A Smoother Way to Train Structured Prediction Models

Nabil Madali , Hugo Mallet

1 Introduction

When dealing with annotation algorithms, annotating the total sample can be a harsh task and can require significant human effort. Thus, in contrast to the full annotation, applying a weak annotation seems to be easier. For example, when solving the problem of semantic analysis sentences, a weak annotation of the training sample can be represented by marking speech types. Moreover, incomplete annotation can be used in the problem of categorizing documents, leading to part of the categories for each of the documents being omitted. In a number of analysis tasks in video sequences, annotation can only be given for key frames. In this paper, the target applications are the tasks of semantic image segmentation. Using weak annotation types for this task is determined by practical expediency. For instance, in the dataset PASCAL VOC 2012 only 2913 of 11540 (25%) images are fully marked, for the rest, only the dense framework of certain categories of objects is known. Moreover, it is often beneficial to use a variety of types of weak annotations because they better characterize various semantic categories.

The literature describes methods that use weak annotations to teach semantic segmentation, but most of them only use image labels as weak annotations. Hence, [30] uses probabilistic graphical model over a set of images to disseminate information about label between images. In this paper, we present a method for teaching semantic segmentation by a mixture of strongly and weakly annotated images. This method allows us to consider different types of weak annotation, even within the same image.

The proposed method teaches the model, pre-telling a full set of categories, having only a weakly annotated teaching sample. This work is based on recent studies using the structural method reference vectors with latent variables [31] for learning tasks with weak observation. In contrast to them, the proposed method uses specialized loss functions that measure consistency of annotation predicted by the algorithm with true annotation.

2 Review structure study

The problem to be solved in structured learning is to find a powerful function $f : X \to Y$, where input and output are structured objects. An object can be a sequence, list, tree (tree structure), bounding box, etc. Among them, X and Y are spatial representation of two distinct objects.
To illustrate this, we will take as example the target detection. Obviously, other interesting applications besides target detection exists. Targets can be labeled with irregular shapes, and different parts of the person can be labeled with different colors to operate the recognition of actions. In order to have a better understanding of the problem, we take a simple bounding box target detection as an example. The unified framework for structural learning can be divided into the following two parts: inference, and training.

Inference : Structured prediction aims to search for score functions ϕ parameterized by $w \in R^d$ that model the compatibility of input $x \in \mathcal{X}$ and output $y \in Y$ as $\phi(x, y, w)$ through a graphical model. Given a score function $\phi(., ., w)$, predictions are made using an inference procedure which, when given an input x, produces the best output:

$$\tilde{y} = \arg\max_{y \in Y} w \cdot \phi(x, y)$$

Given a picture x, we list all possible labeling box y, and for each pair of (x, y), we use $w \cdot \phi$ to calculate a pair of (x, y) with the highest score, and we take the corresponding y as the output. Some commonly used algorithms for inference in target detection are Branch and Bound algorithm [11] and Selective Search [28]. Others used in sentence annotation are Viterbi Algorithm [9] , or genetic algorithm [16] to search. But in our case we always assume that this y has already been obtained.

Training : Given training data :

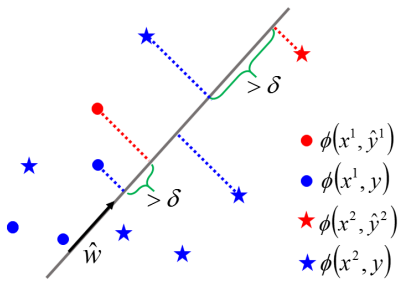
$$\left\{ \left(x^1, \hat{y}^1\right), \left(x^2, \hat{y}^2\right) \ldots, \left(x^{\mathrm{N}}, \hat{y}^{\mathrm{N}}\right) \right\}$$

We define F(x,y) as a linear combination of features consisting of a series of x, y as follows :

$$F(x, y) = w \cdot \phi(x, y)$$

Where w is a parameter learned by using the training data, and ϕ is an artificially defined rule.

After training, we wish the model to achieve :

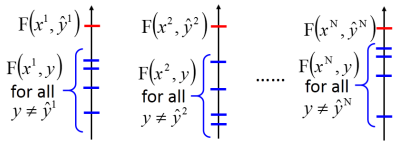


Red dots represents the correct feature points, whereas blue represents the wrong ones. Separability can be understood as the need to find a weight vector whose function is to be equal to $\phi(x, y)$. Doing an inner product can make the correct point value greater by a δ than the wrong point.

