International Journal of Advanced Smart Convergence Vol.5 No3 08-15 (2016)
http://dx.doi.org/10.7236/IJASC.2016.5.3.8

IJASC 16-1-3

Could Decimal-binary Vector be a Representative of DNA Sequence for Classification?

Prima Sanjaya[1], Dae-Ki Kang[2]

[1]*Department of Ubiquitous IT, Dongseo University, 47 Jurye-ro, Sasang-gu, Busan 47011, Republic of Korea, email: mr[dot]primasanjaya[at]gmail[dot]com*

[2]*Department of Computer & Information Engineering, Dongseo University, 47 Jurye-ro, Sasang-gu, Busan 47011, Republic of Korea, email: dkkang[at]dongseo.ac.kr*

Abstract

In recent years, one of deep learning models called Deep Belief Network (DBN) which formed by stacking restricted Boltzman machine in a greedy fashion has beed widely used for classification and recognition. With an ability to extracting features of high-level abstraction and deal with higher dimensional data structure, this model has ouperformed outstanding result on image and speech recognition. In this research, we assess the applicability of deep learning in dna classification level. Since the training phase of DBN is costly expensive, specially if deals with DNA sequence with thousand of variables, we introduce a new encoding method, using decimal-binary vector to represent the sequence as input to the model, thereafter compare with one-hot-vector encoding in two datasets. We evaluated our proposed model with different contrastive algorithms which achieved significant improvement for the training speed with comparable classification result. This result has shown a potential of using decimal-binary vector on DBN for DNA sequence to solve other sequence problem in bioinformatics.

Keywords: *Bioinformatics, DNA Sequence Classification, Deep Learning, Deep Belief Network, Restricted Boltzmann Machine*

1. Introduction

Nowadays, machine learning technique including deep learning have seen significant interest which is can learn layered, hierarchical representation of high dimensional data. It has been successfully implemented in image recognition, speech recognition, natural language processing, video recognition, etc [1-4]. Since deep learning deals with higher dimensional data, in some case it obtains better accuracy if the amount of data also follows, applying this technique into bioinformatics level becomes more challenging. Still, the implementation itself is not extensively fulfilled and need development further. Here, we effectuate deep learning model into DNA sequence classification and tackle some problem of time complexity in training phase.

The most of deep learning model task is in the training phase. Training deep layers with thousands of variables and computations take times longer. In vanishing the gradient descent problem for instance, there are some research have been proposed to overcome this problem such as Dropout [5], DropConnect [6], etc

Manuscript Received: July 04, 2016 / Revised: July 15, 2016 / Accepted: July 30, 2016
Corresponding Author: dkkang@dongseo.ac.kr
Tel:+82-51-320-1724, Fax: +82-51-327-8955
Department of Computer and Information Engineering, Dongseo University,
47 Jurye-ro, Sasang-Gu, Busan 47011, Republic of Korea

In other case, faster training speed of deep model is also demanding. Separating the data into several chunks and processing it with different CPU/GPU, increasing the capacity of CPU/GPU [7] are some approaches for this problem. Traditionally, the existing preprocessing phase method for DNA sequence classification is by using one-hot-vector encoding, but it has new problem of data imbalance. Although this encoding technique works in a small sample of data, but it is impractical for real bigger data such as DNA sequence. In this research we concern about data representation and propose new encoding using decimal-binary encoding method.

2. Background

2.1 Deoxyribonucleic Acid (DNA)

Deoxyribonucleic acid or DNA is the hereditary material and flow biological information in living organisms. A structure of DNA consists of four nucleotides with specific pairs of bases can bound together which are; adenine (A) with thymine (T), and guanine (G) with cytosine (C). As shown in Figure 1, genetic information is delivered from DNA to protein through a procedure called *gene expression* [8]. There are three major steps in gene expression which are: transcription, splicing and translation.

Eukaryotic genes have two subunits of internal structure: exons (protein-coding regions) and introns (non –coding regions). Introns are in between of exons, and the boundary between an intron and an exon is called the *splice junction* (*site*). There are splice site consensus strings called *canonical splicing* patterns. The most frequent patterns are dimer GT (*donor*) and dimer AG (*acceptor*) at the boundaries.

