A Recurrent Neural Network for the Detection of Structure in Methylation Levels along Human Chromosomes

Wim De Mulder, Rafel Riudavets, Martin Kuiper

Norwegian University of Science and Technology
Trondheim, Norway
Emails: wim.demulder@kuleuven.be, rafelriudavets@gmail.com, kuiper@ntnu.no

***Abstract*—Recurrent Neural Networks (RNN) have been used in multiple tasks such as speech recognition, music composition and protein homology detection. In particular, they have shown superior performance in predicting structure in time series data. To our knowledge, RNN have not been used on DNA methylation data. Methylation patterns on chromosomal DNA represent an important form of epigenetic imprinting, a form of epigenetics that results in heritable gene expression and phenotype changes. DNA methylation is one of the mechanisms that a cell uses to fine-tune the expression levels of its individual genes, and it has been shown to affect very specific areas around specific genes. The methylation state of the human chromosomal DNA can be readily assessed with microarray technology, allowing the determination of the methylation status of thousands of positions along the individual chromosomes of the genome. With RNN analysis, we show that these methylation patterns have substantial structure, when relatively large stretches of chromosomes are tested. Furthermore, we show that each chromosome appears to have its own distinctive sequential methylation structure, but that this structure breaks down, to some extent, when normal cells develop into a tumour.**

Keywords–Recurrent neural networks; Nash-Sutcliffe efficiency; Epigenetics.

I. INTRODUCTION

A. *Biological background to the research problem*

Methylation of cytosine residues in DNA represents an important form of epigenetic modification [1]. Although an organism's form and function is based on its genome, characterized by a specific DNA sequence, the epigenetic state of the genome adds an important regulatory component to how specific genes in the genome are expressed and together define an organisms' phenotype, and it is a heritable trait [2]. Epigenetic modifications are not encoded by the nucleotide sequence of the genome [3], but by small modifications in the form of methyl groups or hydroxymethyl groups that are attached either to cytosine residues that are part of so-called CpG stretches ('CpG islands') within the DNA sequence, or to proteins that are bound to the DNA, together forming the biologically active form of DNA called chromatin [4]. The epigenetic state of the chromatin has a considerable plasticity, allowing a cell to adapt to new environmental challenges and demands. DNA methylation can for instance regulate gene expression by recruiting proteins involved in gene repression or by inhibiting the binding of one or more transcription factors to DNA [5]. The epigenetic state, however, is also prone to malfunctions that can result in diseases. Cancer cells represent an important cellular malfunction where among others the methylation state of the chromosomal DNA is affected [6].

A popular technique for determining the methylation status across the genome uses microarray-based Illumina Infinium methylation assays [7]. DNA is denatured (i.e., converted to single strands) and treated with sodium bisulfite, which deaminates the normal, unmethylated cytosine residues and effectively converts them to uracil, while methylated cytosine residues are resistant to this treatment [8]. To estimate the methylation status at a specific CpG site of a chromosome, the Illumina Infinium assay utilizes a pair of DNA probes that can distinguish between sequences with the native cytosine and the converted uracil. Illumina's primer-extension assay measures the intensities of methylated and unmethylated forms (alleles) of individual sites of the genome and the methylation level is calculated based on the signals of this pair of probes. It is convenient to express the methylation level as a value ranging from 0 to 1, referred to as the beta value, and which can be interpreted as the percentage of methylation [9]. Beta values are typically low, as illustrated by the histogram shown in Fig. 1. This histogram is taken from [9] and represents a typical sample measured by the Illumina Infinium HumanMethylation27 BeadChip, which interrogates 27,578 CpG sites where methylation primarily occurs [10].

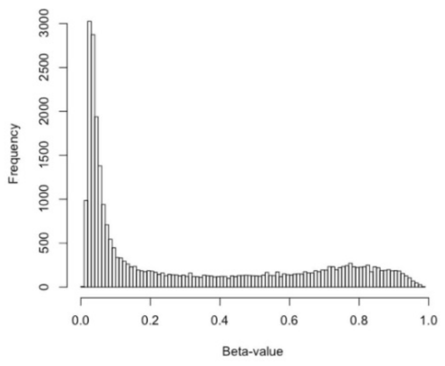

Figure 1. A histogram of beta values (from [9]).

