Introducing Compressed Mixture Models for Predicting Long-Lasting Brake Events

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 51-31 (2018) 840–845

Introducing Compressed Mixture Models for Predicting Long-Lasting Brake Events ⋆

Staf, Emil * **McKelvey, Tomas** **

* *Propulsion Controls & Calibration, Volvo Cars, Gothenburg, 41878 Sweden (e-mail: emil.staf@volvocars.com).*
** *Electrical Engineering, Chalmers University of Technology, Gothenburg, 41258 Sweden (e-mail: tomas.mckelvey@chalmers.se).*

Abstract: With tougher restrictions on emissions the automotive industry is in dire need of additional functionality to reduce emissions. We conduct a case study trying to predict long-lasting brake events, to support the decision-making process when the engine can beneficially be put to idle or shut down to achieve emission reduction. We introduce Compressed Mixture Models, a multivariate and mixed variate kernel density model featuring online training and complexity reduction, and use it for prediction purposes. The results show that the proposed method produces comparable prediction results as a Random Forest Classifier and outperform a Support Vector Classifier. On an urban road a prediction accuracy of 87.4 % is obtained, while a prediction accuracy of 76.4 % on a highway segment using the proposed method. Furthermore, it is possible to use a trained Compressed Mixture Model as a tool for statistical inference to study the properties of the observed realization of the underlying random variables.

Keywords: Machine learning, recursive algorithms, model complexity reduction, prediction methods, probabilistic models, information theory.

1. INTRODUCTION

When working with big data or infinite data streams it is necessary to employ recursive/online training methods. Streaming data applications exists in many fields, e.g. in media, web applications, electrical power systems etc. The automotive industry is yet another example that contains many interesting streaming data applications like, multivariate on-board diagnostics, modeling customer-/vehicle excitation spaces, and modeling drift/aging of components.

Methods for estimating the probability density of a random variable can be divided into two classes, parametric and non-parametric. A popular parametric model is the Gaussian Mixture Model (GMM), treated in e.g. McLachlan and Peel (2000). Conventional methods based on GMM have consistently produced probability density models with satisfying results. Dempster et al. (1977) introduced Expectation Maximization (EM) which is the most popular offline method for estimating the parameters of a GMM according to Kristan et al. (2011). The fact that the number of components needs to be specified in advance in the original EM algorithm makes it less attractive. Zivkovic and van der Heijden (2004) introduced a recursive EM method that starts with a large number of components, larger than the optimal number of components, and discards irrelevant components during training. Figueiredo and Jain (2002) introduced a data-driven method to estimate the number of components to be used in the EM algorithm. The quality of the resulting probability density function (pdf) is directly dependent on the initial choice of number of components. Choosing a proper number of components is especially problematic in infinitely streaming data applications. The Parzen kernel density estimators (KDE), introduced by Parzen (1962), is a non-parametric method with a probability kernel around each observation. Each observation is treated as a component of the mixture and is thus not haunted by the problem of specifying a number of components. However, the complexity of the KDE increases linearly with the number of observations. Thus using KDE to model the underlying pdf of a random variable is infeasible for large or infinite data-sets.

A probability density estimation technique not suffering from the linear increase in complexity, while not having to define the number of components beforehand is an appealing idea. Some studies have been made in this field, e.g. Declercq and Piater (2008) treat every new observation as a normal distribution with a predefined covariance and use a fidelity measure for deciding if model complexity reduction is possible by simplifying two components into one during training. Kristan et al. (2011) introduces an online Kernel Density Estimaton technique (oKDE) which maintain and updates a non-parametric model of the observed data recursively from which it is possible to calculate KDE using an online bandwidth estimation method they propose. The complexity of the KDE is maintained low due to a compression/revitalization scheme.

The method introduced in this study will be referred to as a Compressed Mixture Model (CMM) and uses a recursive learning technique inspired by oKDE. In contrast to oKDE the CMM uses prior information to mimic

⋆ This work has been jointly funded by Volvo Cars and by the research program Fordonsstrategisk Forskning och Innovation (FFI), which is gratefully acknowledged.

