Structured Output Gaussian Processes

– Technical Report –

Botond Bócsi, bboti@cs.ubbcluj.ro

Lehel Csató, lehel.csato@cs.ubbcluj.ro

Faculty of Mathematics and Informatics, Babeş-Bolyai University,
Kogalniceanu 1, 400084 Cluj-Napoca, Romania

Jan Peters, mail@jan-peters.net

Technische Universitaet Darmstadt, Intelligent Autonomous Systems Group,
Hochschulstr. 10, 64289 Darmstadt, Germany

May 11, 2012

Abstract

Structured output learning is applied when capturing relationships in the output space of the data is relevant. Standard machine learning techniques must be extended to achieve good performance in this setting. We propose a method that is based on the insight introduced by *joint kernel support estimation*. We modified the method by using the same data representation but a different loss function – squared loss instead of hinge loss. We show that this change leads to the application of Gaussian processes instead of support vector machines. Our method is validated on two standard structured output learning tasks, object localization in natural images and wighted context free grammar learning. In both tasks we achieved state-of-the art performance. Furthermore, we applied the algorithm for inverse kinematics learning as well, showing that it is applicable in continuous domains with real-time setting.

1 Introduction

Structured output learning deals with learning of a mapping $f : \mathcal{X} \to \mathcal{Y}$ where the output domain $\mathcal{Y}$ has a *structure*. Unlike in the case of classification where $\mathcal{Y}$ is a discrete finite set or in the case of regression where $\mathcal{Y} = \mathfrak{R}$, we allow $\mathcal{Y}$ to have an arbitrary structure. Such problems are not uncommon in real world applications. Consider natural language processing where $\mathcal{Y}$ consists of parse trees, or label sequence learning where $\mathbf{y} = [y_1 \dots y_l] \in \mathcal{Y}$ are the labels of given input sequence $\mathbf{x} = [x_1 \dots x_l] \in \mathcal{X}$ – label sequence learning is used in optical character recognition

(OCR). We cannot treat OCR as an ordinary regression of classification problem since there may be correlation between the labels y_i. Another example is image localization that is also related to structured learning since the output space contains coordinates on images that has to be considered in relation with the image itself. Another important application of structured output learning is sequence alignment, e.g., RNA structure prediction, where $\mathcal{Y}$ contains arbitrary sequences over a given alphabet. Structured output learning methods can also be used when the function $f()$ is not unique. For example, it has been applied with success in robotic control for modeling the inverse kinematics function that is not a one-to-one mapping [Bócsi et al., 2011].

Solving the aforementioned problems is not straightforward using standard machine learning methods, thus, special algorithms are needed [Bakir et al., 2007]. Next, we give a brief presentation of existing structured output learning approaches. Hidden Markov models (HMMs) [Rabiner, 1989] and conditional random fields (CRFs) [McCallum and Sutton, 2006] use probabilistic graphical models to represent the relationships in the output space. The methods define the joint or the conditional probability distribution, respectively, of inputs and outputs. Then, probabilistic inference algorithms are used to make predictions. Note that HMMs and CRFs are conceptually different in the sense that the usage of the joint probability makes HMMs generative methods whereas the conditional probability makes CRFs discriminative methods [Bakir et al., 2007] [Lampert and Blaschko, 2009]. Max-margin Markov networks [Taskar et al., 2004] and structured output support vector machines (SSVMs) [Tsochantaridis et al., 2005] are another type of discriminative models. They define the decision function such that the joint input-output training data is separated from the rest of the state space with the largest possible margin. A generative analogue of SSVMs also exists called joint kernel support estimation (JKSE) [Lampert and Blaschko, 2009]. A different approach is to model the dependencies in the output space using dimensionality reduction. Kernel dependency estimation [Weston et al., 2002] uses kernel principal component analysis in the output space to model these dependencies. Then, learns a regression model in every principal component direction.

