Classifying Leukemia and Gout Patients with Neural Networks

Guryash Bahra and Lena Wiese

Institute of Computer Science, University of Göttingen
wiese@cs.uni-goettingen.de
g.bahra@stud.uni-goettingen.de

Abstract. Machine Learning is one of the top growing fields of recent times and is applied in various areas such as healthcare. In this article, machine learning is used to study the patients suffering from either gout or leukemia, but not both, with the use of their uric acid signatures. The study of the uric acid signatures involves the application of supervised machine learning, using an artificial neural network (ANN) with one hidden layer and sigmoid activation function, to classify patients and the calculation of the accuracy with k-fold cross validation. We identify the number of nodes in the hidden layer and a value for the weight decay parameter that are optimal in terms of accuracy and ensure good performance.

1 Introduction

In medical data analysis, machine learning is a common procedure used for classification of patients suffering from different diseases. In this paper, our focus is on the classification of patients suffering from either leukemia or gout. One of the common factors in leukemia and gout diseases is the uric acid signature in blood. Uric acid concentration in a healthy human in developed countries ranges from 3.5 mg/dl (in infants) to about 6 mg/dl (in adults) [9, 17, 1]. In patients suffering from gout or leukemia the uric acid concentration increases more than the normal range and is therefore regularly monitored and treated. In gout, a combination of genetic mutations and environmental factors causes uric acid concentration to increase. This further results in formation of uric acid crystals which precipitates into joints and causes painful arthritis [9]. As for leukemia, turnover of white blood cells increases the uric acid concentration. The two diseases have different pathophysiology, which – in combination with their treatments – results in different signatures of uric acid concentrations. In this article, supervised learning is performed on uric acid measurements to classify the two diseases, leukemia and gout.

1.1 Machine Learning Techniques

Machine learning involves building of models from the given dataset which can be utilized to make future predictions. This process is executed in two phases: (i)

calculation of unknown dependencies from the input dataset and (ii) prediction of new outputs using those dependencies.

The two common types of machine learning are supervised and unsupervised learning. Supervised learning involves a labelled set of input data to predict the output. In contrast, unsupervised learning involves unlabelled data; because of this, there is no designated output, which implies that the learning model has to identify patterns in the input data. The work in this article is based on a classification problem of supervised learning, which involves categorizing the data into finite classes. More precisely, uric acid concentrations are classified into two classes, leukemia and gout.

1.2 Objectives

The main objective of our work is to perform supervised machine learning, with neural networks, to estimate the accuracy of the system to distinguish between the patients with either gout or leukemia. To achieve this objective, several tasks have to be performed and these are summarized as follows:

- To identify files to be used from Medical Information Mart for Intensive Care (MIMIC) dataset [6, 5].
- To identify data (patients with either gout and leukemia and their uric acid signatures) required for the study.
- To perform supervised learning with 3-fold cross validation on uric acid measurements.

2 Related Work

A wealth of research is done in healthcare with the use of machine learning. We survey some approaches here. In [2] the authors propose an algorithm, BIGBIOCL, for the classification of DNA Methylation Datasets for identifying cancer drivers in patients suffering from either breast, kidney or thyroid carcinomas. Supervised learning is used in [16] for cardiovascular risk prediction; the following algorithms are used: random forest, logistic regression, gradient boosting machines and neural networks. The authors conclude that neural networks performed better than the rest. In [8], the authors review different supervised learning methods, Artificial Neural Networks (ANNs), Bayesian Networks (BNs), Support Vector Machines (SVMs) and Decision Trees (DTs), for prognosis of cancer and its prediction. Their paper even highlights the case studies used for machine learning tools to predict cancer susceptibility, cancer recurrence and cancer survival. In [9] unsupervised feature learning is used on uric acid signatures before supervised learning is applied to classify the patients into gout or leukemia. The work described in their paper is implemented on data taken from Electronic Medical Records [13]. In their paper, it is mentioned that the data is noisy, sparse and irregular, therefore, it is smoothened with the use of Gaussian process regression. On this data, deep learning is performed with the use

of Sparse Autoenconders. Furthermore, the features learned from first and second layers of Sparse Autoencoders, are then utilized for the supervised learning classification task using Logistic Regression.

