Investigation of CI forecasting algorithms for short-time cash demand in ATM network

G. Žylius, V. Vaitkus, and R. Simutis
Department of Automation, Faculty of Electrical and Electronics Engineering, Kaunas University of Technology, Kaunas, Lithuania

Abstract - *Good ATM network cash management requires accurate information of future cash demand. In this paper we compare computational intelligence models when performing cash flow forecasting for one day. Adaptive input selection and model parameter identification are used with every forecasting model in order to perform more flexible comparison. Experimental data contains 200 ATMs from real ATM network with historical period of 26 months. Investigation of historical data length influence for forecasting accuracy with every model is also performed. Results suggest that v-SVR (support vector regression) forecasting model performs best when SMAPE forecasting accuracy measure is used.*

Keywords: Computational Intelligence, Cash Flow, One-Step-Ahead Forecasting.

1 Introduction

In order to optimize cash distribution in ATM network an estimation of cash demand in the future must be done. So cash demand forecasting accuracy determines overall performance of cash management system. Good cash management for ATM network brings savings for retail banks that are related to: 1) dormant cash reduction; 2) reduced replenishment costs; 3) decrease of cash preparation costs; 4) reduction of cash insurance costs.

Various uncertain factors influence cash demand in ATM that makes this process hard to forecast: nonlinear trend and seasonal component mixtures with non-stationary heteroscedastic uncertainty. However approximate empirical relationship between input and output variables can be obtained using complex data-based regression or time series models.

In this research we apply and compare data-based computational intelligence (CI) regression models for one day ahead cash demand forecasting. The dataset used consist of 200 ATMs.

This paper is further divided into following sections: 1) literature review (a review of existing methods applied for cash demand forecasting); 2) forecasting models (specification of each forecasting model used); 3) experimental data and methodology (short explanation of methodology used for forecasting and experimental data specifications); 4) results (analysis of forecasting results); 5) conclusions and future works.

2 Literature review

Process of cash demand in ATM is highly affected by holidays, seasonal and calendar effects [1]. These effects can be incorporated into classical neural network models that could be used for cash demand forecasting [2]. However cash demand varies in time, so more flexible approach [3] to incorporate neural networks for cash demand in ATM is needed. Advantage of popular support vector machines over neural networks applied for regression was experimentally denied [4] when cash demand forecasting with long historical period training data was performed. However, for data with shorter history support vector regression may be more effective. An interval type-2 fuzzy neural network (IT2FNN) applied [5] for cash demand forecasting is another approach that adapts to dynamic nature of ATM cash flow and (as author claims) is better than other systems based on time series.

Despite CI model applications, time series models are also used for cash demand forecasting problem. Researches show that SARIMA models among classical econometric models perform best [6] and even outperform joint forecasting approach using vector time series models [7]. Authors in [8] made a comparison between time series probability density forecast models: linear models, autoregressive models, structural time series models and Markov-switching models. Results showed that Markov-switching models performed best.

Other researchers use cash demand forecasting approaches that deeper investigate cash demand process (beyond aggregated data-based empirical relationships). Random-effects models [9] [10] were used to model individual cash withdrawal patterns for ATM withdrawal forecasting. Researcher in [11] treats intraday cash flow time series as random continuous functions projected onto low dimensional subspace and use functional autoregressive model as predictor of cash flow and intensity of transactions. ATM clustering approach was employed by [12] when integrated forecasting of aggregation of nearby-location ATM cash demand was performed.

3 Forecasting models

3.1 Support vector regression

Support vector machine (SVM) originally proposed in [13] is linear model that is used both for classification and

regression. SVM application for regression is called support vector regression (SVR). Main idea of SVM is to map input data vectors into high dimensional feature space by using kernel functions. By doing so, linear nature of SVM model can be applied to nonlinear function approximation. This type of mapping is called *kernel trick*. In this research we use two types of SVR: 1) *v*-support vector regression (*v*-SVR) and 2) least squares support vector regression (LSSVR).