The main disadvantages of discriminative models over generative methods are that they require clearly labeled training data and usually they are computationally more expensive [Lampert and Blaschko, 2009]. To avoid these drawbacks, we focus on defining models that has a generative nature. In this paper, we propose a *generative like*[1] method to solve structured output learning problems. We model a fitness function of the joint input-output data (represented by a joint feature function), i.e., a function of $\mathcal{X} \times \mathcal{Y}$ that takes values from $\mathfrak{R}$, and maximize the model over $y \in \mathcal{Y}$ as prediction, given $x \in \mathcal{X}$. The joint input-output space can be very large, thus, the explicit modeling of the joint fitness function might be unfeasible. We propose a data driven definition of the function using Gaussian processes (GPs) [Rasmussen and Williams, 2005]. This definition also allows us to discuss the problem in a Bayesian framework since introducing priors on Gaussian process is straightforward [Rasmussen and Williams, 2005].

We aim to solve the following problem: given a training set $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ with $x_n \in \mathcal{X}$ and $y_n \in \mathcal{Y}$, where $\mathcal{Y}$ has some kind of structure, we want to find a mapping $f : \mathcal{X} \to \mathcal{Y}$ that explains the relationship between the inputs x_n and outputs y_n best. To do so, we define a function q[2] $: \mathcal{X} \times \mathcal{Y} \to \mathfrak{R}$ that returns how well a given x and y fit to each other. Then, the prediction for a test point x is calculated by maximizing q over all possible y-s, i.e.,

$$f(x) = \arg\max_y q(x, y). \tag{1}$$

[1] Our method is not a proper generative model since we cannot sample from it, however, it shares several properties with generative methods like JKSE, that is why we refer to it as a *generative like* model.

[2] For some applications we represent the joint data by a feature function $\psi(x, y)$, thus, q is defined not on the Cartesian product of $\mathcal{X}$ and $\mathcal{Y}$ but rather on $\psi(\mathcal{X}, \mathcal{Y})$. In order to keep the notations simple, in the rest of this paper we use the two definitions interchangeably.

The paper is organized as follows. In Section 2.1, we define a Bayesian framework of structured output learning. In Section 2.2 a brief introduction to GPs is given since they form the base of our method. To obtain efficient learning algorithms, some sparsification methods must be defined on GPs, therefore, we address the problem of GP sparsification. In Section 2.3, we give a detailed presentation of the *structured output Gaussian process* (SOGP) method. Section 3 details the relation between SOGP and other existing structured output learning methods. Experimental results obtained from natural language processing, weighted context free grammar learning, and inverse kinematics learning are presented in Section 4 with conclusions drawn in Section 5.

2 Structured Output Gaussian Processes

All of the general structured output learning methods[3] define the decision function as one presented in Equation (1). The distinctive feature of the methods is the different modeling of the function q. In this section, we propose a non-parametric modeling of q in a Bayesian framework. We show that this definition relates to the modeling of q using GPs.

2.1 Bayesian Structured Output Learning

We propose a Bayesian modeling of q since, as we will see later, it induces several beneficial properties of q. For example, we can introduce prior information about the problem we want to solve in a natural manner. Furthermore, we obtain not only a point-wise estimate of q but probability distribution. Using Bayesian inference, the posterior distribution of q looks as follows

$$p(q|\mathcal{D}) \propto p(\mathcal{D}|q)p_0(q), \tag{2}$$

where $p(\mathcal{D}|q)$ is the likelihood of the training set conditioned on q and $p_0(q)$ is the prior distribution of the function q. We define $p_0(q)$ to be Gaussian distributed with mean function $\mu_0(\cdot)$ and covariance function $k_0(\cdot,\cdot)$. Without loss of generality we assume $\mu_0(\cdot) = 0$, however, in real world applications the proper definition of $\mu_0(\cdot)$ may be important.

The definition of a prior on a function space in Equation (2) and then using the data to get the posterior distribution of the function, relates to the modeling of q using GPs. GP models are non-parametric Bayesian methods that define priors directly on function spaces. This property along with the non-parametric nature leads to a model that has a larger expressive power than parametric (Bayesian) models. Next, we give a brief introduction to GPs since our structured output method is based on these models.

