Smart Grid Load Forecasting Using Online Support Vector Regression

Petra Vrablecová[a], Anna Bou Ezzeddine[a], Viera Rozinajová[a], Slavomír Šárik[a], Arun Kumar Sangaiah[b]

[a] Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava, Slovakia
[b] School of Computing Science and Engineering, VIT University, Vellore 632014, Tamil Nadu, India

Abstract. Smart grid, an integral part of a smart city, provides new opportunities for efficient energy management, possibly leading to big cost savings and a great contribution to the environment. Innovations in the power grid and liberalization of the electricity market have significantly changed the character of data analysis in power engineering. Online processing of large amounts of data continuously generated by the smart grid can deliver timely and precise short term power load forecasts, which are essential to control power supply and balance in the grid. They are an important input for interactions on the electricity market where the energy can be contracted even one hour ahead of its consumption to minimize the grid imbalances. In this paper, we present our research in the area of short term power load forecasting methods. The contribution of this paper is two-fold. Firstly, we demonstrate the suitability of online support vector regression (SVR) method to short term power load forecasting and thoroughly explored its pros and cons. Secondly, we present a comparison of ten state-of-the-art forecasting methods in terms of accuracy on public dataset – a valuable asset to load forecasting community. Online SVR was able to achieve accuracy of complex tree-based ensemble methods and advanced online methods.

Keywords: smart grid, short term load forecasting, support vector regression, online processing

https://doi.org/10.1016/j.compeleceng.2017.07.006

1 Introduction

Smart grid has brought new ways of energy management (Kaitovic et al., 2015). The smart grid research branched into several directions – from the core technologies, sensors, network, communication (Li and Zhang, 2014), through grid security (Mahmood et al., 2016), to storing, processing and mining of the smart grid data. In order to get real value from the smart grid, a thorough analysis and processing of a huge amount of generated data has to be performed. The ultimate goal of smart energy approach is to control energy supply and balance effectively. From this point of view, precise prediction models are becoming very important for all stakeholders of the energy market. When considering the fact that 40% of electrical energy is used in the buildings (Sustainable Buildings and Climate Initiative and United Nations Environment Programme, 2016), then it is clear that even a small improvement in prediction accuracy means big cost savings and also a great contribution to the environment.

Smart meters produce streaming data, therefore the processing methods require mining techniques which are different from the classical ones. When considering the main characteristics (3V) of big data, the smart meter data fulfill these characteristics – particularly when the smart meters will be fully deployed (in Europe the plan is to deploy them till 2020). Even a small improvement of the prediction and optimization methods in energy domain means big savings not only from economic, but also from environmental point of view. In order to take informed decisions concerning the smart grid operation, the only way is to process these data online. Therefore, we have decided to investigate online Support Vector Regression method. The ultimate limitations of stream processing are memory and time. Since the stream contains an infinite number of records, it is not possible to store them all in memory or read them more than once. Hence, the stream processing method should work incrementally and allow online processing. The learning algorithms should be able to process new data without intensive usage of the already considered data and also to forget and unlearn old or obsolete information. In order to create flexible models, the techniques should be adaptive – i.e., they should adjust to the features of the current data flow.

In this paper, we present our research in the area of power load forecasting methods. The relevance and usefulness of power load forecasts have been increasing, considering the recent trends of liberated energy market allowing to trade electricity even one hour ahead. To minimize the energy imbalance (i.e., the online gap between contracted supply and actual demand) and the costs associated with it, the market interactions based on accurate power load forecasts need to be performed promptly. Power load forecasting methods that are able to process online stream of data from smart meters became a subject undergoing intense study. We have studied several forecasting methods that are based on statistical, as well as on artificial intelligence, approaches and have compared them with each other. We focused on support vector regression (SVR), because it was established that it is a method with very good accuracy (Humeau et al., 2013). We employed its less known online variant, which needs to store less amount of data than the standard SVR, as it is able to forget certain data. The method is particularly suitable for short-term predictions (i.e., for the next hour or for the next day).

The presented ideas are designed for power engineering domain and they have been verified there. However, many of the proposed methods are applicable also in other areas of smart cities. Vast amounts of data are generated by city systems (e.g., using IoT technologies): transportation data, data about water and gas consumption, social and economic datasets, satellite data, city governance data, social media data, etc. As the character of the generated data is in many cases similar to the smart meter data, the prediction models and optimization methods could be easily adopted.

The paper is structured as follows: Section 2 introduces the state of the art in power load prediction methods and support vector regression applications. The online support vector regression is described in Section 3, whereas the experiments and their evaluation can be found in section 4. We discuss our results in section 5, and the conclusion is in the last section.

2 Related work

The development of power load forecasting as a field started at the end of the 19th century with the invention of a light bulb and the widespread electricity utilization. The forecasting methods evolved from charts and tables to engineering approaches used today. The artificial intelligence affected the field greatly. The most employed approaches for short-term power load forecasting can be, in general, divided into statistical techniques, such as *regression analysis* and *time series analysis*; and *artificial intelligence* (Hong, 2014).

The review by Singh et al. (2012) classifies load forecasting techniques into *traditional forecasting techniques* (e.g. simple and multiple regression, exponential smoothing, iterative reweighted least squares), *modified traditional techniques* (e.g. adaptive demand forecasting, stochastic time series analysis, mainly Box-Jenkins models, such as AR, ARMA and ARIMA, support vector machines based models), and *soft computing techniques* that include artificial intelligence approaches like genetic algorithms, fuzzy logic, neural networks, and knowledge-based systems. The review reports major advantages of soft computing techniques, and growing use of hybrid methods, which combine two or more of these techniques.

In our research, we focus on support vector regression, because its characteristic property is the ability to model nonlinear time series, such as power load series, in a high dimensional feature space via the kernel trick, in which the training data may exhibit linearity (Drucker et al., 1996). Also, it was reported to be a very accurate forecasting method (Humeau et al., 2013). We studied also the performance of other load forecasting techniques to create a self-contained comparison. In the literature, we encountered only a small number of similar reports comparing multiple power load forecasting techniques. Taylor and McSharry (2007) published an empirical comparison of univariate methods for one-day ahead forecasting. Among ARIMA, exponential smoothing and PCA-based methods, the double seasonal Holt-Winters exponential smoothing method (DSHW) was the best. In the more recent comparison of load forecasting methods (Mirowski et al., 2014), the best results were obtained by an ensemble of DSHW and kernel ridge regression (also called sigma support vector regression).

In the next subsection, we briefly describe the techniques that we chose based on their generally good results in forecasting. We aimed for a variety of methods that represent all of the main groups of load forecasting techniques.

2.1 Overview of load forecasting methods

Double Seasonal Holt-Winters Exponential Smoothing (DSHW)

DSHW is an extension of Holt-Winters exponential smoothing method (Winters, 1960). It was introduced by Taylor (2003). Exponential smoothing predicts future values of a time series as a weighted average of past values. The weights of the values decay exponentially as the observations get older. The forecast by DSHW is a combination of 4 time series' components; each is smoothed by its smoothing parameter: level (α), trend (β), and two seasonal components (daily - δ and weekly - ω). We used its multiplicative variant. To improve the forecast accuracy, Taylor included a simple

adjustment for first-order autocorrelation (ϕ). The forecast by DSHW for kth horizon from time t can be expressed as (1).

$$forecast_{t+k} = (level_t + k \cdot trend_t) \cdot dailyseas_{t-ds+k} \cdot weeklyseas_{t-ws+k} + \phi^k \cdot error_t \quad (1)$$

ds and *ws* are lengths of daily and weekly cycles, e.g. 48 and 336 for half-hourly load measurements, $error_t$ is the difference between the actual and forecasted value at time *t*.

Seasonal decomposition of time series by Loess (STL)

STL (Cleveland et al., 1990) decomposes a seasonal time series into three parts: trend, seasonality and remainder. The seasonal component is found by Loess (local regression) smoothing of the original time series. The rest is smoothed to find the trend. The remaining component represents the residuals from the seasonal plus trend fit. These resulting 3 time series were separately forecasted by Holt-Winters exponential smoothing (STL+EXP) and ARIMA (STL+ARIMA) model from Box-Jenkins' methodology (Box et al., 1970). The components' forecasts were summed to obtain the final forecast.

Support Vector Regression (SVR)

SVR tries to find a linear regression function $f(x)$ (2) that can best approximate the actual output vector with an error tolerance ε, and is as flat as possible. Most of the real-world problems (like load forecasting) have a nonlinear solution, therefore SVR maps the input data into a higher dimensional feature space, in which the training data may exhibit linearity, and then linear regression can be used in this new feature space (Drucker et al., 1996).

$$f(x) = w \cdot \phi(x) + b \quad (2)$$

The function $\phi(x)$ is a mapping function from nonlinear to linear space, i.e. a nonlinear kernel function. b is a bias term. To ensure that $f(x)$ is as flat as possible, a function with minimal norm value $\|w\|^2$ has to be found, such that all residuals have a value less than ε. Such a function might not exist and therefore a cost for residuals greater than ε has been introduced. This can be formulated as the primal optimization problem of the nonlinear ε-insensitive support vector regression (ε-SVR) (3).

