21st International Conference on Pattern Recognition (ICPR 2012)
November 11-15, 2012. Tsukuba, Japan

Learning Markov Networks by Analytic Center Cutting Plane Method

Konstiantyn Antoniuk, Vojtěch Franc, Václav Hlaváč
Faculty of Electrical Engineering of the Czech Technical University in Prague
{*antonkos,xfrancv,hlavac*}*@cmp.felk.cvut.cz*

Abstract

During the last decade the super-modular Pair-wise Markov Networks (SM-PMN) have become a routinely used model for structured prediction. Their popularity can be attributed to efficient algorithms for the MAP inference. Comparably efficient algorithms for learning their parameters from data have not been available so far. We propose an instance of the Analytic Center Cutting Plane Method (ACCPM) for discriminative learning of the SM-PMN from annotated examples. We empirically evaluate the proposed ACCPM on a problem of learning the SM-PMN for image segmentation. Results obtained on two public datasets show that the proposed ACCPM significantly outperforms the current state-of-the-art algorithm in terms of computational time as well as the accuracy because it can learn models which were not tractable by existing methods.

1. Introduction

A Pairwise Markov Network (PMN) is a powerful representation of dependencies in structured objects. A PMN is defined by an undirected graph consisting of nodes $\mathcal{T}$ and edges $\mathcal{E} \subseteq \binom{\mathcal{T}}{2}$. The nodes correspond to elementary objects while the edges represent possible interactions between them. Each object $t \in \mathcal{T}$ is characterized by an observation $x_t \in \mathcal{X}$ and a discrete label $y_t \in \mathcal{Y}$. Dependencies between observations and labels are modeled for each object t by a function $q_t(x_t, y_t)$ called unary potential. Dependencies between objects connected by an edge are modeled by functions $g_{tt'}(y_t, y_{t'})$ called pair-wise potentials. The PMNs are typically used for prediction of the labels $\boldsymbol{y} = \big(y_t \in \mathcal{Y} \mid t \in \mathcal{T}\big)$ from a tuple of observations $\boldsymbol{x} = (x_t \in \mathcal{X} \mid t \in \mathcal{T})$ using the prediction rule

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}^{\mathcal{T}}} s(\boldsymbol{x}, \boldsymbol{y}) \tag{1}$$

whose scoring function $s\colon \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \to \mathbb{R}$ is defined as

$$s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t \in \mathcal{T}} q_t(x_t, y_t) + \sum_{tt' \in \mathcal{E}} g_{tt'}(y_t, y_{t'})\,.$$

Evaluation of (1) requires solving an integer program called a *max-sum problem* which is NP-complete in general. Polynomially solvable instances of the max-sum problem are obtained by restricting either the neighbourhood structure $\mathcal{E}$ or the pair-wise potentials $g_{tt'}(y_t, y_{t'})$. In this paper we concentrate on the later case when $\mathcal{E}$ can be arbitrary but $g_{tt'}(y_t, y_{t'})$ are *super-modular* (SM) functions. The definition of SM functions requires the set of labels to be fully ordered; w.l.o.g. we use $\mathcal{Y} = \{1, 2, \ldots, Y\}$ endowed with natural order. Then, the function $g(y, y')$ is SM iff for each pair of labels $(y, y') \in \mathcal{Y}^2$ such that $y+1 \in \mathcal{Y}$,$y'+1 \in \mathcal{Y}$ the following inequality holds

$$g(y, y') + g(y+1, y'+1) \geq g(y, y'+1) + g(y+1, y') \tag{2}$$

We denote (1) with pair-wise potentials (2) as *SM max-sum problem*. It is well known that the SM max-sum problem with two labels, $Y = 2$, can be solved efficiently by graph-cuts [5]. It is not widely known that the SM max-sum problems with more than two labels, $Y > 2$, can be also efficiently solved via transforming them to their two-label equivalents [7]. The *Super-modular Pair-wise Markov Networks* (SM-PMN) are routinely used in computer vision e.g. for image segmentation or stereo matching.