Formally, a weight vector $\hat{w}$ exists such that:

$$\hat{w} \cdot \phi\left(x^1, \hat{y}^1\right) \geq \hat{w} \cdot \phi\left(x^1, y\right) + \delta$$
$$\hat{w} \cdot \phi\left(x^2, \hat{y}^2\right) \geq \hat{w} \cdot \phi\left(x^2, y\right) + \delta$$

For the input training data, only the y corresponding to the given input data will get the largest label value.



2.1 Structure perceptron

The Structure perceptron algorithm [23] solves the classical problem of online learning of halfspaces.

There are N examples $(x_i, \hat{y}^i)$,where $x_i \in R^n$ are feature vectors and $\hat{y}^i$ are labels.

An online algorithm is given x_i in an order, then asked to predict $\hat{y}^i$ and finally the correct label is revealed. The goal is to minimize the number of classification mistakes. The perceptron algorithm starts with an initial guess $w_1 = 0$ for the halfspace, and does the following on :

For each pair of the training example $(x_i, \hat{y}^i)$:

1. Find the label $\tilde{y}^n$ maximizing $w \cdot \phi(x^n, y)$:
$$\tilde{y}^n = \arg\max_{y \in Y} w \cdot \phi(x^n, y)$$
2. If $\tilde{y}^n \neq \hat{y}^n$, update
$$w_{i+1} = w_i + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)$$
else $w_{i+1} = w_i$.

In the case of separability, in order to get $\hat{w}$, we only need to update at most $(R/\delta)^2$ times. Where δ is the interval so that the points that are misclassified and the correct points can be separated linearly, and R is the maximum distance between ϕ(x, y) and ϕ (x, y'); the largest distance between features.

3 Indivisible situation

When the data is non-separable, some weights are still better than others. Thus, we need to define a cost function C to evaluate how bad the effect of w is, and then to choose w that minimizes the cost function C :

$$C^n = max_y[w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n)$$

Where, $C = \sum_n C^n$ and C^n represent the difference between the function value of the group with the best function value in the $n-th$ data minus the correct function value. We know that the best case should be 0. At this time, the correct set of functions corresponds to the largest value. For such a loss function, we use stochastic gradient descent to update the weights. Although there is a *max*, the gradient descent method can still be used. This specific method is as follows.

For t=1 to T :

- randomly pick a training example $(x_i, \hat{y}^i)$
- Find the label $\tilde{y}^n$ maximizing $w \cdot \phi(x^n, y)$
$$\tilde{y}^n = \arg\max_{y \in Y} w \cdot \phi(x^n, y)$$
- update
$$\nabla C^n = \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)$$
$$w = w - \eta \nabla C^n$$

Let suppose now that we use the stochastic gradient descent method to update the data a total of T times and randomly select a training data. At this time, we will find the current maximum y according to x, and use the obtained y to determine which area is currently located. Thus, we write the expression for this area, find the corresponding gradient, and update the gradient.

3.1 Consider the wrong level

In the previous loss function, we suppose that the different possible errors are the same, that is, all errors are treated equally, which is obviously a not reasonable approach.

Lets define the error function $\Delta(\hat{y}, y)$ that define the gap between $\hat{y}$ and y :

$$\Delta(\hat{y}, y) = 1 - \frac{A \cap \hat{(y)}}{A \cup \hat{(y)}}$$

We can observe that $\Delta(\hat{y}, y)$ takes a value between 0 and 1, so that when the two do not overlap, this value takes the maximum value of 1, and when the two coincide, the minimum value is 0.

We modify the loss function to the following form :

$$C^n = max_y[\Delta(\hat{y}, y) + w \cdot \phi(x^n, y)] - w \cdot \phi(x^n, \hat{y}^n)$$

In the case where the box does not match the real function value, we add Δ which will will cause the loss function to be very large. Hence, the loss is stretched larger than the original. The improved loss function can also be trained using the gradient descent method.

This specific method is as follows.