3.1.1 v-SVR

Given the set of data points such that $\boldsymbol{x}_i \in R^n$ is an input vector (*i*-th observation *n*-dimensional vector) $y_i \in R^1$ is a target output, the optimization problem for *v*-SVR algorithm is formulated by following equations [14]:

$$\min_{\boldsymbol{w},\varepsilon,\xi,\xi^*,b}\left\{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}+C\left[\nu\varepsilon+\frac{1}{l}\sum_{i=1}^{l}\left(\xi_i+\xi_i^*\right)\right]\right\}$$
$$\text{subject to}\begin{cases}\left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)+b\right]-y_i\le\varepsilon+\xi_i,\\ y_i-\left[\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)+b\right]\le\varepsilon+\xi_i^*,\\ \xi_i,\xi_i^*\ge 0,\\ \varepsilon\ge 0.\end{cases} \quad (1)$$

Where $\boldsymbol{\phi}(\boldsymbol{x}_i)$ is kernel function that performs mapping of input space to high dimensional feature space (the space where linear regression is performed); $\boldsymbol{w}$ is a parameter vector of *n*-dimensional hyperplane; *b* is hyperplane bias parameter; ξ_i^*, ξ_i are upper and lower training errors (slack variables) subject to ε – insensitive tube; *C* is a cost parameter, that controls the trade-off between allowing training errors and forcing rigid margins; *v* is regularization parameter that controls parameter number of support vectors; *l* – is number of data points (observations). Data points that lie on the boundaries of ε – insensitive tube are called *support vectors*.

In this research we use *v*-SVR code that is implemented in *LIBSVM* library (see [14]).

3.1.2 LSSVR

Least squares support vector regression optimization problem is formulated by following equations [15]:

$$\min_{\boldsymbol{w},b,e}\left\{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}+\frac{\gamma}{2}\sum_{i=1}^{l}e_i^2\right\}$$
$$\text{subject to}\begin{cases}\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)+b+e_i=y_i,\\ \gamma\ge 0.\end{cases} \quad (2)$$

Where e_i are error variables and γ is regularization constant.

In this research we use LSSVR code from *LS-SVMlab* toolbox presented in [16].

Differently from *v*-SVR, LSSVR has no insensitive tube and is only regularized by one parameter (γ). The loss function is quadratic for LSSVR and so the sparseness property for LSSVR is lost. For both *v*-SVR and LSSVR in this research we use Gaussian kernel function with one dispersion parameter that needs to specify (σ). So for *v*-SVR there will be total three parameters (*v*, *C*, σ) and for LSSVR two parameters (γ, σ) to specify.

3.2 Relevance vector regression

Relevance vector regression [17] is model that has same linear functional form as support vector regression:

$$y(\boldsymbol{x};\boldsymbol{w})=\sum_{i=1}^{M}w_iK(\boldsymbol{x},\boldsymbol{x}_i)+w_0. \quad (3)$$

Where $K(\boldsymbol{x},\boldsymbol{x}_i)$ is defined as kernel function and $\boldsymbol{w}$ is model weight vector.

Despite that SVR (except for LSSVR) is sparse model, RVR uses even less support vectors (is more sparse) that are called *relevance vectors* because of Bayesian inference methodology that is used during model parameter and relevance vector determination. RVR uses EM-like (expectation-maximization) learning algorithm and applies *a priori* distributions (because of Bayesian methodology) over parameters without need to be specified with some external parameters by user. With RVR we also use Gaussian kernel as in both LSSVR and *v*-SVR cases.

In this research we use RVR implemented in *SparseBayes* package by author himself (see [17]).

3.3 Feed-forward neural network

We also apply feed-forward neural network (FFNN) models for cash demand forecasting with logarithmic sigmoid transfer functions in the hidden layer and linear transfer function in the output layer. For neural network training we use two backpropagation [18] – based algorithms: 1) Levenberg – Marquardt backpropagation [19]; 2) Levenberg – Marquardt backpropagation with Bayesian regulation [20].

Levenberg – Marquardt backpropagation training algorithm is frequently used as the most effective training algorithm for function approximation (regression) problems and uses mean-squared-error (MSE) cost function. This algorithm employs Jacobian matrix (**J**) w.r.t. network weights. An update of network weights using network output error (*e*) is calculated by formula:

$$\Delta\mathbf{w}=-\frac{\mathbf{J}^{\mathrm{T}}e}{\mathbf{J}^{\mathrm{T}}\mathbf{J}+\mu\mathbf{I}} \quad (4)$$

Where **I** is the identity matrix.

The main idea of this algorithm is to interpolate between Gauss – Newton and gradient descend algorithms by controlling scalar value μ.

Levenberg – Marquardt backpropagation with Bayesian regularization updates weights and bias values according to Levenberg – Marquardt optimization, but it minimizes a linear combination of squared errors and weights. The weight term in loss function doesn't let network to overfit and it corresponds to Bayesian regularization.

