Optimization Methods in Structural SVMs

Ayan Acharya
Department of ECE
University of Texas at Austin
Email: masterayan@gmail.com

Dec 05, 2011

1 Introduction

SVMs have traditionally been solved in the dual space. The introduction of Sequential Minimal Optimization (**SMO**) [14] algorithm made this optimization even faster using some tricks of block-coordinate descent. However, in recent years, researchers got interested in solving the primal problem instead – mainly due to the following reasons:

1. In many practical applications (*e.g.* text classification, word-sense disambiguation, drug design etc.), data is of very high dimension and often are linearly separable and hence nothing much is gained using the kernel trick.

2. Efficient factorization methods (*e.g.* see [5]) can be used for low-rank approximation of the kernel matrix and, therefore, the problem becomes embarrassingly linear.

3. Approximation methods such as *Random Feature Map* [15] can approximate an infinite dimensional non-linear feature map by a finite dimensional one.

Although the initial version of Structural Support Vector Machine (**SSVM**) [19] solved the optimization problem in the dual space, recent surge of interest for primal problem and also the necessity of designing a discriminative latent variable model – Latent Structural Support Vector Machine (**LSVM**) [21] – have brought about some applications of very efficient algorithms like Concave-Convex Procedure (**CCCP**) [22], cutting plane methods (**CPM**) [7], Proximal Bundle Methods [9], Bundle Methods for Regularized Risk Minimization (**BRBM**) [18] and Non-convex Regularized Bundle Method (**NRBM**) [4]. The emphasis in this paper is to highlight the connections among all of these different approaches.

The paper is organized as follows. In sections 2 and 3, the basic ideas of SSVM and LSVM are introduced. In section 4, the optimization method in the dual space for SSVM is illustrated. CCCP algorithm is portrayed in section 5 and bundle methods for optimization of the primal problem in both SSVM and LSVM are explained in detail in section 6.

2 Structural Support Vector Machine (SSVM)

SSVM was proposed to adapt SVMs [20] for dealing with arbitrary discrete output spaces which can be classes, sequences, strings, labeled trees, lattices or graphs. The number of "structures" in the output space can be potentially infinite. To this end, a discriminant function $F : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is proposed which assigns a real score to an instance-structure pair $(x, y) : x \in \mathcal{X}, y \in \mathcal{Y}$. $\mathcal{X}$ denotes the set of features and $\mathcal{Y}$ stands for the output space. Once the discriminant function F is built from the training data, a given instance x from a test data can be assigned a particular structure $\hat{y}$ based on the equation: $\hat{y} = f(x) = \arg\max_{y \in \mathcal{Y}} F$ where $f : \mathcal{X} \to \mathcal{Y}$. There can be numerous possible ways of modeling the function $F(x, y)$ but we will focus on one particular form $F(x, y; w) = \langle w, \Psi(x, y) \rangle$ where $\Psi(x, y)$ is some arbitrary joint representation of the input features and output space and F is parametrized by the weight vector w. An arbitrary loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ can also be defined over a pair of output structures. For hard margin classification (*i.e.* to ensure zero training error), the necessary conditions in **SSVM** are as follows:

$$\langle w, \Psi(x_i, y_i) \rangle \geq \max_{y \in \mathcal{Y} \setminus y_i} \langle w, \Psi(x_i, y) \rangle \ \forall i. \tag{1}$$

The inequalities in 1 can further be decomposed into the following inequalities:

$$\langle w, \Psi(x_i, y_i) \rangle \geq \langle w, \Psi(x_i, y) \rangle \ \forall i, \ \forall y \in \mathcal{Y} \setminus y_i. \tag{2}$$

If $\delta\Psi_i(y) = \langle w, \Psi(x_i, y_i) \rangle - \langle w, \Psi(x_i, y) \rangle$, the inequalities above take more compact form:

$$\langle w, \delta\Psi_i(y) \rangle \geq 0 \ \forall i, \forall y \in \mathcal{Y} \setminus y_i. \tag{3}$$

To facilitate a unique solution w^* to the set of inequalities in 3, it makes sense to select w so that the separation margin γ (i.e. the difference between the scores of the correct label y_i and the closest runner up) is maximal. Further the norm of w has to be constrained somehow as otherwise, one could increase γ arbitrarily by taking any of the components of w to $+\infty$. Therefore, the optimization problem can be written as:

$$\begin{aligned} &\max_{\gamma, w: ||w||=1} \gamma \\ &\text{s.t. } \langle w, \delta\Psi_i(y) \rangle \geq \gamma \ \forall i, \forall y \in \mathcal{Y} \setminus y_i. \end{aligned} \tag{4}$$

It is easy to visualize geometrically [1] that the optimization problem in 4 can be rewritten as:

$$\begin{aligned} \text{SSVM}_0: \quad & \min_w \frac{1}{2}||w||^2 \\ & \text{s.t. } \langle w, \delta\Psi_i(y)\rangle \geq 1 \ \forall i, \ \forall y \in \mathcal{Y} \setminus y_i. \end{aligned} \tag{5}$$

Note that the optimization mentioned above consists of $n(|\mathcal{Y}| - 1)$ linear inequality constraints. To allow non-zero training error (soft margin classification), slack variables can be introduced which renders SSVM_0 in the following form:

$$\begin{aligned} \text{SSVM}_1: \quad & \min_{w,\{\xi_i\}_{i=1}^n} \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^n \xi_i \\ & \text{s.t. } \delta\Psi_i(y) \geq 1 - \xi_i \ \forall i, \forall y \in \mathcal{Y} \setminus y_i \text{ and } \xi_i \geq 0 \ \forall i. \end{aligned} \tag{6}$$

An exact quantitative assessment of the performance of a training algorithm can be made by using the risk which is defined as:

Definition 1. ***Risk:*** *$\mathcal{R}_P^{\Delta}(f) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta(y, f(x))dP(x, y)$, where $P(x, y)$ is the joint distribution of the feature and the output space.*

However, the unavailability of the joint distribution $P(x, y)$ during the training phase forces us to measure the performance accuracy of the training process through empirical risk which is defined as follows:

Definition 2. ***Empirical Risk:*** *$\mathcal{R}_s^{\Delta}(f; w) = \frac{1}{n}\sum_{i=1}^n \Delta(y_i, f(x_i; w))$. $f(x; w)$ signifies the fact that F is parameterized by w.*

Note that SSVM_1 is not amenable to any arbitrary loss function Δ other than a $0-1$ loss function. Hence, Tsochantaridis *et al* [19] suggested two modifications to the problem.