Before the SM-PMN can be applied its potentials q_t, $g_{tt'}$ have to be specified. In principle this can be done by hand but modeling complex objects may require learning. A discriminative learning of a general SM-PMN formulated as a polynomially solvable convex optimization problem has been first proposed in [2]. Unfortunately, existing optimization methods are not efficient enough on real-life instances of this problem. In this paper we attempt to improve the situation by adapting the Analytic Center Cutting Plane Method (ACCPM) [3] for the SM-PMN learning. The ACCPM is a powerful method for non-smooth optimization which has been so far overlooked by the machine learning community. We compared the proposed ACCPM against the Bundle Method for Risk Minimization [8], the current state-of-the-art approach for structured output learning, on a

problem of learning the SM-PMN for image segmentation. Results on two public datasets show that the proposed ACCPM signifiantly outperforms the BMRM algorithm in terms of the time and the accuracy.

The paper organization: the formulation of the SM-PMN learning is given in Section 2. The proposed the ACCPM is derived in Section 3. Section 4 presents experiments and Section 5 concludes the paper.

2. Discriminative learning of Supermodular Pair-wise Markov Networks

In this section we formulate learning of the SM-PMN as a convex optimization problem following [2] and we mention the current state-of-the-art optimization method for its solution together with its weaknesses.

Let us assume we are given a single[1] training example $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}) \in \mathcal{X}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}}$ i.i.d. from an unknown p.d.f. $P(\boldsymbol{x}, \boldsymbol{y})$. The goal is to learn potentials $q_t, g_{tt'}$ of the SM-PMN such that for a given loss function $\Delta\colon \mathcal{Y}^{\mathcal{T}} \times \mathcal{Y}^{\mathcal{T}} \to \mathbb{R}$ the expected risk $E_P(\Delta(\boldsymbol{y}, \operatorname{argmax}_{\boldsymbol{y}'} s(\boldsymbol{x}, \boldsymbol{y}')))$ of the prediction rule (1) is minimal. It is further assumed that the potentials are linear functions, i.e. $q_t(x, y) = \boldsymbol{\theta}^T \boldsymbol{\psi}_t(x, y)$, $g_{tt'}(y, y') = \boldsymbol{\theta}^T \boldsymbol{\psi}_{tt'}(y, y')$ where $\boldsymbol{\theta} \in \mathbb{R}^n$ is an unknown parameter vector to be learned and $\boldsymbol{\psi}_t\colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^n$, $\boldsymbol{\psi}_{tt'}\colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^n$ are fixed maps. Under this assumption the prediction rule (1) becomes an instance of a linear classifier, i.e. the scoring function reads $s(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\theta}^T \boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{\Psi}(\boldsymbol{x}, \boldsymbol{y}) = \sum_t \boldsymbol{\psi}_t(x_y, y_t) + \sum_{tt'} \boldsymbol{\psi}_{tt'}(y_y, y_{t'})$. Learning of the parameter vector $\boldsymbol{\theta}$ of the SM-PMN (1) is formulated as a empirical risk minimization problem

$$\Theta^* = \operatorname*{argmin}_{\boldsymbol{\theta} \in \Theta} f(\boldsymbol{\theta}) := \left[\frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 + r(\boldsymbol{\theta}) \right], \quad (3)$$

where $\Theta = \{\boldsymbol{\theta} \in \mathbb{R}^n \mid g_{tt'}(y, y') = \boldsymbol{\theta}^T \boldsymbol{\psi}_{tt'}(y, y')$ satisfies (2) $\forall tt' \in \mathcal{E}\}$ and

$$r(\boldsymbol{\theta}) = \max_{\boldsymbol{y} \in \mathcal{Y}^{\mathcal{T}}} \left[\Delta(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \boldsymbol{\theta}^T (\boldsymbol{\Psi}(\hat{\boldsymbol{x}}, \boldsymbol{y}) - \boldsymbol{\Psi}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})) \right]. \quad (4)$$

The objective f is composed of the quadratic regularizer used to prevent over-fitting and the risk term r which is a convex upper bound on the empirical loss $\Delta(\hat{\boldsymbol{y}}, \operatorname{argmax}_{\boldsymbol{y}} s(\hat{\boldsymbol{x}}, \boldsymbol{y}))$. The strength of the regularization is controlled by a constant $\lambda \geq 0$ typically determined on validation data. The problem (3) is convex and for $\lambda > 0$ it is strictly convex, i.e. Θ^* is a set of minimizers in general but for $\lambda > 0$ it contains a single point. The problem (3) can be expressed as an equivalent quadratic program with $\mathcal{O}(|\mathcal{Y}|^{|\mathcal{T}|})$ constraints which is not solvable by off-the-shelf methods.

