Improving Credit Client Classification by Using Deep Neural Networks?

Klaus B. Schebesch[1(✉)] and Ralf W. Stecking[2]

[1] Vasile Goldiş Western University, Arad, Romania
kbschebesch@uvvg.ro
[2] Carl von Ossietzky University, Oldenburg, Germany
ralf.stecking@uni-oldenburg.de

Abstract. Credit client classification which is useful for building models to forecast probable defaulting behavior is of obvious practical importance and interest. There are many technical alternatives in order to achieve this goal. A huge variety of *statistical learning* and *nature inspired* black-box techniques where used to search through different classification templates and to apply many kinds of numerical adaptation. Combinatorial rule complexity was traded for massive parameterization. Retraction to computationally less demanding techniques followed and was generally well received by financial practitioners. Explainable modeling also grew in demand. This promised to change with the arrival of deep Neural Networks (dNN), a revival of the repeatedly deprecated classical neural nets, but now with essential improvements in its computational structures at the implementation side and also with some conceptual advances. Using two credit client data sets of different sizes and structure we show which out-of-sample performance measure can be consistently improved by dNN and to what extend manual tuning and modeler's decisions may be delegated to automatic modeling process. We compare these new models to our best performing models published in the past, which were obtained using the same input data.

Keywords: Classification models · Deep Neural Networks · ROC-AUC criterion · Client behavior

1 Introduction

1.1 General Overview

Credit client classification which is useful for building models to forecast probable defaulting behavior is of obvious practical importance and interest [1,4]. The general underlying concept is not restricted to financial credit but applies to many similar situations, where client action is conditioned on class or group affiliation determined by some multivariate data analysis. A huge variety of *statistical learning* and *nature inspired* black-box techniques where used to search through different classification templates and to apply many kinds of numerical

© Springer Nature Switzerland AG 2021

adaptation. Combinatorial rule complexity was traded for massive parameterization. Retraction to computationally less demanding techniques like Logistic Regression and *moderately expensive* Support Vector Machines (SVM, [19]) followed and was generally well received by financial practitioners. Explainable modeling also grew in demand. Over a broad collection of data sets, at least with regard to primary forecasting goals like out-of-sample hit-rate, accuracy, etc., no clear winner emerged, as the relative superiority of single model classes (like Bagging, Random Forests, SVM, CART, ...) was never truly dramatic and data-set dependent, possibly also as a consequence of the prevailing relative sparseness of the credit client data. Considerable *hand-tuning* of models was still needed. This promised to change with the arrival of deep Neural Networks (dNN), a revival of the repeatedly deprecated classical neural nets, but now with essential improvements in its computational structures at the implementation side and also with some conceptual advances. Using two credit client data sets of different sizes and structure we show which out-of-sample performance measure can be consistently improved by dNN and to what extend manual tuning and modeler's decisions may be delegated to automatic modeling process. We compare these new models to our best performing models published in the past, which were obtained by using the same input data.

1.2 Research Using Deep Machine Learning in Credit Client Modeling

In recent years a few research articles appeared in the literature about credit client classification with deep neural networks. Without claiming completeness, but in attempting to cover a wider spectrum of approaches, we include the following:

1. Credit Risk Analysis Using Machine and Deep Learning Models [2].
2. Predicting mortgage default using convolution neural networks [10].
3. Credit scoring with a feature selection approach based deep learning [6].
4. A novel multistage deep belief network based extreme learning machine ensemble learning paradigm for credit risk assessment [20].
5. A Classification Restricted Boltzmann Machine for a comprehensible credit scoring model [18].

These contributions are mainly directed towards analyzing and improving out-of-sample forecasting performance over that of more traditional, and, more often than not, computationally much cheaper methods. The approaches are experimental, whereby effective practical application in the financial industry often hinges on explainable forecasts (transparent rules, which are difficult or impossible to extract from the dNN models).

In the remaining sections we describe our credit client data using a more general view on applied classification problems, hereby highlighting important features and restrictions on data use and accessibility (Sect. 2). Next, we comment on the more recent revival of neural networks, especially in the context of deep machine learning (Sect. 3). Our new results on credit client classification in the light of deep neural networks is presented in Sect. 4 for a small empirical data set and in Sect. 5 for a large data set, respectively. The results are compared to those obtained in extensive past work using the more traditional approaches. Section 6 further explains some specific results. Finally, in Sect. 7 we conclude and present some outlook concerning future applications.

2 Our Credit Scoring Data for Classification Problems

2.1 Empirical Approach – Credit Data Set Description

Formally, every credit client case contains a number of $N > 0$ credit clients. Each client is described by m_0 raw data features. To every feature vector x_i, $i = 1, ..., N$ has an associated label $y_i \in \{-1, +1\}$, with $y_i = 1$ if the client i was eventually *defaulting* (i.e. could not reimburse the outstanding credit). Hence the raw data set is given by the matrix (x, y) which will be split up into a training and test set by the modeler.

In our computational experiments we use two credit client data sets of different size and structure, a small and a big data set, respectively

Data Set I:

- German Credit Data from UCI repository: information about N=1000 clients
- 300 defaulting credit clients: *default rate is* 30%
- Credit client information: m_0=20 input variables, e.g. age, personal status and sex, property, credit amount, savings account and bonds, etc.

Data Set II:

- N=139951 clients for a building and loan credit
- 3692 defaulting credit clients: *one year default rate is* 2.6%
- Credit client information: m_0=12 input variables, e.g. loan–to–value ratio, repayment rate, amount of credit, house type, etc.

After transforming the raw data set into a standardized numerical format, the effective number of input features m may grow (i.e. $m > m_0$, most often by using a binary representation for categorical variables)

2.2 Experimental Setup

Our experimental setup searches for deep Neural Network models which forecast the probable credit defaulting behavior of new clients. A newly arriving client has data $(x, -)$. Hence, the y-part is not yet known, because it would label his/her future behavior and hence represents itself the required prediction target. This is simulated by means of reserving a test data set $N = N_{train} + N_{test}$. The performance of a model is measured over the test data set by the *Receiver Operating Characteristic* (ROC curve) and the so called AUC (area under curve).

