Optimization for Machine Learning

Editors:

Suvrit Sra suvrit@gmail.com
Max Planck Insitute for Biological Cybernetics
72076 Tübingen, Germany

Sebastian Nowozin nowozin@gmail.com
Microsoft Research
Cambridge, CB3 0FB, United Kingdom

Stephen J. Wright swright@cs.uwisc.edu
University of Wisconsin
Madison, WI 53706

This is a draft containing only `sra_chapter.tex` and an abbreviated front matter. Please check that the formatting and small changes have been performed correctly. Please verify the affiliation. Please use this version for sending us future modifications.

The MIT Press
Cambridge, Massachusetts
London, England

Contents

1 Cutting plane methods in machine learning

Vojtěch Franc xfrancv@cmp.felk.cvut.cz
Czech Technical University in Prague
Technická 2, 166 27 Prague 6
Czech Republic

Sören Sonnenburg Soeren.Sonnenburg@tu-berlin.de
Berlin Institute of Technology
Franklinstr. 28/29
10587 Berlin, Germany

Tomáš Werner werner@cmp.felk.cvut.cz
Czech Technical University in Prague
Technická 2, 166 27 Prague 6
Czech Republic

Cutting plane methods are optimization techniques that incrementally construct an approximation of a feasible set or an objective function by linear inequalities, called cutting planes. Numerous variants of this basic idea are among standard tools used in convex nonsmooth optimization and integer linear programing. Recently, cutting plane methods have seen growing interest in the field of machine learning. In this chapter, we describe the basic theory behind these methods and we show three of their successful applications to solving machine learning problems: regularized risk minimization, multiple kernel learning, and MAP inference in graphical models.

Many problems in machine learning are elegantly translated to convex optimization problems, which, however, are sometimes difficult to solve efficiently by off-the-shelf solvers. This difficulty can stem from complexity of either the feasible set or of the objective function. Often, these can be accessed only indirectly via an oracle. To access a feasible set, the oracle either asserts that a given query point lies in the set or finds a hyperplane that separates the point from the set. To access an objective function, the oracle returns the value and a subgradient of the function at the query point. Cutting plane methods solve the optimization problem by approximating the feasible set or the objective function by a bundle of linear inequalities, called cutting planes. The approximation is iteratively refined by adding new cutting planes, computed from the responses of the oracle.

Cutting plane methods have been extensively studied in literature. We refer to Boyd and Vandenberge (2008) for an introductory yet comprehensive overview. For the sake of self consistency, we review the basic theory in Section 1.1. Then, in three separate sections, we describe their successful applications to three machine learning problems.

The first application, Section 1.2, is on learning linear predictors from data based on *regularized risk minimization* (RRM). RRM often leads to a convex but nonsmooth task, which cannot be efficiently solved by general-purpose algorithms, especially for large-scale data. Prominent examples of RRM are support vector machines, logistic regression, and structured output learning. We review a generic risk minimization algorithm proposed by Teo et al. (2007, 2010), inspired by a variant of cutting plane methods known as proximal bundle methods. We also discuss its accelerated version (Franc and Sonnenburg, 2008, 2010; Teo et al., 2010), which is among the fastest solvers for the large-scale learning.

The second application, Section 1.3, is *multiple kernel learning* (MKL). While classical kernel-based learning algorithms use a single kernel, it is sometimes desirable to use multiple kernels (Lanckriet et al., 2004b). Here, we focus on the convex formulation of the MKL problem for classification as first stated in (Zien and Ong, 2007; Rakotomamonjy et al., 2007). We show how this problem can be efficiently solved by a cutting plane algorithm recycling standard SVM implementations. The resulting MKL solver is equivalent to the column generation approach applied to the semi-infinite programming formulation of the MKL problem proposed by Sonnenburg et al. (2006a).

The third application, Section 1.4, is *maximum a posteriori (MAP) inference in graphical models*. It leads to a combinatorial optimization problem which can be formulated as a linear optimization over the marginal polytope (Wainwright and Jordan, 2008). Cutting plane methods iteratively construct a sequence of progressively tighter outer bounds of the marginal polytope, corresponding to a sequence of LP relaxations. We revisit the approach by Werner (2008a, 2010), in which a dual cutting plane method is a straightforward extension of a simple message passing algorithm. It is a generalization of the dual LP relaxation approach by Shlezinger (1976) and the max-sum diffusion algorithm by Kovalevsky and Koval (approx. 1975).

1.1 Introduction to cutting plane methods

Suppose we want to solve the optimization problem

$$\min\{\, f(\boldsymbol{x}) \mid \boldsymbol{x} \in X \,\}\,, \tag{1}$$

where $X \subseteq \mathbb{R}^n$ is a convex set, $f\colon \mathbb{R}^n \to \mathbb{R}$ is a convex function, and we assume that the minimum exists. Set X can be accessed only via the so called *separation oracle* (or *separation algorithm*). Given $\hat{\boldsymbol{x}} \in \mathbb{R}^n$, the separation oracle either asserts that $\hat{\boldsymbol{x}} \in X$ or returns a hyperplane $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \leq b$ (called a *cutting plane*) that separates $\hat{\boldsymbol{x}}$ from X, i.e., $\langle \boldsymbol{a}, \hat{\boldsymbol{x}} \rangle > b$ and $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \leq b$ for all $\boldsymbol{x} \in X$. Figure 1.1(a) illustrates the idea.

cutting plane algorithm

The *cutting plane algorithm* (Algorithm 1.1) solves (1) by constructing progressively tighter convex polyhedrons X_t containing the true feasible set X, by cutting off infeasible parts of an initial polyhedron X_0. It stops when $\boldsymbol{x}_t \in X$ (possibly up to some tolerance).

The trick behind the method is not to approximate X well by a convex polyhedron but to do so *only near the optimum*. This is best seen if X is already a convex polyhedron, described by a set of linear inequalities. At optimum, only some of the inequalities are active. We could in fact remove all the inactive inequalities without affecting the problem. Of course, we do not know which ones to remove until we know the optimum. The cutting plane algorithm imposes more than the minimal set of inequalities but still possibly much fewer than the whole original description of X.

Algorithm 1.1 Cutting plane algorithm

```
Initialization: $t \leftarrow 0$, $X_0 \supseteq X$
loop
    Let $\boldsymbol{x}_t \in \operatorname{argmin}_{\boldsymbol{x} \in X_t} f(\boldsymbol{x})$
    If $\boldsymbol{x}_t \in X$ then stop, else find a cutting plane $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \leq b$ separating $\boldsymbol{x}_t$ from $X$.
    $X_{t+1} \leftarrow X_t \cap \{\, \boldsymbol{x} \mid \langle \boldsymbol{a}, \boldsymbol{x} \rangle \leq b \,\}$
    $t \leftarrow t + 1$
end loop
```

This basic idea has many incarnations. Next we describe three of them, which have been used in the three machine learning applications presented in this chapter. Section 1.1.1 describes a cutting plane method suited for minimization of nonsmooth convex functions. An improved variant thereof, called the *bundle method*, is described in Section 1.1.2. Finally, Section 1.1.3 describes application of cutting plane methods to solving combinatorial optimization problems.

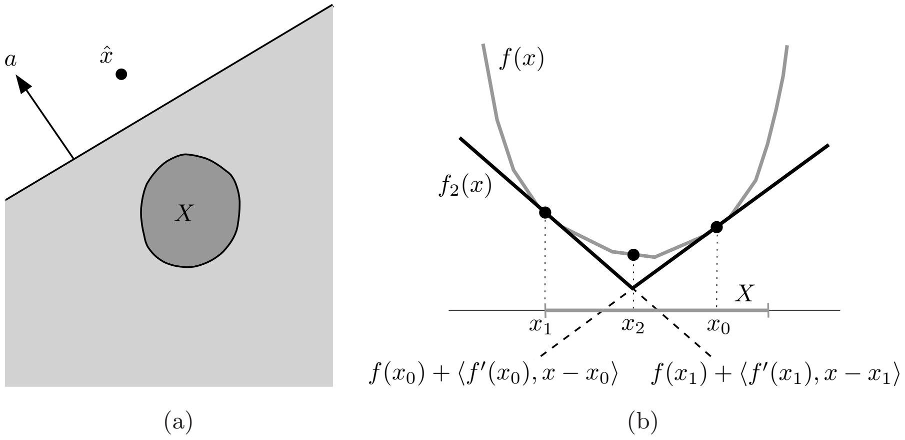

a
$\hat{x}$
X
$f(x)$
$f_2(x)$
X
x_1
x_2
x_0
$f(x_0) + \langle f'(x_0), x - x_0\rangle$
$f(x_1) + \langle f'(x_1), x - x_1\rangle$
(a)
(b)

Figure 1.1: Figure (a) illustrates the cutting plane $\langle \boldsymbol{a}, \boldsymbol{x}\rangle \le b$ cutting off the query point $\hat{\boldsymbol{x}}$ from the light gray halfspace $\{\boldsymbol{x} \mid \langle \boldsymbol{a}, \boldsymbol{x}\rangle \le b\}$ which contains the feasible set X (dark gray). Figure (b) shows a feasible set X (gray interval) and a function $f(\boldsymbol{x})$ which is approximated by a cutting plane model $f_2(\boldsymbol{x}) = \max\{f(\boldsymbol{x}_0) + \langle f'(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0\rangle, f(\boldsymbol{x}_1) + \langle f'(\boldsymbol{x}_1), \boldsymbol{x} - \boldsymbol{x}_1\rangle\}$. Starting from $\boldsymbol{x}_0$, the CPA generates points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2 = \operatorname{argmin}_{\boldsymbol{x}\in X} f_2(\boldsymbol{x})$.

1.1.1 Nonsmooth optimization

When f is a complicated nonsmooth function while the set X is simple, we want to avoid explicit minimization of f in the algorithm. This can be done by writing (1) in the epigraph form as

$$\min\{\, y \mid (\boldsymbol{x}, y) \in Z \,\} \quad \text{where} \quad Z = \{\, (\boldsymbol{x}, y) \in X \times \mathbb{R} \mid f(\boldsymbol{x}) \le y \,\} \; . \tag{2}$$

subgradient

In this case, cutting planes can be generated by means of subgradients. Recall that $f'(\hat{\boldsymbol{x}}) \in \mathbb{R}^n$ is a subgradient of f at $\hat{\boldsymbol{x}}$ if

$$f(\boldsymbol{x}) \ge f(\hat{\boldsymbol{x}}) + \langle f'(\hat{\boldsymbol{x}}), \boldsymbol{x} - \hat{\boldsymbol{x}}\rangle \, , \qquad \boldsymbol{x} \in X \; . \tag{3}$$

Thus, the right-hand side is a linear underestimator of f. Assume that $\hat{\boldsymbol{x}} \in X$. Then, the separation algorithm for the set Z can be constructed as follows. If $f(\hat{\boldsymbol{x}}) \le \hat{y}$ then $(\hat{\boldsymbol{x}}, \hat{y}) \in Z$. If $f(\hat{\boldsymbol{x}}) > \hat{y}$ then the inequality

$$y \ge f(\hat{\boldsymbol{x}}) + \langle f'(\hat{\boldsymbol{x}}), \boldsymbol{x} - \hat{\boldsymbol{x}}\rangle \tag{4}$$

defines a cutting plane separating $(\hat{\boldsymbol{x}}, \hat{y})$ from Z.

This leads to the algorithm proposed independently by Cheney and Goldstein (1959) and Kelley (1960). Starting with $\boldsymbol{x}_0 \in X$, it computes the next iterate $\boldsymbol{x}_t$ by solving

$$\begin{aligned}&(\boldsymbol{x}_t, y_t) \in \operatorname*{argmin}_{(\boldsymbol{x},y)\in Z_t} y \qquad \text{where} \\ &Z_t = \left\{ (\boldsymbol{x}, y) \in X \times \mathbb{R} \mid y \geq f(\boldsymbol{x}_i) + \langle f'(\boldsymbol{x}_i),\, \boldsymbol{x} - \boldsymbol{x}_i \rangle, i = 0, \dots, t-1 \right\} .\end{aligned} \tag{5}$$

Here, Z_t is a polyhedral outer bound of Z defined by X and the cutting planes from previous iterates $\{\boldsymbol{x}_0, \dots, \boldsymbol{x}_{t-1}\}$. Problem (5) simplifies to

$$\boldsymbol{x}_t \in \operatorname*{argmin}_{\boldsymbol{x}\in X} f_t(\boldsymbol{x}) \quad \text{where} \quad f_t(\boldsymbol{x}) = \max_{i=0,\dots,t-1} \left[f(\boldsymbol{x}_i) + \langle f'(\boldsymbol{x}_i),\, \boldsymbol{x} - \boldsymbol{x}_i \rangle \right] . \tag{6}$$

Here, f_t is a *cutting-plane model* of f (see Figure 1.1(b)). Note that $(\boldsymbol{x}_t, f_t(\boldsymbol{x}_t))$ solves (5). By (3) and (6), we have that $f(\boldsymbol{x}_i) = f_t(\boldsymbol{x}_i)$ for $i = 0, \dots, t-1$ and $f(\boldsymbol{x}) \geq f_t(\boldsymbol{x})$ for $\boldsymbol{x} \in X$, i.e., f_t is an underestimator of f which touches f at the points $\{\boldsymbol{x}_0, \dots, \boldsymbol{x}_{t-1}\}$. By solving (6) we do not only get an estimate $\boldsymbol{x}_t$ of the optimal point $\boldsymbol{x}^*$ but also a lower bound $f_t(\boldsymbol{x}_t)$ on the optimal value $f(\boldsymbol{x}^*)$. It is natural to terminate when $f(\boldsymbol{x}_t) - f_t(\boldsymbol{x}_t) \leq \varepsilon$, which guarantees that $f(\boldsymbol{x}_t) \leq f(\boldsymbol{x}^*) + \varepsilon$. The method is summarized in Algorithm 1.2.