- **Slack Rescaling:** In the slack rescaling approach, the inequalities in 6 are replaced by inequalities of the form $\delta\Psi_i(y) \geq 1 - \frac{\xi_i}{\Delta(y_i,y)} \ \forall i, \forall y \in \mathcal{Y} \setminus y_i$. An inequality of this form will penalize a slack variable, if it violates any margin constraint, by the inverse of the loss function associated with that margin constraint.
- **Margin Rescaling:** In margin rescaling approach, the inequality is modified as – $\delta\Psi_i(y) \geq \Delta(y_i, y) - \xi_i \ \forall i, \forall y \in \mathcal{Y} \setminus y_i$. From now on, we will concentrate only on the margin rescaling approach as if the loss function Δ is designed as a convex function, the constraints become convex w.r.t the output space y in most of the cases under some suitable assumption of discriminant function. However, the slack rescaling approach is not at all useless and provides some advantages like scaling invariance w.r.t. the loss function and better performance guarantees.

One should note that any of the above modifications has to ensure that $1/n\sum_{i=1}^n \xi_i^*$ is an upper bound on $\mathcal{R}_s^{\Delta}(f; w)$ – the empirical risk, where ξ_i^* denotes the optimal value of the slack variable ξ_i. This is obvious if we recollect the fact that slack variables are introduced to allow some non-zero training error or empirical risk and minimizing an upper bound of empirical risk is equivalent to minimizing the empirical risk itself. For both margin and slack rescaling approach, this condition is satisfied. For margin rescaling, we observe that $\xi_i \geq (\Delta(y_i, y) - \langle w, \delta\Psi_i(y)\rangle) \ \forall i, \forall y \in \mathcal{Y} \setminus y_i$. Hence, $\xi_i^* = \max\{0, \max_{y_i \neq \hat{y}_i}(\Delta(y_i, \hat{y}_i) - \langle w, \delta\Psi_i(\hat{y}_i)\rangle)\} \ \forall i$, which in turn implies that $\xi_i^* \geq \Delta(y_i, \hat{y}_i) \ \forall i, \hat{y}_i$. $\hat{y}_i$ is the predicted output class for x_i obtained from the training phase.

Based on the above argument, one can further explore that $\Delta(y_i, \hat{y}_i) \leq \left[\max_{\hat{y}}\{\Delta(y_i, \hat{y}) + \langle w, \Psi(x_i, \hat{y})\rangle\} - \langle w, \Psi(x_i, y_i)\rangle\right] \leq \xi_i$. Since, the ultimate target is to reduce the empirical risk and the term $\left[\max_{\hat{y}}\{\Delta(y_i, \hat{y}) + \langle w, \Psi(x_i, \hat{y})\rangle\} - \langle w, \Psi(x_i, y_i)\rangle\right]$ upper bounds the empirical risk and lower bounds the slack variable, one could use this term instead of the slack variable in SSVM_1. Therefore, with the introduction of this new term and the arbitrary loss function Δ, the problem in SSVM_1 (with margin rescaled) can be rewritten as:

$$\text{SSVM}_1^{\Delta m}: \min_w \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^n \left[\max_{\hat{y}}\{\Delta(y_i, \hat{y}) + \langle w, \Psi(x_i, \hat{y})\rangle\} - \langle w, \Psi(x_i, y_i)\rangle\right] \tag{7}$$

Note that the optimization problem is now unconstrained as we have replaced the slack variables by their lower bounds as dictated by the inequalities and hence the inequalities in 6 become redundant.

An example of $\text{SSVM}_1^{\Delta m}$ is a Winner-Takes-All (WTA) classification. To illustrate this, we define $\Psi(x, y) = \phi(x) \otimes \Lambda^c(y)$ where $\phi(x) \in \mathbb{R}^D$ represents some transformation of the original feature vector x. $\Lambda^c(y)$ is a canonical representation of labels $y \in \mathcal{Y} = \{1, 2, \cdots, K\}$ and is defined as follows:

$$\Lambda^c(y) = \left((\delta(y_k, y))_{k=1}^K\right)', \text{ where } \delta(y_k, y) = 1, \text{ if } y = y_k \text{ and } 0, \text{ otherwise.} \tag{8}$$

If $F(x,y;w) = \langle w, \Psi(x,y)\rangle$ and $w = \left((v'_k)_{k=1}^K\right)' : v_k \in \mathbb{R}^D \ \forall k$, then it is easy to show that $F(x,y_k;w) = \langle w, \Psi(x,y_k)\rangle = \langle v_k, \phi(x)\rangle$. For a binary classification problem ($K = 2$) with 0-1 loss function, the constraints in SSVM_1 can be written as:

$$\langle v_{y_i}, \phi(x_i)\rangle - \langle v_y, \phi(x_i)\rangle \geq 1 - \xi_i \ \forall i, \ y \neq y_i, \tag{9}$$

which essentially are the constraints if the standard SVM problem is written as two separate regression problems (corresponding to positive and negative class) with 0-1 loss function.

3 Latent Structural Support Vector Machine

In many structured prediction tasks where SSVM is applied, modeling information is often not available as part of the training data. For example, in noun phrase co-reference resolution [3], although the clustering y of noun-phrases for a training document x is provided, the set of informative links that connects the noun phrases together into clusters are not explicitly specified. In another application of machine translation [2], translations y of sentence x are provided but not the informative links like parse trees and word alignments etc. In such domains, if these informative links are included in the learning framework, the predictive accuracy goes up. LSVM [21] is a discriminative learning approach with hidden variables especially designed to capture such informative links among output structures.