1. Repeated random sub-sampling validation: Divide both credit client data sets **randomly** into *training* and *test* sets with $N_{train} : N_{test} = 2{:}1$.
2. Repeat ten times:
   (a) Model building for every possible training/test set combination: *Linear* and *RBF-SVM* (SVM using the nonlinear radial basis function kernel) plus Logistic regression additionally built as a benchmark model
   (b) Add **(deep) Neural Networks**
   (c) Report test set results

3 Deep Machine Learning: Renaissance of Messy Neural Networks

Being composed of basically simple, highly repetitive structures neural networks may be viewed as fundamental building blocks of machine learning. In principle this was already known in the 1950s and 60s. Their input-output behavior is determined by training data via adapting a large number parameters by minimization of some error (or energy) function. From a structural point of view these networks are highly scalable (up to millions of free parameters with today's average-level computer technology). Furthermore they usually have a completely differentiable structure making them amenable to adaptation ("learning") by standard gradient descent. Being heavily parameterized seems at first to contradict basic principles of statistical modeling, e.g. that the number of examples should be (much) bigger then the number of parameters, and henceforth of input feature dimensions. However, as whole branches of neural networks can be effectively deactivated by appropriate combined action of parameters and inputs, the true capacity of a network is more difficult to assess. But all these attractive features were almost annihilated by the excessive amount of computation required to train even modest-sized networks and henceforth they were considered as "messy" and lost much of their popularity for data-oriented modeling such as classification. However, as classification was to become evermore important, and as it is a natural application of neural networks, namely by separating input space via hyperplanes and compositions thereof, they were never put completely aside by the data-modeling community.

3.1 Deep Neural Networks

Neural networks resurfaced more recently after revisiting older ideas about Boltzmann machines inspired by statistical physics and ideas of learning in stochastic networks [7] and was subsequently vigorously developed by a wider community from computer science and the (applied) computational sciences ([5,11]). An architecturally important enabler for many dNN is the use of an autoencoder [5] which enables self defined convolutions to act as a massive pattern transformer. However, this is more relevant in a data context with geometrical information; hence we do not use this feature for models of our client data, since they do not contain images or any kind of causally or geometrically chained information (although they have been used occasionally for bankruptcy prediction [10]). More relevant to us are (the revival of) error backpropagation and adapted activation functions, like e.g. ReLUs which do not inhibit backprop errors over arbitrarily many network layers [3].

Around the year 2015 dNNs became increasingly accepted as being surprisingly effective and also efficient, especially on supervised classification problems. Many such real-world problems do have, respectively, do need (a) vast (Terrabyte sized) amounts of data, (b) a large number of hidden layers (so called "deep" neural models) and (c) efficient means for (auto-)regularization as a consequence of the huge numbers of free parameters of the initial (untrained) models. Considerable gains in both efficiency but also classification accuracy were seen in some application domains like visual pattern recognition. However, these gains seem to be obtained primarily by much increased hardware performance, using various types of parallelism and co-processing. Both, graphical and tensor processing units (GPUs and TPUs) also proved to be very useful. Finally, there is also new back- and front-end programming [17] for these model classes.

3.2 Our Deep Neural Networks, Activations and Dropout

By now there exists a broad spectrum of high-performance, open domain Machine Learning software instruments, many of them focusing on deep Neural Networks. Examples are TensorFlow, Theano, Torch, Julia-based Flux, to name a few more widely employed. For our computational experiments we use **R** as a experiment chain driver and **Keras** [9] from inside **R** as model chain driver for deep NN which then in turn calls **TensorFlow** for effective model training (other variants like Python – Tensorflow or Julia – Flux are equally possible).

Our procedure of re-modeling credit client our data models in order to test for possible performance gains over the more traditional ML-techniques employed in past models is the following: We start with our best models in terms of ROC-AUC from past work for two credit scoring data sets of different structure and size. We generate dNN models evaluated over training-test data pairs. The models are of vastly different size and of different potential functional complexity. We then evaluate the out-of-sample results in comparison to the benchmark models and inquire into the role of imposed and automatic regularization as well as modalities of early training stop.

4 Re-Modeling Our Small Data Set with dNN

In order to find out if and also in which way dNN may surpass the more traditional ML-based classification models we first turn to our smaller data set. After some preliminary computational experiments with all our credit client data sets, we note that there is no substantial advantage of dNN – at least in terms of the most direct and popular measures of out-of-sample performance, like hit-rate and accuracy – over the more traditional methods described above and used in previous work. As is detailed in Sect. 3.1 with regard to other more structured and much bigger data sets this may now not come as a surprise. One of the most practically useful additional performance criteria in the context of credit client scoring is the Receiver Operating Characteristic (ROC, which is also used in many other application domain) and its aggregate scalar expression, the area under curve (AUC).

4.1 What to Match and Possibly to Surpass: Data Set 1

Our question is then whether for a credit client data set of size $N \approx 10^2 - 10^3$, (d)NN can beat the more traditional Machine Learning models in terms of AUC. More specifically, we also ask in which way this is achieved. To this end, it seems natural to compare the AUC per model obtained on different data subsets. Hence, our general scheme of running experiments is guided by the following:

– We concentrate on out-of-sample **ROC/AUC** as a widely accepted performance criterion.
– The best out-of-sample AUC performance from our past extensive model comparison in [16] averaged over the ten randomly selected (but henceforth retained) train/test sets is *Logistic Regression* (LogReg) model with

Model	Mean	Std. Dev.	Min	Max
LogReg*	0.782	0.021	0.756	0.826

– A very similar AUC performance is obtained by the *Linear kernel SVM*; in a way, due to **relative sparseness** good results for linear-like models are to be expected as there are not enough data to reliably detect an essentially non-linear class-separation function. In this case there are $m = 20$ recorded input features, then "expanded" into a 61-dim representation by using categorical variables (binary) encoding where appropriate.

4.2 How Do Relate Differently Sized dNN to the Best LogReg?

The structure of deep NN can be described along many architectural as well as numerical mathematics features (Sect. 3.1) which are more or less relevant to modeling the given problem data. For our data which has no feature-based geometric structure (as is rather the case in data about images, time-series, etc.) we refrain in our experiments from using an autoencoder block (which tends to remap the geometry to some useful end) and we directly use stacked layers of conventional NN. The items we put to use from the Keras-Tensorflow library are mainly (a) efficient modeling chains offered by Keras-Tensorflow, (b) within some deeper variants of NN the use of ReLU activation functions in order to strongly limit the otherwise fast decay of the backpropagation signals and to reduce training time (c) layer-wise dropout in order to automatically prune the model (i.e. to potentially avoid over-training) and (c) the training with enhanced solution search of very large models (using the Adams optimizer) still amenable to present personal-grade computer power.

In the computational experiments reported below we use the following convention for a short-hand identification of the dNN model structures:

Inputs:	Raw data input dimension plus categorical
Layers:	number of activation nodes (neurons) layers, etc.
Activation functions:	Sigmoid or ReLU
Outputs:	one, as the goal is forecasting a binary class-label
Validation:	share of training data reserved for validation
Dropout:	share of neurons to de-active or to eliminate

Hence our shorthand model description used in the upcoming tables is for instance $80^s_{.2} - 100^r_{.3} - 30^r_{.1} - 1^s(v = 0.25)$ meaning that starting from a constant number of inputs (not included in the notation) we proceed via first layer using sigmoidal activation functions and a dropout rate of 0.2 to the second layer using ReLU activation functions and a dropout rate of 0.3 and so on until reaching the output node of the four layer network. Also 25% of the training examples are reserved for in-sample cross-validation. The split of all data into a training and a test set is assumed to proceed all modeling. We always use a feedforward, fully connected layer to layer structure as a starting architecture. Models with no dropout or partial dropout and without in-sample validation ($v = 0$ or no annotation) have also been used.