2.2 Gaussian Processes

Gaussian processes are non-parametric Bayesian models which define a distribution over functions characterized by the mean function $\mu(\cdot)$ and covariance (or kernel) function $k(\cdot,\cdot)$[Rasmussen and Williams, 2005]. Given a training set $\{(x_n, y_n)\}_{n=1}^N$, the posterior distribution of a test point x_* is Gaussian distributed with mean μ_* and variance σ_*^2 where

$$\begin{aligned} \mu_* &= \mathbf{k}_*^\top (\mathbf{K} + \sigma_0^2 \mathbf{I})^{-1} \mathbf{y} \\ \sigma_*^2 &= k_{**} - \mathbf{k}_*^\top (\mathbf{K} + \sigma_0^2 \mathbf{I})^{-1} \mathbf{k}_*, \end{aligned} \tag{3}$$

[3] We do not consider methods which are specific to given structured output learning problems, rather those which can be applied in a general context.

where $\mathbf{K}^{ij} = k(x_i, x_j)$, $\mathbf{k}_*^i = k(x_i, x_*)$, $k_{**} = k(x_*, x_*)$, $\mathbf{I}$ is the identity matrix of size N, and σ_0^2 is the variance of the measurement noise.

The posterior prediction of a GPs can be viewed from different perspectives (weight-space view, function space view) [Rasmussen and Williams, 2005]. By taking the weight-space perspective, the posterior from Equation (3) is based on the minimization of the squared distance between the prediction of the model and the observations – for details see [Rasmussen and Williams, 2005]. This property becomes interesting when we compare our method with other structured output algorithms from a theoretical point of view.

Now, we address the problem of computational complexity of GPs since it plays an important aspect of comparison in our experiments. The main drawback of GPs is the high computational complexity of the learning process. The memory requirement is quadric in the number of training points whilst the time complexity scales cubically with the number of the data points – caused by the matrix inversion involved in Equations (3). To overcome this problem, several methods have been introduced [Csató, 2002], [Quiñonero Candela and Rasmussen, 2005], [Lawrence et al., 2002], [Snelson and Ghahramani, 2006]. All of the methods aim to reduce the number of the training points with as small *information loss* as possible. The sparsification methods vary by the different definitions of the information loss. We adopt a method proposed by Csató [2002], which can be applied online.

2.3 Structured Output Gaussian Processes

Let us consider the structured output learning framework presented in Section 2.1. In this section, we focus on modeling q from Equation (1) using GPs. The key insight is that the training data provides only *positive* examples of x and y, i.e., we know that $q(x_n, y_n)$ has a high value for all $\{(x_n, y_n)\} \in \mathcal{D}$. Without loss of generality let us suppose this value is 1. Such an unbalanced training set can easily lead to over-fitting – for example, the constant 1 function would give a solution. To avoid over-fitting, the definition of a strong prior is essential. In the rest of this section, we assume a zero mean prior since it keeps the notations simple. The value of the prior is arbitrary as long as it is smaller that the values of $q(x_n, y_n)$. The reason is that we do not care about the *real* value of $q(x_n, y_n)$ but rather where it has its maximum over y.

After the previous assumptions one may look at q as a joint probability function defined on $\mathcal{X} \times \mathcal{Y}$. This would be wrong for at least two reasons. First, as a proper probability distribution it would require to be normalized, however, the calculation of the normalization constant is often intractable [Kass and Raftery, 1995]. Second, nothing assures that some values of q do not go below zero. As a consequence, we refer to q as a fitness function and not as a probability distribution.

To model q, we define a GP on the joint data (x_n, y_n) as input, and 1 as output, with a 0 mean prior. We also have to define a joint kernel function $k(\cdot, \cdot)$ on the space $\mathcal{X} \times \mathcal{Y}$. This kernel function also contains prior knowledge about the problem we want to solve. For details about joint kernels consult Bakir et al. [2007].

