An Intrusion Detection Algorithm for Wireless Sensor Networks

Alaa Eissa
Information Systems Dept., Faculty of Computer Science and Information Systems, Mansoura University, Mansoura, Egypt

Samir Zied
Faculty of Computer and Information Sciences, Zagazig University, EGYPT

Abstract
Feature selection is an important topic in data mining, especially for high dimensional datasets. Feature selection (also known as subset selection) The best subset contains the least number of dimensions that most contribute to accuracy; we discard the remaining, unimportant dimensions. This is an important stage of preprocessing and is one of two ways of avoiding the curse of dimensionality (the other is feature extraction). There are two approaches in Feature selection known as Forward selection and backward selection. Feature selection has been an active research area in pattern recognition, statistics, and data mining communities. The main idea of feature selection is to choose a subset of input variables by eliminating features with little or no predictive information. Feature selection is a preprocessing phase in an intrusion detection system in wireless sensor network. As when we make clustering to sensor nodes to discover anomaly we must doing this preprocessing phase to avoid curse of dimensionality problem and this preprocessing phase will reduce complexity of clustering algorithm.

Keyword
Wireless sensor networks

1. Introduction

Wireless Sensor Networks (WSNs) are distributed measurement systems which consist of a large number of nodes deployed over a geographical area [1][2]. Each node is a low-power device that embeds sensing, processing and communication abilities. Acquired data are locally processed and transmitted through the network to a sink for further processing and data interpretation (e.g., a control room). To achieve the performance required by a distributed measurement system (e.g., event detection, monitoring, forecasting), nodes strictly cooperate, with the cooperation among nodes strongly limited by energy, processing and communication constraints.Selecting features in unsupervised learning scenarios is a much harder problem, due to the absence of class labels that would guide the search. Problems of this kind have been rarely studied in the literature, for exceptions see e.g. [3][4].

2. Feature selection

Feature selection Definition: A "feature" or "attribute" or "variable" refers to an aspect of the data. Usually before collecting data, features are specified or chosen. Features can be discrete, continuous, or nominal. Generally, features are characterized as:
1. Relevant: These are features which have an influence on the output and their role can not be assumed by the rest.
2. Irrelevant: Irrelevant features are defined as those features not having any influence on the output, and whose values are generated at random for each example.
3. Redundant: A redundancy exists whenever a feature can take the role of another (perhaps the simplest way to model redundancy). Problem of selecting some subset of a learning algorithms input variables upon which it should focus attention, while ignoring the rest. Feature selection is the process of selecting the best feature among all the features Because all the features are not useful in constructing the clusters; some features may be redundant or irrelevant thus not contributing to the learning process.
This is an important stage of preprocessing and is one of two ways of avoiding the curse of dimensionality (the other is feature extraction). The main aim of feature selection is to determine a minimal feature subset from a problem domain while retaining a suitably high accuracy in representing the original features. In many real world problems Feature selection is a must due to the abundance of noisy, irrelevant or misleading features. For instance, by removing these factors, learning from data techniques can benefit. To be completely sure of the attribute election, we would ideally have to test all the enumerations of attribute subsets, which is infeasible in most cases as it will result in 2n subsets of n attributes.

Advantages of feature selection:

1. It reduces the dimensionality of the feature space, to limit storage requirements and increase algorithm speed;
2. It removes the redundant, irrelevant or noisy data.
3. The immediate effects for data analysis tasks are speeding up the running time of the learning algorithms.

Manuscript received November 5, 2015
Manuscript revised November 20, 2015

4. Reduce computational complexity of running algorithm.
5. Improving the data quality.
6. Increasing the accuracy of the resulting model.
7. Feature set reduction, to save resources in the next round of data collection or during utilization;
8. Performance improvement, to gain in predictive accuracy;
9. Data understanding, to gain knowledge about the process that generated the data or simply visualize the data

Feature selection approaches:

There are two approaches in Feature selection:

1. **Forward Selection**: Start with no variables and add them one by one, at each step adding the one that decreases the error the most, until any further addition does not significantly decrease the error.

2. **Backward Selection**: Start with all the variables and remove them one by one, at each step removing the one that decreases the error the most (or increases it only slightly), until any further removal increases the error

Significantly. To reduce over fitting, the error referred to above is the error on a validation set that is distinct from the training set.