Table 1. Overall out-of-sample AUC for models trained on the small data set

MODEL	**Mean**	**Std. Dev.**	**Min**	**Max**
LogReg*	0.782	0.021	0.756	0.826
$25^s_{.3} - 1^s$	0.790	0.019	0.766	0.820
Regret AUC	−1.13%	9.18%	−0.89%	0.00%
$100^s_{.3} - 50^s_{.2} - 1^s$	0.785	0.022	0.757	0.826
Regret AUC	−2.96%	21.39%	−3.11%	−0.03%
$100^s_{.3} - 80^s_{.3} - 50^s_{.2} - 1^s$	0.788	0.023	0.757	0.824
Regret AUC	−3.32%	23.42%	−3.00%	−0.36%

The results concerning the NN models on the small data set are displayed in Table 1. They are trained without in-sample validation. Layer-wise **dropout** with rates 0.2–0.3 is used. Hence, no training data is "wasted" for validation sets. Training has been restricted to 80 epochs, which on the basis of running many other test problems form different application domains is judged to be quite extensive for this relatively modest training data volume. The LogReg reference model is taken from previous work [16].

For every new NN model we include a line of measurements which is the exact counterpart of the LogReg-entry. The AUC means, standard-deviations as well as their min and max values are taken over the ten training-test-set pairs, which are exactly the same as those used previously for LogReg. The line added to every new NN model denotes the regret which is produced by stopping the model at epoch 80 instead of (hypothetically) having been able to stop it at the (generally different) epoch which would produce the maximum AUC on the out-of-sample set. Hence negative values denote a standardized measure of loss while positive values indicate a gain. These measurements are for comparative purposes in order to rank the models and should not be interpreted for empirical credit client matters.

This table (Table 1) informs us that the simplest nonlinear neural model, namely $25^{s}_{.3} - 1^{s}$ with $v = 0$ (see the above conventions) is the best regarding mean AUC and with low variation over the training-test data pairs, and it also slightly surpasses the LogReg reference model. The deeper NN models lead to more variation in AUC which means they may adapt better than average to some training-test data set pairs and vice versa. We tested many more model architectures than those reported here. However, the result do not change the overall situation described.

4.3 Pair-Wise Training-Test AUC Curves (80 Training Epochs)

We now report on the ten different deep NN models for every evaluated model architecture, by depicting the AUC-curve on the training set and on the test, respectively.

In Fig. 1 we start with the smallest model from Table 1. i.e. the NN with one hidden layer. In each plot inset we show two curves. The thin curve is the model AUC measured over the training epochs for the training data set. The fat dotted curve is the corresponding evolution of the model AUC over the test set. Obviously, the latter one is the relevant measure of the out-of-sample model performance (the performance practitioners are most interested in). While the AUC grows smoothly over the training epochs as one would expect, the out-of-sample curve is leveling off for most of training-test data pairs. Using the change of slope of the training curve (an information effectively available during training of real life classification models) as a hint for early training stop is however not reliable. Overall, this model is quite robust in that using a wide range of sufficiently large (for more than 15 epochs, say) but otherwise randomly chosen training stops, out-of-sample performance does not fluctuate much.

In contrast to Fig. 1, Fig. 2 depicts the AUC evolution for the deepest or architecturally most complex model from Table 1. The meaning of the curves is the same of that from the previous figure.

This AUC grows smoothly as well but it clearly shows one or more dents over the training epochs. The out-of-sample curve is clearly leveling off for all but one training-test data pair. Using the location of the first dent in the slope of the training curve can be a more reliable hint for early training stop. Overall, this model is slightly over-trained at the last learning epoch (80). In that using a wider range of sufficiently large but otherwise randomly chosen training stops, out-of-sample performance does not fluctuate more than that of the simple NN.

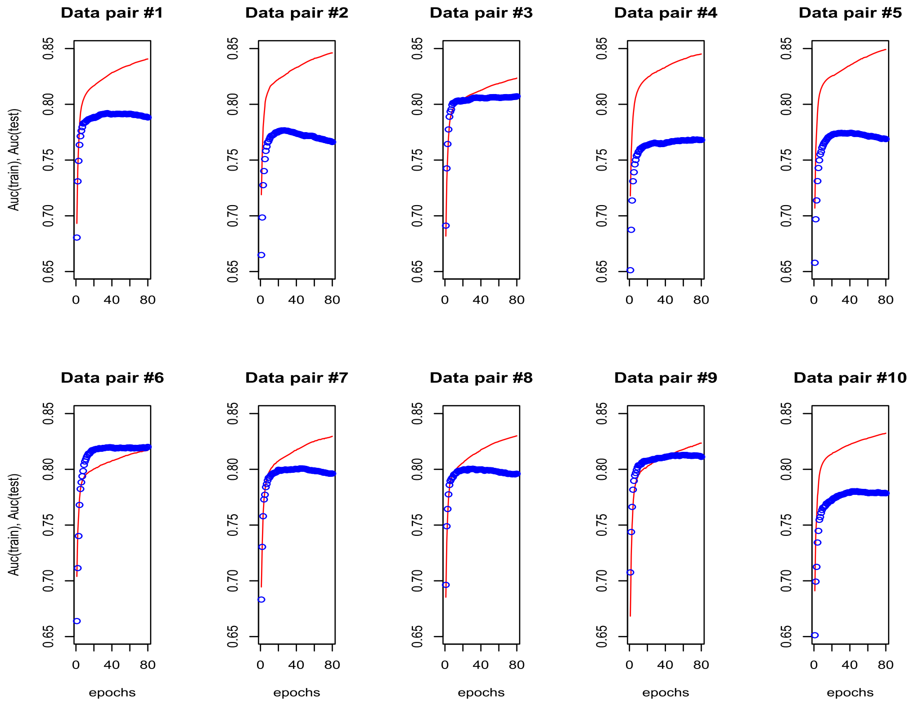

Fig. 1. AUC levels over training and out-of-sample test sets for the $25^s_{.3} - 1^s$ model

5 Re-modeling Our Big Data Set with dNN

Our large credit client data set is taken to represent classification problems in the range $N > 10^5$. This may not be much for technical problems but it is nevertheless a quite substantial data set size for a single financial company. Recall that data pooling is a rather weird but also highly relevant problem in finance owing foremost to various data privacy constraints [16]. Our big data set comprises 12 observed inputs as raw data. Some of the variables are transformed into categorical binary representation leading the 25 effective input variables (client features per case). Furthermore, the data come with highly asymmetric case label frequencies, with non-defaulting credit client vastly outnumbering the defaulting ones. From past extensive experiments the best model turns out to be a Support Vector Machine [14,19] using Radial Basis Function kernels (RBF-SVM). Such types of SVM are in general very powerful at building class separation functions over input space (here client feature space) by using the theoretically appealing and practically very useful concept of support vectors which can be used in many ways to get further insight into the classification model at hand [13].

