Image and Vision Computing 30 (2012) 263–278

Contents lists available at SciVerse ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Editors Choice Article

I²VM: Incremental import vector machines ☆

Ribana Roscher *, Wolfgang Förstner, Björn Waske

Institute of Geodesy and Geoinformation, University of Bonn, Nußallee 15, 53115 Bonn, Germany

ARTICLE INFO

Article history:
Received 16 January 2012
Received in revised form 11 April 2012
Accepted 20 April 2012

Keywords:
Import vector machines
Incremental learning
Concept-drifts

ABSTRACT

We introduce an innovative incremental learner called incremental import vector machines (I²VM). The kernel-based discriminative approach is able to deal with complex data distributions. Additionally, the learner is sparse for an efficient training and testing and has a probabilistic output. We particularly investigate the reconstructive component of import vector machines, in order to use it for robust incremental learning. By performing incremental update steps, we are able to add and remove data samples, as well as update the current set of model parameters for incremental learning. By using various standard benchmarks, we demonstrate how I²VM is competitive or superior to other incremental methods. It is also shown that our approach is capable of managing concept-drifts in the data distributions.

1. Introduction

Incremental learning methods have obtained large interest in areas in which a sequential data treatment is preferred, e.g. in tracking applications or time series analysis. Compared to batch learning methods in which data is available and simultaneously processable, incremental methods have to deal with sequentially incoming data, of which order and number can be arbitrary.

A powerful incremental learning should have the following requirements:

(a) It should perform comparably to its batch learning counterpart. When applied to the same data samples, it should be independent from the sequence of the data. Thus, it should be able to handle real or pseudo-concept-drifts in data distribution without suffering a loss in performance.
(b) It should separate the classes well, regardless of the complexity of data distribution.
(c) It should be able to deal with arbitrarily long data streams, e.g. in order to treat life-long learning tasks.
(d) It should provide reliable posterior probabilities in order to allow for a meaningful evaluation, to serve as input for further processing steps, as e.g. graphical models, or to provide criteria to decide on irrelevant data to keep the model sparse.

To meet requirement (a) and (b), incremental methods in general need a reconstructive and should have a discriminative model component. A reconstructive model component represents the significant sub-domain of the data distribution. It offers robustness against the sequence of data samples, as well as being effective in the case of many competing classes. Additionally, it is also capable of adapting to actual or apparent changes in the data distribution, which appear as concept-drifts to the learner, caused by either changes within the class (intra-class variability) or by changes between the classes (inter-class variability). A discriminative model component though not necessary, is recommendable in order to be efficient for distinguishing similar classes. Moreover, kernel-based learning methods have shown good results, when dealing with complex data distributions.

To meet requirement (c) an incremental learner should be able to add and remove data samples while at the same time adapting the classifier model to the current conditions. In contrary to kernel-based batch methods, incremental kernel-based approaches demand strategies for adding and removing data samples. However, the criteria for adding and removing may differ depending on the application. When dealing with long data streams, a long-term memory may be necessary; this requires a memory-efficient classifier model. Among diverse classifiers, sparse kernel-based batch learning methods have been shown considerable success for achieving good performance with highly complex data distribution. At the same time they are sparse and therefore efficient. An incremental learner should fulfill these properties.

To meet requirement (d) the incremental learner should be based on a probabilistic classifier model. While probabilistic models tend to require more computational resources than non-probabilistic models, probabilities are often of interest and serve as input for further processing steps.

Our contribution is the formulation of I²VM, an incremental realization of import vector machines (IVM) [88]. IVM provide a *kernel-based*, *probabilistic* and *discriminative* learner, and inherently contain a *reconstructive* component. Our contribution is based on the analysis of the learning scheme and the yet unexplored reconstructive component of

☆ This paper has been recommended for acceptance by Konrad Schindler. Editor's Choice Articles are invited and handled by a select rotating 12 member Editorial Board committee.

* Corresponding author. Tel.: +49 228732713.
E-mail addresses: rroscher@uni-bonn.de (R. Roscher), wfoerstn@uni-bonn.de (W. Förstner), bwaske@uni-bonn.de (B. Waske).

0262-8856/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2012.04.004

IVM: though being a discriminative model, it implicitly samples important parts of the underlying distribution. This property makes IVM an ideal basis for an incremental learner, conceptually being able to handle heavy concept-drifts. Our incremental learning scheme deals with adding and removing data samples and the update of the current set of model parameters.

The paper is organized as follows: In Section 2 we review several classifier categories and arrange the IVM and their incremental version into them. Section 3 gives an intuitive derivation of the IVM, which is based on the classical logistic regression model and especially describes the basic learning scheme of the batch IVM. In Section 4 we analyze the reconstructive component of the IVM to show that the discriminative model can also represent important parts of the distribution. Section 5 presents our incremental learning strategy. In Section 6 we finally demonstrate the power of I^2VM using benchmark data sets and a remote sensing application.

2. Literature review

In contrast to batch/off-line models, which have access to all training data, incremental methods train with data becoming sequentially available over time. In general, these methods have some advantages to off-line methods. If there are only a few labeled samples known beforehand, we can start learning immediately and do not have to wait for further samples. This is often undesirable or sometimes simply impossible, e.g. in time series analysis or life-long learning tasks.

Several incremental learning methods have been suggested, extending classical and state-of-the art off-line methods. In this section we discuss the most common incremental classifiers meeting the desired characteristics mentioned in Section 1.

2.1. Performance

Incremental learning methods in general show comparable performances to their batch version when there is no change in the data distribution over time. The most recent methods vary from on-line discriminative kernel density estimation [48], on-line random forests (ORF) [67], incremental support vector machines [87,40,4,28,10], on-line boosting algorithms [32] to on-line nearest neighbor classifier [35]. These methods are able to handle complex distributions, in contrast to incremental linear subspace learning with linear discriminant analysis and principal component analysis [78,43]. To deal with more complex data, some incremental subspace methods have been extended to a kernel version, as e.g. in [15].

Concept-drifts, i.e. changes of the distribution over time, may cause severe problems for incremental learners. Like in tracking applications, old examples are often misleading, which is why newly gathered training vectors should be more beneficial than older ones. The incremental learner should be able to adapt to changes in the underlying distribution of data samples, as in [35,74,33].

Explicitly identifying and handling concept-drifts has been considered in [69,45] which show that incremental learner as on-line boosting and incremental support vector machines can deal well with detected drifts.

A weaker but nevertheless relevant effect may result from changing the sequence of data samples, which results in so-called pseudo-concept-drifts as shown in [82,50,18]. Such drifts can occur for example in medical experiments, if the data are collected for one patient at a time or in controlled experiments with a test plan for which process parameters are tested successively as discussed in [65]. Several approaches have been suggested to overcome this problem, e.g. by introducing a buffer to store difficult samples for a later evaluation [73] or by rearranging the sequence [54]. The authors of [1] introduce a schedule in which the samples are presented in a special arrangement to a neural network classifier to increase the performance. These methods are not useful for an incremental learner due to the requirement that samples should be put into order, and in practice one usually cannot wait until a sufficient part or even all data samples are ready.

To deal with pseudo-concept-drifts, e.g. [78] and [71] stated that in principle incremental methods need a reconstructive model component, which is an inherent property of generative approaches. They model the joint probability of the labels and the features, and in a Bayesian way use priors to derive posterior probabilities. Incremental generative models [48,35,84,22,56,78] can therefore represent current and even previously removed training samples. The approaches are very efficient, because the models can be defined by a few parameters describing the underlying distribution. This however requires the structure of the underlying distribution to be known and approximable with a sufficient accuracy. This is not the case for many real-world data sets that have a complicated distribution without known underlying structure. To overcome this problem, non-parametric density estimation techniques such as k-nearest neighbor or kernel density estimation are used [48,35]. Although they have several advantages, these models can become very complex when the data distribution is approximated and the models have to be reduced for incremental learning settings. They are able to directly incorporate unlabeled samples [51] and the model can be used as approximation of discarded samples, which is needed for robust incremental learning (see [71,78]).

2.2. Discriminative Power

In contrary to generative models, discriminative models such as logistic regression directly model the posterior probability but do not contain a reconstructive component. Also support vector machines (SVM) [79] that directly map the inputs into decisions by determining a discriminant function do not contain this component. Likewise decision trees [8] are also based on the estimation of discriminant functions. During the induction of a decision tree the training data are sequentially into smaller increasingly homogeneous subsets by using a set of decision boundaries defined at each node. Usually only the most relevant feature is used at each node, resulting in many rather simple decision boundaries that are parallel to the axis in the feature space. The accuracy of decision tree classifiers is often improved by using ensemble strategies. Classifier ensembles are based on the combination of independent variants of the same classifier, i.e. the base classifier. Various strategies have been introduced, like boosting [80,25], bagging [6] and random subspace methods [38]. The decision tree-based classifier ensemble random forest [7] is a powerful alternative that combines the two latter strategies. It has been shown for various applications that discriminative classifiers can achieve higher classification accuracies than generative models, if the underlying distribution is complex enough, although the posterior or the decision is very simple [49,79]. This transfers to incremental learners as incremental SVM or ORF which present good performances, e.g. in tracking tasks [74,67].

Logistic regression is one of the oldest discriminative models, which historically goes back to early 19th century [19]. The basic concept has been extended in many respects, such as e.g. [26] which integrate the concept of boosting to an additive logistic regression model or [21] which has incorporated a latent variable. Logistic regression is the basis for advanced models such as incremental logistic regression [2], kernel logistic regression [41,42,11,64] and sparse kernel logistic regression [12,75]. IVM constitute a realization of sparse kernel logistic regression. Among various developments of kernel discriminant classifiers, support vector machines are perhaps one of the most popular approaches in recent applications. However, [63,5,86] have shown that IVM are competitive to SVM. A comprehensive empirical comparison of common batch classifiers can be found in [9]. The positive attributes of generative and discriminative models have been integrated in hybrid generative/discriminative models. The motivation is to formulate a classifier with a reconstructive component while exploiting the discriminative power. Most of the approaches combine a generative and a

discriminative classifier [83,78,71,27], while others learn a generative model using a sequence of discriminative models [76], interpolate between generative and discriminative parameter estimation [53,51,60] or learn a sparse generative model while retaining as much discriminative power as possible [48].

2.3. Long sequences

Incremental linear subspace learning methods adapt the model parameters by processing newly available data samples, while directly discarding the samples after they have been processed. In contrast, kernel-based algorithms allow a simple adaption of the model by adding new training samples. In order to avoid a steadily increasing complexity of the model when learning over a long time, all kernel-based learners need a component for removing data. Approaches such as [40,44,28] show considerable success in incremental learning settings and in general provide update steps to remove data samples in a similar way as data samples are added. The challenge is to perform efficient incremental update steps without suffering a loss in performance.

2.4. Probabilities

Besides the predicted class labels of test data, a probabilistic output of the classifier is often of interest. The probabilities can additionally serve as confidence and can therefore be used for either a more accurate evaluation of the classifier or for a further analysis of the classified data (e.g. [29]). The probabilities can serve as input into a graphical model [63,49]. In the context of active learning, the probabilities can be used to decide on what data samples are to be added to the current training set [52,39]. The authors of [62] have shown that active learning can be performed using an incremental learner, without the need to repeatedly retrain the classifier. The probabilities are used to identify relevant samples for updating the incremental learner, and non-informative samples are removed by exploiting diagnostics tools. In contrast to kernel-based probabilistic models such as IVM and sparse multinomial kernel logistic regression [12,47,75], SVM are non-probabilistic. However, the output of SVM can be transformed to be probabilistic. The most common technique to transform SVM's output to a probability is to fit a sigmoid to the output [58]. Other methods are proposed in [34,72]. Though resulting in an output lying in the range [0,1], this does not imply that the transformed output is a good approximation of the posterior, i.e. the reliability of these values could be inadequate [66,75]. The most recent discussion about this aspect can be found in [23], in which a generative and semi-parametric probabilistic model is presented that is equivalent to linear SVM. Decision trees provide probability estimates, which arise from frequency-based calculations at the leaf nodes. Simply using the counts of classes at the leaf nodes does not give good probability estimates, as already stated in [14]. Therefore smoothing methods [55,59,85] or bagging [14] have been used to improve the reliability of the estimates, which is why random forests to generate better estimates than decision trees. In considerations about IVM we will show that by using an isotropic stationary kernel, the probabilistic model develops a reconstructive component unlike discriminative classifiers such as SVM and random forests, which inherently are not able to.

To recapitulate, import vector machines are a sparse, kernel-based, probabilistic and discriminative multi-class learner with a reconstructive component, which meet all mentioned requirements. Thus, IVM appear to be a suitable starting point for an efficient and robust incremental classifier.

3. Theoretical background

The IVM classifier is based on the classical model of logistic regression, enriched by using kernel features and made efficient by introducing sparseness. This section provides the theoretical basis in order to lay open the reconstructive properties of IVM in the next section and to motivate the extension to I^2VM.

3.1. Notation

We denote vectors $\boldsymbol{g} = [g_i] = [g_1, \ldots, g_I]^T$ with small bold symbols and matrices $G = \left[g_{ij}\right] = \left[\boldsymbol{g}_1, \ldots, \boldsymbol{g}_J\right]$ with elements g_{ij} and column vectors $\boldsymbol{g}_j$ with capital symbols. We use calligraphy symbols for sets. The elements (scalars and vectors) of a set $\mathcal{G}$ can be collected in a vector $\boldsymbol{g}$ or a matrix G by concatenation, using the same letter of the alphabet. The matrix or the vector of the concatenated elements of a subset $\mathcal{F} \in \mathcal{G}$ is denoted with $G_{\mathcal{F}}$ or $\boldsymbol{g}_{\mathcal{F}}$, respectively.