3. Literature Review:

Unsupervised feature selection techniques can be subdivided into two main approaches: clustering and ranking. Clustering techniques aim at maximizing clustering performance according to one (or more) figure of merit. On the contrary, ranking methods aim at selecting a subset of features according to their relevance (and removing redundant or irrelevant features). Both approaches can be considered in WSNs. Clustering techniques provide an explicit relationship among node measurements, i.e., two nodes whose measurements lie in the same cluster have a high affinity in the acquired data. On the other hand, ranking methods provide an implicit relationship among nodes' measurements, i.e., selected features are the most relevant ones, while discarded features are either irrelevant or highly correlated to one (or more) selected features.The choice of the approach depends on the WSN designer's needs. In principle, clustering techniques could be advantageous for routing end energy management while ranking methods may be very useful for distributed decision algorithms. Given that the approach (clustering or ranking) may be a designer's choice, the computational complexity of

The algorithms might be a strong drawback which only rarely can be neglected. here I will provide a comparison between feature selection methods used in wireless sensor networks:

1. **EVOLUTIONARY LOCAL SELECTION ALGORITHMS (ELSA):** ELSA springs from artificial life models of adaptive agents in ecological environments (Menczer and Belew, 1996). In ELSA, an agent may die, reproduce, or neither based on an endogenous energy level that fluctuates via interactions with the environment. The representation of an agent consists of D bits and each of D bits is an indicator as to the corresponding feature is selected or not(1 if a feature is selected, 0 otherwise). Each agent is first initialized with some random solution and an initial reservoir of energy, and competes for a scare resource, energy, based on multi-dimensional fitness and the proximity of other agents in solution space. The mutation operator randomly selects one bit of the agent and flips it. Our commonality-based crossover operator makes the offspring inherit all the common features of the parents. In the selection part of the algorithm, each agent compares its current energy level with a constant reproduction threshold θ. If its energy is higher than θ, the agent reproduces: the agent and its mutated clone that was just evaluated become part of the new population, each with half of the parent's energy. If the energy level of an agent is positive but lower than θ, only the agent itself joins the new population. If an agent runs out of energy, it is killed. The population size is maintained dynamically over iterations and is determined by the carrying capacity of the environment depending on the costs incurred by any action, and the replenishment of resources.

   Advantages and Disadvantages: One of the major advantages of ELSA is its minimal centralized control over agents. By relying on local selection, ELSA minimizes the communication among agents, which makes the algorithm efficient in terms of computational time and scalability. ELSA can be useful for various tasks in which the maintenance of diversity within the population is more important than a speedy convergence to the optimum. Feature selection is one such promising application. Based on the well-covered range of feature vector complexities, ELSA is able to locate most of the Pareto front. However, for problems requiring effective selection pressure, local selection may not be ideal because of its weak selection scheme.

2. **EUCLIDIAN DISTANCE**: Euclidean Distance is the most common use of distance. Euclidean distance or simply 'distance' examines the root of square differences between coordinates of a pair of objects. For each feature Xi calculate Euclidean distance from it to all other features in sample. Euclidean distance d(Xi; Yi) between features Xi and Yi is calculated using the formula:distance(x,y) = {Σi (xi - yi)2 }½ Note that Euclidean (and squared Euclidean) distances are

usually computed from raw data, and not from standardized data.

3. **Advantages and disadvantages**:The distance between any two objects is not affected by the addition of new objects to the analysis, which may be outliers. However, the distances can be greatly affected by differences in scale among the dimensions from which the distances are computed.
4. **FAST CORRELATION BASED FS (FCBF)**: [5]FCBF uses also the symmetrical uncertainty measure. But the search algorithm is very different. It is based on the "predominance" idea. The correlation between an attribute X* and the target Y is predominant if and only if ρy,x*≥ &et X (X ≠X*), ρx,x* < ρy,x* Concretely, a predictor is interesting if its correlation with the target attribute is significant (delta is the parameter which allows to assess this one); there is no other predictor which is more strongly correlated to it.

   Advantages and disadvantages: This approach is very useful when we deal with a dataset containing a very large number of candidate predictors. About the ability to detect the "best" subset of predictors, as we will see in this tutorial.
5. **SEQUENTIAL FORWARD SELECTION (SFS)**: Sequential Forward Selection is the simplest greedy search algorithm. Starting from the empty set, sequentially add the feature x^+ that results in the highest objective function $J(Y_k+x^+)$ when combined with the features Y_k that have already been selected .