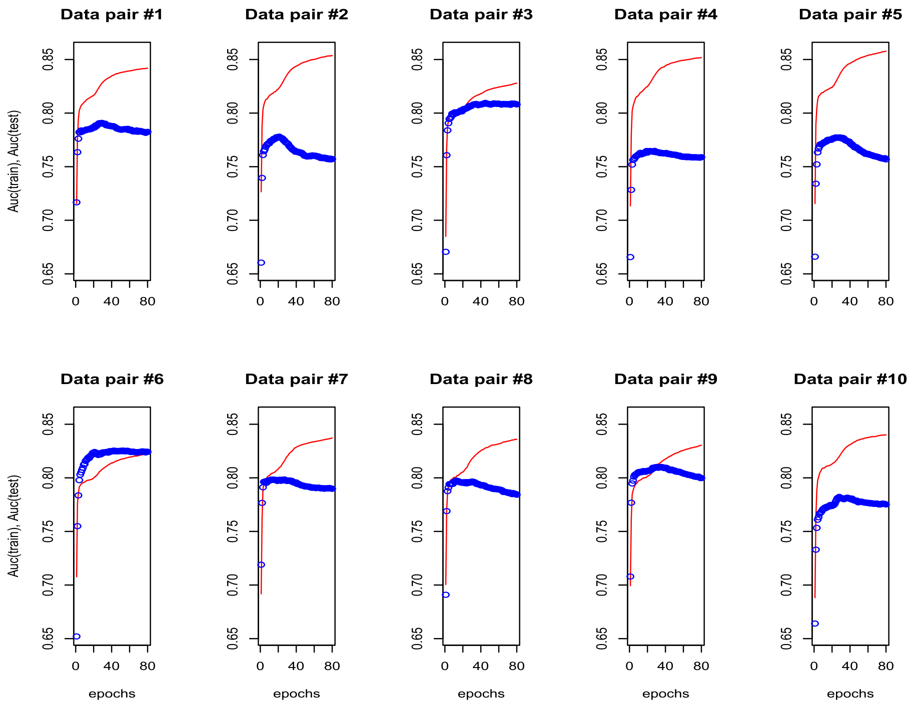

Fig. 2. AUC levels over training and out-of-sample test sets for the $100^s_{.3}-80^s_{.3}-50^s_{.2}-1^s$ model

5.1 What Model to Match and Possibly to Surpass?

Here our question is if deep NN can clearly beat highly principled ML-models such as RBF-SVM. As in the previous sections the question is addressed specifically with regard to our credit client data but now in terms of AUC for $N > 10^5$. Owing to the much bigger data volume all NN models are trained now over 160 epochs. All the other conventions for describing the computational experiments are identical to those from Sect. 4.

In Table 2 we report the results regarding the successively deeper NN models in comparison to the RBF-SVM reference model. After increasing the simple one layer NN architecture to 100 neurons in the hidden layer, and neither using dropout nor in-sample validation, we arrive at a quite remarkable result (at least for these type of data), namely a jump in mean AUC compared to the SVM-model. By using rule-based training stop all outcomes can be further increased (at least marginally). The next model with two hidden layers improve this result further. Note that here too, there is no imposed regularization (no dropout, no in-sample validation). The last two models then behave as one would expect, given the first two: a drop in all measured criteria, especially in the variant without rule-based training stop.

Table 2. Overall out-of-sample AUC for the big data set

Model	Mean	Std. Dev.	Min	Max
RBF-SVM	0.700	0.005	0.692	0.707
$100^s_0 : 1^s$ ($v=0$)	0.713	0.007	0.698	0.723
TrainStop: W	0.718	0.005	0.711	0.727
$100^s_0 : 50^s_0 : 1^s$ ($v=0$)	0.707	0.008	0.695	0.723
TrainStop: F	**0.722**	0.006	**0.713**	**0.732**
$100^s_{.1} : 50^s_{.1} : 1^s$ ($v=0$)	0.566	0.008	0.549	0.576
TrainStop: G	0.716	0.006	0.705	0.725
$100^s_{.1} : 50^s_{.1} : 1^s$ ($v = 0.2$)	0.636	0.048	0.572	0.698
TrainStop: G	0.711	0.005	0.704	0.718

5.2 Pair-Wise Training-Test AUC Curves (160 Training Epochs)

In Fig. 3 we depict the AUC evolution over the ten training-test data pairs for the first NN model from Table 2. Owing to the size of the data set, we train the models over 160 epochs. The expectation is that such a comparatively large number of training epoch will be sufficient for any of the training-test data pairs. Here too, the meaning of the curves is the same with that from the previous figures.

In this case the AUC over the training set grows less smoothly and it is now showing globally a uni-modal (locally a multi-modal) evolution. The out-of-sample curve is clearly leveling off at later epochs and its evolution is locally irregular, although if follows globally approximately the AUC curve over the training set. In this case a useful candidate for a early stopping rule would be the onset of more pronounced irregularity in the training set curve. Overall, this model is over-trained at the last learning epoch (160). Randomly chosen training stops over epochs leads to quite strong fluctuations of out-of-sample performance. Note that in this case there is no imposed regularization of any kind.

In the case of the model with two hidden layers depicted in Fig. 4 we use both types of regularization, dropout with a rate of 10% (of neurons to delete) and a in-sample validation set with 20% of retained cases (out of all available training cases or observations). Here the evolution of the AUC over training epochs is again much smoother, with the shapes of both curves matching to a high degree. While the peak AUC performance is out-of-sample quite good, there may be a

