

# Structured prediction by joint kernel support estimation

Christoph H. Lampert · Matthew B. Blaschko

Received: 18 July 2008 / Accepted: 20 March 2009  
The Author(s) 2009. This article is published with open access at Springerlink.com

**Abstract** Discriminative techniques, such as conditional random fields (CRFs) or structure aware maximum-margin techniques (maximum margin Markov networks ( $M^3N$ ), structured output support vector machines (S-SVM)), are state-of-the-art in the prediction of structured data. However, to achieve good results these techniques require complete and reliable ground truth, which is not always available in realistic problems. Furthermore, training either CRFs or margin-based techniques is computationally costly, because the runtime of current training methods depends not only on the size of the training set but also on properties of the output space to which the training samples are assigned.

We propose an alternative model for structured output prediction, *Joint Kernel Support Estimation (JKSE)*, which is rather generative in nature as it relies on estimating the joint probability density of samples and labels in the training set. This makes it tolerant against incomplete or incorrect labels and also opens the possibility of learning in situations where more than one output label can be considered correct.

At the same time, we avoid typical problems of generative models as we do not attempt to learn the full joint probability distribution, but we model only its support in a joint reproducing kernel Hilbert space. As a consequence, JKSE training is possible by an adaption of the classical one-class SVM procedure. The resulting optimization problem is convex and efficiently solvable even with tens of thousands of training examples. A particular advantage of JKSE is that the training speed depends only on the size of the training set, and not on the total size of the label space. No inference step during training is required (as  $M^3N$  and S-SVM would) nor do we have calculate a partition function (as CRFs do).

Experiments on realistic data show that, for suitable kernel functions, our method works efficiently and robustly in situations that discriminative techniques have problems with or that are computationally infeasible for them.

---

C.H. Lampert (✉) · M.B. Blaschko  
Max Planck Institute for Biological Cybernetics, Tübingen, Germany  
e-mail: [chl@tuebingen.mpg.de](mailto:chl@tuebingen.mpg.de)

M.B. Blaschko  
Department of Engineering Science, University of Oxford, Oxford, UK

**Keywords** Structured prediction · Generative model · Support estimation · Reproducing kernel Hilbert space · Joint kernel function · One-class support vector machine

## 1 Introduction

Structured Prediction deals with the task of learning functions  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , where  $\mathcal{Y}$  is not just a simple boolean or discrete variable, but can have a rich substructure. Typical examples are graph-labeling problems, where for each node in a graph one label has to be selected (e.g. *natural language parsing* or *image segmentation*), or graph-prediction problems, where it has to be decided which nodes in a graph to connect with an edge (e.g. *sequence alignment*, *RNA secondary structure prediction*). Other possible scenarios include hierarchical classification and multi-dimensional regression with strong correlations between the output dimensions. In all these examples, the dependencies between the output units suggest that in each case it makes more sense to solve a single task of predicting structured outputs rather than a collection of independent classification problems. The fact that new examples of applications are continually found shows that the potential of structured prediction techniques is by far not yet fully explored.

While generative techniques to predict sequence structures have been known and used for several decades, it was in particular recent discriminative techniques that have given the field fresh momentum. Conditional random fields (CRFs, Lafferty et al. 2001, KCRFs, Lafferty et al. 2004) and structured maximum margin techniques (maximum margin Markov networks (M<sup>3</sup>N) (Taskar et al. 2003), structured output support vector machines (S-SVM) Tsochantaridis et al. 2005) have established themselves as powerful tools that are not only theoretically well founded in statistical learning theory, but that also achieve state-of-the-art performance. However, the orientation towards discriminative methods did not come for free: to apply these techniques to real-world problems one requires good knowledge about the training data, in particular cleanly labeled training sets. Realistic datasets often contain significant amounts of incorrect or ambiguous labels and researchers in several areas have studied this problem in the setup of structured prediction, e.g. (Har-Peled et al. 2007; Parker et al. 2007). It is our experience that moderate amounts of incomplete or inconsistent labels can already have a significantly negative effect on the performance of learned systems. This is in particular the case for M<sup>3</sup>N and S-SVM, which are trained by enforcing a margin between a sample's label as given in the training set and all other labels in the (typically exponentially large) label space. We will show experimental results for this observation in Sect. 5.

A further drawback, one that CRFs and margin-based techniques have in common, is that both methods are trained iteratively, which often is computationally very costly. Except for simple output structures, solving truly large scale problems is infeasible without making some strong simplifying assumptions.

In the following, we will explain how many of the problems mentioned can be avoided by borrowing from the theory of generative models. However, we also evade the typical problems of generative models, such as the limited expressive power of parametric models, or the curse of dimensionality and the danger of overfitting for non-parametric ones. The crucial idea of our method is to work with the joint-probability density of samples and labels, but instead of fully modelling the density itself, we only estimate its support using a reproducing kernel Hilbert space (RKHS) formulation (Vapnik 1998). We call the resulting technique for structure prediction *joint-kernel support estimation* (JKSE). A positive side effect of the recourse to kernel techniques is that JKSE can be formulated as a convex optimization problem. Its globally optimal solution can be found very efficiently by adapting

the classical one-class SVM (OC-SVM) procedure (Schölkopf et al. 2001). Thus, JKSE can even be applied to datasets with tens of thousands of examples or more, orders of magnitude larger than what many other techniques for structured output prediction can handle.

## 2 Joint kernel support estimation

We follow the common language of the field (Bakır et al. 2007) and treat structured prediction in probabilistic terms as a MAP-prediction problem. Let  $\mathcal{X}$  be the space of observations and  $\mathcal{Y}$  be the space of possible labels. Note that for our setup it is not required that  $\mathcal{X}$  or  $\mathcal{Y}$  decompose into smaller entities like nodes or edges in a graph. We assume that sample-label pairs  $(x, y)$  follow a joint-probability density  $p(x, y)$ , and that a set of i.i.d. samples  $(x_i, y_i)$  for  $i = 1, \dots, n$  is available for training. The task is to learn a mapping  $f : \mathcal{X} \rightarrow \mathcal{Y}$  that minimizes the expected loss in a classification sense, i.e. for a new sample  $x \in \mathcal{X}$ , we have to determine the label  $y \in \mathcal{Y}$  that maximizes the posterior probability  $p(y|x)$ .

It is well known that, mathematically, the discriminative approach of directly modelling  $p(y|x)$  and the generative approach of modelling  $p(x, y)$  are equivalent, because Bayes rule allows one to use either quantity for optimal prediction:

$$\operatorname{argmax}_{y \in \mathcal{Y}} p(y|x) = \operatorname{argmax}_{y \in \mathcal{Y}} p(x, y) \quad \text{for any } x \in \mathcal{X}. \quad (1)$$

In joint-kernel support estimation we follow the generative path and model an expression for  $p(x, y)$  from the given training data. However, density estimation in high dimensional spaces is notoriously difficult. We therefore simplify the problem by assuming that the posterior probability in the feature space is *distinctive*, i.e.  $p(y|x) \gg 0$  for correct predictions  $y$  and  $p(y|x) \approx 0$  for incorrect  $y$ . Consequently,  $p(x, y) \gg 0$  only if  $y$  is a correct label for  $x$ , and it suffices to estimate the *support* of  $p(x, y)$  instead of the full density. Afterwards, we can still use  $f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} p(x, y)$  for prediction. Note that we do not require  $p(y|x)$  to be unimodal. Different  $y \in \mathcal{Y}$  could be “correct” predictions for  $x \in \mathcal{X}$ , as occurs quite commonly in realistic structured prediction tasks.

The generative setup limits JKSE’s ability to *extrapolate*, because  $p(x, y) = p(y|x)p(x)$  and for a test sample  $x \in \mathcal{X}$  this vanishes not only for wrong predictions  $y$  but also if  $x$  lies in an area of  $\mathcal{X}$  that has not been observed during training. On the other hand, the setup also has certain advantages, e.g., it opens the possibility to perform adaptive and online learning, building a smaller model of  $p(x, y)$  first and later extending the label set without having to retrain for the previous labels.