We assume to have a training set $(\boldsymbol{x}_n, y_n) \in \mathcal{T}$, $n = 1, \ldots, N$ of N labeled samples with M-dimensional feature vectors $\boldsymbol{x}_n \in \mathbb{R}^M$ and class labels $y_n \in C = \{1, \ldots, c, \ldots, C\}$. The observations are collected in the (NxM) matrix $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^{\mathrm{T}}$, while the corresponding labels are summarized in the vector $\boldsymbol{y} = [y_1, \ldots, y_N]^{\mathrm{T}}$.

In order to simplify the representation of certain matrix-operations, we use a selection operator $\Psi_{\mathcal{I}}(B)$, selecting the rows $\mathcal{I}$ from a (NxM) matrix B, defined as

$$B_{\_} = \Psi_{\mathcal{I}}(B) = \left[\boldsymbol{e}_i^N\right]_{i \in \mathcal{I}}^T B. \tag{1}$$

with $\boldsymbol{e}_i^N$ the i-th unit vector of length N. In a similar manner, the removal of columns can be formulated with the transposition of the considered matrix.

3.2. Logistic regression

The basic model of logistic regression starts from the two-class classification problem where the posterior probability $P_n(y_n = 1 | \boldsymbol{x}_n; \boldsymbol{\alpha})$ for class 1 of a feature vector $\boldsymbol{x}_n$ is assumed to follow the logistic regression model

$$P_{n1}(\boldsymbol{\alpha}) \doteq P(y_n = 1 | \boldsymbol{x}_n; \boldsymbol{\alpha}) = \frac{1}{1 + exp(-\boldsymbol{\alpha}^T \boldsymbol{x}_n)}. \tag{2}$$

The posterior probabilities for class 2 accordingly are $P_{n2} = 1 - P_{n1}$, The extended feature vector is $\boldsymbol{x}_n^T = \left[1, \boldsymbol{x}_n^T\right] \in \mathbb{R}^{M+1}$ and the extended parameters are $\boldsymbol{\alpha}^T = \left[w_0, \boldsymbol{w}^T\right] \in \mathbb{R}^{M+1}$ containing the bias w_0 and the weight vector w, which are to be learned from training data.

The model can be generalized to the multi-class case with the probabilities $P = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N]$ obtained by the softmax function

$$P_{nc}(\boldsymbol{\alpha}) \doteq P(y_n = c | \boldsymbol{x}_n; \boldsymbol{\alpha}) = \frac{exp\left(\boldsymbol{\alpha}_c^T \boldsymbol{x}_n\right)}{\sum_{c'} exp\left(\boldsymbol{\alpha}_{c'}^T \boldsymbol{x}_n\right)}. \tag{3}$$

The unknown ((M + 1)C) vector $\boldsymbol{\alpha}$ is a concatenation of all C parameter vectors $\boldsymbol{\alpha}_c$ of length $M + 1$. The number of model parameters obviously increases linearly with the number of classes. The parameters can be directly determined solving a multi-class learning task. Three synthetic examples are illustrated in Fig. 1, showing classification results with logistic regression. The posterior probabilities appear intuitive.

Because the probabilities of the classes sum to 1, the model only contains $(C-1)(M+1)$ independent parameters. This can be seen by reducing the ratio in Eq. (3), e.g. by $exp(\boldsymbol{\alpha}_1^T \boldsymbol{x})$ without changing the posteriors, leading to $P_{n1} = 1 / \left(1 + \sum_{c'=2}^{C} exp\left((\boldsymbol{\alpha}_{c'} - \boldsymbol{\alpha}_1)^T \boldsymbol{x}_n\right)\right)$. However, because of its symmetry we prefer Eq. (3) in general, but sometimes use Eq. (2) for derivations. The learning task is to estimate optimal parameters $\boldsymbol{\alpha}_c$ for each class from the training data, while taking into account that (M + 1) parameters are not identifiable.

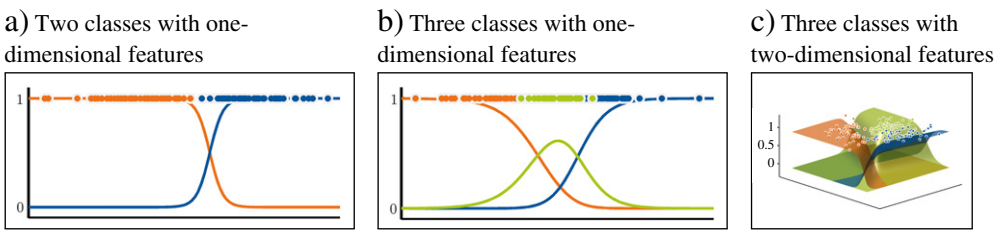


Fig. 1. Classwise posterior probabilities (vertical axis) arising from the multi-class logistic regression classifier.

The classical procedure is based on minimizing the negative log likelihood function

$$\mathcal{Q}_0(\boldsymbol{\alpha}) = -\frac{1}{N}\sum_c \sum_n t_{nc} \log P_{nc}(\boldsymbol{\alpha}) \tag{4}$$

with respect to $\boldsymbol{\alpha}$. The indicator vector $\boldsymbol{t}_n$ for a feature vector $\boldsymbol{x}_n$ belonging to class c_c is the n-th unit vector with all elements zero except element c. The function $\mathcal{Q}_0$, is the cross-entropy (up to the constant factor N) between the target probabilities t_{nc} and the estimated probabilities P_{nc}. The iteration process needs the gradient and the Hessian

$$\nabla \mathcal{Q}_0(\boldsymbol{\alpha}) = \frac{1}{N}\left[X^T(\boldsymbol{t}_c - \boldsymbol{p}_c(\boldsymbol{\alpha}))\right]_{c=1,\ldots,C} \tag{5}$$

$$\nabla \mathcal{Q}_0(\boldsymbol{\alpha}) = \frac{1}{N}\left[X^T R_{cc'}(\boldsymbol{\alpha}) X\right]_{cc'=1,\ldots,C} \tag{6}$$

with the diagonal $(M+1)\times(M+1)$ matrices

$$R_{cc'} = \sum_n \left(\mathrm{Diag}(\boldsymbol{p}_n(\boldsymbol{\alpha})) - \boldsymbol{p}_n(\boldsymbol{\alpha})\boldsymbol{p}_n^T(\boldsymbol{\alpha})\right) \tag{7}$$

depending on the parameters $\boldsymbol{\alpha}$. The (NC) vector of the total gradient $\nabla \mathcal{Q}_0(\boldsymbol{\alpha})$ is the concatenation of the per-class gradients $\nabla_{\boldsymbol{\alpha}_c} \mathcal{Q}_0(\boldsymbol{\alpha}) = \frac{1}{N} X^T(\boldsymbol{t}_c - \boldsymbol{p}_c(\boldsymbol{\alpha}_c))$. The total Hessian is a (NC×NC) matrix and consists of (C×C) blocks $\nabla^2_{\boldsymbol{\alpha}_c,\boldsymbol{\alpha}_{c'}} \mathcal{Q}_0(\boldsymbol{\alpha}) = \frac{1}{N} X^T R_{cc'} X$. The Hessian is negative semi-definite. It has a rank deficiency of $M+1$, since it can be written as

$$\frac{1}{N}\sum_n \left(R_{cc'} \otimes \boldsymbol{x}_n \boldsymbol{x}_n^T\right)$$

and the left factor of the Kronecker product has rank (C-1) and eigenvector 1_C, reflecting that (M+1) parameters are not identifiable. In the special case of the asymmetric two-class model of Eq. (2), which we will use for showing I²VM in Section 5, the gradient and the Hessian consist of only one component, referring to the parameters of class one.

In order to prevent overfitting one may introduce a prior over the parameters and optimize

$$\mathcal{Q}(\boldsymbol{\alpha}) = \mathcal{Q}_0(\boldsymbol{\alpha}) + \frac{\lambda}{2}\boldsymbol{\alpha}^T L \boldsymbol{\alpha} \tag{8}$$

with a positive definite symmetric matrix L, and a positive regularization parameter λ, usually determined by cross-validation on the training data set. The matrix L is usually assumed to be the unit matrix, thus $L = I_{((M+1)C)}$, but in order to prevent a regularization of the bias parameter $\boldsymbol{w}_{0c}$, the diagonal entries for the bias are set to zero. In order to prevent a regularization of the bias parameters w_{0c} each diagonal entry that applies to modulo(Diag$(L) = 1$) is set to zero. Thus, only the steepness of the sigmoid function, defined by $\boldsymbol{w}_c$, is regularized.

Therefore the iteration scheme could simply be formulated with the Newton–Raphson iteration method

$$\boldsymbol{\alpha}_{(t)} = \boldsymbol{\alpha}_{(t-1)} - \left(\nabla^2 \mathcal{Q}_0\left(\boldsymbol{\alpha}_{(t)}\right) + \lambda L\right)^{-1}\left(\nabla \mathcal{Q}_0\left(\boldsymbol{\alpha}_{(t)}\right) + \lambda L \boldsymbol{\alpha}_{(t-1)}\right), \tag{9}$$

where the regularization at the same time enforces regularity. The Newton–Raphson iteration procedure can be reformulated into the iterated reweighted least squares (IRLS) optimization method:

$$\boldsymbol{\alpha}_{c,(t)} = \left(\frac{1}{N} X^T R_c X + \lambda L\right)^{-1} X^T R_c z_c \tag{10}$$

with $R_c = R_{cc}$ and $L = I_{M+1}$, whereby $l_{11} = 0$, and

$$z_c = \frac{1}{N}\left(X\boldsymbol{\alpha}_{c,(t-1)} + R_c^{-1}(\boldsymbol{p}_c - \boldsymbol{t}_c)\right). \tag{11}$$

3.3. Kernel logistic regression

To extend the linear model to a non-linear one, the original features $\boldsymbol{x}_n = [x_{nm}], n = 1,\ldots,N, m = 1,\ldots,M$ are transformed into new ones $\boldsymbol{k}_n = [k_{nn'}], n, n' = 1,\ldots,N$ using a kernel function

$$k_{nn'} = k(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) \qquad \boldsymbol{x}_n, \boldsymbol{x}_{n'} \in \mathcal{T}. \tag{12}$$

To obtain a reconstructive component of the IVM, we use an isotropic stationary kernel, which is translation invariant and only depends on the norm of the directional vector between data samples [30]. Such kernels are also used in non-parametric density estimation techniques. Consequently, the (N×N) kernel matrix $K = [k_{nn'}]$ consists of affinities between the given feature points of the training set and contains the new features as rows or columns, see Fig. 2. One of these kernels is the Gaussian radial basis function kernel

$$k(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) = exp\left(-\frac{1}{2}\frac{|\boldsymbol{x}_n - \boldsymbol{x}_{n'}|^2}{\sigma^2}\right), \tag{13}$$

which we will use for all our considerations and experiments. Also other isotropic stationary kernels such as the Cauchy, Laplacian or exponential kernel could be used.

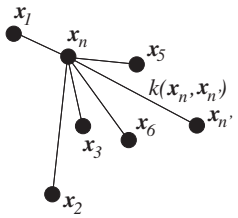


Fig. 2. Kernel features and import vectors. Instead of using the original n-th feature vector $\boldsymbol{x}_n$ one uses the affinities $\boldsymbol{k}_n = [k(\boldsymbol{x}_n, \boldsymbol{x}_{n'})]$ of the n-th data point to all other N training points n', which depends on the distance between $\boldsymbol{x}_n$ to all features $\boldsymbol{x}_{n'}$.

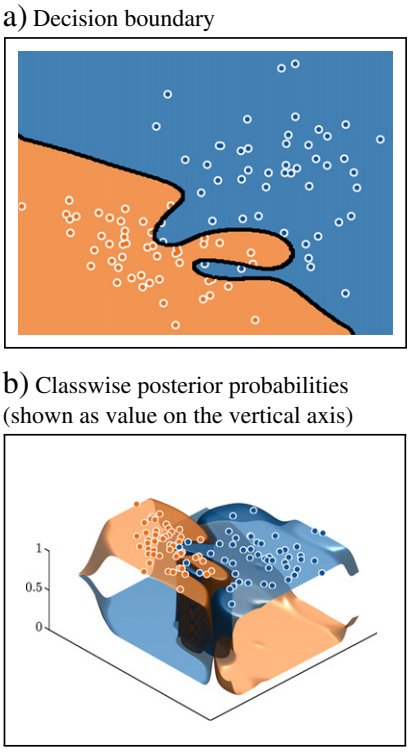


Fig. 3. Result of a two-class problem with two-dimensional features arising from the kernel logistic regression classifier.

The model of kernel logistic regression now presumes the a posteriori probabilities are given by

$$P_{nc}(\boldsymbol{\alpha}) = \frac{exp\left(\boldsymbol{\alpha}_c^T \boldsymbol{k}_n\right)}{\sum_{c'} exp\left(\boldsymbol{\alpha}_{c'}^T \boldsymbol{k}_n\right)} \tag{14}$$

with $\boldsymbol{k}_n$ as the n-th column of the kernel matrix K, the (NC) unknown model parameters $\boldsymbol{\alpha} = [\ldots;\boldsymbol{\alpha}_c;\ldots]$ referring to the C classes.

The parameters are determined in an iterative way, by optimizing the regularized objective function (8) with L=K similar to Eqs. (10) and (11)

$$\boldsymbol{\alpha}_{c,(t)} = \left(\frac{1}{N} K^T R_c K + \lambda K\right)^{-1} K^T R_c z_c \tag{15}$$

$$z_c = \frac{1}{N}\left(K\boldsymbol{\alpha}_{c,(t-1)} + R_c^{-1}(\boldsymbol{p}_c - \boldsymbol{t}_c)\right). \tag{16}$$

Fig. 3 shows a synthetic example classified with kernel logistic regression. The example illustrates that the classes are separated with a complex decision boundary.