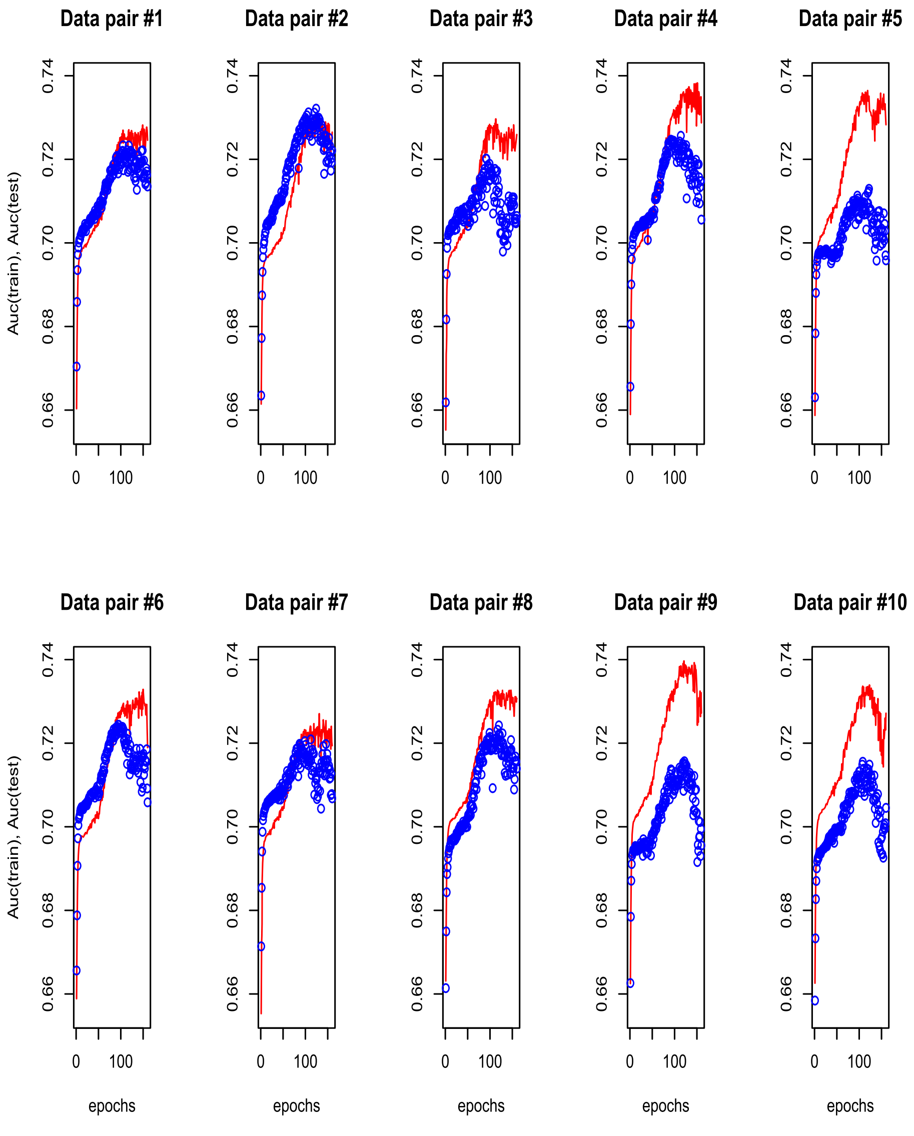

Fig. 3. AUC levels over training and out-of-sample test sets: the $100^s : 1^s$ model (d = 0, v = 0)

sharp drop of performance toward the end of the training epochs. Here a random training stop in later epochs is risk-laden while it is completely feasible over an interval starting at, say, 1/2 and reaching 2/3–3/4 of the maximum epochs.

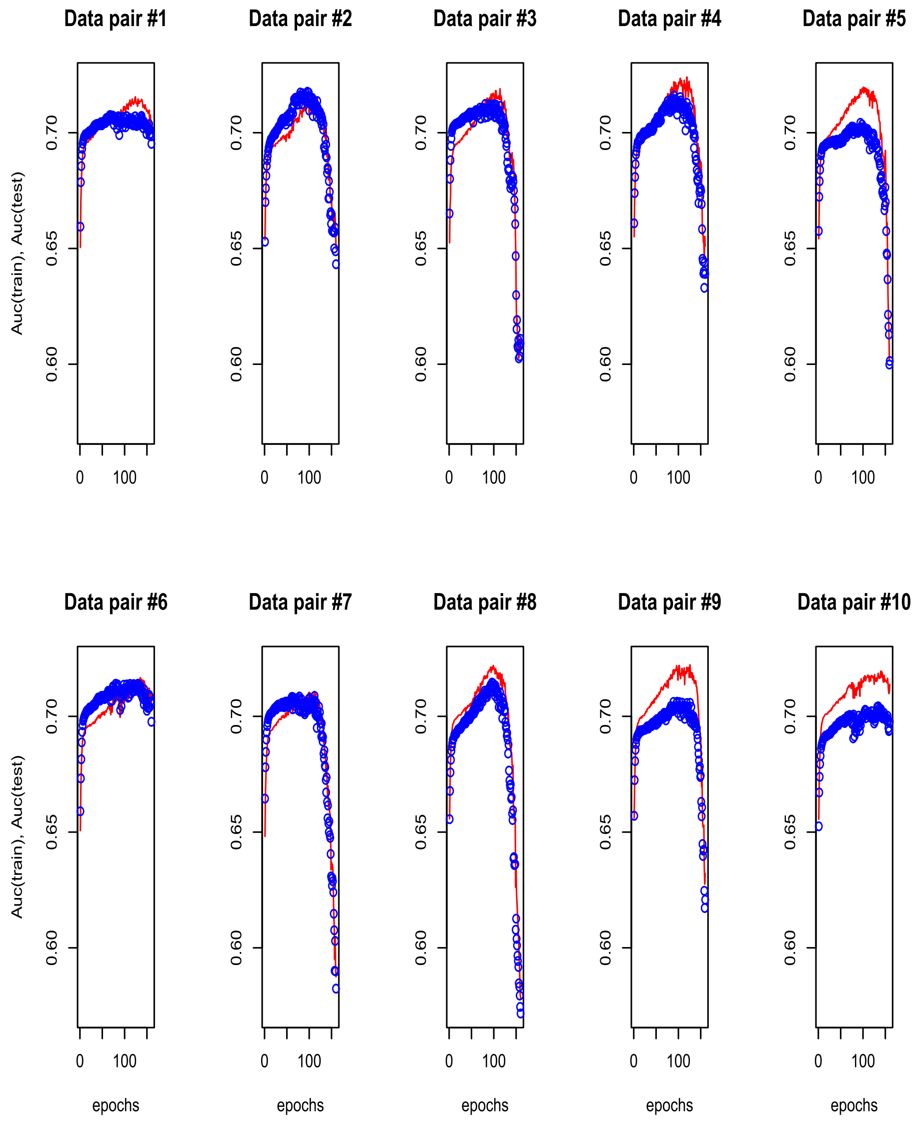

Fig. 4. AUC levels over training and out-of-sample test sets: the $100^s : 50^s : 1^s$ model (d = 0.1, v = 0.2)

5.3 What About Deeper NN Models? Again, for Data Set 2

Should one detect progress in out-of-sample performance by using successively deeper models (i.e. NN models with more layers) this hints at the possible presence of some non-linear dependencies, hitherto unseen by simpler models. If one does not find such progress, this does not preclude the existence of such dependencies in principle, but possibly, that they can't be detected owing to the relative sparseness of data (i.e. relatively few data points N compared to input feature dimension m). Hence, In general, a big number of training/validation examples is in itself not enough to produce more intricate, reliably detectable input-output relations. Data may also be very redundant and/or the number of different class-members may be extremely asymmetrical. The latter is certainly the case in our credit client data. Hence, questions pertaining to this problem area can't be conclusively answered except by explicitly evaluating the resulting models on the application-specific data instances. In our case this means adding more layers to the models (making the NN deeper). Hence, in what follows, we report on using NN with – at least potentially – much more complicated behavior.

