STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **10**, February 2022, pp 160–170.
Published online in International Academic Press (www.IAPress.org)

Hidden Markov Models Training Using Hybrid Baum Welch - Variable Neighborhood Search Algorithm

El Annas Monir*, Mohamed Ouzineb, Badreddine Benyacoub

Institut National de Statistique et d'Economie Appliquée, Rabat, Morocco

Abstract Hidden Markov Models (HMM) are used in a wide range of artificial intelligence applications including speech recognition, computer vision, computational biology and finance. Estimating an HMM parameters is often addressed via the Baum-Welch algorithm (BWA), but this algorithm tends to convergence to local optimum of the model parameters. Therefore, optimizing HMM parameters remains a crucial and challenging work. In this paper, a Variable Neighborhood Search (VNS) combined with Baum-Welch algorithm (VNS-BWA) is proposed. The idea is to use VNS to escape from local minima, enable greater exploration of the search space, and enhance the learning capability of HMMs models. The proposed algorithm has entire advantage of combination of the search mechanism in VNS algorithm for training with no gradient information, and the BWA algorithm that utilizes this kind of knowledge. The performance of the proposed method is validated on a real dataset. The results show that the VNS-BWA has better performance finding the optimal parameters of HMM models, enhancing its learning capability and classification performance.

Keywords Hidden Markov Models, Baum-Welch Algorithm, Variable Neighborhood Search Algorithm

AMS 2010 subject classifications 68T20, 65C60, 68Txx

DOI: 10.19139/soic-2310-5070-1213

1. Introduction

Hidden Markov models have been known as a statistical model with great success and widely used in a vast range of application fields such computational biology, speech processing, pattern recognition, and finance [1, 2, 3, 4, 5, 6, 7] among other disciplines.

Attempts to estimate HMMs parameters were carried out by several authors [9, 10, 11, 12, 13], but the dominant approach to HMM parameter estimation problem is the Baum-Welch algorithm [8], despite its use in practice, the Baum-Welch algorithm can get trapped in local optima of the model parameters. Thus there is need for an algorithm that can escape from the local optimum and then probe the solution space to reach the global optimum of the model parameters.

Variable neighborhood search [15] is among the class of metaheuristics that have provided optimal solutions in many different problem domains, by considering changes of neighborhood in both, the descent phase (to find a local neighborhood optimum), and therfore the perturbation phase (to get out of the corresponding basin of attraction). VNS has been successfully applied to a wide variety of optimization problems such the clustered vehicle problem, the maximum min-sum dispersion problem or the financial derivative problem [16, 17, 18, 19], just to cite several recent works.

*Correspondence to: El Annas Monir (Email: elannas.mounir@gmail.com). Institut National de Statistique et d'Economie Appliquée, Rabat, Morocco.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

With this context, this paper, provide the design, implementation, and experimental setup for a hybridation of the VNS and BWA to approximately estimate optimal model parameters for Hidden Markov models. And assess the performance on a real financial dataset.

The remainder of this paper is organized as follows: In Section 2, we give some basic descriptions of Hidden Markov Models notations, followed by a description of BWA, VNS and the hybrid VNS-BWA algorithms for training HMMs. In Section 3 we describe the data used in this paper, and explains the experiment setup and results. Finally, we conclude this paper.

2. Methodology and Notations

In this section, we expose subsequently the mathematical framework of the discrete case of HMM models, Baum Welch and VNS algorithms. Then we describe the proposed process of estimating HMM models parameters.

2.1. Elements of a Hidden Markov Models

An HMM model is characterized by the following elements :

- S_t : The random variable representing the state at time t, where $0 \leq S_t \leq N-1$ and $0 \leq t \leq T-1$, were N is the number of states in the model, and T being the length of the observation sequence. Let $S = (S_0, S_1, \ldots, S_{T-1})$ be the states sequence.

- O_t : The random variable representing the observation at time t, where $0 \leq O_t \leq M$ and $0 \leq t \leq T-1$, where M is the number of observation symbols, and T the length of the observation sequence. Let $O = (O_0, O_1, \ldots, O_{T-1})$ be the observation sequence.
  In this paper we consider the values of O_t and S_t to be discrete.

