Partial Linearization based Optimization for Multi-class SVM

Pritish Mohapatra[1], Puneet Kumar Dokania[2], C.V. Jawahar[1] and M. Pawan Kumar[3]

[1] IIIT-Hyderabad, [2] CentraleSupelec & INRIA Saclay, [3] University of Oxford

Abstract. We propose a novel partial linearization based approach for optimizing the multi-class SVM learning problem. Our method is an intuitive generalization of the Frank-Wolfe and the exponentiated gradient algorithms. In particular, it allows us to combine several of their desirable qualities into one approach: (i) the use of an expectation oracle (which provides the marginals over each output class) in order to estimate an informative descent direction, similar to exponentiated gradient; (ii) analytical computation of the optimal step-size in the descent direction that guarantees an increase in the dual objective, similar to Frank-Wolfe; and (iii) a block coordinate formulation similar to the one proposed for Frank-Wolfe, which allows us to solve large-scale problems. Using the challenging computer vision problems of action classification, object recognition and gesture recognition, we demonstrate the efficacy of our approach on training multi-class SVMs with standard, publicly available, machine learning datasets.

Keywords: Multi-class SVM, Partial linearization, Optimization

1 Introduction

Many tasks in computer vision can be formulated as multi-class classification problems. In other words, given an image or a video, the task is to assign it a label that belongs to a specified finite set. For example, in the case of object recognition from an image, the label can be car, chair or person. Similarly, for action recognition from a video, actions categories like jumping, kicking or clapping can be candidate labels. There has been extensive research in the area of multi-class classification with a plethora of solutions being proposed [16, 18, 4, 2]. In this work, we focus on multi-class SVM (MC-SVM), which is one of the most popular methods for this task. The MC-SVM model provides a linear function that gives a score for each class. Given a test sample, its class is predicted by maximizing the score. During learning, the MC-SVM objective minimizes an upper bound on the empirical risk of the training data, for which we know the ground-truth labels. The risk is typically measured by the standard $0-1$ loss function. However, any other loss function can be easily substituted into the MC-SVM learning framework.

The size of the MC-SVM learning problem rapidly increases with the number of classes and size of the training dataset. In order to enable the use of MC-SVM with large scale problems, several optimization algorithms for minimizing its learning objective have been proposed in the literature. One of the most successful algorithms is a recent adaptation of the Frank-Wolfe algorithm [8]. Briefly, the algorithm solves the dual of the MC-SVM optimization problem iteratively. At each iteration, it obtains a descent direction by minimizing a linear approximation of the dual objective. It was shown in [10] that the computation of the descent direction corresponds to a call to the the so-called *max-oracle* for each sample. In other words, for each training sample, we maximize over the set of output classes with respect to the loss-augmented scores. As the max-oracle can be solved efficiently for the MC-SVM, the Frank-Wolfe algorithm can be effectively used to learn such models. There are two main advantages of the Frank-Wolfe algorithm. First, the optimal step-size in the descent direction can be computed analytically, thereby avoiding a tedious line search [10]. Second, it can be suitably modified to a block-coordinate version [14], where the max-oracle is solved for only one training sample at each iteration. The gain in efficiency obtained by this version does not affect the accuracy of the solution.

A key disadvantage of the Frank-Wolfe algorithm is that it only provides a very local approximation of the objective function with the aid of the max-oracle. In other words, it effectively focuses on one constraint (the most violated one) of the primal MC-SVM learning problem. In contrast, the exponentiated gradient algorithm [1] makes use of a more informative *expectation oracle*. To elaborate, instead of maximizing, it computes an expectation over the set of output classes with respect to a distribution parameterized by the loss-augmented scores. However, the exponentiated gradient algorithm suffers from the difficulty of choosing an optimal step-size, for which it has to resort to line search. Furthermore, despite the availability of a stochastic version of the algorithm, its worst-case time complexity is worse than that of the Frank-Wolfe algorithm.

In this paper, we propose a novel algorithm for optimizing the MC-SVM learning problem based on partial linearization [17]. Our algorithm provides a natural generalization to the Frank-Wolfe and the exponentiated gradient algorithms, thereby combining their desirable properties. Specifically, (i) it allows for the use of a potentially more informative descent direction based on the expectation oracle; (ii) it computes the optimal step-size in the descent direction analytically; and (iii) it can also be applied in a block coordinate fashion without losing the accuracy of the solution. We demonstrate the efficacy of our approach on the challenging computer vision problems of action classification, object recognition and gesture recognition using standard publicly available datasets.

In certain cases, our method can also be used for efficient optimization of the more general structured SVM (SSVM) models. Specifically, in case of output spaces that have a low tree-width structure, we can employ efficient max-oracles and expectation-oracles. This in turn means that similar to the Frank-Wolfe [10] and exponentiated gradient algorithms [1], our method can be effectively used for learning an SSVM model. We demonstrate this on the problem of handwritten word recognition using a chain structured output space.

2 Related Work

Several algorithms have been proposed for optimizing multi-class SVMs. Most of the popular methods are iterative algorithms that make use of efficient subroutines called *oracles* in each iteration [10, 1, 11, 19, 23]. The most popular algorithms can be bracketed into two classes depending on the type of oracles they use: ones that use the max-oracle [10, 11, 19]; and the ones that use the expectation oracle [1, 23].