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \quad \text{subject to} \quad \begin{aligned} y_i - f(x) &\le \varepsilon + \xi_i \\ f(x) - y_i &\le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* &\ge 0 \end{aligned} \quad (3)$$

C controls the penalty imposed on residuals, which have value greater than ε by ξ_i, resp. ξ_i^*.

By construction of a Lagrangian function from the primal function with Lagrangian multipliers α and α*, the dual optimization problem of ε-SVR can be obtained. The function $f(x)$ is then defined as (4).

$$f(x_i) = \sum_{j=1}^{\ell}(\alpha_j - \alpha_j^*)K(x_i, x_j) + b \quad (4)$$

$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ is a nonlinear kernel function, in this paper we refer to it as Q_{ij}. The difference $(\alpha_j - \alpha_j^*)$ is in this paper denoted as θ_j. The ε-SVR function $f(x)$ is then defined as (5).

$$f(x_i) = \sum_{j=1}^{\ell}\theta_j Q_{ij} + b \quad (5)$$

Each sample in the model must also comply with Karush-Kuhn-Tucker (KKT) conditions. They indicate that all samples inside the ε margin have zero Lagrangian multipliers. Samples with the nonzero multipliers are the *support vectors*.

Additional attributes to the load measurements' training set for this model were binary (dummy) variables representing the time of day (number of variables equaled number of measurements per day)

and weekday (7 variables). A variable equaled 1 only when the current measurement was taken at the time of the day/day of the week, which the variable represented.

We tested several kernels and values of input parameters. The best forecast results were obtained by the (Gaussian) radius basis function (RBF) kernel (6).

$$Q_{ij} = K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) = \exp\left(-\gamma \cdot \|x_i - x_j\|^2\right) \quad (6)$$

x_i and x_j are input samples and σ^2 is variance. $\|x_i - x_j\|^2$ can be identified as the squared Euclidean distance between two samples. σ is a free parameter of RBF, which is often hidden in a simplified definition as $\gamma = \frac{1}{2\sigma^2}$.

Multilayer Perceptron (MLPnet)

Multilayer perceptron is a modification of the standard linear perceptron method, which is able to distinguish not linearly separable data. This artificial neural network model maps sets of input data onto a set of appropriate outputs. In general, it consists of multiple layers of nodes in a directed graph, whereby each layer is fully connected to the next one. All nodes are neurons with a nonlinear activation function (excluding the input nodes). The method is based on a backpropagation technique for training the network (Cybenko, 1989; Rumelhart et al., 1986).

Deep Learning (DLnet)

Deep learning belongs to a broad family of machine learning methods that are based on learning representations of data (Goodfellow et al., 2016; Schmidhuber, 2015). The algorithms attempt to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations.

Additional attributes for the neural network models were load and de-noised load (seasonal plus trend component from STL decomposition) with one-week lag.

Bagging (BAGG)

Bagging (i.e. bootstrap aggregating) predictors generate multiple versions of predictors and use them to determine an aggregated predictor (Breiman, 1996). The aggregation is an average of all predictors. The multiple versions are formed by generation of bootstrap replicates of the training set and using these as new training sets. The bagging method gives substantial gains in accuracy, but a problem is the instability of the prediction method. In the case where perturbing the learning set has significant influence on the constructed predictor, bagging can improve accuracy.

We have used regression tree as a predictor. The training set consisted of historical power load measurements supplemented with daily and weekly seasonal data vectors that defined the time and weekday of each measurement. The vectors were defined as sequences $dsv = (1,2,\dots,s,1,2,\dots,s,\dots)$ and $wsv = (1_1, 1_2, \dots, 1_s, 2_1, 2_2, \dots, 7_{s-1}, 7_s, 1_1, 1_2, \dots)$, where s is the number of measurements per day. The length of the sequences equaled the length of the training set.

Extremely Randomized Trees (XRT)

Extremely randomized trees is a tree-based ensemble method designed for solving supervised classification and regression problems (Geurts et al., 2006). It randomizes both attribute and cut-point choice during splitting the tree nodes. In the extreme case, it builds totally randomized trees which structures are independent of the output values of the learning sample. The influence of the randomization can be seen as a problem of the proper parameter choice.

The training set of the model was supplemented with daily and weekly seasonal vectors (see Bagging).

Random Forest (RF)

The original idea of random decision forests was created by Tin Kam Ho (1998) using the random subspace method. An extension of the algorithm was developed by Breiman (2001) and Adele Cutler (Liaw, 2015). The random forest algorithm designed by Breiman is suitable for classification and regression. The method constructs the large number of decision trees at training time. Its output is the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Extreme Gradient Boosting (XGB)

Extreme gradient boosting is another tree-based ensemble method. It produces new models iteratively in order to form an ensemble of "weak" models. Each next model is created to minimize the loss function calculated from the residuals of the previous ensemble. It supports various objective functions, including regression, classification and ranking. Gradient boosting of regression trees is especially appropriate for mining non-preprocessed data (Friedman, 2001).

Additional attributes for RF and XGB models were 4 seasonal data vectors for daily and weekly periods obtained as a sine and cosine (7) function of values of vectors dsv and wsv (see Bagging); and de-noised load (seasonal plus trend component from STL decomposition) with one-week lag.

$$\left(\frac{sin\left(2\pi\frac{dsv_i}{s}\right)+1}{2}\right)_{i=1}^{|dsv|}, \left(\frac{cos\left(2\pi\frac{dsv_i}{s}\right)+1}{2}\right)_{i=1}^{|dsv|} \text{ and } \left(\frac{sin\left(2\pi\frac{wsv_i}{7}\right)+1}{2}\right)_{i=1}^{|wsv|}, \left(\frac{cos\left(2\pi\frac{wsv_i}{7}\right)+1}{2}\right)_{i=1}^{|wsv|} \quad (7)$$

Table 1. Overview of compared forecasting methods with used parameters.

abbreviation	*method name*	*additional attributes and model parameters*
DSHW	double seasonal Holt-Winters smoothing	smoothing parameters α, β, δ, ω, and ϕ *
STL+ARIMA	seasonal and trend decomposition using Loess + autoregressive moving average model	ARIMA parameters (p, q, d) *
STL+ES	STL + exponential smoothing	smoothing parameter α *
SVR	support vector regression	dummy time and day-of-week variables $C = 1000$, $\varepsilon = 0.06$ (for separate workday/weekend model $\varepsilon = 0.99$)
MLP	multilayer perceptron network	load and de-noised load with one-week lag 3 hidden layers (96, 48 and 24 neurons) maximum number of training epochs: 1200 activation function: tanh learning function: backpropagation learning rate: 0:2
DLnet	deep learning neural network	load and de-noised load with one-week lag 3 hidden layers (96, 48 and 24 neurons) maximum number of training epochs: 1400 activation function: tanh learning rate: 0:01 response distribution: Gaussian
BAGG	bagging	daily and weekly seasonal vectors number of trees: 1000 minimum size of leaves: 5 maximum tree depth: 8
XRT	extremely randomized trees	daily and weekly seasonal vectors number of trees: 500

		size of leaves: 5 no subsetting of samples
RF	random forest	4 daily and weekly, sine and cosine seasonal vectors de-noised load with one-week lag number of trees: 1100 minimum size of terminal nodes: 3 size of training samples: 63.2% of the training data number of randomly sampled variables at each split: 3
XGB	extreme gradient boosting	4 daily and weekly, sine and cosine seasonal vectors de-noised load with one-week lag task: linear regression number of iterations for boosting: 1000 step size shrinkage in update: $\eta = 0{:}02$ maximum depth of a tree: 8

** model parameters were estimated on regular basis from sliding window during incremental forecasting.*

2.2 Power load forecasting with Support Vector Regression

The SVR algorithm was used for the first time for power load forecasting in the 2001 EUNITE competition (Chen et al., 2004). Since then, it has been applied and studied by many researchers. A survey (Sapankevych and Sankar, 2009) maps SVR applications for time series forecasting until 2009. Power load forecasting along with financial data analysis appeared to be the most studied ones.

Nowadays, SVR is still widely used and studied for short term power load forecasting. It is often reported to outperform other advanced computing techniques, such as neural networks (Achanta, 2012; Hu et al., 2013; Sousa et al., 2014). Compared to SVR, neural networks are more error-prone and prone to overfitting or to fall to local minima due to larger number of parameters (Sapankevych and Sankar, 2009). They also require larger amounts of training data compared to SVR.

Humeau et al. showed in their work that SVR forecasts aggregated load better than a load of a single household and determined an aggregation size at which SVR outperforms linear regression – the suitable method for single household's load forecasting. SVR also outperformed MLP in this study (Humeau et al., 2013).

The drawback of SVR is that it can make only one-step ahead forecast by default (e.g. one half-hour/hour ahead depending on the frequency of load measurements). To overcome this, several strategies can be used. Measurements can be aggregated into bigger units with smaller frequency (e.g. 15-min. to hourly measurements), but this strategy can result in loss of majority of the data. Iterative strategy uses the forecasted value as SVR input to forecast next horizon consequently, but this strategy can introduce additional error into the forecast. Direct strategy is the usage of several SVR models to predict simultaneously, e.g. 24 SVR models to predict next 24 hours (Sousa et al., 2014). This strategy can be computationally expensive considering the number of models. Recently, a strategy that adjusts the SVR model to have multiple outputs, was presented (Bao et al., 2014).