Table 3. Overall out-of-sample AUC for increasingly deep NN models

Model	Mean	Std. Dev.	Min	Max
RBF-SVM*	0.700	0.005	0.692	0.707
$400^s_{.2} : 400^s_{.1} : 1^s$ $(v = 0.2)$	0.710	0.006	0.701	0.719
Regret AUC	−0.37%	11.89%	−0.68%	0.00%
$100^s_0 : 30^r_0 : 30^r_0 : 30^r_0 : 1^s$	0.701	0.012	0.678	0.720
TrainStop: G+	0.712	0.005	0.704	0.720
$100^s_0 : 30^s_0 : 30^s_0 : 30^s_0 : 1^s$	0.640	0.050	0.548	0.704
TrainStop: G	0.710	0.006	0.702	0.719
$100^s_0 : 50^s_0 : 30^r_0 : 30^r_0 : 30^r_0 : 1^s$	0.680	0.046	0.555	0.710
TrainStop: G+	**0.716**	0.007	**0.705**	**0.728**

The meaning of the entries of Table 3 is the same as for the previous tables. The regularized model with two large hidden layers beats the reference RBF-SVM slightly in terms of mean, min and max AUC. Regularization does work by smoothing the otherwise irregular training curve (thin continuous line). There are models for 5 out of 10 training-test data pair where out-of-sample AUC is above training AUC. To some degree this was also observed in runs shown in previous figures over the first epochs. As this recurs, we interpret it as a peculiarity of the respective training data sets. After increasing the number of hidden layers similar or better AUC performance can only be obtained by (rule-based) training stop. Note that the use of ReLU activation functions leads to improvements over the models which use sigmoidal activations only: the second and the fourth model are better then the third one. Overall, compared to RBF-SVM, all our deep NN models lead to a higher variance of AUC over our ten training-test data sets. The only exception is the second NN, provided one succeeds in stopping the training appropriately.

5.4 Pair-Wise Training-Test AUC Curves of Deeper NN

For the first NN with two layers containing 400 neurons respectively only the first 40 epochs are shown in Fig. 5. For this model we use both dropout and in-sample validation. For the "deeper" models (Figs. 6 and 7) all 160 epochs are shown. Here we do not use any dropout or in-sample validation, leaving all the work to the parameter adaptions by the error-backprop algorithm.

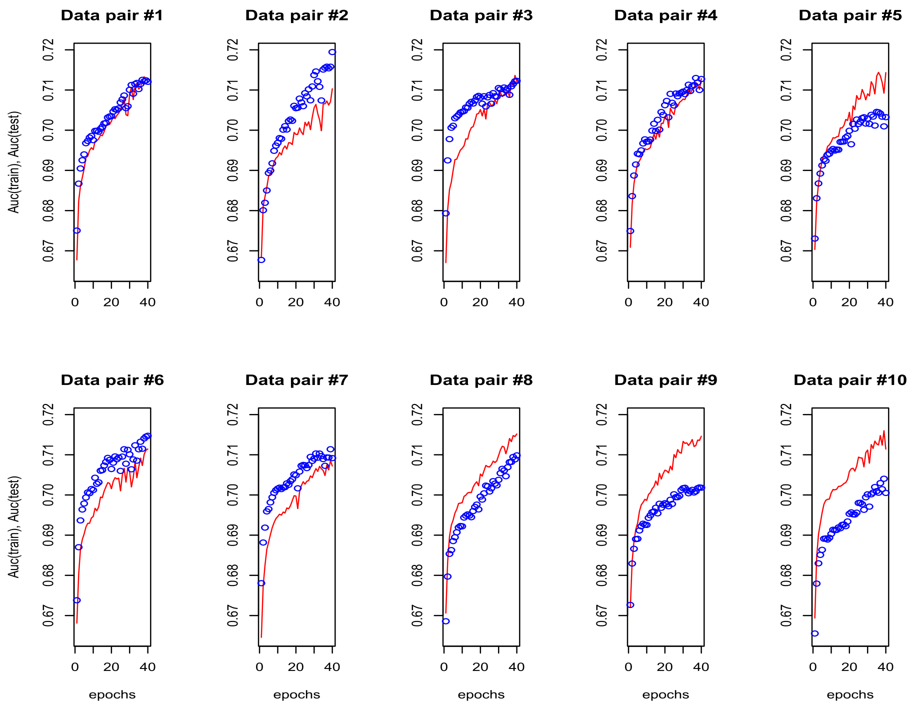

Fig. 5. AUC levels in training and out-of-sample test: the $400^s_{.2} : 400^s_{.1} : 1^s$ model (val = 0.2)

In Fig. 5 we display the first 40 training epochs of the 400 neurons per layer, two-layer NN. In this epoch range, an early-stopping moving-average rule over epoch-wise AUC evolution would work reliably.

In Fig. 6 we display all 160 training epochs of the four hidden layer network (the third NN model from Table 3) which uses no imposed regularization but sigmoidal activations in the higher layers. Provided one could find an adequate early-stopping rules for training (moving-average seems to work here as well) this model would beat the RBF-SVM reference model. A random training stop, however, would produce much inferior results. Note that the model is degrading over the later training epochs, but is also able to recover. It is unclear if, e.g. by using very long training, one could find still better models.