Using the predictive distribution of a GP form Equation (3), the posterior distribution of q at point (x, y) is $p(q|\mathcal{D})(x, y) = \mathcal{N}(\mu_{(x,y)}, \sigma^2_{(x,y)})$ where

$$\begin{aligned} \mu_{(x,y)} &= \mathbf{k}_{(x,y)}^\top (\mathbf{K} + \sigma_0^2 \mathbf{I})^{-1} \mathbf{1} \\ \sigma^2_{(x,y)} &= k_{(x,y)(x,y)} - \mathbf{k}_{(x,y)}^\top (\mathbf{K} + \sigma_0^2 \mathbf{I})^{-1} \mathbf{k}_{(x,y)}, \end{aligned} \tag{4}$$

where $\mathbf{K}^{ij} = k((x_i, y_i), (x_j, y_j))$, $\mathbf{k}^i_{(x,y)} = k((x_i, y_i), (x, y))$, $k_{(x,y)(x,y)} = k((x, y), (x, y))$, $\mathbf{I}$ is the identity matrix of size N, σ_0^2 is the variance of the measurement noise, and $\mathbf{1}$ is the unit vector of length N.

To obtain the predictive function f from Equation (1), q is needed to be maximized over y. To perform the maximization, the variance $\sigma^2_{(x,y)}$ does not contain valuable information and we need only the point-wise estimate of q. The point-wise estimate is the posterior mean of the defined GP, i.e.,

$$q(x, y) \quad = \quad \mu_{(x,y)}. \tag{5}$$

How the maximization from Equation (1) can be done efficiently depends on the problem. Note that when the gradient of q can be calculated one can use gradient descent search [Snyman, 2005]. Observe that q is differentiable as long as the kernel function $k(\cdot, \cdot)$ is differentiable and the gradient of q can be calculated analytically. As a consequence, the gradient search can be done fast. We do not give en explicit form of the gradient since its form depends on the joint kernel function $k(\cdot, \cdot)$.

3 Relation to other methods

The presented SOGP method has a strong relationship with one-class classification methods, such as one-class support vector machines (OC-SVM) [Schölkopf et al., 2001], least square one-class support vector machines (LS-SVM) [Choi, 2009], or one-class classification with GPs [Kemmler et al., 2011]. One-class classification algorithms are unsupervised learning methods used for novelty detection, outliers detection, and density estimation. These methods model a function on the data – note that since it is unsupervised, data consist only of inputs without any labels – and use it as a probability distribution or threshold it to find the outliers. We define a similar function on the joint input-output space, and furthermore, we perform a maximization over the output space to find the best output for a given input. Our work is based on one-class classification with GPs [Kemmler et al., 2011]. Here, different interpretations of the function q are also proposed, such as, the predictive probability of the GP, the negative variance of the GP, and other heuristics. As the other interpretations of q do not have theoretical motivations, we used solely the posterior mean.

Another relationship can be observed with JKSE. JKSE is a structured output method that uses OC-SVM on the joint input-output space and maximizes its prediction similar to Equation (1). OC-SVMs are based on the minimization of the hinge loss [Lecun et al., 2006] between prediction and observation. One can define a similar model based on the quadratic loss [Lecun et al., 2006]. Such a method would be called one-class LS-SVM [Choi, 2009] [Suykens and Vandewalle, 1999] based JKSE. As we have mentioned in Section 2.2, the GP approximation is also based on quadratic loss minimization, thus, JKSE with LS-SVM is equivalent to SOGP. The formulation of the problem in the GP framework is more advantageous for two reasons: (1) it provides a probabilistic treatment where we can introduce prior knowledge into the prediction process in a natural way, (2) and we have access to the GP sparsification methods. As we will show in Section 4, the sparsification is important to keep the complexity low since quadratic loss function based minimizations do not result in such sparse representations like OC-SVMs do. To best of our knowledge, such a method based on quadratic loss minimization has not been investigated in the structured output learning framework.

We highlight the comparison with JKSE and SSVM. All three methods are similar in the sense that they use the same joint data representation. They are different regarding the loss function they minimize. JKSE minimizes the hinge loss, SOGP minimizes the quadratic loss, and SSVM minimizes the perceptron loss [Lecun et al., 2006]. One cannot decide which loss function is the *best* to use, next, we present results of experiments which show that different problems prefer different loss functions.

Figure 1: Examples of test images for object localization.