- randomly pick a training example $(x_i, \hat{y}^i)$
- Find the label $\bar{y}^n$ maximizing $\Delta(\hat{y}, y) + w\phi(x^n, y)$

$$\bar{y}^n = \arg\max_{y\in(Y} \Delta(\hat{y}, y) + w \cdot \phi(x^n, y)$$

- update

$$\nabla C^n = \phi(x^n, \hat{y}^n) - \phi(x^n, \bar{y}^n)$$
$$w = w - \eta \nabla C^n$$

However, the resulting problem is piece wise constant and hard to optimize. Instead, Taskar et al. [26], Tsochantaridis et al. [27] propose to minimize a majorizing surrogate of the task loss, called the structural hinge loss defined on an input-output pair $(x^{(i)}, y^{(i)})$ as

$$f^{(i)}(w) = max_{y\in\mathcal{Y}}\{\phi(x_i, y', w) + \ell(y_i, y)\} - \phi(x_i, y_i, w)$$
$$max_{y\in\mathcal{Y}} = \phi^{(i)}(y, w)$$

This approach, known as max-margin structured prediction, builds upon binary and multi-class support vector machines [5].

When considering a fixed input-output pair $(x^{(i)}, y^{(i)})$, we drop the index with respect to the sample i and consider the structural hinge loss as

$$f(w) = max_{y\in\mathcal{Y}} = \phi^{(i)}(y, w)$$

When the map $w \to \phi(y, w)$ is affine, the structural hinge loss f and the objective function F are both convex - we refer to this case as the structural support vector machine.When $w \to \phi(y, w)$ is a non linear but smooth map, then the structural hinge loss f and the objective F are non-convex.

4 Structural Support Vector Machine

For most classifiers in machine learning, each sample consists of $x \in \mathcal{X}$ Output $y \in \mathcal{Y}$ Composition, and our task is to find a mapping for input to output, ie $\mathcal{X} \to \mathcal{Y}$. In the traditional method, the input space usually belongs to multi-dimensional space. $x \in R^d$ and the output is equal to some finite-size discrete set $\{1, 2, 3, \ldots, N\}$

Using a number to represent the output is applicable in most cases. However, there may be some limitations in the application in particular fields such as vision and natural language processing, because the output may be a more complete representation, so structured learning came into being.

In our case, the output y is allowed to take a multi-dimensional structure similar to the input x, so has to propose a model that is more in line with the actual situation. In this section we will take as example the structured Support vector machine.

Suppose our model parameters are *w*, the input and output are $x \in \mathcal{X}$ with $y \in \mathcal{Y}$. Our goal is to learn a discriminant function f for prediction:

$$\hat{y} = f(x; w) = \arg\max_{y\in(Y} F(x, y, w)$$

We assume that F has a linear relationship with some comprehensive feature of input and output:

$$F(x, y, w) =< w, \Psi(x, y) >$$

Specific features selected $\Psi(x, y)$ depends on the actual application scenario.

In addition, we need to design a loss function that evaluates the prediction results $\Delta : \mathcal{Y} \times \mathcal{Y} \to Y$ And, if the predicted value is closer to the real value, the loss should be smaller, and vice versa, the larger the loss, so the total risk can be defined as:

$$R_P^{\Delta} = \int_{\mathcal{X}\times\mathcal{Y}} \Delta(y, f(x))dP(x, y)$$

Here P is the probability distribution of the data, which is generally unknown to us, so we need to use the training sample data to calculate the emergency risk R_P^{Δ} instead.

Ideally, one or more parameters *w* should be found such that $S = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y} : i = 1, 2, \ldots, n$, the empirical risk calculated above is 0, and its conditions are:

$$\forall i : \quad max_{y\in\mathcal{X}\setminus y_i}\{< w, \Psi(x_i, y) >\} < w, \Psi(x_i, y_i) >$$

The above formula consists of n linear inequalities constraints containing the max function, which can be expanded into $n \parallel \mathcal{Y} \parallel -n$: An ordinary formula:

$$\forall i, \forall y \in \mathcal{Y} \setminus y_i : \quad < w, \delta\Psi_i(y) >> 0$$

Here $\delta\Psi_i(y) =< w, \Psi(x_i, y_i) > - < w, \Psi(x_i, y) >$

For the above multiple constraints, there may be multiple solutions. Here you can use the maximum-margin principle of the classic SVM. $\parallel w \parallel \leq 1$. In this case, the distance between the hyper-plane and the nearest data point is maximized, so the original problem can be transformed into an optimization problem similar to the classic SVM:

$$SVM_0 : min_w \frac{1}{2} \parallel w \parallel^2$$
$$\forall i, \forall y \in \mathcal{Y} \setminus y_i : \quad < w, \delta\Psi_i(y) >\geq 1$$