These papers highlight the use of machine learning in healthcare. In particular, classification of diseases is sought-after, and neural networks are widely used for classification.

3 Data Set

MIMIC-III is the third iteration of a large clinical database MIMIC. It comprises of the medical data of patients admitted to critical care units, Coronary Care Unit (CCU), Cardiac Surgery Recovery Unit (CSRU), Medical Intensive Care Unit (MICU), Neonatal Intensive Care Unit (NICU), Surgical Intensive Care Unit (SICU) and Trauma Surgical Intensive Care Unit (TSICU), at the Beth Israel Deaconess Medical Center in Boston [6, 5]. According to Goldberger et al. [5], the current version of the database is 1.4 as of September 4, 2016, and consists of health-related records of de-identified 46,520 subjects out of which 38,645 are adults and 7,875 are neonates. The patients are de-identified according to Health Insurance Portability and Accountability Act (HIPAA) standards, which involved removal of 18 fields, such as patients' name, telephone numbers, addresses, etc., as listed in HIPAA. It also involved shifting of the dates, including date of birth, by a random offset, as directed by the HIPAA. The database not only includes the information of the vital sign measurements, medicines administered, laboratory measurements, fluid balance, imaging reports, out-of-hospital mortality but also patients' demographics, nurses' and physicians' notes, procedure and diagnostic codes, and more. Note that the MIMIC database was prepared by compiling data from two data sources, CareVue and Metavision Intensive Care Unit (ICU) databases, used at the hospital.

3.1 Dataset Identification

There are 26 Comma Separated Values (CSV) files in the MIMIC-III data set. And, out of those 26 files, the following 4 files are considered for the case study:

- `D_ICD_DIAGNOSES`: gives the ICD-9 codes for gout and leukemia diagnoses
- `DIAGNOSES_ICD`: identifies the hospital admissions and patients suffering from gout and leukemia
- `D_LABITEMS`: gives the ID for uric acid signatures
- `LABEVENTS`: gives the data about uric acid measurements of the patients identified with gout and leukemia

There are 78 ICD-9 codes for leukemia and 11 for gout identified from the `D_ICD_DIAGNOSES` table. Then, from the `DIAGNOSES_ICD` table, a total of 2,837 hospital admissions are identified with above diagnoses, and out of which 618

hospital admissions are for leukemia and 2,219 are for gout. These many admissions correspond to 2,259 patients or unique[1] SUBJECT_IDs, out of which 454 patients suffered from leukemia and 1,805 suffered from gout. Furthermore, 22 patients are common for both the diagnoses, and after removing IDs of those patients, a total of 2,773 hospital admissions are identified for the patients suffering from either leukemia (584 HADM_IDs) or gout (2,189 HADM_IDs) but not both. The 2,773 admissions correspond to 2,215 patients, out of which 1,783 patients suffered from gout and 432 from leukemia. Moreover, the hospital admissions are reduced from 2,773 to 1,076 as there are no records of uric acid measurements for those patients. The number, 1,076, further decreased to 640 as there are no uric acid observations corresponding to those admissions. And finally, these 640 unique admissions correspond to 567 unique patients, out of which 311 suffered from gout and the remaining 256 patients suffered from leukemia.

As for the uric acid signatures, 3 IDs are identified from the D_LABITEMS table. And, correspoding to those IDs, 19,906 observations representing uric acid measurements are identified from the LABEVENTS table. Then, 19,906 observations reduced to 7,076, as the removed observations didn't correspond to the identified SUBJECT_IDs. Furthermore, 7,076 observations reduced to 5,665, as those observations did not correspond to the identified hospital admission IDs.