B. *Goal and organization of this paper*

The methylation levels along a given chromosome have similarities to a time series, where the role of time is substituted by the location of the probes with respect to the DNA. However, whereas time series data are often equidistant, the measurement of methylation sites across a chromosome relies on assays using probes that are designed based on very specific local DNA sequence characteristics (is there a gene? is there a CpG island close to the gene?), resulting in a much more irregular distribution of measured points. The research question we consider here is whether there is still some structure in the sequence of beta values that correspond to probes of which the methylation level is consecutively measured along a given chromosome. Given the fact that the probes are not equidistantly located and that methylation levels arise as the result of complex biological processes, it is not clear whether one should expect some predictable structure to begin with. Fig. 2 shows the beta values of 500 consecutive probes that were collected from a randomly chosen chromosome of some patient that was arbitrarily selected from our data set. The apparently erratic pattern in the figure makes it clear that the human eye alone is not advanced enough to answer our research question. We will, therefore, rely on recurrent neural networks in investigating this research topic. More concretely, we set out to analyze whether recurrent neural networks detect structure among a stretch of consecutive methylation sites and use it to predict the next beta value adjacent to that stretch. If so, this would show that structure is present.

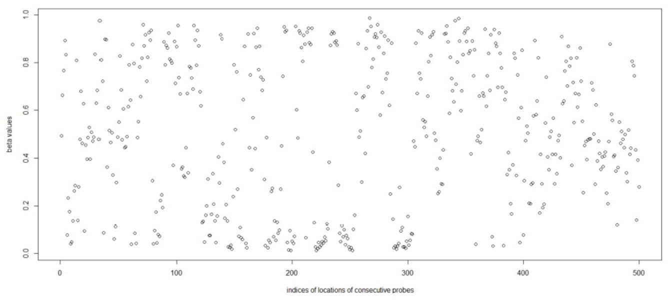

Figure 2. Sample of beta values corresponding to 500 consecutive probes from a randomly chosen chromosome and patient.

RNNs have previously been applied to DNA modification data with various objectives. In [11], the authors built a model combining DNA sequence together with DNA methylation in order to predict missing values in methylation data. The CpG module of the joint model uses a RNN in order to account for correlations between neighbouring CpG sites within and across different cells. Another study from [12] aimed at predicting DNA modifications such as DNA methylation based on training a RNN on raw electric data. In this work we considered DNA methylation events as a time series, where we tested whether methylation events of a position in the DNA are associated to methylation events in its neighbourhood.

The paper is organized as follows. In Section II we provide a brief introduction to recurrent neural networks, and we explain how examples, to be fed into the network, are typically constructed from a sequence of observations. Section III describes the experimental details, which include: 1. the description of the data set, 2. the set of recurrent neural network architectures that are tried on the data set, 3. the construction of training, validation and test sets from the data set, and 4. the performance measures that are used in evaluating the different recurrent neural network architectures. Results are described in Section IV. We first show how a suitable architecture is chosen from the set of considered architectures, and this architecture is then used when addressing the research question. In addition to an RNN performance analysis, we have conducted some additional analyses that are interesting from a biological perspective. Section V concludes the paper.

II. Methods

A. Background on recurrent neural networks

We employ artificial neural networks, more specifically RNNs, to detect structure, if any, in vectors of beta values representing consecutively measured probes. RNNs have shown to achieve state-of-the-art results in many applications with time series or sequential data [13], and enjoy several nice properties such as strong prediction performance as well as the ability to capture long-term temporal dependencies [14]. The distinctive feature of an RNN is that the output of the hidden layer is not only a function of the input at the current iteration, such as in feedforward neural networks, but also of the output of the previous iteration. In fact, since the hidden layer in RNNs takes the output of a previous iteration into account, it takes the output of *all* previous iterations into account, since the output at iteration n depends on the output at iteration $n-1$, which in turn depends on the output at iteration $n-2$, etc. The importance of this feedback loop is that it assigns RNNs a 'memory'. This makes RNNs ideally suited to be used in applications where all information contained in a sequence of values is of potential use to predict the value that comes next. For further details on recurrent neural networks, we refer to [15].