### 2.1 Representation

Like most other structured prediction methods, we model probabilities by log-linear models (Bakır et al. 2007). Since our central quantity of interest is  $p(x, y)$ , we set

$$p(x, y) \equiv \frac{1}{Z} \exp(w^t \phi(x, y)) \quad (2)$$

where  $Z \equiv \sum_{x, y} \exp(w^t \phi(x, y))$  is a normalization constant (the *partition function*). Since our model is generative,  $Z$  does not depend on  $y$  and we can essentially ignore it during training as well as during inference. Consequently, the prediction step reduces to

$$f(x) \equiv \operatorname{argmax}_{y \in \mathcal{Y}} w^t \phi(x, y). \quad (3)$$

In the following, we will assume access to some form of inference algorithm for calculating the argmax in (3) or a suitable approximation to it. We do not require a method to calculate or approximate  $Z$  at any time.

The feature map  $\phi(x, y)$  in (2) is completely generic. While in many situations, such as sequence labeling or image segmentation, it is natural to form  $\phi(x, y)$  by a concatenation or summation of per-site properties and neighborhood features, we do not require such a decomposition for our consideration. In fact,  $\phi(x, y)$  does not have to be an explicit mapping at all, and in the following we will only study the case where it is induced by a suitable positive definite joint kernel function  $k : (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \rightarrow \mathbb{R}$ . The case of an explicitly known  $\phi(x, y)$  can be included in this framework by setting  $k((x, y), (x', y')) \equiv \phi(x, y)^t \phi(x', y')$ .

## 2.2 Parameter learning

Assuming a fixed kernel or feature map, learning consists only of finding a suitable weight vector  $w$  such that the right hand side of (2) reflects  $p(x, y)$  over the training set. Since we are only interested in the support of  $p(x, y)$ , we can use a one-class support vector machine (OC-SVM) for this purpose, see Sect. 2.3 for a review of this technique. The result of OC-SVM training is a representation of  $p(x, y)$  as a linear combination of kernel evaluations with the training samples. Thus, the JKSE prediction function can be written as

$$f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \sum_{i=1}^n \alpha_i k((x, y), (x_i, y_i)). \quad (4)$$

Note that this expansion is generally sparse, i.e. most  $\alpha_i$  have the value 0.

In its training procedure, JKSE reduces the basically generative task of learning  $p(x, y)$  to a maximum margin learning problem of estimating the support of  $p(x, y)$ .

The JKSE training procedure works for arbitrary Mercer kernels (Vapnik 1998) and the resulting optimization problem is convex. Furthermore, only the matrix of joint-kernel values is required, and thus the training time depends only on the size of the training set, not on the structure of the output space. This is in contrast to many other techniques for structured prediction, see Sect. 3 for details.

## 2.3 One-class SVM training

The one-class support vector machine (OC-SVM) was originally introduced to robustly estimate the support and quantiles of probability densities in high dimensional spaces (Schölkopf et al. 2001). In the case of JKSE, we replace the original samples by sample-label pairs and consider them as embedded into the latent Hilbert space  $\mathcal{H}$  that is induced by the joint kernel function  $k((x, y), (x', y'))$ .

The OC-SVM works by estimating a hyperplane in  $\mathcal{H}$  that best separates the training samples from the origin, except for a set of *outliers* which are determined implicitly by the training procedure. A parameter  $\nu \in (0, 1]$  acts as an upper bound to the percentage of outliers, i.e. the larger  $\nu$ , the more freedom the method has to disregard any of the training samples. By this,  $\nu$  simultaneously acts as a regularization parameter.<sup>1</sup> JKSE training using

<sup>1</sup>For support estimation, it is generally more intuitive to study the problem of finding the ball of smallest radius in the feature space that encloses all training points except for the outliers. Both concepts are in fact equivalent for the common class of kernels function where every sample has the same length in feature space, e.g. Gaussian kernels and generalizations (see Martinus and Tax 2001).

OC-SVM can be written in primal form as the quadratic optimization problem

$$\min_{w \in \mathcal{H}, \xi_i \in \mathbb{R}^+, \rho \in \mathbb{R}} \frac{1}{2} \|w\|^2 + \frac{1}{vn} \sum_i \xi_i - \rho \tag{5}$$

subject to

$$\langle w, \phi(x_i, y_i) \rangle_{\mathcal{H}} \geq \rho - \xi_i \quad \text{for } i = 1, \dots, n, \tag{6}$$

where  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  denotes the scalar product in the Hilbert space  $\mathcal{H}$  and  $\rho$  is the offset of the hyperplane from the origin. To actually solve the minimization (5), one applies the representer theorem (Schölkopf and Smola 2002). Consequently, all references to  $\langle \cdot, \cdot \rangle_{\mathcal{H}}$  and  $\phi$  disappear and only kernel evaluations are required. We solve the dual problem:

$$\min_{\alpha} \sum_{ij} \alpha_i \alpha_j k((x_i, y_i), (x_j, y_j)) \tag{7}$$

subject to

$$0 \leq \alpha_i \leq \frac{1}{vn}, \quad \sum_i \alpha_i = 1 \quad \text{for } i = 1, \dots, n. \tag{8}$$

This allows us to reuse existing OC-SVM implementations for JKSE learning: we provide the algorithm with the joint kernel matrix between pairs  $(x_i, y_i)$  instead of the ordinary kernel matrix measuring similarity between samples  $x_i$ . OC-SVM training will then learn coefficients,  $\alpha_i$ , that can directly be used in the context of (4).

Note that in the training procedure, only comparisons between sample pairs from the training set are required. For a training set of size  $n$ , we have to solve a quadratic objective function with  $n^2$  terms for  $n$  unknowns, subject to  $2n + 1$  linear constraints. At no time do we have to evaluate a function over the space of all possible target labels, which would make learning dependent not only on the size of the training set, but also on the label space.

The many theoretical studies of OC-SVMs in the machine learning literature immediately carry over to the training of JKSE. One interesting aspect of this is that—for suitable kernel functions—one can prove consistency results for the approximation of  $p(x, y)$ , see (Vert and Vert 2005). Note that because of the additional argmax operation, this, however, does not imply consistency of JKSE’s prediction step.

### 3 Comparison to other approaches for structured prediction

Joint-kernel support estimation is a new technique in how it merges generative ideas with density estimation using reproducing kernel Hilbert spaces. However, its target application of structured prediction has a long history of successful techniques. In the following we discuss the relationship to some of these.

#### 3.1 Discriminative models

The currently most successful techniques for structured prediction are discriminative models that—like JKSE—are derived from a framework of statistical learning. Discriminative models have the *posterior probability*  $p(y|x)$  as their central quantity of interest. Given a

suitable training set, they learn a model for  $p(y|x)$  and perform prediction for a new sample  $x$  by MAP-estimation

$$f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} p(y|x). \quad (9)$$

The distinction between different learning techniques lies in how they model the posterior, how the model is adapted to the data and how the MAP-prediction is actually performed.

Discriminative models have strong performance guarantees in the limit of infinite training data. Furthermore, the predominant opinion is that they also work better than generative techniques when only limited training data is available, see e.g. (Jaakkola and Haussler 1998; Pernkopf and Bilmes 2005; Vapnik 1998).<sup>2</sup> One intuitive explanation for this is the fact that the posterior  $p(y|x)$  is generally easier to model than the joint  $p(x, y)$ , as it has a simpler shape, often being nearly constant over large areas of the feature space. However, this intuition is clearly motivated by the situation of binary classification, where  $y$  can take only two values. For structured prediction with a huge label space  $\mathcal{Y}$ , not much is known *a priori* about the shape of  $p(y|x)$ .

In the following, we will compare JKSE to the most popular discriminative techniques for structured prediction: conditional random fields and structured maximum-margin techniques.

### 3.1.1 Conditional random fields (CRFs)

CRFs model  $p(y|x)$  by a log-linear model:

$$p(y|x) \equiv \frac{1}{Z(x)} \exp(w^t \phi(x, y)) \quad (10)$$

for a parameter vector  $w$  and a feature function  $\phi(x, y)$ . Note that the partition function  $Z(x) = \sum_{y \in \mathcal{Y}} \exp(w^t \phi(x, y))$  depends on  $x$  since  $p(y|x)$  is normalized only with respect to  $y$ . For MAP-prediction with fixed  $w$ ,  $Z(x)$  can be ignored, but this is not the case during training.