In LSVM, the discriminant function is modified from that of SSVM to account for the hidden variables and is defined as a mapping from $\mathcal{X} \times \mathcal{H} \times \mathcal{Y}$ to $\mathbb{R}$. Here, $\mathcal{H}$ denotes the set of hidden variables. The predictor $f : \mathcal{X} \to \mathcal{Y}$ is also redefined as $f(x;w) = \arg\max_{y\in\mathcal{Y}, h\in\mathcal{H}} \langle w, \Psi(x,h,y)\rangle$. Following the same analysis as SSVM, we can derive the following inequality for LSVM:

$$\Delta(y_i, \hat{y}_i) \leq \left[\max_{\hat{y},\hat{h}} \{\Delta(y_i,\hat{y}) + \langle w, \Psi(x_i,\hat{h},\hat{y})\rangle\} - \max_{\hat{h}} \langle w, \Psi(x_i,\hat{h},y_i)\rangle\right] \leq \xi_i \tag{10}$$

From the above equation, it is easy to see that the objective function in 7 can be modified for LSVM as:

$$\text{LSVM} : \min_w \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^n \left[\max_{\hat{y},\hat{h}} \{\Delta(y_i,\hat{y}) + \langle w, \Psi(x_i,\hat{h},\hat{y})\rangle\} - \max_{\hat{h}} \langle w, \Psi(x_i,\hat{h},y_i)\rangle\right] \tag{11}$$

The objective function is a difference of two convex functions. These are $f(w) = \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^n \left[\max_{\hat{y},\hat{h}} \{\Delta(y_i,\hat{y}) + \langle w, \Psi(x_i,\hat{h},\hat{y})\rangle\}\right]$ and $g(w) = \frac{C}{n}\sum_{i=1}^n \max_{\hat{h}} \langle w, \Psi(x_i,\hat{h},y_i)\rangle$. The difference of two convex functions is not convex in general and Yu & Joachims [21] used CCCP [22] to solve the optimization problem. However, NRBM (discussed in section 6.6) could also be used for solving this optimization problem. We will review CCCP in section 5. But before that, let us discuss about how the problem in SSVM can be solved in the dual space.

4 Solving the Dual Problem in SSVM

Tsochantaridis *et al* [19] suggested a very efficient optimization algorithm to solve the dual problem in SSVM. We will only concentrate on the margin rescaled SSVM (SSVM_1^m) here. Algorithms for other variants of SSVM differ very slightly from that of SSVM_1^m and are beyond the scope of this paper.

The objective of the dual problem of 6 is given by:

$$\mathbf{\Theta}(\alpha) = -\frac{1}{2}\sum_{i,y\neq y_i}\sum_{j,\bar{y}\neq y_j} \alpha_{(iy)}\alpha_{(j\bar{y})}J_{(iy)}J_{(j\bar{y})} + \sum_{i,y\neq y_i} \alpha_{(iy)}\Delta(y_i,y) \tag{12}$$

$$\tag{13}$$

where $J_{(iy)}J_{(j\bar{y})} = \langle \delta\Psi_i(y), \delta\Psi_j(\bar{y})\rangle$ and $\alpha_{(iy)}$ denotes the Lagrangian multiplier corresponding to the margin constraint for point x_i and label $y \neq y_i$. The dual QP, based on this objective, is given by:

$$\max_\alpha \mathbf{\Theta}(\alpha) \text{ s.t. } \alpha \geq 0 \text{ and } \sum_{i,y\neq y_i} \alpha_{(iy)} \leq \frac{C}{n} \ \forall i. \tag{14}$$

The algorithm for solving this dual is given in Algorithm 1. In essence, for each training data, a working set S_i is maintained which keeps account of the constraints selected so far. Whenever some output constraint is violated by more than ϵ margin, it is included in the working set and the optimization in 12 is performed w.r.t. the Lagrangian multipliers that belong to the working set. The optimization in dual is argued to have more advantage compared to the primal QP as,

1. Inner products in the joint feature space allows kernelization.

2. The constraint matrix follows a block diagonal structure where each block corresponds to a training instance. This facilitates a parallel implementation of the optimization framework where the parallelization can be made over instances.

Algorithm 1 Algorithm for Learning SSVM$_1^{\Delta m}$

Input: $\{(x_i, y_i)\}_{i=1}^n, C, \epsilon$.
$S_i \leftarrow \emptyset \ \forall i \in \{1, 2, \cdots, n\}$
repeat
  for $i = 1, \cdots, n$ **do**
1. set up cost function $H(y) = \Delta(y_i, y) - \langle \sum_j \sum_{y' \in S_j} \alpha_{(jy')} \delta\Psi_j(y'), \delta\Psi_i(y) \rangle$
2. compute $\hat{y} = \arg\max_{y \in \mathcal{Y}} H(y)$
3. compute $\xi_i = \max 0, max_{y \in S_i} H(y)$.
4. **if** $H(\hat{y}) > \xi_i + \epsilon$ **then**
5. $S_i \leftarrow S_i \cup \{\hat{y}\}$.
6(a). Variant (a): Solve 14 over $S = \cup_i S_i$.
6(b). Variant (b): Solve 14 over S_i.
7. **end if**
  end for
until no S_i has changed during iteration.

The fact that the above optimization procedure really maximizes the dual objective is given by the following proposition:

Proposition For SSVM$_1^{\Delta m}$ steps 6(a) and 6(b) in Algorithm 1, the improvement $\delta\mathbf{\Theta}$ of the dual is lower bounded by $\delta\mathbf{\Theta} \geq \frac{\epsilon^2}{8R_i^2}$ where $R_i = \max_y ||\delta\Psi_i(y)||$.

We now introduce another key theorem from [19]:

Theorem 4.1. *For a given $\epsilon > 0$, Algorithm 1 terminates after incrementally adding at most* $\max\{\frac{2n\bar{\Delta}}{\epsilon}, \frac{8C\bar{\Delta}\bar{R}^2}{\epsilon^2}\}$ *constraints to the working set S for SSVM$_1^{\Delta m}$. Here $\bar{R} = \max_i R_i$ and $\bar{\Delta} = \max_i \Delta_i$.*

The efficiency of the optimization framework can be noticed from the last theorem. The number of constraints added to the working set is independent of $|\mathcal{Y}|$ which can potentially be infinity. However, the actual number of constraints depends on the chosen representation, more specifically, on the upper bound of $||\Psi(x_i, y) - \Psi(x_i, \bar{y})||^2$ for any $y, \bar{y} \in \mathcal{Y}$. Also, the algorithm converges with a rate of $O(1/\epsilon^2)$. It is interesting to note that adding constraints one-by-one in the dual space corresponds to adding cutting planes in the primal space corresponding to the convex objective. In that sense, solving in the dual space using the proposed algorithm and solving using CPM in the primal space are equivalent. However, primal problems, in general, will not allow kernelization. But this is not a big issue anymore as explained in the introduction. In sections 5 and 6, we will, therefore, concentrate only on solving the primal problems which, in case of LSVM, is non-convex in general.