3.2 Dataset Creation

The data, i.e, uric acid concentrations, are arranged into 567 sequences, grouped according to the patient IDs. These sequences are then broken down to a size of 17 values per row: the first two values are the label (1 for leukemia and 0 for gout) and patient ID, and the remaining 15 values are the uric acid concentrations. Note that the sizes of sequences are unequal. Therefore, there can be multiple rows of data of a single patient, and each row in turn is treated as a new sequence. And, for sequences with less than 15 data values, 0 value is used for the remaining part of the sequence. This resulted in a total of 813 sequences. The sequences are then shuffled with the use of `sample` function provided by R [15].

3.3 K-Fold Cross-Validation

To create training and testing sets used for the learning, the k-fold cross validation method is employed. In k-fold cross validation, the data is randomly divided into k subsets of equal size and a single subset is referred to as fold. Of the k folds, $k-1$ folds are combined to form the training set and the remaining fold is used as the testing set, and the accuracy is calculated for the training and the testing sets which describes the stability of the model. This is then repeated for k iterations, and for every iteration, the testing set comprises of a fold used exactly once. Moreover, to use the model for new predictions and to estimate

[1] In R, *duplicated* function with logical negation operator (!) is used to find unique SUBJECT_IDs.

the overall accuracy of the model, consider the classifier for which the highest accuracy is achieved for the testing set.

As described in the previous paragraph, first the data is divided into equal subsets. Therefore, 813 sequences (from Section 3.2) are divided into 3 equal subsets of size 271 each. Then, supervised learning (as described in Section 4.2) is performed on these subsets for three iterations. For each iteration, the testing set is formed with a single subset used exactly once and the remaining two subsets are used as the training set (of size 542). The accuracy (calculated as in Section 4.2) is reported for every iteration.

4 Method

4.1 Neural Networks

Neural networks are one of several supervised learning classification techniques, and are based on the concept of perceptrons [7, 12]. Neural networks are conceptualized as in the following paragraphs.

Forward Propagation. In this work, a 3-layered neural network is used, where the first layer, L_1, is the input layer, the second layer, L_2, is the hidden layer, and the last layer, L_3, is the output layer. Note that the output of one layer is the input of the next layer, and there are s_l number of nodes in each layer l.

Activation unit, a_i^l, is used to define the output of the ith unit in layer l. Therefore, for the L_1 layer, $a_i^{(1)} = x_i$, where x is the input data. As for the layers L_2 and L_3, the nodes are computational and therefore, activations are calculated as a function of input vector x, weights matrix W and a bias vector b, as given by the Equation 1.

$$a_i^{(l)} = f\left(W_{ij}^{(l-1)} a^{(l-1)} + b_i^{(l-1)}\right) \tag{1}$$

In Equation 1, W_{ij}^{l-1} is the weight or the parameter of the connection between the jth unit in layer $l-1$ and the ith unit in layer l, bias unit b_i^l corresponds to the ith unit in layer l. Function $f : R \to R$ is the activation function, and is usually defined with sigmoid function as shown by the Equation 2; it produces an output ranging from $(0, 1)$.

$$f(z) = \frac{1}{1 + \exp(-z)} \tag{2}$$

As the nodes are calculated starting from the layer L_1 up to layer L_3 in the network, this step is called forward propagation.

An important identity to note is that the derivative $f'(z)$ of sigmoid function $f(z)$ (in Equation 2) is given by the Equation 3, and is used later in the section.

$$f'(z) = f(z)(1 - f(z)) \tag{3}$$

Cost Function. The cost function we use is known as the squared-error cost function. For a given training set $\left\{(x^{(1)}, y^{(1)}), .., (x^{(r)}, y^{(r)}), .., (x^{(m)}, y^{(m)})\right\}$ of m training examples, the cost function is given as Equation 4,

$$J(W,b) = \left[\frac{1}{m}\sum_{r=1}^{m} J\left(W, b; x^{(r)}, y^{(r)}\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_{l-1}}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ij}^{(l)}\right)^2 \tag{4}$$

where the first term is the average sum-of-squares error term and the second term is the *regularisation* term (or the *weight decay* term) which decreases the value of the weights and avoids overfitting. Note that the regularisation term is not applied to bias b [10]. And, λ in Equation 4 is the *weight decay parameter*.