CRFs are typically trained by maximizing the conditional log likelihood of the training data:

$$\max_w \sum_{i=1}^n w^t \phi(x_i, y_i) - \log Z(x_i) \quad (11)$$

where  $(x_i, y_i), i = 1, \dots, n$  is the training set. Additionally, one can add regularization terms, which we omit as they are not relevant to our discussion.

The problem (11) is a *concave* optimization and can be solved by gradient ascent in the parameter vector  $w$ . However, the calculation of the gradient requires knowledge of  $Z(x)$ . Except for a few special output structures, e.g. labeling of tree structures, this is infeasible to calculate exactly. Therefore, approximate inference tools like *loopy belief propagation (LBP)* (Murphy et al. 1999) are employed in most other situations and the expression (11) is solved only approximately.

JKSE differs from CRFs in several fundamental points: JKSE as a generative technique models  $p(x, y)$  instead of  $p(y|x)$ . This makes its partition function  $Z$  independent not only

<sup>2</sup>However, this belief is not unchallenged, see e.g. (Ng and Jordan 2003).

of  $y$  but also of  $x$ , and its value is needed neither in training nor in prediction. Since training of JKSE is done using Hilbert space techniques and not by gradient ascent, we can use feature maps  $\phi(x, y)$  that are only given implicitly by a suitable Mercer kernel. This is a potential source of improved performance in terms of speed as well of prediction accuracy. Even though kernelizations of CRFs exist (see Lafferty et al. 2004), they are generally difficult to construct and train and have not been adopted as enthusiastically as regular CRFs.

### 3.1.2 Structured support vector machines (S-SVMs) and maximum margin Markov networks ( $M^3N$ )

S-SVMs and  $M^3N$ s share most of their fundamental properties. In the following, we will therefore only discuss S-SVMs, even though the comparison applies to both concepts.

Like a CRF, an S-SVM models the posterior  $p(y|x)$  by a log-linear model (10) and performs MAP-prediction. However, it implements a different training procedure: given training samples  $(x_i, y_i), i = 1, \dots, n$ , an S-SVM performs maximum-margin training: for each training sample  $x_i$ , it assumes that all outputs  $y$  that differ from the specified label  $y_i$  in the training set are wrong labelings. It then tries to learn a weight vector that predicts the correct output labels for the training samples with a large margin compared to the wrong output labels, i.e. to all other possible outputs. The resulting optimization problem can be written as

$$\min_{w, \xi_i \geq 0} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \tag{12}$$

subject to

$$\Delta(y, y_i) + w^t \phi(x_i, y) - w^t \phi(x_i, y_i) \leq \xi_i, \quad \text{for } i = 1, \dots, n \text{ and for all } y \neq y_i, \tag{13}$$

where  $\Delta(y, y')$  is a problem specific loss function and we have adopted the *margin-rescaling* formulation (Tsochantaridis et al. 2005). The alternative *slack-rescaling*, which is in some sense more intuitive, leads to a more difficult optimization problem and has therefore not been as widely adopted (Sarawagi and Gupta 2008).

S-SVM and JKSE share the property of easy kernelization. The vector notation  $w^t \phi(x_i, y_i)$  then turns into the implicitly defined scalar product  $\langle w, \phi(x_i, y_i) \rangle_{\mathcal{H}}$  as in Sect. 2.3. The main difference between the methods, however, is the way the labels are used during training, which is again a consequence of the fact that JKSE models  $p(x, y)$  and not  $p(y|x)$ . The S-SVM enforces a margin of  $\Delta(y, y_i)$  between the ground truth label  $y_i$  and all other possible labels  $y \in \mathcal{Y}$ . This is a very powerful concept, if the label  $y_i$  in the training is in fact the unique correct label for  $x_i$ . However, in the presence of label errors, the margin concept will have a doubly negative effect: the vector  $w$  will be encouraged to predict the wrong  $y_i$ , and it will be actively discouraged from predicting anything else, which includes the correct label  $y'$ . Often a good setting of the regularization parameter  $C$  can compensate for this effect, but only up to a certain amount of label noise.

In contrast, JKSE only learns the co-occurrence of the pairs  $(x_i, y_i)$  in the dataset. A wrong label  $y_i$  still has a negative influence, as it increases the density of  $p(x, y)$  in an area of the space where prediction should not occur. However, the density at the position of the correct label  $p(x_i, y')$  is not penalized and  $y'$  might still be predicted if it lies in a high density region. This robustness is supported further by the fact that we make use of the  $\nu$ -mechanism during training (Sect. 2.3). Intuitively, it allows JKSE to ignore a percentage of training points, namely the ones that are “least representative” of the joint distribution. Typically, falsely labeled samples can be expected to behave as such outliers.

A similar effect is observable when samples can have multiple correct outputs. Then the S-SVM tries to enforce a margin between two correct labelings, whereas JKSE makes use of both correct labels in a positive way.<sup>3</sup>

From a practical point, JKSE differs from S-SVM immensely in training speed and scalability. To solve the minimization (12), typically a cutting plane algorithm is employed that requires iterative identification of the most violated one of the constraints (13). Each such step requires inference over the model, i.e. the calculation of an argmax over the label space. Because of the multiple inference steps, S-SVM training is often very slow for interesting problems. Training with more than a few hundred examples becomes computationally infeasible, again except for some special output structures. If exact inference is not possible at all, and only approximate techniques are available, S-SVMs have the additional problem of *error accumulation* during training, because inaccurate inference steps must be applied iteratively. This causes most known theoretic guarantees for S-SVMs to break down and little is known about the theoretical properties of approximate margin-based learning for structured prediction. Even empirical studies on the topic are sparse (but see Finley and Joachims 2008; Kulesza and Pereira 2007).

In contrast, JKSE's training objective function is much easier, and its global optimum can be found reliably and quickly, no matter if MAP-inference is possible exactly or only approximately. Training can be performed very efficiently for thousands of samples, as it equivalent to the training of an ordinary one-class SVM. In Sect. 4 we will show that this can even be extended further by rewriting the one-class problem as a task of binary classification.

### 3.1.3 Other discriminative models

Several other methods have been proposed that are closely related to JKSE. Szedmak et al. have formulated a multi-output problem as the problem of learning a linear operator between input and output Hilbert spaces (Szdemak et al. 2005). The resulting problem is similar to the OC-SVM algorithm, but has some differences. Most importantly, by restricting the interpretation of the algorithm to learning a linear operator between the spaces, they are restricted to joint-kernels that are tensor products, i.e.  $k((x_i, y_i), (x_j, y_j)) = k_X(x_i, x_j)k_Y(y_i, y_j)$ . Yang et al. proposed to use the one-class SVM algorithm for multi-class classification (Yang et al. 2007). They did not extend this to general joint kernels, however.

## 3.2 Generative models

Generative models have a long tradition of successful structured prediction (Baum and Petrie 1966; Geman and Geman 1984; Mumford and Shah 1988; Rabiner 1989). They have been most successful in areas where an underlying physical interpretation gives an intuitive understanding of the observed data, such as *speech recognition* and *image processing*. The knowledge about the process that generated the data allows the construction of class-conditional probability densities  $p(x|y)$ . Given an additional prior distribution  $p(y)$ , MAP-prediction can be performed using Bayes rule as

$$f(x) = \operatorname{argmax}_y p(x|y)p(y). \quad (14)$$

<sup>3</sup>S-SVMs can cope with multiple output labels in the training set by a suitable choice of  $\Delta(y, y_i)$ . However, this only resolves the problem if the training set is completely labeled with all such correct outputs. Unfortunately, such datasets are rarely available for common ambiguous structured prediction task such as functional prediction of genes or machine translation.

The use of generative models for classification of non-physical processes has often been criticized, as  $p(x|y)$  and  $p(x, y)$  both contain information about the distribution of samples that might be difficult to model from data, and—strictly speaking—is not necessary to know for classification. However, if a generating process is in fact known, generative models often have advantages over discriminative approaches, as they are better able to deal e.g. with missing, ambiguous or simply wrong labels.