B. Construction of examples

The purpose of a recurrent neural network is often, as in our work here, to find structure in a data set D that consists of an ordered sequence of values $a_1, \ldots, a_n$. A recurrent neural network will learn this structure from given examples. A common procedure to extract examples from D is to consider a subsequence containing a predefined number of values as input, after which this subsequence is shifted by one position to be used as corresponding output. This algorithm is performed over all possible subsequences to obtain the complete set of examples. The predefined number of consecutive values that is used as input is referred to as the window size w. For example, if $w = 3$, then the first three examples $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), (\mathbf{x}_3, \mathbf{y}_3)$ extracted from D are given by

$$\begin{aligned}
\mathbf{x}_1 &= (a_1, a_2, a_3)\\
\mathbf{y}_1 &= (a_2, a_3, a_4)\\
\mathbf{x}_2 &= (a_4, a_5, a_6)\\
\mathbf{y}_2 &= (a_5, a_6, a_7)\\
\mathbf{x}_3 &= (a_7, a_8, a_9)\\
\mathbf{y}_3 &= (a_8, a_9, a_{10})
\end{aligned}$$

Notice that the examples depend on the window size and that, by construction, the number of input neurons and the number of output neurons both equal the window size.

III. Experimental details

A. Data set

The data set we consider is collected from the Cancer Genome Atlas [16]. More specifically, the data set was produced in the project named TCGA-BRCA [17], which studies

Breast Invasive Carcinoma, and we downloaded it using the TCGAbiolinks R package [18] (version 2.8.4). We selected the samples mapped to the GRCh38 reference genome that had been assessed with the Illumina Human Methylation 450k platform.

For our purpose, it is especially relevant that this data set contains beta values for 24 chromosomes (22 non sex chromosomes plus X and Y chromosome) for 1095 patients in 2 conditions (normal and tumor). However, we do not consider the Y chromosome, since the data set contains only a very few beta values for this chromosome. Furthermore, due to missing data, it turns out that only for 96 patients data is available for normal tissue condition, whereas there is data for 782 patient tumors. To limit computation time, we consider the 96 patients in normal condition and the first 96 patients in tumor condition. Some additional experimental analysis has shown us that this number of patients is large enough to obtain stable results (see also Section IV-B). Thus the data set to be analyzed consists of beta values for 23 chromosomes in 96 patients in normal condition and 96 patients in tumor condition. On average, the data set contains about 17 000 measured beta values per chromosome.

B. *Set of RNN architectures*

The parameters of a RNN that need to be given a value by the user are the number of iterations (also called epochs), the window size w and the number of hidden neurons n_h.

The number of epochs should be chosen low enough to limit computation time, but also large enough to ensure convergence of the weights of the RNN. We have performed a concise preliminary analysis on several sets of examples, running a RNN for 150 epochs and computing the error between real output values and values generated at the output neurons. The analysis showed that 40 epochs is sufficient to reach convergence, as all graphs are very similar to the one shown in Fig. 3, which represents the error over the 150 epochs corresponding to one of these experiments. Consequently, all RNNs are run for 40 epochs.

We consider RNNs with the following window sizes:

$$W = \{10, 30, 50, 70, 90\}$$

To determine suitable choices of numbers of hidden neurons, we can rely on some heuristics. One heuristic is to choose n_h as 2/3 of the sum of the number of input neurons and the number of output neurons [19]. Since the number of input neurons equals the number of output neurons, which in turn equals the window size, we try as number of hidden neurons at least

$$n_{h_1} = \text{round}(2/3 \times 2 \times w)$$

where *round* refers to rounding to the nearest integer. We also try several values that are close to the above value n_{h_1}, namely:

$$\begin{aligned} n_{h_2} &= \text{round}(0.9 \times n_{h_1}) \\ n_{h_3} &= \text{round}(0.8 \times n_{h_1}) \\ n_{h_4} &= \text{round}(1.1 \times n_{h_1}) \\ n_{h_5} &= \text{round}(1.2 \times n_{h_1}) \end{aligned}$$

Finally, we also try n_h as 2/3 of the number of input neurons, since some authors claim that the number of hidden neurons should be between the size of the input layer and the size of the output layer [20], and this heuristic restriction is not fulfilled by any of the number of hidden neurons already chosen (they are all larger than the size of the input layer, and thus also larger than the size of the output layer). We denote this value by n_{h_6}:

$$n_{h_6} = \text{round}(2/3 \times w)$$

This gives the following set of numbers of hidden neurons that we will consider:

$$N_h = \{n_{h_1}, n_{h_2}, n_{h_3}, n_{h_4}, n_{h_5}, n_{h_6}\}$$

Thus, in total we consider $|W| \times |N_h| = 5 \times 6 = 30$ different RNN architectures (the notation $|A|$ for a finite set A refers to the number of its elements).