3.4. Sparse kernel logistic regression

Standard kernel logistic regression uses all training samples to train the classifier, which is computationally excessive and memory intensive for large data sets. Recently several algorithms have been developed, which enforce sparseness in the (kernel) logistic regression model in order to control both the generalization capability of the learned classifier model and the complexity. The most common methods enforce sparseness by introducing a suitable prior [75,12,47] or by subset selection [88]. In the following we will consider the sparse kernel logistic approach, which chooses a subset $\mathcal{V}$ of V feature vectors out of the training set $\mathcal{T}$, comprising samples $X_{\mathcal{V}} = [\boldsymbol{x}_v]$, $v = 1,\ldots,V$. Thus, only affinities $\boldsymbol{k}_v$ between all samples $\mathcal{T}$ and samples in the subset $\mathcal{V}$ are collected in a kernel matrix $K_{\mathcal{V}}$, whereas all other affinities are left out of consideration.

Following Eq. (15) the parameters in iteration t are determined by

$$\boldsymbol{\alpha}_{c,(t)} = \left(\frac{1}{N} K_{\mathcal{V}}^T R_c K_{\mathcal{V}} + \lambda K_R\right)^{-1} K_{\mathcal{V}}^T R_c z_c \tag{17}$$

$$z_c = \frac{1}{N}\left(K_{\mathcal{V}}\boldsymbol{\alpha}_{c,(t-1)} + R_c^{-1}(\boldsymbol{p}_c - \boldsymbol{t}_c)\right). \tag{18}$$

The (N×V) kernel matrix is given by $K_{\mathcal{V}} = [k(\boldsymbol{x}_n, \boldsymbol{x}_v)]$ with $\boldsymbol{x}_n \in \mathcal{T}$, and the (V×V) regularization matrix by $K_R = [k(\boldsymbol{x}_v, \boldsymbol{x}_{v'})]$ with $\boldsymbol{x}_v, \boldsymbol{x}_{v'} \in \mathcal{V}$.

Fig. 4 methodically shows the matrix multiplication, which is conducted in the softmax function (3). The softmax function has to be evaluated in all the iterations of both the optimization procedure and the classification step. For kernel logistic regression the whole kernel matrix is set up and all parameters have non-zero values. When using a sparsity enforcing prior some parameters are set to zero, while the number of features remains the same. The corresponding rows of the kernel matrix therefore have no effect. In contrast to this, sparse kernel logistic regression with subset selection only uses a fraction of the rows. As a rule, in all cases only a very small percentage of the kernel matrix has to be computed, never the complete matrix.

3.5. Import vector machines

The search through all possible subsets to determine the best set is intractable. Therefore [88] proposed import vector machines (IVM), which realize the sparse kernel logistic regression by a subset selection, where the subset is determined in a greedy manner. The subset $\mathcal{V}$, the so-called set of import vectors, is chosen by successively adding data samples to the initially empty set $\mathcal{V}$ until a convergence criterion is reached.

The data samples are selected according their contribution to the solution, i.e. how much their incorporation to $\mathcal{V}$ decreases the objective function. The first import vector is selected as the data point, which yields the best one-class classification that can be interpreted as the point lying in the area of highest class-specific density. The second

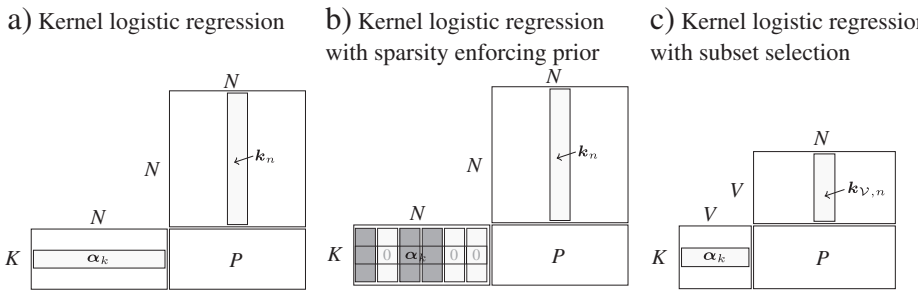


Fig. 4. Schematic figure of the matrix multiplication conducted in the softmax function for different realizations of kernel logistic regression. The boxes depict the kernel (upper right), parameter (left) and resulting posterior probability matrices (lower right), by which the sizes of each matrix are written directly to it.

import vector is chosen to yield the best linear separability, to one of the other classes. Further import vectors try to diminish the difference between the target t_{nc}, being 0 or 1, and the posterior probability P_{nc}, see Eq. (18). Fig. 5 illustrates the effect when only a subset of data samples is used. The affinities between all points will not be used, instead only a subset of it will be used.

In contrast to [88], we use a hybrid forward/backward strategy, which successively adds import vectors to the set, but also tests if import vectors can be removed in each step. Since we start with an empty import vector set and only add import vectors sequentially in the first iterations, the decision boundary can be very different from its final position. A pure forward selection is unable to remove import vectors that become obsolete after the addition of other import vectors. Therefore a removal of import points can lead to a sparser and more accurate solution than only using forward selection steps. To prevent infinite loops between forward and backward steps leading to solutions with similar objective value deselected import vectors are excluded from selection for the next few iterations. The strategy follows the idea of tabu search [31]. The forward/backward strategy is defined in Algorithm 1.

Algorithm 1. IVM in every iteration t, each sample $(\boldsymbol{x}_n, y_n) \in \mathcal{T}_{(t)}$ from the current training set $\mathcal{T}_{(t)}$ is tested to be in the set of import vectors $\mathcal{V}_{(t)}$. The point $(\boldsymbol{x}^*, y_n^*)$ yielding the lowest error $\mathcal{Q}_{(t)}^n$ is included. Import vectors are removed from $\mathcal{V}_{(t)}$ if their exclusion do not increase the optimization function $\mathcal{Q}$ plus a small value ϵ. The algorithm stops as soon as $\mathcal{Q}$ converged.

Initialize $\mathcal{V}_0 := \{\}$, $\mathcal{T} := \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, $t := 0$;
repeat
  Compute for each class c from the current set $\mathcal{V}_{(t)}$;
  foreach $(\boldsymbol{x}_n, y_n) \in \mathcal{T} \setminus \mathcal{V}_{(t)}$ **do**
    Let $\mathcal{V}_{(t)}^n := \mathcal{V}_{(t)} \cup (\boldsymbol{x}_n, y_n)$;
    For each class determine $\alpha_{c,(t)}^n$ from $\mathcal{V}_{(t)}^n$ in a one-step iteration;
    Evaluate error function $\mathcal{Q}_{(t)}^n$;
  end
  Find best point $(\boldsymbol{x}^*, y^*) = (\boldsymbol{x}_n, y_n)$ with $n = \text{argmin}_n \mathcal{Q}_{(t)}^n$, $\mathcal{Q}_{(t)} := \min \mathcal{Q}_{(t)}^n$;
  Update $\mathcal{V}_{(t)} := \mathcal{V}_{(t)} \cup (\boldsymbol{x}^*, y^*)$;
  repeat
    Compute for each class c from the current set $\mathcal{V}_{(t)}$;
    foreach $(\boldsymbol{x}_v, y_v) \in \mathcal{V}_{(t)}$ **do**
      Let $\mathcal{V}_{(t)}^v := \mathcal{V}_{(t)} \setminus (\boldsymbol{x}_n, y_n)$;
      For each class compute $\alpha_{c,(t)}^v$ from $\mathcal{V}_{(t)}^v$ in a one-step iteration;
      Evaluate error function $\mathcal{Q}_{(t)}^v$;
    end
    Find best point $(\boldsymbol{x}^*, y^*) = (\boldsymbol{x}_v, y_v)$ with $v = \text{argmin}_v \mathcal{Q}_{(t)}^v$;
    if $\min \mathcal{Q}_{(t)}^v \leq (\mathcal{Q}_{(t)} + \epsilon)$ **then**
      Update $\mathcal{V}_{(t)} := \mathcal{V}_{(t)} \setminus (\boldsymbol{x}^*, y^*)$;
    end
  until $\mathcal{V}_{(t)}$ cannot be reduced any more ;
  Update $\mathcal{V}_{(t+1)} := \mathcal{V}_{(t)}$, $t := t + 1$;
until $\mathcal{Q}$ converged ;

As [88] proposed, a convergence criterion is used by the ratio $\epsilon = |\mathcal{Q}_{(t)} - \mathcal{Q}_{(t-\Delta t)}| / |\mathcal{Q}_{(t)}|$ with a small integer Δt. Such as the regularization and kernel parameter, the threshold for excluding import vectors ϵ, this criterion influences the sparseness of the model.

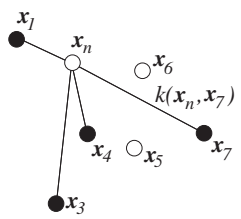


Fig. 5. Import vectors. In order to achieve a small feature vector, here the length of four, only a sparse set of training data is marked as important for the classification which is the set of the so-called import vectors, here being the set {1,3,4,7}. Thus, only a subset of the affinities between all samples are used.

Fig. 6 shows a synthetic example classified with IVM. The algorithm approximates the result of kernel logistic regression, which is underlined by a visual comparison with Fig. 3.

4. Discriminative and reconstructive properties of IVM

In this section we show, that the IVM though being a discriminative model have inherently a reconstructive component. Thus, IVM seem to be suitable for extension to an incremental classifier.

4.1. Hypothesis

Only generative classification models use an estimate for the conditional distribution of the data $P(X|\mathcal{C})$, which is the base for an efficient long-term learning of many classes. Probabilistic discriminative models, including IVM, provide an estimate of $P(\mathcal{C}|X)$. In contrast, discriminative function estimation procedures such as SVM only provide estimates for the class membership of the data samples, which can be transformed to probabilistic outputs. However, as stated in Section 2 this does not imply that the transformed output is a good approximation of the posterior. Additionally, decision trees and random forests, which are frequency-based probabilistic, do not model the conditional distribution of the data. Thus, discriminative incremental learning methods, including probabilistic ones, do not necessarily contain a reconstructive model component such as generative models do.

In the following we consider IVM with an isotropic stationary kernel, which represents affinities between the data samples. Although IVM are based on a discriminative model, which estimates decision boundaries between the classes $\mathcal{C}$, the import vectors are selected in order to increase the cross-entropy optimization function $-\sum_{nc} t_{nc} log P_{nc}(y_n = c|\boldsymbol{x}_n)$, see Eq. (4): All samples $\boldsymbol{x}_n$ with label $y_n = c$ aim for a high posterior probability P_{nc} for class c. Therefore, all samples chosen as import vectors contribute to the posterior probability P_{nc}, though they do not necessarily influence the position of the decision boundary. Using a stationary kernel with suitable kernel parameter implies that (a) import vectors cover the distribution of the data samples,

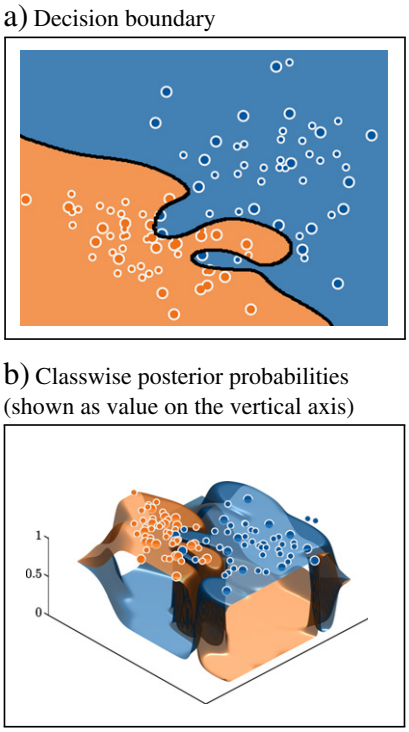


Fig. 6. Result of a two-class problem with two-dimensional features arising from IVM classifier. Import vectors are represented larger than non-import vectors.

(b) non-overlapping areas with a high density of data samples achieve a high posterior probability and (c) areas with no data samples obtain a posterior probability of approximately $\frac{1}{C}$. If we use a non-stationary kernel as a linear or polynomial kernel, characteristic (c) is not met. Properties (a) and (b) are necessary to enable a robust incremental learning even if the distribution of the data samples changes. Property (c) is necessary to enable an immediate response to the emergence of other classes in these areas and to make reliable claims about the posterior probability of test samples. The latter claim is quite intuitive, because we cannot make any assumption about the class membership of test samples lying in areas with no training samples.

4.2. Empirical study on the distribution of import vectors

We analyze this behavior using some representative examples with two classes, see Fig. 7 and Fig. 8. We start with two well separable classes, illustrate the distribution of the import vectors (IVs) and later show the distribution for overlapping classes. In all experiments we generate 500 two-dimensional samples for each class and apply the IVM algorithm. We repeat the procedure and accumulate the IVs, until at least 2000 IVs are accumulated. To visualize the distribution of the IVs we plot the accumulated IVs shown as orange and blue dots. Furthermore, we attach to each IV its Gaussian kernel with assigned kernel width, so that the accumulation of the kernels results in the background shading in the images. In each run we compute the distances between all IVs of each class and report the accumulated histograms for each class together with the theoretical distribution of the distances between randomly selected samples derived from the underlying distribution. In each run we choose as much random samples as IVs were selected.

4.2.1. Separable classes

We generate samples for each class, which are simulated from a Gaussian mixture with two components (top row) and from a uniform distribution (bottom row). We run the total experiment twice, once with the first IV selected according to Algorithm 1 (left column) and once selected randomly (right column). In all cases, we use a common Gaussian radial basis function kernel and choose both suitable kernel parameter $\sigma = 0.25$ and regularization parameter $\lambda = exp(-5)$ to achieve a small misclassification error.