In the following, we compare JKSE to two popular generative approaches, *hidden Markov models (HMMs)* (Rabiner 1989) and *Markov random fields (MRFs)* (Kindermann and Snell 1980).

### 3.2.1 Hidden Markov models (HMMs)

HMMs are directed graphical models for the prediction of per-node labels  $y = (y^1, \dots, y^K) \in \mathcal{Y}$  from an observed sequence  $x = (x^1, \dots, x^K) \in \mathcal{X}$ . Typically, the spaces  $\mathcal{X}$  and  $\mathcal{Y}$  are too high dimensional to directly construct a reliable class-conditional density  $p(x|y)$  and a prior  $p(y)$ . One therefore has to make strong independence assumptions in order to avoid the curse of dimensionality. Typically, one models the appearance of a sequence element  $x^i$  to depend only on its per-node label  $y^i$ . Additionally, one assumes that the prior  $p(y)$  has a Markov property along the sequence, i.e.  $y^i$  depends only on  $y^{i-1}$ . The advantage of HMMs is their intuitive physical interpretation and very fast training.<sup>4</sup> In the described setup, a closed form solution for training by maximum likelihood exists. However, the strong independence assumptions and the restriction to a directed sequence make HMMs too inflexible for complicated structured prediction tasks with long-range interactions. One method to overcome some of these limitations is to discriminatively learn the structure of a directed graphical model (Bilmes 1999).

In contrast, JKSE does not split the joint probability density  $p(x, y)$  into class conditional density and prior, but it models it directly from the training data. JKSE simplifies the problem not by imposing a linear structure with strong independence on the spaces  $\mathcal{X}$  and  $\mathcal{Y}$ , but by concentrating on the *most relevant aspect* of  $p(x, y)$ , namely its support. The curse of dimensionality is avoided by use of Hilbert space techniques that allow efficient density estimation even in high-dimensional feature spaces.

### 3.2.2 Markov random fields (MRFs)

MRFs share the basic property of HMMs that they estimate  $p(x|y)$  and  $p(y)$  separately and perform MAP-estimation by Bayes' rule. However, MRFs are undirected models and can handle other structures than sequences, typically  $2D$  or  $3D$  grids. Again, to avoid the curse of dimensionality, the observed appearance of a node  $x^i$  is assumed to depend only on its label  $y^i$ , but it is possible to model stronger priors by assuming that a label  $y^i$  depends on several other site labels, e.g. all grid neighbors. It is also common to model some sparse long-range dependencies to improve prediction performance.

Except for binary labels and a restricted set of parameterizations for  $p(x^i|y^i)$ , exact MRF training and inference is NP-hard, and one uses approximate training like LBP to find approximate solutions. Nevertheless, MRFs achieve good results, e.g. in tasks such as image denoising or segmentation, where a good prior  $p(y)$  can be constructed based on natural image statistics.

<sup>4</sup>This holds only in the supervised training scenario that we are interested in. Unsupervised and semi-supervised HMM training are difficult problems and fields of active research.

Like HMMs, MRFs require strong independence assumptions in order to make the model feasible. Consequently, a good prior is crucial for MRF performance. JKSE does not require any prior distribution or independence assumptions for training and is therefore generically applicable. However, if domain knowledge is available, it can be included in the kernel function in order to improve the prediction performance.

## 4 Large scale training

Disregarding the time to calculate the kernel matrix, training JKSE is identical to training an *OC-SVM*. In principle this requires the same computational effort as training an ordinary two-class SVM and one should therefore expect that both methods can be applied to problem of similar size and complexity. However, because OC-SVMs are less popular for pattern recognition tasks, significantly less effort has been spent on developing fast training routines and on optimizing the implementations. Existing packages such as `libSVM` can handle thousands of examples, but not tens of thousands. In the following, we therefore discuss possibilities to implement JKSE independently of the existing OC-SVM packages: by reformulating it as a *binary classification problem* and employing fast *stochastic online training techniques*.

### 4.1 One-class classification by binary classification

In their original analysis, Schölkopf et al. showed that, for datasets that can be linearly separated from the origin and *in the case without slack variables*, the weight vector found by optimizing the OC-SVM problem (5) is equivalent to solving the optimization problem of a regular support vector machine for binary classification (Schölkopf et al. 2001; Bertsekas 1995). For their analysis, additional constraints are that there be only positive training examples, and that the hyperplane must pass through the origin. Alternatively, one can allow arbitrary hyperplanes, but add a mirrored copy of the training set with a negative training label. This will learn the same weight vector as the previous construction and the hyperplane in a symmetric problem automatically passes through the origin.

Linear separability can always be enforced by the right choice of kernel, e.g. any kernel with non-negative values. When generalizing the result to the case of slack variables, one and two-class training are still equivalent in the sense that for each  $\nu$ , a corresponding regularization parameter  $C$  exists that results in the same hyperplane. However, the relationship between  $\nu$  and  $C$  becomes non-explicit (Schölkopf et al. 2001; Schölkopf and Smola 2002).

However, we are not interested in equivalence for a specific  $\nu$ , but intend to perform model selection over this parameter anyway. We can therefore make use of the equivalence result and train a two-class SVM with model selection over  $C$  instead.

### 4.2 Stochastic online training

With the availability of larger and larger data collections, machine learning research has focussed increasingly on the creation of methods that not only achieve high prediction accuracy, but that can also be trained efficiently on large datasets, see e.g. (Bottou et al. 2007). As a result, several fast learning algorithms for support vector machines have been developed, many of them limited to linear kernels, e.g. (Hsieh et al. 2008; Joachims 2006; Lin et al. 2008; Shalev-Shwartz et al. 2007), but some also applicable to arbitrary Mercer

kernels (Bordes et al. 2005). Typically, these methods rely on ideas from online learning, such as *stochastic gradient descent* (see e.g. Benveniste et al. 1990). Using such approximate large scale SVM learners in combination with the reformulation of OC-SVM as a regular SVM, we can train JKSE with a dataset of tens of thousands of examples or more. For linear kernels, even millions of examples are within reach.

## 5 Experimental evaluation

We evaluate the performance of JKSE on two realistic *computer vision* tasks: the *localization of objects in natural images* and *optical character recognition*. In the first setup, JKSE performs very well compared to a structured regression method based on S-SVM that has been shown to achieve state-of-the-art performance for similar object localization tasks (Blaschko and Lampert 2008). This allows us to demonstrate the two major claims we made: robustness of JKSE against high amounts of label noise, and the computational efficiencies of training without iterated inference. In the second task, JKSE fails to achieve state-of-the-art performance. By analyzing the reasons for this, we are able to illustrate the importance of the assumptions that JKSE relies on.

### 5.1 Experiments: object localization in images

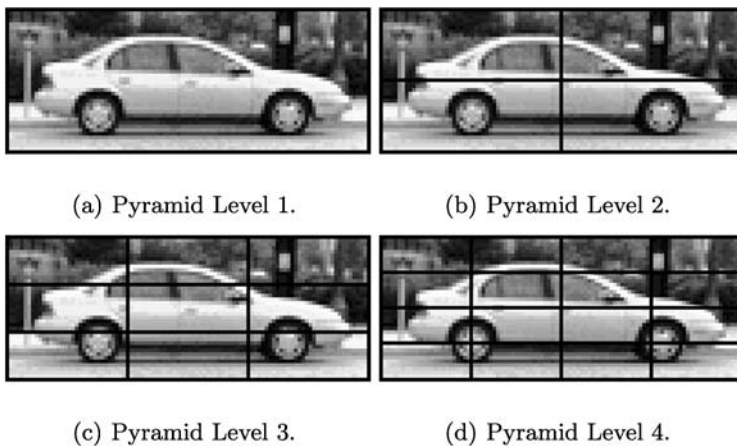
We adopt the setup from (Blaschko and Lampert 2008) to perform object localization by structured prediction: the observations are natural images, and the labels are the coordinates of the bounding box of an object. If an image contains more than one object, any of their bounding boxes is considered a correct label. For the dataset we use the UIUCcars set,<sup>5</sup> choosing the *multiscale* part for training and the *singlescale* part for testing. This leaves us with 108 images showing 139 cars for training, which is close to the upper limit that the S-SVM in this situation can handle in reasonable time. The test set consists of 170 images containing 200 cars. Example images of the dataset are shown in Fig. 1.

