 States that the function is increasing at infinity.*

Proposition 4.1. *Under assumption 4.1, coordinate ascent optimization (cf. Algorithm 1) converges to the global minimum f^* at a linear rate proportional to $2nL_{\max}R_0^2$, that is*

$$\mathbb{E}[f(x_k)] - f^\star \leq \frac{2nL_{max}R_0^2}{k} \tag{4.5}$$

Proof. See (Saltiel and Benhamou, 2018) appendix A.1 first part of the proof. □

Proposition 4.2. *Under the same condition as proposition 4.1 and with $\sigma > 0$, we have an other convergence rate that decreases exponentially fast as follows:*

$$\mathbb{E}[f(x_k)] - f^\star \leq \left(1 - \frac{\sigma}{nL_{\max}}\right)^k (f(x_0) - f^\star) \tag{4.6}$$

Proof. See (Saltiel and Benhamou, 2018) appendix A.1 second part of the proof. □

Remark 4.2. *in the case of a large σ, the second rate of convergence is much faster than the first one.*

Remark 4.3. *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.*

5. Numerical results

We present herein the result of the machine learning experiment with a real life trading strategy. For full reproducibility, full data set and corresponding python code for this algorithm is available publicly on github with the limitation that sensitive data have been either anonymized or removed (like for instance the final pnl curve).

We first compare our method with two other states of the art methods: Recursive Feature Elimination (RFE) and Binary Coordinate Ascent (BCA) as presented in (Zarshenas and Suzuki, 2016).

Recursive Feature Elimination (RFE) (as presented in (Mangal and Holm, 2018)) 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.

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

Table 1. Method Comparison: 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 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.

Binary Coordinate Ascent (BCA) is an iterative deterministic local optimization method to find Feature subset selection (FSS). The algorithm searches throughout the space of binary coded input variables by iteratively optimizing the objective function in each dimension at a time. Because there is no similarities used in the coordinate ascent optimization, it performs slowly compared to OCA method.

On our test sets, we examine the accuracy score (the percentage of good classification). OCA method achieves the Pareto optimality as it reaches 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 its 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, OCA 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.

It is illuminating to look at the histogram of gain and losses of our trades over our 10 years of history. Not surprisingly, we can observed two peaks corresponding to the profit target and stop loss level as shown in figure 2. This is quite obvious, but it is much better to use the pnl 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 as shown in figure 3.

6. 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 4. Our method requires only 350 iterations steps ton converge as opposed to BCA that needs up to 700 iterations steps as it computes blindly variables ignoring similarities between the different variables.

Graphically, we can compute the best candidates for the four

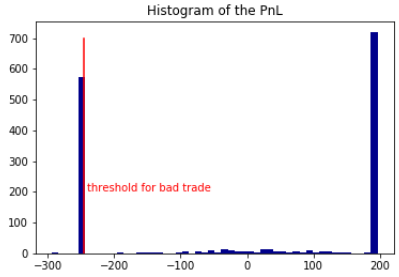

Figure 2. Histogram of the PnL in Dollars amount (re-normalized for anonymity). We can observe two peaks corresponding to the profit target and stop loss levels. This is logical as the trading strategy examined here is a fixed profit target and stop loss strategy. As soon as a trade reaches these levels, the gain or loss is crystallized. If the market stays in trading range and do not reach the level, we have a timeout in the strategy that cut the strategy regardless of its pnl. These cases are rather rare and hence represents very small bars in the histogram.

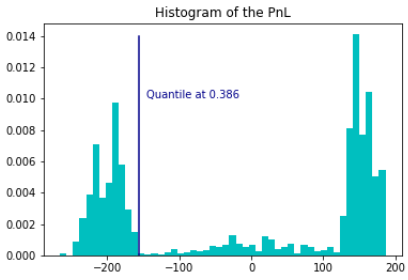

Figure 3. Same Histogram of the PnL but in Euros. Although it may seem very basic, it is important to use the native currency of the algorithmic trading strategy to avoid currency noise. Compared to figure 2, the only difference is to observe the profit and loss not in dollars but in euros as we consolidate all our trading strategies in euros. This is not a good practice as it introduces some additional noise in our labels as the Eur Dollar fx rate randomises slightly the pnl outcome and hence some time out exit may be confused with some bad exits.

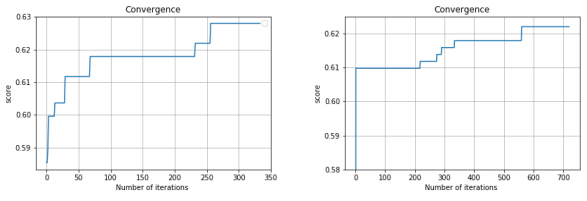

Figure 4. Iterations steps up to convergence for OCA and BCA. OCA method is on the left while BCA is on the right. We see that OCA requires around 350 iteration steps to converge while BCA requires the double around 700 iteration steps to converge

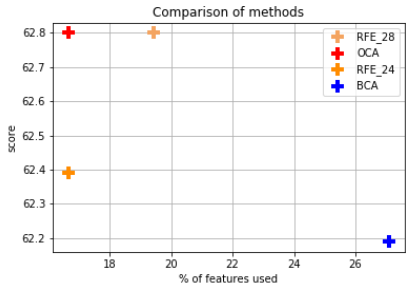

Figure 5. Comparison between the 4 methods. To qualify the best method, it should be in the upper left corner. The desirable feature is to have as little features as possible and the highest score. We can see that the red cross that represents OCA is the best. The color code has been designed to ease readability. Red is the best, orange is a slightly lower performance while blue is the worst.