- A = $\{a_{ij}\}$ the state transition probabilities matrix, where A $\in R^{N\times N}$ and $a_{ij} = P(S_{t+1} = j | S_t = i)$

- B = $\{b_i(k)\}$ the observation probability matrix, where B $\in R^{N\times M}$ and $b_i(k) = P(O_t = k | S_t = i)$

- π = $\{\pi_i\}$ the initial probability vector, where $\pi \in R^N$, and $\pi_i = P(S_0 = i)$

  We denote an HMM as a triplet $\lambda = (\pi, A, B)$

2.2. Baum-Welch learning

A.Training HMM with a single observed variable

In the following, we summarize the Baum-Welch Algorithm (BWA), for estimating an HMM model parametre λ that generates a sequence of observations . We first define the following probabilities [8]:

- $\alpha_t(i) = P(O_0, O_1, \ldots, O_t, S_t = i | \lambda)$:

  for $i = 0, 1 \ldots, N-1$ and $t = 1, 2 \ldots, T-1$
  - $\alpha_0(i) = \pi_i b_i(O_0)$
  - $\alpha_t(i) = [\sum_{j=0}^{N-1} \alpha_{t-1}(j) a_{ji}] b_i(O_t)$

- $\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_{T-1} | S_t = i, \lambda)$:

  for $i = 0, 1 \ldots, N-1$ and $t = T-2, T-3 \ldots, 0$
  - $\beta_{T-1}(i) = 1$

- $\beta_t(i) = \sum_{j=0}^{N-1} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$
- $\zeta_t(i,j) = P(S_t = i, S_{t+1} = j | O, \lambda)$

  for $i, j \in \{0, \dots, N-1\}$ and $t = 0, 2 \dots, T-2$

  - $\zeta_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{k=0}^{N-1} \alpha_{T-1}(k)}$
- $\gamma_t(i) = P(S_t = i | O, \lambda)$

  for $i = 0, 1 \dots, N-1$ and $t = 0, 2 \dots, T-2$

  - $\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{k=0}^{N-1} \alpha_{T-1}(k)} = \sum_{j=0}^{N-1} \zeta_t(i,j)$

The parameter reestimation formulas,are described by the following expressions:

- $\widehat{\pi_i} = \gamma_0(i)$
- $\widehat{a_{ij}} = \frac{\sum_{t=0}^{T-2} \zeta_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)}$
- $\widehat{b_i(k)} = \frac{\sum_{t=0, O_t=k}^{T-1} \gamma_t(i)}{\sum_{t=0}^{T-1} \gamma_t(i)}$

The parameter λ will be estimated iteratively by Baum Welch procedure as follow:

1. Initialize $\lambda = \lambda_0$ for the model.
2. Compute $\alpha_t(i)$, $\beta_t(i)$, $\zeta_t(i,j)$ and $\gamma_t(i)$.
3. Update λ
4. Repeat steps 2 and 3 until $P(O|\lambda)$ no longer increases.

In this paper λ_0 structure is computed as the following counts :

$$a_{ij} = \frac{Number\ of\ transitions\ from\ state\ j\ to\ state\ i}{Number\ of\ times\ in\ state\ i}$$

$$b_i(k) = \frac{Number\ of\ times\ we\ are\ in\ state\ i\ and\ we\ see\ a\ symbol\ k}{Number\ of\ times\ we\ are\ in\ state\ i}$$

$$\pi_i = \frac{Number\ of\ times\ in\ state\ i}{number\ of\ observations}$$

B. Training HMM with multiple observed variables

HMM can be extended to supports multiple observed variables with one common hidden sequence [14]. Assuming independence between the M observed variables. The triplet λ is defined as $\lambda = (\pi, A, B^{0:M-1})$ where the matrix A corresponding to state transition probabilities and π the initial probability vector with the same structure as defined in the section 2.1. The only diference is that there are multiple emission matrices. The m-th emission probability matrix $B^m = \{b_i^m(k)\}$, were $b_i^m(k) = P(O_t^m = o_k^m | S_t = i)$ and o_k^m is a possible value of the observation O_t^m.
The parameter learning process can be performed by means of the Baum Welch Algorithm extended for multiple observed variables based on the probabilities $\alpha_t(i)$, $\beta_t(i)$,$\gamma_t(i)$ and $\zeta_t(i,j)$. Computed as follows :