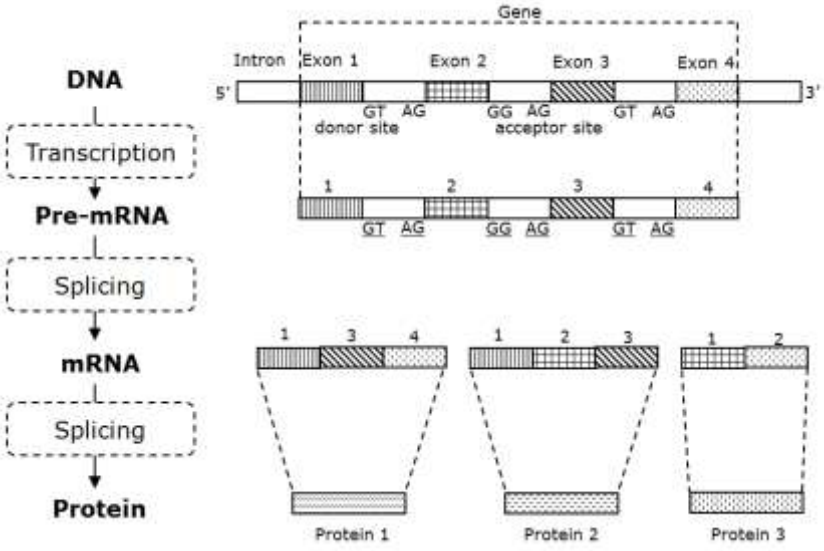

Figure 1. Gene expression process.

DNA and RNA binding proteins play a central role in gene regulation, including transcription and alternative splicing. During transcription, DNA molecule is copied into precursor messenger RNA (pre-mRNA) by the enzyme of polymerase. The general steps of transcription process are; binding RNA polymerase with one or more general transcription factor to promoter DNA, creating a transcription bubble, matching RNA nucleotides to the complementary nucleotides of one DNA strand, forming RNA strand, freeing synthesized RNA strand, including polyadenylation, capping, and splicing if the cell has a nucleus, and the RNA may remain in the nucleus or exit to cytoplasm [9]. For a single gene, various combination of alternative splicing (more than 100.000 that occurs in human cell) gives rise to the enormous diversity of protein.

2.2 Restricted Boltzmann Machine

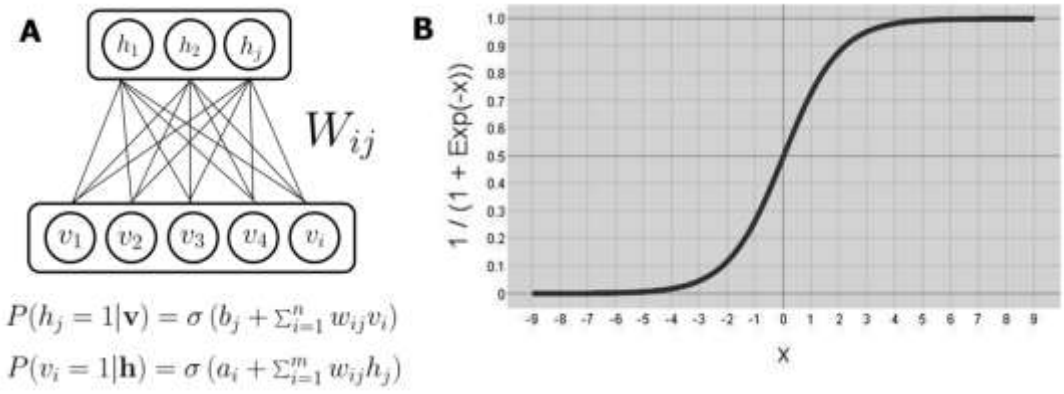

Figure 2. (a) RBM representation (b) sigmoid function

Restricted Boltzmann Machine (RBM) [10] is an undirected bipartite graphical model, where the first layer consists of visible units to observe the data and hidden units in the second layer to capture the dependencies between variables. The visible and hidden layers are fully connected via symmetric undirected weights, determining by the energy of joint configuration which define as