SVR parameters' (epsilon, cost and kernel parameters) selection is essential to achieve good forecasting accuracy. The search in the whole parameters' space (i.e. tuning) is computationally a very expensive operation. Various heuristics for global optimization (Boussaïd et al., 2013) are employed, e.g. simulated annealing (Sousa et al., 2014), fireflies (Hu et al., 2013), particle swarms (Duan et al., 2011). Extensions and improvements of the conventional heuristics spread rapidly during the past years (Medhane and Sangaiah, 2017; Purnomo and Wee, 2016). The very recent work (Sreekumar et al., 2017) shows that optimization of SVR parameters using both genetic algorithm or particle swarms

yields forecasting models with very high accuracy. However, these heuristics are still time consuming and thus not suitable for situations when the forecasting model needs to be created frequently.

2.3 Online power load forecasting

With the emergence of smart grids, large amounts of data became available for analysis of power load and power demand in real time. The nature of data processing changes from batch to stream. Traditional forecasting methods, that generate static models from a fixed training set, are becoming obsolete in environments with continually arriving data (Zliobaite et al., 2012).

Incremental or online processing methods are required to deal with restrictions of stream processing and providing timely and reliable forecasts. Stream is an open-ended ordered sequence of data that can be read only once (or limited amount of times) within a constrained time and memory resources. The data usually flows at high speed and is generated by non-stationary distributions. After the processing of a sample from a stream, the sample is discarded or archived. Unlike incremental methods, online methods do not keep any data in memory, e.g., in the form of a sliding window, and employ one-by-one processing. An essential feature of stream mining method is *adaptivity* to changes that naturally occur in data over time, also referred to as concept drift, non-stationarity or dynamic environment. Ideally, the forecasting method is able to forget old information learned from data and adapts to changes by making the predictions from current and relevant inputs. Extensive reviews on this topic exist by multiple authors (Gama et al., 2014; Žliobaite, 2010).

The power load is mostly affected by calendar events (workday/weekday/holidays), time of the day, seasons and weather (particularly temperature and humidity). The major deviations are mainly caused by holidays, summer leaves in larger factories, social events on local/national level, etc. (Hong, 2013). The trend of making continual forecasts by incremental or online methods manifested in the field of power engineering as well. The results indicated that forecasts' adaptation using concept drift detector can improve prediction accuracy (Bosnić et al., 2014; Pratama et al., 2016).

Li et al. applied a forecasting model based on Online Sequential Extreme Learning Machine to clusters of consumers with similar behavior. Though the forecasting was performed online, the clustering was not. The proposed method was able to mine electricity behaviors deeply and the clustering improved the accuracy of the forecasting model (Li et al., 2016). The positive effects of clustering of consumers on accuracy was also reported when linear regression, multilayer perceptron network and support vector regression were employed for 1 and 24 hours ahead power load forecasting (Wijaya et al., 2015).

Online power load forecasting is particularly important for energy imbalance reduction. The online energy gap between contracted supply and actual demand can be minimized by timely market interactions between stakeholders of the liberated electricity market. The stakeholders are able to contract electricity even one hour ahead. The costs associated with energy imbalance are high. Inefficient energy management has also negative environmental effects. Therefore, the research of advanced power load forecasting methods, that process online stream of data from smart meters, is stimulated and became a hot topic. Besides power load forecasting methods, other innovative solutions for the energy imbalance have been arising , e.g. online energy storage scheduling (Chakraborty and Okabe, 2016).

In the consecutive sections, we present online support vector regression approach and show its application to short term power load forecasting.

3 Online Support Vector Regression

The traditional approach of forecasting using support vector regression (SVR) is based on batch learning when a set of data is divided in a defined ratio into a training set and a testing set. The input parameters of an SVR model are determined by searching the space of all possible parameter values until an optimal combination is found (i.e. by tuning method such as grid search). The model created this way can make reliable short-term forecasts, but is not applicable to longer forecasts because it is not flexible enough to react to changes and its accuracy decreases over time. This results in need for re-training of the model from scratch, which is, computationally, a very expensive operation that consists of new selection of SVR input parameters and training of the model on the data merged with newly arrived data.

Online SVR was proposed in 2001 (Cauwenberghs and Poggio, 2001) for the first time. Unfortunately, it was not widely accepted then and even today it might be still unknown to many practitioners. Since then, it was followed-up by only a few works (e.g. (Laskov et al., 2006; Ma et al., 2003; Martin, 2002; Montana and Parrella, 2008)). Limited interest in online SVR might be caused by nonexistence of widely used implementations, unlike the well-known implementations of batch SVR algorithm (e.g. LIBSVM (Chang and Lin, 2011), SVM^{light} (Joachims, 1999), SMO (Platt, 1998)). Very recently, an implementation of online SVR has been applied to peak electricity load forecasting, but the work did not state any specific results (Dhillon et al., 2016).

3.1 Method overview

Support vector regression (SVR) is a method of prediction that is based on the principles of an SVM classifier. The aim of SVR is to find a function whose values in the marginal area are closer than the distance of ε to the target. This space is defined by borders, similar to the SVM classifier.

If the value of the error does not exceed the threshold of ε, this error is considered to be zero and can be ignored, as well as we can neglect points far from the boundary hyperplane because they do not affect the results.

Given a training set $T = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, \ell\}\}$, where $\mathbf{x}_i \in \mathbb{R}^N$ and $y_i \in \mathbb{R}$. On a feature space $\mathbb{F}$ we construct a linear regression function (8):

$$f(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}) + b \tag{8}$$

where $\mathbf{w}$ is a vector in $\mathbb{F}$, and $\boldsymbol{\phi}(\mathbf{x})$ maps the input $\mathbf{x}$ to a vector in $\mathbb{F}$. The $\mathbf{w}$ and bias term b in (8) are obtained by solving the optimization problem (3).

A margin function $h(\mathbf{x}_i)$ for the i^{th} sample $\mathbf{x}_i$ is defined as (9):

$$h(\mathbf{x}_i) = f(\mathbf{x}_i) - y_i = \sum_{j=1}^{\ell} \theta_j Q_{ij} + b - y_i \tag{9}$$

where $Q_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T \boldsymbol{\phi}(\mathbf{x}_j)$ is a kernel function (Vapnik, 1995) and $\theta_j = \alpha_j - \alpha_j^*$ is a coefficient difference, where α_j, α_j^* are Lagrangian multipliers.

According to the Karush-Kuhn-Tucker (KKT) conditions (Vapnik, 1995), at most one of α_i and α_i^* will be nonzero, and both are nonnegative. We can deduce from KKT the following conditions for the margin function:

$$h(\mathbf{x}_i) > \varepsilon \qquad \theta_j = -C$$

$$h(\mathbf{x}_i) = \varepsilon \qquad -C < \theta_j < 0$$

$$\begin{aligned} \varepsilon > h(\mathbf{x}_i) > -\varepsilon \qquad & \theta_j = 0 \\ h(\mathbf{x}_i) = -\varepsilon \qquad & 0 < \theta_j < C \\ h(\mathbf{x}_i) < -\varepsilon \qquad & \theta_j = C \end{aligned} \tag{10}$$

The samples in the training set T are in the training process separated into one of three subsets: support vectors (S), error vectors (E) and remaining vectors (R) (Fig. 1).

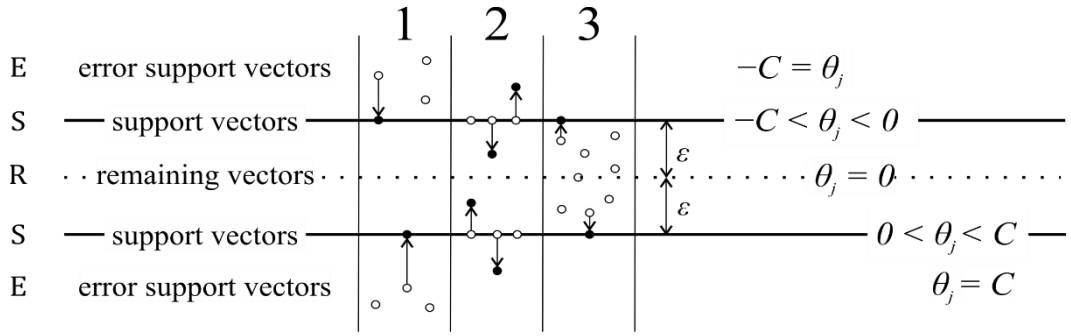

Fig. 1. Geometrical interpretation of possible migrations of vectors in SVR model.

If we want to suggest the exact online support vector machine for regression, we need to define three incremental actions:

- add one new vector,
- remove one vector,
- update one vector.

Adding one new vector

A new vector $\mathbf{x}$ is added by inspecting conditions (10) of an error vector, support vector or remaining vector. The basic idea is to change gradually the coefficient difference θ_{new} corresponding to the new sample $\mathbf{x}_{new}$.

Remove or update one vector

If a vector $\mathbf{x}$ is in the set R, then it does not contribute to the SVR solution, and removing it from the training set is trivial. On the other hand, if the vector is in the S or E set, then the idea is to reduce the value of the θ_{new} to zero gradually. In this case, it is important to ensure that all the other samples in the training set continue to satisfy the KKT conditions (similarly to the vector addition). Therefore, vector migrations happen.