An additional part of the dataset consists of 1050 smaller images which were pre-cropped to show either a car or background region. This makes them useless for the task of object localization, but as we will see later, we can make use of them for fast model selection. All images are represented by densely sampled SURF image descriptors (Bay et al. 2006), which are quantized into 1000 visual word clusters, see (Lampert et al. 2008) for details. As a joint-kernel function we choose the *localization kernel* from (Blaschko and Lampert 2008): given two sample-label pairs  $(x, y)$  and  $(x', y')$ , it forms a 4-level spatial pyramid



**Fig. 1** Examples images of the UIUCcars dataset for object localization. The task is to predict tight bounding boxes for the car objects. Images are of different sizes and can contain more than one car, i.e. more than one output label can be correct

<sup>5</sup><http://l2r.cs.uiuc.edu/~cogcomp/Data/Car/>.



**Fig. 2** An illustration of the spatial pyramid features used in the object localization experiments. At the first level of the pyramid (a) all visual words within the object region are collected into a single histogram, while at pyramid level four (d) visual words are collected into 16 different histograms depending on which spatial bin they are located

bag-of-words histograms of those feature points within  $x$  and  $x'$  that fall into the box regions  $y$  and  $y'$  respectively (Fig. 2). The resulting histograms are combined into a kernel value by either a linear scalar product or a  $\chi^2$ -kernel. The former has the advantage that very fast MAP-inference is possible using an integral-image trick. This makes exact S-SVM training feasible. The latter is generally accepted as a better kernel for computer vision tasks, but MAP-inference has to be done by exhaustively scanning over all image locations and is therefore computationally very costly.

We are interested in the performance of JKSE and S-SVM for training sets with different amounts of label noise. To simulate this, we artificially introduce label errors into the dataset by swapping bounding box coordinates between different training images. While this preserves the overall label statistics, the image contents at the positions given by the swapped labels will not necessarily show cars and therefore obstruct the learning process. The percentage of swapped labels is a free parameter,  $r$ , that we vary between 0% (perfect labels) and 100% (random labels).

### 5.1.1 Model selection

Training JKSE in the situation described takes only a few seconds. Evaluation takes also in the order of seconds for the linear kernel function, whereas for the  $\chi^2$  kernel it takes a several minutes per image. This is because within each image there are tens of thousands of possible object locations, and for each, a high-dimensional non-sparse histogram has to be formed and the classifier evaluated.

The S-SVM has identical evaluation time, as it solves the same inference problem. However, training requires iterative solution of the MAP estimate, each corresponding to a full evaluation of the prediction function over many images. This procedure is only feasible for the linear kernel, and even with the fast integral image trick, the total training duration was approximately 5 hours. Within this time, on average close to 3,500 calls to the MAP-estimate were performed accounting for 97% of total training time.

For methods with very long training time, as the S-SVM in our case, *model selection* is always a difficult issue. Except in special cases, it is not practical to perform full cross-validation runs even for a single value like the regularization parameter  $C$ . We therefore rely on a simplified criterion: we train S-SVM using values  $C \in \{10^{-3}, 10^{-2}, \dots, 10^2\}$ . For testing, we use the weight vector of the value that achieves the highest area under curve when used as a classifier on the set of small additional images that we left out during training because they were pre-cropped. In order to facilitate comparability, we follow the same procedure for JKSE to select  $\nu \in \{0.05, 0.1, \dots, 1.0\}$ . Note, however, that JKSE is in fact fast enough to perform full cross-validation, and this could be expected to improve the localization performance to a certain extent.

## 5.2 Results

The localization performance of S-SVM and JKSE are measured by precision-recall curves which are depicted in Figs. 3 and 4. The former shows the results of S-SVM and JKSE with linear kernels. As one can see, S-SVM achieves higher precision and recall than JKSE for noiseless data as well as when 10% and 30% of labels are scrambled. One can assume that it is the S-SVM's Tikhonov regularization that successfully compensates the disturbance introduced by the label errors. However, when the label error rate reaches 50% or more, S-SVM performance takes a huge dive, and at 90% label errors, performance is basically random. A notable anomaly is that the 50%-curve lies below the 70%-curve. As there is no fundamental reason for this, we believe it to be an artifact of the model selection procedure. In fact, S-SVM in this setup has proven rather sensitive to a good choice of  $C$ .

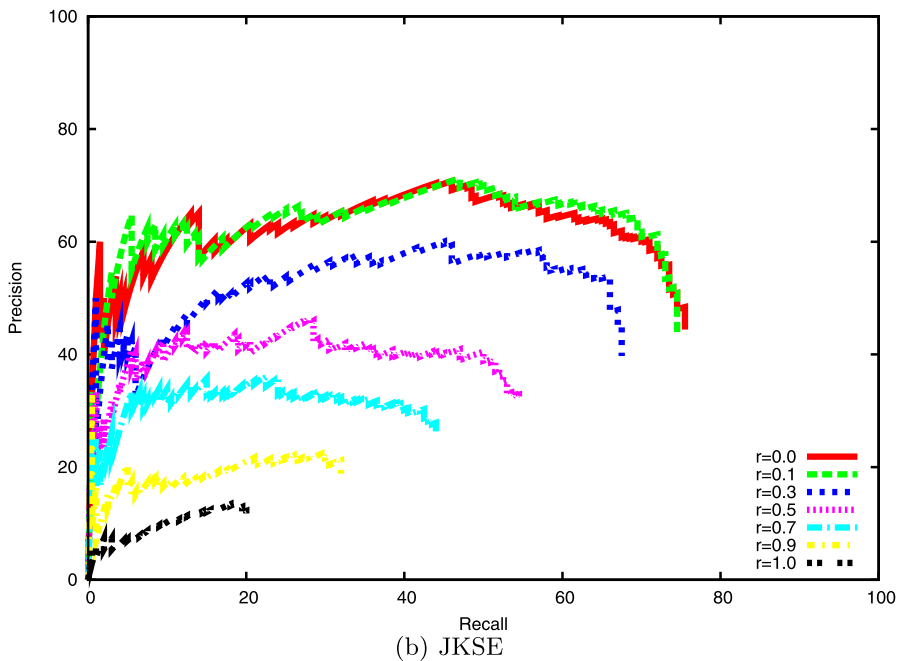
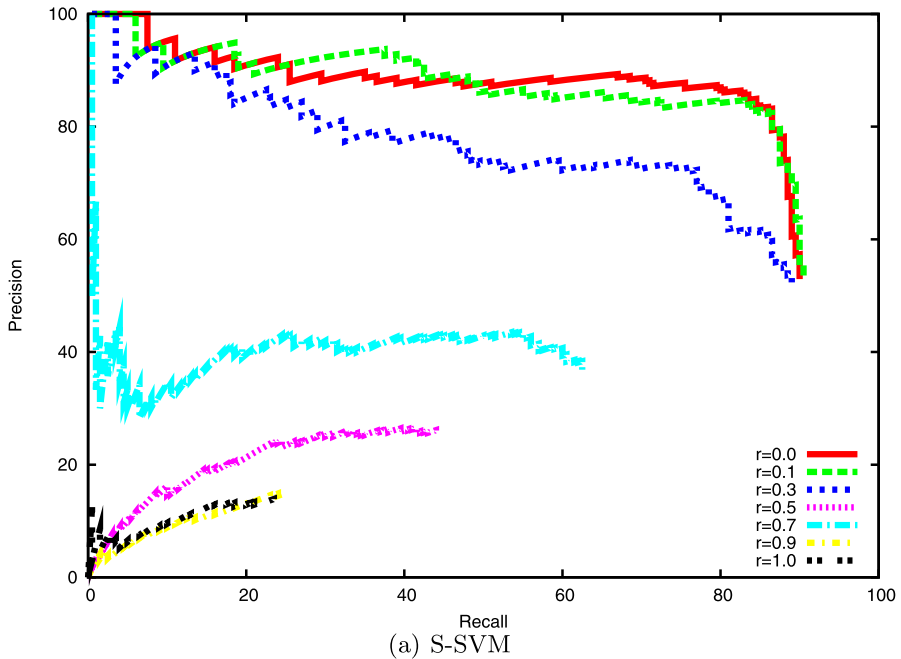
In contrast, JKSE with a linear kernel starts from a lower precision level, but its performance decreases more continuously when the amount of label errors increase. Even for 90% label noise, JKSE achieves non-trivial localization performance. We attribute this somewhat surprising behavior to a successful application of the  $\nu$ -formalism, as  $\nu = 0.95$  was chosen at this level, thereby correctly treating a large amount of the training data as outliers. Furthermore, our analysis showed that JKSE is rather insensitive to the choice of  $\nu$ .