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j \quad (1)$$

The model parameters are weight (W), visible bias a, and hidden bias b. W are the symmetric weight of $V \times H$ dimensions. The absence of connection between units in the hidden layer make it is independent given the state of the visible variables and vice versa [11]. The joint distribution under the model is given by $P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})}$. Since the partition function $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$ and the probability of observing visible units **v** is given by $P(\boldsymbol{v}) = \frac{1}{z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$, which is tractable by the Gibbs sampling, therefore estimating the probability distribution in the hidden layer **h** and visible layer **v** is fast and easy, which is given by $P\left(h_j = 1 \,|\boldsymbol{v}\right) = \sigma(b_j + \sum_{i=1}^{n} w_{ij} v_i)$, where a sigmoid activation function $\sigma(x) = \frac{1}{1+\exp(-x)}$. Similarly, the activation probability in the visible units **v** is given by $P(v_i = 1 \,|\boldsymbol{h}) = \sigma(a_i + \sum_{i=1}^{m} w_{ij} h_j)$. We can measure the minimum log-likelihood of input data distribution in RBM training.

2.2 Deep Belief Network for Text

Deep Belief Network (DBN) is one of widely explored deep learning models which has RBM structure at the top of layer forming sigmoid belief network in the lower layer to visible layer [12]. It is a generative probabilistic model formed by stacking the RBMs to reach the equilibrium state in the hidden layer once running Gibbs sampling in a greedy layer-wised fashion. Since, it is trained by generating data in an unsupervised way, DBN is an appropriate models for reconstruction of given data. In addition, this model can be used to perform classification. For example, adopting back propagation algorithm in DBN transforms the DBN into Deep Neural Network (DNN) with outstanding classification performance.

DBN has been applied in text data problem such as topic categorization, spam detection, and web classification. They convert the text data into one-dimensional sequence by using vocabulary in dictionary and match each word to represent vector called word vector. The size of word vectors are fixed for each model and the values of them are either as a part of the DBN or fixed by some other method such as

word2vector [13].

In email spam detection for instance—suppose we have *n* words in $\mathbf{V}_n = [word_1, word_2, word_3, \ldots word_n]$ which consist all vocabulary in the email, by applying term frequency-inverse document frequency algorithm (TF-IDF), we obtain the *x* most important words from $\mathbf{V}_n$ to the dictionary $\mathbf{D}_x$. After that, by matching each word in the dictionary $\mathbf{D}_x$ that exist in the email will convert the vocabulary into binary vectors. The binary vector has the same dimension as the keyword dictionary vector. Thus the data for binary classification task we feed into classification algorithm consist of $i \times x$ matrix like $\mathbf{B}_x = [0,1,1,0,\ldots \boldsymbol{x}]$ where *i* is the number of email.

3. Proposed Method

We propose a new encoding for DNA sequence dataset which is using decimal-binary vector to represent the nucleotides sequence. The binary vector will be compared to one-hot-vector. Then we organizing DBN by stacked the RBM with different contrastive divergence algorithm.

3.1 Encoding

3.1.1 One-hot-vector Encoding

In one-hot-vector encoding, we only use four characters in the dictionary which are A (Adenine), T (Thymine), G (Guanine), C (Cytosine) to convert DNA sequence into binary. In Figure 3, we show an example of translating a DNA sequence into binary vector. As shown in the example, every character in the sequence has different one-hot-vector. For example A = [1,0,0,0] ; T = [0,1,0,0] ; etc. This one-hot vector will be translating DNA sequence each character. So if we have the sequence of "ATA", the binary vector is B = [1,0,0,0,0,1,0,0,1,0,0,0]. According to this, if we have 64 sequences, the binary vector dimension becomes four times bigger because one character will be represented of 4 bits one-hot-vector, therefore we introduce new encoding method by using decimal-binary vector.