Algorithm 1.2 Cutting plane algorithm in epigraph form

```
Initialization: $t \leftarrow 0$, $\boldsymbol{x}_0 \in X$, $\varepsilon > 0$
repeat
    $t \leftarrow t + 1$
    Compute $f(\boldsymbol{x}_{t-1})$ and $f'(\boldsymbol{x}_{t-1})$.
    Update the cutting plane model $f_t(\boldsymbol{x}) \leftarrow \max_{i=0,\dots,t-1} \left[ f(\boldsymbol{x}_i) + \langle f'(\boldsymbol{x}_i),\, \boldsymbol{x} - \boldsymbol{x}_i \rangle \right]$
    Let $\boldsymbol{x}_t \in \operatorname{argmin}_{\boldsymbol{x}\in X} f_t(\boldsymbol{x})$.
until $f(\boldsymbol{x}_t) - f_t(\boldsymbol{x}_t) \leq \varepsilon$
```

In Section 1.3, this algorithm is applied to *multiple kernel learning*. This requires solving the problem

$$\min\{\, f(\boldsymbol{x}) \mid \boldsymbol{x} \in X \,\} \qquad \text{where} \qquad f(\boldsymbol{x}) = \max\{\, g(\boldsymbol{\alpha}, \boldsymbol{x}) \mid \boldsymbol{\alpha} \in A \,\} . \tag{7}$$

X is a simplex and function g is linear in $\boldsymbol{x}$ and quadratic negative semi-definite in $\boldsymbol{\alpha}$. In this case, the subgradient $f'(\boldsymbol{x})$ equals the gradient $\nabla_{\boldsymbol{x}} g(\hat{\boldsymbol{\alpha}}, \boldsymbol{x})$ where $\hat{\boldsymbol{\alpha}}$ is obtained by solving a convex quadratic program $\hat{\boldsymbol{\alpha}} \in \operatorname{argmax}_{\boldsymbol{\alpha}\in A}\, g(\boldsymbol{\alpha}, \boldsymbol{x})$.

1.1.2 Bundle methods

Algorithm 1.2 may converge slowly (Nemirovskij and Yudin, 1983) because subsequent solutions can be very distant, exhibiting a zig-zag behavior, thus many cutting planes do not actually contribute to the approximation of f

around the optimum $\boldsymbol{x}^*$. Bundle methods (Kiwiel, 1983; Lemaréchal et al., 1995) try to reduce this behavior by adding a stabilization term to (6). The *proximal bundle methods* compute the new iterate as

proximal bundle methods

$$\boldsymbol{x}_t \in \operatorname*{argmin}_{\boldsymbol{x} \in X} \{ \, \nu_t \|\boldsymbol{x} - \boldsymbol{x}_t^+\|_2^2 + f_t(\boldsymbol{x}) \, \} \; ,$$

where $\boldsymbol{x}_t^+$ is a current prox-center selected from $\{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{t-1}\}$ and ν_t is a current stabilization parameter. The added quadratic term ensures that the subsequent solutions are within a ball centered at $\boldsymbol{x}_t^+$ whose radius depends on ν_t. If $f(\boldsymbol{x}_t)$ sufficiently decreases the objective, the *decrease step* is performed by moving the prox-center as $\boldsymbol{x}_{t+1}^+ := \boldsymbol{x}_t$. Otherwise, the *null step* is performed, $\boldsymbol{x}_{t+1}^+ := \boldsymbol{x}_t^+$. If there is an efficient line-search algorithm, the decrease step computes the new prox-center $\boldsymbol{x}_{t+1}^+$ by minimizing f along the line starting at $\boldsymbol{x}_t^+$ and passing through $\boldsymbol{x}_t$. Though bundle methods may improve the convergence significantly they require two parameters: the stabilization parameter ν_t and the minimal decrease in the objective which defines the null step. Despite significantly influencing the convergence, there is no versatile method for choosing these parameters optimally.

In Section 1.2, a variant of this method is applied to *regularized risk minimization* which requires minimizing $f(\boldsymbol{x}) = g(\boldsymbol{x}) + h(\boldsymbol{x})$ over $\mathbb{R}^n$ where g is a simple (typically differentiable) function and h is a complicated nonsmooth function. In this case, the difficulties with setting two parameters are avoided because g naturally plays the role of the stabilization term.

1.1.3 Combinatorial optimization

A typical combinatorial optimization problem can be formulated as

$$\min\{ \, \langle \boldsymbol{c}, \boldsymbol{x} \rangle \mid \boldsymbol{x} \in C \, \} \, , \tag{8}$$

where $C \subseteq \mathbb{Z}^n$ (often just $C \subseteq \{0,1\}^n$) is a finite set of feasible configurations, and $\boldsymbol{c} \in \mathbb{R}^n$ is a cost vector. Usually C is combinatorially large but highly structured. Consider the problem

$$\min\{ \, \langle \boldsymbol{c}, \boldsymbol{x} \rangle \mid \boldsymbol{x} \in X \, \} \quad \text{where} \quad X = \operatorname{conv} C \; . \tag{9}$$

Clearly, X is a polytope (bounded convex polyhedron) with integral vertices. Hence, (9) is a linear program. Since a solution of a linear program is always attained at a vertex, problems (8) and (9) have the same optimal value. The set X is called the *integral hull* of problem (8).

Integral hulls of hard problems are complex. If a problem (8) is not polynomially solvable then inevitably the number of facets of X is not polynomial. Therefore (9) cannot be solved explicitly. This is where Algorithm 1.1 is used. The initial polyhedron $X_0 \supseteq X$ is described by a tractable number of linear inequalities and usually it is already a good approximation of X, often but not necessarily we also have $X_0 \cap \mathbb{Z}^n = C$. The cutting plane algorithm then constructs a sequence of gradually tighter LP relaxations of (8).

A fundamental result states that a linear optimization problem and the corresponding separation problem are polynomial-time equivalent (Grötschel et al., 1981). Therefore, for an intractable problem (8) there is no hope to find a polynomial algorithm to separate an arbitrary point from X. However, a polynomial separation algorithm may exist for a *subclass* (even intractably large) of linear inequalities describing X.

After this approach was first proposed by Dantzig et al. (1954) for the travelling salesman problem, it became a breakthrough in tackling hard combinatorial optimization problems. Since then much effort has been devoted to finding good initial LP relaxations X_0 for many such problems, subclasses of inequalities describing integral hulls for these problems, and polynomial separation algorithms for these subclasses. This is the subject of *polyhedral combinatorics* (e.g., Schrijver, 2003).

In Section 1.4, we focus on the NP-hard combinatorial optimization problem arising in MAP inference in graphical models. This problem, in its full generality, has not been properly addressed by the optimization community. We show how its LP relaxation can be incrementally tightened during a message passing algorithm. Because message passing algorithms are dual, this can be understood as a *dual* cutting plane algorithm: it does not add constraints in the primal but variables in the dual. The sequence of approximations of the integral hull X (the marginal polytope) can be seen as arising from lifting and projection.

1.2 Regularized risk minimization

Learning predictors from data is a standard machine learning problem. A wide range of such problems are special instances of the *regularized risk minimization.* In this case, learning is often formulated as an unconstrained minimization of a convex function

$$\boldsymbol{w}^* \in \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^n} F(\boldsymbol{w}) \qquad \text{where} \qquad F(\boldsymbol{w}) = \lambda\Omega(\boldsymbol{w}) + R(\boldsymbol{w})\,. \tag{10}$$

The objective $F\colon \mathbb{R}^n \to \mathbb{R}$, called *regularized risk*, is composed of a regularization term $\Omega\colon \mathbb{R}^n \to \mathbb{R}$ and empirical risk $R\colon \mathbb{R}^n \to \mathbb{R}$ which are both convex functions. The number $\lambda \in \mathbb{R}_+$ is a predefined regularization constant and $\boldsymbol{w} \in \mathbb{R}^n$ is a parameter vector to be learned. The regularization term Ω

is typically a simple, cheap-to-compute function used to constrain the space of solutions in order to improve generalization. The empirical risk R evaluates how well the parameters $\boldsymbol{w}$ explains the training examples. Evaluation of R is often computationally expensive.

Example 1.1. *Given a set of training examples* $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathbb{R}^n \times \{+1, -1\})^m$*, the goal is to learn a parameter vector* $\boldsymbol{w} \in \mathbb{R}^n$ *of a linear classifier* $h \colon \mathbb{R}^n \to \{-1, +1\}$ *which returns* $h(\boldsymbol{x}) = +1$ *if* $\langle \boldsymbol{x}, \boldsymbol{w} \rangle \geq 0$ *and* $h(\boldsymbol{x}) = -1$ *otherwise. Linear support vector machines (Cortes and Vapnik, 1995) without bias learn the parameter vector* $\boldsymbol{w}$ *by solving (10) with the regularization term* $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$ *and the empirical risk* $R(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle\}$ *which, in this case, is a convex upper bound on the number of mistakes the classifier* $h(\boldsymbol{x})$ *makes on the training examples.*

There is a long list of learning algorithms which in their core are solvers of a special instance of (10), see, e.g. Schölkopf and Smola (2002). If F is differentiable, (10) is solved by algorithms for a smooth optimization. If F is nonsmooth, (10) is typically transformed to an equivalent problem solvable by off-the-shelf methods. For example, learning of the linear SVM classifier in Example 1.1 can be equivalently expressed as quadratic program. Because off-the-shelf solvers are often not efficient enough in practice a huge effort has been put into development of specialized algorithms tailored to particular instances of (10).

Teo et al. (2007, 2010) proposed a generic algorithm to solve (10) which is a modification of the proximal bundle methods. The algorithm, called *bundle method for risk minimization* (BMRM), exploits the specific structure of the objective F in (10). In particular, only the risk term R is approximated by the cutting-plane model while the regularization term Ω is without any change used to stabilize the optimization. In contrast, standard bundle methods introduce the stabilization term artificially. The resulting BMRM is highly modular and was proven to converge in $\mathcal{O}(\frac{1}{\varepsilon})$ iterations to an ε-precise solution. In addition, if an efficient line-search algorithm is available, BMRM can be drastically accelerated with a technique proposed by Franc and Sonnenburg (2008, 2010); Teo et al. (2010). The accelerated BMRM has been shown to be highly competitive with state-of-the-art solvers tailored to particular instances of (10).

In the next two sections, we describe BMRM algorithm and its version accelerated by line-search.