5 Solving the Optimization Problem in LSVM using CCCP

As pointed out in section 3, the objective in LSVM involves the difference of two convex functions $f(w)$ and $g(w)$ and can be solved using CCCP – first introduced by Yuille & Rangarajan [22]. CCCP is built on the observation that any function with bounded Hessian can be written as a sum of a convex and a concave function. If any arbitrary function $J(w)$ (bounded below) can be written as $J(w) = f(w) - g(w)$ with both $f(w)$ and $g(w)$ being convex, then an update of the form $\nabla f(w_{t+1}) = \nabla g(w_t)$ will monotonically decrease the function $J(w)$ in each iteration and hence will converge to a minimum or saddle point of $J(w)$. The proof of this is very simple and can be found in [22]. An illustration of the update is shown in figure 1 where the two convex functions correspond to $f(w)$ (the upper curve) and $g(w)$ (the lower curve). The algorithm starts with a point w_0, calculates the gradient $\nabla g(w_0)$ and tries to find the point w_1 such that $\nabla f(w_1) = \nabla g(w_0)$. On the next step, $\nabla g(w_1)$ is computed and w_2 is chosen such that $\nabla f(w_2) = \nabla g(w_1)$.

The algorithm has similarities with some other more general work which shows that any arbitrary closed set is the projection of a difference of two convex sets in a space with one more dimension (*e.g.* see `http://www.mai.liu.se/Opt/MPS/News/tuy.html`). One interesting theorem in [22] which connects variational bounding algorithms like surrogate functions and majorization [11], lower-bound maximization [13] with CCCP is provided below:

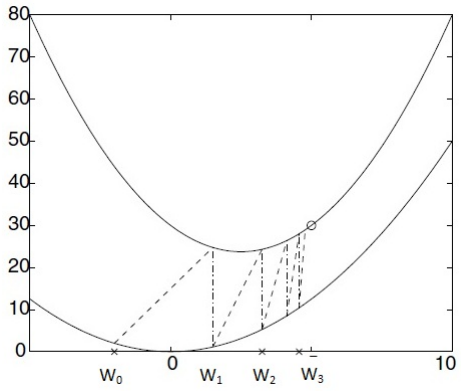

Figure 1: Concave-Convex Procedure

Theorem 5.1. *Any CCCP algorithm to extremize $J(w)$ (with bounded Hessian) can be expressed as variational bounding by first decomposing $J(w)$ as a convex function $f(w)$ and a concave function $-g(w)$ and then at each iteration starting at w_t set $\hat{J}_t(w) = f(w) - g(w_t) - \langle w - w_t, \nabla g(w_t)\rangle$. $\hat{J}_t(w) \geq J(w)$ is the function minimized in variational bounding.*

Based on this theorem, a variant of CCCP can be provided which is shown in Algorithm 2. In particular,

Algorithm 2 Concave-Convex Procedure (CCCP)

Set t = 0 and initialize w_0.
repeat
1. Find hyperplane v_t such that $-g(w) \leq -g(w_t) + \langle w - w_t, v_t\rangle \ \forall w$.
2. Solve $w_{t+1} = \arg\min_w f(w) + \langle w, v_t\rangle$.
3. $t = t + 1$.

until $[f(w_t) - g(w_t)] - [f(w_{t-1}) - g(w_{t-1})] < \epsilon$.

solving the optimization problem in LSVM involves computing the following in step 1 of Algorithm 2:

$$\hat{h}_i^* = \arg\max_{\hat{h}\in\mathcal{H}} \langle w, \Psi(x_i, \hat{h}, y_i)\rangle \ \forall i. \tag{15}$$

The hyperplane constructed is given by: $v_t = \sum_{i=1}^{n} \Psi(x_i, \hat{h}_i^*, y_i)$. Computation of the new iterate w_{t+1} in step 2 is done by solving the following optimization problem:

$$\min_w \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^{n}\left[\max_{\hat{y},\hat{h}}\{\Delta(y_i,\hat{y}) + \langle w, \Psi(x_i,\hat{h},\hat{y})\rangle\} - \max_{\hat{h}}\langle w, \Psi(x_i,\hat{h}_i^*, y_i)\rangle\right] \tag{16}$$

Yu & Joachims [21] used proximal bundle method for solving this optimization problem. In essence, the optimization alternates between minimizing over $\{\hat{h}_i\}_{i=1}^n$ and w in each iteration.

6 Bundle Method for Regularized Risk Minimization

Bundle methods originated from the cutting plane method (CPM – [7]) where the key idea is to lower bound a convex objective by piece-wise linear approximations. Before delving deeper into this topic, let's recall the following definitions.

Definition 3. ***Subgradient:*** *s' is called a subgradient of a convex function J at w' (with the assumption that $J(w') < +\infty$) iff $J(w) \geq J(w') + \langle w - w', s'\rangle \ \forall w$.*

Definition 4. ***Subdifferential:*** *The set of all subgradients of a convex function J at w' (with the assumption that $J(w') < +\infty$) are called subdifferentials and is conveniently denoted by $\partial_w J(w')$.*

6.1 Cutting Plane Method

CPM, in an iterative method, tries to optimize the lower bound of the original objective $J(w)$ given by:

$$J_t^{\text{CP}}(w) := \max_{1\leq i\leq t}\{J(w_i - 1) + \langle w - w_{i-1}, s_i\rangle\}, \tag{17}$$