Migration of vectors between sets

One vector can migrate only from its current set to a neighbor set. Fig. 1 shows geometrical interpretation of each set (E, R, S) and possible migrations (1, 2, 3). While one vector remains in E or R sets, its θ value does not change.

Coefficients for migration of vectors between sets

The relation between $\Delta h(\mathbf{x}_i)$, $\Delta\theta_j$ and Δb is given by (11).

$$\begin{aligned} \Delta h(\mathbf{x}_i) &= h\left(\mathbf{x}_{i_{new}}\right) - h\left(\mathbf{x}_{i_{old}}\right) = \\ &= \sum_{j=1}^{\ell} \theta_{j_{new}} Q_{ij} + b_{new} - \sum_{j=1}^{\ell} \theta_{j_{old}} Q_{ij} + b_{old} = \end{aligned} \tag{11}$$

$$= \sum_{j=1}^{\ell} \Delta\theta_j \, Q_{ij} + \Delta b$$

We identified two coefficients that are important for vector migration within subsets (E, R, S):

1. If $\mathbf{x}_i \in S$

$$\Delta h(\mathbf{x}_i) = 0 = \sum_{j=1}^{\ell} \Delta\theta_j \, Q_{ij} + \Delta\theta_{new} Q_{i\,new} + \Delta b \tag{12}$$

$$1 + \frac{\Delta b}{\sum_{j=1}^{\ell} \Delta\theta_j Q_{ij}} = \Delta\theta_{new} \left(- \frac{Q_{i\,new}}{\sum_{j=1}^{\ell} \Delta\theta_j Q_{ij}} \right) \tag{13}$$

$$\frac{\sum_{j=1}^{\ell} \Delta\theta_j Q_{ij} + \Delta b}{\sum_{j=1}^{\ell} \Delta\theta_j Q_{ij}} = \Delta\theta_{new} \cdot coef_1 \tag{14}$$

$$coef_1 = \frac{\sum_{j=1}^{\ell} \Delta\theta_j Q_{ij} + \Delta b}{\Delta\theta_{new} \sum_{j=1}^{\ell} \Delta\theta_j Q_{ij}} = - \frac{Q_{i\,new}}{\sum_{j=1}^{\ell} \Delta\theta_j Q_{ij}} \tag{15}$$

2. If $\mathbf{x}_i \in E$ or $\mathbf{x}_i \in R$

$$\Delta h(\mathbf{x}_i) = \sum_{j=1}^{\ell} \Delta\theta_j \, Q_{ij} + \Delta\theta_{new} Q_{i\,new} + \Delta b \tag{16}$$

$$= \Delta\theta_{new} \left(\sum_{j=1}^{\ell} \Delta\theta_j Q_{ij} \frac{\sum_{j=1}^{\ell} \Delta\theta_j Q_{ij} + \Delta b}{\Delta\theta_{new} \sum_{j=1}^{\ell} \Delta\theta_j Q_{ij}} + Q_{i\,new} \right)$$

$$= \Delta\theta_{new} \left(coef_1 \cdot \sum_{j=1}^{\ell} \Delta\theta_j Q_{ij} + Q_{i\,new} \right)$$

$$= \Delta\theta_{new} \cdot coef_2$$

$$coef_2 = coef_1 \cdot \sum_{j=1}^{\ell} \Delta\theta_j Q_{ij} + Q_{i\,new} \tag{17}$$

To identify the migrations of support vectors (S) we defined coefficient $coef_1$, see (12) – (15).

Migrations of support vectors (S) affect the migrations of vectors in other two subsets (R and E). $coef_2$ (see (16) and (17)) reflects the dependence of vector migrations from subsets R, E on support vector migrations.

The process of online incorporation of a new vector into SVR model is described by the algorithm 1.

Algorithm 1. Online SVR training

```
01: if model contains the threshold number of vectors
02:     until the weight of the oldest vector is not minimal
03:         find vector with minimal weight in model
04:         update (migrate) the vector
05:     remove the oldest vector
06: add new vector to SVR model
07: if it is an inner vector by KKT conditions
08:     end of action
09: else
10:     until the vector does not meet KKT conditions for support or outer vectors
11:         find minimal needed updates (migrations) of vectors in model
12:         update model vectors
```

vector removal

vector addition

The number of vectors in model is limited by a *threshold* value to not keep all incoming vectors in the model and exclude the oldest vectors.

To estimate the input parameters of the SVR model, we chose to use particle swarm optimization (PSO) algorithm (Boussaïd et al., 2013). Sreekumar et al. (2017) demonstrated that PSO optimization of SVR parameters lead to increased prediction accuracy.

4 Evaluation

Our evaluation consisted of a set of experiments focused on:

a. the comparison of accuracy of online SVR and the standard SVR method,
b. the selection of kernel function,
c. the accuracy of very short term forecast by online SVR,
d. the comparison of short term forecast's accuracy of selected state-of-art methods and online SVR,
e. the SVR parameter optimization by biologically inspired algorithms,
f. the computation and memory complexity of online SVR.

4.1 Data and evaluation measures

Data used for evaluation comes from the Smart metering Project of the Commission for Energy Regulation (CER) in Ireland. The project took place in 2007 and its purpose was to undertake trials to assess the performance of smart meters, their impact on consumers' energy consumption and the economic case for a wider national rollout. The data is available in Irish Social Science Data Archive (ISSDA)[1]. Power consumption of over 5,000 households and small and medium enterprises was measured during the years 2009 – 2010 every 30 minutes (i.e. 48 measurements per day). In our experiments, we used aggregated measurements of 3,639 consumers that contained no missing values. We used two test sets of different length:

- 1 month (September 20th to October 20th 2009) and
- 6 months (July to December 2010).

These test sets were used in related papers concerning power load forecasting (Li et al., 2016; Wijaya et al., 2015). Fig. 2 depicts the weekly profile of aggregated data. We used data normalization to $\langle 0,1 \rangle$ interval for forecasting.

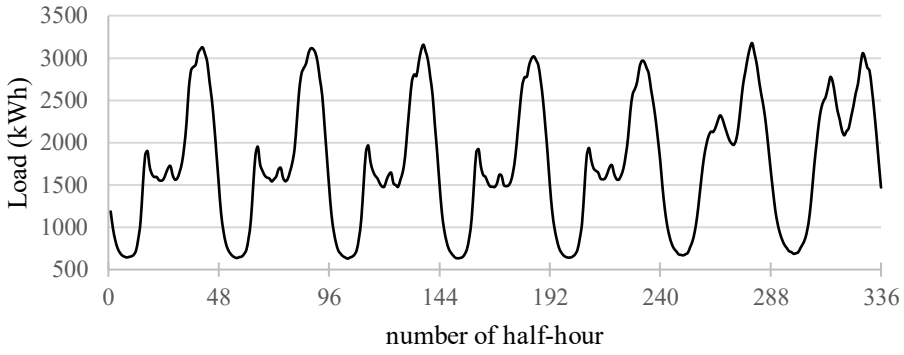

Fig. 2. Mean weekly profile of aggregated data (Mon-Sun).

[1] htttp://www.ucd.ie/issda/data/commissionforenergyregulationcer/

To measure forecast error, we use mean average percentage error, i.e. MAPE measure (18).

$$MAPE = 100 \times \frac{1}{n} \sum_{t=1}^{n} \left| \frac{a_t - f_t}{a_t} \right| \tag{18}$$

where a_t and f_t are actual and forecasted values at time *t*, and *n* is the length of time series.

4.2 Online SVR vs. traditional SVR

The initial experiment was performed on a short sample of data and its purpose was to verify if the online SVR method achieves the same accuracy as the standard SVR method. We forecasted using two SVR models:

- the standard SVR model that was regularly created from scratch; each time a new sample arrived and was added to the training set of the model,
- online SVR model that incorporated the arriving sample into the model and did not have to be re-trained.

We used the identical input parameters for both models: $\varepsilon = 0.01$, $C = 5$, RBF kernel with $\gamma = 0.15$. The MAPE errors of both models on various lengths of trained data are shown in Table 2. The errors are not significantly different. The models differ only in the number of samples that needed to be stored in memory (i.e., batch vs. online processing).

Table 2. MAPE error of standard SVR and online SVR methods on various trained data count.

trained data count	10	20	50	100	500	1000
standard SVR	2.71 %	2.43 %	1.75 %	1.38 %	0.91 %	0.89 %
online SVR	2.85 %	2.47 %	1.76 %	1.42 %	0.89 %	0.89 %

4.3 Feature space mapping

The accuracy of SVR model depends on the chosen kernel function. Typically, the radial basis function (RBF) kernel is used in SVR, so the model is nonparametric, i.e., the RBF kernel maps every input to an infinite dimensional space. This way, the model can incorporate unbounded amount of data to represent more and more complex relationships (constant adaptation).

In our experiment, we forecasted online the power load for the next half-hour during one-month test set (1,488 half-hours) using SVR models with various kernel functions. We compared MAPE of every model to identify the best kernel function for our method (see Table 3). Vectors contained the last 48 measurements (the last day) before the forecasted horizon. The number of vectors in model was limited by $threshold = 500$. The parameters of each model (ε, C, and kernel function parameters) were estimated by PSO algorithm with the swarm size of 35 and the maximum number of 10 iterations on three days preceding the test set.