3.1.2 Decimal-binary Encoding

Instead of using nucleotides vocabulary as dictionary vector, decimal-binary vector is transforming the sequence as decimal representation. As mention before, DNA has only four nucleotides. This amount can be stored into two bits of binary number. For example A can be represented as 0 which is "00" in binary number, Same as T = [0,1] which is 1, and so on. After that, given sequence of "ATA", the binary vector is B = [0,0,0,1,0,0] (See Figure 3). Although the dimension of the encoded sequence is expanding from the original one, comparative to one-hot-vector encoding—it is half reducing from 4 bits each character in sequence to 2 bits. So, this approach indicates more effective in practice.

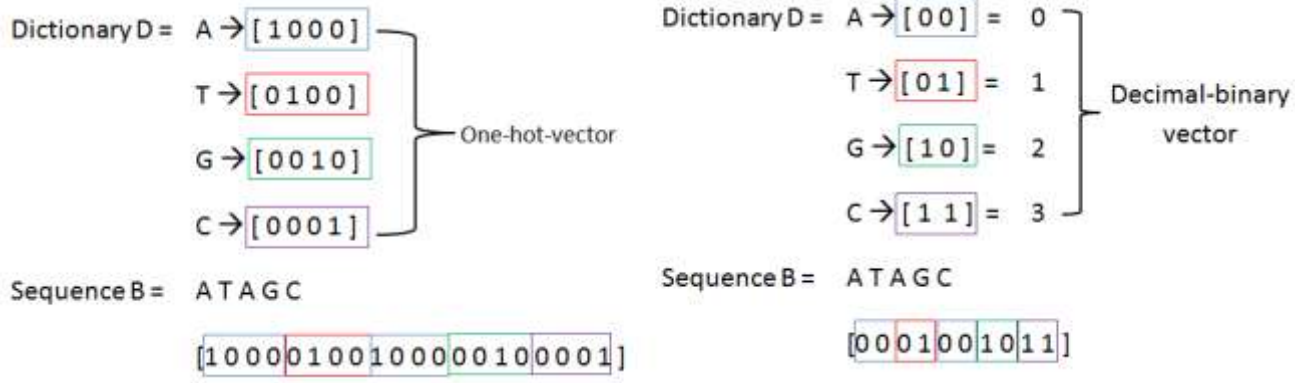

Figure 3. left: one-hot-vector encoding ; right: decimal-binary vector encoding

3.2 Forming DBN by Training Stack of RBMs and Fine-tuning

The building block of a DBN is a probabilistic model called Restricted Boltzmann Machine (RBM) [14]. To learn a DBN, RBMs are applied recursively with the feature activation produced by one RBM acting as the data for training the next RBM in the stack. Suppose *x* be the input, and $\mathbf{h}^i$ the hidden variables at layer *i*, with joint distribution, $P(x, \mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^l) = P(x|\mathbf{h}^1)P(\mathbf{h}^1|\mathbf{h}^2) \ldots P(\mathbf{h}^{l-2}|\mathbf{h}^{l-1})P(\mathbf{h}^{l-1}|\mathbf{h}^l)$, where all the conditional layers are factorized conditional distribution for which computation of probability and sampling are tractable. One consideration the hidden layer $\mathbf{h}^i$ is a binary random vector element of $\mathbf{h}_j^i$ with running Gibbs sampling alternatively using a sigmoid function. After training a stack of RBM with proposed approach, we added an output layer on the top of them to form a DBN. By forming gradient descent on supervised cost, whole network parameters are minimizing the error (e.g. fine-tuned).