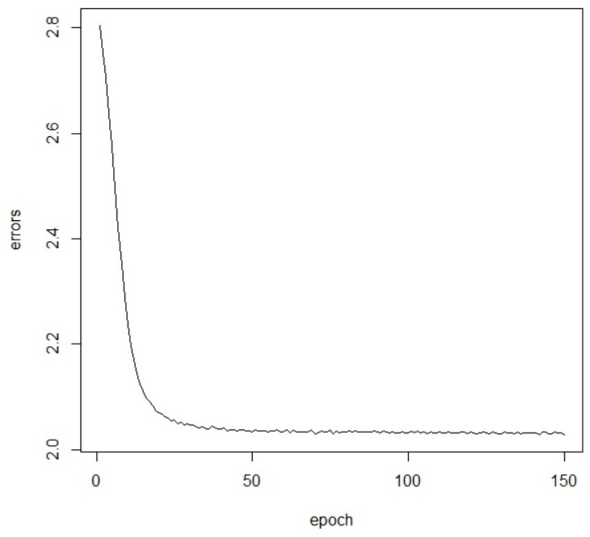

Figure 3. Evolution of the error of a RNN over 150 epochs on some set of examples

C. *Training, validation and test sets*

For each combination of patient, condition and chromosome, we train a separate RNN. A first motivation for this is that the beta values on a specific chromosome of a particular patient in a given condition can have its own characteristics; that is, the distribution of these beta values can be significantly different from the other distributions. Secondly, obtaining different RNNs allows to compare results with respect to the patients, the chromosome and the condition. Finally, since the number of given beta values per chromosome is very large (as mentioned above, there are on average some 17 000 measured beta values for each individual chromosome), each RNN will still have enough examples to learn from.

The beta values for each triplet of the form (patient, chromosome, condition) are converted into examples according to the method described in Section II-B. Each set of examples is then split into a training set, a validation set and a test set, as follows: the first 60% of the examples are used as training examples, the next 20% as validation examples, and the last 20% as test examples. The training set is used to train the selected RNN architectures described in Section III-B for each (patient, chromosome, condition) triplet, the validation set is used to select the best RNN architecture, and the test set allows to evaluate the performance of the selected architecture on the task of detecting structure in the sequences of beta values.

D. *Performance measures to evaluate RNNs*

To evaluate the performance of an architecture on a given test set containing the examples $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$, where $\mathbf{x}_i$ and $\mathbf{y}_i$ refer to input vectors and output vectors, respectively, we will use several performance measures. These measures all evaluate performance in terms of the difference between the real outputs $\mathbf{y}_1, \ldots, \mathbf{y}_m$, and the outputs $\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_m$, generated by the considered RNN. We will use the notation $\mathbf{y}_k(i)$ to refer to the ith component of $\mathbf{y}_k$. By construction it holds that $\mathbf{y}_k(i) = \mathbf{x}_k(i+1)$ for $1 \leq i < w$ (see Section II-B). Therefore, all output components, except for the last one (given by $\mathbf{y}_k(w)$), are also components of the input vector. The consequence is that evaluation in terms of the difference between *all* true and estimated output values might result in a biased view on performance, so we also evaluate performance by restricting to the last output component. For each considered performance measure P we therefore have a first version that evaluates performance by comparing $\mathbf{y}_1(1), \ldots, \mathbf{y}_1(w), \ldots, \mathbf{y}_m(1), \ldots, \mathbf{y}_m(w)$ to $\hat{\mathbf{y}}_1(1), \ldots, \hat{\mathbf{y}}_1(w), \ldots, \hat{\mathbf{y}}_m(1), \ldots, \hat{\mathbf{y}}_m(w)$, and we denote this instance of the performance measure by P_a. But we also consider a second version P_b where the values $\mathbf{y}_1(w), \ldots, \mathbf{y}_m(w)$ are compared to the estimated values $\hat{\mathbf{y}}_1(w), \ldots, \hat{\mathbf{y}}_m(w)$.