Table 3. MAPE error of SVR model with various kernel functions.

kernel function	$K(\mathbf{x}_i, \mathbf{x}_j) =$	optimized parameters	MAPE
RBF	$e^{-\gamma \cdot \|\mathbf{x}_i - \mathbf{x}_j\|^2}$	$\varepsilon = 0.001,\ C = 5.309,\ \gamma = 0.266$	2.29 %
polynomial	$(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$	$\varepsilon = 0.001,\ C = 0.304,\ d = 4.14$	2.43 %
polynomial	$(\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$	$\varepsilon = 0.001,\ C = 3.379$	2.58 %
polynomial	$(a \cdot \mathbf{x}_i \cdot \mathbf{x}_j + c)^d$	$\varepsilon = 0.001,\ C = 0.496,$ $a = 0.027,\ c = 9.482,\ d = 2.961$	2.99 %
linear	$\mathbf{x}_i \cdot \mathbf{x}_j$	$\varepsilon = 0.001,\ C = 3.379$	3.69 %

The best results were achieved with RBF kernel. Parametric models with linear or polynomial kernel function produced slightly worse results. Since their feature space has a finite number of dimensions (unlike the RBF feature space), it was harder and harder to incorporate new vectors into model with the rising number of trained data, i.e. the addition of a new vector required more and more migrations of model vectors. The disadvantage of polynomial kernel function is also a higher number of parameters. We also experimented with kernels similar to RBF that are used in nonparametric models, such as sigmoid, logistic, tricube, Epanenchnikov, but the results were very poor.

4.4 One-hour forecast

In this experiment, we let one online SVR forecast for periods of one month and half-year. We made a half-hourly forecast on the original data and an hourly forecast on data obtained by aggregation of half-hours to hours. The parameters of the SVR model (ε, C and γ) were estimated by running PSO algorithm with the swarm size of 35 and the maximum number of 35 iterations on three days preceding the test set. The SVR model made the forecast for the next horizon based on measurements from the last day (i.e. the last 48 or 24 measurements were used as a new vector for the SVR). The number of vectors in the model was unlimited. MAPE values for both test sets and forecasting horizons are in the Table 4.

Table 4. MAPE of online SVR model half-hourly and hourly forecasts on two datasets.

forecasting horizon	30 min.		1 hour	
test set	1 month	6 months	1 month	6 months
MAPE	2.26 %	2.16 %	2.84 %	2.55 %

We can observe that MAPE for the longer test set is lower. When we examined the behavior of MAPE over time, we observed that MAPE was decreasing with every arrival of a new sample (i.e. with every new sample incorporated in the model, see Fig. 3). The phenomenon can also be seen in Table 2 and in difference between MAPE for 30min. and 1 hour forecasts (30min. dataset contained double number of samples than 1h. dataset).

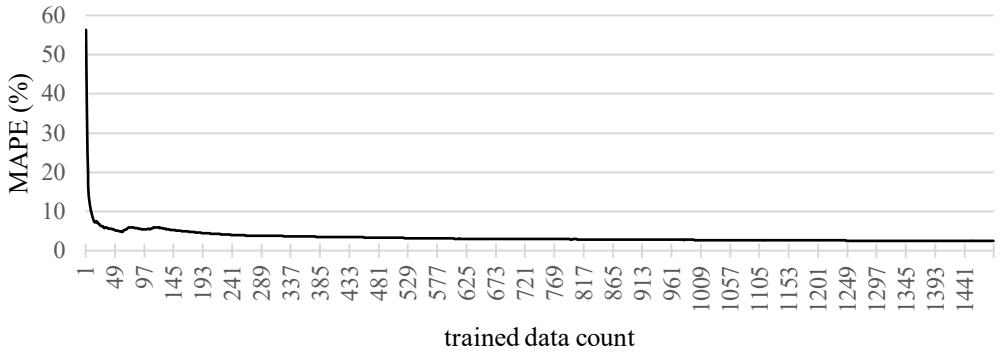

Fig. 3. MAPE of online SVR model during 1 month of forecasting.

We compared the results to the recent work on short term power load forecasting where authors used the same 6-month test set (Wijaya et al., 2015). Their cluster-cased aggregated forecasting (CBAF) approach firstly separates the consumers into clusters, the hourly consumption of each cluster is forecasted separately and in the end the results were aggregated to obtain the final forecast. They also used the SVR model (among other worse performing methods). Its parameters were estimated by selection from tested sets of values. Their SVR model's MAPE value for one cluster, i.e. aggregated

load of all consumers, was about 2.65 % and with the rising number of clusters they managed to achieve MAPE of about 2.45 % for 10 clusters. Our result by online SVR (2.55 %) was very similar. The slightly lower error may be caused by the usage of PSO algorithm to estimate the SVR parameters instead of their manual tuning.

4.5 One-day forecast

In this experiment, we compare MAPE of online SVR to 10 state-of-the-art methods (see section 2.1). We let the methods forecast the next 48 half-hours iteratively (incrementally) from a sliding window. At each "slide" of the window the forecasting models were trained from scratch on data from the new window. We determined the size of the window experimentally for each method by training them on sliding windows of sizes 3-14 on the 6-month test set. To achieve lower error, we also tried the combinations of separate workday/weekday models for selected methods. In total, we compared 17 configurations of the state-of-the-art methods to the online SVR.

To overcome the drawback of SVR, i.e. the ability to forecast only one step ahead, we used the direct strategy and forecasted simultaneously with 48 online SVR models every half-hour of the next day. We divided the dataset into 48 pieces and each model forecasted the next half-hourly load based on the last 48 values of the respective half-hour (i.e. the load during that time of the day from the last 48 days). Parameters of each SVR model (ε, C and γ) were estimated by running PSO with the swarm size of 35 and the maximum number of 35 iterations on 144 values preceding the respective half-hourly test set.

MAPE of the best configuration of each method is shown in Table 5.

Table 5. MAPE of various state-of-the-art methods trained incrementally using sliding window; and online SVR on 6-month test set.

	window (days)	*MAPE*
RF-week	3	4.12 %
BAGG	14	4.60 %
online SVR	**1**	**4.72 %**
XRT-week	3	4.72 %
BAGG-week	8	4.79 %
DSHW	14	5.02 %
STL+ARIMA-week	6	5.10 %
XGB-week	10	5.24 %
XGB	14	5.25 %
XRT	14	5.27 %
STL+ES-week	7	5.29 %
RF	8	5.47 %
SVR-week	9	5.52 %
SVR	14	5.63 %
DLnet	14	5.97 %
MLP	14	6.56 %
STL+ARIMA	8	7.33 %
STL+ES	12	9.24 %

suffix "-week" means that separate workday/weekday models were used.

Online SVR had the third lowest MAPE. The results show that the best performing methods were the tree-based ensembles (RF, BAGG, XRT) with hundreds to thousands of trees in the ensemble. The

second best performing group of models (after online SVR) were the time series analysis-based models that consider the double seasonality of power load time series (DSHW, STL+ARIMA-week, STL+ES-week). The poor performance of standard SVR might be caused by bad parameter selection and by the low number of samples available in the window. We could see that the accuracy of SVR increases with the number of trained examples. While the online SVR was trained on all samples (though it did not need to keep them in memory because of its online nature of data processing), the standard SVR model was always trained on window-limited amount of data. The neural networks approaches have many parameters. It is hard to design a suitable architecture of the network and train it so it produces accurate results. The worst performing were the time series analysis methods that did not take double seasonality of the power load series into account. Although the time series was decomposed, one seasonality was probably not properly captured. The "-week" variants of all methods with separate workday/weekend models eliminated weekly seasonality and provided almost in all cases better results than their variants with one model for all days of the week.

Besides the incrementally forecasting state-of-the-art methods, we compared the results of online SVR to a recent online method based on Online Sequential Extreme Learning Machine (OS-ELM) (Li et al., 2016). Li et al. firstly clustered the consumers and then forecasted the load of each cluster. The system load forecast was a sum of the clusters' forecasts. Historical load and temperature were used to construct forecasts. Average and maximum daily MAPE was evaluated on the one-month test set. The best results were achieved with 4 clusters of consumers: 2.47 % and 4.21 %. Fig. 4 (left) depicts average daily MAPE of online SVR on the one-month test set. Online SVR's results were 3.00 % and 4.01 %.

Though the average value was worse than OS-ELM, the maximum value was lower and it can be seen in Fig. 4 where the forecast error of online SVR is quite stable and does not increase much over time, unlike OS-ELM. We can observe the similar behavior even on daily error during one month. The error is the highest during the first days and then it lowers and stabilizes with the increasing number of trained data. The incrementally forecasting methods do not have the ability to improve their accuracy over time because they create the forecasting model each time from the scratch and do not remember anything they have learned in the past. The "cold start" of the online method can be avoided if we let the model forecast and learn from data for a certain amount of time before the actual forecasting should start.

