

Clustering Approach to the Problem of Human Activity Recognition using Motion Data

Szymon Wawrzyniak
 Nicolaus Copernicus
 University
 Toruń, Poland
 Email: wawrzyniak@wp.pl

Wojciech Niemirowicz
 Nicolaus Copernicus
 University and
 Warsaw University
 Warsaw, Poland
 Email: wniemirowicz@gmail.com

Abstract—This paper describes authors’ solution to the task set in *AAIA’15 Data Mining Competition: Tagging Firefighter Activities at a Fire Scene* (<https://knowledgepit.fedcsis.org/contest/view.php?id=106>). Method involves LDA classification on a pre-processed time series data with a unique label transformation technique using K-Means clustering. Data were collected from accelerometer and gyroscope readings.

I. INTRODUCTION

ACTIVITY recognition of a person using motion data aims to label actions and activities. The task has applications in medicine, sports and surveillance, depending on the technology used. One of the most interesting approaches is sensor-based human activity recognition using data collected from accelerometer and gyroscope readings. By applying machine learning methods to the time series data and labels prepared by the experts, it becomes possible to classify a single person’s motion or more general set of motions that can be assigned to a specific group, for instance: firefighters. With the use of a tracking system we can monitor their position and infer potential threats during the action. Such classification task was introduced to the contestants of *AAIA’15 Data Mining Competition: Tagging Firefighter Activities at a Fire Scene* [1].

The organizers provided contestants with data generated by “smart jacket”—a wearable set of body sensors for monitoring kinematics and psychophysical condition of firefighters. The sensors were registering firefighter’s vital functions (i.e. ECG, heart rate, respiration rate, skin temperature) and movement (i.e. seven sets of accelerometers and gyroscopes placed on torso, hands, arms and legs).

Each row represented a window data. The consecutive devices’ data were placed one after another, creating 17242 elements long row. First 42 elements described vital functions and last 17200 elements (43 chunks of 400 readings) - the movement. One chunk stood for the time between a reading and the start of time window. The window data consisted of triaxial readings (from every accelerometer/gyroscope) that were collected during the 1.8s period (every 4.5ms, on average). Figure 1 presents some window data from three accelerometers. The importance of Z axis (for the classification of posture) can be easily noticed (torso). We can observe long right leg motion that started the whole body move, and during it, a short and firm motion of the left leg appeared. It is already

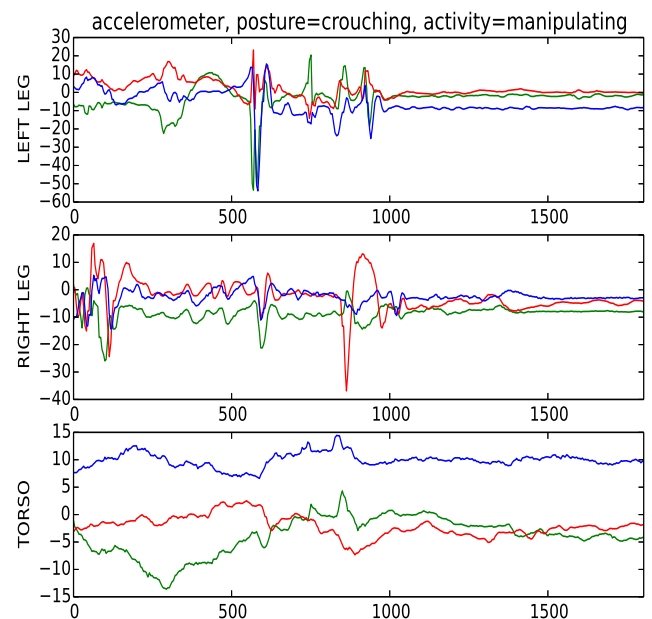


Fig. 1. Triaxial accelerometer readings from different body parts.

hard to link it to any of the postures, but it can be interpreted as three-step move (right leg → left leg → right leg). Torso Z axis (blue) reading points out a motion toward bottom that was disrupted when left leg slowed down. It is probably crouching. Similar analysis can be performed for the activity label, but it seems to be a much more complicated task.

