A SIMPLE PROCESS TO SPEED UP MACHINE LEARNING METHODS: APPLICATION TO HIDDEN MARKOV MODELS

Khadoudja Ghanem

MISC Laboratory, Mentouri University , Rue Ain El Bey, Constantine, Algeria ;
gkhadoudja@yahoo.fr

ABSTRACT

An intrinsic problem of classifiers based on machine learning (ML) methods is that their learning time grows as the size and complexity of the training dataset increases. For this reason, it is important to have efficient computational methods and algorithms that can be applied on large datasets, such that it is still possible to complete the machine learning tasks in reasonable time. In this context, we present in this paper a simple process to speed up ML methods. An unsupervised clustering algorithm is combined with Expectation, Maximization (EM) algorithm to develop an efficient Hidden Markov Model (HMM) training. The idea of the proposed process consists of two steps. The first one involves a preprocessing step to reduce the number of training instances with any information loss. In this step, training instances with similar inputs are clustered and a weight factor which represents the frequency of these instances is assigned to each representative cluster. In the second step, all formulas in the classical HMM training algorithm (EM) associated with the number of training instances are modified to include the weight factor in appropriate terms. This process significantly accelerates HMM training while maintaining the same initial, transition and emission probabilities matrixes as those obtained with the classical HMM training algorithm. Accordingly, the classification accuracy is preserved. Depending on the size of the training set, speedups of up to 2200 times is possible when the size is about 100.000 instances. The proposed approach is not limited to training HMMs, but it can be employed for a large variety of MLs methods.

KEYWORDS

Machine learning methods, HMM, EM, clustering, time processing.

1. INTRODUCTION

Machine learning (ML) methods are a set of "programming by example" methods, they are data driven, and are able to examine large amounts of data. The goal of these methods is to build high-quality predictive models in order to understand the intrinsic difficulty of a given learning problem and explain observed phenomena in actual experiments [1].

The main problem of classifiers based on ML methods is that their learning time grows as the size and complexity of the training dataset increases because training with only a few data will lead to

Sundarapandian et al. (Eds): ICAITA, SAI, SEAS, CDKP, CMCA, CS & IT 08,
pp. 161–171, 2012. © CS & IT-CSCP 2012 DOI : 10.5121/csit.2012.2514

an unreliable performance. Indeed, due to advances in technology, the size and dimensionality of data sets used in machine learning tasks have grown very large and continue to grow by the day. For this reason, it is important to have efficient computational methods and algorithms that can be applied on very large data sets, such that it is still possible to complete the machine learning tasks in reasonable time.

The speeding up of the estimation process of the built model parameters is a common problem of ML methods, although it raises several challenges. Many methods have been employed to overcome this problem. These methods can be classified into two categories: the first category consists of methods which compress data, either the number of instances or the set of features (attributes) which characterize instances. The second category consists of methods which reduce running time in the execution and compilation level.

In the first category, methods attempt to reduce the learning time by preprocessing the training data by reducing instance set [2], using random undersampling and vector quantization (VQ)[3, 4, 5], but the main problem with these techniques is the loss of information about the input data. In [6], authors presented a method for speeding up dynamic program algorithms used for solving HMM decoding and training. In [7], authors combine K-means clustering and SVM to speed up the real-time learning. In[8, 9, 10, 11], authors discussed the combination between VQ and SVM. Recently, selective sampling or active learning has been used to reduce the number of training data. Instance selection with very large datasets is an optimization problem that attempts to maintain the mining quality while minimizing the sample size [12]. It reduces data and enables a data mining algorithm to function and work effectively with very large datasets. There is a variety of procedures for sampling instances from a large dataset [13].

Active learning optimizes learning from a minimum number of instances. Examples of SVM training based on active learning include probabilistic active support vector learning algorithm [14] and confident-based active learning [15].