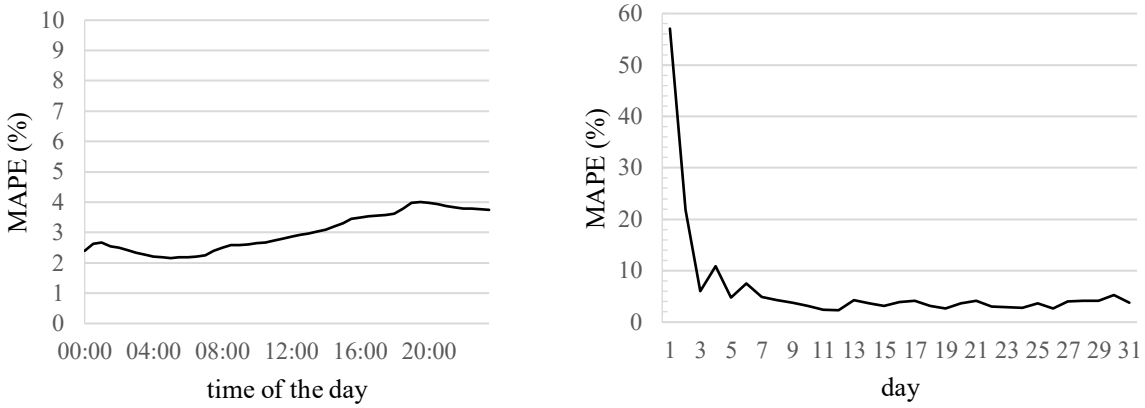

Fig. 4. Average MAPE curve during 24 hours on one-month test set (left).
Daily MAPE values during one month (right).

4.6 SVR parameter optimization

To improve the forecasts of SVR model, we focused more on the SVR parameter optimization methods. The choice of SVR parameters ε, C and parameters of kernel function (γ for RBF) is essential to achieve low forecasting error. Globally optimal parameters can be determined by a grid search, which is very exhausting operation considering time and memory, because it searches the whole parameter space. Therefore, optimization methods, such as biologically inspired algorithms, e.g., particle swarm optimization and genetic algorithm, are exploited to find optimal SVR parameters. These methods are less demanding than grid search, but do not guarantee to find global optimum. Nevertheless, it has been shown that they can produce very good SVR parameter estimates (Sreekumar et al., 2017).

In previous experiments, we exploited PSO. In this experiment, we explored another biologically inspired algorithm that was reported to outperform PSO – cuckoo optimization algorithm (COA) (Rajabioun, 2011). We adjusted the original algorithm, so the cuckoo population could dynamically change during the search – it varied from 10 to 15. This adjustment lowered the number of iterations the algorithm needs to converge to an optimum.

SVR parameters (ε, C, γ) were optimized using PSO and COA and MAPE errors of the respective SVR models were compared. We forecasted 30min. ahead power load during one-month test set (1,488 values). The experiment was performed 5 times and the average values were observed. The parameters of the optimization algorithms were selected experimentally (see Table 6).

Table 6. PSO and COA parameters.

PSO		COA	
max. number of iterations	30	max. number of iterations	4
swarm size	30	cuckoo population size	15
inertia weight	0.2	step size	2
cognitive parameter	0.5	min. number of eggs	2
social parameter	0.5	max. number of eggs	4

Average MAPE of SVR models with parameters optimized by PSO ($MAPE_{PSO}$) on one-month test set was 2.20 %. In the case of COA ($MAPE_{COA}$), it was 2.18 %. On Fig. 5 are depicted the differences between average MAPE errors during one month. The initial MAPE was lower by 0.23 % when COA was used instead of PSO. As it is shown in Fig. 4, with the increasing number of trained data the error of online SVR model lowers and stabilizes, because the model adapts over time. After adaptation, MAPE of models was similar. $MAPE_{COA}$ was lower by 0.02 %. Differences in MAPE during the one-month test set were significant by Wilcoxon signed-rank test.

We also compared the execution time of both optimization algorithms. COA algorithm is slower than PSO but it needs less iterations to converge to optimum and can be parallelized, so in the end, the times were almost identical.

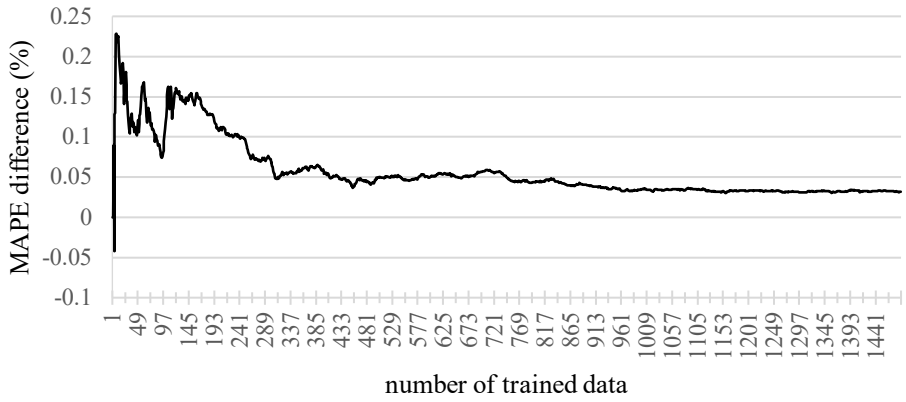

Fig. 5. Difference between SVR models with parameters optimized by PSO and COA ($MAPE_{PSO} - MAPE_{COA}$).

4.7 Computation and memory complexity

Computation complexity of online SVR training, i.e. incorporation of a new vector into the existing model, depends on the complexity of "vector addition" and "vector removal" actions (see Algorithm 1). These two actions have quadratic complexity because they both involve computation of error (9) for each vector in model to determine which of three sets they belong to. They also involve computation of coefficients $coef_1$ (15) and $coef_2$ (17), which also contain matrix multiplication. Moreover, the coefficient's calculations might repeat several times depending on how many vectors need to be migrated before a vector can be added/removed. The "vector removal" action is performed only if the number of vectors in model has exceeded $threshold$. The worst case would be if every vector in model would have to be migrated during these actions – $O(n^3)$ where n is the number of vectors in model. But that is rarely the case and the complexity is closer to $O(n^2)$.

On Fig. 6, we can see the comparison of execution times of online SVR training and traditional SVR training. We used LIBSVM library for traditional SVR that is claimed to have similar computation complexity, i.e. $O(n^2)$ – $O(n^3)$ (Chang and Lin, 2011) when RBF kernel is used. The measurements were performed on the one-month dataset, 30-min. forecast (1,488 values). In this case, the $threshold$ value was set to be 2,000, i.e. no vectors were removed from the model.

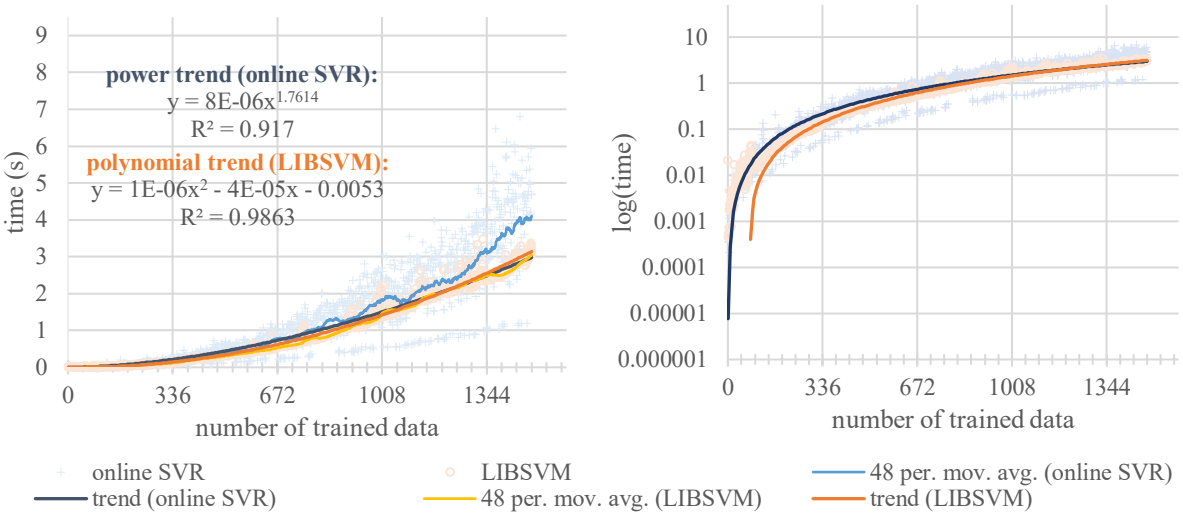

Fig. 6. Comparison of training time of online SVR model and traditional SVR model (using LIBSVM library).

Both implementations depend quadratically on number of vectors in model. We can see that online SVR also depends on number of migrated vectors in model during new vector addition. The points that form bottom boundary of the online SVR execution times represent the cases when the new vector was added to inner vectors, i.e. no vectors had to be migrated.

On Fig. 7 we can see the case when the $threshold$ value was set to be 1,000. When the $threshold$ was exceeded, the execution time increased because the "vector removal" action was also performed. On the other hand, with the constant number of vectors in model, the execution time depended only on the number of migrated vectors during "vector removal" and "vector addition" actions and oscillated around a constant trendline.