There were three sets of data: one training (20000 rows) and two testing datasets (respectively 2000 and 18000 rows). The first testing dataset was a random selection of observations from a bigger set (20000 rows) and obviously the second one was only the remaining part. All the data were labeled by experts. There were two labels (posture and main activities). The task was to find a classification method that maximizes the *score* value. The contestants had to compute (a solution *s* that consisted of) two vectors of label predictions (a matrix with two *preds* columns). Each one was compared to the proper

vector of label values (from a matrix with two *labels* columns) to calculate the final score in the following manner: for l - total amount of label values, $i \in \{1, \dots, l\}$ (if we replace names with numbers) and $j \in \{1, \dots, 18000\}$ we have

$$ACC_i(preds, labels) = \frac{|\{j : preds_j = labels_j = i\}|}{|\{j : labels_j = i\}|}$$

$$BAC(preds, labels) = \left(\sum_{i=1}^l ACC_i(preds, labels) \right) / l$$

and we denote by:

BAC_p – balanced accuracy for labels describing the posture,
 BAC_a – bal. acc. for labels describing the main activity,

then the final score in the competition for a solution s will be computed as:

$$score(s) = (BAC_p(s) + 2 * BAC_a(s)) / 3.$$

To gain satisfactory performance and reduce the amount of data needed to perform the task, authors proposed the following steps:

- 1) preprocessing - data columns were processed to produce new and less numerous set of columns;
- 2) label transformation - new data, original labels and K-Means clustering method were used to generate new and more numerous set of label values;
- 3) classification - new data and new label(s) provided input for learning. The resulting classifier allowed authors to predict new label values for testing set (testing data had to be transformed in the same way as the training data in the preprocessing stage). In the end these new label values were translated to original label values.

It's important to notice that authors' method requires every new dataset to pass the preprocessing stage, because the classification is performed on the preprocessing's output. Clustering is performed (only once) for the training dataset.

II. RELATED WORK

There are various approaches to the problem of activity recognition, for instance: feature extraction (*FE*) using moving window over time series and subsequent matching. The raw data are converted differently and treated as input for the main processing. Many works describe sensor-based systems, e.g. [2]. Their authors create usually specific set of devices (including accelerometers) to collect data for a model training. The data can be collected from one or more subjects. Using more than one subject can significantly affect the overall accuracy. In paper [4] authors contribute a linear-time method for extracting features from acceleration sensor signals in order to identify human activities, that is based on Support Vector Machines (*SVM*) classifier with a linear kernel. For three types of activities they gained high accuracy (> 92% for each one), when one person's activity was the data source. When the number of subjects was increased by 4, they gained about 10% loss. The amount of 20 was critical for the efficiency in the case of one of these activities (the accuracy decreased to

16.67%). The loss is serious, but it must be considered that if we produce realtime systems, it is expected to prefer simpler but robust systems.

Feature extraction is used to produce new attributes and is usually combined with machine learning algorithms. It is supplied with the data in a form of windows that were collected arbitrarily from the raw dataset. Every window includes readings from devices and is converted to one row usually by setting readings one after another.

Authors of [6] present the whole procedure from receiving the acceleration signal to the final classification. The simplest method is to transform every window to a vector of statistics. More sophisticated methods include Principal Component Analysis (*PCA*), Linear Discriminant Analysis (*LDA*), Fourier Transform or Autoregressive Model. There are various techniques to enrich context awareness by adding environmental variables and analyzing vital signs, but there are no effective techniques to automatically select windows or choose proper windows length. The size of a window is strongly related to the activity and type of extracted features. Authors mentioned windows from 0.08 up to 30 seconds long which they found during their research. It is also needed to reject attributes that contain redundant or irrelevant information and perform Feature Selection (*FS*). The common method is Minimum Redundancy Maximum Relevance [7], which minimizes average mutual information between selected features and maximizes mutual information between classes and features. There are many different algorithms applied to the learning stage, including decision trees, Naive Bayes, Bayesian Networks, *SVM*, Hidden Markov Models, regression methods and k Nearest Neighbours (*kNN*). The most standard evaluation metric is the *accuracy* measure, which is the total fraction of correctly predicted label values.