Fig. 7 shows that the IVs obviously are spread over the entire data set. The left column in Fig. 7 shows that there is a hole around the center of each class distribution. The reason is, that IVs which are selected first lie in a region with a high density measured by the kernel, i.e. the center of each class. Furthermore, because of the sparsity property of the IVM algorithm and the property that IVs tend to repel one another, the neighboring samples are "covered" by IVs in the center of the distribution. Therefore, the distribution of the IVs is not identical to the density of the data samples. IVs are rather distributed so that areas with an adequate occurrence of data samples are uniformly covered by IVs. The effect is underlined by the histogram $h(d)$ of the distances between IVs. The histogram $h(d)$ of the Gaussian mixture distribution as well as the uniform distribution tends to have two peaks. Distances around the right peak belong to distances between samples in the center and non-centered IVs. In the example using Gaussian mixtures, the right peak also contains distances between the sub-distributions, the left

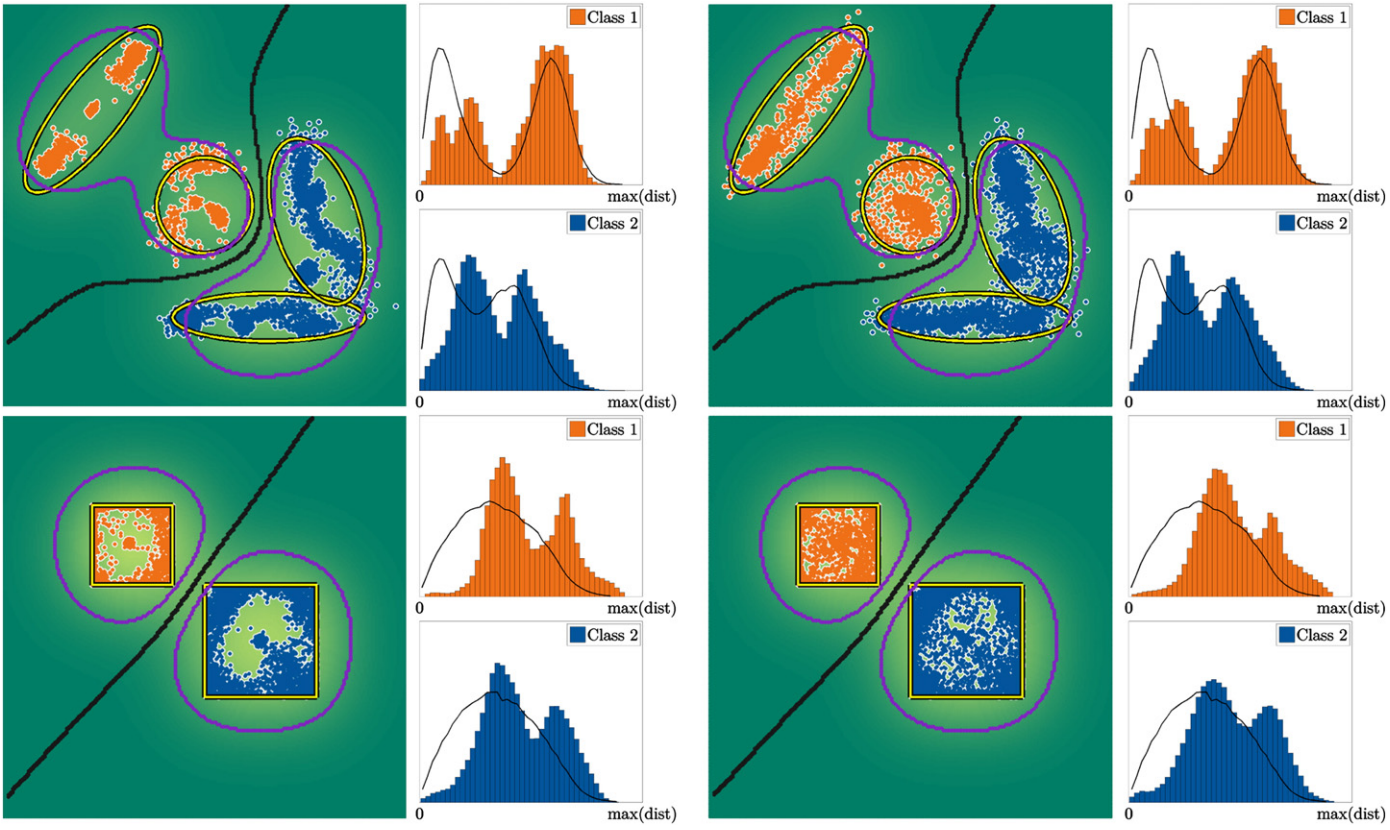


Fig. 7. Empirical distribution (dots) of import vectors (IVs) for features in a two class scenario with well separated classes: Gaussian (upper row) and uniform (lower row) distribution of the features. The size of the shown region is 2.5×2.5. The decision boundaries (black) estimated by IVM and the isolines of the 25% and the 75% posterior probabilities (violet) are given. Joint distribution from 500 runs of an IVM, each yielding appr. 15 IVs. The incremental sampling of the import vectors starts either with the best vector (left, see Algorithm 1) or with a random vector (right). The contours of the true distributions are given in yellow (3-σ contour for the upper row). The agglomerated histograms give the distribution of the distances between the IVs per run, indicating that IVs tend to lie separately, as the occurrences of small distances between IVs appear to be less frequent, compared to the theoretical distribution shown as lines. For the explanation of the holes see the text below.

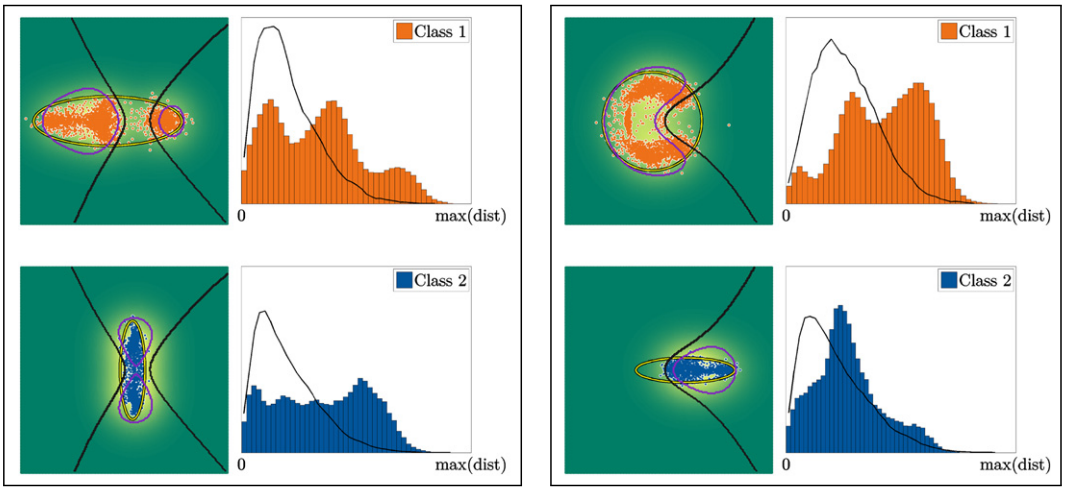


Fig. 8. Empirical distribution (dots) of import vectors (IVs) for Gaussian-distributed features in a two class scenario with overlapping classes. The 3-σ contour of $P(x|\mathcal{C})$ is given in yellow, the 75% contours of the posterior $P(\mathcal{C}|x)$ are given in violet for class c_k resp. Most of the IVs of a class lie in the acceptance region of the corresponding class.

peak are distances between the IVs in the center or distances outside the center.

We also report the analysis, provided that the first IV is chosen randomly in order to identify its purpose, see right column in Fig. 7. Now both peaks are less distinctive. Nevertheless, small distances between IVs appear to be less frequent than sampled from the original density.

4.2.2. Overlapping classes

The situation is slightly different in case of overlapping classes, see Fig. 8. Here, the import vectors belonging to one class in most cases do not lie in the acceptance region of the other class, but they uniformly cover the posterior probability distribution. Again, we observe the existence of IVs far off the decision boundary. The sparsity property in all cases is underlined by the fact that very small distances are the exception. The IVs try to keep away from each other, i.e. they do not represent a density function.

The empirical analysis on the distribution of the import vectors clearly indicates the IV selection leads to a sampling of the acceptance domain of the given training data. This empirical finding, which appears quite plausible, of course should be complemented by a theoretical analysis, making the statements more precise.

4.3. Comparison to SVM

We compare our results to that of standard SVM. SVM find an optimal nonlinear decision boundary by minimizing the objective function

$$\mathcal{Q}_{\mathrm{SVM}} = \frac{1}{N}\sum_n [1 - t_n f(\boldsymbol{x}_n)]_+ + \frac{\lambda}{2}\|\boldsymbol{f}\|^2 \tag{19}$$

with $f(\boldsymbol{x}_n) = \sum_n \alpha_n \boldsymbol{k}_n$. Basic SVM aim to maximize the margin between the hyperplane and the closest training samples, the so-called support vectors. They are a binary classifier, with the decision rule given by the sign of $f(\boldsymbol{x}_n)$.

The respective right plots in Fig. 9 show results for SVM with kernel parameter $\sigma = 0.25$ and $\lambda = exp(-5)$, which we also use for the experiments with IVM. We further show the results for $\lambda = exp(0)$ and $\sigma = 1.5$ in the respective left plot for both the Guassian mixture and the uniform distribution. In contrast to the uniformly distributed IVs, support vectors (SVs) of SVM are only selected if they are necessary for discrimination. Thus, most of the SVs are positioned near the decision boundary within the obtained margin in the feature space of the kernels. This not necessarily results in SVs being located near to the decision boundary in the original feature space, as can also be seen in the examples given by [68], p. 207, and [36], p. 425. We also plot the 75% isolines of the posterior probabilities (violet) which we obtain using Platt's method [58]. The figures show that the probabilities strongly depend on the decision boundaries. Areas, which lie far from the decision boundary achieve a high posterior probability, no matter if there are data samples. We observe the same effect, when using non-stationary kernels with IVM.

Fig. 10 shows the results for SVM for overlapping classes. The kernel and regularization parameter are determined via cross-validation. In these examples the SVs tend to be positioned near the decision boundary. This is underline by the histograms over the distances between the SVs, which shows that SVs are positioned close together covering only a small area. It can also be observed that overlapping areas also achieve a relatively high posterior probability, if they lie far enough away from the decision boundary.

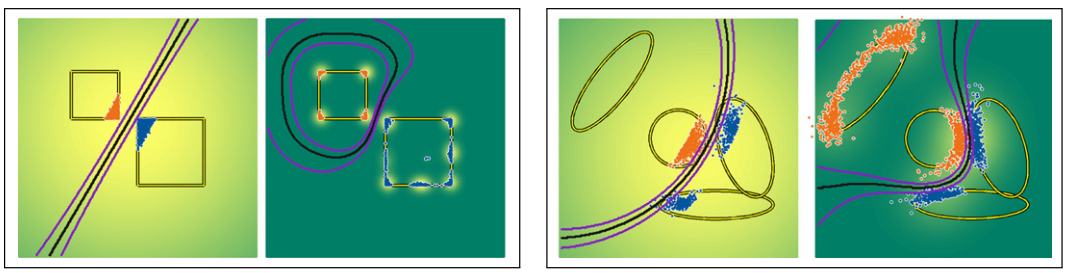


Fig. 9. Empirical distribution (dots) of support vectors (SVs) for features in a two class scenario with well separated classes: Uniform (left) and Gaussian (right) distribution of the features. The decision boundaries (black) estimated by SVM and the 25% and the 75% posterior probabilities (violet) obtained by Platt's method [58] are given. We use the regularization parameter $\lambda = exp(0)$ and the kernel parameter $\sigma = 1.5$ for the respective left plot and $\lambda = exp(-5)$ and $\sigma = 0.25$ for the respective right plot.

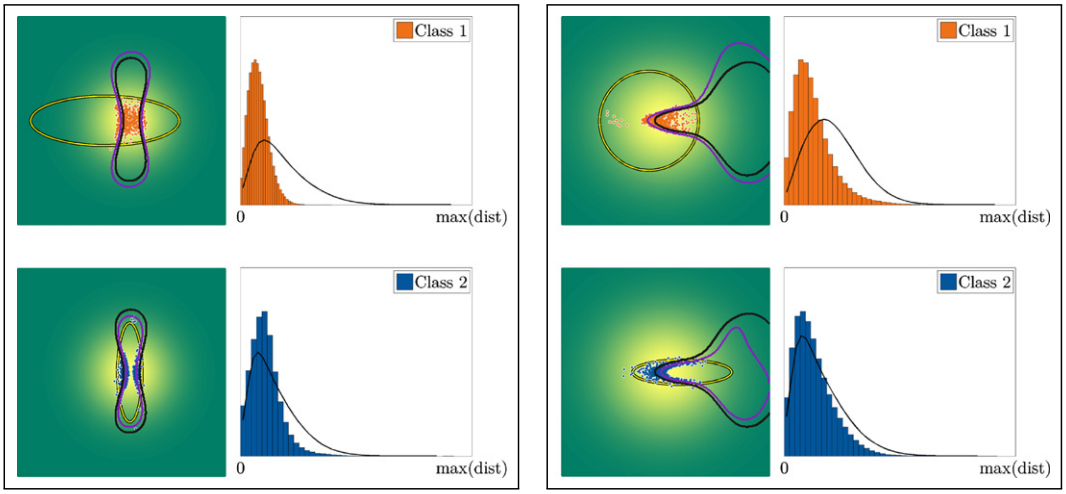


Fig. 10. Empirical distribution (dots) of support vectors (SVs) for Gaussian-distributed features in a two class scenario with overlapping classes. The 3-σ contour of $P(x|\mathcal{C})$ is given in yellow, the 75% contours of the posterior $P(\mathcal{C}|x)$ obtained by Platt's method [58] are given in violet for class c_k resp. Most of the SVs lie near to the decision boundary.