   Advantages and disadvantages: SFS performs best when the optimal subset has a small number of features. The main disadvantage of SFS is that it is unable to remove features that become obsolete after the addition of other features.
6. **SEQUENTIAL BACKWARD ELIMINATION (SBE)**: Sequential Backward Elimination works in the opposite direction of SFS. Also referred to as SBS (Sequential Backward Selection). Starting from the full set, sequentially remove the feature x^- that results in the smallest decrease in the value of the objective function $J(Y-x^-)$. Notice that removal of a feature may actually lead to an increase in the objective function $J(Yk-x^-)>J(Yk)$. Such functions are said to be non-monotonic.

   Advantages and disadvantages: SBS works best when the optimal feature subset has a large number of features, since SBS spends most of its time visiting large subsets. The main limitation of SBS is its inability to reevaluate the usefulness of a feature after it has been discarded.
7. **PLUS-L MINUS-R SELECTION (LRS)**: Plus-L Minus-R is a generalization of SFS and SBS. If L>R, LRS starts from the empty set and repeatedly adds 'L' features and removes 'R' features. If L<R, LRS starts from the full set and repeatedly removes 'R' features followed by 'L' feature additions.

   Advantages and disadvantages: LRS attempts to compensate for the weaknesses of SFS and SBS with some backtracking capabilities. Its main limitation is the lack of a theory to help predict the optimal values of L and R.
8. **FEATURE SIMILARITY SELECTION ALGORITHM (FSSA)**: The algorithm proposes a novel unsupervised feature selection technique which exploits feature dependency/similarity to reduce redundancy but does not require searching the feature space. The algorithm starts by clustering the features with a k-means algorithm (the value of k is user-defined) according to the

$$\lambda_2(x,y)=\frac{1}{2}\left(\mathrm{var}(x)+\mathrm{var}(y)-\sqrt{(\mathrm{var}(x)+\mathrm{var}(y))^2-4\,\mathrm{var}(x)\,\mathrm{var}(y)(1-\rho(x,y)^2)}\right)$$

   figure of merit. x and y are feature vectors, var(●) denotes the variance and p (x, y) the correlation between x and y . λ2 , which is the maximal information compression index, is the eigenvalue along the direction normal to the principal component and represents the amount of reconstruction error introduced when the dataset is projected to a reduced space in the best possible way [6]. In other words, λ2 measures the minimum information loss caused by a reduction in the feature number of the dataset. After the clustering phase, the algorithm selects a single feature from each cluster (the other features of the clusters are discarded). The selected features represent the final feature subset.

   Advantages and disadvantages: The proposed algorithm has a very low computational complexity. Moreover, the suggested maximal information compression index is invariant both to the rotation and to the translation of the dataset. On the contrary, it is very sensitive to the scaling transformation as well as by the parameter's choice.
9. **STREAM WISE FEATURE SELECTION:** The Stream Wise Feature Selection Algorithm Considers Candidate Features Which Are Presented Sequentially To The Selection Engine (The Feature Set Does Not Need To Be Available In Advance). For Each Candidate Feature The Algorithm Computes The Following Test [7]: If The Reduction In The Minimum Description Length Provided By The Inclusion Of The Candidate Feature Into The Current Feature Set Is Larger Than A User-Defined Threshold, The Feature Is Selected; Otherwise It Is Discarded. The Algorithm Then Proceeds To The Next Candidate Feature And, If The Candidate Feature Has Been Selected, The Threshold For Adding New Features Is Increased. [8]

Advantages and disadvantages: The proposed algorithm is fast and able to discard features with constant values but is not effective in case of less than 10 features. Moreover, selection of the algorithm's thresholds is a critical issue.

10. **NEURO-FUZZY FEATURE SELECTION (NNFS):** the algorithm relies on the fuzzy feature evaluation index (ffei) which is a figure of merit that measures the similarity between the samples in the original and in the reduced space for a set of transformed features. In particular, ffei decreases when the similarity of two samples belonging to the same cluster increases or the dissimilarity of two samples belonging to different clusters increases. The first intuitive version of this algorithm consists in exploring all possible subsets of features and selecting the one with the lowest value of ffei. It suggests also an ad-hoc neural network (nn) whose objective is to minimize the ffei through unsupervised learning: each pair of samples in the original feature space is presented at the input layer of the nn and the weights of the nn are updated by using a gradient-descent technique aiming at minimizing the ffei. The topping criteria of the learning algorithm are the maximum number of iterations or a minimum value of the feei. At the end of the learning phase, the weights of the nn represent the relevance of the individual features in characterizing/discriminating different clusters.
11. **ADVANTAGES AND DISADVANTAGES:** the proposed algorithm allows us for defining a ranking of the features according to their contribution to the clusters "separability". On the contrary, it does not perform a proper feature clustering. Moreover, the training phase of the nn severely affects the overall computational complexity.