- $\alpha_0(i) = \pi_i \prod_{m=0}^{M-1} b^m(O_0^m)$
- $\alpha_t(i) = [\sum_{j=0}^{N-1} \alpha_{t-1}(j) P(S_t = i, S_{t-1} = j, \lambda)] \prod_{m=0}^{M-1} P(O_t^m | S_t = i, \lambda)$

  $= [\sum_{j=0}^{N-1} \alpha_{t-1}(j) a_{ji}] \prod_{m=0}^{M-1} b^m(O_t^m)$
- $\beta_t(i) = \sum_{j=0}^{N-1} P(S_{t+1} = i, S_t = j, \lambda) \prod_{m=0}^{M-1} P(O_{t+1}^m | S_{t+1} = i, \lambda) \beta_{t+1}(i)$

  $= [\sum_{j=0}^{N-1} a_{ij} \beta_{t+1}(i)] \prod_{m=0}^{M-1} b^m(O_{t+1}^m)$
- $\beta_{T-1}(i) = 1$
- $\zeta_t(i,j) = \frac{\alpha_t(i)\beta_{t+1}(j) a_{ij} \prod_{m=0}^{M-1} b^m(O_{t+1}^m)}{\sum_{j=0}^{N-1} \alpha_{T-1}(j)}$
- $\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=0}^{N-1} \alpha_{T-1}(j)} = \sum_{j=0}^{N-1} \zeta_t(i,j)$

The parametre λ can be updated as follows:

- $\widehat{\pi_i} = \gamma_0(i)$
- $\widehat{a_{ij}} = \frac{\sum_{t=0}^{T-2} \zeta_t(i,j)}{\sum_{t=0}^{T-2} \gamma_t(i)}$
- $\widehat{b_i^m(k)} = \frac{\sum_{t=0/O_t^m = o_k^m}^{T-1} \gamma_t(i)}{\sum_{t=0}^{T-1} \gamma_t(i)}$

2.3. *Variable Neighborhood Search Algorithm*

The basic idea of VNS algorithm [15] is to explore a set of predefined neighborhoods to provide a better solution. The algorithm begin by a descent method to a local minimum, then explore distant neighbourhoods of this solution. Each time, one or several points within the defined neighbourhood are used as initial solutions for a local descent. The method jumps from a solution to a new one if and only if a better solution has been found. More formally, Firstly, a finite set of neighborhood structures N_k, $(k = 1, \ldots, k_{max})$ is defined, and $N_k(x)$ the set of solutions in the kth neighborhood of x. Next an initial solution x is randomly generated or constructed. At each iteration, the neighborhood index k is initialized to 1. VNS's procedure is composed of three steps: shaking, local-search and move. In the shaking step, a solution $x^{'}$ in the k^{th} neighborhood of the incumbent solution x is randomly generated. Then, a local-search procedure is applied to the shaking's output $x^{'}$. The local search's output is denoted $x^{''}$. If $x^{''}$ is better than x, $x^{''}$ replaces x from which the search continues with $k = 1$. Otherwise, k is incremented and a new shaking step starts using the $(k+1)^{th}$ neighborhood structure. The procedure continues until termination criteria is met. The most common ones are to fix a maximum computational time allowed, a maximum number of iterations, or a maximum number of iterations between two contiguous improvements. The pseudocode of a VNS algorithm is given by Algorithm 1.

Further modifications to basic VNS as well as several strategies for parallelizing are discussed in [16, 20].