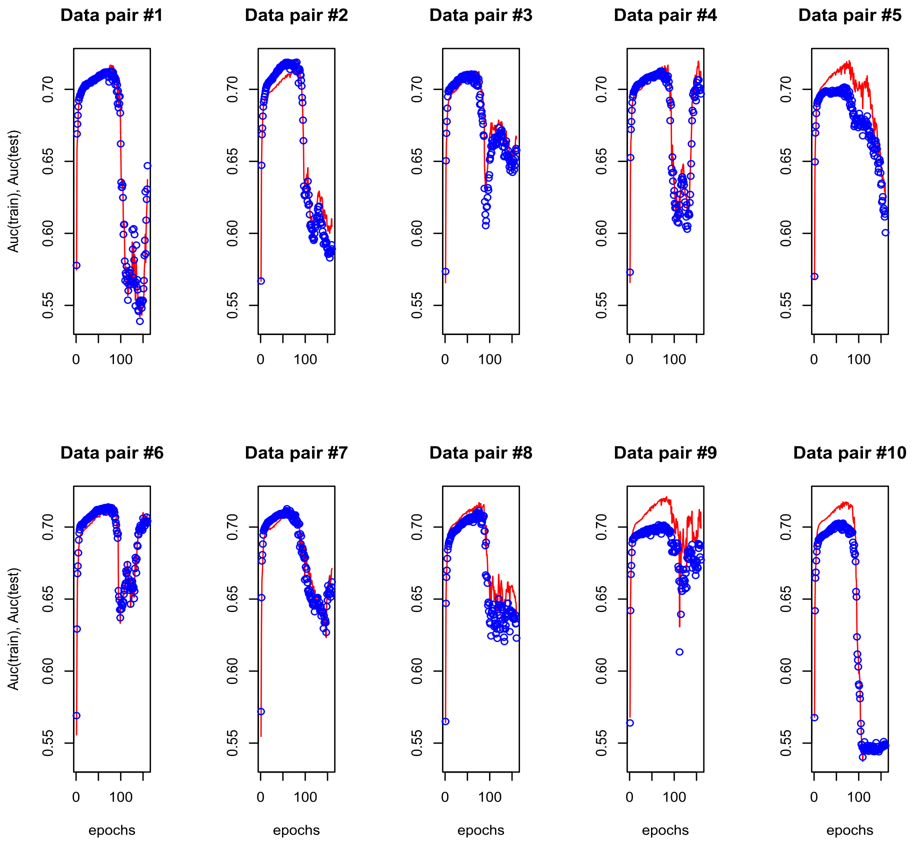

Fig. 6. AUC levels in training and out-of-sample test: the $100^s : 30^s : 30^s : 30^s : 1^s$ model

Finally, in Fig. 7 we also display all 160 training epochs of the five hidden layer network (the fourth NN model from Table 3) which also does not use imposed regularization but ReLU activations in all higher layers. Compared to the last described model (in Figure 6) this model does degrade much less over later training epochs. Hence, using adequate early-stopping rules for training (moving-average seems to work here quite well) are less risky to apply. This model would beat the RBF-SVM reference model (together with the second NN model from Table 3 which also uses ReLU activations) by the largest margins.

A recommendation for further progress in finding the appropriate (as well as robustly good performing) models is to further study possible rule-based stopping criteria for deep NN training. This may proceed for instance by starting with simple moving-average based rules and may employ more elaborate shape analysis of learning curves on training sets as true out-of-sample sets are not available in real world applications.

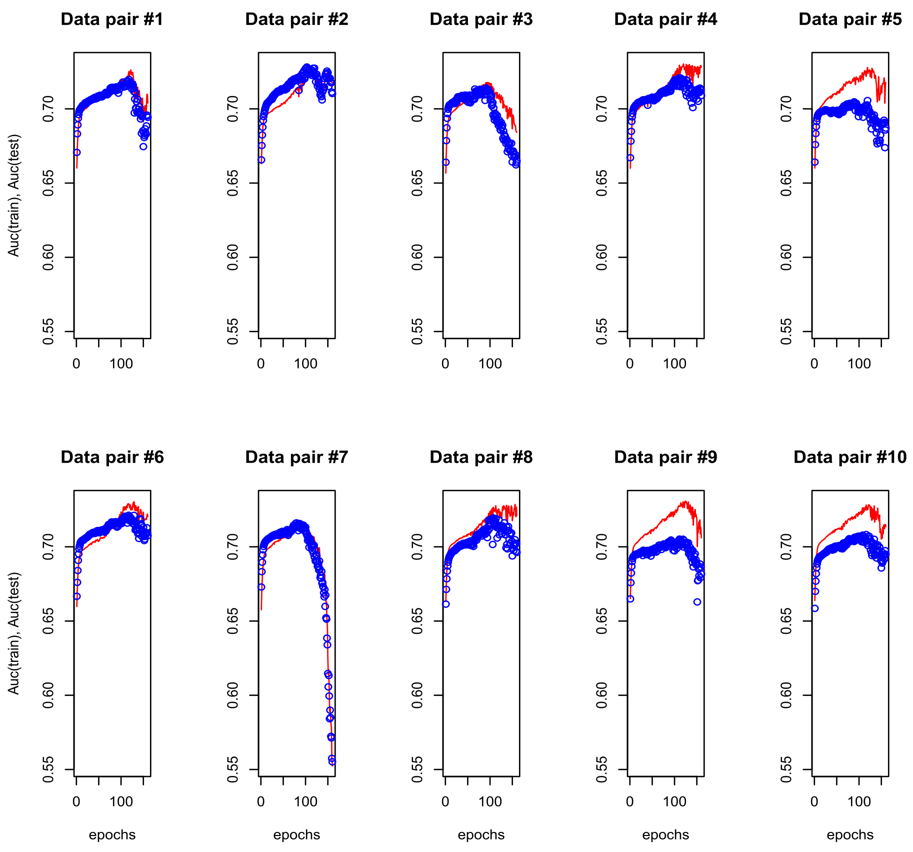

Fig. 7. AUC levels in training and out-of-sample test: the $100^s : 50^s : 30^r : 30^r : 30^r : 1^s$ model

6 Some Attempts at Explaining Our Results

The gain of using deep neural networks for our credit client data is significant – but not dramatically so, as one may expect. Especially for the "deeper" models, ROC-AUC is more volatile and is conditioned on the ability of the modeler to be able to make a good guess for the training epoch for which a model will produce a higher (or perhaps maximal) ROC-AUC value over a test set.

6.1 About Computational Cost of Our Models

For the models on our small data set, Logistic Regression is competitive. Compared to any other more complex modeling technique (SVM, dNN, ...) the computational costs are negligible. For the models on our big data set, we have to compare RBF-SVM against dNN. Due to the use of the RBF-kernel and due to the need to perform grid search for both hyperparameters "capacity" and "kernel-width" the cost of the SVM is not significantly lower compared to that of the highly refined and optimized dNN model-building via Tensorflow (time consumption on an Intel i5 4-core computer is around 2–3 h for the whole computational cycle, including 10-fold train-test splits per model).

6.2 Why Are dNN Superior with Regard to ROC-AUC?

ROC curves are calculated by confronting the vector of the binary, e.g. (0, 1)-reference output for every data example (case) with the vector of continuous probabilities of choice (the effective model output) for the cases by varying a real valued "cutoff". As all the measures based on discrete information (hit-rates, etc.) do not differ very much between our models, only the particular fine-structures of the output-probabilities acts differently on the results as the cut-off is gradually shifted. In past modeling [15] we observed – especially in capacity constrained SVM-models, which intentionally allow for a certain amount of misclassification in order to avoid overtraining – the formation of substantially different vectors of output probabilities for the same data when using just slightly different nonlinear kernels (e.g. of Coloumb-type instead of RBF, see [8,15]. A similar effect may hold in the present model population, as dNN are capable of more diverse input-output mappings.