Then adding the soft border allows us to get :

$$SVM_1 : min_w \frac{1}{2} \parallel w \parallel^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \quad s.t \forall i \xi \geq 0$$

$$\forall i, \forall y \in \mathcal{Y} \setminus y_i : \quad < w, \delta\Psi_i(y) >\geq 1 - \xi_i$$

ξ_i added here are margin violations. The article also mentions that it can be replaced by its second norm sum to get another SVM deformation. Observing the mathematical optimization problem extracted from Structural SVM, it can be seen that its objective function is actually similar to the classic SVM. Turn into $\parallel w \parallel^2$ to implement the maximum-margin idea, and then attach a soft boundary regularizer to prevent the existence of anomalous points from deviating too far from the hyperplane. The difference resides mainly in the underground constraint. Classic SVM limits the distribution of sample points on both sides of the hyperplane and maintains a sufficient distance, while the Structural SVM is constructed in advance by limiting x,y. The size of the loss function is used as a constraint. The simple and rude punishment

of the former model is avoided here (such as misclassification penalty 1 and sub-pair penalty 0). Different degrees of punishment are applied to samples with different classification errors, which obviously will have an impact on the accuracy of the classifier.

4.1 Cutting-plane training of structural SVMs

The formulation of the SSVM objective with linear constraints in problem 4 is a convex problem. However, the number of linear constraints can be intractably large, so the problem is typically treated using the cutting plane method described in Tsochantaridis, et al [27] instead. This cutting plane method exploits the sparsity and structure of the problem to reduce the number of constraints needed to be considered.

A low-dimensional example of the problem we want to solve :

For $w, \epsilon^1, ..., \epsilon^N$ minimizing C

$$C = \frac{1}{2} \parallel w \parallel^2 + \lambda \sum_n \epsilon^n$$

For $\forall y \neq \hat{y}^n$:

$$w \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, y)) \geq \Delta(\hat{(y)}, y) - \epsilon^n, \epsilon^n \geq 0$$

Suppose we do not consider the restricted part, but only consider the minimized part, and also assume that *w* has only one dimension. The surface of the loss function is shown in the figure bellow

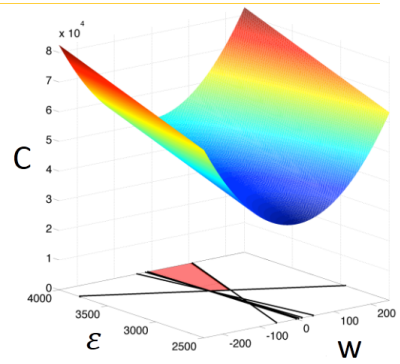


Figure 1: Depiction of the learning QP and approximated feasibility region. http://abnerguzman.com/publications/gkb$_d$iscml12.*pdf*

We can observe that many constraints restrict the range of our independent variables to a red quadrilateral. However, in the actual problem, the quantity of constraint price adjustment is too much. At this time, the cut plane method is needed to solve it.

In the parameter space composed of *w* and ϵ^i , the color represents the value of C. In the case of no limitation, the cyan point corresponds to the minimum value. In the case of limitation, only the embedded polygon area meets the constraints

$$C = \frac{1}{2}\|w\|^2 + \lambda \sum_{n=1}^{N} \varepsilon^n$$

Thus, we need to find the minimum value in this area .

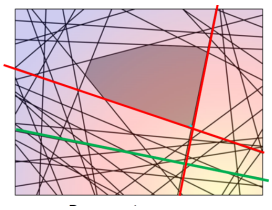


Figure 2: Illustrative example of SSVM learning. SSVM problem repeatedly finds the next most violated constraint until set of constraints is a good approximation.

Although there are many constraints, most of them are redundant and do not affect the solution of the problem : only the red one is useful, while the green one is obviously useless.

It was originally exhaustive to consider $y \neq \hat{y}^n$. However, we now need to remove those ineffective lines and retain only useful lines. These influential line sets can be defined as *Working Set*, which is represented by A^n .