A max-oracle sub-routine maximizes the loss-augmented score over the output space. In other words, given the current estimate of the parameters and a training sample, it returns the output that maximizes the sum of the classifier score and the loss. The sub-gradient descent algorithm [19] calls the max-oracle to compute the sub-gradient of the primal objective and uses it as the update direction in each iteration. The cutting-plane algorithm [11] uses the max-oracle to get the most violating constraint or the cutting plane. It accumulates the cutting planes to generate increasingly accurate approximations to the primal problem that it solves in each iteration. The recent adaptation [10] of the Frank-Wolfe algorithm to the MC-SVM and SSVM learning problems uses the max-oracle to compute the conditional gradient of the dual problem. All three aforementioned algorithms have a complexity of $O(1/\epsilon)$, where ϵ is the user-specified optimization tolerance. However, in practice, the block-coordinate Frank-Wolfe algorithm has been shown to provide faster convergence on a variety of problems [10].

In contrast to the max-oracle, the expectation-oracle computes an expectation over the output space with respect to a distribution parameterized by the loss-augmented scores. In [1], the expectation-oracle is used to make exponentiated gradient updates [12], which guarantees descent in each iteration. The Bregman projection based excessive gap reduction technique presented in [23] also uses the expectation oracle. While this algorithm has a highly competitive complexity of $O(1/\sqrt{\epsilon})$, the method does not work with noisy oracles and hence cannot lend itself to a stochastic or a block-coordinate version.

As will be seen shortly, our approach naturally generalizes over algorithms from both the categories with the use of a temperature hyperparameter. When the temperature is set to 0, the expectation oracle resembles the max-oracle and our method reduces to the Frank-Wolfe algorithm[10]. Importantly, for a non-zero temperature, the use of the expectation-oracle can provide us with a less local approximation of the objective function. Hence, for the multi-class SVM learning problem, it may be beneficial to use the expectation-oracle instead of the max-oracle. Another key aspect of our algorithm is that it chooses an optimal step-size at each iteration. If we instead fix the step-size to 1 in every iteration and use a non-zero value of the temperature hyperparameter, then our method reduces to the exponentiated gradient algorithm [1]. Moreover, unlike the cutting plane [11] and the excessive gap reduction [23] algorithms our approach allows for a block-coordinate version, which leads to faster rate of convergence without affecting the accuracy of the solution.

3 Preliminaries

3.1 The Multi-class SVM Optimization Problem

We provide a brief overview of the multi-class SVM (MC-SVM) optimization problem. Given an input $\mathbf{x} \in \mathcal{X}$ the aim of multi-class classification is to predict the output y that belongs to the output space $\mathcal{Y}$. If the number of classes is denoted by c, the output space $\mathcal{Y} = \{1, \dots, c\}$. Let the feature representation of sample $\mathbf{x}$ be $\boldsymbol{\varphi}(\mathbf{x})$, then a joint feature map $\Phi(\mathbf{x}, y) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}^d$ is defined as

$$\Phi(\mathbf{x}, y) = [\boldsymbol{v}_1^\top \ \dots \ \boldsymbol{v}_j^\top \ \dots \ \boldsymbol{v}_c^\top]^\top \tag{1}$$
$$where, \ \boldsymbol{v}_j = \begin{cases} \boldsymbol{\varphi}(\mathbf{x}) \text{ if } j = y, \\ \mathbf{0} \quad \text{otherwise.} \end{cases}$$

A multi-class SVM, parameterized by $\mathbf{w}$, provides a linear prediction rule as follows: $h_{\mathbf{w}}(\mathbf{x}) = \text{argmax}_{y \in \mathcal{Y}} \left(\mathbf{w}^\top \Phi(\mathbf{x}, y)\right)$. Given a set of labelled samples $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, the parameter vector $\mathbf{w}$ is learnt by solving the following convex optimization problem:

$$\min_{\mathbf{w}, \xi} \ \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i \tag{2}$$
$$\text{s.t.} \quad \mathbf{w}^\top \Psi_i(y) \geq \Delta(y_i, y) - \xi_i, \forall i \in [n], \forall y \in \mathcal{Y}$$

Here, $\Psi_i(y) = \Phi(\mathbf{x}_i, y_i) - \Phi(\mathbf{x}_i, y)$ and the loss incurred for predicting y, given the ground truth y_i for the sample $\mathbf{x}_i$, is defined as

$$\Delta(y_i, y) = \begin{cases} 0 \text{ if } y = y_i, \\ 1 \text{ otherwise.} \end{cases} \tag{3}$$

We use $[n]$ to denote the set $\{1, 2, \dots, n\}$ and shall use $\Delta_i(y)$ as a short hand for $\Delta(y, y_i)$. The Lagrangian dual of problem (2) is given by:

$$\min_{\boldsymbol{\alpha} \geq 0} \ T(\boldsymbol{\alpha}) = -\mathbf{b}^\top \boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^\top A^\top A \boldsymbol{\alpha} \tag{4}$$
$$\text{s.t.} \ \sum_{y \in \mathcal{Y}} \boldsymbol{\alpha}_{iy} = 1, \forall i \in [n].$$

Here the dual variable vector $\boldsymbol{\alpha}$ is of size $m = n \times c$; $\mathbf{b} \in \mathcal{R}^m$ is defined as $\mathbf{b} = \{b_{iy} = \frac{1}{n}\Delta_i(y) \mid i \in [n], y \in \mathcal{Y}\}$ and the matrix $A \in \mathcal{R}^{d \times m}$ is defined as $A = \{A_{iy} = \frac{1}{\lambda n}\Psi_i(y) \in \mathcal{R}^d \mid i \in [n], y \in \mathcal{Y}\}$.