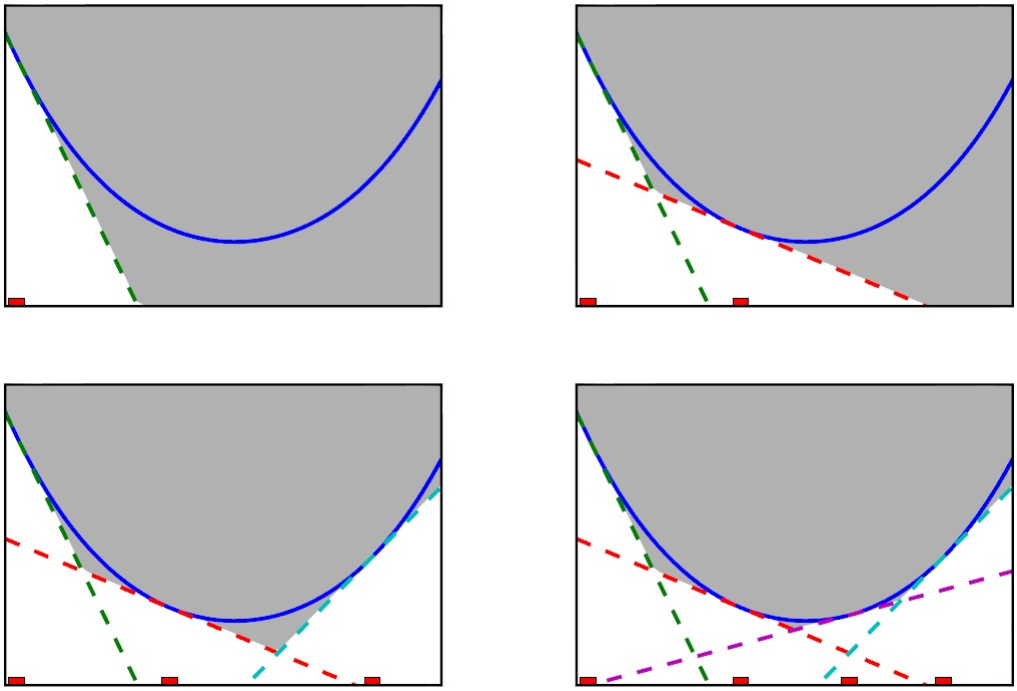

Figure 2: Explaining Cutting Plane Method – at each iteration the piecewise linear lower bound is minimized and a new linearization is added at the minimizer (red rectangle).

where s_i is the subgradient evaluated at location w_{i-1} in the i^{th} iteration. It is obvious that $J_t^{\text{CP}}(w)$ is the point-wise maximum of the supporting hyperplanes discovered till the t^{th} iteration. Suppose, $w_t := \arg\min_w J_t^{\text{CP}}(w)$. If w^* is the optimal solution of $J(w)$ then, by assumption, $J(w_i) \geq J(w^*)\ \forall i$ and, as a consequence, $\min_{1 \leq i \leq t} J(w_i) \geq J(w^*)$. Also, $J(w) \geq J_t^{\text{CP}}(w)\ \forall w$ which follows from the fact that the duality gap is always non-negative. Therefore, $\min_{1 \leq i \leq t} J(w_i) \geq J(w^*) \geq J_t^{\text{CP}}(w_t)\ \forall t$ and hence the gap around the optimal solution w^* is squeezed more and more in successive iterations. This procedure is continued till the gap $\epsilon_t = \min_{1 \leq i \leq t} J(w_i) - J_t^{\text{CP}}(w_t)$ goes below a pre-defined threshold ϵ. Note that the initialization in such process might be a bit tricky as taking argmin over hyperplanes might lead the solution to negative infinity and one has to be careful while implementing. Moreover, convergence in CPM can be slow when a new solution moves far away from an old solution causing a "zig-zag" behavior in the iterates (as it is not guaranteed that $J(w_{t+1}) \leq J(w_t)\ \forall t$). This is precisely the reason that led researchers to develop more efficient versions of CPM one of which is proximal bundle method. Figure 2 shows first four iterations of approximating a convex function by supporting hyperplanes.

6.2 Proximal Bundle Method

In proximal bundle method [9], large steps between successive iterates are prevented with the help of a "prox function" (or according to machine learning jargon a "regularizer") which is added to the objective $J_t^{\text{CP}}(w)$ and the value of the solution $\bar{w}_t$ at the t^{th} iteration is calculated according to the following equation:

$$\bar{w}_t := \arg\min_w \{\frac{\zeta_t}{2}||w - \hat{w}_{t-1}||^2 \ + J_t^{\text{CP}}(w)\} \tag{18}$$

It can be proved that $\bar{w}_t - w_{t-1}$ is a descent direction for the objective $\{\frac{\zeta_t}{2}||w - \hat{w}_{t-1}||^2 \ + J_t^{\text{CP}}(w)\}$ and hence a line search (optionally) can be performed along this direction to get the value of the next iterate w_t. Based on the reduction of the objective function, a decision can be made whether to update the current iterate $\hat{w}_{t-1}$ or not which corresponds to the "serious step" and "null step" respectively. Moreover, one needs to update the parameter ζ_t after each iteration for faster convergence (though, theoretically, this is optional). Suppose temporarily that $\zeta_k = \bar{\zeta} > 0\ \forall k$. As argued in [9], if $\bar{\zeta}$ is very large, almost all steps will become serious and will result in slow descent. On the other hand, a small $\bar{\zeta}$ will result in each serious step being followed by many null steps. The steps of the algorithm are listed in Algorithm 3. Interestingly, apart from proximal bundle method, there are two more methods that penalize large gaps between iterates with different forms of prox functions. These are i) trust region [17] and ii) level set [12].

Algorithm 3 Proximal Bundle Method

Input: $\epsilon \geq 0$, $\rho \in (0, 1)$, w_0.
Set t = 0 and initialize $\hat{w}_0 = w_0$.
repeat
1. $t = t + 1$.
2. Compute $J(w_{t-1})$ and $s_t \in \partial_w J(w_{t-1})$.
3. Update model: $J_t^{\text{CP}}(w) = \max_{1 \leq i \leq t}\{\langle w - w_{i-1}, s_i \rangle + J(w_{i-1})\}$.
4. $\bar{w}_t = \arg\min_w J_t^{\text{CP}}(w) + \frac{\zeta_t}{2}||w - \bar{w}_{t-1}||^2$.
5. $\epsilon_t = J(\hat{w}_{t-1}) - \left[J_t^{\text{CP}}(\bar{w}_t) + \frac{\zeta_t}{2}||\bar{w}_t - \bar{w}_{t-1}||^2\right]$
6. **if** $\epsilon_t < \epsilon$ **return** $\bar{w}_t$
7. Line Search: $\eta_t = \arg\min_{\eta \in \mathbb{R}} J(\hat{w}_{t-1} + \eta(\bar{w}_t - \hat{w}_{t-1}))$.
8. $w_t = \hat{w}_{t-1} + \eta_t(\bar{w}_t - \hat{w_{t-1}})$.
9. **if** $J(\bar{w}_{t-1}) - J(w_t) \geq \rho\epsilon_t$ **then**
10. SERIOUS STEP: $\bar{w}_t = w_t$.
11. **else**
12. NULL STEP: $\bar{w}_t = \bar{w}_{t-1}$.

end if .
end loop.