Algorithm 1.3 Bundle Method for Regularized Risk Minimization (BMRM)

```
input & initialization: $\varepsilon > 0$, $\boldsymbol{w}_0 \in \mathbb{R}^n$, $t \leftarrow 0$
repeat
    $t \leftarrow t + 1$
    Compute $R(\boldsymbol{w}_{t-1})$ and $R'(\boldsymbol{w}_{t-1})$
    Update the model $R_t(\boldsymbol{w}) \leftarrow \max_{i=0,\ldots,t-1} R(\boldsymbol{w}_i) + \langle R'(\boldsymbol{w}_i),\, \boldsymbol{w} - \boldsymbol{w}_i \rangle$
    Solve the reduced problem $\boldsymbol{w}_t \leftarrow \operatorname{argmin}_{\boldsymbol{w}} F_t(\boldsymbol{w})$ where $F_t(\boldsymbol{w}) = \lambda\Omega(\boldsymbol{w}) + R_t(\boldsymbol{w})$
until $F(\boldsymbol{w}_t) - F_t(\boldsymbol{w}_t) \le \varepsilon$
```

1.2.1 Bundle method for regularized risk minimization

Following optimization terminology, we will call (10) the *master problem.* Using the approach by Teo et al. (2007), one can approximate the master problem (10) by its *reduced problem*

$$\boldsymbol{w}_t \in \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^n} F_t(\boldsymbol{w}) \qquad \text{where} \qquad F_t(\boldsymbol{w}) = \lambda\Omega(\boldsymbol{w}) + R_t(\boldsymbol{w})\,. \tag{11}$$

The reduced problem (11) is obtained from the master problem (10) by substituting the cutting-plane model R_t for the empirical risk R while the regularization term Ω remains unchanged. The cutting-plane model reads

$$R_t(\boldsymbol{w}) = \max_{i=0,\ldots,t-1} \big[R(\boldsymbol{w}_i) + \langle R'(\boldsymbol{w}_i),\, \boldsymbol{w} - \boldsymbol{w}_i\rangle\big]\,, \tag{12}$$

where $R'(\boldsymbol{w}) \in \mathbb{R}^n$ is a subgradient of R at point $\boldsymbol{w}$. Since $R(\boldsymbol{w}) \ge R_t(\boldsymbol{w})$, $\forall \boldsymbol{w} \in \mathbb{R}^n$, the reduced problem's objective F_t is an underestimator of the master objective F. Starting from $\boldsymbol{w}_0 \in \mathbb{R}^n$, BMRM of Teo et al. (2007) (Algorithm 1.3) computes a new iterate $\boldsymbol{w}_t$ by solving the reduced problem (11). In each iteration t, the cutting-plane model (12) is updated by a new cutting plane computed at the intermediate solution $\boldsymbol{w}_t$ leading to a progressively tighter approximation of F. The algorithm halts if the gap between the upper bound $F(\boldsymbol{w}_t)$ and the lower bound $F_t(\boldsymbol{w}_t)$ falls bellow a desired ε, meaning that $F(\boldsymbol{w}_t) \le F(\boldsymbol{w}^*) + \varepsilon$.

Solving the reduced problem

In practice, the number of cutting planes t required before the algorithm converges is typically much lower than the dimension n of the parameter vector $\boldsymbol{w} \in \mathbb{R}^n$. Thus, it is beneficial to solve the reduced problem (11) in its dual formulation. Let $A = [\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{t-1}] \in \mathbb{R}^{n\times t}$ be a matrix whose columns are the subgradients $\boldsymbol{a}_i = R'(\boldsymbol{w}_i)$ and let $\boldsymbol{b} = [b_0, \ldots, b_{t-1}] \in \mathbb{R}^t$ be a column vector whose components equal to $b_i = R(\boldsymbol{w}_i) - \langle R'(\boldsymbol{w}_i),\, \boldsymbol{w}_i\rangle$. Then the reduced problem (11) can be equivalently expressed as

$$\boldsymbol{w}_t \in \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^n, \xi\in\mathbb{R}} \big[\lambda\Omega(\boldsymbol{w}) + \xi\big] \quad \text{s.t.} \quad \xi \ge \langle \boldsymbol{w},\, \boldsymbol{a}_i\rangle + b_i\,,\ i = 0, \ldots, t-1\,. \tag{13}$$

The Lagrange dual of (13) reads (Teo et al., 2010, Theorem 2)

$$\boldsymbol{\alpha}_t \in \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^t} \big[-\lambda\Omega^*(-\lambda^{-1}A\boldsymbol{\alpha}) + \langle\boldsymbol{\alpha},\, \boldsymbol{b}\rangle\big] \quad \text{s.t.} \;\; \|\boldsymbol{\alpha}\|_1 = 1\,, \boldsymbol{\alpha} \geq 0\,, \tag{14}$$

where $\Omega^*\colon \mathbb{R}^n \to \mathbb{R}^t$ denotes the Fenchel dual of Ω defined as

$$\Omega^*(\boldsymbol{\mu}) = \sup\big\{\langle\boldsymbol{w},\, \boldsymbol{\mu}\rangle - \Omega(\boldsymbol{w}) \;\big|\; \boldsymbol{w}\in\mathbb{R}^n\big\}\,.$$

Having the dual solution $\boldsymbol{\alpha}_t$, the primal solution can be computed by solving $\boldsymbol{w}_t \in \operatorname{argmax}_{\boldsymbol{w}\in\mathbb{R}^n}\big[\langle\boldsymbol{w},\, -\lambda^{-1}A\boldsymbol{\alpha}_t\rangle - \Omega(\boldsymbol{w})\big]$ which for differentiable Ω simplifies to $\boldsymbol{w}_t = \nabla_{\boldsymbol{\mu}}\Omega^*(-\lambda^{-1}A\boldsymbol{\alpha}_t)$.

Example 1.2. *For the quadratic regularizer $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$ the Fenchel dual reads $\Omega^*(\boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{\mu}\|_2^2$. The dual reduced problem (14) boils down to the quadratic program*

$$\boldsymbol{\alpha}_t \in \operatorname*{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^t}\left[-\frac{1}{2\lambda}\boldsymbol{\alpha}^T A^T A\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\boldsymbol{b}\right] \quad s.t. \quad \|\boldsymbol{\alpha}\|_1 = 1\,, \boldsymbol{\alpha}\geq 0$$

and the primal solution can be computed analytically by $\boldsymbol{w}_t = -\lambda^{-1}A\boldsymbol{\alpha}_t$.

Convergence guarantees

The convergence of Algorithm 1.3 in a finite number of iterations is guaranteed by the following theorem:

Theorem 1.3. *(Teo et al., 2010, Theorem 5) Assume that (i) $F(\boldsymbol{w}) \geq 0$, $\forall\boldsymbol{w}\in\mathbb{R}^n$, (ii) $\max_{\boldsymbol{g}\in\partial R(\boldsymbol{w})}\|\boldsymbol{g}\|_2 \leq G$ for all $\boldsymbol{w}\in\{\boldsymbol{w}_0,\ldots,\boldsymbol{w}_{t-1}\}$ where $\partial R(\boldsymbol{w})$ denotes the subdifferential of R at point $\boldsymbol{w}$, and (iii) Ω^* is twice differentiable and has bounded curvature, that is, $\|\partial^2\Omega^*(\boldsymbol{\mu})\| \leq H^*$ for all $\boldsymbol{\mu}\in\{\,\boldsymbol{\mu}'\in\mathbb{R}^t \mid \boldsymbol{\mu}' = \lambda^{-1}A\boldsymbol{\alpha}\;,\|\boldsymbol{\alpha}\|_1 = 1\;,\boldsymbol{\alpha}\geq 0\}$ where $\partial^2\Omega^*(\boldsymbol{\mu})$ is the Hessian of Ω^* at point $\boldsymbol{\mu}$. Then Algorithm 1.3 terminates after at most*

$$T \leq \log_2\frac{\lambda F(0)}{G^2H^*} + \frac{8G^2H^*}{\lambda\varepsilon} - 1$$

iterations for any $\varepsilon < 4G^2H^*\lambda^{-1}$.

Furthermore, for a twice differentiable F with bounded curvature Algorithm 1.3 requires only $\mathcal{O}(\log\frac{1}{\varepsilon})$ iterations instead of $\mathcal{O}(\frac{1}{\varepsilon})$ (Teo et al., 2010, Theorem 5). The most constraining assumption of Theorem 1.3 is that it requires Ω^* to be twice differentiable. This assumption holds, e.g., for the quadratic $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$ and the negative entropy $\Omega(\boldsymbol{w}) = \sum_{i=1}^n w_i\log w_i$ regularizers. Unfortunately, the theorem does not apply for the ℓ_1-norm regularizer $\Omega(\boldsymbol{w}) = \|\boldsymbol{w}\|_1$ often used to enforce sparse solutions.

1.2.2 BMRM algorithm accelerated by line-search

BMRM can be drastically accelerated whenever an efficient line-search algorithm for the master objective F is available. An accelerated BMRM for solving linear SVM problem (c.f. Example 1.1) has been first proposed in Franc and Sonnenburg (2008). Franc and Sonnenburg (2010) generalized the method for solving (10) with an arbitrary risk R and quadratic regularizer $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$. Finally, Teo et al. (2010) proposed a fully general version imposing no restrictions on Ω and R. BMRM accelerated by the line-search, in Teo et al. (2010) called LS-BMRM, is described by Algorithm 1.4.

Algorithm 1.4 BMRM accelerated by line-search (LS-BMRM)

```
1:  input & initialization: $\varepsilon \geq 0$, $\theta \in (0,1]$, $\boldsymbol{w}_0^b$, $\boldsymbol{w}_0^c \leftarrow \boldsymbol{w}_0^b$, $t \leftarrow 0$
2:  repeat
3:      $t \leftarrow t + 1$
4:      Compute $R(\boldsymbol{w}_{t-1}^c)$ and $R'(\boldsymbol{w}_{t-1}^c)$
5:      Update the model $R_t(\boldsymbol{w}) \leftarrow \max_{i=1,\ldots,t-1} R(\boldsymbol{w}_i^c) + \langle R'(\boldsymbol{w}_i^c), \boldsymbol{w} - \boldsymbol{w}_i^c \rangle$
6:      $\boldsymbol{w}_t \leftarrow \operatorname{argmin}_{\boldsymbol{w}} F_t(\boldsymbol{w})$ where $F_t(\boldsymbol{w}) = \lambda\Omega(\boldsymbol{w}) + R_t(\boldsymbol{w})$
7:      Line-search: $k_t \leftarrow \operatorname{argmin}_{k \geq 0} F(\boldsymbol{w}_t^b + k(\boldsymbol{w}_t - \boldsymbol{w}_{t-1}^b))$
8:      $\boldsymbol{w}_t^b \leftarrow \boldsymbol{w}_{t-1}^b + k_t(\boldsymbol{w}_t - \boldsymbol{w}_{t-1}^b)$
9:      $\boldsymbol{w}_t^c \leftarrow (1-\theta)\boldsymbol{w}_{t-1}^b + \theta\boldsymbol{w}_t$
10: until $F(\boldsymbol{w}_t^b) - F_t(\boldsymbol{w}_t) \leq \varepsilon$
```

Unlike BMRM, LS-BMRM simultaneously optimizes the master and reduced problems' objectives F and F_t, respectively. In addition, LS-BMRM selects cutting planes that are close to the best-so-far solution which has a stabilization effect and, moreover, such cutting planes have a higher chance of actively contributing to the approximation of the master objective F around the optimum $\boldsymbol{w}^*$. In particular, there are three main changes compared to BMRM:

1. BMRM-LS maintains the best-so-far solution $\boldsymbol{w}_t^b$ obtained during the first t iterations, i.e., $F(\boldsymbol{w}_0^b), \ldots, F(\boldsymbol{w}_t^b)$ is a monotonically decreasing sequence.

2. The new best-so-far solution $\boldsymbol{w}_t^b$ is found by searching along a line starting at the previous solution $\boldsymbol{w}_{t-1}^b$ and crossing the reduced problem's solution $\boldsymbol{w}_t$. This is implemented on lines 7 and 8.

3. The new cutting plane is computed to approximate the master objective F at the point $\boldsymbol{w}_t^c \leftarrow (1-\theta)\boldsymbol{w}_t^b + \theta\boldsymbol{w}_t$ (line 9) which lies on the line segment between the best-so-far solution $\boldsymbol{w}_t^b$ and the reduced problem's solution $\boldsymbol{w}_t$. $\theta \in (0,1]$ is a prescribed parameter. Note that $\boldsymbol{w}_t^c$ must not be set directly to $\boldsymbol{w}_t^b$ in order to guarantee convergence (i.e., $\theta = 0$ is not allowed). It was found experimentally (Franc and Sonnenburg, 2010), that value $\theta = 0.1$

works consistently well.

Convergence guarantees

LS-BMRM converges in $\mathcal{O}(\frac{1}{\varepsilon})$ iterations to ε-precise solution:

Theorem 1.4. *(Teo et al., 2010, Theorem 7) Under the assumption of Theorem 1.3 Algorithm 1.4 converges to the desired precision after*

$$T \leq \frac{8G^2H^*}{\lambda\varepsilon}$$

iterations for any $\varepsilon < 4G^2H^*\lambda^{-1}$.

Efficient line-search algorithm

LS-BMRM requires at line 7 to solve a line-search problem

$$k^* = \operatorname*{argmin}_{k \geq 0} f(k) \quad \text{where} \quad f(k) = \lambda\Omega(\boldsymbol{w}' + k\boldsymbol{w}) + R(\boldsymbol{w}' + k\boldsymbol{w}) \,. \tag{15}$$

Franc and Sonnenburg (2008, 2010) proposed a line-search algorithm which finds the exact solution of (15) if $\Omega(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2$ and

$$R(\boldsymbol{w}) = \sum_{i=1}^{m} \max_{j=1,\ldots,p} (u_{ij} + \langle \boldsymbol{v}_{ij}, \boldsymbol{w} \rangle) \,, \tag{16}$$

where $u_{ij} \in \mathbb{R}$ and $\boldsymbol{v}_{ij} \in \mathbb{R}^n$, $i = 1, \ldots, m$, $j = 1, \ldots, p$, are some fixed scalars and vectors, respectively. In this case, the subdifferential of $\partial f(k)$ can be described by $\mathcal{O}(pm)$ line segments in 2D. Problem (15) can be replaced by solving $\partial f(k) \in 0$ w.r.t. k which is equivalent to finding among the line segments the one intersecting the x-axis. This line-search algorithm finds the exact solution of (15) in $\mathcal{O}(mp^2 + mp \log mp)$ time. The risk (16) emerges in most variants of the support vector machines learning algorithms, e.g., binary SVMs, multi-class SVMs or SVM regression. Unfortunately, the algorithm is not applicable if p is huge which excludes applications to the structured output SVM learning (Tsochantaridis et al., 2005).