[1] This assumption greatly simplifies notation but all introduced algorithms can be easily extended for a finite number of examples.

The current state-of-the art method to solve (3) is the Bundle Method for Risk Minimization (BMRM) [8, 4] implemented e.g. in the popular StructSVM. The BMRM requires an oracle which for any $\boldsymbol{\theta}$ computes the value of $r(\boldsymbol{\theta})$ and its sub-gradient $\boldsymbol{r}'(\boldsymbol{\theta})$. Computing $r(\boldsymbol{\theta})$ and $\boldsymbol{r}'(\boldsymbol{\theta})$ requires solving the maximization (4) which is tractable if $\boldsymbol{\theta} \in \Theta$ and the loss Δ is additively decomposable over the objects and the edges like, e.g. the Hamming loss $\Delta(\boldsymbol{y}, \boldsymbol{y}') = \sum_t [\![y_t \neq y_{t'}]\!]$, in which case (4) is an instance of the SM max-sum problem [2].

A notable advantage of the BMRM are its convergence guarantees: for any $\varepsilon > 0$ the BMRM returns ε-optimal solution of (3) (i.e. $\boldsymbol{\theta}$ satisfying $f(\boldsymbol{\theta}) \leq f(\boldsymbol{\theta}^*) + \varepsilon$, $\boldsymbol{\theta}^* \in \Theta^*$) in $\mathcal{O}(\log_2 \lambda + \frac{C}{\lambda \varepsilon})$ iterations at most where C is a problem dependent constant. Sadly, the actual time required by the BMRM is overly high if it is applied to real-life problems. The computational time steeply soars for low values of λ as suggests the bound $\mathcal{O}(\log_2 \lambda + \frac{C}{\lambda \varepsilon})$. Unfortunately, in typical applications of SM-PMN the regularization is not needed, i.e. $\lambda = 0$ is optimal. E.g, in the image segmentation a single annotated image provides as many examples of label-observation interactions as it has the number of pixels and thus the over-fitting is unlikely. The BMRM does not allow setting $\lambda = 0$ and for values $\lambda \to 0$ it requires excessively long times.

3. Analytic Center Cutting Plane Method

In this section we derive an instance of the ACCPM [3] suitable for solving the problem (3). The ACCPM is a representative of the localization methods which require an initial *localization set* $\mathcal{P}_0 \subseteq \mathbb{R}^n$ containing the set of minimizers Θ^*. We construct $\mathcal{P}_0$ as an intersection of the feasible set Θ and the origin centered box whose sides are aligned with the axes, i.e.

$$\mathcal{P}_0 = \Theta \cap \{\boldsymbol{\theta} \in \mathbb{R}^n \mid |\theta_i| \leq B\,, i = 1, \ldots, n\}\,, \quad (5)$$

where the constant $B > 0$ has to be selected such that $\mathcal{P}_0 \supseteq \Theta^*$. This can be achieved by using a problem specific knowledge or by setting B to a conservatively large value. Because our feasible set Θ is an intersection of linear constraints (defining the condition of supermodularity (2)) the set $\mathcal{P}_0$ is a closed convex polyhedron which can be represented by m_0 linear inequalities, i.e. $\mathcal{P}_0 = \{\boldsymbol{\theta} \in \mathbb{R}^n \mid \boldsymbol{\theta}^T \boldsymbol{a}_i \leq b_i\,, i = 1, \ldots, m_0\}$. The idea behind localization methods is to build a series of progressively shrinking nested subsets $\mathcal{P}_0 \supset \mathcal{P}_1 \cdots \supset \mathcal{P}_t$, each being a superset of Θ^*, until $\mathcal{P}_t$ becomes a close approximation of Θ^*. Starting from the initial $\mathcal{P}_0$, the new localization sets are build iteratively by adding a single linear constrain at each iteration, i.e. $\mathcal{P}_{t+1} = \mathcal{P}_t \cap \{\boldsymbol{\theta} \in \mathbb{R}^n \mid \boldsymbol{\theta}^T \boldsymbol{a}_t \leq b_t\}$ where the added constraint is constructed so that it cuts off the half-space $\{\boldsymbol{\theta} \in$