To minimize the cost function $J(W,b)$ as a function of weights matrix W and bias vector b, every parameter $W_{ij}^{(l)}$ and $b_i^{(l)}$ is initialized to a random value near to 0. And then an optimization algorithm, for example gradient descent, is applied for minimization.

Gradient Descent Algorithm. Gradient descent updates the parameters per iteration as mentioned in Equation 5, where α is the learning rate and $\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)$ and $\frac{\partial}{\partial b_i^{(l)}} J(W,b)$ are derivatives of the overall cost function $J(W,b)$. Parameters are updated as

$$\begin{aligned} W_{ij}^{(l)} &= W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b) \\ b_i^{(l)} &= b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W,b) \end{aligned} \tag{5}$$

Back Propagation Algorithm. To calculate the partial derivates, $\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)$ and $\frac{\partial}{\partial b_i^{(l)}} J(W,b)$, as mentioned in Equation 5, back propagation algorithm is applied and is described as follows. The first step is to calculate the error term δ for every computational node, starting from layer L_3, or the output layer, to layer L_2, in the network. As stated in [10], "the error term measures how much the node is responsible for any errors in the output". Finally, the parameters, W and b, which minimize the cost function, $J(W,b)$, are calculated with the gradient descent algorithm.

4.2 Implementation of Supervised Learning using Neural Networks

The steps carried out to perform supervised learning (in Octave [4]) are described in the following paragraphs.

Define s_l and λ. To begin, the nodes s_l in all the layers (in Equation 4) and the weight decay parameter λ (in Equation 4) are defined. According to our dataset (from Section 3.2), the input size is 15, that is, the number of nodes (s_1) in layer 1 (L_1) is 15. This is because, from Section 3.2, out of 17 values per row, 15 values are the uric acid measurements. The number of output nodes is 1, as the label (leukemia or gout) per row is single-valued (from Section 3.2). The number of nodes in hidden layer s_2 and the value of weight decay parameter λ is varied in order to assess the changes (positive, negative, or no change) in the result.

Initialization of Weights W and Biases b. The weights in matrix W are to be initialized to a value close to 0 [3] and therefore, are randomly initialized from the interval [-0.5, 0.5] [11]. The biases b are initialized to 0.

Cost Function Calculation. The calculation of cost function $J(W, b)$ (as in Section 4.1) is implemented according to the pseudo-code below.

1. **compute activation matrix** $a^{(l)}$
   Activation matrix is computed according to Equation 1.
   a. for layer $l = 2$: $a^{(2)} = f\left(W^{(1)} * (data)^T\right) + b^{(1)}$
   Note: $(data)^T$ is transpose of $data$ and f is sigmoid function (described by Equation 2).
   b. for layer $l = 3$: $a^{(3)} = f(W^{(2)} * a^{(2)}) + b^{(2)}$
2. **calculate weight regularisation term**
   All the weights in the layers, $l = 1, 2$, are added, and multiplied with $\frac{\lambda}{2}$ (as described by Equation 4).
3. **calculate cost function**
   The cost function $J(W, b)$ is determined.
   a. $cost = 0$
   b. for $i = 1 \ldots m$: $cost = cost + \frac{1}{2}(a^{(3)}(i) - y(i))^2$
   c. $J(W, b) = \frac{1}{m} cost + weight\ regularisation\ term$
   Note: the number of training examples (from Section 3.3) is $m = 542$, $cost$ is a temporary variable, y is label with value either 0 (for gout) or 1 (for leukemia), and $weight\ regularisation\ term$ is from the previous step.

Subsequently, the Limited-memory-Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method is used for minimization of the cost function. The L-BFGS algorithm is implemented by the function minFunc (by Mark Schmidt [14]).

Calculation of Derivatives. The calculation of partial derivatives, $\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b)$ and $\frac{\partial}{\partial b_i^{(l)}} J(W, b)$, as described in Section 4.1, is implemented according to the pseudo-code below.