4 Experiments

In this section, we present the evaluation of SOGP on two common structured output learning tasks, i.e., object localization in images and weighted context free grammar learning. We also show that SOGP is applicable in continuous domains for learning multivalued functions. Such a non-unique function is the inverse kinematics function of a redundant robotic arm. The later experiment also provides evidence that by applying sparsification methods, the complexity of SOGP can be reduced as much as it is applicable in real-time setting.

4.1 Object localization in images

We used a similar setup of Lampert and Blaschko [2009] to use structured output learning for object localization in natural images. We used the UIUCcars[4] data-set to perform the experiment. The training set contained 550 black-and-white images of different type of cars. Each image had a dimension of 40×100. The test set consisted of 170 images with different sizes, however, the cars on them had roughly the same size as the cars from the training images – Figure 1 shows examples of test images. The task was to find the bounding boxes of the cars from the test images based on the training examples. Note that the test images might contain more than one cars and any of the correct bounding boxes were considered a correct label. The input space contained the images with the cars whereas the output space contained the coordinates of the left-up corner of the bounding box. As the joint input-output representation we segmented the sub-image covered by the actual bounding box into 9 equal parts and calculated the color histogram for each part. The totality of the histograms were the joint representation of the image and the bounding box. The maximization from Equation (1) was performed by a full search on the image. Note that there are more efficient searching methods, however, we were not interested in the speed of the search but rather in the accuracy of the representation.

In this experiment, we analyzed the efficiency of SOGP in relation with JKSE since the conception of this method is very closer to SOGP. In particular, we were interested in the gain provided by the dense data representation of SOGP in contrast to the sparse representation of JKSE. We were also interested in the gain (or loss) caused by the GP sparsification algorithms on contrast to JKSE and sparse LS-SVM. Therefore, we performed the object localization experiment with SOGP, JKSE, SOGP with a sparse GP, and JKSE with sparse LS-SVM. The GP sparsification was based on Csató [2002] whilst the LS-SVM sparsification on de Kruif and de Vries [2003]. Initially, JKSE has selected 63 images form the 550 as support points, thus, we set the maximum number of the support points 63 to all sparsification methods. In this way, we obtain a fair comparison. For every experiment we used squared exponential kernels.

As a measure of performance, we used the percentage of the recalled cars. Note that this number depend on the required precision, thus, we show results with different precision levels. Results are shown in Figure 1. One can see that SOGP clearly outperforms the other methods.

[4] http://l2r.cs.uiuc.edu/~cogcomp/Data/Car/

Method	Recall (%) / precision			
SOGP	54.71	58.82	67.06	69.41
SOGP with sparse OGP	48.24	53.53	62.94	65.88
JKSE with OC-SVM	45.29	50.59	59.41	61.76
JKSE with sparse LS-SVM	45.29	51.76	61.18	62.35

Table 1: Object localization results. The percentage of successful car recalls as a function of the required precision. SOGP outperforms the other methods even when sparsification is applied, and the number of support points are the same as with JKSE.

SOGP with sparse GP is also better than the other methods, meaning that using GP sparsification, better performance can be achieved based on the same number of support points. One would expect that SOGP with sparse GP would produce the same results as JKSE with sparse LS-SVM. However, as results show, GP sparsification methods are more accurate and better developed.

4.2 Weighted Context Free Grammar Learning

In this experiment, we used a similar setup to Tsochantaridis et al. [2005]. The goal was to predict a parse tree of a sequence of terminal symbols of a wighted context free grammar. For both training and testing data, we generated random sentences from a highly ambiguous weighted context free grammar – 90% of the sentences had more than one possible parse trees. The lengths of the sentences were between 15 and 25. The Chomsky normal form of the grammar contained 12 non-terminal symbols, 44 terminal symbols, and 54 rules. We used 1000-1000 sentences for training and testing respectively. The joint data (sentence and parse tree) was represented by a vector with the length of the number of the rules in the grammar. Each position of the vector contained how many times the respective rule has been use in the generation of the parse tree [Tsochantaridis et al., 2005]. The maximization over all possible parse trees was done by the Cocke–Younger–Kasami algorithm [Manning and Schütze, 1999], [Tsochantaridis et al., 2005]. For SOGP and SSVM, we used linear kernels since the complexity of SSVM highly depends on the type of the kernel and we wanted to keep the comparison fair.