Both Levenberg – Marquardt backpropagation and Levenberg – Marquardt backpropagation with Bayesian regularization use validation set as stopping criterion in order to speed up training process. As parameters we use number of neurons in hidden layer. For forecasted value estimation we use average ensemble of neural networks, because of random weight initialization during training.

In this research we use *MATLAB Neural Network Toolbox* for FFNN forecasting.

3.4 Generalized regression neural networks

Generalized Regression Neural Network (GRNN) [21] is special case of radial basis function (RBF) neural network. It's first layer is the same as for radial basis neural network, but second layer is different.

GRNN does not require an iterative training procedure (error back propagation). Training procedure requires only specification of radial basis function spread parameter. It uses radial basis functions to cover input space and approximates function as weighted linear combination of radial basis functions. Number of RBF function is equal to number of observations. Each RBF is formed for each data point vector that is a center of RBF. RBF transfer function values are calculated according to input value Euclidean distance from the central point.

In this research we use GRNN implemented in *MATLAB Neural Network Toolbox*.

3.5 Adaptive neuro-fuzzy inference system

Adaptive neuro-fuzzy inference system (ANFIS) [22] is a combination of neural network and fuzzy inference system features. ANFIS model architecture with two membership functions is depicted in Fig. 1. ANFIS architecture has fuzzy layer (1), product layer (2), normalization layer (3), defuzzification layer (4) and summation layer (5).

For a 1st order of Sugeno fuzzy model, a typical ***IF-THEN*** rule set can be expressed as:

1) ***IF*** *x is* A_1 ***AND*** *y is* B_1 ***THEN*** $f_1 = p_1x + q_1y + r_1$;
2) ***IF*** *x is* A_2 ***AND*** *y is* B_2 ***THEN*** $f_2 = p_2x + q_2y + r_2$.

Further each of five layer functionality is shortly explained:

1 layer. Forms output, which determines membership degree in each of membership functions (μ_{A1}, μ_{A2}, μ_{B1}, μ_{B2}):

$$O_{1,i} = \mu_{A_i}(x), i = 1, 2, \quad (5)$$

$$O_{1,i} = \mu_{B_{i-2}}(y), \quad i = 3, 4. \quad (6)$$

2 layer. In this layer each node is fixed and represents weight of particular rule. In each node ***AND*** operation is performed, which is product of inputs:

$$O_{2,i} = w_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y), \quad i = 1, 2. \quad (7)$$

3 layer. Each node of this layer is also fixed and calculates normalized rule excitation degree:

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2. \quad (8)$$

4 layer. This layer is not fixed as other and parameters (p_i, q_i, r_i) are estimated during training process. Output of nodes are calculated as:

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i(p_i x + q_i y + r_i). \quad (9)$$

5 layer. This is an output layer, where output value is calculated as a sum of all inputs:

$$O_{5,1} = \sum_i \overline{w}_i f_i = \frac{\sum_i \overline{w}_i f_i}{\sum_i w_i}. \quad (10)$$

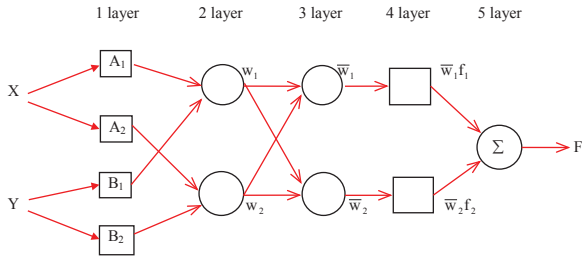

Fig. 1. Illustration of ANFIS model architecture with two membership functions.

ANFIS model training is usually performed using two training algorithms: gradient steepest descend backpropagation or hybrid algorithm. Hybrid learning combines gradient descend backpropagation and least squares methods. Backpropagation is used for parameters that are related with input membership functions, while least squares is applied for output function parameters (that are linear w.r.t. parameters). In this research we use both training algorithms and select best using validation set.

For membership function parameter initialization we use FCM (fuzzy c-means) clustering that extract set of rules that model input data behavior. So for ANFIS model number of clusters is parameter that we also select using validation set.

In this research we use ANFIS model that is implemented in *MATLAB Fuzzy Logic Toolbox*.

3.6 Extreme learning machines

Extreme learning machines (ELM) [23] are another type of single-hidden feed-forward neural networks that randomly chooses hidden nodes and analytically determines the output weights. Main advantage of this kind of learning over traditional backpropagation learning used in neural networks is speed.