1. **compute error terms** $\delta^{(l)}$
   a. for the output layer $l = 3$: $\delta^{(3)} = -\left(y - a^{(3)}\right) \cdot a'^{(3)}$
   Note: a' is the derivative of the sigmoid activation function. On using the identity of sigmoid function, as in Equation 3, $\delta^{(3)}$ in Step **a** is defined as: $\delta^{(3)} = -\left(y - a^{(3)}\right) \cdot a^{(3)} \cdot \left(1 - a^{(3)}\right)$
   b. for layers $l = 2$: $\delta^{(2)} = (W^{(2)})^T \delta^{(3)} \cdot a'^{(3)}$
   Note: $(W)^T$ is transpose of weight matrix W and a' is again the derivative of the sigmoid activation function.
2. **compute partial derivatives of the overall cost function** $J(W, b)$
   a. for layer $l = 2$: $\frac{\partial}{\partial W^{(2)}} J(W, b) = \frac{1}{m} \delta^{(3)} (a^{(2)})^T + \lambda W^{(2)}$ and $\frac{\partial}{\partial b^{(2)}} J(W, b) = \frac{1}{m} \delta^{(3)}$
   b. for layer $l = 1$: $\frac{\partial}{\partial W^{(1)}} J(W, b) = \frac{1}{m} \delta^{(2)} data + \lambda W^{(1)}$ and $\frac{\partial}{\partial b^{(1)}} J(W, b) = \frac{1}{m} \delta^{(2)}$
   Here, for the input layer L_1, $a^{(1)} = data$ as mentioned in Section 4.1. Note that in Steps **a** and **b**, δ is the error term computed in the previous step.

Update parameters *W* and *b*. The weights matrix W and the bias vector b for layers L_1 and L_2, which minimize the cost function $J(W, b)$, are recalibrated after every iteration of the L-BFGS algorithm. In other words, W and b are equal to final values of the partial derivatives of the overall cost function, $\frac{\partial}{\partial W^{(l)}} J(W, b)$ and $\frac{\partial}{\partial b^{(l)}} J(W, b)$ respectively.

Calculate Accuracy. To calculate the accuracy of the machine learning model on the training and the testing sets, forward propagation (as described in Section 4.1) is performed such that for layers $l = 2, 3$: $a^{(l)} = f(W^{(l-1)} * a^{(l-1)}) + b^{(l-1)}$. Note that the activation matrix of layer 1, $a^{(1)} = data$ (either training or testing), f is sigmoid function, and W and b are calculated using the L-BFGS algorithm.

Then, the activation vector of the output layer, $a^{(3)}$, is used to assign labels to the data (either training set or testing). If the value of an element in $a^{(3)}$ is greater or equal to 0.5, then, label 1 (corresponding to leukemia) is assigned to the element, else label 0 (corresponding to gout) is assigned. These assigned labels then form the prediction vector of size 542×1 in case of training set and of 271×1 in case of testing set.

Each element in prediction vector is then compared with the corresponding actual label of the data. If the labels are the same, 1 is assigned to a comparison vector; if labels are not the same, then 0 is assigned to the comparison vector. Furthermore, the average is calculated for the comparison vector, which is of size 542×1 in case of training set and of 271×1 in case of testing set. The average multiplied with 100 gives the accuracy of the model in percent.

5 Results

This section describes the results of supervised learning. The results represent the neural network model's ability to distinguish between patients suffering from

either gout or leukemia. The accuracies are determined for 5 different cases, resulting from the change in the values of weight decay parameter λ and number of hidden layer nodes s_2 (as in Section 4.2). Case 1 is where $s_2 = 10$ & $\lambda = 0.0001$; Case 2 is $s_2 = 25$ & $\lambda = 0.0001$; Case 3 is $s_2 = 5$ & $\lambda = 0.0001$; Case 4 is $s_2 = 5$ & $\lambda = 0.00001$; Case 5 is $s_2 = 5$ & $\lambda = 0.000001$.