It is possible to cheaply evaluate the objective of the primal MC-SVM formulation since the following problem lends itself to efficient optimization. Specifically, in order to compute the MC-SVM objective at a given set of parameters $\mathbf{w}$, we can solve the following problem for each sample i.

$$\bar{y}_i = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \, \Delta_i(y) - \mathbf{w}^\top \Psi_i(y). \tag{5}$$

Given $\overline{y}_i$, the value of the slack variable $\xi_i = \Delta_i(\overline{y}_i) - \mathbf{w}^\top \Psi_i(\overline{y}_i)$. We refer to the above problem as the *max-oracle*. Let $P(y)$ denote the probability distribution over the set of output classes, parameterized by the loss augmented scores, that is,

$$P(y) = \frac{\exp\left(\Delta_i(y) - \mathbf{w}^\top \Psi_i(y)\right)}{\sum_{y \in \mathcal{Y}} \exp\left(\Delta_i(y) - \mathbf{w}^\top \Psi_i(y)\right)}. \tag{6}$$

The max-oracle gives the most probable class according to the distribution $P(y)$. It has been shown through several works, including cutting-plane algorithms [11], subgradient descent [19] and Frank-Wolfe [14], that an inexpensive max-oracle is sufficient to minimize problem (2) and/or its Lagrangian dual (4) efficiently.

As we will see shortly, our work exploits the fact that, for multi-class classification problems, a related problem known as the *expectation-oracle* can be solved efficiently as well (with the same time complexity as the max-oracle). While the max-oracle gives the most probable class, the expectation-oracle returns an expectation over the complete output space with respect to the distribution $P(y)$. By cleverly exploiting this observation, we obtain a natural generalization of the Frank-Wolfe algorithm that retains many of its desirable properties such as: guaranteed descent direction, analytically computable optimal step size and guaranteed convergence even in block-coordinate mode. At the same time it also allows the use of the expectation-oracle to find a valid descent direction that can often lead to improved performance in practice.

3.2 Partial Linearization

Let us consider the following optimization problem with a convex and continuously differentiable objective $T(\boldsymbol{\alpha})$ defined over a compact and convex feasible set U: $\min_{\boldsymbol{\alpha} \in U} \quad T(\boldsymbol{\alpha})$. For this problem, Patriksson [17] proposes a framework that unifies several feasible-direction finding methods for non-linear optimization through the concept of partial linearization of the objective function. The idea of partial linearization is to construct a convex approximation to the original objective $T(\boldsymbol{\alpha})$ at each iteration. The approximation involves substituting the original function with a surrogate function. Furthermore, in order to model the difference between the original function and the surrogate function, we add a first order approximation of this difference.

Formally, at each iteration k, we solve the following problem:

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in U} \quad T^k(\boldsymbol{\alpha}) = {} & f(\boldsymbol{\alpha}, \boldsymbol{\alpha}^k) + T(\boldsymbol{\alpha}^k) - f(\boldsymbol{\alpha}^k, \boldsymbol{\alpha}^k) \\ & + [\nabla T(\boldsymbol{\alpha}^k) - \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^k, \boldsymbol{\alpha}^k)]^T (\boldsymbol{\alpha} - \boldsymbol{\alpha}^k). \end{aligned} \tag{7}$$

The term $f(\boldsymbol{\alpha}, \boldsymbol{\alpha}^k)$ denotes the surrogate function defined at the current solution $\boldsymbol{\alpha}^k$. The term $T(\boldsymbol{\alpha}^k) - f(\boldsymbol{\alpha}^k, \boldsymbol{\alpha}^k) + [\nabla T(\boldsymbol{\alpha}^k) - \nabla_{\boldsymbol{\alpha}} f(\boldsymbol{\alpha}^k, \boldsymbol{\alpha}^k)]^T(\boldsymbol{\alpha} - \boldsymbol{\alpha}^k)$ is the first order Taylor expansion of the actual error term $T(\boldsymbol{\alpha}^k) - f(\boldsymbol{\alpha}, \boldsymbol{\alpha}^k)$ and is used as an approximation for it. Patriksson [17] showed that the approximation proposed in equation (7) actually preserves the gradient of the original objective function. This guarrantees that a valid descent direction for the approximate problem (7) is also a valid descent direction for the original problem. The optimal solution $\bar{\boldsymbol{\alpha}}^k$ to problem 7 gives a descent direction. This allows us to update the solution as $\boldsymbol{\alpha}^{k+1} = (1-\gamma)(\boldsymbol{\alpha}^k) + (\gamma)(\bar{\boldsymbol{\alpha}}^k)$, where γ is the step-size that can be determined via line search in general. Interestingly, in some special cases, including the one considered in this work, the optimal step-size can also be computed analytically, which avoids the tedious line search. For convergence, $f(\mathbf{x}, \mathbf{y})$ has to be convex and continuously differentiable with respect to $\mathbf{x}$ and continuous with respect to $\mathbf{y}$. We adapt the partial linearization method for solving problem (4) in the following section.