In subsequent matching we introduce time series set of patterns and use some technique to measure similarity like distance measure (Euclidean, cosine etc.). There are effective ways to handle shifts by using Dynamic Time Warping (as in [2]). Since it is not a statistical approach, the pattern data should be carefully chosen. The choice is much more difficult, when the data are noisy and variations within classes are large.

Some authors include (to the moving window approach) special representation like bag-of-features (BoF) representation [5], which produces local features vectors out of windows smaller than the activity vectors itself, that allow them to create a "motion" vocabulary. The authors even compare their framework to another obtained using subsequent matching approach. Across all vocabulary sizes, the average misclassification error for subsequent matching approach ranges from 37% to 54%. Although the BoF framework performs better for almost every size value, it is more important that it becomes stable in the sense of the misclassification error (for the vocabulary sizes of 50 and more, the variation decreases significantly).

III. CLUSTERING APPROACH TO THE COMPETITION TASK

The basic strategy of finding the best solution was to follow the *score* value changes and apply such method modifications

that brought even small improvement. In general, it was indeed a trial and error approach. It should be emphasized that the training dataset and test dataset were obtained from recordings of different groups of firefighters and that fact could lead to overfitting issues making crossvalidation useless (such situation is mentioned at the end of (III-A)).

After simple preprocessing stage (that is described in next paragraphs) the calculations were performed for different classifiers and three most efficient were left for further research. Those three classifiers used *LDA*, *SVM* and *kNN* algorithm. When authors decided to include label transformation, they left only *LDA* and *SVM* classifiers (and examined the performance of *SVM* classifier with One vs Rest strategy). When there was no *score* improvement some simple modifications of the preprocessing stage were included (by adding more statistical values to the set).

The three-step method is based on the feature extraction approach. Authors had no previous experience and wanted to introduce themselves to this kind of methodology. Preprocessing stage consists of generating statistical values and scaling. Each row is taken and reshaped to a matrix that represents a window data. We count 10 statistics and scale columns separately (by subtracting mean and dividing by standard deviation). We obtain new data and cluster observations to create new labels. The algorithm for creating new labels requires two kinds of data: label values and labels from the clustering. K-Means clustering method was chosen. The algorithm produces vector of new label values. The new training data and the new label values are an input for machine learning algorithms. Before the testing stage it is necessary to preprocess the testing data the same way as described above. When we have trained the classifier and produced new testing data we can perform prediction. The label values that we obtain are the ones that need to be translated to the original values.

A. Preprocessing

Authors decided to choose only movement data for the learning process. Due to time constraints and other obligations there was no opportunity to focus on vital data, which seemed to be much more complicated to analyze.

Each row in dataset represented one observation. There were a total of 42 devices (accelerometers and gyroscopes), which generated simultaneous data for a short period (every row was reshaped to a window that had 42 time series columns). Each time series column consisted of 400 readings.

$$(D_1, \dots, D_{16800}) \rightarrow \left| D_{(i \bmod 400+1)(i \div 400+1)} \right|_{i=1, \dots, 16800}$$

The starting point was to keep all the 42 sources of data, but it had to be preprocessed to lower its dimension. For each device and axis data there were created 10 new attributes (minimum, maximum, mean, standard deviation, first quartile, median, third quartile, interquartile range, skewness and kurtosis). If we introduce some simplification and write that D^{jk} stands for a column of j th device's k th statistic value

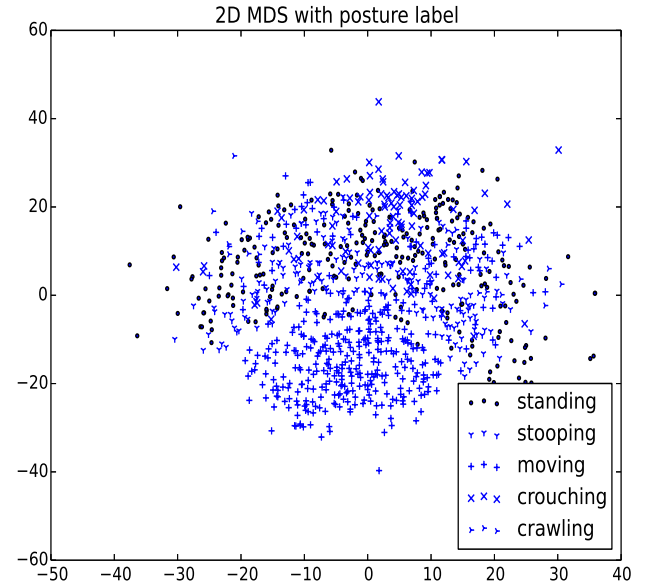


Fig. 2. The result of 2D MDS with posture label.