We use the following performance measures: Root Mean Square Error (RMSE), Nash-Sutcliffe efficiency (NSE) and Index of Agreement (IA). All measures P are described below in their P_a version; the P_b counterpart is simply obtained by restricting the output values to the one corresponding to the last component.

The RMSE is given by

$$RMSE_a \quad = \quad \sqrt{\sum_{k=1}^{m}\sum_{i=1}^{w}\frac{(\mathbf{y}_k(i) - \hat{\mathbf{y}}_k(i))^2}{mw}}$$

It is clear that the lower the RMSE, the better the network.

The NSE, proposed in [21], is given by

$$NSE_a \quad = \quad 1 - \frac{\sum_{k=1}^{m}\sum_{i=1}^{w}\Big(\mathbf{y}_k(i) - \hat{\mathbf{y}}_k(i)\Big)^2}{\sum_{k=1}^{m}\sum_{i=1}^{w}\Big(\mathbf{y}_k(i) - \overline{\mathbf{y}}\Big)^2}$$

where $\overline{\mathbf{y}}$ refers to the average of all $\mathbf{y}_k(i)$ values. The NSE lies between -∞ and 1.0, where 1 denotes perfect fit. An NSE lower than zero indicates that the simple average $\overline{\mathbf{y}}$ is a better predictor than the obtained approximations $\hat{\mathbf{y}}_k(i)$.

The IA was proposed in [22], and is defined as:

$$IA_a \quad = \quad 1 - \frac{\sum_{k=1}^{m}\sum_{i=1}^{w}\Big(\mathbf{y}_k(i) - \hat{\mathbf{y}}_k(i)\Big)^2}{\sum_{k=1}^{m}\sum_{i=1}^{w}\Big(|\hat{\mathbf{y}}_k(i) - \overline{\mathbf{y}}| + |\mathbf{y}_k(i) - \overline{\mathbf{y}}|\Big)^2}$$

The range of the IA is $[0, 1]$. The higher the value, the better the model, with 1 denoting perfect fit.

IV. Results

A. *Application to a small subset of patients*

Since each triplet (patient, chromosome, condition) is associated with a different data set, there are in total $96 \times 23 \times 2 = 4416$ individual data sets. Ideally, each of these data sets is modeled by the RNN architecture that is most suitable to this specific data structure. To find the best architecture among the selected set of 30 architectures (cf. Section III-B) would require the training of $4416 \times 30 = 132\,480$ RNNs, a prohibitive task in terms of computation time. Therefore, we select the first five patients in normal condition and the first five patients in tumor condition, and train each of the considered architectures only on these patients. In Section IV-B we analyze the performance of the trained networks on the validation sets related to this subsample of patients, and check if there exists one specific architecture that is suitable for the whole subsample. It is then hypothesized that this single architecture will also perform well for all 96 patients in both conditions. The application of this architecture to all patients is performed in Section IV-C. To compare architectures on the subsample of validation sets, in order to find a single suitable one, we only use $RMSE_a$, described in Section III-D. The complete set of performance measures will be used in Section IV-C, where we evaluate the architecture that is presumably suitable for all patients.

B. *Choice of a suitable recurrent neural network architecture*

As outlined in the previous section, we trained the 30 selected RNN architectures on each of the 23 chromosomes on a patient subsample consisting of five normal and five tumor conditions. This implies that each architecture is trained on $23 \times 5 \times 2 = 230$ individual data sets (23 chromosomes, five patients, two conditions). One possible way to compare the performance of the different architectures, is to plot the lowest, the highest and the average $RMSE_a$, over the selected ten different patient conditions and over the chromosomes, and this for each architecture. The result is shown in Fig. 4, where the X-axis denotes the architectures; for example, 50_5 refers to the architecture with $w = 50$ and $n_h = n_{h_5}$. The following observations can be drawn:

- Low window sizes, up to and including 50, perform better than higher window sizes, at least in terms of the average and the highest RMSE (the lowest RMSE is more or less the same for all architectures).
- For all window sizes that are 50 or smaller, the choice of the number of hidden neurons has no effect on the performance.