In another hand, to reduce the number of features (attributes), many selection methods were tested. Feature subset selection is the process of identifying and removing as many irrelevant and redundant features as possible [16,20,21]. This reduces the dimensionality of the data and enables machine learning methods to operate faster and more effectively. The fact that many features depend on one another often unduly influences the accuracy of supervised ML classification models. This problem can be addressed by constructing new features from the basic feature set [17]. These newly generated features may lead to the creation of more concise and accurate classifiers. Moreover, the presence of irrelevant features can make many ML methods like Neural Network training very inefficient, even impractical [16,18,19].

In the second category, methods attempt to reduce time training by reducing time compilation and by using parallel computation.

In [22], authors presented a new approach to reduce the training time of a machine learning based compiler. This is achieved by focusing on the programs which best characterize the optimization space by using unsupervised clustering in the program feature space. Furthermore, they are able to learn a model which dispenses with iterative search completely allowing integration within the normal program development cycle. They evaluated their clustering approach on the EEMBCv2 benchmark suite and show that they can reduce the number of training runs by more than a factor

of 7. This translates into an average 1.14 speedup across the benchmark suite compared to the default highest optimization level.

In [16] authors discussed the possibility of a parallel implementation of Viterbi algorithm and Baum & walch algorithm algorithms.

In [23] authors argue that modern graphics processors far surpass the computational capabilities of multicore CPUs, and have the potential to revolutionize the applicability of deep unsupervised learning methods. They develop general principles for massively parallelizing unsupervised learning tasks using graphics processors.

This paper focuses on reducing ML methods time training. It presents a simple reduction instance-based learning technique without eliminating any of these instances and without referring to cloud computing, grid computing or parallel computing. Given a set of training data attached to each of two or more than two classes, an automatic analysis of behavior based on a simple clustering of training instances is proposed. Each representative cluster is weighted with the number of similar instances. The classical ML training algorithm is modified consequently by taking in consideration the new set of training instances with the associated weight. The proposed process allows reducing time computing.

The rest of this paper is organized as follows. The next section briefly reviews the HMM, as it represents the first ML method on which we apply our proposed technique. Section 3 describes the proposed technique for reduced time learning algorithm. Obtained results are discussed in section 4. Section 5 discusses the possibility of an extended usability of the proposed technique to other ML methods. Finally, section 6 concludes the paper.

2. HIDDEN MARKOV MODELS[24]

A Hidden Markov Model is a model for sequential data. It is a doubly embedded stochastic process with an underlying stochastic process that is not observable (hidden) , but can only be observed through another set of stochastic process that produce the sequence of observations. The performance of a generative model like the HMM depends heavily on the availability of an adequate amount of representative training data to estimate its parameters, and in some cases its topology.

2.1 Elements of an HMM:

An HMM is characterized by the following:

1- N : the number of hidden states in the model. For many practical applications there is often some physical significance attached to the states or to the sets of states of the model. We denote the individual state as S= {S1, S2…Sn} and the state at time t as qt.

2- M : the number of distinct observation symbols. The observation symbols correspond to the physical output of the system being modeled. We denote the individual symbols as V={V1, V2…VM}

3- The state transition probability distribution A={aij}where: aij=P[qt+1=Sj|qt=Si] 1<=i, j<=N.

4- The observation symbol probability distribution in state j, B={bj(k)}, where:
Bj(k)=P[Vk at t|qt=Sj] 1<=j<=N; 1<=k<=M.

5- The initial state distribution π ={ πi} where π =P[q1=Si] 1<=i<=N.

Given appropriate values of N, M, A, B and π the HMM can be used as generator to give an observation sequence : O= O1O2….OT.
The compact notation: λ =(A,B,π) is used to indicate the complete parameter set of the model.