Fig. 7. Training time of online SVR and LIBSVM ($threshold = 1000$). Legend is in Fig. 6.

We also explored the time complexity dependency on the size of the vectors (see Table 7). In the previous experiments, we always used vectors of size 48 (the last day of power load measurements). The execution time is lower with the decreasing maximal number of vectors in model ($threshold$ or n) and higher with the increasing size of vectors (m). The standard deviation of execution time decreases with the decreasing number of vectors in the model, because the number of vectors (and execution time) is static after the $threshold$ is exceeded. With the increasing size of the vectors, the deviation is increasing, too. MAPE error manifested exactly opposite behavior as the execution time, i.e. the forecast is more accurate when the model contains higher number of vectors of bigger size.

Table 7. Dependency of mean online SVR training execution time (± standard deviation) and MAPE on number of vectors in model (n) and vector size (m).

	execution time			MAPE		
n \ m	48	336	1,008	48	336	1,008
2,000*	1.48 s ± 1.59 s			2.29 %		
1,000	1.48 s ± 1.46 s	2.84 s ± 2.38 s		2.33 %	2.10 %	
500	0.53 s ± 0.34 s		2.27 s ± 1.29 s	2.36 %		2.01 %

** test set contained only 1,488 values.*

Memory complexity of the method depends on the number of vectors in model, which is bounded from above by $threshold$, and the size of the vectors m. To add a new vector, the last m values of time series have to be in memory to form the new vector, as opposed to traditional SVR where the last $threshold$ number of data have to be available in memory to create SVR model.

5 Discussion and future work

The online support vector regression (SVR) is able to provide forecasts with the same accuracy as the standard SVR method. Its advantage is the online processing of data in one-by-one fashion. The method does not need to remember all historical data. It is able to learn from incoming data and to forget unused and obsolete information. Its forecast accuracy increases with the number of processed data unlike the accuracy of incrementally forecasting methods which are regularly re-trained from a sliding window. We have showed that it is suitable for short term power load forecasting.

In comparison of selected state-of-the-art methods, the tree-based ensemble methods, such as bagging, random forests or extremely randomized trees, had similar or better accuracy than online SVR. The forecasts of these methods are computed as an average forecast of hundreds or even thousands of regression trees. These methods need longer chunks of data to train. Besides the load values, additional information about time or date type is required to represent attributes in the tree nodes. Because these methods have a large set of parameters, it is inefficient to use an optimization method to tune them and a deeper knowledge and experience is needed to utilize these methods successfully. The positive feature of SVR is the small number of parameters that can be optimized by heuristics, e.g. biologically inspired algorithms. This makes SVR substantially easier to use in practice. Particle swarm optimization proved to be a good heuristic to find suitable parameters for SVR. We managed to further improve forecast accuracy by cuckoo optimization algorithm. Selection of a proper optimization heuristic and its parameters directly influences the SVR model's accuracy. Exploration of various optimization techniques for SVR parameters optimization is one of the possible directions of our future research in this area.

The disadvantage of SVR is its single output. We used direct strategy, i.e. a set of simultaneously working SVR models to forecast each (half-)hour of the next day. Each model worked only with the series of values from the respective (half-)hour. An iterative strategy or re-formulation of the SVR model to multiple-output (e.g. (Bao et al., 2014)) could result in improvement of accuracy as the model would not work with values separated day apart from each other and the continuity of the data would be preserved.

We showed that online SVR achieved similar accuracy as current advanced online methods for power load forecasting (Li et al., 2016; Wijaya et al., 2015), which employ customers' clustering and aggregate forecasts for groups of similar consumers to obtain the final forecast. Moreover, these methods utilize additional meteorological data apart from historical power load time series. Inclusion of additional features that influences power load could also lead to improvement of forecast accuracy. It is well known that power load is strongly affected by weather (particularly by temperature and humidity). The effect is noticeable especially in the areas where air conditioners are heavily used. The weather could be easily incorporated in our method as a part of the support vectors. At the time of our research, the weather data was not available. Moreover, we assumed the effect of weather would be minimal considering the temperatures in Ireland range throughout the year from 0 to 25°C.

In our SVR model, we exploited the popular radial basis kernel function, which proved to yield the most accurate power load forecasts. We explored several other kernel functions (e.g., linear, polynomial, sigmoid). A kernel function tailored for power load data could possibly lead to some improvement. Recently, a support vector regression model with an adjusted variant of RBF kernel, called sigma SVR, has been reported to produce accurate power load forecasts (Mirowski et al., 2014). The modification implemented possibility for γ parameter of RBF to have different values for each feature of a support vector.

The last studied property of online SVR algorithm was its computation and memory complexity. We found out that our implementation of the algorithm had computation complexity $O(n^2) - O(n^3)$ where n is number of vectors in SVR model. The traditional SVR implementation – LIBSVM library – manifested the same complexity. By bounding the number of vectors in model from above via $threshold$ value the training time does not increase over time and the effect of past input data is eliminated (the oldest vector is removed when $threshold$ value is exceeded). The online SVR's training time is also directly dependent on the number of vectors that migrate during the vector additions and removals from the SVR model, and on the size of the vectors. Review and further optimization of our implementation is expected to improve the online SVR's execution time.

Memory complexity depends on the maximal number of vectors in model and vectors' size. The traditional SVR method needs all the historical data to create a model, i.e. a data chunk of size of the maximal number of vectors in model. For online learning, it is sufficient for the method to keep in memory only a data chunk of the most recent values, which size equals the size of one vector. Size of the SVR vector is typically (much) smaller than number of vectors in model. One of the advantages of online SVR is the ability to directly process and forecast incoming data without the need to store it or to query a database for it.

The study of various SVR parameter optimization heuristics, design of kernel functions for power load data, utilization of (online) clustering of consumers, incorporation of additional features in support vectors (e.g. weather) and optimization of our implementation and multi-output SVR model are ideas and subjects of our future work that can lead to improvement of both online SVR method and power load forecast's accuracy.

6 Conclusion

Smart city systems can generate vast amounts of data (e.g., using IoT technologies): transport data, data about water and gas consumption, social and economic datasets, satellite data, city governance data, social media data, etc. Smart power grid is also an integral part of smart city that produce large amounts of data. They are useful especially for the electricity market stakeholders. Precise power load forecasts are essential to control power supply and balance in the grid. Short term load forecast is particularly useful for power providers who have to procure energy based on the current demand as precisely as possible and minimize the costs for energy imbalance. The accurate forecasts are important inputs for interactions on the liberated electricity market where the missing or extra energy can be traded between the stakeholders even one hour ahead.

The processing of continually arriving data from smart meters demands transition from the batch learning methods to incremental and online forecasting. The advantages of online methods are one-by-one data processing, ability to learn continually without re-training of the model from scratch and ability to forget old or obsolete information. These features make them superior when non-stationarity (e.g. concept drifts in power load caused by weather, holidays, etc.) is present in data.

The contribution of this paper is two-fold. Firstly, we presented the application of online SVR method for power load forecasting and showed its advantages (accuracy, online processing, small amount of data kept in memory) and disadvantages (single-output, cold start). We demonstrated that it is suitable for short term power load forecasting and is able to achieve accuracy of complex ensemble methods and advanced online methods. Via set of experiments we described its behavior concerning various values and forms of its parameters, inputs or kernel functions in power load forecasting scenario. Our findings can be utilized in other application areas, e.g. in other smart city systems.

Secondly, we provided a comparison of ten state-of-the-art forecasting methods on a public dataset. In the literature, we have encountered only a small number of similar reports comparing multiple power load forecasting techniques. The comparisons and extensive studies are usually made on datasets of private companies. We consider this a very useful and reusable asset for the forecasting community.

Acknowledgments

This contribution was created with the support of the Research and Development Operational Programme for the project “International Centre of Excellence for Research of Intelligent and Secure Information-Communication Technologies and Systems”, ITMS 26240120039, co-funded by the ERDF; the Scientific Grant Agency of the Slovak Republic, grant No. VG 1/0646/15 and VG 1/0752/14; and the Slovak Research and Development Agency under the contract No. APVV-16-0213.

References

Achanta, R., 2012. Long Term Electric Load Forecasting using Neural Networks and Support Vector Machines. Int. J. Comput. Sci. Technol. 3, 266–269. doi:10.1.1.228.7746

Bao, Y., Xiong, T., Hu, Z., 2014. Multi-step-ahead time series prediction using multiple-output support vector regression. Neurocomputing 129, 482–493. doi:10.1016/j.neucom.2013.09.010

Bosnić, Z., Demšar, J., Kešpret, G., Pereira Rodrigues, P., Gama, J., Kononenko, I., 2014. Enhancing data stream predictions with reliability estimators and explanation. Eng. Appl. Artif. Intell. 34, 178–192. doi:10.1016/j.engappai.2014.06.001

Boussaïd, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Inf. Sci. (Ny). 237, 82–117. doi:10.1016/j.ins.2013.02.041

Box, G.E.P., Jenkins, G.M., Reinsel, G.C., 1970. Time Series Analysis: Forecasting and Control. Holden-Day, San Francisco, CA, USA.