We compared sparse SOGP with *probabilistic context free grammar* (PCFG) learning [Johnson, 1998], that is a maximum likelihood based algorithm, and SSVM. Figure 2 shows that the results are very similar for every method. SOGP is slightly better than PCFG, however, we could not achieve the accuracy of SSVM. A possible explanation is that the perceptron loss is more suitable for weighted context free grammar learning.

4.3 Learning Inverse Kinematics

In this experiment, we learned the inverse kinematics function of a simulated Barrett WAM robotic arm with 7 degrees of freedom. We performed this rather unusual experiment for structured output learning to highlight two points: (1) SOGP is applicable in continuous domains where the maximization form Equation (1) cannot be done by exhaustive search, and (2) the complexity of the method can be kept low (using sparsification) as it can be used in a real-time setting. We followed the idea of Bócsi et al. [2011] regarding how structured output learning can be applied for inverse kinematics learning.

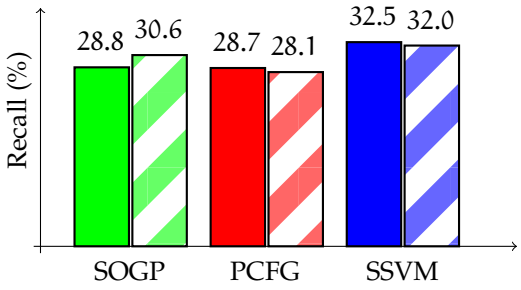


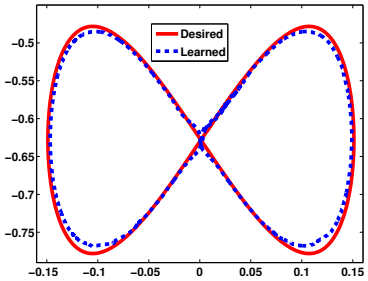
Figure 2: Results for weighted context free grammar learning. Correct parse tree recalls (%) for the training (left, solid columns) and test (right, striped columns) set. We could achieve state-of-the-art performance but we could not outperform SSVM.

Inverse kinematics functions map the coordinates of the end-effector $\mathbf{x}$ (3 dimensional Cartesian coordinates of the end point of the robot arm) into joint angles $\boldsymbol{\theta}$. Learning inverse kinematics functions relates to modeling multivalued functions since different joint configurations can lead to the same end-effector position, as shown in Figure 3(b).

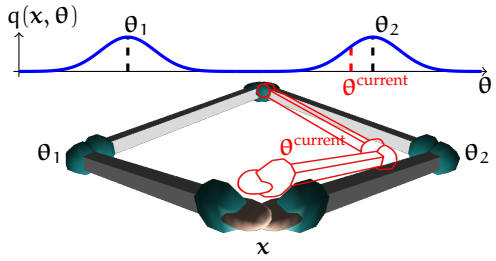
To collect training data, we used an analytical controller to draw a figure eight in the end-effector space (see Figure 3(a)) with two different initial joint configurations. The data from the two experiments were merged, thus, we obtained an ambiguous training set. We used the following joint data representation $[\mathbf{x}\ \mathbf{y}\ \sin(\mathbf{y})\ \cos(\mathbf{y})]$. The sines and cosines of the joint angles were added since the forward kinematics highly depends on these values. During the learning process, the number of the support points was limited to 500 to keep the prediction time low. As the kernel function, we used squared exponential kernel on the presented joint data representation. The maximization from Equation (1) has been done by conjugate gradient search [Snyman, 2005] starting from current joint position $\boldsymbol{\theta}^{\text{current}}$. The search scheme is presented on Figure 3(b). Note that since q is a smooth and differentiable function – Equations (4) and (5) –, we could use the analytical gradient that resulted in significant speed-up of the search. We defined a prior other than the zero mean prior. A smaller prior probability has been assigned to joint configurations that are close to the physical limits of the robot, thus, we could avoid to damage it. Another possibility is to define a higher prior probability around a given rest posture as keeping the arm in a *comfortable*, save position. After inverse kinematics was learned, the Barrett arm was able to follow the trajectory of a figure eight defined in the Cartesian space, results are shown on Figure 3(a).