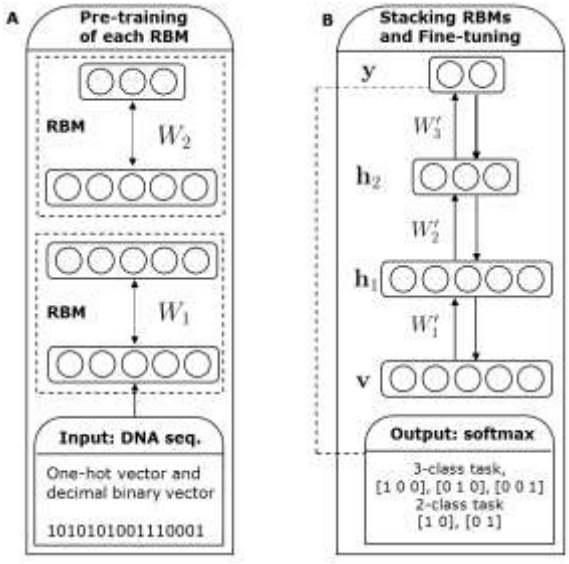

Figure 4. Proposed Method : (a) pre-training (b) fine tuning

4. Experimental Result and Discussion

In this section, we assess the applicability of decimal-binary vector to DBN and compare it with one-hot vector encoding in different standard contrastive divergence algorithms.

4.1 Dataset

We consider 2 datasets of DNA sequence. First is splice dataset [15]. It is molecular biology splice-junction gene sequences with associated imperfect domain theory. It has 3190 instances with 61 attibutes. All examples and categories of "ei" and "ie"—include every "split-gene" for primates were taken from Genbank 64.1. The non-splice examples were taken from sequences known not to include a splicing site. The second dataset is promoter dataset [16]. It is an *escherichia coli* promoter gene sequences with partial domain theory. It has 106 instances with 58 attributes.

Each dataset was separated into training set (Tr), validation set (Vl) and test set (Ts). Training set is used for both training and fine-tuning. Validation set is used for calculating the best configuration of hyper-parameters, and testing set is used for assessing the fix hyper-parameters to classify such problems. For splice dataset, we divided data into 1642 instances of training set, 1536 instances of validation set, and 1536 instances of test set. For promoter dataset, each encoding method was separated into 80(Tr) - 26(Vl) - 26(Ts).

4.2 General Setup

Here are the setup parameters of DBN. All dataset was set to 50, 100, 150, and 200 epoch of learning iterations. For splice dataset, we chose 100 instances each mini-batch and the last 25 persistent chains for PCD training. On the other hand, both mini-batch and persistent chains was selected 10 instances for promoter dataset. We ran experiment with {0.1} value of learning rate. We also included momentum to smooth the trajectory of the gradient descent, which was {0.5, 0.4, 0.3, 0.2, 0.1}. Finally, we added {2 $\times 10^{-4}$} L2 penalty weight decay to the objective function for encouraging smaller weights. The architecture of DBN each dataset are;

- splice (one-hot-vector)-240-500-500-2000-3;
- splice (decimal-binary-vector)-120-500-500-2000-3;
- promoter (one-hot-vector)-228-500-500-2000-2;
- promoter (decimal-binary-vector)-114-500-500-2000-2;

4.3 Result and Discussion

4.3.1 Result on Splice Dataset

Table 1. Splice Test Error Result

Dataset	Architecture	Method	Epoch	Time elapsed (s)	Error
Splice (one-hot-vector)	240-500-500-2000-3	CD-1	50	119.880	0.0737
			100	232.473	0.0568
			150	348.712	0.0659
			200	460.405	0.0776
		CD-10	50	505.930	0.0855
			100	930.317	0.0691
			150	1386.114	0.0652
			200	1831.227	0.0828
Splice (decimal-binary-vector)	140-500-500-2000-3	CD-1	50	115.742	0.0861
			100	226.237	0.0855
			150	334.834	0.0757
			200	446.120	0.0705
		CD-10	50	480.418	0.0802
			100	904.016	0.0685
			150	1351.253	0.0711
			200	1809.344	0.0848

In splice dataset, overall our proposed method has shown comparative test classification error with faster training time than one-hot vector encoding. For CD-1, the training time interval—from 50-200 epoch are 4.139 s, 6.236 s, 13.878 s, 14.285. From this, we can see the time differences followed the increasing number of epoch. Same as CD-10, the accuracy show comparable result in average of 0.00345 difference error with faster training time.