10.1016/j.ifacol.2018.10.115

measurement uncertainty of the observation similar to the work of Declercq and Piater (2008). The previous studies on estimating a probability density function recursively focused exclusively on handling real-valued data. Our main contribution is the introduction of an online probability density estimation technique, which by observing only a single sample at a time constructs a multivariate probability estimate, featuring complexity reduction and able to handle mixed variate data. The novelty of the CMM lies in the ability to handle mixed-variate data types, which to the authors best knowledge has not yet been studied for a recursive learning technique of a kernel density estimate. The additional types of data studied are discrete valued, but the results should be general enough to handle additional data types.

The automotive industry is relying on the development of new technologies to meet the demanding emission regulations. One such concept is Stop-In-Drive-In-Speed (SIDIS), where the engine is put into idle when the vehicle is decelerating. An even better solution from an emission side stand-point would be to shut down the engine completely when the vehicle is not accelerating. Even though this sounds appealing it is not a good idea to naively shut down the engine as soon as the vehicle is decelerating. A shut down needs to be long-lasting to be considered successful from an emission reduction point of view.

In this paper we develop a predictor of long-lasting brake events to support the decision-making process when the engine can beneficially be put to idle or shut down to achieve emission reduction. We compare the prediction accuracy of three methods namely Support Vector Machines (SVM), Random Forest (RF), and CMM introduced in this paper.

The remaining sections of the paper is structure in the following way. In Section 2 the Compressed Mixture Model is introduced alongside the supporting mathematical concepts and the recursive training technique. Next, in Section 3 the case study is presented together with prediction results for the prediction methods considered. A discussion regarding the findings in this study is given in Section 4 and finally some conclusions are stated in Section 5.

2. MODEL

In this section the construction of the Compressed Mixture Model (CMM) and the incremental learning technique is presented together with the supporting theory.

2.1 Feature Vector

In this study a feature vector $\mathbf{x}$ is an observation (or realization) of a random variable vector $\boldsymbol{\mathcal{X}} : \boldsymbol{\Omega} \to E$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}^C \\ \mathbf{x}^D \end{bmatrix} = [\mathbf{x}_1, \ldots, \mathbf{x}_d]^\mathsf{T},$$

consisting of d independent random variables, where $\mathbf{x}_j$ is the observation of the jth random variable. The random variables considered either discrete, or continuous values and either univariate or multivariate. The sample space $\boldsymbol{\Omega} = \boldsymbol{\Omega}^C \times \boldsymbol{\Omega}^D$, where $\boldsymbol{\Omega}^C$ is the samples space of all continuous random variables, and $\boldsymbol{\Omega}^D$ is the samples space of all discrete random variables. Also, $\mathbf{x}^C$ and $\mathbf{x}^D$ corresponds to the continuous and discrete elements of the feature vector respectively.

2.2 Kernels

To model the data statistically, each type of data is assigned a probability kernel.

The kernel used for a discrete data-type is a categorical distribution $x \sim \mathcal{D}(\mathbf{p})$

$$\phi(x; \mathbf{p}) = \begin{cases} p^i & \text{if } x = x_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\sum_i p^i = 1$. A binary data-type is a special case of a discrete-type, which therefore can be modeled using the categorical distribution.

The kernel used for a continuous data-type is the multivariate normal distribution, and the multivariate case $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$\phi(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^k \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}, \tag{2}$$

where $^\mathsf{T}$ is the transpose and det() the determinant.