4. The proposed system(Multiple Stream wise Feature Selection (MSFS)

Stream wise feature selection has a natural extension to multiple streams. Each set of features (feature class) is taken to be its own stream, with its own index, keeping track of how many features from that stream have been tested, and its own wealth, measuring how successful the stream has been in producing useful features. Thus, a stream gains or loses wealth based only on how successful it has been in producing beneficial features. The question of which feature to select next for testing is easily resolved: at each iteration, the next feature is taken from the stream with the most permissive threshold, i.e., from the feature class with the highest probability of producing a beneficial feature, i.e. the feature class having maximum wealth.

The Multiple Stream wise Feature Selection (MSFS) algorithm is given in Algorithm 1. It is quite similar to the simple Streamwise feature selection algorithm, but keeps track of wealth and features tested separately for each stream, j. In order to continue to guarantee against overfitting, each of the k different feature streams is only given w0/k initial wealth. The function get new feature() can just get the next feature in the set (the ijth feature in stream j) or, as discussed below, it can dynamically generate new features. As always, we set the constants α Δ and w0 to 0.5. If a fraction $1/(mj - 1)$ of the features are added to stream j, each additional feature changes the wealth wj by $\alpha\Delta /mj - w/2i$, and wj will approach $\alpha\Delta /mj$. When good features are concentrated in one stream, then that stream will soon have the highest wealth, and features will be drawn preferentially from that stream. If there are k (equally sized) streams with all good features in a single stream, then one will only need to consider slightly more than a fraction 1/k of the features. (Very few features in the other streams need to be considered.) The wealth in the good stream approaches $\alpha\Delta/(p/k\ q)$ (where q is the number of true features and p is the total number of features), rather than the value of $\alpha\Delta/(p/q)$ in the single stream setting, thus allowing more features with marginal statistical significance to be retrieved from the good stream. Similar, but less strong benefits occur in the more realistic case where the streams simply contain different densities of good features. Since we can order the features within a stream, streams such as PCA components (arranged from largest to smallest eigenvalue) will quickly be recognized as good. The stream with the original features will learn a threshold that is more stringent than the PCA stream, but much more permissive than the p^2/2 interaction terms, which is likely to be explored last. In the worst case, when the good features are randomly dispersed across the streams, features are drawn sequentially from each of the feature classes (streams), all of which have success at the same rate, and the resulting penalty (remembering that each stream starts out with w0/k of the wealth) looks asymptotically exactly like running a single stream. Having 1/k of the wealth in each stream will reduce the chance of finding features very early in the streams (i.e., among the first few features tried), but the effect of the initial allocation of wealth is rapidly dissipated, and the wealth approaches an asymptote determined by $\alpha\Delta$ and by the fraction of features found to be significant. It is not easy to get comparable benefits from grouping features into classes using batch feature selection algorithms, whether they be stepwise regression or LARS/elastic net. Since all features are considered simultaneously in batch algorithms, one has to "pay the penalty" for looking at all of them when avoiding over fitting. This difficulty can be overcome by applying a different weight to each of the groups of variables, but each of these weights would then

have to be chosen via cross validation–basically requiring searching for a vector of penalties in a space of ℜ^k.

Algorithm 1 MSFS using Alpha-investing

1: **for** j = 1 to k **do**
2: $w_j = w_0/k$; // initial wealth for j-th stream
3: $i_j = 1$; // index of features for j-th stream
4: **end for**
5: model = {}; // initially no features in model
6: **while** features remain **do**
7: // select next stream
8: j = $argmax_j(w_j/i_j)$; // over all streams with remaining features
9: x = get_new_feature(j, i_j); // generate new feature on stream j
10: $\alpha = w_j/2i_j$;
11: // is p-value of new feature below threshold?
12: **if** (get_p_value(x,model) <= α) **then**
13: // accept
14: add_feature(x,model); // add x to the model
15: $w_j := w_j + \alpha\Delta - \alpha$; // increase wealth
16: **else**
17: // otherwise, reject
18: $w_j := w_j - \alpha$; // decrease wealth
19: **end if**
20: $i_j := i_j + 1$;
21: **end while**

Feature Generation and Selection:

The "stream wise" view supports flexible ordering on the generation and testing of features. Features can be generated dynamically based on which features have already been added to the model. All that is required is a method of generating features that does not look at the y values, and an estimation package which given a proposed feature for addition to the model returns a p-value for the corresponding coefficient or, more generally, the change in likelihood of the model resulting from adding the feature. One can also test the same feature more than once by using multiple passes of Stream wise Feature Selection.