2.2 The Three Basic Problems for HMMs:

Given the form of HMM there are three basic problems of interest that must be solved for the model to be useful in real world applications. These problems are the following:

1- Given the observation sequence $O=O_1O_2\ldots O_T$, and a model λ =(A,B,π) how do we efficiently compute P(O/λ) , the probability of the observation sequence given the model?
2- Given the observation sequence $O=O_1O_2\ldots O_T$, and a model λ =(A,B,π) how do we choose a corresponding state sequence $Q=q_1q_2\ldots q_T$ which is optimal in some meaningful sense .
3- How do we adjust the model parameters λ=(A,B,π) to maximize P(O/λ)?

Problem three is the crucial one for most applications of the HMMs, since it allows us to optimally adapt model parameters to observed training data to create best models for real phenomena. The iterative procedure like Baum-Welch method or equivalently the EM (Expectation, maximization) [25] method is used to solve this problem. One associated problem to this solution is the time taken by the training process. In this paper, we aim to reduce training time by reducing the original training set via an unsupervised clustering. Any information loss during clustering is performed, we introduce a weight factor that is assigned to each training instance which represent each cluster, and modify all expectation formulas to include the weight factor in appropriate terms. The weight factor is the “frequency” of observing an instance with similar behavior (characteristics) in the training set. Consequently the obtained model parameters: A, B, π are equal to those obtained with the classical HMM training algorithm but in less time.

3. THE PROPOSED PROCESS

In many real world classification problems there is significant data redundancy in training data, consequently, the time and memory complexity grows with sequence length, the number of training sequences, and with the number of HMM states N. For such problems one might be able to reduce the learning time by preprocessing the data.

Training HMM means optimizing the model parameters (A,B,π) to maximize the probability of the observation sequence P(O|λ). In order to obtain a good estimation of HMM; a big amount of computation and a large database is required.

HMM profiles are usually trained with a set of sequences that are known to belong to a single category. Consequently this set of sequences is often redundant in the sense that most of these sequences have the same values taken by the different features describing this category.

Many real world applications are sensitive to these phenomena as instance, in medicine, and for many patients reached with the same disease can have the same symptoms associated with this disease. In gender recognition or skin recognition, many subjects can have same characteristics (color, texture…). In facial expression recognition, many subjects display the same description of any facial expression (e.g Joy : the distance between mouth corners increases, the distance between lips relaxes or increases and the distance between lids decreases or relaxes) [26].
In image analysis, in many times, the extracted wavelets or FFT coefficients from some image parts are the same when the images are belonging to the same category, etc…

Our aim is to reduce the size of the training set, so, the time of the training process without reducing generalization (classification) accuracy. In this context, our approach involves a preprocessing step to reduce the number of training samples, so we do not require that all data to be resident in main memory (minimize storage and time training requirements). To this end, we apply an unsupervised clustering on the original training data set to extract representative clusters. To preserve training data from any information loss, we associate a weight factor to each cluster. This weight factor is the frequency of observing instances with similar behavior (characteristics). This approach is suitable for many real world applications where there is significant data redundancy. After that, we modify all expectation formulas computed with the classical HMM learning method (EM) to include the weight factor in appropriate terms.

The proposed approach reduces significantly the run-time of HMM training and the storage requirement, while providing accurate discovery and discrimination of different category behaviors when applying the unsupervised clustering algorithm.

3-1 Clustering Process

Given an original training set, we apply an unsupervised clustering on all training instances that belong to a particular class. That is, unsupervised clustering is performed independently on each class. There are numerous clustering techniques including the K-means, fuzzy C-means, hierarchical clustering, and self-organizing maps. The common problem with most of these methods is the choice of the number of clusters or the number of neighbors. In our approach we propose a simpler clustering algorithm where, it is important to note that the number of clusters is not known a priori.