Figure 4 shows results for JKSE with the  $\chi^2$  kernel function. In comparison with Fig. 3 one can clearly see that JKSE's localization accuracy is improved, even increasing it over the results achieved by S-SVM. Adding up to 30% label noise hardly decreases the accuracy compared to perfect labels. For higher noise levels the performance decreases, however always staying clearly above the results for S-SVM. Even when 90% of training labels are modified compared to the original dataset, JKSE reaches a recall level of 50% and over most of the plot precision lies above 40%.

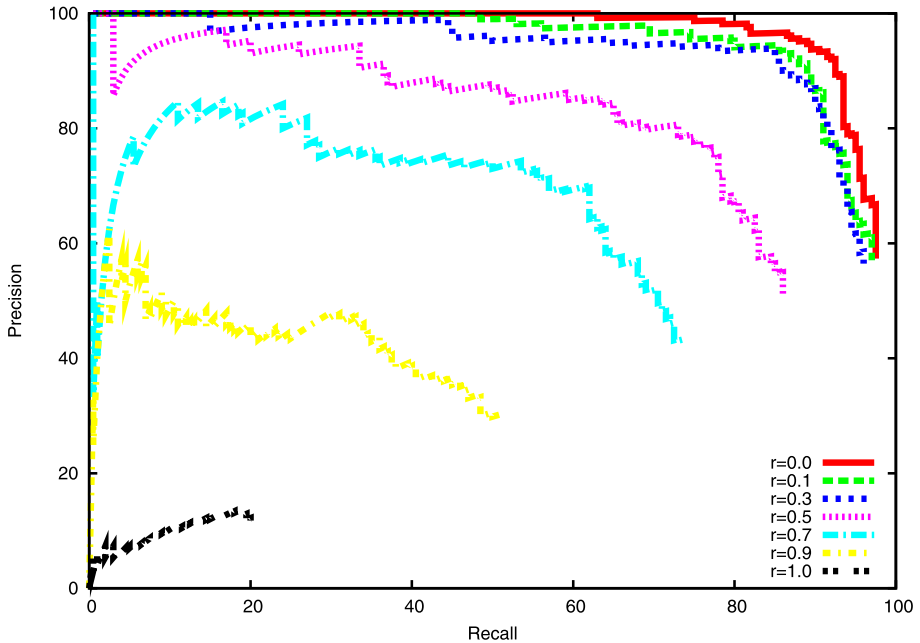
Table 1 summarizes the results in terms of the *equal error rates*, i.e. the operating point of the system where the number of false positive detections equals the number of missed detections. It shows the same trend as the precision–recall curve: JKSE is outperformed by S-SVM when using a linear image kernel, but JKSE with a  $\chi^2$ -kernel function dominates both other methods in terms of the error rate achieved. Clearly, this improved performance is a direct consequence of the use of a better kernel function. It is likely that S-SVM based localization would profit from using a  $\chi^2$  localization kernel as well. However, as mentioned above, training S-SVM with such a kernel is not computationally feasible with current techniques.

## 5.3 Experiment: sequence labeling

*Sequence labeling* is another area of research where structured prediction has proved useful for many applications, e.g. *part-of-speech tagging* and *optical character recognition*. Altun



**Fig. 3** Precision-recall plots of localization performance of S-SVM (a) and JKSE (b) for different levels of label noise (percentage indicated by  $r$ , i.e.  $r = 1.0$  means completely randomized input data). At low noise levels, S-SVM clearly dominates JKSE in terms of accuracy. When the noise reaches 50% or more, S-SVM's performance drops sharply, whereas JKSE's only gradually decreases. At 90% randomized labels, JKSE is still able to achieve better than random performance



**Fig. 4** Precision-recall plots of localization performance of JKSE with  $\chi^2$ -kernel. Precision and recall are clearly improved over the linear kernel. The resulting accuracy is also higher than with discriminatively trained S-SVM training (Fig. 3). Even if 90% of the training data is mislabeled ( $r = 0.9$ ), localization performance is reasonable

**Table 1** Equal-error rates in UIUC Car object localization task: S-SVM with linear kernel and JKSE with linear and  $\chi^2$ -kernel at different noise levels  $r$

	$r = 0.0$	$r = 0.1$	$r = 0.3$	$r = 0.5$	$r = 0.7$	$r = 0.9$	$r = 1.0$
S-SVM (linear)	18%	16%	27%	76%	55%	92%	91%
JKSE (linear)	36%	35%	43%	60%	68%	79%	91%
JKSE ( $\chi^2$ )	8%	11%	12%	24%	37%	62%	91%

et al. (2003) introduced a kernel specifically designed for this task that has the following form:

$$k_{\text{seq}}((x, y), (x', y')) = \sum_{r=1}^m \sum_{s=1}^{m'} k^{rs}((x, y), (x', y')) \tag{15}$$

with

$$k^{rs}((x, y), (x', y')) = \delta(y^r, y'^s) \left( k_{\text{base}}(x_i^r, x_i'^s) + \sum_{t=1}^T \delta(y^{r-t}, y'^{s-t}) \right). \tag{16}$$

Here  $x = (x_1, \dots, x_m)$  and  $x' = (x'_1, \dots, x'_{m'})$  are sequences of arbitrary length and  $y = (y_1, \dots, y_m)$  and  $y' = (y'_1, \dots, y'_{m'})$  are their corresponding labelings.  $k_{\text{base}}$  is a base kernel

**Table 2** Per-node error rates for JKSE and other methods on the *OCR* sequence labeling task (Perceptron and OLaRank results from Bordes et al. 2008). JKSE fails to solve the problem

Method	S-SVM	Perceptron (Collins 2002)	OLaRank (Bordes et al. 2008)	JKSE
Error rate	21.1%	48.6%	24.2%	88.3%

between elements of  $x$  and  $x'$  and  $T$  is the assumed length of interaction when interpreting the sequences as Markov chains.

A major advantage of the kernel  $k_{\text{seq}}$  is that it allows MAP inference by a Viterbi algorithm, which is efficient at least for  $T = 1$ . This makes it a feasible choice for the joint-kernel map in a structured SVM setup.

In the following, we use  $k_{\text{seq}}$  as joint-kernel map in the JKSE procedure, showing that this does not lead to good prediction performance. Subsequently, we analyze and explain this behavior.

## 5.4 Results

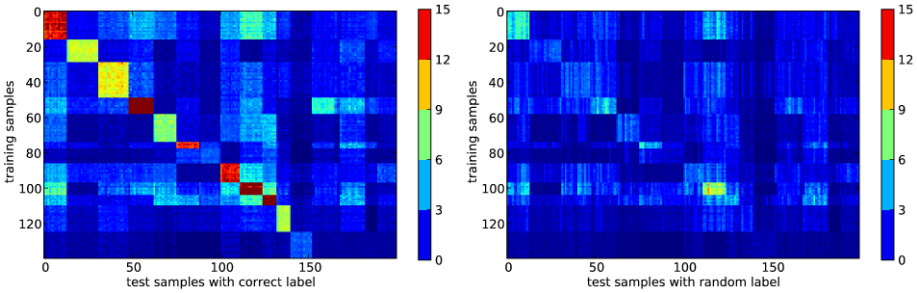
We evaluate JKSE with the *sequence*-kernel on the *Optical Character Recognition (OCR)* dataset.<sup>6</sup> It consists of handwritten *words*, i.e. sequences of letters, that were collected from different writers. The original data was collected for (Kassel 1995), but we use the subset introduced in (Taskar et al. 2003), which has become a standard dataset to benchmark sequence labeling techniques, see (Bakır et al. 2007). There are 10 predefined cross-validation splits each containing approximately 600 example for training, 600 for model selection and 5500 examples for testing. In order to be comparable to previous studies, we use a linear kernel for  $k_{\text{base}}$  and set  $T = 1$ , allowing efficient inference. The resulting error rates for JKSE, S-SVM and two other methods from the literature are given in Table 2.