4 Partial Linearization for Multi-class SVM Optimization

The dual multi-class SVM problem defined in problem (4) has a compact convex feasible set and has a continuously differentiable convex objective. This allows us to use the partial linearization method to solve the optimization problem. However, as the above description shows, partial linearization is a very general framework. For it to be applied successfully, we need to ensure that we make the right choice for the surrogate function. Specifically, the resulting problem (7) must lend itself to efficient optimization. Furthermore, in our case, we would like to ensure that problem (7) captures the information regarding how much each constraint of the primal multi-class SVM problem is violated, similar to the expectation-oracle. To this end, we define the surrogate function as follows:

$$f(\boldsymbol{\alpha}, \boldsymbol{\alpha}^k) = \frac{\tau}{n} \sum_{i \in [n]} \sum_{y \in \mathcal{Y}} \boldsymbol{\alpha}_{iy} \log(\boldsymbol{\alpha}_{iy}). \tag{8}$$

Here, τ is a non-negative hyperparameter, which we refer to as the temperature. In the following subsection, we show that for the above choice of surrogate function, the partial linearization approach generalizes both the Frank-Wolfe and the exponentiated gradient algorithm.

4.1 Partial linearization in the dual space

When we use the surrogate function defined in equation (8) for partial linearization of the dual multi-class SVM problem, the form of the update direction in the resulting optimization algorithm is described by the following proposition.

Proposition 1. *If the surrogate function is defined as*

$$f(\boldsymbol{\alpha}, \boldsymbol{\alpha}^k) = \frac{\tau}{n} \sum_{i \in [n]} \sum_{y \in \mathcal{Y}} \boldsymbol{\alpha}_{iy} \log(\boldsymbol{\alpha}_{iy}),$$

then the update direction $\mathbf{s}^k$ *in iteration* k *for given* i *and* $y \in \mathcal{Y}$ *can be computed as*

$$\mathbf{s}^k_{iy} = \frac{\exp\left(\log(\boldsymbol{\alpha}^{k-1}_{iy}) + \frac{1}{\tau}(\Delta_i(y) - {\mathbf{w}^{k-1}}^\top \Psi_i(y))\right)}{z_i}. \tag{9}$$

The proof of the above proposition is provided in Appendix 1.1 (supplementary material).

In equation 9, for each sample i, $\mathbf{s}^k_i(y)$ forms a probability distribution over the set of classes $\mathcal{Y}$. In the primal setting, this is equivalent to having an expectation over the entire output space as the update direction and is therefore similar to an expectation-oracle. In each iteration of the algorithm, given the update direction, we need to perform a line search to find the optimal step size γ in that direction. Since the dual multi-class SVM problem involves optimizing a quadratic function, it is possible to analytically compute the optimal step-size. The following proposition that gives the form of the optimal step size directly follows from the work of Jaggi *et al.* [14].

Proposition 2. *The optimal step-size along the update direction* $\mathbf{s}^k$ *can be computed to be equal to*

$$\gamma = \frac{< \boldsymbol{\alpha}^{k-1} - \mathbf{s}^k, -\mathbf{b} + A^\top A\boldsymbol{\alpha}^{k-1} >}{\lambda||A(\boldsymbol{\alpha}^{k-1} - \mathbf{s}^k)||^2}. \tag{10}$$

Here it should be observed that setting the temperature parameter τ to 0 results in a distribution $\mathbf{s}^k_i$ that has probability 1 for the label

$$\bar{y}_i = \underset{y \in \mathcal{Y}}{\text{argmax}} \left(L_i(y) - {\mathbf{w}^{k-1}}^T \Psi_i(y)\right)$$

and 0 elsewhere. This results in an update direction that is the same as that of the Frank-Wolfe algorithm and thus reduces the partial linearization method to the Frank-Wolfe algorithm [14]. Moreover, fixing the step-size γ to 1 for all iterations, reduces our approach to the exponentiated-gradient algorithm [1]. Hence, our partial linearization based approach for optimizing the MC-SVM problem generalizes both the Frank-Wolfe as well as the exponentiated-gradient algorithms. Importantly, the descent direction obtained using some $\tau > 0$ can be significantly better than that obtained using $\tau = 0$. This is illustrated in the following example.

In problem 4, let $n = 1$, $\lambda = 1$, $A = [2, 0, 0; 0, 1, 0; 0, 0, 3]$ and $b = [1; 1; 0]$. Assume, after the $(k-1)^{th}$ iteration of the optimization algorithm, the location in the feasible set is $\alpha^{k-1} = [0.125, 0.5, 0.5]^\top$. Now, if we take $\tau = 0$, the descent direction for the k^{th} iteration can be computed to be $\mathbf{s}^k_{\tau=0} = [1, 0, 0]^\top$. Similarly, for $\tau = 1$, the descent direction would be $\mathbf{s}^k_{\tau=1} = [0.199, 0.796, 0.005]^\top$. In each case, we take the optimal step in the descent direction. It can be verified that while the step along $s^k_{\tau=0}$ reduces the objective function by 0.5341, the step along $s^k_{\tau=1}$ reduces the objective function by a bigger value of 1.2550.

This is primarily due to the fact that the Frank-Wolfe algorithm ($\tau = 0$) constraints the descent directions to be only towards vertices of the feasible domain polytope. For instance in this example, $\mathbf{s}^k_{\tau=0}$ can only take values from among $\{[1,0,0]^\top, [0,1,0]^\top, [0,0,1]^\top\}$, which prevents it from taking a more direct path towards the optimal solution ($[0.25, 1, 0]^\top$) which lies on one of the facets of the polytope and hence away from the direction of any of the vertices. On the other hand, with $\tau > 0$, our algorithm can explore more direct descent paths towards the solution.