Another way to evaluate the different architectures is to record the number of times that a given architecture performs best, i.e. the number of times that it has lowest RMSE, and that a given architecture performs worst, i.e. that it has the highest RMSE, and this over all (patient, chromosome, condition) triplets from the subsample. Experiments in this respect show that the first architecture, with $w = 10$ and $n_h = n_{h_1} = 7$, performs best in most cases.

Based on these findings, we decide to use the architecture $(w = 10, n_h = 7)$ as the overall best architecture. We will use this architecture in the next section to analyze the complete data set. Since the best architecture happens to be the one with smallest window size among the selected set of window sizes

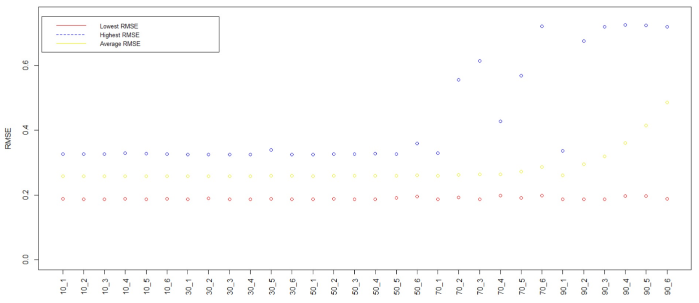

Figure 4. Lowest, highest and average RMSE over the selected patients and over the chromosomes, and this for each architecture

W (which has as smallest element ten), one might wonder if even smaller window sizes perform better. Therefore, we also tried window size five, with varying numbers of hidden neurons, and performed several experiments. However, the obtained results were very similar to the results for window size ten. Therefore, we restrict attention to the $(w = 10, n_h = 7)$ architecture, the results of which are outlined in the next section.

Although our subsample of ten patients is small, we applied the $(w = 10, n_h = 7)$ architecture on the associated subsample of test sets to obtain some preliminary results. However, since this architecture was also applied to the much larger set of 96 patients, and since the results on the corresponding test sets turned out to be very similar to those obtained on the small subsample, we discuss the results only for the large set of 96 patients (cf. next section). The fact that the results for the small subsample and for the large set are similar indicate that the set of 96 patients is large enough to safely assume that the obtained outcomes are significant.

C. Application to all patients

We now evaluate the $(w = 10, n_h = 7)$ architecture on the set of 96 patients in tumor condition and in normal condition. Evaluation is done by training this architecture on each of the $96 \times 23 \times 2 = 4416$ corresponding training sets, and then measuring the performance on the associated test sets using the measures described in Section III-D. As a benchmark, performance is also evaluated with respect to a random permutation of the training sets.

The results are shown in Figs. 5-10. Each figure contains one of the two versions of each of the performance measures P (the version being P_a or P_b, cf. Section III-D). Furthermore, each figure contains two histograms of values of P: one that relates to performance on the test sets after training has been performed on the original training sets, and one where training has been done on a random permutation of the training sets. The figures indicate the following:

- There is a substantial difference between the results related to the permutated training sets and the non-permutated ones. The important implication is that the beta values along a chromosome are non randomly distributed. There is structure in a given sequence of consecutively measured beta values, and recurrent neural networks are able to detect this structure, at least to some extent.