6.3 Bundle Method for Regularized Risk Minimization

BRBM was proposed for solving the regularized risk minimization problem which appear in different machine learning algorithms like SVM, Gaussian Processes [16], Logistic Regression, Conditional Random Fields [10], LASSO and so on. The function $J(w)$, in such cases, consist of two terms – one denoting some regularizer $\Omega(w)$ and the other one involves some empirical risk $R(w)$ and can be written as:

$$J(w) = \lambda\Omega(w) + R(w). \tag{19}$$

where λ is some relative weight assigned to the regularizer. Teo *et al* [18] identified that the machineries of proximal bundle method fits exactly this form of the objective function. One can recover proximal bundle method by replacing λ and $\Omega(w)$ by ζ_t and $||w - \hat{w}_{t-1}||^2$ respectively. The process is depicted in Algorithm 4. Note that, unlike proximal bundle method, the optimization process does not involve any "serious step" or "null step". In fact, this method can be viewed as a special case of proximal bundle method where the "proximal center" is always set at zero and every step is a "null step". One can also add some option of line search in

Algorithm 4 BMRM

Input: $\epsilon \geq 0$.
Set t = 0 and initialize w_0.
repeat
1. $t = t + 1$.
2. Compute $a_t \in \partial_w R(w_{t-1})$ and $b_t = R(w_{t-1}) - \langle w_{t-1}, a_t \rangle$.
3. Update model: $R_t^{\text{CP}}(w) = \max_{1 \leq i \leq t}\{\langle w, a_i \rangle + b_i\}$.
4. $w_t = \arg\min_w J_t(w)$.
5. $\epsilon_t = \min_{0 \leq i \leq t} J(w_i) - J_t(w_t)$.

until $\epsilon_t \leq \epsilon$.
return w_t.

the optimization framework whenever it is cheap enough (or exact). This will only boost the convergence speed which Franc & Sonnenburg [6] has validated empirically. As BMRM with line search generalizes this work, we will try to review it first in the next section and then return back to some other interesting aspects of BMRM and related algorithms.

6.4 Optimized Cutting Plane Algorithm for Support Vector Machines

A potential problem in CPM is the large gap between iterates which can be prevented using proximal bundle methods. However, there is no guarantee that the cutting plane discovered in some t^{th} step will be an active constraint in the vicinity of w^* (although it eventually will be due to the convergence of CPM). Neither it is certain that the new solution w_{t+1} will actually improve upon the original objective $J(w)$ at point w_t. To speed up the convergence of the proximal bundle method in linear SVM, Franc & Sonnenburg [6] proposed the optimization framework OCAS shown in Algorithm 5. Their work is based on optimizing the primal problem

in linear SVM which takes the following form:

$$J(w) = \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^{n}\max\{0, 1 - y_i\langle w, x_i\rangle\} \tag{20}$$

A reduced problem of the above is given by the following:

$$\hat{J}_t(w) = \frac{1}{2}||w||^2 + \frac{C}{n}R_t(w), \tag{21}$$

where $R_t(w)$ is a piece-wise linear lower bound on the convex risk term in equation 20. Unlike CPA, OCAS aims

Algorithm 5 Optimized Cutting Plane Algorithms for SVM

Input: $\{(x_i, y_i)\}_{i=1}^n, \epsilon, C$
Set t = 0 and initialize $w_0^b = 0$.
repeat
1. Compute w_t by solving the reduced problem in 21.
2. Compute a new best-so-far solution w_t^b using the line search $w_t^b = \min_{k\geq 0} J(w_{t-1}^b(1-k) + w_t k)$.
3. Add a new cutting plane that approximates the risk R at the point $w_t^c = w_t^b(1-\lambda) + w_t\lambda$ where $\lambda \in (0, 1]$ is a pre-defined parameter.
4. $t = t + 1$.
5. **until** $J(w_t^b) - \hat{J}_t(w_t) \leq \epsilon$.

at optimizing both $J(w)$ and $\hat{J}_t(w)$ simultaneously. As obvious from the steps of the algorithm, it maintains the best-so-far solution (w_{t-1}^b). The new best-so-far solution (w_t^b) after the current (t^{th}) iteration is selected by a line search along the direction of the hyperplane joining the reduced problem's current solution w_t and w_{t-1}^b. The new cutting plane is then selected to approximate $J(w)$ at a point w_t^c lying in the vicinity of w_t^b (see step 3 in Algorithm 5). This way, OCAS tries to select cutting planes that can actively contribute to the approximation of $J(w)$ around the optimum value w^*. The following theorem deals with the convergence of OCAS:

Theorem 6.1. *For any $\epsilon > 0$, $C > 0$, $\lambda \in (0, 1]$, and any training set $\{(x_i, y_i)\}_{i=1}^n$, Algorithm 5 satisfies the stopping condition in step 5 after at most $\max\{\frac{2C}{\epsilon}, \frac{8C^3Q^2}{\epsilon^2}\}$ iterations where $Q = \max_i ||x_i||$.*

6.5 BMRM with Line Search

As pointed out earlier in section 6.3, BMRM with line search is just a generalization of OCAS for a convex regularizer $\Omega(w)$ and convex empirical risk $R(w)$. The steps of the optimization framework is shown in Algorithm 6 and one can easily identify the connection of OCAS and BMRM with line search.