$\mathbb{R}^n \mid \boldsymbol{\theta}^T \boldsymbol{a}_t > b_t\}$ which does not contain Θ^*. The parameters $\boldsymbol{a}_t$, b_t of the added constraint determine a hyperplane $\{\boldsymbol{\theta} \in \mathbb{R}^n \mid \boldsymbol{\theta}^T \boldsymbol{a}_t = b_t\}$, called *a cutting plane*, separating Θ^* from a query point $\boldsymbol{\theta}_t \in \mathcal{P}_t$, i.e.

$$\boldsymbol{\theta}^T \boldsymbol{a}_t \leq b_t \,, \boldsymbol{\theta} \in \Theta^* \quad \text{and} \quad \boldsymbol{\theta}_t^T \boldsymbol{a}_t \geq b_t \,, \tag{6}$$

hold. It is easy to show [1] that the parameters of the cutting plane can be computed as follows

$$\boldsymbol{a}_t = \boldsymbol{f}'(\boldsymbol{\theta}_t) \,, \quad b_t = f_t^{\text{best}} - f(\boldsymbol{\theta}_t) + \boldsymbol{f}'(\boldsymbol{\theta}_t)^T \boldsymbol{\theta}_t \,, \tag{7}$$

where $f(\boldsymbol{\theta}_t)$ and $\boldsymbol{f}'(\boldsymbol{\theta}_t) \in \mathbb{R}^n$ are the objective value and the subgradient of the objective f evaluated at the query point $\boldsymbol{\theta}_t$. The scalar f_t^{best} is the best available estimate of the optimal value typically computed as $f_t^{\text{best}} = \min_{i=0,\ldots,t} f(\boldsymbol{\theta}_t)$.

There exist several instances of the localization methods which differ in the way they select the query point $\boldsymbol{\theta}_t \in \mathcal{P}_t$. In particular, the ACCPM computes the query point as an *analytic center* of $\mathcal{P}_t$, i.e.

$$AC\left(\mathcal{P}\right) = \operatorname*{argmin}_{\boldsymbol{\theta}} - \sum_{i=1}^{m} \log\left(b_i - \boldsymbol{\theta}^T \boldsymbol{a}_i\right) \,. \tag{8}$$

As a byproduct of solving (8) for given $\mathcal{P}_t$ we can obtain a lower bound l_t on the optimal value $f(\boldsymbol{\theta}^*)$ (see [1] for details). We define the best lower bound as $l_t^{\text{best}} = \max_{i=0,\ldots,t} l_t$. The upper f_t^{best} and the lower l_t^{best} bound allows to compute the optimality gap which can be used as a rigorous stopping condition.

Algorithm 1 shows a pseudo code of the ACCPM tailored for solving the problem (3). A convergence of the ACCPM in a finite number of iterations was proved in [10]. An excellent description of a practical implementation of the ACCPM is in [1] which we followed in our work. In particular, we use the Infeasible Start Newton Method with the backtracking line search algorithm and the pruning method maintaining only $3n$ cutting planes.

Algorithm 1: ACCPM for solving (3)

Input: $\varepsilon > 0$ and $\mathcal{P}_0$ constructed by (5)
Output: ε-optimal solution $\boldsymbol{\theta}_t$
repeat
  Compute query point $\boldsymbol{\theta}_t = AC\left(\mathcal{P}_t\right)$.
  Compute cutting-plane params. $\boldsymbol{a}_t$, b_t by (6).
  Update the localization set
  $\mathcal{P}_{t+1} = \mathcal{P}_t \cap \left\{\boldsymbol{\theta} \in \mathbb{R}^n | \boldsymbol{\theta}^T \boldsymbol{a}_t \leq b_t\right\}$
  $t := t + 1$
until $1 - l_t^{\text{best}} / f_t^{\text{best}} \leq \varepsilon$;

4. Experiments

We compare the proposed ACCPM and the BMRM on the problem of image segmentation.