Here iterative method is used to update the parameters. Initialize $A^1 ... A^N$

- Solve the QP problem :
  For $w, \epsilon^1, ..., \epsilon^N$ minimizing C

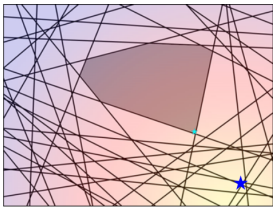
  $$C = \frac{1}{2} \parallel w \parallel^2 + \lambda \sum_n \epsilon^n$$

  For $\forall y \neq \hat{y}^n$:

  $$w \cdot (\phi(x^n, \hat{y}^n) - \phi(x^n, y)) \geq \Delta(\hat{(y)}, y) - \epsilon^n, \epsilon^n \geq 0$$

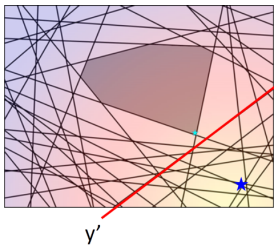
- obtain solution *w*
- add elements into $A^1 ... A^N$

Originally, to solve the QP problem, we need to consider all possible labels y, but if given a *Working set*, we only need to consider the label y in this set. Then it is relatively simple to solve the *QP* problem. Lets assume that we found the corresponding *w* according to the *Working Set*, and then recheck with *w* in order to find new members to join the *Working set*. Thus the *Working Set* will change. Then the *QP* problem will be solved according to the new *Working set*. In this case, we will get a new *w* again. New members can be added to the *Working set*, and iterate continuously until *w* no longer changes.

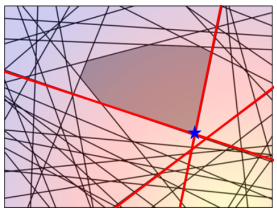


Suppose that the initial value of A^n is the empty set. Hence, there are no constraints, and the result of solving *QP* is the corresponding blue point, but there are numerous lines that cannot meet the conditions.

Find out which one is the highly violated constraints. Then we put $A^n = A^n \cup \{y'\}$.



According to the only member y' in the newly obtained *Working set*, find a new minimum value. Based on this set, we can solve the *QP* problem, and then obtain a new w. Although the new w and minimum value are obtained, there are still constraints that do not meet the conditions and need to be considered. Thus, the idea is to keep adding the most violated constraints to the working set and solve the *QP* problem again until all the highly violated constraints were added to the *Working set*. Finally there are three lines in the *Working set*. According to these lines, the points in the solution interval can be determined, giving us the problem' solution.



Optimization Problem 1 (OP1). The regularized hinge-loss SSVM learning problem can be formulated as a *QP* with exponentially many constraints. We work with the margin-rescaling variant of the 1-slack formulation of Joachimset al. [12]:

$$min_{w,\epsilon\geq 0}\frac{1}{2}w^T w + C\xi$$

$$\frac{1}{n}\sum_{i=1}^{n}[\Psi(x_i,y_i)-\Psi(x_i,\bar{y}_i)] \geq \frac{1}{n}\sum_{i=1}^{n} -\xi\ell(y_i,\bar{y}_i) \quad \forall \bar{Y} \in \mathcal{Y}^n$$

Note that 1-slack SSVMs involves $\mid \mathcal{Y} \mid^n$ constraints, one for each possible n-tuple of labels $\bar{Y} = (\bar{y}_1, ..., \bar{Y}_n)\mathcal{Y}^n \in$, but there is a single slack variable ξ shared across all constraints

Cutting-Plane Training of SSVMs. Algorithm 1 provides a cutting-plane approach to solving OP1. At every iteration, the algorithm computes the solution over the current *Working-set W* (Line 4) and then finds the most violated constraint (Lines 5-7) to add to *W* (Line 8). The algorithm stops when the most violated constraint is violated less than a desired precision ξ (Line 9).

Algorithm 1 Training Structural SVMs

1: Input: $S = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}, C, \epsilon$
2: $\mathcal{W} \leftarrow \emptyset$
3: **repeat**
4: $(\mathbf{w}, \xi) \leftarrow \text{argmin}_{\mathbf{w},\xi\geq 0}\ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$
$s.t.\quad \frac{1}{n}\mathbf{w}^T\sum_{i=1}^{n}\left[\Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\bar{\mathbf{y}}_i)\right] \geq \frac{1}{n}\sum_{i=1}^{n}\ell(\mathbf{y}_i,\bar{\mathbf{y}}_i) - \xi_i$
5: **for** $i = 1, \ldots, n$ **do**
6: $\hat{\mathbf{y}}_i \leftarrow \text{argmax}_{\mathbf{y}}\left\{\ell(\mathbf{y}_i,\mathbf{y}) + \mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y})\right\}$
7: **end for**
8: $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_n)\}$
9: **until** $\frac{1}{n}\sum_{i=1}^{n}\ell(\mathbf{y}_i,\hat{\mathbf{y}}_i) - \frac{1}{n}\mathbf{w}^T\sum_{i=1}^{n}\left[\Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\hat{\mathbf{y}}_i)\right] \leq \xi + \epsilon$
10: **return** $(\mathbf{w}, \xi)$

Joachimset al. [12] showed that Algorithm 1 converges in polynomial time, in fact the number of iterations is independent of the size of the training-set.