1.2.3 Conclusions

A notable advantage of BMRM is its modularity and simplicity. One only needs to supply a procedure to compute the risk $R(\boldsymbol{w})$ and its subgradient $R'(\boldsymbol{w})$ at a point $\boldsymbol{w}$. The core part of BMRM, i.e., solving the reduced problem, remains for a given regularizer Ω unchanged. Thus, many existing learning problems can be solved by a single optimization technique. Moreover, one can easily experiment with new learning formulations just by specifying the risk term R and its subgradient R' without spending time on development of a new solver for that particular problem.

The convergence speed of BMRM and the accelerated LS-BMRM has been extensively studied on a variety of real-life problems in domains ranging from the text classification, bioinformatics and computer vision to computer security systems (Teo et al., 2007; Franc and Sonnenburg, 2008, 2010; Teo et al., 2010). Compared to the state-of-the-art dedicated solvers, BMRM is typically slightly slower, however, it is still competitive and practically useful. On the other hand, the LS-BMRM has proved to be among the fastest optimization algorithms for a variety of problems. Despite the similar theoretical convergence times, in practice, the LS-BMRM is on average by an order of magnitude faster than BMRM.

The most time-consuming part of BMRM, as well as LS-BMRM, is the evaluation of the risk R and its subgradient R'. Fortunately, the risk, and thus also its subgradient, are typically additively decomposable which allows for an efficient parallelization of their computation. The effect of the parallelization on the reduction of the computational time is empirically studied in Franc and Sonnenburg (2010); Teo et al. (2010).

A relatively high memory requirements of BMRM/LS-BMRM may be the major deficiency if the method is applied to large-scale problems. The method stores in each iteration t a cutting plane of size $\mathcal{O}(n)$, where n is the dimension of the parameter vector $\boldsymbol{w} \in \mathbb{R}^n$, which leads to $\mathcal{O}(nt)$ memory complexity not counting the reduced problem which is typically much less memory demanding. To alleviate the problem, Teo et al. (2010) propose a limited memory variant of BMRM maintaining up to K cutting planes aggregated from the original t cutting planes. Though the memory limited variant does not have an impact on the theoretical upper bound of the number of iterations, in practice, it may significantly slow down the convergence.

The implementations of BMRM and LS-BMRM can be found in the SHOGUN machine learning toolbox (Sonnenburg et al., 2010) or in the open-source packages BMRM (`http://users.cecs.anu.edu.au/~chteo/BMRM.html`) and LIBOCAS (`http://cmp.felk.cvut.cz/~xfrancv/ocas/html/index.html`).

1.3 Multiple kernel learning

Multiple kernel learning (MKL) (e.g., Bach et al., 2004) has recently become an active line of research. Given a mapping $\Phi : \mathcal{X} \mapsto \mathbb{R}^n$ that represents each object $\boldsymbol{x} \in \mathcal{X}$ in n-dimensional feature space[1], a kernel machine employs a

[1] For the sake of simplicity, we consider the n-dimensional Euclidean feature space. However, all the methods in this section can be applied even if the objects are mapped

kernel function

$$\mathbf{k}(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle$$

to compare two objects $\boldsymbol{x}$ and $\boldsymbol{x}'$ without ever *explicitly* computing $\Phi(\boldsymbol{x})$. Ultimately, a kernel machine learns $\boldsymbol{\alpha}$-weighted linear combination of kernel functions with bias b

$$h(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i \mathbf{k}(\boldsymbol{x}_i, \boldsymbol{x}) + b\,, \tag{17}$$

where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ is a set of training objects. For example, the support vector machine (SVM) classifier uses the sign of $h(\boldsymbol{x})$ to assign a class label $y \in \{-1, +1\}$ to the object $\boldsymbol{x}$ (e.g., Schölkopf and Smola, 2002).

Traditionally, just a single kernel function has been used. However, it has proven beneficial to consider not just a single, but multiple kernels in various applications (see Section 1.3.4). Currently, the most popular way to combine kernels is via convex combinations, i.e., introducing

$$B = \left\{ \boldsymbol{\beta} \in \mathbb{R}^K \,\middle|\, \|\boldsymbol{\beta}\|_1 = 1\,, \boldsymbol{\beta} \geq 0 \right\}, \tag{18}$$

the *composite kernel* is defined as

$$\mathbf{k}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{k=1}^{K} \beta_k \mathbf{k}_k(\boldsymbol{x}, \boldsymbol{x}')\,, \qquad \boldsymbol{\beta} \in B\,, \tag{19}$$

where $\mathbf{k}_k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $k = 1, \ldots, K$, is a given set of positive-definite kernels (Schölkopf and Smola, 2002). Now, in contrast to single kernel algorithms, MKL learns in addition to the coefficients $\boldsymbol{\alpha}$ and b the weighting over kernels $\boldsymbol{\beta}$.

In Section 1.3.1, we review convex MKL for classification and, in Section 1.3.2, we show that this problem can be cast as minimization of a complicated convex function over a simple feasible set. In Section 1.3.3, we derive a CPA that transforms the MKL problem into a sequence of linear and quadratic programs, of which the latter can be efficiently solved by existing SVM solvers. Section 1.3.4 concludes this part.

1.3.1 Convex multiple kernel learning

Various MKL formulations have been proposed (Lanckriet et al., 2004b; Bach et al., 2004; Sonnenburg et al., 2006a; Varma and Babu, 2009; Kloft

into arbitrary Reproducing Kernel Hilbert Space (Schölkopf and Smola, 2002).

et al., 2009; Bach, 2009; Nath et al., 2009; Cortes et al., 2009). Here we focus solely on the convex optimization problem for classification as first stated by Zien and Ong (2007); Rakotomamonjy et al. (2007). Note that the same authors have shown that the mixed-norm approaches of Bach et al. (2004); Sonnenburg et al. (2006a) are equivalent.

Let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} \in (\mathcal{X} \times \{-1, +1\})^m$ be a training set of examples of input $\boldsymbol{x}$ and output y assumed to be i.i.d. from an unknown distribution $p(\boldsymbol{x}, y)$. The input $\boldsymbol{x}$ is translated into a compositional feature vector $(\Phi_1(\boldsymbol{x}); \ldots; \Phi_K(\boldsymbol{x})) \in \mathbb{R}^{n_1+\cdots+n_k}$ that is constructed by a set of K mappings $\Phi_k \colon \mathcal{X} \to \mathbb{R}^{n_k}$, $k = 1, \ldots, K$. The goal is to predict y from an unseen $\boldsymbol{x}$ by using a linear classifier

$$y = \operatorname{sgn}\big(h(\boldsymbol{x})\big) \quad \text{where} \quad h(\boldsymbol{x}) = \sum_{k=1}^{K} \langle \boldsymbol{w}_k, \, \Phi_k(\boldsymbol{x}) \rangle + b \,, \tag{20}$$

its parameters $\boldsymbol{w}_k \in \mathbb{R}^{n_k}$, $k = 1, \ldots, K$, $b \in \mathbb{R}$, are learned from the training examples. Using the definition $\frac{x}{0} = 0$ if $x = 0$ and ∞ otherwise, the parameters of the classifier (20) can be obtained by solving the following convex *primal MKL optimization problem* (Zien and Ong, 2007; Rakotomamonjy et al., 2007)

$$\begin{aligned}
\min \quad & \frac{1}{2} \sum_{k=1}^{K} \frac{1}{\beta_k} \, \|\boldsymbol{w}_k\|_2^2 + C \sum_{i=1}^{m} \xi_i \qquad\qquad (21) \\
\text{w.r.t.} \quad & \boldsymbol{\beta} \in B \,, \boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) \in \mathbb{R}^{n_1+\cdots+n_K} \,, \boldsymbol{\xi} \in \mathbb{R}^m, b \in \mathbb{R} \\
\text{s.t.} \quad & \xi_i \geq 0 \text{ and } y_i \left(\sum_{k=1}^{K} \langle \boldsymbol{w}_k, \Phi_k(\boldsymbol{x}_i) \rangle + b \right) \geq 1 - \xi_i, \; i = 1, \ldots, m \,.
\end{aligned}$$

Analogously to the SVMs, the objective of (21) is composed of two terms. The first (regularization) term constrains the spaces of the parameters $\boldsymbol{w}_k$, $k = 1, \ldots, K$, in order to improve the generalization of the classifier (20). The second term, weighted by a prescribed constant $C > 0$, is an upper bound on the number of mistakes the classifier (20) makes on the training examples. In contrast to SVMs, positive weights $\boldsymbol{\beta}$ with ℓ_1-norm constraint (see (18)) are introduced to enforce block-wise sparsity, i.e., rather few blocks of features Φ_k are selected (have non-zero weight $\boldsymbol{w}_k$). Since $\frac{1}{\beta_k} \gg 1$ for small β_k, non-zero components of $\boldsymbol{w}_k$ experience stronger penalization and thus the smaller β_k the smoother $\boldsymbol{w}_k$. By definition, $\boldsymbol{w}_k = 0$ if $\beta_k = 0$. Note that for $K = 1$, the MKL problem (21) reduces to the standard two-class linear SVM classifier.

1.3.2 Min-max formulation of multiple kernel learning

To apply kernels, the primal MKL problem (21) must be reformulated such that the features vectors $\Phi_k(\boldsymbol{x}_i)$ appear in terms of dot products only. Following Rakotomamonjy et al. (2007) we can rewrite (21) as

$$\min\{F(\boldsymbol{\beta}) \mid \boldsymbol{\beta} \in B\}\ , \tag{22}$$

where $F(\boldsymbol{\beta})$ is a shortcut for solving the standard SVM primal on the $\boldsymbol{\beta}$-weighted concatenated feature space

$$\begin{aligned} F(\boldsymbol{\beta}) = \min \quad & \frac{1}{2}\sum_{k=1}^{K}\frac{1}{\beta_k}\,\|\boldsymbol{w}_k\|_2^2 + C\sum_{i=1}^{m}\xi_i \\ \text{w.r.t.} \quad & \boldsymbol{w} = (\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) \in \mathbb{R}^{n_1+\cdots+n_K}\,, \boldsymbol{\xi} \in \mathbb{R}^m, b \in \mathbb{R} \\ \text{s.t.} \quad & \xi_i \geq 0 \text{ and } y_i\left(\sum_{k=1}^{K}\langle \boldsymbol{w}_k, \Phi_k(\boldsymbol{x}_i)\rangle + b\right) \geq 1-\xi_i\,, i = 1,\ldots,m. \end{aligned} \tag{23}$$

Note, that in (23) the weights $\boldsymbol{\beta}$ are fixed and the minimization is only over $(\boldsymbol{w}, \boldsymbol{\xi}, b)$. The Lagrange dual of (23) reads (Rakotomamonjy et al., 2007)

$$D(\boldsymbol{\beta}) = \max\{S(\boldsymbol{\alpha},\boldsymbol{\beta}) \mid \boldsymbol{\alpha} \in A\} \quad \text{where} \quad S(\boldsymbol{\alpha},\boldsymbol{\beta}) = \sum_{k=1}^{K}\beta_k S_k(\boldsymbol{\alpha})\,, \tag{24}$$

and S_k and A are defined as follows:

$$\begin{aligned} S_k(\boldsymbol{\alpha}) &= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \langle \Phi_k(\boldsymbol{x}_i),\, \Phi_k(\boldsymbol{x}_j)\rangle \\ A &= \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid 0 \leq \alpha_i \leq C\,, i = 1,\ldots,m\,, \sum_{i=1}^{m}\alpha_i y_i = 0\}\ . \end{aligned} \tag{25}$$

Note, that (24) is equivalent to solving the standard SVM dual with the composite kernel (19). Because (23) is convex and the Slater's qualification condition holds, the duality gap is zero, i.e. $F(\boldsymbol{\beta}) = D(\boldsymbol{\beta})$. Substituting $D(\boldsymbol{\beta})$ for $F(\boldsymbol{\beta})$ in (22) leads to an equivalent *min-max MKL problem*

Min-Max Problem

$$\min\{D(\boldsymbol{\beta}) \mid \boldsymbol{\beta} \in B\}\ . \tag{26}$$

Let $\boldsymbol{\beta}^* \in \operatorname{argmax}_{\boldsymbol{\beta}\in B} D(\boldsymbol{\beta})$ and $\boldsymbol{\alpha}^* \in \operatorname{argmax}_{\boldsymbol{\alpha}\in A} S(\boldsymbol{\alpha},\boldsymbol{\beta}^*)$. Then the solution of the primal MKL problem (21) can be computed analytically as

$$\boldsymbol{w}_k^* = \beta_k^* \sum_{i=1}^{m}\alpha_i^* y_i \Phi_k(\boldsymbol{x}_i) \quad \text{and} \quad b^* = y_i - \sum_{k=1}^{K}\langle \boldsymbol{w}_k^*,\, \Phi_k(\boldsymbol{x}_i)\rangle\ , i \in J\ , \tag{27}$$

where $J = \{j \in \{1, \dots, m\} \mid 0 < \alpha_i^* < C\}$. The equalities (27) follow from the Karush-Kuhn-Tucker optimality conditions of the problem (23) (e.g., Schölkopf and Smola, 2002). Note, that in practice, b^* is computed as an average over all $|J|$ equalities which is numerically more stable.