5. Incremental import vector machines

If the data samples become available sequentially, it is reasonable and more efficient to update the IVM incrementally, rather than recomputing them from scratch. In this section we introduce the I^{2}VM learner. We describe how to deal with new training samples becoming available at the current time step, how to remove training samples from older time steps and how to update the set of import vectors. We will describe the learning procedure for the two class situation, dropping the index c for a simplifying notation.

5.1. Representing the current state of the learner

The state of the learner is represented by two sets and additional matrices and vectors, which will change over time:

- A training set $\mathcal{T} = \{(\boldsymbol{x}_n, t_n)\}, n = 1, \ldots, N$ being a subset of all training samples seen so far.
- An import vector set $\mathcal{V} = \{\boldsymbol{x}_v, t_v\}, v = 1, \ldots, V$ being a subset of $\mathcal{T}$. It is used for defining the classifier model and implicitly specifies the decision boundaries.

The following vectors and matrices support efficient learning, and are partially redundant.

- the (N×V) kernel matrix $K = [k_v]$,
- the V-vector $\boldsymbol{\alpha}$ of parameters,
- the N-vector $\boldsymbol{p}$ of probabilities, and for efficiency the (N×N) matrix R,
- the (V×V) matrix $H = \nabla^2 \mathcal{Q}(\boldsymbol{\alpha})$ being the Hessian of the optimization function and, for efficiency, its inverse H^{-1}, and
- the V-vector z.

Thus the state is represented by

$$S = \left\{\mathcal{T}, \mathcal{V}; K, \boldsymbol{\alpha}, \boldsymbol{p}, R, H, H^{-1}, z\right\}. \tag{20}$$

When working in the incremental setting, we indicate the current time step with one prime and next time step with a double prime. The learning procedure is a Markov chain in which a new state S'' only depends on the previous state S', and the set of training data $\Delta\mathcal{T}_+$ becomes available between state S' and S''.

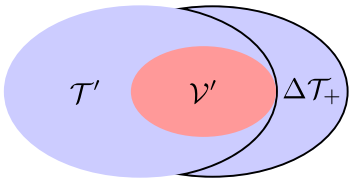
5.2. Learning procedure

For the incremental learner, we distinguish between three update steps that serve a different purpose.

(1) The addition of training vectors allows the learner to add new samples to already existing ones. As a result, the learner is able to adapt to new class-specific samples or enhance or support existing ones.
(2) The removal of training vectors is necessary to maintain the learner of bounded size. Depending on the application non-informative, counter-supportive or the oldest data samples are removed.
(3) The set of import vectors has to be adapted, like in classical IVM. This includes the adding of the new IVs from the new training samples to catch concept-drifts and the removal of some import vectors, which do not represent the best subset any more.

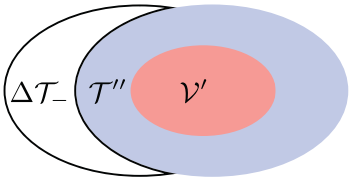
In detail, we have the following three update steps:

1. Adding training vectors: Extend the old set of training vectors $\mathcal{T}'$ to $\mathcal{T}_+$ by including new vectors $\Delta\mathcal{T}_+ = \Delta\mathcal{T}''$:



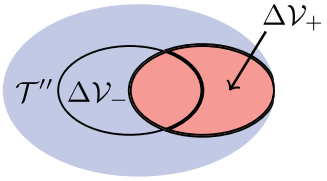
$$\mathcal{T}_+ = \mathcal{T}' \cup \Delta\mathcal{T}_+ \tag{21}$$

2. Remove training vectors $\Delta\mathcal{T}_-$, but not if they are currently included in the set of import vectors $\mathcal{V}''$:



$$\mathcal{T}'' = \mathcal{T}_+ \setminus \Delta\mathcal{T}_- \tag{22}$$

3. Incremental and decremental IV-learning: Add import vectors $\Delta\mathcal{V}_+ \subset (\mathcal{T}'' \; \mathcal{V}')$, and remove import vectors $\Delta\mathcal{V}_- \subset \mathcal{V}'$ with a hybrid greedy forward/backward selection as described in Algorithm 1 in Section 3.5 until the error function $\mathcal{Q}$ converges:



$$\mathcal{V}'' = (\mathcal{V}' \setminus \Delta\mathcal{V}_-) \cup \Delta\mathcal{V}_+. \tag{23}$$

The criteria for deleting training vectors depend on the application. In tracking applications, training vectors can be removed according their "age", or if older class-specific features are counter-supportive to newer ones. In contrary to this, for the incremental learning of large data sets the sequence of the incoming training vectors should not influence the result. In this case, a suitable criterion is to remove the training vectors depending on their contribution to the model.

There are also several ways to realize step 3. One possibility is to test each data sample in the current training set to be an import vector. Another possibility, which more likely corresponds to incremental learning, is only to test the newly acquired data samples. We choose the latter possibility, which is more efficient than the first one.

The three update steps are now made explicit.

5.3. Incremental and decremental update

In the following we describe the incremental and decremental update of training and import vectors for the two-class case. For convenience we show the update of import vectors only for one single import vector. The update can be extended to simultaneously or successively add or remove groups of import vectors.

5.3.1. Adding training vectors

We add ΔN_+ training vectors ΔX_+ with targets Δt_+ so that $N_+ := N' + \Delta N_+$. The kernel submatrix of the new training vectors is given by ΔK_+. We extend the kernel matrix and the target vector to

$$\underset{N_+ XV}{K_+} = \begin{bmatrix} K' \\ \Delta K_+ \end{bmatrix}, \tag{24}$$

$$\underset{N_+ X1}{t_+} = \begin{bmatrix} t \\ \Delta t_+ \end{bmatrix}. \tag{25}$$

Using the current parameters α' we first obtain the posteriors $\Delta\boldsymbol{p}_+$ from Eq. (14), ΔK_+ from Eq. (12), ΔR_+ from Eq. (7) using $\Delta\boldsymbol{p}_+$ and Δz_+ from Eq. (16).

Starting from the Hessian

$$H' = \frac{1}{N'} K'^T R' K' + \lambda K'_R \tag{26}$$

and its inverse from the previous iteration we update these matrices and the parameters $\boldsymbol{\alpha}'$ using the IRLS procedure, yielding

$$\boldsymbol{\alpha}_+ = H_+^{-1}\left(K'^T R' z' + \Delta K_+^T \Delta R_+ \Delta z_+\right) \tag{27}$$

with

$$\begin{aligned} H_+ &= \frac{1}{N_+} K_+^T \begin{bmatrix} R' & \\ & \Delta R_+ \end{bmatrix} K_+ + \lambda K'_R, \\ &= H' + \Delta H_+, \\ &= \frac{N'}{N_+}\left(H' + \frac{1}{N'}\Delta K_+^T \Delta R_+ \Delta K_+ + \lambda \frac{\Delta N_+}{N'} K_R\right), \end{aligned} \tag{28}$$

$$\Delta z_+ = \Delta K_+ \boldsymbol{\alpha}' + \Delta R_+^{-1}(\Delta\boldsymbol{p}_+ - \Delta\boldsymbol{t}_+). \tag{29}$$

With H' given from Eq. (26) we update and invert the Hessian with Eq. (28). Since the inverse H_+^{-1} is only of size $(V' \times V')$ and the number of import vectors is usually small, the inverse can be computed significantly faster than for non-sparse kernel logistic regression.

Using $\boldsymbol{\alpha}_+$ we finally determine $\boldsymbol{p}_+$ from Eq. (14), R_+ from Eq. (7), and finally z_+ from Eq. (16), yielding the state $S_+ = \{\mathcal{T}_+, \mathcal{V}'; K_+, \boldsymbol{\alpha}_+, \boldsymbol{p}_+, R_+, H_+, H_+^{-1}, z_+\}$.

5.3.2. Removing training vectors

We delete I training vectors with indices $\mathcal{I}$ and obtain the partitioning of the matrix K_+

$$K_+ = \begin{bmatrix} K_- \\ \Delta K_- \end{bmatrix} \text{ with } \Delta K_- = \Psi_{\mathcal{I}}(K_+), \tag{30}$$

and the target vector $\boldsymbol{t}_+$

$$\boldsymbol{t}_+ = \begin{bmatrix} \boldsymbol{t}_- \\ \Delta\boldsymbol{t}_- \end{bmatrix} \text{ with } \Delta\boldsymbol{t}_- = \Psi_{\mathcal{I}}(\boldsymbol{t}_+), \tag{31}$$

and the deleted parts of R_+ and z_+

$$\Delta R_- = \Psi_{\mathcal{I}}\left(\Psi_{\mathcal{I}}(R_+)^T\right), \tag{32}$$

$$\Delta z_- = \Psi_{\mathcal{I}}(z_+). \tag{33}$$

The update (27) can be formulated for a decreasing number of training vectors $N'' := N_+ - \Delta N_-$ in a similar manner, using an efficient determination of H_- as in Eq. (28):

$$\boldsymbol{\alpha}_- = H_-^{-1}\left(K_+{}^T R_+ z_+ - \Delta K_-^T \Delta R_- \Delta z_-\right) \tag{34}$$

with

$$\begin{aligned} H_- &= \frac{1}{N''} K_-^T \left(\Psi_{\mathcal{T}''\setminus\mathcal{I}}\left(\Psi_{\mathcal{T}''\setminus\mathcal{I}}(R_+)^T\right)\right) K_- + \lambda K_R, \\ &= H_+ - \Delta H_-, \\ &= \frac{N_+}{N''}\left(H_+ - \frac{1}{N_+}\Delta K_+^T \Delta R_+ \Delta K_+ - \lambda \frac{\Delta N_-}{N_+} K_R\right) \end{aligned} \tag{35}$$

and the Hessian H_+ from Eq. (28). Applying α_- we again obtain $\boldsymbol{p}_-$, R_-, and z_- using Eqs. (14), (7) and (16), yielding the state $S_- = \{\mathcal{T}'', \mathcal{V}'; K_-, \boldsymbol{\alpha}_-, \boldsymbol{p}_-, R_-, H_-, H_-^{-1}, z_-\}$.

5.3.3. Adding and removing import vectors

In the following we show how to add and remove import vectors. For convenience we only show the addition and the removal of one import vector, in each case given the state S_-. In general an update step consists of several insertions and removals of import vectors, depending on how much the model must be adapted.

5.3.3.1. Adding import vectors. To add an import vector $\{\boldsymbol{x}^{\text{add}}, y^{\text{add}}\} \in \mathcal{T}''$ out of the current set of N'' training vectors, we define a kernel vector $\Delta\tilde{\boldsymbol{k}}_+ = [k(\boldsymbol{x}_n, \boldsymbol{x}^{\text{add}})]$, a kernel vector $\Delta\tilde{\boldsymbol{k}}_{R,+} = [k(\boldsymbol{x}_v, \boldsymbol{x}^{\text{add}})]$ and a kernel scalar $\Delta\tilde{k}_{R,+} = k(\boldsymbol{x}^{\text{add}}, \boldsymbol{x}^{\text{add}})$. We extend the kernel matrix and the

regularization matrix to

$$\tilde{K}_+ = \begin{bmatrix} K_- & \Delta\tilde{k}_+ \end{bmatrix}, \tag{36}$$

$$\tilde{K}_{R,+} = \begin{bmatrix} K'_R & \Delta\tilde{k}_{R,+} \\ \Delta\tilde{k}_{R,+}^T & \Delta\tilde{k}_{R,+} \end{bmatrix} \tag{37}$$

leading to $V_+ := V' + 1$ import vectors. The parameters $\tilde{\boldsymbol{\alpha}}_+$ are obtained from

$$\tilde{\boldsymbol{\alpha}}_+ = \tilde{H}_+^{-1}\tilde{K}^T R_- z_-. \tag{38}$$

We use the Sherman–Morrison–Woodbury (SMW) formula [37] to efficiently compute the inverse in Eq. (38) with

$$\begin{aligned} \tilde{H}_+^{-1} &= \left(\frac{1}{N''}\tilde{K}_+^T R_- \tilde{K}_+ + \lambda\tilde{K}_{R,+}\right)^{-1}, \\ &= \begin{bmatrix} H_- & \boldsymbol{a} \\ \boldsymbol{a}^T & b \end{bmatrix}^{-1} = \begin{bmatrix} H_-^{-1} + \left(H_-^{-1}\boldsymbol{a}\right)e^{-1}\left(\boldsymbol{a}^T H_-^{-1}\right) & -H_-^{-1}\boldsymbol{a}e^{-1} \\ -e^{-1}\boldsymbol{a}^T H_-^{-1} & e^{-1} \end{bmatrix} \end{aligned} \tag{39}$$

where

$$\boldsymbol{a} = \frac{1}{N''}\tilde{K}_+^T R_- \Delta\tilde{\boldsymbol{k}}_+ + \lambda\Delta\tilde{\boldsymbol{k}}_{R,+}, \tag{40}$$

$$b = \frac{1}{N''}\Delta\tilde{\boldsymbol{k}}_+^T R_- \Delta\tilde{\boldsymbol{k}}_+ + \lambda\Delta\tilde{k}_{R,+}, \tag{41}$$

$$e = b - \boldsymbol{a}^T H_-^{-1}\boldsymbol{a}. \tag{42}$$

Note that the determination of the inverse $\tilde{H}_+^{-1}$ only incorporates an inverse e^{-1} of size 1×1, since the inverse H_-^{-1} is already given from the last step. Using $\tilde{\boldsymbol{\alpha}}_+$ we obtain $\tilde{\boldsymbol{p}}_+$, $\tilde{R}_+$, and $\tilde{z}_+$, yielding the state $\tilde{S}_+ = \left\{\mathcal{T}'', \mathcal{V}_+; \tilde{K}_+, \tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{p}}_+, \tilde{R}_+, \tilde{H}_+, \tilde{H}_+^{-1}, \tilde{z}_+\right\}$.