Given N number of observations ($\boldsymbol{x}_i$, y_i), single layer neural network output with M hidden nodes is modeled as:

$$o_i = \sum_{j=1}^{M} \beta_j f(w_j x_i + b_j). \quad (11)$$

Where $\boldsymbol{x}_i$ is ith input vector; $\boldsymbol{w}_j$ is weight vector connecting the jth hidden node and the input nodes; β_j is weight scalar connecting jth hidden node and output node; b_j is bias parameter of jth hidden node.

In this research we use linear output nodes and sigmoid hidden nodes. Above equation can be written in vector form:

$$\mathrm{o} = \mathrm{H}\beta. \tag{12}$$

Where $\mathbf{H}$ is $N \times M$ hidden layer output matrix and $\mathbf{H}_{i,j} = f(\boldsymbol{w}_j \boldsymbol{x}_i + b_j)$.

The solution of applying extreme learning machines theory is simply estimated as:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\boldsymbol{o}\,. \tag{13}$$

Where $\mathbf{H}^{\dagger} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}$ is Moore – Penrose generalized inverse (pseudoinverse) matrix.

Because of speed of ELM we use larger average ensemble than with FFNN models to estimate forecasted value. As a parameter estimated using validation set for ELM we use number of neurons in hidden layer.

A *MATLAB* implementation of classical ELM is used in this research, which is available at webpage (see [23]).

4 Experimental data and methodology

Experimental data consist of 200 ATM real world daily cash demand time series, for historical period equal 26 months. Historical data period used for training varies from 6 months to 2 years (6, 12, 18 and 24 months) and forecasting one day ahead is performed for two months for each of 200 ATM daily cash demand time series. 10-fold cross-validation procedure is used for input and parameter selection with training set for every forecasting model. As forecasting accuracy measure we use symmetric mean absolute percentage error:

$$SMAPE = \frac{100}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{0.5(|\hat{y}_i| + |y_i|)}\,. \tag{14}$$

Where $\hat{y}_i$ is ith predicted value and y_i ith true value.

For all CI models following inputs were used: 1) week number; 2) day of the month; 3) cash flow value one day before $(i-1)$; 4) cash flow value 7 days before $(i-7)$; 5) cash flow value 14 days before $(i-14)$; 6) cash flow value 21 day before $(i-21)$; 7) cash flow value 28 days before $(i-28)$; 8) sum of cash flow values for last 5 days. All those 8 inputs were categorized into four groups with following input sets: {1, 2}; {3}; {4, 5, 6, 7}; {8} (it was decided to do so in order to save computational time and preliminary experiments showed that this way of categorizing is reasonable). All those four groups were used for feature selection (when using 10-fold cross-validation with parameter selection) concluding to 15 different feature set combinations (using binomial formula $\sum_{k=1}^{n}\binom{n}{k} = 2^n - 1 = 2^4 - 1 = 15$) for reduced number of feature selection, whereas considering each of 8 features separately would conclude to 255 different feature combinations which was not accepted for practical purposes.

The example of cash demand time series is depicted in Fig. 2. This illustration shows the complexity of ATM cash flow demand and seasonality patterns: mixture of amplitude varying yearly, monthly and weekly seasonality (it is seen from autocorrelation function) including nonlinear trend that varies for different ATMs and also nonstationary noise which represents uncertainty degree in the cash demand process.

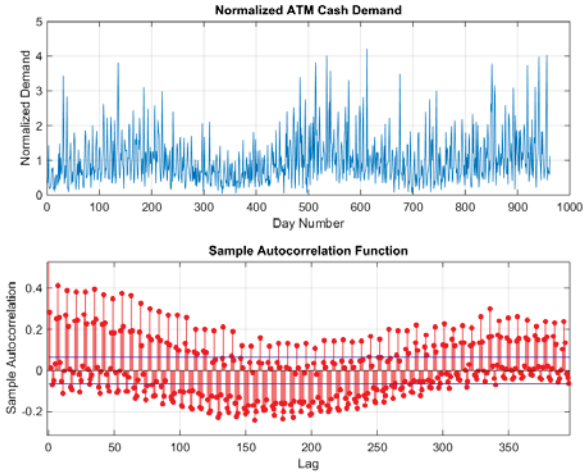

Fig. 2. An example of ATM cash demand.

5 Results

Overall forecasting results are presented in Table 1. It is seen that on average most accurate model is ν-SVR with 2 year training/validation dataset. Also results suggest that using 2 years of historical daily data yields best forecasting results over all models. However, interesting result is that using 18 month (1,5 year) history yields slightly less accurate forecasting results than using 12 month (1 year). This suggests that there was some disturbance event in the history that significantly affected data generating process. However as expected 6 month (0,5 year) data history significantly worsens forecasting accuracy of every forecasting model.