By substituting (27) and $\mathbf{k}_k(\boldsymbol{x}_i, \boldsymbol{x}) = \langle \Phi_k(\boldsymbol{x}_i), \Phi_k(\boldsymbol{x}) \rangle$ in the linear classification rule (20), we obtain the kernel classifier (17) with the composite kernel (19). In addition, after substituting $\mathbf{k}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for the dot products $\langle \Phi_k(\boldsymbol{x}_i), \Phi_k(\boldsymbol{x}_j) \rangle$ in (25) we can compute all the parameters of the kernel classifier without ever using the features $\Phi_k(\boldsymbol{x}_i)$ explicitly.

1.3.3 Solving MKL via cutting planes

In this section, we will apply the cutting plane Algorithm 1.2 to the min-max MKL problem (26).

It follows from (24) that the objective D is convex since it is a pointwise maximum over an infinite number of functions $S(\boldsymbol{\alpha}, \boldsymbol{\beta})$, $\boldsymbol{\alpha} \in A$, which are linear in $\boldsymbol{\beta}$ (e.g., Boyd and Vandenberghe, 2004). By Danskin's theorem (e.g., Proposition B.25 in Bertsekas, 1999), the subgradient of D at point $\boldsymbol{\beta}$ equals the gradient $\nabla_{\boldsymbol{\beta}} S(\hat{\boldsymbol{\alpha}}, \boldsymbol{\beta})$ where $\hat{\boldsymbol{\alpha}} \in \operatorname{argmax}_{\boldsymbol{\alpha} \in A} S(\boldsymbol{\alpha}, \boldsymbol{\beta})$, i.e, the subgradient reads

$$D'(\boldsymbol{\beta}) = [S_1(\hat{\boldsymbol{\alpha}}); \dots ; S_K(\hat{\boldsymbol{\alpha}})] \in \mathbb{R}^K \; . \tag{28}$$

Note, that computing $D(\boldsymbol{\beta})$ and its subgradient $D'(\boldsymbol{\beta})$ requires solving the convex quadratic program (24) which is equivalent to the standard SVM dual computed on the composite kernel (19) with a fixed weighting $\boldsymbol{\beta}$ (Rakotomamonjy et al., 2007). Thus, existing SVM solvers are directly applicable.

Having the means to compute D and its subgradient D', we can approximate the objective D by its cutting-plane model

$$\begin{aligned} D_t(\boldsymbol{\beta}) &= \max_{i=0,\dots,t-1} \left[D(\boldsymbol{\beta}_i) + \langle D'(\boldsymbol{\beta}_i), \boldsymbol{\beta} - \boldsymbol{\beta}_i \rangle \right] \\ &= \max_{i=0,\dots,t-1} \langle \boldsymbol{\beta}, D'(\boldsymbol{\beta}_i) \rangle \; . \end{aligned} \tag{29}$$

The points $\{\boldsymbol{\beta}_0, \dots, \boldsymbol{\beta}_{t-1}\}$ can be computed by Kelley's CPA (Algorithm 1.2) as follows. Starting with $\boldsymbol{\beta}_0 \in B$, a new iterate is obtained by solving

$$\boldsymbol{\beta}_t \in \operatorname*{argmin}_{\boldsymbol{\beta} \in B} D_t(\boldsymbol{\beta}) \; , \tag{30}$$

which can be cast as a linear program. Note, that since the feasible set B is bounded so is the solution of (30). In each iteration t, the obtained point $\boldsymbol{\beta}_t$ is an estimate of the optimal $\boldsymbol{\beta}^*$, and it is also used to update the cutting

Algorithm 1.5 Cutting plane algorithm for solving the MKL problem. The algorithm requires solving a simple LP (line 7) and a convex QP (line 3) which is equivalent to the standard SVM dual.

```
1: Initialization: $t \leftarrow 0$, $\boldsymbol{\beta}_0 \in B$ (e.g. $\boldsymbol{\beta}_0 = [\frac{1}{K}; \ldots; \frac{1}{K}]$), $\varepsilon > 0$
2: repeat
3:     Let $\boldsymbol{\alpha}_t \in \operatorname{argmax}_{\boldsymbol{\alpha} \in A} S(\boldsymbol{\alpha}, \beta_t)$
4:     Compute $D(\boldsymbol{\beta}_t) \leftarrow S(\boldsymbol{\alpha}_t, \boldsymbol{\beta}_t)$ and $D'(\boldsymbol{\beta}_t) = [S_1(\boldsymbol{\alpha}_t); \ldots; S_K(\boldsymbol{\alpha}_t)]$
5:     $t \leftarrow t + 1$
6:     Update the cutting plane model $D_t(\boldsymbol{\beta}) \leftarrow \max_{i=0,\ldots,t-1} \langle D'(\boldsymbol{\beta}_i), \boldsymbol{\beta} \rangle$
7:     Let $\boldsymbol{\beta}_t \in \operatorname{argmin}_{\boldsymbol{\beta} \in B} D_t(\boldsymbol{\beta})$
8: until $D(\boldsymbol{\beta}_{t-1}) - D_t(\boldsymbol{\beta}_t) \leq \varepsilon$
```

plane model (29). The process is repeated until the gap between $D(\boldsymbol{\beta}_{t-1})$ and $D_t(\boldsymbol{\beta}_t)$ falls below a prescribed ε, meaning that $D(\boldsymbol{\beta}_t) \leq D(\boldsymbol{\beta}^*) + \varepsilon$ holds. Algorithm 1.5 summarizes the method.

Note that originally Sonnenburg et al. (2006a) converted the problem (26) into a *semi-infinite linear problem* (SILP) that was solved by column generation. However, the SILP is equivalent to the epigraph form of (26) (see Section 1.1.1) and the column generation results in the exact same Algorithm 1.5.

Since large-scale SVM training problems are usually solved by so-called decomposition techniques like chunking (e.g., used in Joachims, 1999), one may significantly speedup Algorithm 1.5 by alternately solving for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ within the SVM solver avoiding to solve the full SVM model with a high precision (Sonnenburg et al., 2006a). Furthermore, as noted in Section 1.2.1, potential oscilations occuring in cutting plane methods can be reduced by the bundle methods, as has been done by Xu et al. (2009a).

1.3.4 Conclusions

Multiple Kernel Learning has been used in various applications across diverse fields like bioinformatics, image analysis, signal processing, and biomedical applications like brain computer interfaces. It is being applied to fusing heterogeneous data (Lanckriet et al., 2004a; Sonnenburg et al., 2006b; Zien and Ong, 2007; Rakotomamonjy et al., 2008; Varma and Babu, 2009), to understand the learned kernel classifier (Sonnenburg et al., 2005), feature selection (Szafranski et al., 2008; Xu et al., 2009b; Subrahmanya and Shin, 2010) or automated model selection Sonnenburg et al. (2006a). In this section, we have illustrated that the min-max formulation of MKL problem (22) can be converted into a sequence of linear and quadratic programs, of which the LP is of a simple nature and the QP can be directly solved using any of the various existing SVM solvers. There exist further extensions of this approach not discussed in this section, e.g. an infinite dimensional version of the min-max MKL which was proposed by Argyriou et al. (2006). We provide efficient implementations of MKL in the SHOGUN machine learning toolbox (Sonnenburg et al., 2010).

1.4 MAP inference in graphical models

MAP inference in graphical models (Wainwright and Jordan, 2008) leads to the following NP-hard combinatorial optimization problem: *given a set of variables and a set of functions of (small) subsets of the variables, maximize the sum of the functions over all the variables*. This is also known as the weighted constraint satisfaction problem (Rossi et al., 2006, chapter 9).

The problem has a natural LP relaxation, proposed independently by Shlezinger (1976), Koster et al. (1998), and Wainwright et al. (2005). It is crucial to optimize the LP in the dual because primal methods do not scale to large problems, which is not done in (Koster et al., 1998). The relaxation was extended by Wainwright et al. (2005), Wainwright and Jordan (2008) and Johnson et al. (2007) to a hierarchy of progressively tighter LP relaxations. Komodakis et al. (2007) pointed out that the LP approach can be seen as a dual decomposition of the problem to tractable subproblems.

Several authors proposed to tighten the relaxation incrementally. First, primal methods were proposed (Koster et al., 1998; Sontag and Jaakkola, 2007; Sontag, 2007), then dual methods (Werner, 2008a, 2010; Kumar and Torr, 2008; Sontag et al., 2008; Komodakis and Paragios, 2008). Not all of the authors related these incremental schemes to cutting plane methods.

We revisit here the approach by Werner (2008a, 2010), which, we believe, captures the very core of the dual cutting plane approach to MAP inference in a clean and transparent way. It is a generalization of the dual LP relaxation approach by Shlezinger (1976) and the max-sum diffusion algorithm by Kovalevsky and Koval (approx. 1975), which have been recently reviewed by Werner (2005, 2007).

The approach is surprisingly simple and general. Every subset of the of variables is assigned a function (‘interaction’), all of them except a small part (which defines the problem) being initially zero. Max-sum diffusion passes messages between pairs of the variable subsets, acting as reparameterizations of the problem which monotonically decrease its upper bound. While in the extreme case all pairs of variable subsets are coupled like this, coupling only some of them results in a relaxation of the problem. Any time during diffusion we can tighten the relaxation by coupling new pairs – this results in an incremental scheme, recognized as a dual cutting plane method.

After introducing notation, we construct the integer hull of the problem and the hierarchy of its LP relaxations in Section 1.4.2. In Sections 1.4.3 and 1.4.4 we dualize the LP relaxation and describe the max-sum diffusion algorithm which optimizes the dual. In Section 1.4.5 we augment this to a dual cutting plane algorithm and discuss the corresponding separation problem. Section 1.4.6 explains the geometry of this cutting plane algorithm in the primal domain, relating it to the marginal polytope.

1.4.1 Notation and problem definition

Let V be an ordered set of *variables* (the order on V is used only for notation consistency). A variable $v \in V$ attains *states* $x_v \in X_v$, where X_v is the (finite) *domain* of the variable. The *joint domain* of a subset $A \subseteq V$ of the variables is the Cartesian product $X_A = \prod_{v \in A} X_v$, where the order of factors is given by the order on V. A tuple $x_A \in X_A$ is a *joint state* of variables A. An *interaction* with *scope* $A \subseteq V$ is a function $\theta_A \colon X_A \to \overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty\}$.

Let $E \subseteq 2^V$ be a hypergraph on V (a set of subsets of V). Every variable subset $A \subseteq V$ is assigned an interaction, while θ_A is identically zero whenever $A \notin E$. Having to deal with so many interactions may seem scary – but it will always be evident that the vast majority of them do not contribute to sums and are never visited in algorithms. Our task is to compute

$$\max_{x_V \in X_V} \sum_{A \in E} \theta_A(x_A) = \max_{x_V \in X_V} \sum_{A \subseteq V} \theta_A(x_A) \; . \tag{31}$$

E.g., if $V = (1, 2, 3, 4)$ and $E = \{(1, 3, 4), (2, 3), (2, 4), (3)\}$ then (31) reads $\max\limits_{x_1,x_2,x_3,x_4} [\theta_{134}(x_1, x_3, x_4) + \theta_{23}(x_2, x_3) + \theta_{24}(x_2, x_4) + \theta_3(x_3)]$. Note, as V is an ordered set we use $(\cdots)$ rather than $\{\cdots\}$ to denote V and its subsets.

We will use $T = \{\, (A, x_A) \mid A \subseteq V,\ x_A \in X_A \,\}$ to denote the set of all joint states of all variable subsets ('T' stands for '$t$uples'). All interactions θ_A, $A \subseteq V$, will be understood as a single vector $\boldsymbol{\theta} \in \overline{\mathbb{R}}^T$.