(for every observation), then we can easily write new dataset as

$$D = (D^{jk}),$$

for $j = 1, \dots, 42$

and $k \in \{max, min, std, q25, med, q75, IQR, skew, kurt\}$.

D consisted of 420 columns, where every D^{jk} was standardized.

When a crossvalidation was performed (using randomly selected halves of the unstandardized data), the crossvalidation *score* values were always greater than 0.9. Using the same classifiers to generate solutions ended up with receiving competition *scores* around 0.2.

When the final classifier set was chosen, a new (ineffective and finally abandoned) strategy was introduced to improve the *score* by adding new quantiles (0.1 and 0.9 quantiles and/or 0.4 and 0.6 quantiles).

B. A premise to label modification

It was an original authors' idea to improve *score* by transforming the given set of label values. Furthermore, a new set of label values had to be generated according to the new data D . The foundation of this procedure lies in assumption that some of label values form too vast data aggregates due to physical differences between firefighters (there was no information about the number). The task would be much more complicated if these aggregates were intermingled. Multi-Dimensional Scaling (MDS) gives some insight into data structure. Figure 2 shows two-dimensional MDS performed on D (1000 observations) plotted with distinction on the first label values.

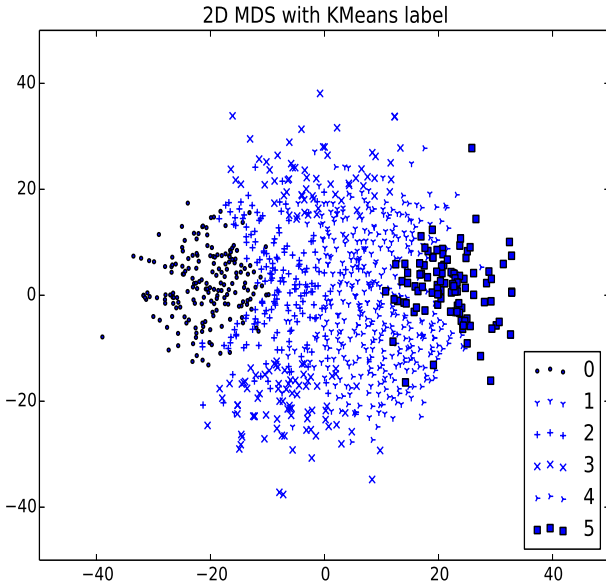


Fig. 3. The result of 2D MDS with K-Means label.

Most of the elements labeled as 'moving' form a structure that is easily separable. On the other hand, if we consider 'standing' posture there is no compact subset that could be separated from other points. It is important to note, that correspondingly larger weights are assigned to smaller sets (for the purposes of *score* evaluation), then any dispersion should be treated as equally serious. Therefore there is strong reason to think that this kind of posture could be difficult to classify. If we label previous data with K-Means (for $k = 6$, Figure 3) we get another graph that strongly indicates existence of apparent groups. It should be understood that the example is only an outline to the idea and does not imply any general statement. Similar proceedings can be made for the main movement label. There would be some clarity issues with plotting (16 values to be marked), so authors decided to leave it. Obviously, the actual calculations for the second label were performed and the resulting graph points to the same conclusions.

C. Label transformation

K-Means clustering was performed on D and it tagged observations with CLV labels. To generate the label for further training we needed original label values LV . Both CLV and LV were vectors of equal length of 20000. We got two mappings

$$D \xrightarrow{K-Means} CLV,$$

$$(LV, CLV) \xrightarrow{algorithm} NLV.$$

D. Algorithm

K-Means algorithm (python's `sklearn.cluster.KMeans` implementation which uses `kmeans++` algorithm for choos-