Batch nonsmooth optimization algorithms such as cutting plane methods are appropriate for problems with small or moderate sample sizes [27],[12]. Stochastic nonsmooth optimization algorithms such as stochastic subgradient methods can tackle problems with large sample sizes [22],[25]. However both families of methods achieve the typical worst-case complexity bounds of nonsmooth optimization algorithms and cannot easily leverage a possible hidden smoothness of the objective.

5 Smoothing Inference for Structured Prediction

[21] introduced a general framework that allows us to bring the power of accelerated incremental optimization algorithms to the realm of structured prediction problems,to achieve near-optimal convergence rates [**?**]. Accelerated incremental optimization algorithms demonstrate stable and fast convergence behavior on a wide range of problems.

We focus on the problem of training a structural support vector machine (SSVM). The same ideas can be applied to other structured prediction models to obtain faster training algorithms.

Consider the optimization problem arising when training structural support vector machines::

$$min_{w\in R^d}F(w) := \frac{1}{n}\sum_{i=1}^{n} f^{(i)}(w) + \frac{\lambda}{2}\|w\|_2^2 \qquad (1)$$

where each $f^{(i)}$ is the structural hinge loss.

$$f^{(i)}(w) = max_{y\in\mathcal{Y}}\{\phi(x_i, y', w) + \ell(y_i, y)\} - \phi(x_i, y_i, w) \quad (2)$$

Where, the task loss ℓ is assumed to possess appropriate structure so that the maximization inside 2, known as loss augmented inference, is no harder than the inference problem in 1.

The learning problem 1 is the minimization of the structural hinge losses on the training data $(x_i, y_i)_{i=1}^n$ with a regularization penalty.

Smoothing for inference was used to speed up iterative algorithms for continuous relaxations, [21] define smooth inference oracles in the context of black-box first-order optimization and establish worst-case complexity bounds for incremental optimization algorithms making calls to these oracles. To smooth the structural hinge loss, [21] proposed to decompose the structural hinge loss as composition of as linear mapping

$$g : \begin{cases} R^d \to R^{|\mathcal{Y}|} \\ w \to (\psi(y', w))_{y'\in\mathcal{Y}} = Aw + b \end{cases}$$

and a max function with

$$h : \begin{cases} R^{|\mathcal{Y}|} \to R \\ z \to max_{i\in|\mathcal{Y}|} z_i \end{cases}$$

The structural hinge loss can now be expressed as

$$f = h \circ g$$

Then it can be easily smoothed through its dual formulation to obtain a smooth surrogate of 2.

A convex, non-smooth function h can be smoothed by taking its infimal convolution with a smooth func-tion [2]. We now recall its dual representation, which Nesterov [18] first used to relate the amount of smoothing to optimal complexity bounds.

Lemma 5.1 *(Nesterov [17]) if w is 1-strongly convex with respect to $\|.\|_a$, then $f_{\mu w}$ is ($\|A\|_{2,a}^2 / u$)-smooth ,and approximates f for any w as*

$$\mu \quad min_{u\in\Delta^{m-1}} w(u) \leq f(w) - f_{\mu w}(w) \leq \mu \quad max_{u\in\Delta^{m-1}} w(u)$$

where,Δ^{m-1} is the probability simplex in $R^{|\mathcal{Y}|}$ and μ is the smoothing coefficient.