Further import vectors can be added by extending the kernel and the regularization matrix again and using the updated entities of $\tilde{S}_+$.

5.3.3.2. Removing import vectors. Using the set $\mathcal{V}' = \left\{1, \ldots, i, \ldots, V'\right\}$ of indices of the current import vectors a removal of an import vector $\{\boldsymbol{x}_i, y_i\}$ reduces the kernel and regularization matrix and yields:

$$\tilde{K}_- = \left(\Psi_{\mathcal{V}'\,i}\left(K_-^T\right)\right)^T, \tag{43}$$

$$\Delta\tilde{K}_- = \left(\Psi_i\left(K_-^T\right)\right)^T, \tag{44}$$

$$\tilde{K}_{R,-} = \Psi_{\mathcal{V}'\,i}\left(\Psi_{\mathcal{V}'\,i}\left(K'_R\right)^T\right), \tag{45}$$

$$\Delta\tilde{k}_{R,-}^T = \Psi_i\left(\Psi_i(K_R)^T\right), \tag{46}$$

the consequence being that the Hessian H_- is reduced to

$$\tilde{H}_- = \Psi_{\mathcal{V}'\setminus i}\left(\Psi_{\mathcal{V}'\setminus i}(H_-)^T\right). \tag{47}$$

We can efficiently compute the inverse Hessian $\tilde{H}_-^{-1}$ by updating H_-^{-1}, given from the last step, with

$$\tilde{H}_-^{-1} = H_-^{-1} - h_{ii}\boldsymbol{h}_i^T\boldsymbol{h}_i, \tag{48}$$

Table 1
Characteristics of the used benchmark data sets.

Data set	#Train	#Test	#Classes	#Features
DIGITS	1900	1800	10	612
DIGITS(2 classes)	1900	1800	2	612
DNA	1400	1186	3	180
USPS	7291	2007	10	256
SATIMAGE	3104	2000	6	36
VOWEL	528	462	11	10

whereby $\boldsymbol{h}_i$ is the ith row from the Hessian H_-. Both, the i-th row and the i-th column of $\tilde{H}_-^{-1}$ become zero and must be removed to compute the updated parameters with

$$\tilde{\alpha}_- = \tilde{H}^{-1}\tilde{K}_-^T R_- z_-. \tag{49}$$

Using $\tilde{\boldsymbol{\alpha}}_-$ we obtain $\tilde{\boldsymbol{p}}_-$, $\tilde{R}_-^T$, and $\tilde{z}_-$. Further import vectors can be removed in the same way by a further reduction of the kernel matrix and the Hessian.

Depending on the number of insertions and/or deletions of training and import vectors we finally have the new state S'', consisting of the new sets $\mathcal{T}''$ and $\mathcal{V}''$, the kernel matrix K'', the parameters $\boldsymbol{\alpha}''$, the probabilities $\boldsymbol{p}''$ and R'', the Hessian H'', its inverse H''^{-1}, and the corrections z, which will be updated in the next iteration.

6. Experiments

In the following section the potential of I^{2}VM is presented for different experiments, using (i) widely used machine learning benchmark data sets and (ii) remote sensing images for land cover classification of large areas. In (i) the performance of I^{2}VM is compared to other well-known incremental learners. In (ii) we show that I^{2}VM are a suitable classifier for self-training, which can be used for land cover classification of large areas. For a more detailed analysis of the experiment, we refer to the publication of [62].

6.1. Classification of benchmark data sets

In order to evaluate I^{2}VM, we show that the incremental method is competitive to a) its batch counterpart and b) to other incremental learning algorithms, by using benchmark data sets in the first experiment. Additionally, to underline our findings in Section 4 we sort the data to simulate a pseudo-concept-drift within the data.

We use well-known machine learning data sets, including the DIGITS data set [70], consisting of images of digits from 0 to 9, as well as the data sets DNA, USPS, SATIMAGE and VOWEL from the LibSVM repository [13]. For the DIGITS data set we compute HoG features [20] of each image and use them for classification purposes. For the DIGITS (2 classes) data set, class 1 is composed of the images of digits 0, 2, 4, 6 and 8 and

Table 2
Classification error on common machine learning data sets. Bold numbers indicate the best result of an incremental learner. For comparison we also report the result of IVM in the last column.

Data set	Inc. PCA/LDA [%]	Online RF [%]		LaRank [%]	I^{2}VM [%]	Batch IVM [%]
		1 epoch	10 epochs			
DIGITS	25.7	26.7	18.4	11.8	**10.0**	9.6
DIGITS(2 classes)	39.3	13.9	9.6	**6.4**	7.0	7.1
DNA	23.4	22.4	7.9	**5.6**	5.8	5.5
USPS	11.8	20.7	10.9	**4.3**	5.4	5.6
SATIMAGE	13.6	15.7	11.5	**9.4**	9.6	9.6
VOWEL	84.4	58.0	44.8	46.1	**42.6**	42.2
Synthetic	34.5	19.9	6.3	7.4	**5.3**	5.8

Table 3
Classification error on synthetic data set with pseudo-concept-drift in one class and sorted machine learning data sets. The best results of an incremental learner are bold printed. The numbers in brackets indicate the difference to the corresponding result with unsorted data using the same classifier.

Data set (sorted)	Inc. PCA/LDA [%]	Online RF [%]		LaRank [%]	I²VM [%]
		1 epoch	10 epochs		
DIGITS	25.7(±0.0)	23.9(−2.8)	17.7(+0.7)	14.6(+2.8)	**9.8(−0.2)**
DIGITS (2 classes)	42.9(+3.6)	18.3(+4.4)	12.4(+2.8)	15.8(+9.4)	**7.2(+0.2)**
DIGITS reverse (2 classes)	43.1(+3.8)	15.5(+2.8)	11.7(+2.1)	10.7(+4.3)	**7.1(+0.1)**
DNA	23.2(−0.2)	42.1(+27.2)	11.4(+0.9)	28.5(+22.9)	**5.8(±0.0)**
USPS	11.6(−0.2)	18.6(−2.1)	14.3(+3.4)	13.7(+9.4)	**5.8(+0.4)**
SATIMAGE	14.3(+0.7)	18.5(+2.8)	16.3(+4.8)	10.5(+1.1)	**10.1(+0.5)**
VOWEL	90.9(+6.5)	53.0(−5.0)	44.4(−0.4)	47.5(+1.4)	**43.1(+0.5)**
Synthetic	34.5(±0.0)	24.0(+4.1)	6.5(+0.2)	24.0(+16.6)	**5.5(+0.2)**

class 2 is composed of the images of digits 1, 3, 5, 7 and 9. The characteristics of the data sets are collected in Table 1.

6.1.1. Static data

We compare I²VM to three multi-class classifiers: the online random forests (ORF, [67]), incremental PCA/LDA [78] and LaRank [4], an incremental SVM. For the ORF and the LaRank algorithm we use the software available online and for the incremental PCA/LDA, IVM and I²VM we use our own implementation in Matlab and C++. Moreover, batch IVM is used for comparison. The kernel and regularization parameter for LaRank and I²VM are determined via 5-fold crossvalidation. For the ORF we report the results for 1 and 10 epochs, where in both variants every individual data sample is read once in each epoch.

The classification results are shown in Table 2.

Table 2 shows the classification result of IVM, I²VM and other incremental learners. It can be seen, that I²VM is competitive to its batch version. The results also indicate that the discriminative classifiers LaRank and I²VM perform better than ORF and the generative PCA/LDA.

6.1.2. Data with concept-drift

Concept drifts, i.e. changes of the distribution of the features over time can be critical for actual discriminative models, especially for larger changes. Therefore one can expect learners which only have a discriminative model component, such as SVM, to perform worse than learners which contain a reconstructive model component.

In our first experiment we analyze the performance of I²VM on machine learning benchmark data sets, after sorting the data to simulate a pseudo-concept-drift. We sorted the training data according to their class membership, i.e. the first part of the sequence contains all data samples with class label $c_n < \frac{C}{2}$ and the second part contains all data samples with class label $c_n \geq = \frac{C}{2}$. The classification results for the different learners are given in Table 3.

All incremental learners except I²VM result in significantly larger error rates for the sorted data set in comparison to the unsorted data set. This indicates the I²VM to be more flexible than the other considered incremental learners. The result also indicates, that LaRank and ORF outperform the generative classifier PCA/LDA. Thus I²VM combines the power of a discriminative classifier with a generative component in order to be robust against concept-drifts.

The second experiments show the effect of a real concept-drift. We use synthetically generated data with two classes, of which one class is fixed and the other has a concept-drift. The fixed class is a Gaussian distribution placed in the center of Fig. 11. The drifting class is generated by adding Gaussian distributed data samples with drifted mean and changed variance, each time according to the sequence shown in the figure. We start with 25 samples per class simulated from two Gaussian distributions to train the classifier. Then the distribution of the second class changes for each time step. The test data is generated from the complete Gaussian mixture distribution.

Fig. 12 shows the classification error of I²VM as function of the used training samples. The number of training samples is identical to the number of added training samples, because there is no need for an initial trained model. The plot shows the result for the synthetic data set with pseudo-concept-drift and random ordering. The stepwise shape of the curve of the sorted data set indicates when a mixture of the distribution of class 2 was learned and included in the model. Both orderings yield the same result, so at the end I²VM are independent of the sorting of the data.

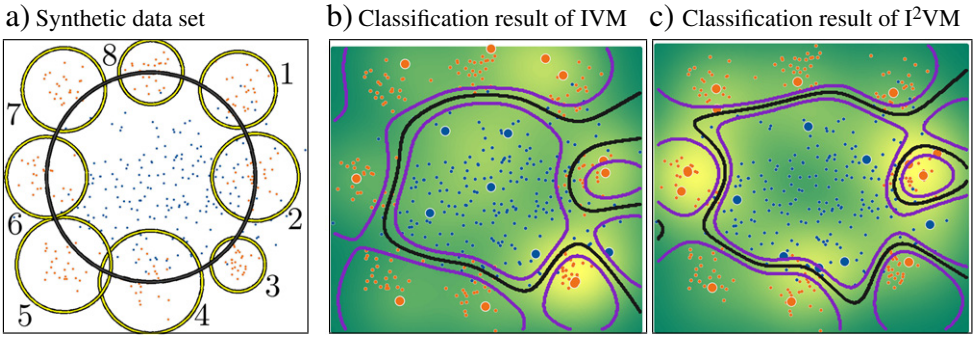


Fig. 11. Synthetic data set with generated concept-drift in the distribution of one class. Left: The yellow circles are the 3-σ intervals of the covariance matrices of the Gaussian mixture distribution of class 1 and the dark gray circle is the Gaussian distribution of class 2 without concept-drift. The numbers next to the yellow circles indicate the order, in which the samples are presented to the learner. We start with 2 Gaussian distributions, one for each class, and further ones with concept-drift to the training samples. Middle: Result with IVM. Right: Result with I²VM. The underlying color in the middle and right image represents the frequency of the import vectors, the black line is the estimated decision boundary and the violet lines are the $P(\mathcal{C}|X) = 0.75$ isolines. The fat blue and orange dots are the IVs after finishing the training. The small blue and orange dots are the training samples.

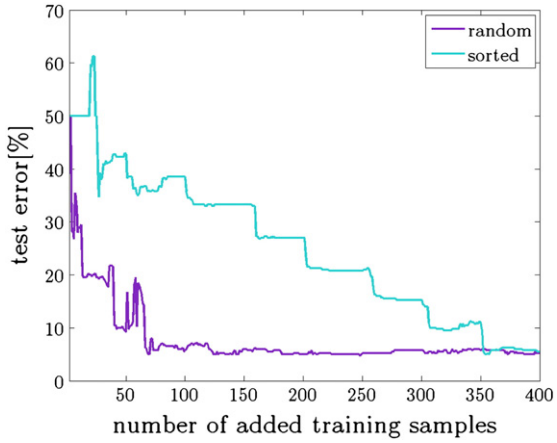


Fig. 12. Classification error of I²VM as a function of used training samples in the synthetic data set with pseudo-concept-drift and random ordering. The number of used training samples is identical to the number of added training samples, because there is no need for an initial trained model.

Fig. 13. The area of Rondonia with overlayed Landsat images displayed with bands 4–3–2. Numbers in the upper left corner of the Landsat images are referred to the numbers in the text. The three image strips are acquired in September (images 1, 4 and 7), July (images 2, 5 and 8), and August 2009 (images 3, 6 and 9). The underlaid image is taken from Google Earth. Paths between the 9 images across-track and along-track. Each pair of overlapping images i and j has a common area i–j which is used to transfer the classification model from i to j or vice versa.

Fig. 11 shows the generated synthetic data set and the resulting classification result of the I²VM. The middle and the right plot shows, that the IVs are spread over the entire data set, i.e. the IVs covers old and new samples equally. Both, the batch and incremental version do not differ significantly.