2.3 Compressed Mixture Model

Essentially, the Compressed Mixture Model is a finite mixture model

$$f(\mathbf{x}) = f(\mathbf{x}^C, \mathbf{x}^D) = \sum_{i=1}^{N} \lambda_i \Phi(\mathbf{x}; \boldsymbol{\theta}_i), \tag{3}$$

where λ_i is the mixing coefficient of the ith component, $\boldsymbol{\theta}_i$ are the parameters for component i, and Φ is a component defined by a kernel product of its d elements

$$\Phi(\mathbf{x}; \boldsymbol{\theta}_i) = \prod_{j=1}^{d} \phi_j(\mathbf{x}_j; \boldsymbol{\theta}_{ij}), \tag{4}$$

where $\phi_j(\mathbf{x}_j; \boldsymbol{\theta}_{ij})$ is the jth element (a probability kernel) of the ith component, $\mathbf{x}_j$ is the jth element in the feature vector and $\boldsymbol{\theta}_{ij}$ is the jth element kernel parameters of the ith component. The jth element of all the components in the CMM are of the same kernel type.

A CMM is a probability density function

$$\sum_{\mathbf{x}^D \in \boldsymbol{\Omega}^D} \left(\int_{\boldsymbol{\Omega}^C} f(\mathbf{x}^C, \mathbf{x}^D)\, d\mathbf{x}^\mathbf{C} \right) = 1. \tag{5}$$

2.4 Recursive Training

During the recursive training a linear increase in model complexity with the number of observations is avoided by aggregating similar components. The aggregated component contains updated parameters using a moment matching technique for two components $\Phi_1 = \Phi(\mathbf{x}; \boldsymbol{\theta}_1)$ and $\Phi_2 = \Phi(\mathbf{x}; \boldsymbol{\theta}_2)$

$$\Phi(\mathbf{x}; \boldsymbol{\theta}_*) = \boldsymbol{\psi}(\Phi_1, \Phi_2). \tag{6}$$

where $\boldsymbol{\psi}$ is the aggregation operator for two components. The moment matching is performed individually on the

element level for the kernels of the jth element $\phi_{1j} = \phi(\mathbf{x}; \boldsymbol{\theta}_{1j})$ and $\phi_{2j} = \phi(\mathbf{x}; \boldsymbol{\theta}_{2j})$

$$\boldsymbol{\theta}_{*j} = \psi(\phi_1, \phi_2), \tag{7}$$

where ψ is the element aggregation operator, defined for each kernel type. The weight for the aggregated component is given by $\lambda_* = \lambda_1 + \lambda_2$, where λ_1, λ_2 are the mixing weights of component Φ_1 and Φ_2 respectively.

Aggregation operators Here the aggregation operators for the different kernels are defined and the result that $\lambda_* = \lambda_1 + \lambda_2$ is frequently used. Starting with the aggregation operator for two discrete kernels $\phi_1 = \phi(x; \mathbf{p}_1)$ and $\phi_2 = \phi(x; \mathbf{p}_2)$

$$\phi(x; \mathbf{p}_*) = \psi(\phi_1, \phi_2) \tag{8}$$

where

$$p_*^i = \frac{\lambda_1 p_1^i + \lambda_2 p_2^i}{\lambda_*}.$$

For two multivariate continuous kernels $\phi_1 = \phi(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\phi_2 = \phi(\mathbf{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$

$$\phi(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) = \psi(\phi_1, \phi_2) \tag{9}$$

where

$$\boldsymbol{\mu}_* = \frac{\lambda_1 \boldsymbol{\mu}_1 + \lambda_2 \boldsymbol{\mu}_2}{\lambda_*},$$

$$\boldsymbol{\Sigma}_* = \sum_{i=1}^{2} \frac{\lambda_i (\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\mathsf{T})}{\lambda_*} - \boldsymbol{\mu}\boldsymbol{\mu}^\mathsf{T}.$$

By aggregating components into composite components the complexity of the model is reduced at the expense of model accuracy. Thus, there is a trade-off between model complexity and model accuracy. Aggregating components which are similar, with respect to some similarity measure, it is possible to maintain the information loss at an acceptable level while achieving model complexity reduction.