Table 1. Forecasting mean SMAPE (%) results

-	2 year training	1,5 year training	1 year training	0,5 year training	Avg.
ANFIS	44,12	44,24	44,17	45,71	44,56
ELM	44,18	44,20	44,00	45,95	44,58
LMBR-FFNN	44,03	44,13	44,26	45,74	44,54
LM-FFNN	43,87	44,07	44,10	45,61	44,41
GRNN	44,14	44,29	43,97	45,08	44,37
LS-SVR	43,85	44,05	43,88	45,28	44,27
***v*-SVR**	**43,72**	43,73	44,12	45,62	44,30
RVR	44,98	45,12	45,12	45,94	45,29
Avg.	44,11	44,23	44,20	45,62	-

Fig. 3 illustrates SMAPE distribution forecasting results using 2 year historical period for every model

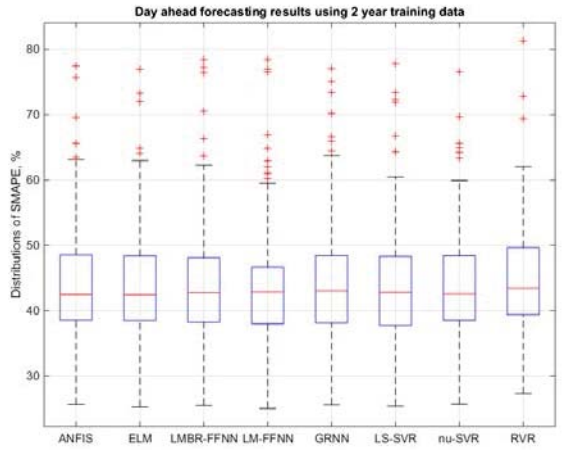

Fig. 3. SMAPE distribution for each forecasting model (2 year training case).

Table 2 show what percentage of all 200 ATMs every model gave best forecasting results. For example 10% show that model gave best forecasting results among all models for 20 ATMs. The results show that *v*-SVR most often yield best accuracy for all training cases.

Table 2. Percentage of how often each model was most accurate for every training case separately.

-	2 year training	1,5 year training	1 year training	0,5 year training
ANFIS	9,0%	7,0%	15,0%	13,5%
ELM	8,5%	11,0%	10,5%	6,0%
LMBR-FFNN	11,0%	9,5%	10,5%	10,5%
LM-FFNN	14,5%	11,5%	11,0%	11,5%
GRNN	16,0%	15,0%	13,5%	10,5%
LS-SVR	8,0%	7,0%	11,5%	16,0%
***v*-SVR**	21,0%	28,5%	18,5%	19,0%
RVR	12,0%	10,5%	9,5%	13,0%

Fig. 4 show the relationship between ATM average (median) cash demand and forecasting accuracy averaged over all models. Illustration clearly confirms the aggregation advantage: uncertainty approaches minimum (forecasting increases) as aggregation of more population values (aggregation of larger cash demand amounts) takes place.

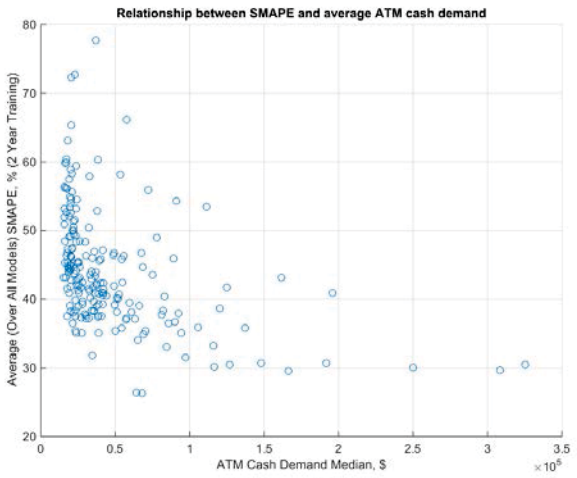

Fig. 4. Relationship between forecasting accuracy (averaged over all models) and ATM cash demand median (2 year training case)

6 Conclusions and future works

One day ahead forecasting results show that *v*-SVR performs best compared to other models when using adaptive input selection. Also results confirm that using longer history can increase forecasting accuracy. However the relationship between historical data length and average forecasting accuracy was not smooth, showing that 1 year historical data period sometimes is better than using 1,5 year historical data. This suggests that some important factors affecting data generating process took place and deeper investigation of those factors (that can be related to structural breaks) is needed.