Clustering algorithm:

Begin
N: is the number of categories;
ni: is the number of instances (sequence observation) associated to category i; i=1..N
$C_k(i)$: cluster number k of category i; 1<=k<=ni; j=sequence length
&
$C_k(i,j+1)$= frequency of sequence observation
(instance with same features values)

For all categories(N)
  Put the first instance from the training set of the category i in table C of clusters;
  Initialize the frequency field by 1;
  For all instances belonging to each category(ni)
    Compare the new instance(inst_nouv) with all instances (inst_in) in the cluster table;
    If dist(Inst_nouv,inst_in)=0
      Update frequency field(+1);

```
        Else
             Add a new cluster with the new instance;
             Initialize the frequency field with 1;
        End
   End
End
End.
```

The Euclidian distance is used to compare the new sequence and each sequence in the cluster table. As we are interested only by similar sequences (distance equal to 0), we have not to store these distances which means no space requirement. Also the clustering process is not time-consuming compared to time training.

Other known distances can be used instead of the Euclidian one in the case where no preparation of training instances is performed (e.g see section 4.1).

Once clustering is completed, we use each instance in the cluster table as a new training instance. Its weight factor can be found in the associated frequency field.

In many studies, clusters with small weight are considered as noisy instances so they are removed. It is not the case in all studies, in our experiments, each cluster describe a specific behavior associated to the considered class (e.g Anger or disgust are two universal facial expressions which can be displayed in different ways, each cluster of each class (facial expression) can be a specific description of each facial expression so it is not a noisy instance), in this case only a human expert can remove these clusters to improve classification accuracy.

3.2. The Modified EM Algorithm for HMM Training

Generally, the training of Hidden Markov Models is done by the EM Algorithm to create the best fitting HMM from a given set of sequences F.

The EM algorithm for HMMs

- Initialization : set initial parameters λ^0 to some value;
- For t = 1..num iterations
    - Use the forward_backward algorithm to compute all expected counts :
      Number of transitions from state S_i to S_j using the precedent set of parameters λ^{t-1}
      Number of emissions of symbol V_k from state S_j using the precedent set of parameters λ^{t-1}
      Update the parameters based on the expected counts using formulas (2) and (3).

End.

From the initial HMM M and the initial parameter set λ we can generate a new optimized parameter set λ'. Using the concept of counting event occurrences, the following formulas are given for re_estimation of the new HMM parameters:

- π_i'= The expected frequency in state S_i at time t=1 **(1)** ;
- a_{ij}'= the expected number of state S_i to state S_j transitions/ the expected number of transitions from state S_i **(2);**
- $b_j(k)$'= The expected number of times the observation V_k is generated from state S_j **(3);**

These computations are performed for all training instances, in our case, these computations are performed for the new set of training instances, it consists on the clustered training set instances so they are iterated for m=1..k_i (k_i is the number of clusters of class i), multiplying each formula by $C_k(i,j+1)$ (number of similar instances);

As a result the transition probability from state S_i to state S_j and the emission probability of the observation V_k from state S_j increase with the number of instances (redundant and non redundant). More than that, time running in the computation of all these expectations and for all iterations is significantly reduced.

4. EXPERIMENTAL RESULTS

The experiments were conducted on both (i) extracted information from (MMI database[27]) videos and (ii) synthetically generated instances. Our aim is to analyze the capability of the proposed method in reducing training set instances without information loss and reducing training time. Our code source is written in Matlab, and, our experiments are done on: Intel Dual-Core 1.47GHz machine with 2G memory.

4.1. MMI database

The MMI database is a resource for Action unit (AU) and basic emotion recognition from face video. It consists of V parts; each part is recorded in different conditions. The second set of data recorded was posed displays of the six basic facial expressions: Joy, Sadness, Anger, Disgust, Fear and Surprise. In total, 238 videos of 28 subjects were recorded. All expressions were recorded twice. In our experiments [28], we aim to discover novel clusters with different behavior relative to each class (generate for each facial expression a set of clusters which allow describing all possible behaviors associated with each studied class). To this end, facial characteristic points were detected on the first frame of each video and tracked, then, characteristic distances were computed leading to several time series. Obtained time series were analyzed to produce a smaller set of time series which represent occurrence order of facial features deformations of each subject with each facial expression (This is what we call data preparation).