The accuracy is computed three times (I1-I3) per case; this is because weights in W are randomly initialized (see Section 4.2) and therefore, give slightly different values for each iteration. For the final accuracy with 3-fold cross validation (measured in percent), the accuracies are averaged out (Avg).

Case		**Cross Validation on Original Dataset**							
		test set: fold 1		test set: fold 2		test set: fold 3		Average	
		train	test	train	test	train	test	train	test
Case 1: $s_2 = 10$, $\lambda = 0.0001$	I1	88.74	84.50	90.59	81.54	91.32	78.22	90.22	81.42
	I2	88.74	86.34	90.03	83.39	90.77	76.01	89.84	81.91
	I3	88.56	85.97	90.40	83.39	90.95	78.22	89.97	82.52
	Avg	**88.68**	**85.60**	**90.34**	**82.77**	**91.01**	**77.48**	**90.01**	**81.95**
Case 2: $s_2 = 25$ $\lambda = 0.0001$	I1	89.48	85.23	90.77	82.65	91.51	78.22	90.58	82.03
	I2	89.48	86.71	91.32	80.81	91.69	79.70	90.83	82.41
	I3	89.48	85.23	91.14	81.91	91.88	78.96	90.83	82.03
	Avg	**89.48**	**85.72**	**91.07**	**81.79**	**91.69**	**78.96**	**90.74**	**82.15**
Case 3: $s_2 = 5$ $\lambda = 0.0001$	I1	85.60	84.87	87.82	82.28	89.66	81.91	87.69	83.02
	I2	85.60	85.97	88.56	83.02	89.66	78.59	87.94	82.52
	I3	85.42	85.23	88.37	83.02	89.29	78.22	87.69	82.15
	Avg	**85.54**	**85.35**	**88.25**	**82.77**	**89.53**	**79.57**	**87.77**	**82.56**
Case 4: $s_2 = 5$ $\lambda = 0.00001$	I1	86.71	86.71	88.37	83.02	90.40	79.33	88.49	83.02
	I2	87.26	83.02	87.26	83.39	86.71	83.02	87.07	83.14
	I3	88	87.08	88.56	80.81	89.66	77.49	88.74	81.79
	Avg	**87.32**	**85.60**	**88.06**	**82.40**	**88.92**	**79.94**	**88.1**	**82.65**
Case 5: $s_2 = 5$ $\lambda = 0.000001$	I1	86.16	85.60	85.60	83.39	87.63	80.81	86.46	83.26
	I2	85.42	85.60	88	83.02	90.22	78.59	87.88	82.40
	I3	86.16	85.60	88.19	82.65	90.22	77.49	88.19	81.91
	Avg	**85.91**	**85.60**	**87.26**	**83.02**	**89.35**	**78.96**	**87.51**	**82.52**

Table 1. Accuracies (in %) of neural network with 3-folds cross validation.

From Table 1 we can observe that (although the highest average training set accuracy is 90.74 in case 2) the highest average testing set accuracy is 82.65 in case 4. Hence the settings of case 4 seem to be the best out of all cases.

We implemented the steps to carry out the supervised learning presented in the previous sections in Octave [4]. Data preprocessing was done in R. We measured the runtime of the neural network model learning phases for all 5 cases. Executions are run on a Ubuntu 16.04.2 LTS system with 8GB RAM, 64 bit intel core i5 processor and 1TB of hard disk. Table 2 shows that case 4 indeed provides a good average execution time.

Case	**Runtime** (s)
Case 1: when $s_2 = 10$ & $\lambda = 0.0001$	30.03
Case 2: when $s_2 = 25$ & $\lambda = 0.0001$	179.76
Case 3: when $s_2 = 5$ & $\lambda = 0.0001$	16.08
Case 4: when $s_2 = 5$ & $\lambda = 0.00001$	22.07
Case 5: when $s_2 = 5$ & $\lambda = 0.000001$	27.72

Table 2. Execution time (in seconds) for k-fold cross validation.