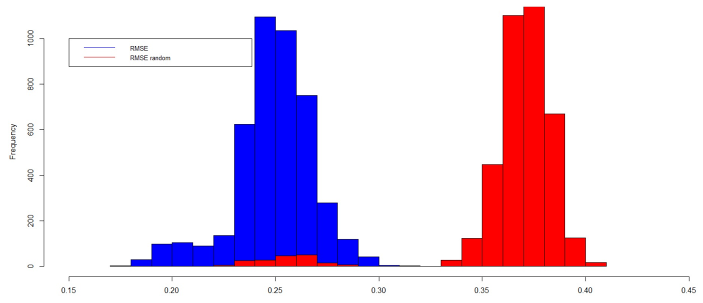

Figure 5. $RMSE_a$

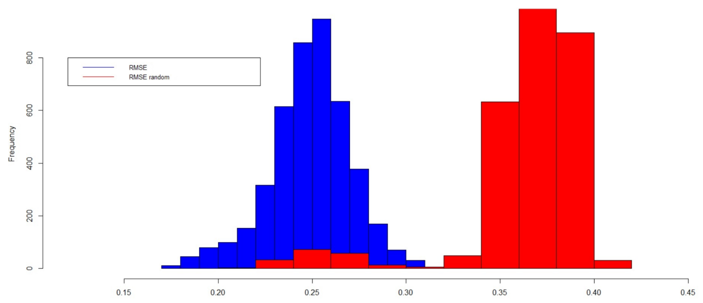

Figure 6. $RMSE_b$

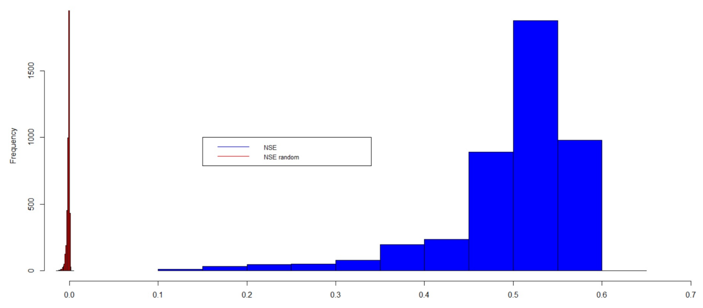

Figure 7. NSE_a

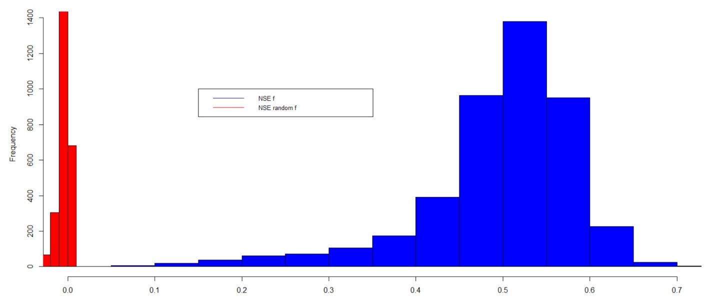

Figure 8. NSE_b

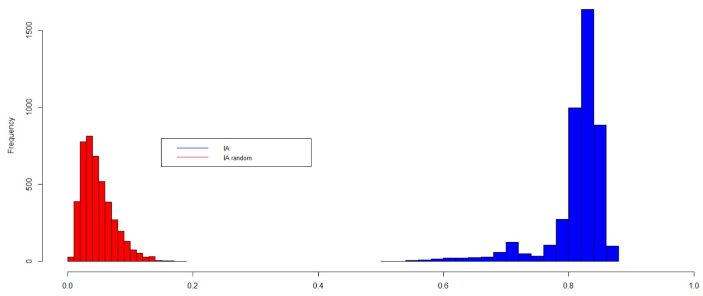

Figure 9. IA_a

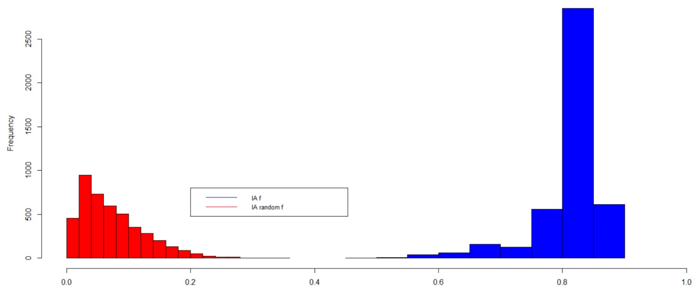

Figure 10. IA_b

- There do not exist theoretically derived thresholds on the values of each of the performance measures to demarcate good and bad models. However, it seems justified to conclude that the values of the performance measures related to the non permutated training sets are acceptable or even very good. Especially the IA attains very high values, with most values higher than 0.8.
- As expected, the P_b version of the performance measures (which takes only the last output component into account) has worse values than its P_a counterpart (taking all output components into account). However, the histograms of both versions are not substantially different (roughly speaking, the histograms of P_b appear to be shifted to the right over only a small distance), implying that the recurrent neural network performs still reasonably well when evaluation is limited to the last output component, i.e. the only component that is truly predicted.