Component Distance The similarity between two components can be assessed by a component distance. In this study a distance based on the Kullback-Leibler (KL) divergence, a measure of how one probability distribution diverges from another, is proposed. KL divergence is additive for independent distributions, which is the case for a component defined as a product of independent kernels. Thus, it is possible to define a relevant component distance using KL divergence. The component distance between $\Phi_1 = \Phi(x; \boldsymbol{\theta}_1)$ and $\Phi_2 = \Phi(x; \boldsymbol{\theta}_2)$ is defined as

$$D(\Phi_1, \Phi_2) = \sum_{j=1}^{d} \frac{D_{KL}(\phi_{1j}, \phi_{2j}) + D_{KL}(\phi_{2j}, \phi_{1j})}{2d}. \tag{10}$$

Here $D_{KL}(\phi_{1j}, \phi_{2j})$ is the KL divergence between the two distributions ϕ_{1j} and ϕ_{2j} from the jth element of the two components Φ_1 and Φ_2 respectively, and d is the number of component elements. KL divergence is non-symmetrical, but the component distance in (10) is symmetrical and also invariant to the number of component elements. There exists closed form solutions of the KL divergence for all the studied kernels in this paper.

The KL divergence between two discrete kernels $\phi_1 = \phi(x; \mathbf{p}_1)$ and $\phi_2 = \phi(x; \mathbf{p}_2)$ is,

$$D_{KL}(\phi_1, \phi_2) = \sum_{i=1}^{k} p_1^i \log\left(\frac{p_1^i}{p_2^i}\right), \tag{11}$$

where k are the number of possible outcomes for the two discrete distributions. In the case when any $p_j^i = 0$ the component distance in (10) becomes infinite. To address this problem, and allow compression, the KL divergence for a discrete component element is forced to 0 during training.

The KL divergence between two multivariate normal distributions $\phi_1 = \phi(\mathbf{x}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\phi_2 = \phi(\mathbf{x}; \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ is given by

$$D(\phi_1, \phi_2) = \frac{\operatorname{tr}\left(\boldsymbol{\Sigma_2}^{\mathbf{1}} \boldsymbol{\Sigma_1}\right) - k + \ln\left(\frac{\det(\boldsymbol{\Sigma}_2)}{\det(\boldsymbol{\Sigma}_1)}\right)}{2} + \\ + \frac{(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\mathsf{T} \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)}{2}, \tag{12}$$

where tr() is the trace of a matrix, k is the number of dimensions of the multivariate normal distributions, and det() is the determinant of a matrix. In the univariate case the calculations can be performed without using matrix operations, which is also true for the aggregation method calculation in (9).

Pseudo-Code The recursive training process of adding a new observation is stated in Algorithm 1.

Algorithm 1. Recursive training technique for CMM.
Input: M_{n-1} - CMM, $\mathbf{x}^n$ - nth observation.

1: $\Phi = \Phi(\mathbf{x}; \boldsymbol{\theta}(\mathbf{x}^n))$, $\lambda = \frac{1}{n}$, $M = M_{n-1}$ ▷ intialize
2: $\lambda_i \leftarrow (1 - \frac{1}{n})\lambda_i \, \forall \, i$ s.t. $\Phi_i \in M$ ▷ rescale weights
3: Find $\Phi^* \in M$ s.t. $D(\Phi, \Phi^*) \leq D(\Phi, \Phi_j) \, \forall \, \Phi_j \in M$.
4: **if** $D(\Phi, \Phi^*) < \epsilon$ **then** ▷ model complexity reduction
$M \leftarrow M \setminus \Phi^*$
$\hat{\Phi} = \boldsymbol{\psi}(\Phi, \Phi^*)$ ▷ aggregate components
$\hat{\lambda} = \lambda + \lambda^*$ ▷ aggregate component weights
Let $\lambda \leftarrow \hat{\lambda}$ and $\Phi \leftarrow \hat{\Phi}$ and return to step 3.
5: **else**
$M \leftarrow M \cup \Phi$ ▷ add Φ with weight λ to model
return M_n

Below, the recursive training process is described in words.