New features can be generated in many ways. Each way produces a new feature class for use in MSFS. For example, in addition to the p original features, p^2 pair wise interaction terms can be formed by multiplying all p^2 pairs of features together. In practice, we generate three interaction streams:

1. interactions of features that have already been selected with themselves.
2. interactions of the selected features with the original features.
3. all interactions of the original features.

This requires dynamic generation of the feature stream, since the interaction terms (1) and (2) can not be specified in advance, as they depend on which features have already been selected. The dynamic feature generation and selection schemes, namely (1) and (2) above yield significantly more accurate models on real data sets compared to the approaches which do not use these dynamic interactions.

Interaction terms are one example of a more general class of generated features, including features formed from transformations of the original features (square root, log, etc.), or combinations of them including, for instance, PCA. Such generated features frequently lead to substantially better predictive models, but it is not obvious which of the transformations will be most useful. By putting each into its own stream, and using MSFS, one can try many transformations at relatively little cost. In contrast, in a conventional batch method, one would need to look at all the features in all the streams, at significant computational cost and, worse, at the cost of statistical power of needing to use a larger penalty to control against overfitting. Including separate feature classes for algebraic transformations and PCAs of original features, gives improvement in predictive power as it is evaluated on NIPS 2003 Datasets Next it is tested on two of the NIPS 2003 (Guyon, 2003) datasets, namely arcene (100 observations, 10000 features) and gisette (100 observations,5000 features). These datasets is selected to demonstrate that this approach works well in the case in which there are no naturally occurring feature classes in data. I want to underscore the fact that augmenting the feature set with PCAs and algebraic transformation like "squares" of the features can give improved performance.

Synthetic feature classes is created for these datasets by clustering the features using k-means with Euclidean distance. It turns out that features in each class are similar, but sufficiently different to provide non - redundant signal for prediction of the responses (y's). In addition to this feature classes corresponding to the top 50 PCAs and the "squares" of the features was also added. This gave k + 2 feature classes in all, where k is the number of k-means clusters. The results are shown in Table 1 for the case when k = 1000 clusters. Fig. 1 shows that the accuracy is not a strong function of the number of clusters, however reasonable clusters must exist, e.g. for the NIPS "dexter" dataset most of the points fell in a single cluster, and clustering did not help. This skewed clustering for "dexter" can be explained by the fact that it is sparse hence we did not use it and instead used "arcene" and "gisette" both of which are dense. Due to paucity of space we have not shown the running times of various methods for the NIPS datasets, but the trend is similar to the one for WSD data as the underlying optimization criterion and hence the computational complexities of the various methods remain the same.

In both the NIPS datasets, MSFS (I) and MSFS select features from the feature class corresponding to top 50 PCAs, but do not select any features from "squares"

feature class, which implies that PCA is a good transformation to use as it gives highly predictive features whereas "squares" is not. Generating and testing these extra PCA and "squares" feature classes incur only a minor overhead as the number of total features is only roughly doubled. For both the datasets, MSFS(I) shows the highest accuracy for all cluster sizes followed by Group Lasso/ MKL as is obvious from Fig. 1 (All the results are statistically significant at 5% significance level in a paired t-test).

Table 1: Results on the NIPS datasets (10 Fold CV Classification Errors).Note: The cross validation errors are "proper" test errors and no parameters have been tuned on them.