Breiman, L., 2001. Random Forests. Mach. Learn. 45, 5–32. doi:10.1023/A:1010933404324

Breiman, L., 1996. Bagging Predictors. Mach. Learn. 24, 123–140. doi:10.1023/A1018054314350

Cauwenberghs, G., Poggio, T., 2001. Incremental and decremental support vector machine learning. Adv. Neural Inf. Process. Syst. 409–415. doi:10.1.1.21.1720

Cleveland, R.B., Cleveland, W.S., McRae, J.E., Terpenning, I., 1990. STL: A Seasonal-Trend Decomposition Procedure Based on Loess. J. Off. Stat. 6, 3–73.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control. Signals, Syst. 2, 303–314. doi:10.1007/BF02551274

Dhillon, J., Rahman, S.A., Ahmad, S.U., Hossain, M.J., 2016. Peak electricity load forecasting using online support vector regression, in: 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE, pp. 1–4. doi:10.1109/CCECE.2016.7726784

Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., Vapnik, V., 1996. Support Vector Regression Machines, in: Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996. pp. 155–161.

Duan, P., Xie, K., Guo, T., Huang, X., 2011. Short-Term Load Forecasting for Electric Power Systems Using the PSO-SVR and FCM Clustering Techniques. Energies 4, 173–184. doi:10.3390/en4010173

Friedman, J.H., 2001. Greedy Function Approximation: A Gradient Boosting Machine. Ann. Stat. 29, 1189–1232.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Comput. Surv. 46, 1–37. doi:10.1145/2523813

Geurts, P., Ernst, D., Wehenkel, L., 2006. Extremely randomized trees. Mach. Learn. 63, 3–42. doi:10.1007/s10994-006-6226-1

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Hong, T., 2014. Energy Forecasting : Past, Present and Future. Foresight Int. J. Appl. Forecast. 43–48.

Hong, W.-C., 2013. Intelligent Energy Demand Forecasting, Lecture Notes in Energy. Springer London, London. doi:10.1007/978-1-4471-4968-2

Hu, Z., Bao, Y., Xiong, T., 2013. Electricity load forecasting using support vector regression with

memetic algorithms. Sci. World J. 2013, 10. doi:10.1155/2013/292575
Humeau, S., Wijaya, T.K., Vasirani, M., Aberer, K., 2013. Electricity load forecasting for residential customers: Exploiting aggregation and correlation between households, in: 2013 Sustainable Internet and ICT for Sustainability (SustainIT). IEEE, pp. 1–6. doi:10.1109/SustainIT.2013.6685208
Chakraborty, S., Okabe, T., 2016. Robust energy storage scheduling for imbalance reduction of strategically formed energy balancing groups. Energy 114, 405–417. doi:10.1016/j.energy.2016.07.170
Chang, C.-C., Lin, C.-J., 2011. LIBSVM. ACM Trans. Intell. Syst. Technol. 2, 1–27. doi:10.1145/1961189.1961199
Chen, B.-J., Chang, M.-W., Lin, C.-J., 2004. Load Forecasting Using Support Vector Machines: A Study on EUNITE Competition 2001. IEEE Trans. Power Syst. 19, 1821–1830. doi:10.1109/TPWRS.2004.835679
Joachims, T., 1999. Making large-scale support vector machine learning practical, in: Advances in Kernel Methods. MIT Press, pp. 169–184.
Kaitovic, I., Lukovic, S., Malek, M., 2015. Unifying Dependability of Critical Infrastructures: Electric Power System and ICT: Concepts, Figures of Merit and Taxonomy, in: 2015 IEEE 21st Pacific Rim International Symposium on Dependable Computing (PRDC). IEEE, pp. 50–59. doi:10.1109/PRDC.2015.38
Laskov, P., Gehl, C., Krüger, S., Müller, K.-R., 2006. Incremental Support Vector Learning: Analysis, Implementation and Applications. J. Mach. Learn. Res. 7, 1909–1936.
Li, W., Zhang, X., 2014. Simulation of the smart grid communications: Challenges, techniques, and future trends. Comput. Electr. Eng. 40, 270–288. doi:10.1016/j.compeleceng.2013.11.022
Li, Y., Guo, P., Li, X., 2016. Short-Term Load Forecasting Based on the Analysis of User Electricity Behavior. Algorithms 9, 80. doi:10.3390/a9040080
Liaw, A., 2015. Breiman and Cutler's Random Forests for Classification and Regression.
Ma, J., Theiler, J., Perkins, S., 2003. Accurate On-line Support Vector Regression. Neural Comput. 15, 2683–2703. doi:10.1162/089976603322385117
Mahmood, K., Ashraf Chaudhry, S., Naqvi, H., Shon, T., Farooq Ahmad, H., 2016. A lightweight message authentication scheme for Smart Grid communications in power sector. Comput. Electr. Eng. 52, 114–124. doi:10.1016/j.compeleceng.2016.02.017
Martin, M., 2002. On-Line Support Vector Machine Regression, in: Proceedings of the 13th European Conference on Machine Learning. Springer.
Medhane, D.V., Sangaiah, A.K., 2017. Search space-based multi-objective optimization evolutionary algorithm. Comput. Electr. Eng. 58, 126–143. doi:10.1016/j.compeleceng.2017.01.025
Mirowski, P., Chen, S., Ho, T.K., Yu, C.-N., 2014. Demand Forecasting in Smart Grids. Bell Labs Tech. J. 18, 135–158. doi:10.1002/bltj
Montana, G., Parrella, F., 2008. Learning to Trade with Incremental Support Vector Regression Experts, in: Corchado, E., Abraham, A., Pedrycz, W. (Eds.), Hybrid Artificial Intelligence Systems. Third International Workshop, HAIS 2008, Burgos, Spain, September 24-26, 2008. Proceedings. Springer Berlin Heidelberg, pp. 591–598. doi:10.1007/978-3-540-87656-4_73
Platt, J., 1998. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines.
Pratama, M., Lu, J., Lughofer, E., Zhang, G., Er, M.J., 2016. Incremental Learning of Concept Drift Using Evolving Type-2 Recurrent Fuzzy Neural Network. IEEE Trans. Fuzzy Syst. 1–1. doi:10.1109/TFUZZ.2016.2599855
Purnomo, H.D., Wee, H.M., 2016. Particle swarm optimisation with adaptive selection of inertia weight strategy. Int. J. Comput. Sci. Eng. 13, 38. doi:10.1504/IJCSE.2016.077731
Rajabioun, R., 2011. Cuckoo Optimization Algorithm. Appl. Soft Comput. 11, 5508–5518. doi:10.1016/j.asoc.2011.05.008
Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning internal representations by error propagation, in: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. MIT Press, pp. 318–362.
Sapankevych, N., Sankar, R., 2009. Time series prediction using support vector machines: A survey. IEEE Comput. Intell. Mag. 4, 24–38. doi:10.1109/MCI.2009.932254

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural Networks 61, 85–117. doi:10.1016/j.neunet.2014.09.003

Singh, A.K., Ibraheem, Khatoon, S., Muazzam, M., Chaturvedi, D.K., 2012. Load forecasting techniques and methodologies: A review. ICPCES 2012 - 2012 2nd Int. Conf. Power, Control Embed. Syst. doi:10.1109/ICPCES.2012.6508132

Sousa, J.C., Jorge, H.M., Neves, L.P., 2014. Short-term load forecasting based on support vector regression and load profiling. Int. J. Energy Res. 38, 350–362. doi:10.1002/er.3048

Sreekumar, S., Verma, J., A, S., Kumar, R., 2017. Comparative Analysis of Intelligently Tuned Support Vector Regression Models for Short Term Load Forecasting in Smart Grid Framework. Technol. Econ. Smart Grids Sustain. Energy 2, 1. doi:10.1007/s40866-016-0018-x

Sustainable Buildings and Climate Initiative, United Nations Environment Programme, 2016. Why buildings [WWW Document]. URL http://www.unep.org/sbci/AboutSBCI/Background.asp (accessed 1.15.17).

Taylor, J.W., 2003. Short-Term Electricity Demand Forecasting Using Double Seasonal Exponential Smoothing. J. Oper. Res. Soc. 54, 799–805. doi:10.1057/palgrave.jors.2601589

Taylor, J.W., McSharry, P.E., 2007. Short-Term Load Forecasting Methods: An Evaluation Based on European Data. IEEE Trans. Power Syst. 22, 2213–2219. doi:10.1109/TPWRS.2007.907583

Tin Kam Ho, 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20, 832–844. doi:10.1109/34.709601

Vapnik, V.N., 1995. The nature of statistical learning theory. Springer.

Wijaya, T.K., Vasirani, M., Humeau, S., Aberer, K., 2015. Cluster-based Aggregate Forecasting for Residential Electricity Demand using Smart Meter Data, in: IEEE Int. Conf. Big Data. Santa Clara, CA, USA.

Winters, P.R., 1960. Forecasting Sales by Exponentially Weighted Moving Averages. Manage. Sci. 6, 324–342. doi:10.1287/mnsc.6.3.324

Zliobaite, I., Bifet, A., Gaber, M., Gabrys, B., Gama, J., Minku, L., Musial, K., 2012. Next challenges for adaptive learning systems. ACM SIGKDD Explor. Newsl. 14, 48. doi:10.1145/2408736.2408746

Žliobaite, I., 2010. Learning under Concept Drift: an Overview. Vilnius University.