The number of instances, attributes and parameters calculation will effect time effectiveness. Reducing some of them might cost computation easy and faster. In this case, we are reducing the number of attributes without disregard of the representation objective. As we shown in Table 1, decimal-binary vector encoding performed better training speed in various contrastive divergence algorithms.

4.3.2 Result on Promoter Dataset

Table 2. Promoter Test Error Result

Dataset	Architecture	Method	Epoch	Time elapsed (s)	Error
Promoter (one-hot-vector)	228-500-500-2000-2	CD-1	50	25.020	0.1154
			100	43.580	0.1538
			150	61.305	0.1923
			200	79.710	0.1923
		CD-10	50	95.494	0.1923
			100	195.896	0.1538
			150	300.968	0.1538
			200	423.450	0.2308
Promoter (decimal-binary-vector)	114-500-500-2000-2	CD-1	50	24.873	0.1923
			100	43.428	0.1538
			150	61.758	0.2308
			200	79.140	0.2308
		CD-10	50	105.126	0.1538
			100	203.874	0.1923
			150	295.647	0.1923
			200	386.827	0.1923

For promoter dataset, both decimal-binary vector and one-hot vector exceeded comparative test error. Because the number of instance is small, it shown less significant difference. However, from Table 2, we can see that proposed encoding method still can work for classification. Same, for all database each encoding method, it obtained faster training speed.

5. Conclusion

The deep belief network has shown its excellent performance in many study fields. In this research, it also worked well in dealing with DNA sequence. By comparing one-hot vector encoding and our proposed—decimal binary encoding, and applying DBN model, we have achieved significant performance classification with faster training speed. Since DBN has an ability of extracting high-level abstraction feature, as also dealing with the huge data, our proposed method might be helpful to the discovery of classification task over DNA level.

6. Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (no.NRF-2015R1D1A1A01061328).

References

[1] G. E. Hinton, “Learning multiple layers of representation,” TRENDS Cognitive Sci., vol. 11, no. 10, pp. 428–434, 2007.

[2] G. E. Dahl, M. Ranzato, A. Momamed, and G. E. Hinton, “Phone Recognition with the Mean-Covariance Restricted Boltzmann Machines,” in Advances in Neural Information Processing Systems NIPS. Cambridge, MA, USA: MIT Press, 2010.

[3] G. Attardi, "a Deep Learning NLP pipeline" in Proceedings of NAACL-HLT 2015, pp. 109-115, May 31-June 5, 2015.

[4] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in CVPR, 2014.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[6] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," Icml, no. 1, pp. 109–111, 2013.

[7] Al-Absi, A. A., and Kang, D.-K., "Long-read Alignment with Parallel MapReduce Cloud Platform," BioMed Research International, Vol. 2015, 2015.

[8] Lockhart, David J and Winzeler, Elizabeth A. Genomics, gene expression and DNA arrays. Nature, 405(6788): 827–836, 2000.

[9] Clancy, S, "DNA transcription". Nature Education 1(1):41, 2008.

[10] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets." Neural computation, vol. 18, no. 7, pp. 1527–54, 2006

[11] A. Fischer, C. Igel, An Introduction to Restricted Boltzmann Machines," Progress in Pattern Recognition, Image Analysiss, Computer Vision, and Applications, vol. 7441, pp. 14-36, 2012.

[12] T. Tieleman, "Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient," Proceedings of the 25th International Conference on Machine Learning, vol. 307, p. 7, 2008.

[13] Mikolov, T., Chen, K., Greg, C. and Dean, J. (2013) Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

[14] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, "Greedy layer-wise training of deep networks," Advances in neural information processing systems, 19, p.153., 2007

[15] Splice Dataset, Machine Learning Repository, University of California, https://archive.ics.uci.edu/ml/machine-learning-databases/molecular-biology/splice-junction-gene-sequences/

[16] Promoter dataset, Machine Learning Repository, University of California, https://archive.ics.uci.edu/ml/machine-learning-databases/molecular-biology/promoter-gene-sequences/