5 Discussion

We proposed an extension of JKSE to solve structured output learning problems. The same joint data representation was used but a different loss function has been minimized. The squared loss function was applied instead of the hinge loss. This change leads to the application of GPs instead of support vector machines. Since GPs are the same as LS-SVM, SOGP is equivalent to JKSE with LS-SVM, however, to best of our knowledge it has not been used in structured output learning. Furthermore, approaching the problem from a Bayesian probabilistic point of view has several benefits: (1) the probabilistic framework provides the introduction of priors in a natural way, and (2) the GP sparsification methods provide fast algorithms applicable in real-time setting. Experiments show that we could achieve state-of-the-art performance on standard



(a) Figure eight tracking of the 7 degrees of freedom Barrett WAM.



(b) Illustration of the structured output inverse kinematics algorithm prediction scheme.

Figure 3: (a) Result of the SOGP based figure eight tracking. (b) During the training process $\mathbf{x}$ has been reached by two different joint configurations θ_1 and θ_2, therefore, $q(\mathbf{x}, \theta_1) = q(\mathbf{x}, \theta_2)$. However, as the current joint configuration θ^{current} is closer to θ_2, the algorithm chooses a prediction that is closer to θ_2 [Bócsi et al., 2011].

structured output learning tasks.

Acknowledgements

B. Bócsi wishes to thank for the financial support provided from program: Investing in people! PhD scholarship, project co-financed by the European Social Fund, sectoral operational program, human resources development 2007 - 2013. Contract POSDRU 88/1.5/S/60185 – "Innovative doctoral studies in a knowledge based society". B. Bócsi and L. Csató acknowledge the support of the Romanian Ministry of Education, grant PN-II-RU-TE-2011-3-0278.

References

G. H. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. V. N. Vishwanathan, editors. *Predicting Structured Data*. Neural Information Processing. The MIT Press, September 2007. ISBN 0262026171.

B. Bócsi, D. Nguyen-Tuong, L. Csató, B. Schoelkopf, and J. Peters. Learning inverse kinematics with structured prediction. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 698–703, San Francisco, USA, 2011.

Y.-S. Choi. Least squares one-class support vector machine. *Pattern Recognition Letters*, 30:1236–1240, October 2009.

L. Csató. *Gaussian Processes - Iterative Sparse Approximations*. PhD thesis, Aston University, UK, 2002.

B. de Kruif and T. de Vries. Pruning error minimization in least squares support vector machines. *IEEE Transactions on Neural Networks*, 14(3):696–702, 2003.

M. Johnson. Pcfg models of linguistic tree representations. *Computatoinal Linguistics*, 24:613–632, December 1998.

R. E. Kass and A. E. Raftery. Bayes Factors. *Journal of the American Statistical Association*, 90(430): 773–795, 1995. ISSN 01621459.

M. Kemmler, E. Rodner, and J. Denzler. One-class classification with gaussian processes. In *Proceedings of the 10th Asian conference on Computer vision - Volume Part II*, ACCV'10, pages 489–500. Springer-Verlag, 2011.

C. H. Lampert and M. B. Blaschko. Structured prediction by joint kernel support estimation. *Machine Learning*, 77:249–269, December 2009. ISSN 0885-6125.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *Neural Information Processing Systems(NIPS)*, pages 609–616. MIT Press, 2002.

Y. Lecun, S. Chopra, R. Hadsell, F. J. Huang, G. Bakir, T. Hofman, B. Schdž˝lkopf, A. Smola, and B. T. (eds. A tutorial on energy-based learning. In *Predicting Structured Data*. MIT Press, 2006.

C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

A. McCallum and C. Sutton. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, December 2005.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, December 2005. ISBN 026218253X.

B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computations*, 13:1443–1471, July 2001. ISSN 0899-7667.

E. Snelson and Z. Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264. MIT press, 2006.

J. A. Snyman. *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Applied Optimization, Vol. 97. Springer-Verlag New York, Inc., 2005.

J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9:293–300, June 1999.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *NIPS*, pages 873–880, 2002.