2.4. *VNS-HMM MODEL*

In VNS-BWA training, we first choose an initial solution $\lambda^0 = (\pi^0, A^0, B^0)$, then we re-estimate λ^0 iteratively using BWA, until convergence or reaching iterations limit, to a solution λ^*. Therefore, λ^* is encoded into a string W=$(w_0, w_1, \ldots, w_p)$ (Fig.1), and used to construct an initial solution for the VNS algorithm, the solution space includes all probable parameters with a specified number of observation states and hidden states.

In this paper, we used the basic VNS variant that executes alternately a simple local search procedure and a shaking procedure, together with a neighborhood change step until fulfilling a predefined stopping criterion. The steps of Basic VNS using sequential neighborhood change step are as given in Algorithm 1.

Algorithm 1 Main steps of the VNS algorithm

```
1: Initialization. Select the set of neighborhood structures $N_k$, $(k = 1, \dots, k_{max})$
2: Find Initial Solution x
3: while Termination criterion is not met do
4:     $k \leftarrow 1$
5:     while $k < k_{max}$ do
6:         Shaking. Generate a point $x^{'} \in N_k(x)$ at random
7:         Local search. Apply some local search method with $x^{'}$ as the initial solution. Denote with $x^{''}$ the so-obtained local optimum.
8:         Move or not. If this local optimum is better than the current solution, x $\leftarrow$ $x^{''}$, and k $\leftarrow$ 1, Otherwise, k $\leftarrow$ k+1
9:     end while
10: end while
```

In the proposed VNS algorithm, two neighborhood actions, are defined to generat the neighborhood structures of a candidate solution:

The first neighborhood action is obtained by replacing components of the candidate solution by a number r, as $(w_0, w_1, \dots, r, \dots, w_p)$, where $0 \leq r \leq 1$.

The second neighborhood action is obtained by adding a number r to components of the candidate solution, as $(w_0, w_1, \dots, w_i + r, \dots, w_p)$, and normalize.

We propose two fitness functions to evaluate the degree of optimality based on the learning type, when the learning is supevised, we maximize the
$accuracy = f(W) = \frac{The\ number\ of\ correctly\ classified\ observations}{The\ total\ number\ of\ observations}$
, and when the learning is unsupervided, we maximize the likelihood of observation sequence $g(W) = Log(P(O|\lambda))$.

At the beginning of VNS, the first neighbourhood (k $\leftarrow$ 1) is selected and at each iteration, the parameter k is increased (k $\leftarrow$ k+1). The process of increasing the parameter k occurs when the algorithm is trapped at a local optimum, that until the maximum size of the shaking phase, k_{max}, is reached, Then, k is re-initialized to the first neighbourhood (k $\leftarrow$ 1).

The process of optimization is stopped when the termination condition is met (e.g. a maximum allowed CPU time, reaching to the maximum number of iteration itermax or increasing the optimal fitness under a specified threshold), and the best solution is adopted as the HMM parameter λ^{best}.

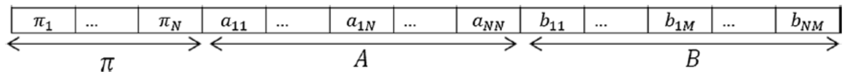


Figure 1. Encoding of the parametre λ

This process the hybrid VNS-BWA algorithm is shown in (Fig.2).

3. Experimental Study

This section describes the experimental study used for evaluating the implementations our model, including the test instances, evaluation metrics, the data set split, experiment designe, and experiments results.