6.2. Large area land cover classification

Land cover classification is one of the main applications in the field of remote sensing image analysis. Currently, the development in land cover classification of remote sensing images is mainly driven by methods from the field of machine learning and pattern recognition and the increased availability of remote sensing data [61]. Furthermore, increasing processing power and more sophisticated algorithms enables faster processing of huge data sets. However, most methods are focusing on spatial limited areas, while many applications require large area land cover maps [16,24]. Hence, an appropriate classification of large areas is an important and ongoing research topic in the field of remote sensing [46,57,81]. The classification of data with high spatial resolution and often relative narrow swath, e.g. scenes from Landsat-line data, is usually performed on individual images. In addition, most classifications are based on a supervised method, which requires the – often costly and time consuming – selection of adequate training data in the area covered by each individual image. As a matter of fact, the mapping of large areas is challenging and requires an efficient classification concept. Nevertheless, the use of Landsat data seems interesting, due to (i) the long-term data archive, (ii) the availability of free Landsat data by the USGS and (iii) the relatively large swath width and adequate spatial resolution. The authors of [46], for example overcome the challenges mentioned above by using overlapping areas of neighboring Landsat scenes. In the beginning an initial image, with corresponding training data, is individually classified by a standard SVM. The classification result is used to identify new training samples in the overlapping areas. These training samples are used to train a new SVM model for the neighboring scene. While this strategy was successfully used for generating a forest/non-forest map (i.e. a binary classification), [62] extended the concept to multi-class problems. Moreover, I²VM are used, which enables the efficient classification of neighboring Landsat scenes with a larger number of training data.

In the following we want to exploit the possibility of transferring a classifier across multiple neighboring scenes using I²VM. Though we use the original data we are aware that adequate preprocessing, see e.g. [3] can reduce the spatial and temporal variability and thus further increase the performance.

6.2.1. Data set

Fig. 13 shows our chosen study site around Rondonia in South America. The data set contains 9 Landsat 5 TM images from 2009, which are freely available from the USGS Landsat archive. The area covers about 285,000 km².

The study site is dominated by forests and agriculture/deforestation, and characterized by typical spatial patterns and temporal variation caused by different acquisition dates, i.e. three image strips are acquired in September (images 1, 4 and 7), July (images 2, 5 and 8), and August 2009 (images 3, 6 and 9).

The classification is aiming at 4 land cover classes, focused on FOREST, AGRICULTURE, WATER and URBAN. A limited number of training and test data were collected by photointerpretation, using the Landsat images themselves and high-resolution images in Google Earth.

The land cover classes occur in every scene, but not in every overlap area. In each scene tens of thousands of pixels are annotated. We choose 500 pixels from each class for training and the rest for testing.

6.2.2. Self-training for sequential mosaic classification

Self-training as a special case of active learning can be used to classify each scene in a mosaic, when only a limited number of labeled data samples are available. For further discussion of active learning we refer to [77]. We pursue the following strategy: Starting with an initial classification in a single image, the classification result in the overlapping area is used to retrain or update the learned classifier for a neighboring scene. Given two neighboring scenes I_c, the current scene, and I_t, the target scene, a classifier is trained on I_c and labels the unlabeled data in the overlapping area. The predictions selected regarding some criteria are used as new training labels. These samples and the features of I_t are used as new training samples to retrain or update the classifier, and finally to classify the image. Subsequently the remaining images can be classified by following the same procedure.

The general concept of this approach seems advantageous due to the fact that training samples are only required for the initial image. However, this approach requires that all classes of interest occur in each overlapping area. This cannot be guaranteed particularly for small classes. To overcome this problem we keep old labeled training data and use I²VM to incrementally adapt the model to the current target scene. Although this approach is useful to acquire new labeled samples, it should be noted that using self-training by itself does not necessarily lead to better results. The main problem of self-training is that acquired samples with incorrect labels reinforce themselves. It is

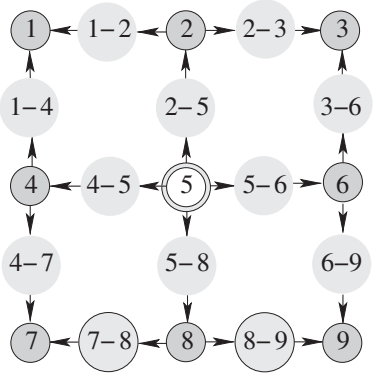


Fig. 14. Paths between the 9 images across-track and along-track. Each pair of overlapping images i and j has a common area i–j which is used to transfer the classification model from i to j or vice versa.

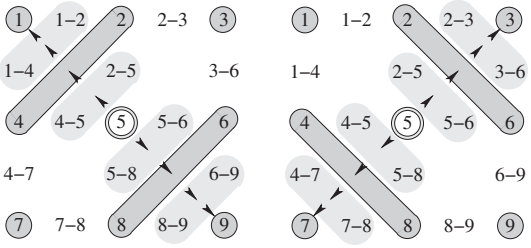


Fig. 15. Paths between the 9 images across-track, i.e. diagonal to the grid. The transfer from one image to a diagonal one uses the two neighbors and the four overlapping regions.

caused by the assumption that the classifier's own predictions with high confidence are correct. As one can imagine this may not always be the case. Also correctly acquired samples may deteriorate the model if they cause an imbalance in the data distribution or their distribution is different to that of the labeled samples. If the number of labeled samples in the initial scene is to small the model may not be accurate enough to acquire new samples with correct labels. On the other hand, if the number of labeled samples is too big, the model may not be improved anymore. Therefore, the identification of confident samples, which also ensure an improvement of the classifier model usually demand further strategies.

6.2.3. Experimental setup

In our experiments we use the center image I_5 as initial scene to train the initial classifier. We randomly choose 500 points per class and train the IVM as described in Section 3.5.

We compare different paths for transferring the classification model from the center image to the other images:

1. We transfer the classification model to the direct neighbors, i.e., 2, 4, 6 and 8, using the corresponding overlapping regions 5–2, 5–4, 5–6, and 5–8.
2. We transfer the classification model to the indirect neighbors, i.e., to 1, 3, 7 and 9. This can again be performed in two ways:
   (a) We use single paths via one of the two direct neighbors, e.g. $5\rightarrow2\rightarrow1$ or $5\rightarrow4\rightarrow1$ (Fig. 14).
   (b) We merge two single paths. E.g. we simultaneously use the overlapping regions 5–4 and 5–2 for reaching the image pair (2,4), and then use the two overlapping regions 1–2 and 1–4 for reaching the image 1, see Fig. 15, left.

Within each overlapping area we choose the most confident predictions measured by the probability and randomly select not more than 1000 training samples per class. We fix the confidence threshold to 0.8, i.e. samples with a posterior probability in [0, 0.8] are discarded and only those in [0.8, 1] are considered as new training samples. All training samples are at least 10 pixel apart from training samples assigned to other classes to ensure a more reliable acquisition.

To evaluate our results we classified each image separately, using a common IVM classification and individual training data for each image (from now on referred to as reference classifications). We compare I^{2}VM to the following approaches:

Old model. We use the initial trained model in scene I_c to classify the target scene I_t.
IVM (overlap). We only use the training data from the overlapping area of I_c and I_t as proposed in [46].
IVM (batch). We use the initial training data and the newly acquired training data from the overlapping area(s) and retrain the model from scratch.
I^{2}VM. We update our initial classifier incrementally by adding the new training data without retraining from scratch.

We report the overall accuracy (OA), averaged class-wise accuracies (AA) and the kappa coefficient [17] for the classification. For a more detailed analysis of the experiment and the data set, we refer to the publication of [62].

Table 4
Accuracy assessment of classified target scenes. We distinguish the two separate paths via a single image (first 4 rows) and the unique path via the two direct neighbor images. The best results regarding the overall accuracy are printed in bold.

Transfer	Reference			Old model			IVM (overlap)			I^{2}VM			IVM (batch)		
	OA	AA	Kappa	OA	AA	Kappa	OA	AA	Kappa	OA	AA	Kappa	OA	AA	Kappa
5→2	95.4	96.1	0.94	**93.9**	94.9	0.92	86.9	88.1	0.82	93.6	94.6	0.91	93.6	94.6	0.91
5→4	97.0	96.7	0.96	63.0	71.2	0.49	82.2	78.0	0.73	**92.0**	91.8	0.88	91.8	91.7	0.88
5→6	96.5	96.5	0.95	83.3	90.8	0.76	89.8	89.6	0.84	94.0	94.8	0.91	**94.2**	94.8	0.91
5→8	96.8	97.3	0.95	**97.2**	92.4	0.95	96.9	74.6	0.95	97.1	91.2	0.95	97.1	91.4	0.95
5→4→1	93.0	91.3	0.89	79.8	82.8	0.71	64.2	50.0	0.48	**80.5**	84.3	0.73	80.4	84.3	0.72
5→2→1	93.0	91.3	0.89	79.8	82.8	0.71	**93.8**	72.8	0.90	93.0	87.0	0.90	92.9	85.4	0.89
5→(4+2)→1	93.0	91.3	0.89	79.8	82.8	0.71	–	–	–	**92.6**	83.9	0.89	**92.6**	84.4	0.89
5→2→3	95.8	96.4	0.92	73.2	87.8	0.60	**92.2**	73.9	0.84	83.9	90.8	0.73	83.1	90.7	0.72
5→6→3	95.8	96.4	0.92	73.2	87.8	0.60	66.1	50.0	0.14	**90.3**	89.3	0.82	88.7	89.8	0.80
5→(2+6)→3	95.8	96.4	0.92	73.2	87.8	0.60	–	–	–	93.1	88.8	0.87	**93.2**	88.8	0.87
5→4→7	92.6	94.4	0.88	61.2	79.0	0.50	54.0	61.2	0.27	65.8	83.7	0.56	**66.0**	83.8	0.56
5→8→7	92.6	94.4	0.88	61.2	79.0	0.50	**92.2**	86.6	0.87	91.5	85.2	0.86	91.6	86.0	0.86
5→(4+8)→7	92.6	94.4	0.88	61.2	79.0	0.50	–	–	–	**91.4**	85.1	0.86	91.3	84.2	0.86
5→6→9	93.5	94.3	0.90	87.5	91.1	0.82	53.7	60.9	0.27	90.0	92.1	0.85	**90.8**	92.7	0.86
5→8→9	93.5	94.3	0.90	87.5	91.1	0.82	73.8	58.2	0.58	93.0	93.5	0.89	**93.4**	94.0	0.90
5→(6+8)→9	93.5	94.3	0.90	87.5	91.1	0.82	–	–	–	92.9	93.8	0.89	**93.0**	93.6	0.89

6.2.4. Results and discussion

Table 4 shows the reference classification of each Landsat scene in the mosaic. The overall accuracy ranges from 93% for scene 1 to 97.0% for scene 4. The initial image with ID 5 has an overall accuracy of 97.3%. Moreover, Table 4 shows the accuracy assessment following the proposed method.

6.2.4.1. Classifying direct neighbors. The results show, that in case of the initial transfer in the vertical direction ($5 \rightarrow 2$, $5 \rightarrow 8$) the old model achieve the highest accuracies. We assume that the reason for this fact is the identical acquisition date of all three images. Nevertheless, using the information in the overlapping area only performs worse in terms of accuracy due to the fact that an overlapping area does not necessarily comprise training samples for each class. In that case mis-classified pixels are incorrectly used as new training labels.

Using the difference to the reference classification, the loss in the overall accuracy is 12.1% on average when using the old model, 7.5% when only training with the overlapping area and 2.3% when training with the batch or incremental version of the IVM using old and new data.

6.2.4.2. Classifying indirect neighbors. The results for the chain classification indicate that using the old model is not an adequate strategy for a reliable land cover classification. The average loss in accuracy using the old model is 18.3%, 5.7% when using only training data from the overlapping areas, 1.1% using I^2VM and 1.0% using the batch version.

The highest accuracy can be achieved when the chain contains scenes of the same acquisition date as long as possible. Also, going parallel and perpendicular to the strip direction appears to be better than to go diagonally. The results show, that the difference in the accuracy between IVM and I^2VM is not significant.

7. Conclusion

We introduce a sparse kernel-based discriminative incremental learner called I^2VM, which can deal with the inclusion and removal of new samples and the update of the model parameters without retraining from scratch. We show, that I^2VM inherently have a reconstructive component. This can be explained by the optimization criterion, which primarily aims to optimize the quality of the posterior probability of the classification, rather than optimizing the decision boundary. The selected import vectors cover the acceptance region of their class. We demonstrated, that holding reconstructive and discriminative properties show powerful incremental learning and enable an efficient classification. Our experiments confirm that I^2VM are competitive to their batch version and more efficient in handling concept-drifts than well-known incremental learners. We also presented an approach, which enables the efficient multi-class classification of large neighboring Landsat scenes. We use I^2VM in combination with self-training to update the initial classifier with new training data acquired in the overlapping areas between neighboring Landsat scenes. The experiments showed that keeping data samples from the current scene can be necessary to achieve a good classification result.

Regarding the properties of IVM and I^2VM, further research should enhance focus on a more efficient selection of import vectors. Furthermore, because of the reconstructive component, IVM seem to be a suitable one-class classifier.