1.4.2 The hierarchy of LP relaxations

LP formulation

We define a mapping $\boldsymbol{\delta} \colon X_V \to \{0, 1\}^T$ as follows: $\delta_A(y_A)(x_V)$ equals 1 if joint state y_A is the restriction of joint state x_V on variables A, and 0 otherwise. Here, $\delta_A(y_A)(x_V)$ denotes the (A, y_A)-component of vector $\boldsymbol{\delta}(x_V) \in \{0, 1\}^T$. This lets us write the objective function of (31) as a scalar product,

$$\sum_{A \subseteq V} \theta_A(x_A) = \sum_{A \subseteq V} \sum_{y_A} \theta_A(y_A)\, \delta_A(y_A)(x_V) = \langle \boldsymbol{\theta}, \boldsymbol{\delta}(x_V) \rangle \; .$$

Problem (31) can now be reformulated as

$$\max_{x_V \in X_V} \sum_{A \subseteq V} \theta_A(x_A) = \max_{x_V \in X_V} \langle \boldsymbol{\theta}, \boldsymbol{\delta}(x_V) \rangle = \max_{\boldsymbol{\mu} \in \boldsymbol{\delta}(X_V)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle = \max_{\boldsymbol{\mu} \in \operatorname{conv} \boldsymbol{\delta}(X_V)} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle$$

where $\boldsymbol{\delta}(X_V) = \{\, \boldsymbol{\delta}(x_V) \mid x_V \in X_V \,\}$. This expresses problem (31) in the form (9), as a linear optimization over the integral hull $\operatorname{conv} \boldsymbol{\delta}(X_V) \subseteq [0,1]^T$.

Let $I = \{\, (A, B) \mid B \subseteq A \subseteq V \,\}$ denote the set of hyperedge pairs related by inclusion, i.e., the inclusion relation on 2^V. For any $J \subseteq I$, we define a polytope $\mathcal{M}(J)$ to be the set of vectors $\boldsymbol{\mu} \in [0,1]^T$ satisfying

$$\sum_{x_{A \setminus B}} \mu_A(x_A) = \mu_B(x_B) \;, \qquad (A, B) \in J,\ x_B \in X_B \;, \tag{32a}$$

$$\sum_{x_A} \mu_A(x_A) = 1 \;, \qquad A \subseteq V \;. \tag{32b}$$

What is this object? Any $\boldsymbol{\mu} \in \mathcal{M}(J)$ is a set of distributions $\mu_A \colon X_A \to [0,1]$ over every subset $A \subseteq V$ of the variables. Constraint (32b) normalizes the distributions. Constraint (32a) couples pairs of distributions, imposing that μ_B is the marginal of μ_A whenever $(A, B) \in J$. E.g., if $A = (1,2,3,4)$ and $B = (2,4)$ then (32a) reads $\sum_{x_1,x_3} \mu_{1234}(x_1, x_2, x_3, x_4) = \mu_{24}(x_2, x_4)$.

integral hull

For brevity, we will use the shorthand $\mathcal{M}(I) = \mathcal{M}$. We claim that

$$\operatorname{conv} \boldsymbol{\delta}(X_V) = \mathcal{M} \;. \tag{33}$$

To see it, let us write a convex combination of the elements of $\boldsymbol{\delta}(X_V)$,

$$\boldsymbol{\mu} = \sum_{x_V} \mu_V(x_V)\, \boldsymbol{\delta}(x_V) \;, \tag{34}$$

where $\mu_V(x_V)$ denote the coefficients of the convex combination. But μ_V is already part of $\boldsymbol{\mu}$. The (A, y_A)-component of vector (34) reads

$$\mu_A(y_A) = \sum_{x_V} \mu_V(x_V)\, \delta_A(y_A)(x_V) = \sum_{y_{V \setminus A}} \mu_V(y_V) \;.$$

But this is (32a) for $(A, B) = (V, A)$.

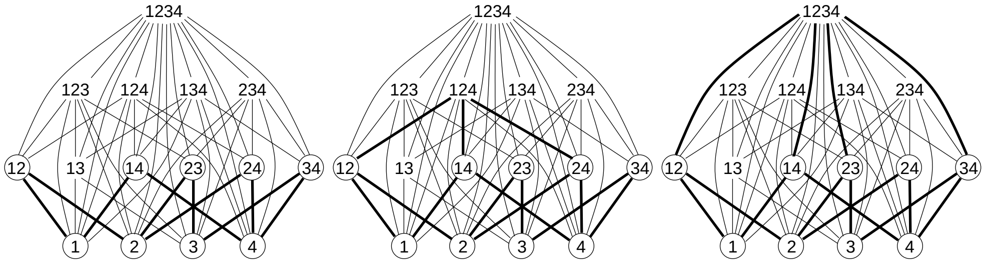
hierarchy of LP relaxations

By imposing only a subset of all possible marginalization constraints (32a), an outer relaxation of the integral hull $\operatorname{conv} \boldsymbol{\delta}(X_V) = \mathcal{M}$ is obtained. Namely, for any $J \subseteq I$ we have $\mathcal{M}(J) \supseteq \mathcal{M}$, hence

$$\max\{\, \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle \mid \boldsymbol{\mu} \in \mathcal{M}(J) \,\} \tag{35}$$

is a linear programming relaxation of problem (31), i.e., its optimum is an upper bound on (31). All possible relaxations form a partially ordered hierarchy, indexed by $J \subseteq I$. Figure 1.2 shows examples.

We remark that the hierarchy could be made finer-grained by selecting also *subsets* of joint states, i.e., by imposing marginalization equality (32a) for $(A, B, x_B) \in J$ where $J \subseteq I = \{\, (A, B, x_B) \mid B \subseteq A \subseteq V,\ x_B \in X_B \,\}$.



(a) (b) (c)

Figure 1.2: The Hasse diagram of the set 2^V of all subsets of $V = (1, 2, 3, 4)$. The nodes depict hyperedges $A \subseteq V$ (with hyperedge $\emptyset$ omitted) and the arcs depict hyperedge pairs $(A, B) \in I$. The hyperedges in circles form the problem hypergraph $E = \{(1), (2), (3), (4), (1, 2), (1, 4), (2, 3), (2, 4), (3, 4)\}$, the interactions over the non-circled hyperedges are zero. Any subset $J \subseteq I$ of the arcs yields one possible relaxation (35) of problem (31). Subfigures (a), (b), (c) show three example relaxations, with J depicted as thick arcs.

1.4.3 The dual of the LP relaxation

constructing the dual

Rather than solving the linear program (35) directly, it is much better to solve its dual. This dual is constructed as follows. Let matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ be such that $\boldsymbol{A\mu} = \boldsymbol{0}$ and $\boldsymbol{B\mu} = \boldsymbol{1}$ is the set of equalities (32a) and (32b), respectively. Then (35) can be written as the left linear program below:

$$\begin{aligned} \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle &\to \max & \langle \boldsymbol{\psi}, \boldsymbol{1} \rangle &\to \min && \text{(36a)}\\ \boldsymbol{A\mu} &= \boldsymbol{0} & \boldsymbol{\varphi} &\lesseqgtr \boldsymbol{0} && \text{(36b)}\\ \boldsymbol{B\mu} &= \boldsymbol{1} & \boldsymbol{\psi} &\lesseqgtr \boldsymbol{0} && \text{(36c)}\\ \boldsymbol{\mu} &\geq \boldsymbol{0} & \boldsymbol{\varphi A} + \boldsymbol{\psi B} &\geq \boldsymbol{\theta} && \text{(36d)} \end{aligned}$$

On the right we wrote the LP dual, such that in (36b-d) a constraint and its Lagrange multiplier are always on a same line (‘$\lesseqgtr \boldsymbol{0}$’ means that the variable vector is unconstrained). By eliminating the variables $\boldsymbol{\psi}$, the dual reads

$$\min_{\boldsymbol{\varphi}} \sum_{A \subseteq V} \max_{x_A} \theta_A^{\boldsymbol{\varphi}}(x_A) \tag{37}$$

where we abbreviated $\boldsymbol{\theta}^{\boldsymbol{\varphi}} = \boldsymbol{\theta} - \boldsymbol{\varphi}\boldsymbol{A}$. The components of vector $\boldsymbol{\theta}^{\boldsymbol{\varphi}}$ read

$$\theta^{\boldsymbol{\varphi}}_A(x_A) = \theta_A(x_A) - \sum_{B|(B,A)\in J} \varphi_{BA}(x_A) + \sum_{B|(A,B)\in J} \varphi_{AB}(x_B) \tag{38}$$

where $\boldsymbol{\varphi} = \{\, \varphi_{AB}(x_B) \mid (A,B) \in J,\ x_B \in X_B \,\}$. Next we explain the meaning of (38) and (37).

reparameterizations

A *reparameterization* is a transformation of $\boldsymbol{\theta}$ that preserves the objective function $\sum_{A\subseteq V} \theta_A$ of problem (31). The simplest reparameterization is done as follows: pick two interactions θ_A and θ_B with $B \subseteq A$, add an arbitrary function (a 'message') $\varphi_{AB}\colon X_B \to \mathbb{R}$ to θ_A, and subtract it from θ_B:

$$\theta_A(x_A) \leftarrow \theta_A(x_A) + \varphi_{AB}(x_B)\ , \qquad x_A \in X_A\ , \tag{39a}$$

$$\theta_B(x_B) \leftarrow \theta_B(x_B) - \varphi_{AB}(x_B)\ , \qquad x_B \in X_B\ . \tag{39b}$$

E.g., if $A = (1,2,3,4)$ and $B = (2,4)$ then we add a function $\varphi_{24}(x_2,x_4)$ to $\theta_{1234}(x_1,x_2,x_3,x_4)$ and subtract it from $\theta_{24}(x_2,x_4)$. This preserves $\theta_A + \theta_B$ (because φ_{AB} cancels out) and hence also $\sum_{A\subseteq V} \theta_A$. Applying reparameterization (39) to all pairs $(A,B) \in J$ yields (38).

Thus, (38) describes reparameterizations, i.e., for every x_V and $\boldsymbol{\varphi}$ we have

$$\sum_{A\subseteq V} \theta_A(x_A) = \sum_{A\subseteq V} \theta^{\boldsymbol{\varphi}}_A(x_A)\ .$$

Besides this, (38) preserves (for feasible $\boldsymbol{\mu}$) also the objective of the primal program (36): $\boldsymbol{A\mu} = \mathbf{0}$ implies $\langle \boldsymbol{\theta}^{\boldsymbol{\varphi}}, \boldsymbol{\mu}\rangle = \langle \boldsymbol{\theta} - \boldsymbol{\varphi}\boldsymbol{A}, \boldsymbol{\mu}\rangle = \langle \boldsymbol{\theta}, \boldsymbol{\mu}\rangle$.

upper bound

By the well-known max-sum dominance, for any $\boldsymbol{\theta}$ we have

$$\max_{x_V} \sum_{A\subseteq V} \theta_A(x_A) \le \sum_{A\subseteq V} \max_{x_A} \theta_A(x_A)\ , \tag{40}$$

thus the right-hand side of (40) is an upper bound on (31). This shows that the *dual (37) minimizes an upper bound on (31) by reparameterizations*.

Note that for each $(A,B) \in J$, marginalization constraint (32a) corresponds via duality to message φ_{AB}. The larger is J, the larger is the set of reparameterizations (38) and hence the smaller the optimal value of (37).

When is inequality (40) (and hence the upper bound) tight? It happens if and only if the independent maximizers of the interactions agree on a common global assignment, i.e., if there exists $y_V \in X_V$ such that

$$y_A \in \operatorname*{argmax}_{x_A} \theta_A(x_A)\ , \qquad A \subseteq V\ .$$

CSP on active joint states

We will further refer to the set $\operatorname{argmax}_{x_A} \theta_A(x_A)$ as the *active joint states* of interaction θ_A. The test can be cast as the *constraint satisfaction problem*

(CSP) (Mackworth, 1991; Rossi et al., 2006) formed by the active joint states of all the interactions (Shlezinger, 1976; Werner, 2007, 2010). Thus, if after solving (37) this CSP is satisfiable for $\boldsymbol{\theta}^{\boldsymbol{\varphi}}$, the relaxation is tight and we have solved our instance of problem (31) exactly. Otherwise, we have only an upper bound on (31).

1.4.4 Max-sum diffusion

fixed point

Max-sum diffusion is a simple convergent 'message-passing' algorithm to tackle the dual LP. It seeks to reparameterize $\boldsymbol{\theta}$ such that

$$\max_{x_{A\setminus B}} \theta_A(x_A) = \theta_B(x_B)\ , \qquad (A,B) \in J,\ x_B \in X_B\ . \tag{41}$$

update

The algorithm repeats the following iteration:

Enforce (41) for a single pair $(A, B) \in J$ by reparameterization (39).

This is done by setting $\varphi_{AB}(x_B) = [\theta_B(x_B) - \max_{x_{A\setminus B}} \theta_A(x_A)]/2$ in (39). The algorithm converges to a fixed point when (41) holds for all $(A, B) \in J$.