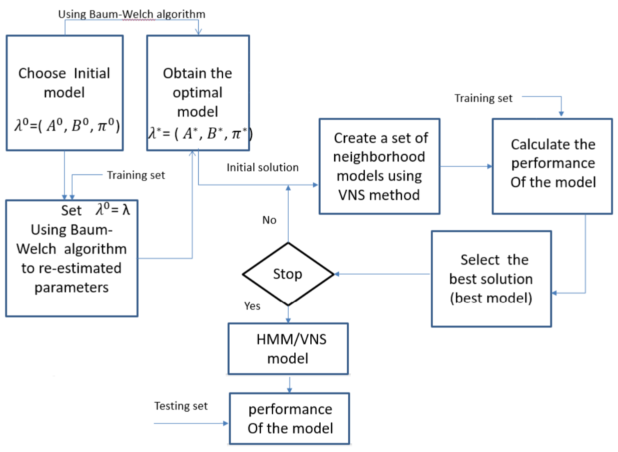

Using Baum-Welch algorithm
Choose Initial model λ^0=(A^0, B^0, π^0)
Obtain the optimal model λ^*= (A^*, B^*, π^*)
Training set
Set λ^0= λ Using Baum-Welch algorithm to re-estimated parameters
Initial solution
Create a set of neighborhood models using VNS method
Training set
Calculate the performance Of the model
No
Stop
Yes
Select the best solution (best model)
HMM/VNS model
Testing set
performance Of the model

Figure 2. The flow chart of the VSN-HMM hybrid model

3.1. *Test Instances*

The data set used in this study is a public data set available from Lending Club [21]. Lending Club is one of the largest online peer-to-peer lending agencies in the United States. Lending Club makes available loan characteristic information from the years 2007 to 2018. Excluding records containing obvious errors and the characteristics, with missing information, and by keeping accepted records, with both good/bad statuses observed. The final data set used for analysis consist of 1,266,782 issued loans including 247426 defaults, with 12 attributes, 5 numerical and 7 categorical: loan amount, annual income, fisco score, dti ratio, interest rate, address state, employment length, home ownership, grade, purpose, delinquency in 2 years, and the loan status.

Table 1 and 2 summerize the descriptive statistics of the lending club dataset.

Table 1. Summary statistics of numerical variables in Lending Club dataset

	loan_amnt	int_rate	annual_inc	dti	fico_score
mean	14601.1	13.2	77900.5	18.1	698.1
std	8746.5	4.8	71082.9	9.6	31.7
min	500.0	5.3	33.0	-1.0	627.0
25%	8000.0	9.8	48000.0	11.8	672.0
50%	12075.0	12.7	65000.0	17.5	692.0
75%	20000.0	16.0	92500.0	23.9	712.0
max	40000.0	31.0	10999200.0	999.0	847.5

Table 2. Summary statistics of categorical variables in Lending Club dataset

	grade	emp_length	home_ownership	purpose	addr_state	delinq_2yrs	Class
unique	7	11	6	14	51	31	2
top	B	10+ years	MORTGAGE	debt_c	CA	0	0

3.2. Binning

Binning techniques are extensively used in machine learning applications. Particulary, in credit scoring[22]. There are various discretization tools to solve the optimal binning problem. In this paper, a recent optimal binning discretization tool is employed to determine the optimal discretization of the variables of the lending club dataset [23].

Table 3. Lending club dataset binning summary

name	data type	n_bins	iv	gini	quality score
annual_inc	numerical	13	0.0248918	0.0878219	0.0230973
loan_amnt	numerical	9	0.043861	0.115813	0.0587686
fico_score	numerical	13	0.128666	0.188358	0.500633
dti	numerical	15	0.0730933	0.15316	0.274635
int_rate	numerical	14	0.483349	0.37248	0.940025
home_ownership	categorical	3	0.0305578	0.0910782	0.0923027
delinq_2yrs	categorical	3	0.00271364	0.0196557	0.00466775
grade	categorical	5	0.472347	0.364479	0.868725
emp_length	categorical	8	0.00154237	0.0216408	2.2341e-05
purpose	categorical	4	0.0170691	0.0616792	0.0452641
addr_state	categorical	10	0.0157176	0.0664218	0.047441

Table 3 shows the binning results for each characteristic of the lending club dataset: name, data type and the number of bins. In Addition to several quality metrics: Gini index, Information Value (IV), and the quality score. The resulting bins will be used to perform data transformations step for HMM modeling. The binning process is also used to address statistical noise, data scaling , and reducing the HMM model complexity.

3.3. Evaluation Metrics

There are many indicators for measuring the performance of a credit scoring models. In this work, we consider the AUC metric it refers to the area under the receiver operating characteristic curve (ROC), which is a comprehensive indicator reflecting the continuous variables of sensitivity and specificity. The AUC mesure range usually between 0.5 and 1. If a classifier has an AUC value is 0.5, it means that the classifier randomly guesses the samples value. The higher the AUC value, the better predictive power of the classifier.