4.2. Synthetic Time Series

To generate synthetic data, the Markov Model with its transition and emission probabilities matrixes constructed with the MMI database was used. Accordingly, for a specific alphabet size M=10 observations, several training sequences were produced.

With both data, we evaluated our experiments on an HMM with three states(3), sequence length equal to five (5) from where we have extracted the most interesting subsequences which length is three (3), the number of possible observations is ten (10) and the number of iterations is fifty (50). The estimated time training of the EM and EM based cluster training algorithms are presented on table 1 according to each experiment:

Table 1. Time training of EM and EM based cluster training algorithms.

Number of sequence observations	Number of clusters	Time of clustering /s	Time of EM training/s	Time of EM based cluster training/s	DIFF in number of times
100	10	0	0.5304	0.0936	**5.6**
1.000	19	0.0312	5.234	0.162	**32.3**
10.000	24	0.2652	44.3979	0.1945	**228.26**
100.000	25	4.6020	497.4092=8mn29s	0.2106	**2361.87**

Results on this table are computed as the mean of ten (10) executions of the two algorithms. In term of processing speed, the EM training takes 0.5304 seconds to learn 100 training sequences. With the EM based cluster training, it takes 0.0936 seconds, it means that it is 5.6 times faster than EM training, knowing that the number of produced clusters is equal to 10 and time clustering is equal to 0 seconds.

When the number of training sequences is equal to 1.000, the EM based cluster training is 32.3 times faster than the EM training. When the number of training sequences is equal to 10.000, the EM based cluster training is 228.26 times faster than the EM training. Finally, when the number of training sequences is equal to 100.000, the EM based cluster training is 2361.87 times faster than the EM training. We note also that time clustering is negligible compared to time training.
The experiments show that the difference in time training between the two algorithms increases dramatically with the number of training sequences in favor of the proposed modified EM training algorithm (it reduces). These results suggest that the proposed method can be used especially with very large datasets.

Obtained results show that the proposed process reduces not only time computation requirements, but, it also reduces significantly storage requirements. Because HMM profiles are usually trained with a set of sequences that are known to belong to a single category, the clustering process produce a small number of clusters. These clusters will represent the new set of training instances. Experimental results show that both EM and EM based cluster training algorithms achieve similar: Initial (π), Transition (A) and Emission (B) probabilities matrixes. Consequently the classification accuracy is maintained.

5. GENERALIZATION TO OTHER ML METHODS

Obtained results in section 4 show that the proposed idea is well suited with very large databases with redundant data. In order to generalize it, we study the possibility of an extended usability of the proposed technique to other ML methods. Neural network and SVM are two models where a large sample size is required in order to achieve its maximum prediction accuracy.

5.1 Multilayered Perceptron

Back Propagation (BP) algorithm [29] is the most well-known and widely used learning algorithm to estimate the values of the weights used when training a network. Generally, BP algorithm includes the following six steps:

1. Present a training sample to the neural network.

2. Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.

3. For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is the local error.

4. Adjust the weights of each neuron to lower the local error.

5. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.

6. Repeat the steps above on the neurons at the previous level, using each one's "blame" as its error.

The back propagation algorithm will have to perform a number of weight modifications before it reaches a good weight configuration. For n training instances and W weights, each repetition/epoch in the learning process takes O(nW) time; but in the worst case, the number of epochs can be exponential to the number of inputs [30]. The greatest problem with the BP training algorithm or its variants is that they are too slow for most applications. Many approaches were proposed to speed up the training [30].