Table 2 shows that JKSE is not able to solve the sequence prediction task satisfactorily, achieving error rates near the chance level. In fact, an analysis of the output sequences predicted by JKSE shows that it has a strong bias towards a small subset of labels, and most of the other ones it did not predict at all.

## 5.5 Discussion

In the following, we interpret the failure of JKSE on the sequence prediction task as a direct consequence of the fact that using the kernel  $k_{\text{seq}}$  violates the fundamental assumption underlying JKSE, namely that  $p(x, y)$  must be close to zero for *sample-label* pairs  $(x, y)$  when  $y$  not the correct label for  $x$ . Note that, because of (4), the modeled density  $p(x, y)$  depends monotonically on the entries of the kernel matrix, i.e. JKSE treats  $p(x, y)$  as large, if the kernel values  $k((x, y), (x_i, y_i))$  are large for at least some support vectors  $(x_i, y_i)$ . We illustrate this in Fig. 5. It shows the kernel values for  $k_{\text{seq}}$  in two cases: The left image shows an excerpt of the matrix of kernel entries between the training examples and the test examples when both are labeled according to the error-free ground truth. One observes a block structure that indicates that  $k_{\text{seq}}((x, y), (x_i, y_i))$  is large when comparing similar words with each other and relatively small otherwise. The right illustration shows the kernel matrix for the same data samples, but with each label sequence  $y$  a randomly shuffled version of the correct one, where during the randomization we ensured that the correct and incorrect label

<sup>6</sup><http://ai.stanford.edu/~btaskar/ocr/>.



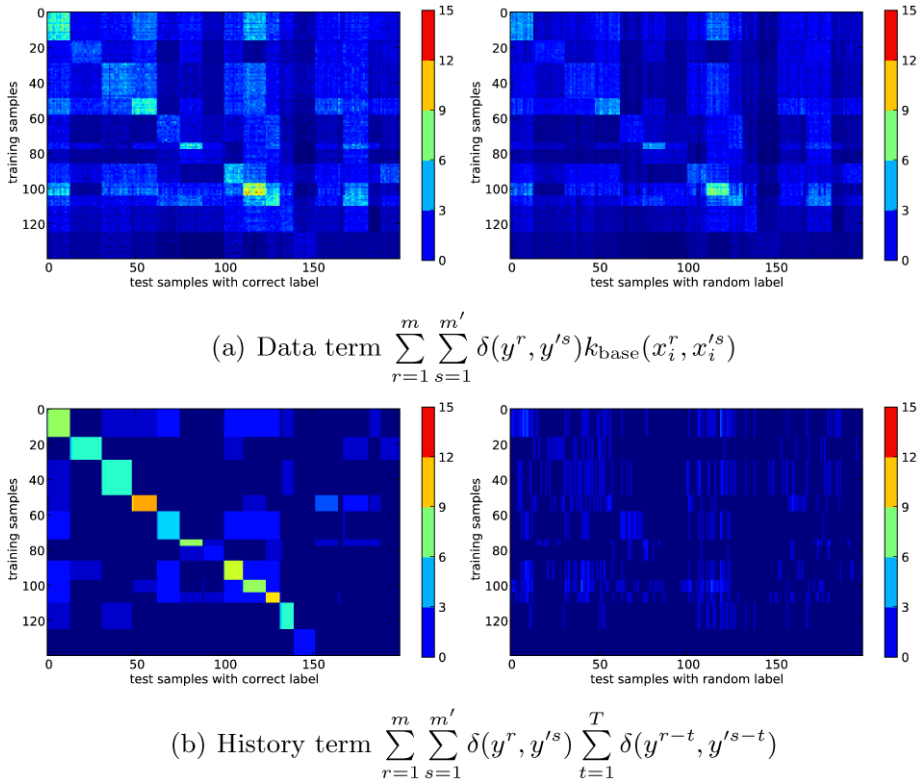
**Fig. 5** Kernel matrices between training and test data (excerpt) for the sequence labeling task (see (15)). In the *left image*, the test data is paired with the correct sequence labels. In the *right image*, the labels within each sequences are randomly shuffled

coincide in not more than 25% of positions. The resulting kernel matrix *still* has non-zero values and a block structure, indicating that  $k_{\text{seq}}((x, y), (x_i, y_i))$  can be large even when  $y$  is a wrong labeling of  $x$ . As a consequence, we cannot reliably model a suitable  $p(x, y)$  from  $k_{\text{seq}}$  and JKSE fails.

The reason for this can be seen from (15).  $k_{\text{seq}}$  has a structure resembling a *set kernel*: its value for two labeled sequences is the sum over local kernel evaluations  $k^{rs}$  between at all locations in both strings. At positions where both sequences have the same local label,  $k^{rs}$  is the sum of two components: one term measuring on the similarity of the sequence elements, and one term that compares the previous labels within the sequences (see (16)). Consequently, we can also decompose  $k_{\text{seq}}$  into two additive components: a *data term*, that mainly measures the similarity of the data samples, and a *history term*, that measures the similarity within the label sequence. This construction makes the sequence kernel susceptible to mislabelings, because even if a local label  $y^r$  for a local token  $x^r$  is incorrect,  $k_{\text{seq}}$  will contain the additive contribution  $k(x^r, x'^s)$  from all local position  $s$  at which  $y^r$  would have been the correct label. We demonstrate this effect visibly in Fig. 6: for randomized test labels, the *history*-term becomes much smaller than in the case of correct labels, see Fig. 6(b).<sup>7</sup> However, the *data* term still has large values, even when a wrong labeling is used, see Fig. 6(a).

The *localization kernel* that we used for object detection task of Sect. 5.1 does not have the same kind of problems. Figure 7 shows the kernel entries between training and test examples, where the latter have either the correct object locations as label (left), or randomly sampled locations in the image, overlap the ground truth by not more than 50% (right). In the case of a linear kernel (Fig. 7(a)), the kernel values between training and correctly labeled test examples is high in most cases, and only few test examples have high values when labeled incorrectly. With a  $\chi^2$ -kernel (Fig. 7(b)) both kernel matrices have nearly constant values with the values for correct test labels significantly higher than for incorrect test labels. This shows that the estimate of  $p(x, y)$  will be large only if  $y$  is a correct labeling of  $x$ , and the assumption underlying JKSE is fulfilled.

<sup>7</sup>Note, however, that there are also incorrect labelings that cause a high value of the history term, e.g. cyclic permutations of the correct label sequence.

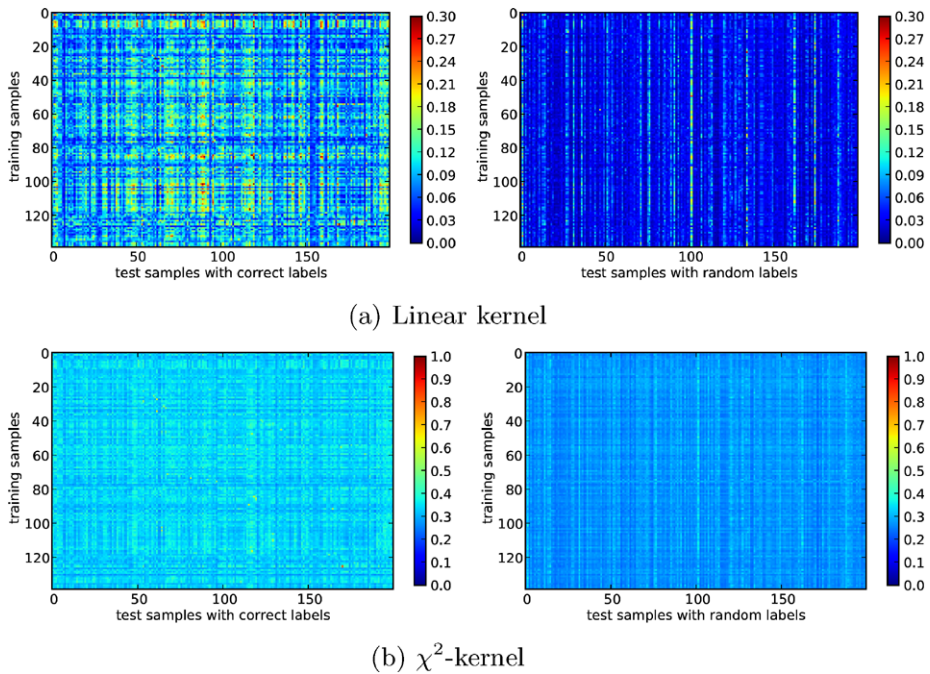


**Fig. 6** Contribution of data and history term to the kernel matrices in Fig. 5 between training and test sequences with correct labelings (*left*) and random labelings (*right*). While randomizing the test labels strongly reduces the history part, the data term remains significantly large