To mesure fitting performance of the proposed model we use the Accuracy metric: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ Where true positive (TP) is the number of instances that actually belong to the good group that were correctly classified as good by the classifier, true negative (TN) refer to the number of instances that belong to the bad group and correctly classified as bad, false positive (FP) to the number of instances that are from bad group but mistakenly classified as good, and false negative (FN) refer to the number of instances from the good group but incorrectly classified as bad.

3.4. Dataset split

To minimize the impact of data dependency and improve the reliability of the estimates, k-fold cross validation is used to create random partitions of the dataset as follow :

1. The data set is split into k mutually folds of nearly equal size.
2. Choose the first subset for testing set and the k-1 remainer for training set.
3. Build the model on the training set.
4. Evaluate the model on the testing set by calculating the evaluation metrics.
5. Alternately choose the following subset for testing set and the k-1 remainder for training set.

6. The structure of the model is then trained k times each time using $k - 1$ subsets (training set) for training and the performance of the model is evaluated k-1 on the remaining subset (testing set).
7. The predictive power of classifier is obtained by averaging the k validation fold estimates found during the k runs of the cross validation process.

Common values for K values are 5 and 10. In this experiment, we take the value of K as 5. In this experiment, cross-validation was mainly used to assist in the adjustment of model parameters and the evaluation of the final model results.

3.5. Experimental Design

In the experimental setup using lending club p2p dataset. The risk being good or bad applicant is represented by two hidden states in the HMM model, denoted as S_0, and S_1. Each observation sequence correspond to an input variable preprocessed into a canonical form for its HMM model where the number of observation symbols are the number of bins resulting this procedure of discretization defined in section 3.2. We compute the initial HMM model structure for each characteristic as described is section 2.2.A.

To compare the classification performance of the VNS-HMM model with HMM model we proceed as follows:

Step 1: Randomly under-sample the dataset to handle the class imbalance.

Step 2: Split the data set is split into 5 mutually folds of nearly equal size.

Step 3: Build the credit scoring models using the training folds.

Step 4: Utilize the classifiers built up in Step 3 to predict the PD and label of samples in test folds

Step 5: Evaluate and compare the VNS-HMM and HMM models performances by averaging the results values and ploting the ROC curves.

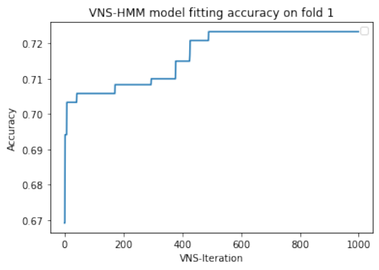
Additionally, we include a total of six machine learning classifiers, namely Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Random Forest (RF), Extreme Gradient Boosting (XGBoost), CatBoost, and Light Gradient Boosting Machine (LightGBM) to compare their performances to the VNS-HMM model.

3.6. Experiments Results and Analysis

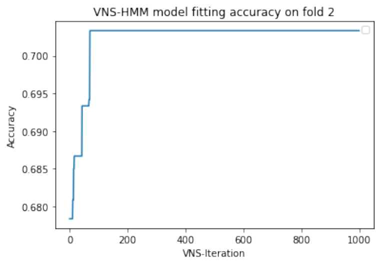
Figure 3 presents the fitting accuracies over the iterations of VNS on each of the training folds. The numerical AUC results and the ROC curves of the proposed VNS-HMM model and the HMM models, are shown in Figure 4.

From the experimental results, the following conclusions can be drawn. First, employing the VNS method for HMM training data yelled significant better accuracies while performing the search strategy. VNS was able to provides the opportunity for an improved fitting accuracy and enabled to escape local optimums, representing the effectiveness of our approach. Furthermore the VNS training model did not require many iterations to achieve better performance. Second, the predictive performance on AUC of the VNS-HMM outperforming those of HMM in the five folds.