Method	arcene (k=1000) μ ± σ (#f)	Gisette (k=1000) μ ± σ (#f)
MSFS (I)	10.7±0.8 (5.3)	4.6±0.9 (23.33)
MSFS	12.9±0.4 (4.4)	7.1±0.9 (20)
SFS	16.5±0.8 (4.4)	8.1±0.8 (18)
Stepwise RIC	16.1±0.9 (4.2)	8.3±1.0 (6.33)
Elastic Nets	16.5±0.7 (20.1)	8.3±0.7 (92.67)
Lasso	16.3±1.0 (13.4)	9.1±1.2 (92.33)
GL/MKL	11.8±0.2 (225)	6.2±0.1 (48.97)
Poly. SVM	13.1±0.2 (-)	7.0±0.3 (-)

5. Conclusion

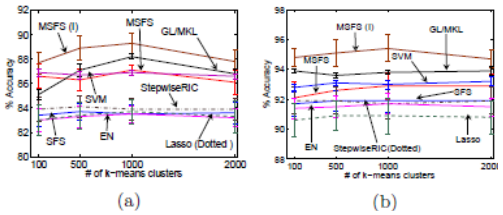

Figure 1: Accuracy as a function of number of clusters for (a) arcene and (b)gisette datasets

We have shown that Feature selection is an important preprocessing phase in an intrusion detection system in wireless sensor network, as it reducing computational complexity of running algorithm. We use the fact that features come in classes can lead to significant gains in predictive accuracy. We introduce literature review of feature selection algorithms used in wireless sensor networks, advantages and disadvantages of each of them, and provide feature selection algorithm that not used before in wireless sensor network that is Multiple Stream wise Feature Selection (MSFS) algorithm that is simple, requiring less than a page of Matlab code, and extremely fast, since each potential feature is considered only once. Moreover MSFS can be extended to include dynamically generated interaction terms. Doing so generally gives significant improvement in performance accuracy over batch methods which would need to include all interaction terms. Also, as demonstrated above, MSFS allows one to test whether new transformations/projections of the features will be of any help or not by incurring only a little overhead on the overall computational cost and, even more importantly, little loss of statistical power to avoid over fitting.

Finally, MSFS is computationally much less expensive than the state of the art "batch" methods. And when features selection algorithm come to faster and more efficient then the process of clustering nodes is also more efficient, and also the process of intrusion detection is more efficient.

References

[1] D. Estrin, L. Girod, G. Pottie and M. Srivastava "Instrumenting the world with wireless sensor networks." In Proceedings of the IEEE Acoustic, Speech, and Signal, vol. 4, pp. 2033–2036 (2001).

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless sensor networks: a survey", Comp. Net., Vol.38, 4, pp 393-422 (2002)

[3] M.H. Law, A.K. Jain, and M.A.T. Figueiredo. Feature selection in mixture-based clustering. In Advances in Neural Information Processing Systems, volume 15, (2003).

[4] A. v.Heydebreck, W. Huber, A. Poustka, and M. Vingron. Identifying splits with clear separation: a new class discovery method for gene expression data. Bioinformatics, 17, (2001).

[5] t. liu, s. liu, z. chen, and w. ma, "an evaluation on feature selection for text clustering," proc. ieee int'l conf. machine learning (icml'03, pp. 488-495, (2003).

[6] P.A. Devijver and J. Kittler, Pattern Recognition: A Statistical Approach. Englewood Cliffs: Prentice Hall, (1982).

[7] H. Ungar, J. Zhou, D. P. Foster, and R. A. Stine. Streaming feature selection using iic. In AI&STAT'05, (2005).

[8] J. Rissanen, " Hypothesis selection and testing by the mdl principle", The Computer Journal, vol. 42, pp. 260-269, (1999).

[9] Bach, F. Consistency of the group lasso and multiple kernel learning. JMLR, 9, 1179–1225, (2008).

[10] Dhillon, P. S., Foster, D., & Ungar, L. Efficient feature selection in the presence of multiple feature classes. ICDM (pp. 779–784), (2008).

[11] E. Krupka, A. Navot, N. T. Learning to select features using their properties. JMLR, 9, 2349–2376, (2008).

[12] Lin, D., Pitler, E., Foster, D. P., & Ungar, L. H. In defense of ℓ 0. ICML Workshop, (2008).

[13] Rakotomamonjy, A., Bach, F., Canu, S., & Grandvalet, Y. Simplemkl. JMLR, 9, 2491–2521, (2008).

[14] J. Zhou, D. P. Foster, R. A. Stine, and L. H. Ungar. Streaming feature selection using alphainvesting. In ACM SIGKDD'05, (2005).

[15] C. Alippi, M. Roveri, "Just-in-time Adaptive Classifiers. Part I. Detecting non-stationary Changes", IEEE Transactions on Neural Networks, Vol. 19, No. 7, pp. 1145 – 1153,(2008).