6 Discussion and Conclusion

In our experiment a neural network was designed with one hidden layer to classify gout and leukemia patients based on their uric acid measurements. The optimal settings that we identified comprise the lowest number of nodes in the hidden layer as well as a medium value for the weight decay parameter. These settings also provide a good runtime.

Our experiment starts with identifying the tables from the MIMIC-III database. Then, from those tables, patients with gout and leukemia diseases, and their corresponding uric acid measurements, are identified. The data is then cleansed, and is further used for supervised learning. The neural network (in Sections 4.1 & 4.2) for the supervised learning is designed using one hidden layer and sigmoid activation function. Accuracy, is then calculated to measure the effectiveness of the model.

In future work we will investigate whether using more layers improves the accuracy. Using tanh activation function, instead of sigmoid activation function, and observing its effect on the accuracy is another enhancement we plan to study. Moreover, verification with a larger dataset will be necessary to validate our results.

References

1. Alvarez-Lario, B., MacArron-Vicente, J.: Is there anything good in uric acid? QJM: An International Journal of Medicine 104, 1015–1024 (2011)
2. Celli, F., Cumbo, F., Weitschek, E.: Classification of large dna methylation datasets for identifying cancer drivers. Big Data Research (2018)
3. Changhau, I.: Weight Initialization in Artificial Neural Networks (2017), https://isaacchanghau.github.io/2017/05/24/Weight-Initialization-in-Artificial-Neural-Networks/
4. Eaton, J.W., Bateman, D., Hauberg, S., Wehbring, R.: GNU Octave version 4.2.0 manual: a high-level interactive language for numerical computations (2016), http://www.gnu.org/software/octave/doc/interpreter
5. Goldberger, A.L., Amaral, L.A.N., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals, vol. 101. Circulation Electronic Pages (13 June 2000)

6. Johnson, A.E., Pollard, T.J., Shen, L., wei H. Lehman, L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L.A., Mark, R.G.: MIMIC-III, a freely accessible critical care database. Scientific Data (2016)
7. Kotsiantis, S.B.: Supervised machine learning: A review of classification techniques. In: Maglogiannis, I.G., Karpouzis, K., Wallace, M. (eds.) Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies, pp. 3–24. IOS Press (2007)
8. Kourou, K., Exarchos, T.P., Exarchos, K.P., Karamouzis, M.V., Fotiadis, D.I.: Machine learning applications in cancer prognosis and prediction. Computational and Structural Biotechnology Journal 13, 8 – 17 (2015)
9. Lasko, T.A., Denny, J.C., Levy, M.A.: Computational Phenotype Discovery Using Unsupervised Feature Learning over Noisy, Sparse, and Irregular Clinical Data. PLOS ONE 8(8) (2013)
10. Ng, A., Ngiam, J., Foo, C.Y., Mai, Y., Suen, C.: UFLDL Tutorial (2013), http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial
11. Nguyen, D., Widrow, B.: Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In: IJCNN International Joint Conference on Neural Networks. vol. 3 (1990)
12. Nielsen, M.A.: Neural Networks and Deep Learning. Determination Press (2015), http://neuralnetworksanddeeplearning.com/
13. Roden, D.M., Pulley, J.M., Basford, M.A., Bernard, G.R., Clayton, E.W., Balser, J.R., Masys, D.R.: Development of a large-scale de-identified DNA biobank to enable personalized medicine, vol. 84. Clinical Pharmacology and Therapeutics (21 May 2008)
14. Schmidt, M.: minFunc: unconstrained differentiable multivariate optimization in Matlab (2005), https://www.cs.ubc.ca/ schmidtm/Software/minFunc.html
15. Team, R.C.: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2014), https://www.r-project.org/
16. Weng, S.F., Reps, J., Kai, J., Garibaldi, J.M., Qureshi, N.: Can machine-learning improve cardiovascular risk prediction using routine clinical data? PLOS ONE 12(4), 1–14 (2017)
17. Wilcox, W.: Abnormal serum uric acid levels in children, vol. 128. The Journal of Pediatrics (1996)