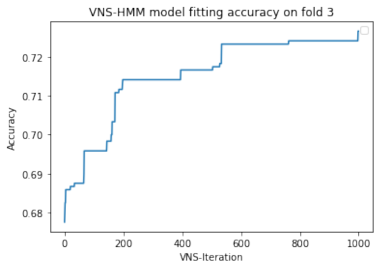
Table 4 show compared performance of the proposed model and the other classifiers. VNS-HMM model obtains the best value for AUC followed by SVM and third best value in term of accuracy after SVM and MLP.



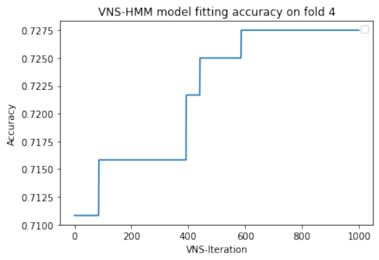
(a) VNS-HMM fitting accuracy on the fold 1



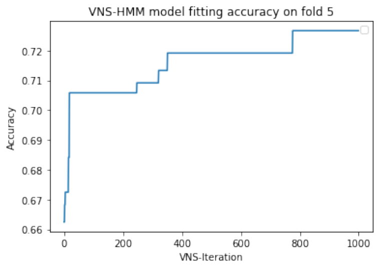
(b) VNS-HMM fitting accuracy on the fold 2



(c) VNS-HMM fitting accuracy on the fold 3

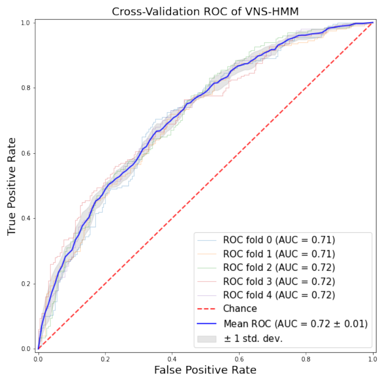


(d) VNS-HMM fitting accuracy on the fold 4

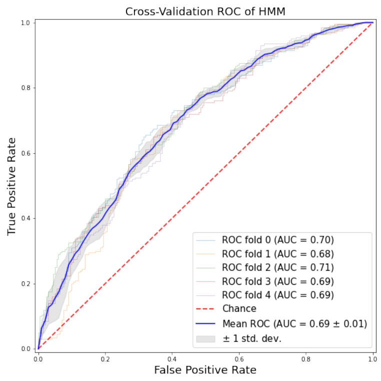


(e) VNS-HMM fitting accuracy on the fold 5

Figure 3. VNS-HMM fitting accuracy on the 5 training folds of the cross validation



(a) 5 CV ROC of VNS-HMM



(b) 5 CV ROC of HMM

Figure 4. 5 CV ROC of the HMM and VNS-HMM models

Table 4. Models performance comparaison using the Lending Club dataset

	Accuracy	AUC
VNS-HMM	**0.6456**	**0.7182**
SVM	**0.6522**	**0.7111**
CatBoost	0.6433	**0.7043**
MLP	**0.6467**	0.7015
Lightgbm	0.6200	0.6775
RF	0.6300	0.6771
Xgboost	0.6067	0.6493

Conclusion

This paper proposed VNS-BWA as a new approach for hidden markov models training. Where VNS is used in combination with BWA to explores the search space for the optimal parameter structure of HMMs. The result demonstrated the ability of the VNS-BWA to find better HMM parametres and enhancing it's predictive performance. We limited our approch to discret HMM models, but it can deal with continuous observations when the density functions are normal distributions or a known probability distribution which is not feasible in our case. Several directions remain to explore. Since both VNS and BWA support GPU-computing, we consider working on a parallel implementation to tackle larger problems. Also, it would be very interesting to compare VNS-HMM model to semi-supervised techniques in machine learning for future research.