Algorithm 6 BMRM with Line Search

Input: $\epsilon \geq 0$, $\Theta \in (0, 1]$, w_0^b.
Set t = 0 and initialize $w_0^c = w_0^b$.
repeat
1. $t = t + 1$.
2. Compute $a_t \in \partial_w R(w_{t-1}^c)$ and $b_t = R(w_{t-1}^c) - \langle w_{t-1}^c, a_t\rangle$.
3. Update model: $R_t^{\text{CP}}(w) = \max_{1\leq i\leq t}\{\langle w, a_i\rangle + b_i\}$.
4. $w_t = \arg\min_w J_t(w)$.
5. Line Search: $\eta_t = \arg\min_{\eta\in\mathbb{R}} J(w_{t-1}^b + \eta(w_t - w_{t-1}^b))$.
6. $w_t^b = w_{t-1}^b + \eta_t(w_t - w_{t-1}^b)$.
7. $w_t^c = w_t^b(1-\theta) + \theta w_t$.
8. $\epsilon_t = \min_{0\leq i\leq t} J(w_i) - J_t(w_t)$.

until $\epsilon_t \leq \epsilon$.
return w_t^b.

However, the key contribution of Teo *et al.* is the analysis of the convergence of BMRM which they proved to be either $O(1/\epsilon)$ or $O(\log(1/\epsilon))$ under some suitable assumptions, while all of the earlier methods were shown to exhibit $O(1/\epsilon^2)$ rate of convergence. The theorems supporting these claims are given below:

Theorem 6.2. *Assume that* $\max_{u \in \partial_w R(w)} ||u|| \leq G \ \forall w \in dom(J)$. *Also assume that* Ω^* *has bounded curvature, that is,* $||\partial^2_\mu \Omega^*(\mu)|| \leq H^* \ \forall \mu \in \lambda^{-1} \sum_{i=1}^{t+1} \alpha_i a_i : \alpha_i \geq 0, \forall i$ *and* $\sum_{i=1}^{t+1} \alpha_i = 1$. *In this case we have:*

$$\epsilon_t - \epsilon_{t+1} \geq \frac{\epsilon_t}{2} min(1, \lambda \epsilon_t / 4G^2 H^*) \tag{22}$$

Furthermore, if $||\partial^2_w J(w)|| \leq H$, *then we have:*

$$\epsilon_t - \epsilon_{t+1} \geq \begin{cases} \epsilon_t/2 & \text{if } \epsilon_t > 4G^2H^*/\lambda \\ \lambda/8H^* & \text{if } 4G^2H^*/\lambda \geq \epsilon_t \geq H/2 \\ \lambda\epsilon_t/4HH^* & \text{otherwise.} \end{cases} \tag{23}$$

Theorem 6.3. *Assume that* $J(w) \geq 0 \ \forall w$. *Under the assumptions of Theorem 6.2, we can give the following convergence guarantee for Algorithm 4. For any* $\epsilon < 4G^2H^*/\lambda$, *the algorithm converges to the desired precision after:*

$$n \leq log_2 \frac{\lambda J(0)}{G^2 H^*} + \frac{8G^2 H^*}{\lambda \epsilon} - 1 \tag{24}$$

steps. Furthermore if the Hessian of $J(w)$ *is bounded, convergence to any* $\epsilon \leq H/2$ *takes at most the following number of steps:*

$$n \leq log_2 \frac{\lambda J(0)}{4G^2 H^*} + \frac{4H^*}{\lambda} \max[0, H - 8G^2H^*/\lambda] + \frac{4HH^*}{\lambda} log(H/2\epsilon). \tag{25}$$

Also, under the assumptions of Theorem 6.2, Algorithm 6 converges to the desired precision ϵ *after* $n \leq \frac{8G^2H^*}{\lambda\epsilon}$ *steps for any* $\epsilon < \frac{4G^2H^*}{\lambda}$.

6.6 Non-Convex Regularized Bundle Method

In Teo *et al*'s formulation, it is assumed that the empirical risk is convex. However, in many applications and particularly in LSVM, the empirical risk is not convex and CCCP seemed like the only tool to use until Do & Artières [4] came out with Non-convex Regularized Bundle Method (NRBM). To explain NRBM, let's consider a quadratic regularized risk function given as follows:

$$J(w) = \frac{\lambda}{2}||w||^2 + R(w). \tag{26}$$

The key idea in NRBM is pretty simple. BRBM fails to minimize a non-convex function as the constructed hyperplanes in successive iterations are not guaranteed to underestimate the objective for all values of w. One particular consequence of the situation is shown in figure 3. After the hyperplanes at w_1 and w_2 are constructed, optimization using the point-wise supremum of the approximate objective leads to w_3 which is a local maxima. In particular, when a hyperplane $c_{w'} = \langle a_{w'}, w \rangle + b_{w'}$ overestimates the objective at some point w, a "conflict" is said to have occurred between $c_{w'}$ and w. Standard strategy so far to overcome this kind of conflict has been lowering the cutting plane [8] (by adjusting the offset $b_{w'}$) such that it underestimates the objective at the solutions of all the iterations performed so far. Note that in BRBM, the approximate objective $J_t^{\text{CP}}(w)$ increases after every iteration by construction. However, this is not guaranteed anymore if the cutting planes are lowered after every iteration for non-convex risks. Consequently, the convergence rate is not defined.

We now describe in detail the steps of NRBM shown in Algorithms 7 and 8. Instead of solving "conflict" of a cutting plane with all the solutions so far, NRBM tries to solve the "conflict" of a newly discovered cutting plane with the best-so-far solution w_t^b. Therefore, NRBM only cares whether the following condition is satisfied or not:

$$c_{w_t}(w_t^b) = \langle a_{w_t}, w_t^b \rangle + b_{w_t} \leq R(w_t^b). \tag{27}$$

If the above inequality is not satisfied NRBM searches for a hyperplane $c_t(w_t)$ which satisfies the following equation:

$$\frac{\lambda}{2}||w_t||^2 + c_t(w_t) \geq J(w_t^b). \tag{28}$$

This phase is called "SolveConflict". Equation 28 is essential for defining the convergence of the algorithm and details of the convergence analysis can be found in the original paper [4]. The two conditions given in equations 27 and 28 can be rewritten as:

$$\begin{aligned} b_t &\leq R(w_t^b) - \langle a_{w_t, w_t^b} \rangle = U \\ b_t &\geq J(w_t^b) - \frac{\lambda}{2}||w_t||^2 - \langle a_{w_t, w_t} \rangle = L, \end{aligned} \tag{29}$$