Algorithm 1 The algorithm for creating a new label

Require: LV (names replaced with numbers), CLV

Ensure: NLV

- 1: Find the length $lenLV$ of LV vector
 - 2: Create NLV vector with the length equal to $lenLV$
 - for** each unique $i \in LV$ **do**
 - 4: Create l_num vector of every observation number that has its LV equal to i
 - Create l_list vector by finding all unique CLV s within the observations with numbers from l_num
 - 6: Create a dictionary $dict$ with keys from l_list , such that for an element $l_list[j]$ assign a value of $(lenLV \cdot j + i)$
 - for** each consecutive k from l_num **do**
 - 8: $NLV[k] \leftarrow dict[CLV[k]]$
 - end for**
 - 10: **end for**
-

ing initial seeds) allowed us to label observations. The second mapping was carried out as shown in Algorithm 1.

To find the best k value authors proposed to seek for a local maximum within a tiny range of small k values (from 2 to 8). There were only 13 test computations at all, because prediction of values can be done separately for every label. We set one k value (for one label and other k was fixed for the remaining label), performed clustering, learning (and classification) and then checked the *score* of our prediction. The maximum *score* (one of seven) indicated k value, which we treated as optimal value and from this moment this k value was fixed. If we found more than one maximum *score*, then we would chose the one with the smallest k value. The *score* maximum for the second label indicated a pair of k values (the one fixed and the new found) that was the optimal choice. Finally, $k = 6$ was selected for the first label and $k = 5$ was selected for the second label.

When the k value was chosen, it was necessary to create and preserve a dictionary to be able to translate every new label value to the original label value. The other way was to propose a method that did employ the clustering to the testing stage, but authors wanted to avoid such requirement. If our machines provided better computing performance we would examine larger range of k values. Then, if the optimum was very high, the computing time would be significantly extended (and the clustering would have to be performed twice, if we included it into testing stage). Due to possible practical applications such effect is not expected.

The probability distribution of the new label seems to be quite obvious to count. If we assume that lv_i stands for i th possible label value and clv_j stands for j th possible cluster level value (derived from K-Means clustering), and we assume that the probability distributions of appropriate labels are known, then the probability of the occurrence of the new label value ($nlv_{i,j}$) produced using those two values can be counted as below

$$P(nlv_{i,j}) = P(clv_j | lv_i) = \frac{P(clv_j, lv_i)}{P(lv_i)}.$$

Furthermore, a big score difference (about 0.2) was observed between the approach that implied use of label transformations as described above and the one that did not (in the favor of the first one).

Another approach is to use only one label with all possible (96) combinations of values. When it was applied to the training labels, there could be found only 24 values. However, the *scores* were so low that in authors' opinion there was no premise to examine it more precisely (for instance by applying K-Means).

E. Classification

The classifier training phase was conducted using newly generated data and vectors of new label values. Then, the trained classifier was used to predict label values for the preprocessed testing dataset. Its output consisted of values that had to be translated to original label values. It had to be an unambiguous transformation, so a new set of label values had to be more numerous than the original one (an equinumerous case is trivial). This assumption was provided if we examine the specifics of the label transformation algorithm. Let T be the test data preprocessed in the same way as D , NLV_{pred} be the vector of predicted labels (which is part of the submitted solution) and respectively NLV_{pred} be the vector of predicted new label values. Generally, we can enclose this stage in three steps

$$\begin{aligned} (D, NLV) &\xrightarrow{\text{learning}} \text{classifier}, \\ T &\xrightarrow{\text{classifier}} NLV_{pred}, \\ NLV_{pred} &\xrightarrow{\text{dictionary}} LV_{pred}. \end{aligned}$$

F. Results

Authors constructed several classifiers (pairs of classifiers, one for every label):

- *LDA* with clustering approach, $k \in \{2, \dots, 8\}$, 13 classifiers, dataset D ;
- *LDA* with clustering approach, $k \in \{2, \dots, 6\}$, 9 classifiers for a dataset obtained in the same manner as D , but with different set of statistics

$$(min1, max1, std1, q25, med1, q75, kurt1, skew);$$

- *LDA* with clustering approach, $k \in \{2, \dots, 6\}$, 9 classifiers for a dataset obtained in the same manner as D , but with different set of statistics

$$(min1, max1, std1, med1, kurt1, skew1);$$

- *LDA* with clustering approach, fixed ks , a dataset obtained in the same manner as D , but with different set of statistics