## 6 Conclusions

We have proposed Joint Kernel Support Estimation (JKSE), a technique that allows structured prediction with a training procedure that is as efficient as ordinary SVM training. The crucial insight is to choose a generative modelling of the joint sample-label probability density, and to learn it using a one-class SVM for margin-based support estimation. The resulting algorithm is on the one hand very efficient, as no iterated inference has to be performed at training time. On the other hand it is robust, because it does not assume that all label output labels for a sample are included in the training set, and it can handle mislabeled data through use of the  $\nu$ -formalism.

From the point of pure classification performance with identical joint kernel functions, joint kernel support estimation is not as powerful as discriminative techniques. In particular, its performance depends crucially on whether the kernel used supports the underlying assumption that  $p(x, y)$  vanishes if  $y$  is an incorrect label for  $x$ . Therefore, if the prerequisites to apply discriminative techniques are fulfilled, and if sufficient time for training is available, then these should be preferred. However, we have also shown that JKSE is a valuable contribution to the area of structured prediction, as it can still be applied in situations where CRF or S-SVM have problems or even fail: when training is not computationally feasible or only so by choosing a suboptimal kernel, and also, when the provided training data contain



**Fig. 7** Kernel matrices between training and test data for the object localization task. In the *left images*, the test data is paired with the correct object positions as labels. In the *right image*, the labels for each test samples are chosen randomly. The kernel matrices are relatively uniform, with larger values for correct sample-label pairs  $(x, y)$  than for incorrect ones

too many incomplete or incorrect labels. While few of the standard benchmark datasets in machine learning are of this type, this situation is quite common in realistic pattern recognition problems, especially in domains where ground truth cannot easily be generated even by humans, e.g., bioinformatics or medical image annotation.

The introduction of a new generative procedure for structure prediction poses many questions of both theoretical and practical nature. Clearly, as for any new prediction technique, one has to ask for consistency and generalization bounds of JKSE. While density estimation using OC-SVMs is well studied, and certain consistency results are known (see Vert and Vert 2005), the subsequent MAP-estimate is inherently nonrobust and leaves many open problems, just as is the case for the discriminative S-SVM.

An issue of special interest for us is the construction of joint-kernel functions that are better suited to joint-space support estimation. In ordinary binary or multiclass classification, kernels can be thought of as generalized similarity measures between samples, and most of the joint-kernels proposed in the literature follow this tradition. Possibly the most prominent examples are tensor-product kernels  $k((x, y), (x', y')) = k_{\mathcal{X}}(x, x') \cdot k_{\mathcal{Y}}(y, y')$  which define similarity of pairs as the simultaneous similarity of the components. However, the intuition behind JKSE shows us that a good joint kernel does not have to decide whether  $x$  is similar to  $x'$  and  $y$  to  $y'$ , but only *if  $y$  is as good a label to  $x$  as  $y'$  is to  $x'$* . This leaves much freedom in kernel design that we plan to investigate in future work.

**Acknowledgements** This work was funded in part by the EC project CLASS, IST 027978, and the PAS-CAL2 network of excellence.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution Noncommercial License which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and source are credited.

## References

- Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden Markov support vector machines. In *ICML* (pp. 3–10).
- Bakır, G. H., Hofmann, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (2007). *Predicting structured data*. Cambridge: MIT Press.
- Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 41.
- Bay, H., Tuytelaars, T., & Van Gool, L. J. (2006). SURF: speeded up robust features. In *ECCV* (pp. 404–417).
- Benveniste, A., Métivier, M., & Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximations*. New York: Springer.
- Bertsekas, D. P. (1995). *Nonlinear programming*. Nashua: Athena Scientific.
- Bilmes, J. A. (1999). *Natural statistical models for automatic speech recognition*. PhD thesis, University of California, Berkeley.
- Blaschko, M. B., & Lampert, C. H. (2008). Learning to localize objects with structured output regression. In *ECCV*.
- Bordes, A., Ertekin, S., Weston, J., & Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6.
- Bordes, A., Usunier, N., & Bottou, L. (2008). Sequence labelling SVMs trained in one pass. In *ECML/PKDD* (pp. 146–161).
- Bottou, L., Chapelle, O., DeCoste, D., & Weston, J. (Eds.). (2007). *Large scale kernel machines*. Cambridge: MIT Press.
- Collins, M. (2002). Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *EMNLP*.
- Finley, T., & Joachims, T. (2008). Training structural SVMs when exact inference is intractable. In *ICML*.
- Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6.
- Har-Peled, S., Roth, D., & Zimak, D. (2007). Maximum margin coresets for active and noise tolerant learning. In *IJCAI*.
- Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, & Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear SVM. In *ICML*.
- Jaakkola, T. S., & Haussler, D. (1998). Exploiting generative models in discriminative classifiers. In *NIPS*.
- Joachims, T. (2006). Training linear SVMs in linear time. In *ACM KDD*.
- Kassel, R. (1995). *A comparison of approaches to on-line handwritten character recognition*. PhD thesis, Massachusetts Institute of Technology, Cambridge.
- Kindermann, R., & Snell, J. L. (1980). *Markov random fields and their applications*. Providence: American Mathematical Society.
- Kulesza, A., & Pereira, F. (2007). Structured learning with approximate inference. In *NIPS*.
- Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *ICML*.
- Lafferty, J., Zhu, X., & Liu, Y. (2004). Kernel conditional random fields: representation and clique selection. In *ICML*.
- Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2008). Beyond sliding windows: object localization by efficient subwindow search. In *CVPR*.
- Lin, C.-J., Weng, R. C., & Keerthi, S. S. (2008). Trust region Newton method for logistic regression. *Journal of Machine Learning Research*.
- Martinius, D., & Tax, J. (2001). *One-class classification: Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology.
- Mumford, D., & Shah, J. (1988). Optimal approximations by piecewise smooth functions and variational problems. *Communications on Pure and Applied Mathematics*, XLII(5).
- Murphy, K. P., Weiss, Y., & Jordan, M. (1999). Loopy belief propagation for approximate inference: an empirical study. In *UAI* (pp. 467–475).
- Ng, A. Y., & Jordan, M. I. (2003). On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In *NIPS*.

- Parker, C., Fern, A., & Tadepalli, P. (2007). Learning for efficient retrieval of structured data with noisy queries. In *ICML*.
- Pernkopf, F., & Bilmes, J. A. (2005). Discriminative versus generative parameter and structure learning of Bayesian network classifiers. In *ICML*.
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77.
- Sarawagi, S., & Gupta, R. (2008). Accurate max-margin training for structured output spaces. In *ICML*.
- Schölkopf, B., & Smola, A. (2002). *Learning with kernels*. Cambridge: MIT Press.
- Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7), 1443–1471.
- Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: primal estimated sub-gradient solver for SVM. In *ICML*.
- Szdemak, S., Saunders, C., Shawe-Taylor, J., & Rousu, J. (2005). Learning hierarchies at two-class complexity. In *NIPS workshop on kernel methods and structured domains*.
- Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. In *NIPS*.
- Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6.
- Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.
- Vert, R., & Vert, J.-P. (2005). Consistency of one-class SVM and related algorithms. In *NIPS*.
- Yang, X.-Y., Liu, J., Zhang, M.-Q., & Niu, K. (2007). A new multi-class SVM algorithm based on one-class SVM. In *ICCS* (pp. 677–684).