Algorithm 1 *Partial linearization for optimizing multi-class* SVM

```
 1: $\mathcal{D} = (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)$
 2: Initialize $\boldsymbol{\alpha}^0$ such that $\mathbf{w}(\boldsymbol{\alpha}^0) \sim [0]^d$, $k \leftarrow 1$
 3: repeat
 4:     for all $i \in [n]$ do
 5:         $\forall y \in \mathcal{Y}$,
 6:         $\mathbf{s}^k_{iy} \leftarrow \frac{\exp\left(\log(\boldsymbol{\alpha}^{k-1}_{iy}) + \frac{1}{\tau}(\Delta_i(y) - \mathbf{w}^{k-1\top}\Psi_i(y))\right)}{z_i}$
 7:     end for
 8:     Optimal step size, $\gamma \leftarrow \frac{<\boldsymbol{\alpha}^{k-1} - \mathbf{s}^k, -\mathbf{b} + A^\top A\boldsymbol{\alpha}^{k-1}>}{\lambda||A(\boldsymbol{\alpha}^{k-1} - \mathbf{s}^k)||^2}$
 9:     Update $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}^k \leftarrow (1-\gamma)\boldsymbol{\alpha}^{k-1} + (\gamma)\mathbf{s}^k$
10:     Update $\mathbf{w}$, $\mathbf{w}^k \leftarrow A\boldsymbol{\alpha}^k$
11:     $k \leftarrow k+1$
12: until Convergence
13: Optimal parameter, $\mathbf{w}$
```

The partial linearization algorithm for optimizing the dual MC-SVM problem is outlined in Algorithm 1. Step 6 in Algorithm 1 requires us to explicitly compute the update direction corresponding to each dual variable. For the MC-SVM problem, as the number of dual variables is a reasonable $(number\ of\ samples) \times (number\ of\ classes)$, $\mathbf{s}^k_{iy}$ can be efficiently computed for every sample $\mathbf{x}_i$ as the marginal probability of each class y. Once we have the update direction we take a step in that direction with optimal step-size γ as computed in Step 8. Then the dual and the primal variables are updated to complete an iteration of the algorithm.

4.2 Block-Coordinate Partial Linearization

In many tasks, it is very common to learn classification models using very large datasets. In such scenarios, learning an MC-SVM model using the partial linearization algorithm described in Algorithm 1 can be very slow. This is because, each update iteration of Algorithm 1 requires a pass through the entire dataset. In order to circumvent this expensive step, we present a block-coordinate version of the algorithm, which updates the model parameters after every single sample encounter. Algorithm 2 outlines the details of the block-coordinate partial linearization algorithm. The key difference is that, unlike Algorithm 1, Algorithm 2

Algorithm 2 *Block-Coordinate Partial linearization for optimizing multi-class* SVM

```
 1: $\mathcal{D} = (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)$
 2: Initialize $\boldsymbol{\alpha}^0$ such that $\mathbf{w}(\boldsymbol{\alpha}^0) \sim [0]^d$, $k \leftarrow 1$
 3: Initialize a $(d \times n)$ matrix $W$ such that $i^{th}$ column of $W$, $\mathbf{w}_i = \mathbf{w}(\boldsymbol{\alpha}_i^0)$
 4: repeat
 5:     Chose a random $i \in [n]$
 6:     $\forall y \in \mathcal{Y}$,
 7:     $\mathbf{s}_{iy}^k \leftarrow \frac{\exp\left(\log(\boldsymbol{\alpha}_{iy}^{k-1}) + \frac{1}{\tau}(\Delta_i(y) - \mathbf{w}^{k-1\top}\Psi_i(y))\right)}{z_i}$
 8:     Optimal step size, $\gamma \leftarrow \frac{<\boldsymbol{\alpha}_i^{k-1} - \mathbf{s}_i^k, -\mathbf{b} + A^\top A\boldsymbol{\alpha}_i^{k-1}>}{\lambda||A(\boldsymbol{\alpha}_i^{k-1} - \mathbf{s}_i^k)||^2}$
 9:     Update $\boldsymbol{\alpha}_i$, $\boldsymbol{\alpha}_i^k \leftarrow (1-\gamma)\boldsymbol{\alpha}_i^{k-1} + (\gamma)\mathbf{s}_i^k$
10:     Update $\mathbf{w}_i$, $\mathbf{w}_i^k \leftarrow A\boldsymbol{\alpha}_i^k$
11:     Update $\mathbf{w}$, $\mathbf{w}^k \leftarrow \mathbf{w}^{k-1} - \mathbf{w}_i^{k-1} + \mathbf{w}_i^k$
12:     $k \leftarrow k+1$
13: until Convergence
14: Optimal parameter, $\mathbf{w}$
```

does not have to loop through all the samples in the training set before updating the primal variable vector. Instead, we pick a random sample i from the training set (step 5) and compute the marginals just for this sample. Accordingly we update the primal weight vector $\mathbf{w}$ with this new marginal for sample i while the marginals for all other samples remain unchanged. This is similar to the coordinate descent method and is more efficient compared to the batch method as instead of solving n convex optimization problems, we have to solve only one in each iteration. As shown in the following proposition, this improvement in run-time does not affect the accuracy of the solution.

Proposition 3. *The block-coordinate partial linearization algorithm is guaranteed to converge to the global optima of the multi-class* SVM *learning problem.*

The proof of the above proposition is provided in Appendix 1.3 (supplementary material).