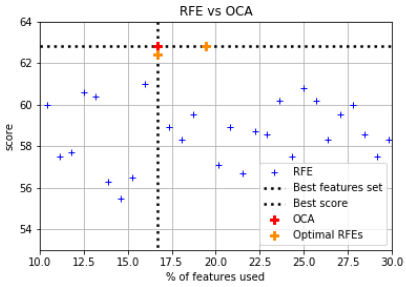

Figure 6. Comparison between OCA and RFE. Zoom on the methods. For RFE, we provide the score for various features set in blue. The two best RFE performers points are the orange cross marker points that are precisely the one listed in table 1. The red cross marker point represents OCA. It achieves the best efficiency as it has the highest score and the smallest feature set for this score.

methods listed in table 1 in figure 5 and 6. 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 6 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.

We then 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 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 7. Since the blue curve that represents the combination of our algorithmic trading strategy and the oca method is above the orange one, we experimentally validate that using machine learning enhances the overall profitability of our trading strategy by avoiding the bad trades.

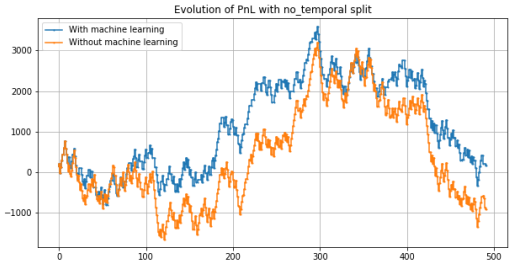

Figure 7. Evolution of the PnL with a randomized test set. The orange curve represents our algorithmic trading strategy without any machine learning filtering while the blue line is the result of the combination of our algorithmic trading strategy and the oca method to train our xgboost method

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. The method of splitting the two sets: training and test set into two sets relies on a temporal split, hence the title of our figure 8.

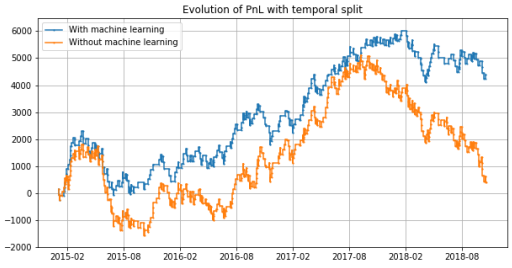

Figure 8. Evolution of the PnL with a test set given by the last third of the data to take into account temporality in our data set. The orange curve represents our algorithmic trading strategy without any machine learning filtering while the blue line is the result of the combination of our algorithmic trading strategy and the OCA method to train our xgboost method

Last but not least, we can zoom the two curves when taking the test set with a temporal split. We clearly see that the method performs well to avoid selecting bad trades and hence the blue line decreases less than the orange one as shown in figure 9.

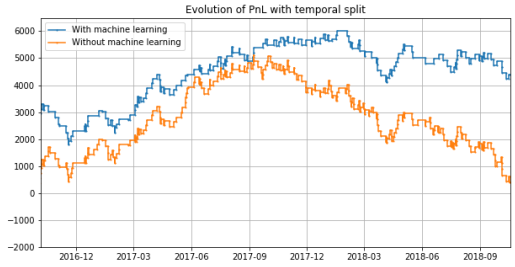

Figure 9. Zoom of the evolution of the PnL with a temporal split.The orange curve represents our algorithmic trading strategy without any machine learning filtering while the blue line is the result of the combination of our algorithmic trading strategy and the OCA method to train our xgboost method

7. Conclusion

Algorithmic trading method can be enhanced with supervised learning method. The challenge is to use measurements and information regrouped into features to detect before orders are electronically sent to the exchange highly probable non successful trades. Because the logic of the algorithmic trading strategy may be challenging to understand, an agnostic supervised learning method can come to the rescue. However, choosing the best features in our initial features set is tricky as more data simultaneously provide additional information and noise at the same time. We present here OCA, a new feature selection method that leverages similarities between features. This method is not very demanding in terms of features knowledge and can efficiently select best features without testing all possible features sets. This changes the features selection problem from an NP hard one into a polynomial one. When implemented on real case strategies, we can empirically validate that the supervised learning method enhances overall trading profitability. As we ask the algorithm to detect in pre-trade operations highly unsuccessful candidates, the method is logically able to reduce overall draw-downs. The method developed herein is quite general and can be applied to any general supervised learning binary classification. In further work, we would like to explore reinforcement learning method to adjust our method for capacity constraints as this is a limitation of the supervised learning approach.

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.

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.

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

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.

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.

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.

Saltiel, D., Benhamou, E., 2018. Feature selection with optimal coordinate ascent (OCA). arXiv e-prints , arXiv:1811.12064arXiv:1811.12064.

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.

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.