We remark that originally (Kovalevsky and Koval, approx. 1975), max-sum diffusion was formulated for problems with only binary (no unary) interactions. The generalization (41) by Werner (2008a, 2010) is interesting in the fact that (41) has exactly the same form as (32a). We pursued this idea further in (Werner, 2008b).

Doing reparameterization by messages rather than by modifying $\boldsymbol{\theta}$ yields Algorithm 1.6. To correctly handle infinite weights, the algorithm expects that $[\theta_B(x_B) > -\infty] \Leftrightarrow [\max_{x_{A\setminus B}} \theta_A(x_A) > -\infty]$ for every $(A, B) \in J$.

Algorithm 1.6 Max-sum diffusion

```
repeat
    for $(A,B) \in J$ and $x_B \in X_B$ such that $\theta_B(x_B) > -\infty$ do
        $\varphi_{AB}(x_B) \leftarrow \varphi_{AB}(x_B) + [\theta^{\varphi}_B(x_B) - \max_{x_{A\setminus B}} \theta^{\varphi}_A(x_A)]/2$
    end for
until convergence
```

properties

The diffusion iteration decreases or preserves, but never increases, the upper bound. In general, the algorithm does not find the global minimum of (37) but only a certain local minimum (where 'local' is meant w.r.t. block-coordinate moves), which is nevertheless very good in practice. These local minima are characterized by *local consistency* (Rossi et al., 2006, chapter 3) of the CSP formed by the active joint states.

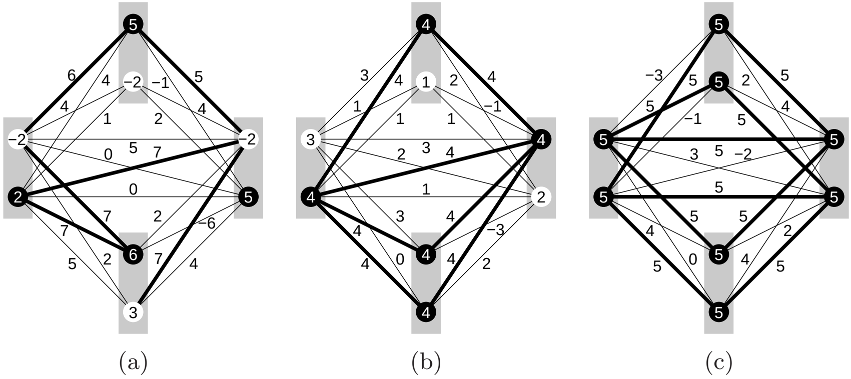


Figure 1.3: The visualization of a problem with $|X_v| = 2$ variable states and hypergraph E as in Figure 1.2. The variables are shown as boxes, their numbering is $\boxed{\begin{smallmatrix} & 1 & \\ 2 & & 4 \\ & 3 & \end{smallmatrix}}$. Variable states are shown as circles, joint states of variable pairs as edges. Weights $\theta_A(x_A)$, $A \in E$, are written in the circles and next to the edges. Active joint states are emphasized (black circles, thick edges). Example (a) is not a diffusion fixed point, (b,c) are diffusion fixed points for J from Figure 1.2a. Examples (a,b) are reparameterizations of each other (this is not obvious at the first sight), (c) is not a reparameterization of (a,b). For (b), a global assignment x_V can be composed of the active joint states and hence inequality (40) is tight. For (a,c), no global assignment x_V can be composed of the active joint states, hence inequality (40) is not tight.

Note that the only non-trivial operation in Algorithm 1.6 is computing the max-marginals $\max_{x_{A\setminus B}} \theta^{\boldsymbol{\varphi}}_A(x_A)$. By (38), this is an instance of problem (31). When $|A|$ is small (such as for a binary interaction), computing the max-marginals is trivial. But even when $|A|$ is large, depending on the function θ_A and on J there may exist an algorithm polynomial in $|A|$ to compute $\max_{x_{A\setminus B}} \theta^{\boldsymbol{\varphi}}_A(x_A)$. In that case, Algorithm 1.6 still can be used.

If $\theta_A = 0$, it depends only on J whether $\max_{x_{A\setminus B}} \theta^{\boldsymbol{\varphi}}_A(x_A)$ can be computed in polynomial time. E.g., in Figure 1.2c we have $\theta_{1234} = 0$ and hence, by (38), $\theta^{\boldsymbol{\varphi}}_{1234}(x_1, x_2, x_3, x_4) = \varphi_{1234,12}(x_1, x_2) + \varphi_{1234,23}(x_2, x_3) + \varphi_{1234,34}(x_3, x_4) + \varphi_{1234,14}(x_1, x_4)$. Thus we have a problem on a cycle, which can be solved more efficiently than by going through all states (x_1, x_2, x_3, x_4).

diffusion solves subproblems

This suggests that diffusion in a sense exactly solves certain small subproblems (which links it to the dual decomposition interpretation (Komodakis et al., 2007)). This can be formalized as follows. Let $A \in F \subseteq 2^A$. Clearly,

$$\max_{x_A} \sum_{B\in F} \theta_B(x_B) \le \sum_{B\in F} \max_{x_B} \theta_B(x_B) \tag{42}$$

for any $\boldsymbol{\theta}$, which is inequality (40) written for subproblem F. Let $J = \{\, (A, B) \mid B \in F \,\}$. In this case, the minimal upper bound for subproblem F is tight. To see it, just do reparameterization (39) with $\varphi_{AB} = \theta_B$ for

$B \in F$, which results in $\theta_B = 0$ for $B \in F \setminus \{A\}$, hence (42) is tight trivially. What is not self-evident is that diffusion finds the *global* minimum in this case. It does: if $\boldsymbol{\theta}$ satisfies (41) for $J = \{\, (A, B) \mid B \in F \,\}$ then (42) is tight.

1.4.5 Dual cutting plane algorithm

incremental scheme

The relaxation can be tightened *incrementally* during dual optimization. Any time during Algorithm 1.6, the current J can be extended by any $J' \subseteq I$, $J' \cap J = \emptyset$. The messages φ_{AB} for $(A, B) \in J'$ are initialized to zero. Clearly, this does not change the current upper bound. Future diffusion iterations can only preserve or improve the bound, so the scheme remains monotonic. This can be imagined as if the added variables φ_{AB} extended the space of possible reparameterizations and diffusion is now trying to take advantage of it. If the bound does not improve, all we will have lost is the memory occupied by the added variables. Algorithm 1.7 describes this.

In the primal domain, this incremental scheme can be understood as a cutting plane algorithm. We discuss this later in Section 1.4.6.

Algorithm 1.7 Dual cutting plane algorithm

```
1: Initialization: Choose $J \subseteq I$ and $\mathcal{J} \subseteq 2^I$.
2: repeat
3:     Execute any number of iterations of Algorithm 1.6.
4:     Separation oracle: Choose $J' \in \mathcal{J}$, $J \cap J' = \emptyset$.
5:     $J \leftarrow J \cup J'$
6:     Allocate messages $\varphi_{AB}$, $(A, B) \in J'$, and set them to zero.
7: until no suitable $J'$ can be found
```

separation test

On line 4 of Algorithm 1.7 the separation oracle is called, which chooses a promising extension J' from some predefined set $\mathcal{J} \subseteq 2^I$ of candidate extensions. We assume $|\mathcal{J}|$ is small so that $\mathcal{J}$ it can be searched exhaustively. For that, we need a test to recognize whether a given J' would lead to a (good) bound improvement. We refer to this as the *separation test*.

Of course, a trivial necessary and sufficient separation test is to extend J by J' and run diffusion till convergence. One easily invents a faster test:

> *Execute several diffusion iterations only on pairs J'. If this improves the bound, then running diffusion on $J \cup J'$ would inevitably improve the bound too.*

This local test is sufficient but not necessary for improvement because even if running diffusion on J' does not improve the bound, it may change the problem such that future diffusion iterations on $J \cup J'$ improve it.

greediness

Even with a sufficient and necessary separation test, Algorithm 1.7 is 'greedy' in the following sense. For $J_1', J_2' \subseteq I$, it can happen that extending J by J_1' alone or by J_2' alone does not lead to a bound improvement but extending J by $J_1' \cup J_2'$ does. See (Werner, 2010) for an example.

The extension J' can be an arbitrary subset of I. One form of extension has a clear meaning: pick a hyperedge A not yet coupled to any other hyperedge, choose $F \subseteq 2^A$, and let $J' = \{\, (A, B) \mid B \in F \,\}$. This can be seen as 'connecting' a so far disconnected interaction θ_A to the problem.

adding zero subproblems

An important special case is connecting a zero interaction, $\theta_A = 0$. Because, by (38), we have $\theta_A^{\boldsymbol{\varphi}}(x_A) = \sum_{B \in F} \theta_B(x_B)$, we refer to this extension as *adding a zero subproblem* F. In this case, the separation test can be done more efficiently than by running diffusion on J'. This is based on the fact stated at the end of Section 1.4.4: if inequality (42) is not tight for current $\boldsymbol{\theta}^{\boldsymbol{\varphi}}$ then running diffusion on J' will surely make it tight, i.e., improve the bound. Note, we do not need $A \in F$ here because $\theta_A = 0$. The gap in (42) is an estimate of the expected improvement.

This has a clear interpretation in CSP terms. Inequality (42) is tight if and only if the CSP formed by the active joint states of interactions F is satisfiable. If this CSP is unsatisfiable then J' will improve the bound. Therefore, *the separation oracle needs to find a (small) unsatisfiable subproblem of the CSP formed by the active joint states.*

For instance, Figure 1.3c shows a problem after diffusion convergence, for J defined by Figure 1.2a. The CSP formed by the active joint states is not satisfiable because it contains an unsatisfiable subproblem, the cycle $F = \{(1,2),(1,4),(2,4)\}$. Hence, adding zero subproblem F (which yields J from Figure 1.2b) and running diffusion would improve the bound. Adding the zero cycle $F = \{(1,2),(1,4),(2,3),(3,4)\}$ (yielding J from Figure 1.2c) or the whole zero problem $F = E$ would improve the bound too.

Figure 1.4 shows a more complex example.

removing zero subproblems

Message passing algorithms have a drawback: after extending J, they need a long time to re-converge. This can be partially alleviated by adding multiple subproblems at a time and doing so before full convergence. As some of the added subproblems might later turn out redundant, we found helpful to remove redundant subproblems occasionally – which can be done without sacrificing monotonicity of bound improvement. This is a (dual) way of 'constraint management', often used in cutting plane methods.

1.4.6 Zero interactions as projection, marginal polytope

zero interactions act as projection

In the beginning, formula (31), we added all possible zero interactions to our problem. This has proved natural because the problem is after all defined only up to reparameterizations and thus any zero interaction can become

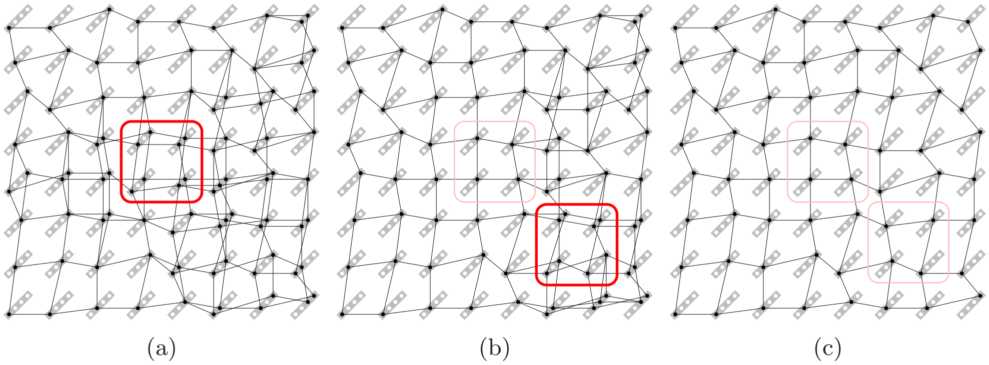


Figure 1.4: Two steps of the cutting plane algorithm for a problem with the 8×8 grid graph E and $|X_v| = 4$ variable states. The set $\mathcal{J}$ of candidate extensions contains all cycles of length 4. Only the active joint states are shown. Subfigure (a) shows the problem after diffusion has converged for $J = \{\, (A,B) \mid B \subseteq A;\ A,B \in E \,\}$. The upper bound is not tight because of the depicted unsatisfiable subproblem (an inconsistent cycle). Adding the cycle and letting diffusion re-converge results in problem (b) with a better bound. The original cycle is now satisfiable but a new unsatisfiable cycle occurred. Adding this cycle solves the problem, (c).

non-zero. Now, let us see how the LP relaxation would look like without adopting this abstraction. Let $T(E) = \{\, (A, x_A) \mid A \in E,\ x_A \in X_A \,\}$ denote the restriction of the set T to hypergraph E. Since zero interactions do not contribute to the objective function of (35), (35) can be written as

$$\max\{\, \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle \mid \boldsymbol{\mu} \in \mathcal{M}(J) \,\} = \max\{\, \langle \pi_{T(E)}\boldsymbol{\theta}, \boldsymbol{\mu} \rangle \mid \boldsymbol{\mu} \in \pi_{T(E)}\mathcal{M}(J) \,\} \tag{43}$$

where $\pi_{D'}\boldsymbol{a} \in \overline{\mathbb{R}}^{D'}$ denotes the projection of a vector $\boldsymbol{a} \in \overline{\mathbb{R}}^{D}$ on dimensions $D' \subseteq D$, thus $\pi_{D'}$ deletes the components $D \setminus D'$ of $\boldsymbol{a}$. Applied to a set of vectors, $\pi_{D'}$ does this for every vector in the set. Informally, (43) shows that *zero interactions act as the projection of the feasible set onto the space of non-zero interactions*.

marginal polytope

The set $\pi_{T(E)}\mathcal{M} \subseteq [0,1]^{T(E)}$ is recognized as the *marginal polytope* (Wainwright et al., 2005) of hypergraph E. Its elements $\boldsymbol{\mu}$ are the marginals over variable subsets E of some global distribution μ_V, which not necessarily is part of $\boldsymbol{\mu}$. The marginal polytope of the complete hypergraph $\pi_{T(2^V)}\mathcal{M} = \mathcal{M}$ is of fundamental importance because all other marginal polytopes are its projections. For $J \subseteq I$, the set $\pi_{T(E)}\mathcal{M}(J) \supseteq \pi_{T(E)}\mathcal{M}$ is a relaxation of the marginal polytope, which may contain elements $\boldsymbol{\mu}$ that no longer can be realized as the marginals of any global distribution μ_V.