References

[1] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 41–48.
[2] C. Bishop, Pattern Recognition and Machine Learning, vol. 4, Springer, New York, 2006.
[3] C. Bodart, H. Eva, R. Beuchle, R. Rasi, D. Simonetti, H. Stibig, A. Brink, E. Lindquist, F. Achard, Pre-processing of a sample of multi-scene and multi-date Landsat imagery used to monitor forest cover changes over the tropics, ISPRS J. Photogramm. Remote Sens. 66 (2011) 555–563.
[4] A. Bordes, L. Bottou, P. Gallinari, J. Weston, Solving MultiClass support vector machines with LaRank, Proc. International Conference on Machine Learning, 2007.
[5] A.C. Braun, U. Weidner, S. Hinz, Support vector machines, import vector machines and relevance vector machines for hyperspectral classification — a comparison, Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing, June 2011, pp. 1–4.
[6] L. Breiman, Bagging Predictors, Machine Learning 24 (2) (1996) 123–140.
[7] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[8] L. Breiman, J. Friedman, R. Olshen, C. Stone, L. Breiman, W. Hoeffding, R. Serfling, J. Friedman, O. Hall, P. Buhlmann, et al., Classification and regression trees, Mach. Learn. 19 (1984) 293–325.
[9] R. Caruana, A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, Proc. International Conference on Machine Learning, ACM, 2006, pp. 161–168.
[10] G. Cauwenberghs, T. Poggio, Incremental and decremental support vector machine learning, Proc. Advances in Neural Information Processing Systems, 2001, pp. 409–415.
[11] G. Cawley, N. Talbot, Efficient model selection for kernel logistic regression, Pattern Recognit. 2 (2004) 439–442.
[12] G. Cawley, N. Talbot, M. Girolami, Sparse multinomial logistic regression via Bayesian L1 regularisation, Proc. Advances in Neural Information Processing Systems, 2007.
[13] C. Chang, C. Lin, LIBSVM: A Library for Support Vector Machines, , 2001.
[14] N. Chawla, D. Cieslak, Evaluating probability estimates from decision trees, American Association for Artificial Intelligence, 2006.
[15] T. Chin, D. Suter, Incremental kernel principal component analysis, IEEE Trans. Image Process. 16 (6) (2007) 1662–1674.
[16] J. Cihlar, Land cover mapping of large areas from satellites: status and research priorities, Int. J. Remote Sens. 21 (6–7) (2000) 1093–1114.
[17] J. Cohen, et al., A coefficient of agreement for nominal scales, Educ. Psychol. Meas. 20 (1) (1960) 37–46.
[18] A. Cornuéjols, Getting order independence in incremental learning, Proc. European Conference on Machien Learning, Springer, 1993, pp. 196–212.
[19] J. Cramer, Logit Models from Economics and Other Fields, Cambridge University Press, 2003.
[20] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 886–893.
[21] T. Deselaers, T. Gass, G. Heigold, H. Ney, Latent log-linear models for handwritten digit classification, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011.
[22] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories, Comput. Vis. Image Underst. 106 (2007) 59–70.
[23] V. Franc, A. Zien, B. Schlkopf, Support vector machines as probabilistic models, Proc. International Conference on Machine Learning, 2011, pp. 665–672.
[24] S. Franklin, M. Wulder, Remote sensing methods in medium spatial resolution satellite data land cover classification of large areas, Progr. Phys. Geogr. 26 (2) (2002) 139–172.
[25] Y. Freund, R. Schapire, Experiments With a New Boosting Algorithm, International Conference on Machine Learning, Morgan Kaufmann Publishers, Inc., 1996, pp. 148–156.
[26] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Ann. Stat. 28 (2000) 337–407.
[27] M. Fritz, B. Leibe, B. Caputo, B. Schiele, Integrating representative and discriminative models for object category detection, Proc. IEEE International Conference on Computer Vision, 2005.
[28] G. Fung, O. Mangasarian, Incremental support vector machine classification, Proc. International Conference on Knowledge Discovery and Data Mining, 2002, pp. 247–260.
[29] F. Giacco, C. Thiel, L. Pugliese, S. Scarpetta, M. Marinaro, Uncertainty analysis for the classification of multispectral satellite images using SVMs and SOMs, IEEE Trans. Geosci. Remote Sens. 99 (2010) 1–11.
[30] M. Genton, Classes of kernels for machine learning: a statistics perspective, The Journal of Machine Learning Research 2 (2002) 299–312.
[31] F. Glover, Tabu Search-Part I, ORSA Journal on Computing 1 (3) (1989) 190–206.
[32] H. Grabner, H. Bischof, On-line boosting and vision, Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2006, pp. 260–267.
[33] H. Grabner, C. Leistner, H. Bischof, Semi-supervised on-line boosting for robust tracking, Proc. IEEE European Conference on Computer Vision, Springer, 2008, pp. 234–247.
[34] Y. Grandvalet, J. Mariéthoz, S. Bengio, Interpretation of SVMS with an application to unbalanced classification, Proc. Advances in Neural Information Processing Systems, 2005.
[35] S. Gu, Y. Zheng, C. Tomasi, S. Gu, Y. Zheng, C. Tomasi, Efficient visual object tracking with online nearest neighbor classifier, Proc. Asian Conference on Computer Vision, 2010.
[36] T. Hastie, R. Tibshirani, J. Friedman, The elements of statistical learning, Springer Series in Statistics, vol. 1, 2009.
[37] N. Higham, Accuracy and Stability of Numerical Algorithms, Society for Industrial Mathematics, 2002.
[38] T. Ho, The Random Subspace Method for Constructing Decision Forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832–844.

[39] J. Iglesias, E. Konukoglu, A. Montillo, Z. Tu, A. Criminisi, Combining generative and discriminative models for semantic segmentation of ct scans via active learning, Proc. Information Processing in Medical Imaging, Springer, 2011, pp. 25–36.
[40] M. Karasuyama, I. Takeuchi, Multiple incremental decremental learning of support vector machines, IEEE Trans. Neural Netw. 21 (7) (2010) 1048–1059.
[41] P. Karsmakers, K. Pelckmans, J. Suykens, Multi-class kernel logistic regression: a fixed-size implementation, Proc. of the International Joint Conference on Neural Networks, 2007, pp. 1756–1761.
[42] S. Keerthi, K. Duan, S. Shevade, A. Poo, A fast dual algorithm for kernel logistic regression, Mach. Learn. 61 (1) (2005) 151–165.
[43] T. Kim, S. Wong, B. Stenger, J. Kittler, R. Cipolla, Incremental linear discriminant analysis using sufficient spanning set approximations, Proc. IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.
[44] J. Kivinen, A. Smola, R. Williamson, Online learning with kernels, IEEE Trans. Signal Process. 52 (8) (2004) 2165–2176.
[45] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, Proc. International Conference on Machine Learning, vol. 11, Morgan Kaufmann, San Francisco, CA, USA, 2000.
[46] J. Knorn, A. Rabe, V. Radeloff, T. Kuemmerle, J. Kozak, P. Hostert, Land cover mapping of large areas using chain classification of neighboring Landsat satellite images, Remote Sens. Environ. 113 (5) (2009) 957–964.
[47] B. Krishnapuram, L. Carin, M. Figueiredo, A. Hartemink, Sparse multinomial logistic regression: fast algorithms and generalization bounds, IEEE Trans. Pattern Anal. Mach. Intell. (2005) 957–968.
[48] M. Kristan, A. Leonardis, Online discriminative kernel density estimation, Proc. IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 581–584.
[49] S. Kumar, M. Hebert, Discriminative random fields, Int. J. Comput. Vis. 68 (2) (2006) 179–201.
[50] P. Langley, Learning in humans and machines, Learning towards an interdisciplinary learning science, Ch. Order Effects in Incremental, vol. 136, Pergamon, 1995, p. 141.
[51] J. Lasserre, C. Bishop, T. Minka, Principled hybrids of generative and discriminative models, Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, IEEE, 2006, pp. 87–94.
[52] J. Li, J. Bioucas-Dias, A. Plaza, Hyperspectral image segmentation using a new Bayesian approach with active learning, IEEE Trans. Geosci. Remote Sens. 49 (99) (2011) 3947–3960.
[53] A. McCallum, C. Pal, G. Druck, X. Wang, Multi-conditional learning: generative/discriminative training for clustering and classification, Proc. of the National Conference on Artificial Intelligence, vol. 21, AAAI Press; MIT Press, Menlo Park, CA; Cambridge, MA; London, 1999, p. 433.
[54] K. McKusick, P. Langley, Constraints on tree structure in concept formation, Proc. International Joint Conference on Artificial Intelligence, vol. 810, 1991, p. 816, Citeseer.
[55] A. Niculescu-Mizil, R. Caruana, Predicting good probabilities with supervised learning, Proc. International Conference on Machine Learning, ACM, 2005, pp. 625–632.
[56] S. Pang, S. Ozawa, N. Kasabov, Incremental linear discriminant analysis for classification of data streams, IEEE Trans. Syst. Man Cybern. 35 (2005) 905–914.
[57] A. Pekkarinen, L. Reithmaier, P. Strobl, Pan-European forest/non-forest mapping with Landsat ETM+ and CORINE Land Cover 2000 data, ISPRS J. Photogramm. Remote Sens. 64 (2) (2009) 171–183.
[58] J. Platt, N. Cristianini, J. Shawe-Taylor, Large margin DAGs for multiclass classification, Proc. Advances in Neural Information Processing Systems, vol. 12, 2000, pp. 547–553, Citeseer.
[59] F. Provost, P. Domingos, Tree induction for probability-based ranking, Mach. Learn. 52 (3) (2003) 199–215.
[60] R. Raina, Y. Shen, A. Ng, A. McCallum, Classification with hybrid generative/discriminative models, Proc. of the International Joint Conference on Neural Networks, vol. 16, 2003.
[61] J. Richards, Analysis of remotely sensed data: the formative decades and the future, IEEE Trans. Geosci. Remote Sens. 43 (3) (2005) 422–432.
[62] R. Roscher, B. Waske, W. Förstner, Incremental import vector machines for large area land cover classification, Proc. IEEE/ISPRS Workshop on Computer Vision for Remote Sensing of the Environment, 2011.
[63] R. Roscher, B. Waske, W. Förstner, Incremental import vector machines for classifying hyperspectral data, IEEE Trans. Geosci. Remote Sens. 9 (2012) 1–11.
[64] V. Roth, Probabilistic discriminative kernel classifiers for multi-class problems, Pattern Recognit. 2191 (2001) 246–253.
[65] S. Rüping, Incremental learning with support vector machines, Proc. IEEE International Conference on Data Mining, IEEE, 2001, pp. 641–642.
[66] S. Rüping, A simple method for estimating conditional probabilities in SVMs, LWA 2004-Lernen-Wissensentdeckung-Adaptivität, Humboldt-Universität, 2004, Citeseer.
[67] A. Saffari, C. Leistner, J. Santner, M. Godec, H. Bischof, On-line random forests, Proc. On-line Learning for Computer Vision Workshop, 2009, pp. 1393–1400.
[68] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, 2002.
[69] M. Scholz, R. Klinkenberg, Boosting classifiers for drifting concepts, Intell. Data Anal. 11 (1) (2007) 3–28.
[70] A.K. Seewald, Digits—A Dataset for Handwritten Digit Recognition Tech. rep., Austrian Research Institute for Artificial Intelligence, 2005 TR-2005-27.
[71] D. Skocaj, M. Uray, A. Leonardis, H. Bischof, Why to combine reconstructive and discriminative information for incremental subspace learning, Proc. Computer Vision Winter Workshop, 2006.
[72] P. Sollich, Bayesian methods for support vector machines: evidence and predictive class probabilities, Mach. Learn. 46 (1) (2002) 21–52.
[73] L. Talaveral, J. Roure, A buffering strategy to avoid ordering effects in clustering, Proc. European Conference on Machine Learning, Springer, 1998, pp. 316–321.
[74] F. Tang, S. Brennan, Q. Zhao, H. Tao, Co-tracking using semi-supervised support vector machines, Proc. IEEE International Conference on Computer Vision, 2007, pp. 1–8, URL , http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?ar number=4408954.
[75] M. Tipping, Sparse Bayesian learning and the relevance vector machine, J. Mach. Learn. Res. 1 (2001) 211–244.
[76] Z. Tu, Learning generative models via discriminative approaches, Proc. IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2007, pp. 1–8.
[77] D. Tuia, F. Ratle, F. Pacifici, M. Kanevski, W. Emery, Active learning methods for remote sensing image classification, IEEE Trans. Geosci. Remote Sens. 47 (7) (2009) 2218–2232.
[78] M. Uray, D. Skocaj, P.M. Roth, H. Bischof, A. Leonardis, Incremental LDA learning by combining reconstructive and discriminative approaches, Proc. British Machine Vision Conference, 2007.
[79] V. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag, 2000.
[80] P. Viola, M. Jones, Robust real-time face detection, Int. J. Comput. Vis. 57 (2) (2004) 137–154.
[81] W. Walker, C. Stickler, J. Kellndorfer, K. Kirsch, D. Nepstad, Large-Area classification and mapping of forest and land cover in the Brazilian Amazon: a comparative analysis of ALOS/PALSAR and Landsat data sources, IEEE J. Sel. Top. Appl. Earth Observations Remote Sens. 3 (4) (2010) 594–604.
[82] S. Wenzel, L. Hotz, The role of sequences for incremental learning, in: J. Filipe, A.L.N. Fred, B. Sharp (Eds.), Proc. International Conference on Agents and Artificial Intelligence, vol. 1, INSTICC Press, Valencia, Spain, January 22–24 2010, pp. 434–439.
[83] J. Xue, D. Titterington, On the generative–discriminative tradeoff approach: interpretation, asymptotic efficiency and classification performance, Comput. Stat. Data Anal. 54 (2) (2010) 438–451.
[84] K. Yoshii, M. Goto, K. Komatani, T. Ogata, H. Okuno, An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model, IEEE Trans. Audio Speech Lang. Process. 16 (2008) 435–447.
[85] B. Zadrozny, C. Elkan, Learning and making decisions when costs and probabilities are both unknown, Proc. International Conference on Knowledge Discovery and Data Mining, ACM, 2001, pp. 204–213.
[86] Z. Zhang, G. Dai, M. Jordan, Bayesian generalized kernel mixed models, J. Mach. Learn. Res. 12 (2011) 111–139.
[87] J. Zheng, H. Yu, F. Shen, J. Zhao, An online incremental learning support vector machine for large-scale data, Proc. International Conference on Artificial Neural Networks, Springer, 2010, pp. 76–81.
[88] J. Zhu, T. Hastie, Kernel logistic regression and the import vector machine, Comput. Graph. Stat. 14 (2005) 185–205.