The inputs to the recursive training technique are the currently trained model M_{n-1} and the new observation (feature vector) $\mathbf{x}^n$. Here, M_{n-1} is the CMM trained on the sequence of $n-1$ observations $\{\mathbf{x}^1, \ldots, \mathbf{x}^{n-1}\}$. In step 1, a component $\Phi = \Phi(\mathbf{x}, \boldsymbol{\theta}(\mathbf{x}^n))$ is initialized, as defined in (4). The kernel parameters $\boldsymbol{\theta}(\mathbf{x}^n)$ are given by, $\mu = x_*^n$ and predefined σ^2, $\boldsymbol{\mu} = \mathbf{x}_*^n$ and predefined $\boldsymbol{\Sigma}$, and $p^i = 1$ (if $x_*^n = x_i$) if the $*$th component element is univariate continuous, multivariate continuous, and discrete respectively. The mixing weight corresponding to the new component is set to $\lambda = \frac{1}{n}$. Also, $M = M_{n-1}$ to simplify notation. In step 2, the mixing weights for all components in M are downscaled with the factor $1 - \frac{1}{n}$. This is to assert that the sum of all mixing weights in M plus the new components mixing weight should sum to 1. In step 3, the closest component to Φ in M is found, referred to as Φ^*, using the component distance defined in (10). Step 4, model complexity reduction, is performed if the component distance between Φ and Φ^* is smaller than the maximum allowed distance threshold ϵ. The distance threshold is a hyperparameter of the CMM and asserts that model complexity reduction can be performed while

limiting the information loss. The closest component Φ^* is removed from the model M and the two components Φ and Φ^* are aggregated resulting in $\hat{\Phi}$ with mixing weight $\hat{\lambda}$. The aggregation is performed using the aggregation operator for two components defined in (6). To recursively find new complexity reductions made possible, let $\lambda \leftarrow \hat{\lambda}$ and $\Phi \leftarrow \hat{\lambda}$ and return to step 3. If all component distances are larger than or equal to ϵ, step 5 is performed and Φ with mixing weight λ is added to M. At last, the updated model M is returned.

We argue that the worst case time complexity is $O(n^2)$ for Algorithm 1, when applied incrementally on n observations. The two steps, step 2 and step 3, are linear with respect to the number of components $n_c \leq n$. The other steps are constant with respect to n_c and thus also n. The worst imaginable scenario is if no model complexity reduction is allowed until the last observation when one model complexity reduction starts a chain reaction of reductions until only one component remains. This corresponds to

$$\begin{aligned} T(n) &= \underbrace{\sum_{i=1}^{n_c} O(i)}_{\text{training}} + \underbrace{\sum_{i=0}^{n_c-1} O(n_c - i)}_{\text{reduction}} \leq \\ &\leq \sum_{i=1}^{n} O(i) + \sum_{i=0}^{n-1} O(n-i) = O(n^2), \end{aligned} \tag{13}$$

and a worst case time complexity of $O(n^2)$ is obtained.

3. APPLICATION & RESULTS

As mentioned in the introduction it is of interest to predict long-lasting brake events for supporting decision-making when to perform SIDIS. In this study, for each unique road segment, we create a prediction model, using static geospatial information together with vehicle state data. The original data contained geospatial information that together with a map matching tool and OpenStreetMap (OSM) data over the Gothenburg region was used to identify the road ID and lane direction from where the data were generated. Two road segments was selected for the case study: R_1 - an urban road segment (Fabriksgatan) and R_2 - a highway segment (E6).

3.1 Data

The available data have been collected from real-life driving using a few test vehicles. The data consists of e.g. geospatial positions (longitude λ, latitude ϕ), vehicle speed v and the actual power output on the drive shaft p.