4.3 Partial Linearization for Structured SVM Optimization

The multi-class SVM solves a prediction problem in which the output space is a set of classes. However, for many tasks, the output space can have a more complicated structure. The structured SVM (SSVM), which is a generalization of the binary SVM to structured output spaces, can effectively model such structures. Given an input $\mathbf{x} \in \mathcal{X}$, the aim is to predict the output $\mathbf{y}$ that belongs to a structured space $\mathcal{Y}(\mathbf{x})$. Borrowing the notations from section 3.1, a structured SVM, parameterized by $\mathbf{w}$, provides a linear prediction rule as follows: $h_{\mathbf{w}}(\mathbf{x}) = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} \left(\mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y})\right)$. Given a set of labelled samples $\mathcal{D} =$

$\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$, the parameter vector $\mathbf{w}$ is learnt by solving the following convex optimization problem:

$$\min_{\mathbf{w},\xi} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \xi_i, \quad \text{s.t. } \mathbf{w}^\top \Psi_i(\mathbf{y}) \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i, \forall i \in [n], \forall \mathbf{y} \in \mathcal{Y}_i \quad (11)$$

The key differences from the multi-class SVM formulation are that here we can have any general joint feature map $\Phi(\mathbf{x}, \mathbf{y})$ and loss function $\Delta(\mathbf{y}_i, \mathbf{y})$ designed to effectively model the structure of the output space. The Lagrangian dual of problem (11) is given by:

$$\min_{\boldsymbol{\alpha}\geq 0} T(\boldsymbol{\alpha}) = -\mathbf{b}^\top\boldsymbol{\alpha} + \frac{\lambda}{2}\boldsymbol{\alpha}^\top A^\top A\boldsymbol{\alpha}, \quad \text{s.t. } \sum_{\mathbf{y}\in\mathcal{Y}_i} \boldsymbol{\alpha}_{iy} = 1, \forall i \in [n]. \quad (12)$$

Here the dual variable vector $\boldsymbol{\alpha}$ is of size $m = \sum_{i=1}^{n} |\mathcal{Y}_i|$; $\mathbf{b}$ and A have the same definition as in section 3.1.

In general the size of output space can be exponential in the number of output variables. This would result in exponentially large number of primal constraints and dual variables, which can be hard to deal with. However, these problems can be overcome by making clever use of the structure of the output space. The key observation behind our effective partial linearization based optimization algorithm is that we can efficiently compute the marginals of the output variables. Now, when the output space $\mathcal{Y}_i$ has a low tree-width graph structure, it is possible to efficiently compute the exact marginals of the output variables, by solving the expectation-oracle problem. This can be done using a message passing algorithm over a junction tree corresponding to the underlying graph of the output space [22]. In such a setting, our algorithm can be used for efficient optimization of the SSVM learning problem for low tree-width models. We discuss the partial-linearization algorithm for learning low tree-width SSVM models in detail in the supplementary material. We demonstrate the applicability of such an approach on the task of handwritten word recognition.

5 Experiments

We now demonstrate the efficacy of our algorithm on the challenging multi-class classification tasks of action classification, object recognition and gesture recognition. We also present some preliminary results for tree-structured models on the task of handwritten word recognition.

5.1 Datasets and Tasks

Action Classification

Dataset. We use the PASCAL VOC 2011 [6] action classification dataset for our experiments. This dataset consists of 4846 bounding boxes of persons, each of which is labeled using one of ten action classes. It includes 3347 'trainval' person bounding boxes for which the ground-truth action classes are known.

Modelling and Features. We train a multi-class SVM as an action classifier using 2800 labelled bounding boxes from the 'trainval' set. We use the standard poselet [15] activation features as sample feature for each person bounding box. The feature vector consists of 2400 action poselet activations and 4 object detection scores. We refer the reader to [15] for details regarding the feature vector.

Object Recognition on CIFAR-10 dataset

Dataset. We use the CIFAR-10 dataset [13] for this set of experiments. It consists of a total of 60,000 images of 10 different object classes with 6,000 images per class. The dataset is divided into a 'trainval' set of 50,000 images and a 'test' set of 10,000 images.

Modelling and Features. We train a multi-class SVM for object recognition on the trainval set. To represent each image, we use a feature representation that is extracted from a trained Convolutional Neural Network. Specifically, we pass the resized image as input to the VGG-NET [20] network and use the activation vector of its penultimate layer as the feature vector. The length of the resulting feature vector is 4096.

Object Recognition on PASCAL VOC dataset

Dataset. We use the PASCAL VOC 2007 [5] object detection dataset, which consists of a total of 9963 images of which 5011 images are in the 'trainval' set. All the images are labelled to indicate the presence or absence of the instances of 20 different object categories. Each image can have multiple instances of an object and we are provided with tight bounding boxes around each of them.

Modelling and Features. We train a multi-class SVM for object recognition on 12,608 object bounding boxes extracted from the trainval set. For each object bounding box, we use a feature representation extracted from a trained Convolutional Neural Network (CNN). Specifically, we pass the bounding box as input to the CNN and use the activation vector of the penultimate layer of the CNN as the feature vector. Inspired by the work of Girshick *et al.* [9], we use the CNN that is trained on the ImageNet dataset [3], by rescaling each window to a fixed size of 224×224. The length of the resulting feature vector is 4096.