When applying the proposed process, training instances with similar inputs are clustered in one cluster and each cluster is weighted with the number of similar instances. After that, all formulas associated with the number of training instances are modified to include the weight factor in appropriate terms. Consequently, a new set of training instances is produced, so, each repetition/epoch takes O(n'W) time where :

$$O(n'W) <= O(nW) \quad ; \quad n' <= n : n' = \text{number of training set clusters.}$$

5.2 Support Vector Machines

Support Vector Machines (SVMs) is another supervised machine learning technique (Vapnik, 1995). Generally, SVMs algorithm includes the following steps:

1) Introduce positive Lagrange multipliers. This gives Lagrangian:

$$Lp = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha i yi(xiw - b) + \sum_{i=1}^{N} \alpha i$$

2) Minimize L_P with respect to w, b. This is a convex quadratic programming(QP)problem.

3) In the solution, those points for which $\alpha i > 0$are called "support vectors"

Training the SVM is done by solving N^{th} dimensional QP problem, where N is the number of instances in the training dataset. Solving this problem in standard QP methods involves large matrix operations, as well as time-consuming numerical computations, and is mostly very slow and impractical for large problems [30].

When applying the proposed process, training instances with similar inputs are clustered in one cluster and each cluster is weighted with the number of similar instances. After that, all formulas associated with the number of training instances are modified to include the weight factor in appropriate terms. Consequently, a new set of training instances is produced, so, training the SVM is done by solving N'^{th} dimensional QP problem, where N' is the number of clusters formed after clustering the training instances where : N'<=N.

In general, the proposed idea can be adapted to all ML methods which involve a number of iterations equal to the number of instances in their processing. The proposed process reduces the number of iterations and consequently reduces the time training of these ML methods.

6. CONCLUSION

In this paper, a simple process to speed up ML methods is studied. An unsupervised clustering algorithm is combined with EM algorithm to develop an efficient HMM training to tackle the problem of HMM time training of large datasets. The idea of the proposed process consists of two steps. In the first step, training instances with similar inputs are clustered in one cluster and each cluster is weighted with the number of similar instances. In the second step, all formulas associated with the number of training instances are modified to include the weight factor in appropriate terms. Through empirical experiments, we demonstrated that the EM based cluster training algorithm reduces significantly the HMM time training and storage requirements. Furthermore, the proposed approach is not limited to HMMs it can be employed for a large variety of ML methods. In the future, we plan to explore new methods of reducing time training without any information loss and test these methods on different ML methods.

REFERENCES

[1] De Mantaras and Armengol E., (1998) "Machine learning from examples: Inductive and Lazy methods". Data & Knowledge Engineering. 25, 99-123.

[2] Wilson D.R. and Martinez T.R., (2000) "Reduction Techniques for Instance-Based Learning Algorithms" Machine Learning. 38, 257-286.

[3] Linde, Y., Buzo A., and Gray R.M., (1980) "An Algorithm for Vector Quantizer Design" IEEE Transactions on Communications. 702-710.

[4] Gersho A. and Gray R.M., (1992) "Vector Quantization And Signal Compression" Kluwer Academic Publishers.

[5] Antonio M.P., José C.segura, Antonio J.Rubio , Pedro Garcia and José L.Pérez, (1996) "Discriminative codebook design using multiple VQ in HMM-Based speech recognizers" IEEE trans on Speech and Audio processing. 4 (2), 89-95.

[6] Yury Lifshits, Shay Mozes, OrenWeimann and Michal Ziv-Ukelson, (2009) "Speeding Up HMM Decoding and Training by Exploiting Sequence Repetitions" Journal Algorithmica. 54 (3), 379–399. Springer-Verlag, USA.

[7] Wang, J., Wu X. and Zhang C., (2005) "Support vector machines based on K-means clustering for real-time business intelligence systems" International Journal of Business Intelligence and Data Mining, 1(1), 54-64.

[8] Scholkopf B. and Smola A.J., (2002) "Learning with Kernels, Support Vector Machines, Regularization, Optimization, and Beyond" MIT Press.

[9] Veropoulos K., Campbell C. and Cristianini N., (1999) "Controlling the Sensitivity of Support Vector Machines" International Joint Conference on Artificial Intelligence (IJCAI99). pp: 55-60.