The vehicles uses a regeneration scheme and the brake events had to be deducted using the actual power output on the drive shaft p, according to

$$b_i = \begin{cases} True & \text{if } p < -10 \text{ kW}, \\ b_{i-1} & \text{if } -10 \text{ kW} \leq p \leq 5 \text{ kW}, \\ False & \text{if } p > 5 \text{ kW}, \end{cases}$$

where b_i is the brake event value at time instance i. A a dead zone is implemented to enhance the driving experience.

We define the time until the vehicle is accelerating again by t_i, where $t_i = 0$ if the brake event value b_i is $False$ otherwise t_i holds the time until the next brake event value that is $True$.

Finally a signal indicating if the brake is long-lasting or not is defined as

$$y_i = \begin{cases} True & \text{if } t_i > 4 \text{ s}, \\ False & \text{otherwise}. \end{cases}$$

The feature vectors $\mathbf{X}$ and labels $\mathbf{Y}$ are given by

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{i_{B_1}} \\ \vdots \\ \mathbf{x}_{i_{B_m}} \end{bmatrix} = \begin{bmatrix} v_{i_{B_1}} & p_{i_{B_1}} & \lambda_{i_{B_1}} & \phi_{i_{B_1}} \\ \vdots & \vdots & \vdots & \vdots \\ v_{i_{B_m}} & p_{i_{B_m}} & \lambda_{i_{B_m}} & \phi_{i_{B_m}} \end{bmatrix}, \mathbf{Y} = \begin{bmatrix} y_{i_{B_1}} \\ \vdots \\ y_{i_{B_m}} \end{bmatrix}.$$

In the remainder of this section the data will be split into two parts, training- and testing data. Using a split factor of 0.8 makes the training set contain 80% and the test set contain 20% of the original data set. The training data set will be used to train the model and choose suitable hyperparameters using the K-fold cross-validation technique.

3.2 Set-Up of Methods

The machine learning methods considered in this case study are Compressed Mixture Model (CMM), Support Vector Classifier (SVC), and Random Forest (RF). Python is the programming language of choice and the scikit-learn methods for Random Forest and SVC are used. The default options for the methods in scikit-learn is used, except for the hyperparameters varied which are presented in Section 3.4. A non-linear kernel (radial-basis function) is used for SVC. The CMM is defined using four univariate continuous kernels to model, v, p, λ and ϕ and a binary kernel to model the label y.

3.3 Predictions

Using a CMM, predictions can be made using the conditional probabilities, i.e. a likelihood-ratio test, of a new feature vector $\mathbf{x}$ and a label y as follows

$$g(\mathbf{x}, y) = \frac{f(\mathbf{x}|y = True)}{f(\mathbf{x}|y = False)} \lessgtr \xi, \tag{14}$$

where $y = 1$ is predicted to be $True$ if $g(\mathbf{x}, y) > \xi$, and $False$ otherwise.

3.4 Optimizing Hyperparameters

To find the optimal choice for hyperparameters the models are cross-validated on the training data set using $K = 5$ folds. The different combinations of hyperparameters are evaluated using the Area Under the Curve (AUC) metric, i.e. the area under the Receiver Operating Characteristics (ROC) curve. In this study a full-factorial optimization routine together with the K-fold cross-validation scheme is used. The resulting choices of hyperparameters for each machine learning method evaluated, maximizing AUC, on each of the two road segments analyzed R_1 and R_2 are presented in Table 1.

In Table 1, ϵ is the maximum component distance threshold, C is the penalty parameter, γ is the kernel coefficient, n_{est} is the number of estimators, and n_{feat} is the maximum

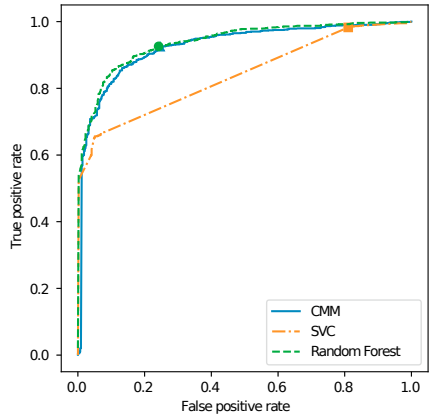

Fig. 1. ROC-curves for the different methods on R_1 test data. The points where the optimal F_1-score occurs are presented using different markers, a triangle for CMM, a square for SVC, and a circle for Random Forest.