Gesture Recognition

Dataset. We use the MSRC-12 data set [7] which contains 594 sequences of motion capture data obtained using a Kinect sensor. Each sequence corresponds to a person repeatedly performing one out of the 12 gestures represented in the dataset. For each frame of the sequence, we are given the 3D world position of 20 human body joints. In addition to the sequence level gesture annotations, we are also provided with frame level annotations which we ignore in our experiments.

Modelling and Features. We treat each sequence as a single sample and train a multi-class latent-SVM for sequence level gesture recognition. The exact location of the gesture in a sequence is held by a latent variable. We represent a sequence $\mathbf{x}$ using a feature vector $\phi(\mathbf{x}, h)$ which is extracted from the frame in the sequence denoted by the latent variable h. We extract the same 130 dimensional feature vector from a frame as used in [7].

Handwritten Word Recognition

Dataset. We use the OCR dataset [21] for our experiments. The dataset consists of 6251 images of handwritten words. We use 626 images for training and the rest for testing. Each word image is already segmented into individual characters. Each character can be of one of the 26 classes: $\{a, ..., z\}$.

Modelling and Features. The dataset provides the handwritten-word images in binary format. Each segmented character image in the dataset is of size 16×8 pixels. We use binary pixel values of the character images to construct a 128 dimensional feature vector for each character. We use an indicator basis function to represent the correlation between adjacent characters. We also use indicator basis functions to represent location independent bias for each of the characters and additional bias for the first and the last characters of any word. This makes the overall size of the feature vector equal to $(128 \times 26 + 26 \times 26 + 26 + 26 \times 2) = 4082$. Note that the underlying graph has a 'chain' structure, which enables the computation of exact marginals via sum-product belief propagation [22].

5.2 Methods

For all the tasks, we compare the runtime of our block-coordinate partial linearization (BCPL) approach to those of two baseline algorithms for solving the multi-class SVM or the SSVM optimization problem, namely the block-coordinate Frank-Wolfe algorithm [14] (BCFW) and the online exponentiated gradient (OEG) algorithm [1]. We ran each of the algorithms for 3 different values $(0.1, 0.01, 0.001)$ of the regularization parameter λ. For most practical setups λ is chosen to be very low since large datasets avoid the problem of high generalization error via overfitting. In all the experiments, we used a fixed temperature of $\tau = 0.01$ for our algorithm. For OEG, we repeated the experiments for 8 different values $(100, 10, 1, 0.1, 0.01, 0.001, 0.0001, 10^{-5}, 10^{-10})$ of the temperature parameter τ and report he results for the best performing value. We initialize all the optimization algorithms in a manner which ensures that the weight parameters are almost equal to 0. In each iteration of training, we sample without repetition from the dataset. For the BCPL and OEG algorithms, in order to avoid getting stuck on a facet of the domain polytope, we truncate the step size γ at each iteration to $1 - \epsilon$. Where, $\epsilon = 2.2204 \times 10^{-16}$ is the machine epsilon.

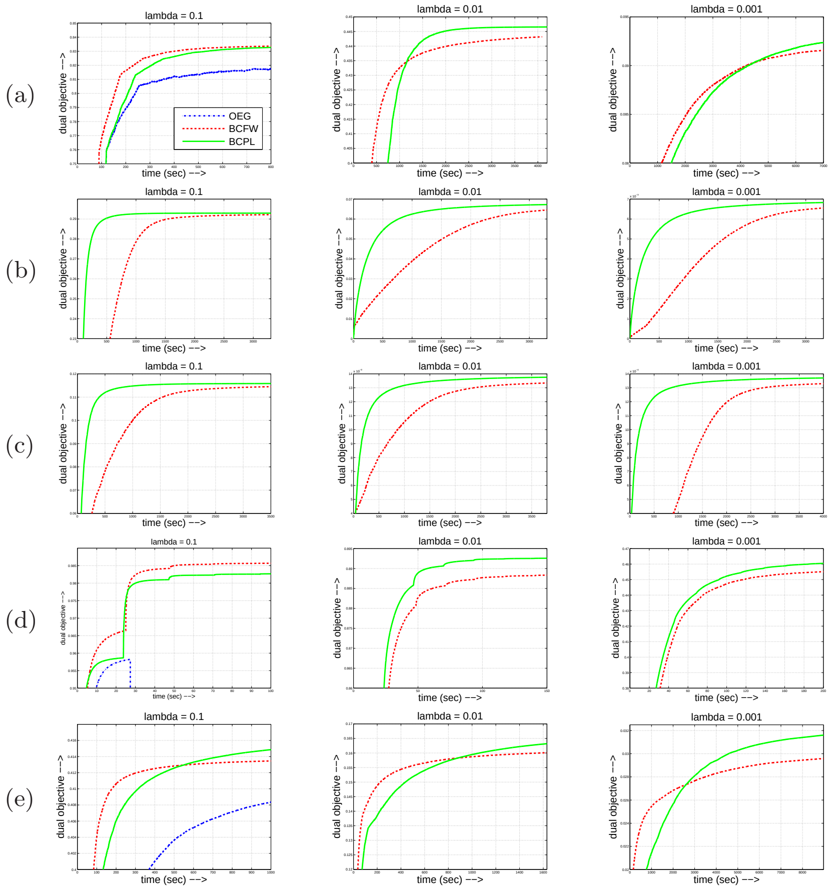

Fig. 1: *Comparison of different optimization algorithms in terms of change in the dual objective (negative of the objective of problem (4)) with respect to training time. The results correspond to (a) Action classification (b) Object recognition on* CIFAR-*10 (c) Object recognition on* PASCAL VOC *(d) Gesture recognition (e) Hand written word recognition. The figures are zoomed-in along the vertical axis to highlight the differences between the top most competing methods. Note that for* $\lambda = 0.01$ *and* $\lambda = 0.001$*, the exponentiated gradient algorithm performs significantly worse than the other two methods, and is therefore not visible in the plots. This figure is best viewed in colour.*