While $\operatorname{conv} \boldsymbol{\delta}(X_V) = \mathcal{M}$ is the integral hull of the problem $\max\{\, \langle \boldsymbol{\theta}, \boldsymbol{\mu} \rangle \mid \boldsymbol{\mu} \in \boldsymbol{\delta}(X_V) \,\}$, the polytope $\operatorname{conv} \pi_{T(E)}\boldsymbol{\delta}(X_V) = \pi_{T(E)} \operatorname{conv} \boldsymbol{\delta}(X_V) = \pi_{T(E)}\mathcal{M}$ is the integral hull of the problem $\max\{\, \langle \pi_{T(E)}\boldsymbol{\theta}, \boldsymbol{\mu} \rangle \mid \boldsymbol{\mu} \in \pi_{T(E)}\boldsymbol{\delta}(X_V) \,\}$.

local vs. non-local relaxations

Following (Wainwright et al., 2005), we say a relaxation J is *local in E* if $A, B \in E$ for every $(A, B) \in J$. E.g., in Figure 1.2 only relaxation (a) is local. For local relaxations, the distributions μ_A, $A \notin E$, are not coupled to any other distributions and the action of $\pi_{T(E)}$ on $\mathcal{M}(J)$ is simple: it just removes these superfluous coordinates. Thus, $\pi_{T(E)}\mathcal{M}(J)$ has an explicit description by a small (polynomial in $|E|$) number of linear constraints.

For non-local relaxations, the effect of the projection is in general complex and the number of facets of $\pi_{T(E)}\mathcal{M}(J)$ is exponential in $|E|$. It is well-known that to compute the explicit description of a projection of a polyhedron can be extremely difficult – which suggests that to directly look for the facets of $\pi_{T(E)}\mathcal{M}$ might be a bad idea. Non-local relaxations can be seen as a *lift-and-project approach*: we lift from dimensions $T(E)$ to dimensions T, impose constraints in this lifted space, and project back onto dimensions $T(E)$.

primal view on the cutting plane algorithm

Now it is clear what is the geometry of our cutting plane algorithm in the primal space $[0, 1]^{T(E)}$. Suppose max-sum diffusion has found a global optimum of the dual and let $\boldsymbol{\mu}^* \in [0, 1]^{T(E)}$ be a corresponding primal optimum. A successful extension of J means that a set (perhaps exponentially large) of cutting planes is added to the primal that separates $\boldsymbol{\mu}^*$ from $\pi_{T(E)}\mathcal{M}$. However, $\boldsymbol{\mu}^*$ is not computed explicitly at all (and, let us remark, it is expensive to compute $\boldsymbol{\mu}^*$ from a dual optimum for large problems). In fact, $\boldsymbol{\mu}^*$ may not even exist because diffusion may find only a local optimum of the dual – we even need not run diffusion to full convergence.

1.4.7 Conclusions

We have presented the theory of the cutting plane approach to the MAP inference problem, as well as a very general message passing algorithm to implement this approach. In comparison with other similar works, the theory, and Algorithm 1.6 in particular, is very simple. We have shown that for the case of adding subproblems, separation means finding a (small) unsatisfiable subproblem of the CSP formed by the active joint states.

We assumed, in Section 1.4.5, that the set $\mathcal{J}$ of candidate extensions is tractably small. Is there a polynomial algorithm to select an extension from an intractably large set $\mathcal{J}$? In particular, is there a polynomial algorithm to find a small unsatisfiable subproblem (most interestingly, a cycle) in a given CSP? This is currently an open problem. An inspiration for finding such algorithms are local consistencies in CSP (Rossi et al., 2006, chapter 3).

Let us remark that several polynomial algorithms are known to separate intractable families of cutting planes of the *max-cut polytope* (Deza and Laurent, 1997), closely related to the marginal polytope. Some of them have been applied to MAP inference by Sontag and Jaakkola (2007) and Sontag

(2007). As these algorithms work in the primal space, they cannot be used in our dual cutting plane scheme – we need a *dual separation algorithm*.

Acknowledgments

Vojtěch Franc was supported by the Czech Ministry of Education project 1M0567 and by EC projects FP7-ICT-247525 HUMAVIPS, PERG04-GA-2008-239455 SEMISOL. Soeren Sonneburg was supported by the EU under the PASCAL2 Network of Excellence (ICT-216886) as well as DFG Grants MU 987/6-1 and RA-1894/1-1. Tomáš Werner has been supported by the EC grant 215078 (DIPLECS), the Czech government grant MSM6840770038, and the Grant Agency of the Czech Republic grant P103/10/0783.

References

A. Argyriou, R. Hauser, C. A. Micchelli, and M. Pontil. A dc-programming algorithm for kernel selection. In *Proc. Intl. Conf. Machine Learning*, pages 41–48. ACM Press, 2006.

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 105–112, 2009.

F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *Proc. Intl. Conf. Machine Learning*. ACM Press, 2004.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999. ISBN 1-886529-00-0.

S. Boyd and L. Vandenberge. Localization and cutting-plane methods. Stanford University, California, USA, Unpublished lecture notes, 2008. URL `http://see.stanford.edu/materials/lsocoee364b/05-localization_methods_notes.pdf`.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. ISBN 0521833787.

E. W. Cheney and A. A. Goldstein. Newton's method for convex programming and Tchebycheff approximation. *Numerische Mathematik*, 1:253–268, 1959. ISSN 0029-599X.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995.

C. Cortes, M. Mohri, and A. Rostamizadeh. Learning non-linear combinations of kernels. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 396–404, 2009.

G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

M. M. Deza and M. Laurent. *Geometry of Cuts and Metrics*. Springer, Berlin, 1997.

V. Franc and S. Sonnenburg. OCAS optimized cutting plane algorithm for support vector machines. In *Proc. Intl. Conf. Machine Learning*, pages 320–327. ACM Press, 2008.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *Journal of Machine Learning Research*, 10:2157–2192, 2010.

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

T. Joachims. Making large–scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods — Support Vector Learning*, pages 169–184, Cambridge, MA, USA, 1999. MIT Press.

J. K. Johnson, D. M. Malioutov, and A. S. Willsky. Lagrangian relaxation for MAP estimation in graphical models. In *Allerton Conf. Communication, Control and Computing*, 2007.

J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

K. C. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27:320–341, 1983.

M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.-R. Müller, and A. Zien. Efficient and accurate lp-norm multiple kernel learning. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 997–1005. MIT Press, 2009.

N. Komodakis and N. Paragios. Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles. In *European Conf. on Computer Vision*, 2008.

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. Intl. Conf. Computer Vision*, 2007.

A. Koster, C. P. M. van Hoesel, and A. W. J. Kolen. The partial constraint satisfaction problem: Facets and lifting theorems. *Operations Research Letters*, 23(3–5):89–97, 1998.

V. A. Kovalevsky and V. K. Koval. A diffusion algorithm for decreasing the energy of the max-sum labeling problem. Glushkov Institute of Cybernetics, Kiev, USSR. Unpublished, personally communicated to T. Werner by M. I. Schlesinger., approx. 1975.

M. P. Kumar and P. H. S. Torr. Efficiently solving convex relaxations for MAP estimation. In *Proc. Intl. Conf. Machine Learning*, pages 680–687. ACM, 2008.

G. Lanckriet, T. D. Bie, N. Cristianini, M. Jordan, and W. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20:2626–2635, 2004a.

G. Lanckriet, N. Cristianini, L. E. Ghaoui, P. Bartlett, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004b.

C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. New variants of bundle methods. *Mathematical Programming*, 69(1–3):111–147, 1995.

A. Mackworth. Constraint satisfaction. In *Encyclopaedia of Artificial Intelligence*, pages 285–292. John Wiley, 1991.

J. S. Nath, G. Dinesh, S. Ramanand, C. Bhattacharyya, A. Ben-Tal, and K. R. Ramakrishnan. On the algorithmics and applications of a mixed-norm based kernel learning formulation. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 844–852, 2009.

A. S. Nemirovskij and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* Wiley Interscience, New York, 1983.

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In *Proc. Intl. Conf. Machine Learning*, pages 775–782, 2007.

A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming.* Elsevier, 2006.

B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, 2002.

A. Schrijver. *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics).* Springer, 2003.

M. I. Shlezinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Cybernetics and Systems Analysis*, 12(4):612–628, 1976. Translation from Russian.

S. Sonnenburg, G. Rätsch, and C. Schäfer. Learning interpretable SVMs for biological sequence classification. In S. Miyano, J. P. Mesirov, S. Kasif, S. Istrail, P. A. Pevzner, and M. Waterman, editors, *Research in Computational Molecular Biology, 9th Annual International Conference (RECOMB)*, volume 3500, pages 389–407. Springer-Verlag, 2005.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 7:1531–1565, July 2006a.

S. Sonnenburg, A. Zien, and G. Rätsch. ARTS: Accurate Recognition of Transcription Starts in Human. *Bioinformatics*, 22(14):e472–e480, 2006b.

S. Sonnenburg, G. Rätsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder, C. Gehl, and V. Franc. The SHOGUN machine learning toolbox. *Journal of Machine Learning Research*, 11:1799–1802, June 2010. URL `http://www.shogun-toolbox.org`.

D. Sontag. Cutting plane algorithms for variational inference in graphical models. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2007.

D. Sontag and T. Jaakkola. New outer bounds on the marginal polytope. In *Advances in Neural Information Processing Systems 7*, 2007.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Conf. Uncertainty in Artificial Intelligence (UAI)*, 2008.

N. Subrahmanya and Y. C. Shin. Sparse multiple kernel learning for signal processing applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:788–798, 2010.

M. Szafranski, Y. Grandvalet, and A. Rakotomamonjy. Composite kernel learning. In *Proc. Intl. Conf. Machine Learning*, 2008.

C. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proc. Intl. Conf. Knowledge Discovery and Data Mining*, 2007.

C. Teo, S. Vishwanathan, A. Smola, and V. Quoc. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11: 311–365, 2010.

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.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *Proc. Intl. Conf. Machine Learning*, pages 1065–1072, New York, NY, USA, 2009. ACM Press.

M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on (hyper)trees: message passing and linear programming approaches. *IEEE Trans. Information Theory*, 51(11):3697–3717, 2005.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

T. Werner. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, June 2008a.

T. Werner. Revisiting the linear programming relaxation approach to Gibbs energy minimization and weighted constraint satisfaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, August 2010.

T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7): 1165–1179, July 2007.

T. Werner. Marginal consistency: Unifying constraint propagation on commutative semirings. In *Intl. Workshop on Preferences and Soft Constraints (co-located with Conf. on Principles and Practice of Constraint Programming)*, pages 43–57, September 2008b.

T. Werner. A linear programming approach to max-sum problem: A review. Technical Report CTU–CMP–2005–25, Center for Machine Perception, Czech Technical University, December 2005.

Z. Xu, R. Jin, I. King, and M. Lyu. An extended level method for efficient multiple kernel learning. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 1825–1832, 2009a.

Z. Xu, R. Jin, J. Ye, M. R. Lyu, and I. King. Non-monotonic feature selection. In *Proc. Intl. Conf. Machine Learning*, pages 1145–1152, 2009b.

A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *Proc. Intl. Conf. Machine Learning*, pages 1191–1198. ACM Press, 2007.