[10] Karakoulas G. and Shawe-Taylor J., (1998) "Optimizing classifiers for imbalanced training sets" In Advances in neural information processing systems. MIT Press, Cambridge, MA, USA.

[11] Lin Y., Lee Y. and Wahba G., (2002) "Support Vector Machines for Classification in Nonstandard Situations" Machine Learning, 46(1-3). 191 - 202.
[12] Liu, H. and Motoda H., (2001) "Instance Selection and Constructive Data Mining" Kluwer, Boston.
[13] Reinartz T., (2002) "A Unifying View on Instance Selection, Data Mining and Knowledge Discovery" 6, 191–210, Kluwer Academic Publishers.
[14] Mitra P., Murthy C. and Pal S., (2004) "A probabilistic active support vector learning algorithm" IEEE Transactions on Pattern Analysis and Machine Intelligence. 26 (3), 413–418.
[15] Li M. and Sethi I., (2006) "Confidence-based active learning" IEEE Transactions on Pattern Analysis and Machine Intelligence. 28 (8), 1251–1261.
[16] Yu L. and Liu H., (2004) "Efficient Feature Selection via Analysis of Relevance and Redundancy" Journal of Machine learning research (JMLR). 5, 1205-1224.
[17] Markovitch S. and Rosenstein D., (2002) "Feature Generation Using General Construction Functions" Machine Learning. 49, 59-98.
[18] Nigel Williams, Sebastian Zander, Grenville Armitage, (2006) "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification" ACM SIGCOMM Computer Communication Review. 36 (5), 5-16. USA.
[19] Tan Feng, (2007) "Improving Feature Selection Techniques for Machine Learning" Computer Science Dissertations. Paper 27. http://digitalarchive.gsu.edu/cs_diss/27.
[20] Ozcift, A. and Gulten A., (2012) "A robust multi-class feature selection strategy based on rotation forest ensemble algorithm for diagnosis of Erythemato-Squamous diseases" Journal of medical systems. 36 (2), 941-951.
[21] Mark A.Hall and Lloyd A.Smith, (1999) "Feature Selection for Machine Learning: Comparing a Correlation_based Filter Approach to the Wrapper" In proceedings of 12 international Florida AI Research. AAAI Press, pp:235-239. USA.
[22] Thomson J., O'Boyle M., Fursin G. and Bj¨orn F., ."Reducing Training Time in a One-shot Machine Learning-based Compiler" 22th international workshop. Proc. Languages and compilers for parallel computing. pp. 399-407. USA.
[23] Rajat R. , Anand M. , Andrew Y.Ng, (2009) "Large-scale Deep Unsupervised Learning using Graphics Processors" In ACM Proceedings of the 26th International Conference on Machine Learning. 382, 110. Canada.
[24] Rabiner L.R., (1989) "A tutorial on hidden Markov models and selected applications in speech recognition" Proceedings of the IEEE. 77, 257-286.
[25] Neal R.M., Hinton G.E., (1998) "A new view of the EM algorithm that justifies incremental, sparse and other variants" in: M.I. Jordan (Ed.), Learning in Graphical Models, Kluwer Academic Publishers. pp. 355–368.
[26] Carroll J.M, Russell J., (1997) "Facial Expression in Hollywood's Portrayal of Emotion" J. Personality and social psychology. 72, 164-176.
[27] MMI Database : http://www.mmifacedb.com/
[28] Ghanem K. and Caplier A. (2011) "Occurrence order detection of face features deformations in posed expressions" ICIEIS, Part II, CCIS 252, pp.56-66, Springer-Verlag.
[29] Rumelhart and McClelland, (1986) "Parallel Distributed Processing" volume 1 and 2. MIT Press.
[30] Kotsiantis S.B., (2007) "Supervised Machine Learning: A Review of Classification Techniques" Informatica. 31, 249-268.