5.3 Results

We report the performance of the different methods in terms of the increase in the dual MC-SVM or the SSVM objective function with respect to training time. Figure 1 provides the detailed plots for the experiments for different values of λ. As can be observed from the plots, in most cases, our BCPL algorithm converges faster than BCFW and OEG. It should be noted that the relative difference between the rate of convergence of the two algorithms may seem comparatively small.

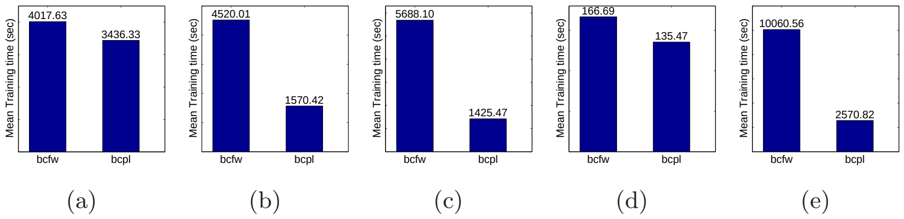

Fig. 2: *Comparison of Block-coordinate Frank-Wolfe (*BCFW*) and Block-coordinate Partial linearization (*BCPL*) in terms of the mean training time. The results correspond to (a) Action classification (b) Object recognition on* CIFAR-*10 (c) Object recognition on* PASCAL VOC *(d) Gesture recognition (e) Hand written word recognition.*

However, due to the low absolute rate of convergence of both the algorithms in the later stages, this small gap leads to significant saving in terms of iterations and time for our algorithm. The OEG algorithm performs consistently worse than the other 2 algorithms for these set of experiments. For all the tasks, we also report the mean time taken for training by our method and the Frank-Wolfe algorithm. For each task, the training time is averaged over all values of λ. Figure 2 shows that our approach consistently does better than the Frank-Wolfe algorithm. Note that since we solve a convex optimization problem, all the methods are guaranteed to converge to the same or very similar solutions. Hence, we have focused on only a comparison of the run time in the paper.

6 Discussion

We proposed a partial linearization based approach for optimizing multi-class SVM, which naturally generalizes the Frank-Wolfe and the exponentiated gradient algorithms. Our method introduces the key temperature hyperparameter for which we keep a fixed value through out the optimization. This leaves scope for exploring ideas for varying the temperature across iterations for faster convergence. In this work, we discussed our approach only in context of multi-class classification models and structured SVM models that have a tree structure. However, the efficacy of our approach in the context of loopy graphs that require approximate computation of the expectation oracle is still unknown. Another interesting direction for future research would be to explore the applicability of our approach for variations of the SVM optimization problem (such as those that use soft constraints), or for other learning frameworks such as convolutional neural networks.

Acknowledgements

Pritish is supported by the TCS Research Scholar Program.

References

1. Collins, M., Globerson, A., Koo, T., Carreras, X., Bartlett, P.L.: Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. The Journal of Machine Learning Research, 2008
2. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. The Journal of Machine Learning Research, 2002
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009)
4. Engel, J.: Polytomous logistic regression. Statistica Neerlandica (1988)
5. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html
6. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The PASCAL visual object classes (VOC) challenge. IJCV (2010)
7. Fothergill, S., Mentis, H., Kohli, P., Nowozin, S.: Instructing people for training gestural interactive systems. In: SIGCHI Conference on Human Factors in Computing Systems, 2012
8. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval research logistics quarterly (1956)
9. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
10. Jaggi, M.: Revisiting frank-wolfe: Projection-free sparse convex optimization. In: ICML (2013)
11. Joachims, T., Finley, T., Yu, C.J.: Cutting-plane training of structural svms. Machine Learning, Springer, 2009
12. Kivinen, J., Warmuth, M.: Relative loss bounds for multidimensional regression problems. JMLR, 2001
13. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
14. Lacoste-Julien, S., Jaggi, M., Schmidt, M., Pletscher, P.: Block-coordinate frank-wolfe for structural svms. In: ICML (2012)
15. Maji, S., Bourdev, L., Malik, J.: Action recognition from a distributed representation of pose and appearance. In: CVPR (2011)
16. Malouf, R.: A comparison of algorithms for maximum entropy parameter estimation. In: Conference on Natural language learning, 2002
17. Patriksson, M.: Partial linearization methods in nonlinear programming. Journal of Optimization Theory and Applications, Springer, 1993
18. Quinlan, J.: Classification and regression trees. Programs for Machine Learning (2011)
19. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: Primal estimated sub-gradient solver for svm. Mathematical programming, Springer, 2011
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
21. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: NIPS (2004)
22. Wainwright, M.J., Jordan, M.: Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning (2008)
23. Zhang, X., Saha, A., Vishwanathan, S.: Accelerated training of max-margin markov networks with kernels. Theoretical Computer Science, Elsevier, 2014