$$(max, min, std, q10, q25, med, q75, q90, IQR, skew, kurt);$$

- *LDA* with clustering approach, fixed ks , a dataset obtained in the same manner as D , but with different set of statistics

$$(max, min, std, q25, q40, med, q60, q75, IQR, skew, kurt);$$

- *LDA* with clustering approach, fixed ks , a dataset obtained in the same manner as D , but with different set of statistics

$$(max, min, std, q10, q25, q40, med, q60, q75, q90, IQR, skew, kurt);$$

- *LDA* without clustering approach, dataset D ;
- *SVM* (kernels: *rbf*, *linear*) for the first label, *LDA* for the second label, both with clustering approach, $k \in \{3, 4, 5\}$, k for the second label was fixed, 6 classifiers, dataset D ;
- *SVM* (kernels: *rbf*, *linear*) with One vs Rest strategy for the first label, *LDA* for the second label, both with clustering approach, $k \in \{3, 4, 5\}$, k for the second label was fixed, 6 classifiers, dataset D ;
- *kNN* without clustering approach, (number of neighbours) $k \in \{5, 7, 10\}$, 5 classifiers, dataset D ;

To sum up the overall classification performance, *LDA* classifier that uses approach described in part (III-D) with $k = 6$ for the first label and $k = 5$ for the second label performed best among other classifiers.

The *score* for the less numerous testing set was 0.8067 and for the remaining set of observations was 0.77288289.

IV. CONCLUSION

When the *SVM* classifier with One vs Rest strategy (for the first label) and *LDA* classifier (for the second label) were combined, they produced only slightly worse *scores* (about 0.79) than the best one (0.8067), assuming that predictions for the second label remained unchanged. An interesting observation was made when the vector of predictions for the first label was compared to the corresponding output of the final solution. Only about half of label value predictions were the same, which probably indicates a problem with a proper classification of the first label.

Taking into account simplicity and significant improvements in evaluation of the model due to application of K-Means approach and small difference between first and second stage scores, the authors' method can be treated as suitable for the activity recognition task. Furthermore, the model behaved stable (for different ks) and performed very efficiently in terms of learning time.

The authors are also aware that the results and performance can be dependant on implementation issues, so it should be noticed that everything was prepared in Python with the use of *sklearn* library (standarization, clustering, classifiers) [3].

REFERENCES

- [1] Michał Meina and Andrzej Janusz and Krzysztof Rykaczewski and Dominik Ślęzak and Bartosz Celmer and Adam Krasuski, *Tagging Firefighter Activities at the Emergency Scene: Summary of AAIA'15 Data Mining Competition at Knowledge Pit*, Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, editors: Maria Ganzha and Leszek A. Maciaszek and Marcin Paprzycki, In print September 2015
- [2] Michał Meina and Bartosz Celmer and Krzysztof Rykaczewski, *Towards Robust Framework for On-line Human Activity Reporting Using Accelerometer Readings*, Proceedings of the Active Media Technology - 10th International Conference, Warsaw, 2014, pp. 347-358, http://dx.doi.org/10.1007/978-3-319-09912-5_29

- [3] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E., *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, No. 12, 2011, pp. 2825-2830
- [4] M. Khan, S. I. Ahamed, M. Rahman, and R. O. Smith, *A feature extraction method for real time human activity recognition on cell phones*, isQoLT, 2011. <http://www.mridulkhan.com/pdf/khan-QoL10.pdf>
- [5] Mi Zhang and Alexander A. Sawchuk, *Motion primitive-based human activity recognition using a bag-of-features approach*, Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium (IHI '12), ACM, New York, pp. 631-640. <http://doi.acm.org/10.1145/2110363.2110433>
- [6] Oscar D. Lara, Miguel A. Labrador, *A Survey on Human Activity Recognition using Wearable Sensors*, Communications Surveys & Tutorials, IEEE, vol. 15, No. 3, 2013, pp. 1192-1209, doi:10.1109/surv.2012.110112.00192
- [7] Hanchuan Peng (Member, IEEE), Fuhui Long, and Chris Ding, *Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, No. 8, 2005, <http://doi.ieeecomputersociety.org/10.1109/TPAMI.2005.159>