Setting SM-PMN model The goal is to assign each pixel of an image to a semantic class like *background, cow, horse* using the SM-PMN prediction rule (1). The objects $\mathcal{T}$ are image pixels and edges $\mathcal{E}$ describe 4-neighborhood structure on the pixels. The observation $x_t \in \mathbb{R}^3$ is an RGB color and $y_t \in \{1, \ldots, Y\}$ is the semantic label. The unary potential $q_t(x_t, y_t)$ is set to be a quadratic function of the RGB vector parametrized by 10 numbers. The pair-wise potentials $g_{tt'}(y_t, y_{t'})$ are homogeneous but different for horizontal and vertical edges. I.e. the SM-PMN has $10Y + 2Y^2$ parameters in total to be learned. We used the Hamming loss, i.e. the reported errors correspond to the percentage of misclassified pixels in the image.

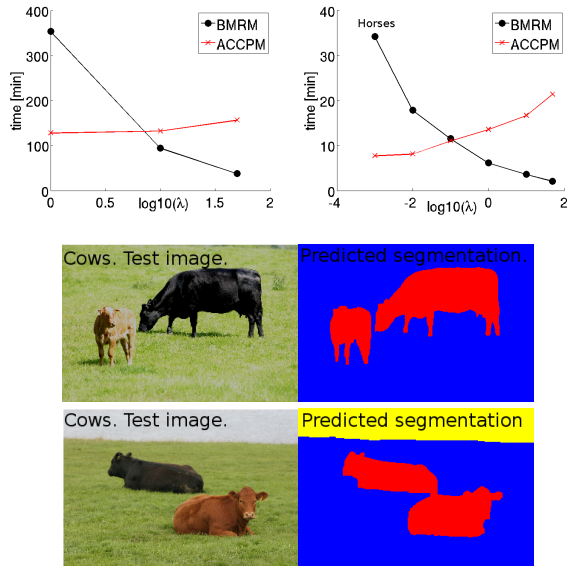

Figure 1: Top: CPU time vs. value of λ for the ACCPM and the BMRM on Cows and Horse images. Bottom: Examples of test images and estimated segmentation.

Datasets We used two sets of fully annotated images. The first set contains 32 images of cows selected from the MSRC database [9] when we improved the original annotation by GrabCut [6]. The cow images have 213×320 pixels which are annotated to three classes: *cow, grass, sky*. The second set contains 28 images of horses selected from the Weizmann Horse Database[2]. The horse images have 200×300 pixels which are annotated to two classes: *horse, background*.

Implementation We implemented the BMRM and the ACCPM in Matlab with critical parts written in C++. Both methods used the same functions for evaluation of the objective and its sub-gradient which are the most time consuming parts. In all experiments we set the precision parameter to $\varepsilon = 0.01$. The quadratic program required by the BMRM was solved by the commercial package MOSEK[3]. The experiments run on PC

[2]Available at http://www.msri.org/people/members/eranb/
[3]http://www.mosek.com/

with AMD CPU 2000 GHz and 66GB RAM.

Experiment 1: Model selection We split the annotated images to *training, validation and test* subsets using partitioning 10 + 10 + 12 for Cows and 9 + 9 + 10 for Horses. We trained the SM-PMN for a range of λ's on the training subset and evaluated on the validation subset. The SM-PMN with the minimal validation error is selected as the best and evaluated on the test subset. We repeated this process for 3 randomly chosen trn/val/tst splits and computed the mean errors and their standard deviations. The training and validation errors obtained by the ACCPM are summarized in Table 1. It is seen that the best model is obtained for $\lambda = 0$ for which the test errors are: 4.42 ± 1.27 for Cows and 17.80 ± 5.47 for Horses. We repeated the same experiment with the BMRM, however, only for non-zero λ's because $\lambda \to 0$ prevents the BMRM from convergence. The training and validation errors for the BMRM are almost identical to the values for ACCPM and hence not reported. The test errors obtained for the lowest λ the BMRM could manage were: 6.50 ± 2.72 for Cows and 18.41 ± 3.98 for Horse, i.e. clearly higher than the ACCPM errors.