REFERENCES

1. Velasco, A., James, B. T., Wells, V. D., Girgis, H. Z. (2019). Look4TRs: A de-novo tool for detecting simple tandem repeats using self-supervised hidden Markov models. Bioinformatics.
2. Pachter L, Alexandersson M, Cawley S (2002) Applications of generalized pair hidden Markov models to alignment and gene finding problems. J Comput Biol 9(2):389-399
3. Mao, S., Tao, D., Zhang, G., Ching, P. C., Lee, T. (2019). Revisiting Hidden Markov Models for Speech Emotion Recognition. ICASSP 2019
4. Han, S. Y., Liefbroer, A. C., Elzinga, C. H. (2019). Mechanisms of family formation: An application of Hidden Markov Models to a life course process. Advances in Life Course Research.
5. Singh, M., Kumar, S., Kumar, S., Garg, T. (2019). Credit Card Fraud Detection Using Hidden Markov Model. International Journal of Engineering and Computer Science, 8(11), 24878-24882.
6. Monir, E.A., Ouzineb, M., Benyacoub, B.: Multi dimensional Hidden markov model for credit scoring systems in Peer-To-Peer (P2P) lending. BDNT (2019). LNNS, vol. 81, pp. 73-83. Springer, Cham.
7. Ge-Er, T., Chang-Zheng, H., Jin, X., Xiao-Yi, J.: Customer credit scoring based on HMM/GMDH hybrid model. Knowl. Inf. Syst. 36(3), 731-747 (2013).
8. Baum L.E., Petrie T., Soules G. Weiss N., A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, Ann. Math. Stat., Vol. 41, No. 1, pp. 164-171. (1970)
9. R. Thomsen, Evolving the topology of hidden Markov models using evolutionary algorithms, in: Proceedings of Parallel Problem Solving from Nature VII (PPSN-2002), 2002, pp. 861-870.
10. T.-Y. Chen, X.-D. Mei, J.-S. Pan, S.-H. Sun, Optimization of hmm by the tabu search algorithm., Journal Information Science and Engineering 20 (5) (2004) 949-957.
11. Aupetit, S., Monmarché, N., Slimane, M. (2006). Hidden Markov Models Training by a Particle Swarm Optimization Algorithm. Journal of Mathematical Modelling and Algorithms, 6(2), 175-193.
12. Hidden markov models training using population-based metaheuristics S Aupetit, N Monmarché, M Slimane - Advances in metaheuristics for hard optimization, (2007)
13. Kordnoori, S., Mostafaei, H., Behzadi, M. H. (2019). PSO Optimized Hidden Markov Model Performance Analysis for IEEE 802.16/WiMAX Standard. Wireless Personal Communications.
14. Li, Xiaolin, Marc Parizeau, and Rjean Plamondon. Training hidden markov models with multiple observations-a combinatorial method. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, no. 4: 371-377.(2000)
15. N. Mladenovic, P. Hansen, Variable neighborhood search, Computers and Operations Research 24 (1997) 1097-1100.
16. Hansen, P., Mladenovic, N., Perez, J.A.M.: Variables neighborhood search: methods and applications. Ann. Oper. Res. 175, 367-407 (2010)
17. Z. Amirgaliyeva, N. Mladenovic, R. Todosijevic, and D. Urosevic. Solving the maximum min-sum dispersion by alternating formulations of two different problems. European Journal of Operational Research, 2016.

18. Defryn and K. Sorensen. A fast two-level variable neighborhood search for the clustered vehicle routing problem. Computers and Operations Research, 2017. ISSN 0305-0548.
19. N. Amaldass, C. Lucas, N. Mladenovic, Variable neighbourhood search for Financial derivative problem, Yugoslav Journal of Operations Research 29 (2019) 359-373. .
20. Jose A. Moreno Perez, Pierre Hansen and Nenad Mladenovic, Parallel Variable Neighborhood Search. Parallel Metaheuristics: A New Class of Algorithms Chapter 11 (pages 247-266)
21. https://www.lendingclub.com/info/download-data.action
22. Siddiqi, N. (2017). Intelligent credit scoring: Building and implementing better credit risk scorecards (2nd ed.). Hoboken, NJ: Wiley.
23. G Navas-Palencia. Optimal binning: mathematical programming formulation (2020) http://arxiv.org/abs/2001.08025