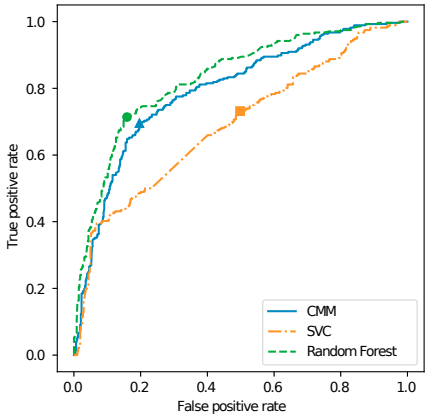

Fig. 2. ROC-curves for the different methods on R_2 test data. The points where the optimal F_1-score occurs are presented using different markers: a triangle for CMM, a square for SVC, and a circle for Random Forest.

number of features to consider when searching for the best split. Prior information used for the CMM kernel parameters are found in Table 2.

3.5 Test Results

Using the optimal set of hyperparameters in Table 1 together with the priors in Table 2 the three models are compared on the test data.

In Figure 1 and Figure 2 the *ROC*-curves for the different models on the test data set are presented for R_1 and R_2 respectively. Table 3 holds the *AUC*-metrics corresponding to the *ROC*-curves in the figures. The two methods CMM and Random Forest are producing fairly similar *ROC*-curves, while SVC deviates significantly from the others in the middle region of the *ROC*-curve. The methods tend to predict long-lasting brake events better on the urban road segment R_1 than on the highway road segment R_2. There might be several explanations why the methods have a higher prediction accuracy on the urban road segment than on the highway road segment, but the influence of static objects like e.g. crossings, traffic lights etc. could be one of the key explanations.

Confusion matrices corresponding to the marked points in Figure 1 and Figure 2 are presented in Table 4, and Table 5 respectively.

Table 1. Hyperparameters

Road Segment	CMM	SVC		RF	
	ϵ	C	γ	n_{est}	n_{feat}
R_1	1e-1	1e-3	1e2	150	2
R_2	1e-1	1e3	1e-1	150	1

Table 2. CMM priors

σ_v	σ_p	σ_λ, σ_ϕ
3	5e3	2e-5

Table 3. *AUC*-scores

Road	CMM	SVC	RF
R_1	0.926	0.839	0.940
R_2	0.794	0.690	0.830

Table 4. Confusion matrices for R_1

	CMM		SVC		RF	
	AP	AN	AP	AN	AP	AN
PP	960	78	1020	18	961	77
PN	108	330	355	83	106	332

Table 5. Confusion matrices for R_2

	CMM		SVC		RF	
	AP	AN	AP	AN	AP	AN
PP	192	84	202	74	197	79
PN	95	389	241	243	77	407

The abbreviations AP, AN, PP, and PN stands for actual positive, actual negative, predicted positive, and predicted negative respectively. A confusion matrix contains information about prediction performance. The two possible faults are a false alarm and a false rejection, which are varying in importance between applications. The F_1-score weight false alarms and false rejections equally and is used in this case study. The confusion matrices are calculated for the points in the *ROC*-curves which maximized the F_1-score, i.e. the points marked in Figure 1 and Figure 2 for the respective methods. The prediction accuracy scores in these points are presented in Table 6.