	Horses		Cows	
λ	TRN%	VAL%	TRN%	VAL%
50	25.70 ± 5.53	25.59 ± 5.39	26.93 ± 3.29	31.65 ± 5.11
10	24.04 ± 5.54	24.33 ± 6.22	21.06 ± 4.81	28.87 ± 6.12
1	14.26 ± 3.15	18.91 ± 2.93	2.90 ± 0.91	6.25 ± 1.73
0	11.44 ± 3.66	17.25 ± 3.27	1.55 ± 0.32	4.29 ± 1.74

Table 1: Training and validations accuracy for different value of λ for Horse and Cow images.

Experiment 2: Training time In this experiment we learned the SM-PMN from all images using the ACCPM and the BMRM. For a range of λ we measured the CPU time, the number of iterations and the final objective value for both methods. The results are summarized in Table 2 and in Figure 1. Due to the low value of the precision parameter both methods achieve the same precise solution as can be seen from the values of the objective function, however, the computational time is significantly different. The CPU time of the ACCPM scales gracefully with varying value of λ. The CPU time of the BMRM soars steeply for $\lambda \to 0$. On Cow images the BMRM did not converge after 500 even for $\lambda = 1$ still much higher than optimal $\lambda = 0$. The ACCPM not only allows to train with arbitrary low λ but it requires considerably lower CPU time. The number of iterations, which is an implementation invariant performance measure, behaves similarly as the CPU time.

5. Conclusions

We have proposed the ACCPM algorithm for learning the SM-PMN. Experiments show that the proposed ACCPM outperforms the so far state-of-the-art BMRM in terms of computational time for problems which require small regularization constant. The proposed ACCPM is able to work completely without regularization which is not possible by the BMRM yet it is useful in problems with many examples and few parameters. The proposed algorithm is generic and can be easily adapted for learning of other structured output models.

	ACCPM			BMRM		
λ	Time [min]	Iter	$f(\boldsymbol{\theta})$	Time [min]	Iter	$f(\boldsymbol{\theta})$
Horse images						
50	21.42	152	57.29	2.10	26	57.36
10	16.71	129	55.65	3.62	44	55.75
1	13.60	104	51.21	6.14	76	51.32
0	7.38	62	46.11			
Cow images						
50	156.76	266	57.56	37.77	54	57.76
10	132.60	240	49.96	94.65	112	50.10
1	128.07	235	25.17	353.59	500*	25.43
0	63.26	165	5.52			

Table 2: CPU time, number of iterations and objective value vs. value of λ for the ACCPM and the BMRM on Horse and Cow images; * means not converged.

Acknowledgments The authors were supported by EC projects FP7-ICT-247525 HUMAVIPS and PERG04-GA-2008-239455 SEMISOL and the Visegrad Scholarship contract No. 51100258.

References

[1] S. Boyd. Analytic center cutting-plane method. Stanford University, California, USA, Unpublished lecture notes, http://www.stanford.edu/class/ee364b/, 2008.

[2] V. Franc and B. Savchynskyy. Discriminative learning of max-sum classifiers. *Journal of Machine Learning Research*, 9:67–104, 2008.

[3] J. Goffin and J. Vial. On the computation of weighted analytic centers and dual ellipsoids with the projective algorithm. *Math. Program.*, 60:81–92, 1993.

[4] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[5] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, 2004.

[6] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, Aug. 2004.

[7] D. Schlesinger and B. Flach. Transforming an arbitrary minsum problem into a binary one. Technical report, Dresden University of Technology, 2006.

[8] C. H. Teo, S. Vishwanthan, A. J. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. *J. Mach. Learn. Res.*, 11:311–365, 2010.

[9] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proc. IEEE Intl. Conf. on Computer Vision (ICCV)*, 2005.

[10] Y. Ye. Complexity analysis of the analytic center cutting plane method that uses multiple cuts. *Mathematical Programming*, 78:85–104, 1997.