6.3 Concerning the Prediction of the Best Moment for Training Stop

To a certain extend we can predict the best stopping moment (in terms of number of epochs) by fitting piecewise (local) linear regressions to re-scaled learning curves resulting from minimizing the binary cross-entropy which is used as the effective objective. This is useful if the models with the maximum out-of-sample AUC value tend to be near inflection points between local (linear) approximations. This is true in a majority of cases but there are notable exceptions. Combining such models may yield still superior out-of-sample AUC values. Other variants are to use the AUC learning curve (from training) to predict best stopping moments for out-of-sample AUC levels. Furthermore, in order to improve overall AUC levels, one may seek to optimize directly with regard to AUC instead of reading it off as a co-metric (personal communications by Frank-Michael Schleif). This, however, may require non-differentiable methods for model parameter search.

7 Conclusion and Outlook

Credit scoring data come in different input shapes and volumes. Their associated classification problems face both relative data sparseness and substantial noise. However, owing to various reasons, for instance data privacy, data availability of such applications will and cannot be increased at will. Hence, besides the our long-lasting aim of increasing forecasting performance (e.g. [13]), more recently incorporating privacy constraints [16], and to better understand the data themselves [12], here we are concerned with what aspects of modeling can be further improved. Our extensive tests using deep neural networks (dNN) for two very different credit client data sets do find that training and validating classification models leads to limited improvement over the more traditional out-of-sample performance – as is measured by hit-rate and classifications accuracy. However using these neural models with AUC as a criterion for training stop, proves to be significant in that it is often reliable and the resulting (superior) out-of-sample AUC is in itself a robust measure of overall model performance and stability. As truly dramatic improvements over base models are very unlikely for our data, we nevertheless conclude that (1) dNN may produce significantly better out-of-sample forecasts than standard ML or statistical learning models; and even the grossly over-sized dNN model variants still produce highly competitive results with limited training effort. (2) Dropout as a regularization method of dNN may produce more stability w.r.t. advantageous training stops; it hence may replace the widely practiced in-sample validation. In general, it may be stated that such and related modeling successes may be mainly attributed to the ability of today's *computationally advanced* dNN to successfully auto-regularize. (3) For our bigger data set, which is more difficult to classify and in some sense degenerate, dNN clearly show their potential. Again such dNN are highly robust against over-specification. These findings concern our implicit goal of maximizing out-of-sample ROC/AUC. However, as far as using dropout in place of in-sample validation is concerned, the message carries over to other modeling problems and for other performance measures (e.g. recurrent dNN and MSE-error, as other own work finds). Finally, (4) the chances of obtaining valuable models by using dNN increases if the financial industry manages someday (against all odds) to merge their client data bases. The source for training data would become truly huge, and perhaps more structured, including contexts, news, client attitudes and sentiment, all conveniently time-stamped. This may then lead to additional and interesting modeling targets.

References

1. Abdou, H., Pointon, J.: Credit scoring, statistical techniques and evaluation criteria: a review of the literature. Intell. Syst. Account. Finance Manag. **18**, 59–88 (2011)
2. Addo, P.M., Guègan, D., Hassani, B.: Credit risk analysis using machine and deep learning models. Risks **6**(38), 1–13 (2018). https://doi.org/10.3390/risks6020038
3. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. arXiv:1206.5533 [cs.LG], pp. 1–20 (2012). https://arxiv.org/pdf/1206.5533.pdf
4. Experian, The credit reference agency explained, a guide for consumer advisers, experian (2013). http://www.experian.co.uk/downloads/consumer/creditRefAgencyExplained.pdf
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, Cambridge (2016)

6. Ha, V.-S., Nguyen, H.-N.: Credit scoring with a feature selection approach based deep learning. In: MATEC Web of Conferences 54, 7th International Conference on Mechanical, Industrial, and Manufacturing Technologies, pp. 1–5 (2016). https://doi.org/10.1051/matecconf/20165405004
7. Hinton, G.: A practical guide to training restricted Boltzmann machines. Working Paper UTML TR 2010–003, pp. 1–20, University of Toronto (2010). http://www.cs.toronto.edu/~hinton/absps/guideTR.pdf
8. Hochreiter, S., Mozer, M.C., Obermayer, K.: Coulomb classifiers: generalizing support vector machines via an analogy to electrostatic systems. In: Advances in Neural Information Processing Systems 15, pp. 561–568, MIT Press (2003)
9. Keras, called from R. https://cran.r-project.org/web/packages/keras/index.html. Accessed 19 Sept 2020
10. Kvamme, H., Sellereite, N., Aas, K., Sjursen, S.: Predicting mortgage default using convolutional neural networks. Expert Syst. Appl. pp. 1–43 (2018). https://doi.org/10.1016/j.eswa.2018.02.029. Accepted for publication
11. Rackauckas, C., et al.: Universal differential equations for scientific machine learning. arXiv:2001.04385v3, pp. 1–45 (2020)
12. Schebesch, K.B., Stecking, R.W.: Topological data analysis for extracting hidden features of client data. In: Doerner, K.F., Ljubic, I., Pflug, G., Tragler, G. (eds.) Operations Research Proceedings 2015. ORP, pp. 483–489. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-42902-1_65
13. Schebesch, K.B., Stecking, R.W.: Support vector machines for credit applicants: detecting typical and critical regions. J. Oper. Res. Soc. **56**(9), 1082–1088 (2005)
14. Schölkopf, B., Smola, A.: Learning with Kernels. The MIT Press, Cambridge (2002)
15. Stecking, R., Schebesch, K.B.: Comparing and selecting SVM-kernels for credit scoring. In: Spiliopoulou, M., Kruse, R., Borgelt, C., Nürnberger, A., Gaul, W. (eds.) From Data and Information Analysis to Knowledge Engineering. Studies in Classification, Data Analysis, and Knowledge Organization. Springer, Berlin (2006). https://doi.org/10.1007/3-540-31314-1_66
16. Stecking, R.W., Schebesch, K.B.: Classification of credit scoring data with privacy constraints. Intell. Data Anal. **19**(s1), S3–S18 (2015)
17. Song, M., Hu, Y., Chen, H., Li, T.: Towards pervasive and user satisfactory CNN across GPU microarchitectures. In: International Symposium on High Performance Computer Architecture (HPCA), pp. 1–12 (2017)
18. Tomczak, J.M., Zieba, M.: Classification restricted Boltzmann machine for comprehensible credit scoring model. Expert Syst. Appl. **42**(2), 1789–1796 (2015). https://doi.org/10.1016/j.eswa.2014.10.016
19. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
20. Yu, L., Yang, Z., Tang, L.: A novel multistage deep belief network based extreme learning machine ensemble learning paradigm for credit risk assessment. Flex. Serv. Manuf. J. **28**(4), 576–592 (2015). https://doi.org/10.1007/s10696-015-9226-2