V. Conclusion

In this paper, we have shown the applicability of recurrent neural network analysis for the detection of structure in sequences of measured methylation levels along human chromosomes. The considered task is inherently challenging, due to the fact that a specific methylation value is the result of complex biological processes and because probes where measurements are collected are not equidistantly located. Nevertheless, our work demonstrates that structure is present in sequences of methylation levels, and that recurrent neural networks are able to detect this structure. The obtained results are relevant to both the machine learning and the biological community.

References

[1] P. Jones and D. Takai, "The Role of DNA Methylation in Mammalian Epigenetics," Science, vol. 293, 2001, pp. 1068–1070.

[2] T. Atkinson and M. Halfon, "Regulation of Gene Expression in the Genomic Context," Computational and Structural Biotechnology Journal, vol. 9, 2014, p. e201401001.

[3] B. Jin, Y. Li, and K. Robertson, "DNA Methylation: Superior or Subordinate in the Epigenetic Hierarchy?" Genes Cancer, vol. 2, 2011, pp. 607–617.

[4] E. Greer and Y. Shi, "Histone Methylation: a Dynamic Mark in Health, Disease and Inheritance," Nature Reviews Genetics, vol. 13, 2012, pp. 343–357.

[5] L. Moore, T. Le, and G. Fan, "DNA Methylation and its Basic Function," Neuropsychopharmacology, vol. 1, 2013, pp. 23–38.

[6] M. Eirlich, "DNA Methylation in Cancer: Too Much, but also Too Little," Oncogene, vol. 21, 2002, pp. 5400–5413.

[7] C. Thirlwell et al., "Genome-wide DNA Methylation Analysis of Archival Formalin-fixed Paraffin-embedded Tissue Using the Illumina Infinium HumanMethylation27 BeadChip," BMC Bioinformatics, vol. 52, 2010, pp. 248–254.

[8] M. Frommer et al., "A Genomic Sequencing Protocol that Yields a Positive Display of 5-methylcytosine Residues in Individual DNA Strands," BMC Bioinformatics, vol. 89, 1992, pp. 1827–1831.

[9] P. Du et al., "Comparison of Beta-value and M-value Methods for Quantifying Methylation Levels by Microarray Analysis," BMC Bioinformatics, vol. 11, 2010.

[10] Y. Xin et al., "Role of CpG Context and Content in Evolutionary Signatures of Brain DNA Methylation," Epigenetics, vol. 6, 2011, pp. 1308–1318.

[11] Q. Liu et al., "Detection of DNA base modifications by deep recurrent neural network on Oxford Nanopore sequencing data," Nature Communications, vol. 10, 2017, p. 2449.

[12] ——, "Detection of DNA base modifications by deep recurrent neural network on Oxford Nanopore sequencing data," Nature Communications, vol. 10, 2019, p. 2449.

[13] J. Connor, R. Martin, and L. Atlas, " Recurrent Neural Networks and Robust Time Series Prediction," IEEE Transactions on Neural Networks, vol. 5, 1994, pp. 240–254.

[14] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent Neural Networks for Multivariate Time Series with Missing Values," Scientific Reports, vol. 8, 2018.

[15] W. De Mulder, S. Bethard, and M.-F. Moens, "A Survey on the Application of Recurrent Neural Networks to Statistical Language Modeling," Computer Speech & Language, vol. 30, 2015, pp. 61–98.

[16] C. G. Atlas, https://cancergenome.nih.gov/, last access: August 20, 2020.

[17] TCGA-BRCA, https://portal.gdc.cancer.gov/projects/TCGA-BRCA, last access: August 20, 2020.

[18] TCGAbiolinks, https://bioconductor.org/packages/release/bioc/html/TCGAbiolinks.html, year = last access: August 20, 2020, language = english,.

[19] F. Panchal and M. Panchal, " Review on Methods of Selecting Number of Hidden Neurons in Artificial Neural Network," IJCSMC, vol. 3, 2014, pp. 455–464.

[20] A. Blum, Ed., Neural networks in C++. Wiley, 1992.

[21] J. Nash and J. Sutcliffe, "River flow forecasting through conceptual models, Part I - A discussion of principles," Journal of Hydrology, vol. 10, 1970, pp. 282–290.

[22] C. Willmot, "On the validation of models," Physical Geography, vol. 2, 1981, pp. 184–194.