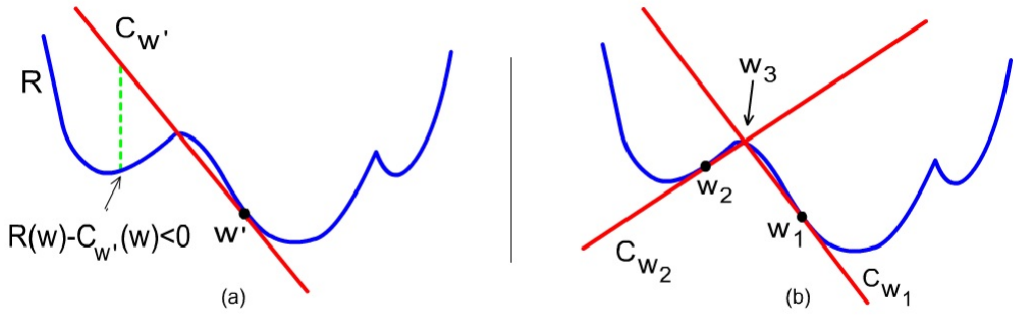

Figure 3: Explaining overestimation of non-convex objective by cutting planes

Algorithm 7 NRBM

Input: $\epsilon \geq 0, \lambda$.
Set t = 0 and initialize w_0.
repeat
1. $t = t + 1$.
2. Define $c_{w_t} = \langle a_{w_t}, w \rangle + b_{w_t}$ s.t. $c_{w_t}(w_t) = R(w_t)$ and $a_{w_t} \in \partial_w R(w_t)$.
3. $w_t^b = \arg\min_{w_j \in \{w_1, w_2, \cdots, w_t\}} J_t(w)$.
4. **if** condition 27 is not satisfied then
5. $c_t =$ SolveConflict (w_t^b, w_t, c_{w_t}).
6. **else**
7. $c_t = c_{w_t}$.
8. Compute $w_{t+1} = \arg\min_w J_t^{\mathrm{CP}}(w)$.
9. $\epsilon_t = J(w_t^b) - J_t^{\mathrm{CP}}(w_t)$.

until $\epsilon_t \leq \epsilon$.
return w_t^b.

where L and U define lower and upper bounds respectively on b_t. Clearly, if $L \leq U$, any value of $b_t \in [L, U]$ will work. However, if it is not true then tuning b_t alone is not enough and one has to tune both a_t and b_t to satisfy the constraints in 27 and 28. This tuning corresponds to steps 4 and 5 in Algorithm 8. The theorem concerned with the convergence of NRBM (similar to Theorem 6.3) is given below:

Theorem 6.4. *Algorithm 7 produces an approximation gap $\epsilon_T \leq \epsilon$ after T steps where $T \leq T_0 + \frac{8G^2}{\lambda\epsilon} - 2$ with $T_0 = 2log_2 \frac{\lambda||w_1 + a_1/\lambda||}{G} - 2$ and G being an upper bound on the norm of cutting planes direction parameters a_i. The algorithm converges with a rate $O(1/(\lambda\epsilon))$.*

7 Conclusion

It is interesting to note that the pool of the optimization algorithms (apart from CCCP and NRMB) described so far have their basis on the simple observation that a closed proper convex function can be described as point-wise supremum of its supporting hyperplanes. CCCP, on the other hand, is a far more general and elegant algorithm and have connections with many other optimization techniques as mentioned in section 5. NRBM is a general purpose tool for regularized empirical risk minimization where the risk can be non-convex.

Acknowledgement

I am thankful to Dr. Raymond J. Mooney and Sreangsu Acharyya for helpful discussions and advice.

Algorithm 8 SolveConflict

1. **Input:** w_t^b, w_t, c_{w_t} with parameters (a_{w_t}, b_{w_t}).
2. **Output:** c_t with parameters (a_t, b_t).
3. Compute L, U according to 29.
4. **if** $L \leq U$ **then** $[a_t, b_t] = [a_{w_t}, L]$.
5. **else** $[a_t, b_t] = \big[-\lambda w_t^b, J(w_t^b) - \frac{\lambda}{2}||w_t||^2 - \langle a_t, w_t \rangle \big]$.

Bibliography

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] Peter F. Brown, Vincent J.Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311, 1993.

[3] C. Cardie and K. Wagsta. Noun phrase coreference as clustering, 1999.

[4] Trinh-Minh-Tri Do and Thierry Artières. Large margin training for hidden markov models with partially observed states. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 265–272, New York, NY, USA, 2009. ACM.

[5] Shai Fine, Katya Scheinberg, Nello Cristianini, John Shawe-taylor, and Bob Williamson. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

[6] Vojtěch Franc and Soeren Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 320–327, New York, NY, USA, 2008. ACM.

[7] 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.

[8] Krzysztof C. Kiwiel. Methods of descent for non-differentiable optimization. 1985.

[9] Krzysztof C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46(1):105–122, Jan 1990.

[10] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[11] Kenneth Lange, David R. Hunter, and Ilsoon Yang. Optimization Transfer Using Surrogate Objective Functions. *Journal of Computational and Graphical Statistics*, 9(1), 2000.

[12] Claude Lemaréchal, Arkadii Nemirovskii, and Yurii Nesterov. New variants of bundle methods. *Math. Program.*, 69:111–147, July 1995.

[13] S. P. Luttrell. Partitioned mixture distributions: An adaptive bayesian network for low-level image processing. *IEEE Proceedings on Vision, Image and Signal Processing*, 141:251–260, 1994.

[14] John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.

[15] Ali Rahimi and Ben Recht. Random features for large-scale kernel machines. In *In Neural Infomration Processing Systems*, 2007.

[16] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[17] H. Schramm and J. Zowe. *A version of the Bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results.* Report. Mathematisches Inst., 1990.

[18] Choon Hui Teo, S. V. N. Vishwanathan, Alex J. Smola, and Quoc V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

[19] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, December 2005.

[20] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[21] Chun Nam John Yu and Thorsten Joachims. Learning structural SVMs with latent variables. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1169–1176, New York, NY, USA, 2009. ACM.

[22] A. L. Yuille and Anand Rangarajan. The concave-convex procedure (cccp). In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.