Table 6. Prediction accuracy scores

Road	CMM	SVC	RF
R_1	0.874	0.747	0.876
R_2	0.764	0.586	0.795

4. DISCUSSION

According to the results presented in Section 3 the proposed technique CMM produce competitive results in comparison with two state-of-the-art machine learning classifiers, Random Forest, and SVC. CMM tends to outperform SVC, while producing comparable results as Random Forest. This statement becomes evident by examining the *ROC*-curves in Figure 1, and Figure 2, since a better performance is obtained for a smaller false positive rate while the true positive rate is high. Thus, curves which are closer to the top-left corner have a better performance in general. The proposed technique is generative, which grants the possibility to study properties of the underlying/modeled random variable post-mortem training. The generative property of a CMM is especially interesting in situations when working with big data or infinite data streams. Time complexities for training the treated machine learning algorithms with respect to the number of observations n are:

- SVC - at least $O(n^2)$ for smaller values of C and $O(n^3)$ when C gets large, according to Bottou and Lin (2007).
- Random Forest - $O(pn^2 \log n)$ in the worst case and $\Theta(pn \log^2 n)$ on average, according to Louppe (2014), where p is the number of elements in the feature vector.
- CMM - is $O(n^2)$ in the worst case according to (13).

Thus, CMM is not more efficient in time complexity than the other methods if all data is available in advance. The recursive training technique utilized by CMM is an advantage, since it enables the possibility to update the model as new data becomes available. The methods Random Forest, and SVC requires training data being available in advance and thus will have to retrain on all data as new data is observed. There have been studies on extending these techniques to handle streaming data. An extension to the Random Forest algorithm, Streaming Random Forest, is presented by Abdulsalam et al. (2007). They show that the extended algorithm has comparable accuracy to the original method. Rai et al. (2009) presents StreamSVM, a $\frac{3}{2}$-approximation to the optimal solution of the minimum enclosing ball problem. They provide experimental evidence that StreamSVM is competitive with alternative techniques with a much simpler solution.

5. CONCLUSIONS

We have proposed a multivariate and mixed-variate kernel density estimation technique suitable for streaming data situations. The proposed technique utilizes a complexity reduction scheme, aggregating similar kernels, and maintains a compressed model of the underlying probability density distribution. The case study shows that the proposed method produces comparable prediction results as Random Forest and higher accuracy than Support Vector Classifier. The recursive training technique along-side the generative property promotes CMM as a solid candidate for online streaming applications. A trained model can be used as a predictor, as shown in the conducted case-study. Furthermore, it is possible to use a trained model as a tool for statistical inference to study the properties of the observed realization of the underlying random variables.

ACKNOWLEDGEMENTS

Special thanks to Krister Johansson (Volvo Cars) for his supervision and initial version of the Compressed Mixture Model.

REFERENCES

Abdulsalam, H., Skillicorn, D.B., and Martin, P. (2007). Streaming random forests. In *Proceedings of the 11th International Database Engineering and Applications Symposium*, 225–232. IEEE Computer Society, Washington, DC, USA.

Bottou, L. and Lin, C.J. (2007). Support vector machine solvers. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (eds.), *Large Scale Kernel Machines*, 301–320. MIT Press, Cambridge, MA.

Declercq, A. and Piater, J.H. (2008). Online learning of gaussian mixture models - a two-level approach. In *3rd International Conference on Computer Vision Theory and Applications (VISAPP)*, 605–611.

Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), 1–38.

Figueiredo, M.A.T. and Jain, A.K. (2002). Unsupervised learning of finite mixture models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(3), 381–396.

Kristan, M., Leonardis, A., and Skoca, D. (2011). Multivariate online kernel density estimation with gaussian kernels. *Pattern Recognition*, 44(10-11), 2630–2642.

Louppe, G. (2014). Understanding Random Forests: From Theory to Practice. *ArXiv e-prints*.

McLachlan, G.J. and Peel, D. (2000). *Finite mixture models*. Wiley Series in Probability and Statistics, New York.

Parzen, E. (1962). On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3), pp. 1065–1076.

Rai, P., Daumé, H., and Venkatasubramanian, S. (2009). Streamed learning: One-pass svms. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, 1211–1216. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Zivkovic, Z. and van der Heijden, F. (2004). Recursive unsupervised learning of finite mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5), 651–656.