In the future works we are planning to investigate more deeply joint forecasting approach, when cash demand information of few or more ATMs is integrated or aggregated to increase forecasting accuracy. Also this study only contained one day ahead forecasting, when more often multiple step ahead forecasting is needed. We also aim to investigate multiple step ahead forecasting strategies with CI models more deeply in the future.

7 References

[1] Rodrigues, P. & Esteves, P. (2010). Calendar effects in daily ATM withdrawals. *Economics Bulletin*, *30*(4), 2587–2597.

[2] Kumar, P. & Walia, E. (2006). Cash forecasting: and application of artificial neural networks in finance. *International Journal of Computer Science & Applications, 3*(1), 61–77.

[3] Simutis, R., Dilijonas, D., Bastina, L. & Friman, J. (2007). A flexible neural network for ATM cash demand forecasting. In M. N. Katehakis, D. Andina & N. Mastorakis (Eds.), *Proceedings of the sixth WSEAS international conference on computational intelligence, man-machine*

systems and cybernetics (CIMMACS 07) (pp. 162–165), Tenerife, Spain: WSEAS.

[4] Simutis, R., Dilijonas, D. & Bastina, L. (2008). Cash demand forecasting for ATM using neural networks and support vector regression algorithms. In L. Sakalauskas, G. W. Weber & E. K. Zavidskas (Eds.) *Proceedings of the twentieth EURO mini conference on continuous optimization and knowledge-based technologies (EurOPT-2008)* (pp. 416–421), Neringa, Lithuania: VGTU press Technika.

[5] Darwish, S. M. (2013). A methodology to improve cash demand forecasting for ATM network. *International Journal of Computer and Electrical Engineering*, *5*(4), 405–409.

[6] Gurgul, H. & Suder, M. (2013). Modeling of withdrawals from selected ATMs of the Euronet network. AGH Managerial Economics, 13, 65–82.

[7] Wagner, M. (2010). Forecasting daily demand in cash supply chain. *American Journal of Economics and Business Administration*, *2*(4), 377–383.

[8] Brentnall, A. R., Crowder, M. J. & Hand, D. J. (2010b). Predictive-sequential forecasting system developement for cash machine stocking. *International Journal of Forecasting*, *26*, 764–776.

[9] Brentnall, A. R., Crowder, M. J. & Hand, D. J. (2008). A statistical model for the temporal pattern of individual automated teller machine withdrawals. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *57*(1), 43–59.

[10] Brentnall, A. R., Crowder, M. J. & Hand, D. J. (2010a). Predicting the amount individuals withdraw at cash machines using a random effects multinomial model. *Statistical Modeling*, *10*(2), 197–214.

[11] Laukaitis, A. (2008). Functional data analysis for cash flow and transactions intensity continuous-time prediction using Hilbert-valued autoregressive processes. *European Journal of Operational Research*, *185*, 1607–1614.

[12] Ekinci, Y., Lu, J. C. & Duman E. (2015). Optimization of ATM cash replenishment with group-demand forecasts. *Expert Systems with Applications*, *42*, 3480–3490.

[13] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

[14] Chih-Chung, C. & Chih-Jen, L. (2011). LIBSVM : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*(27), 1–27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[15] Suykens, J. A. K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J. (2002). *Least Squares Support Vector Machines*. Singapore: World Scientific.

[16] Pelckmans, K., Suykens, J. A. K., Van Gestel, T., De Brabanter, J., Lukas, L., Hamers, B., De Moor, B., Vandewalle, J. (2002). LS-SVMlab : a Matlab/C toolbox for Least Squares Support Vector Machines. *Internal Report 02-44, ESAT-SISTA*. KU Leuven (Leuven, Belgium).

[17] Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–244. Software available at http://www.miketipping.com/sparsebayes.htm.

[18] Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagation errors. *Nature*, *323*, 533–536.

[19] Hagan, M. T., & Menhaj, M. (1994). Training feed-forward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, *5*(6), 989–993.

[20] Dan Foresee, F. & Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian learning. In *Proceedings of the 1997 International Joint Conference on Neural Networks* (pp. 1930–1935).

[21] Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks*, *2*(6), 568–576.

[22] Jang, J.-S. R. (1993). ANFIS: adaptive-network-based fuzzy inference systems. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(3), 665–685.

[23] Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, *70*, 489–501. Software available at http://www.ntu.edu.sg/home/egbhuang/elm_codes.html.