The smoothing $h_{\mu w}$ of h by a strongly convex function w with smoothing coefficient $\mu > 0$ is defined as

$$h_{\mu w}(z) = max_{u\in\Delta^{m-1}} . z^T u - \mu w(u)$$

The smooth approximation by simply smoothing the non-smooth max function of the structural hinge loss is then given by

$$f_{\mu w} = h_{\mu w} \circ g$$

When g is smooth and Lipschitz continuous. As choices for *w* we focus on the squared euclidean norm denoted as :

$$\ell_2^2 := \frac{1}{2}(\|u\|_2^2 - 1)$$

The gradient $\nabla h_{\mu\ell_2^2}(z)$ is given by the projection of z/μ onto the simplex, which selects a small number, denoted $K_{z/\mu}$, of its largest coordinates . We shall approximate this projection by fixing independently of z/μ and defining

$$h_{\mu,K}(z) = max_{u\in\Delta^{K-1}}\{z_{[k]}^T u - \mu\ell_2^2\}$$

As an approximation of $h_{\mu,\ell_2^2}(z)$, where $z_{_{[k]}\in R^K}$ denote the K largest components of z. If $K_{z/u} < K$ this approximation is exact .

The resulting surrogate is denoted $f_{\mu K} = h_{\mu K} \circ g$,the gradient of the smooth surrogate $f_{\mu K}$ can be written using the chain rule.

In particular, when g is an affine map

$$g(w) = Aw + b$$

If follows that $f_{\mu K}$ is ($\|A\|_{\beta,\alpha}^2 / u$)-smooth with respect to $\|.\|_\beta$ for $\mu_1 \geq \mu_2 \geq 0$, we have

$$(\mu_1 - \mu_2) \quad min_{u\in\Delta^{m-1}} w(u) \leq f_{\mu_1 w}(w) - f_{\mu_2 w}(w)$$

$$\leq \mu \quad max_{u\in\Delta^{m-1}} w(u)$$

[21] Define a smooth inference oracle as a first-order oracle for a smooth counter part of the structural hinge loss. Standard analysis of first-order methods assumes availability of exact first-order information. Namely, the oracle must provide at each given function f and point $w \in dom(f)$, the exact values of the function $f(w)$ and its gradient $v \in \partial f(w)$.

However, in many convex problems, including those obtained by smoothing techniques, the objective function and its gradient are computed by solving another auxiliary optimization problem. In practice, we are often only able to solve these subproblems approximately. Hence, in that context, numerical methods solving the outer problem are provided with inexact first-order information.

[21] define three variants of a smooth inference oracle:

- max-oracle return $f(w)$ and $\nabla\psi(y^*, w) \in \partial f(w)$,where $y^* \in argmax_{y\in\mathcal{Y}}\psi(y', w)$
- exp-oracle return $f_{-\mu H}(w)$ and $\nabla f_{-\mu H}(w) = E_{y'\sim p_u}[\nabla\psi(y', w)]$,where $p_u \propto exp(\psi(y', w)/\mu)$
- the top-K oracle computes the K best outputs $\{y_{(i)}\}_{i=1}^K = \mathcal{Y}_k$ satisfying

$$\psi(y_{(1)}, w) \geq ... \geq \psi(y_{(k)}, w) \geq gmax_{y\in\mathcal{Y}\setminus\mathcal{Y}_k}$$

to return $f_{\mu,K}(w)$ and $\nabla f_{\mu,K}(w)$ as surrogates for $f_{\ell_2^2,K}$ and $\nabla f_{\ell_2^2,K}$

The implementation of inference oracles depends on the structure of the output. For graphs with a tree structure or bounded tree-width, the max oracle is implemented by dynamic programming (DP) algorithms such as the popular Viterbi algorithm. The exp-oracle can be achieved by replacing the max in DP with log-sum-exp and using back-propagation at $O(1)$ times the cost of the max oracle. The top-K oracle is implemented by the top-K DP algorithm which keeps track of the K largest intermediate scores and the back-pointers at $O(K)$ times the cost of the max oracle.

Though the gradient of the composition $f_{\mu K} = h_{\mu K} \circ g$ can be written using the chain rule, its actual computation for structured prediction problems involves computing over all $m =\mid Y \mid$ of its components,which may be intractable. However, in the case of ℓ_2^2 smoothing, projections onto the simplex are sparse. The projection of z/μ onto the simplex picks out some number K_z/μ of the largest entries of z/μ- we refer to this as the sparsity of proj$\Delta f_{m-1}(z/\mu)$. This fact motivates the top-K strategy: given $\mu > 0$, fix an integer K a priori and consider as surrogates for $h_{u\ell_2^2}$ and $\nabla h_{u\ell_2^2}$ respectively

$$h_{\mu K}(z) := max_{u \in \Delta^{K-1}} \{ \langle z_{[k]}, \mu \ell_2^2 \rangle \}$$

and

$$\tilde{\nabla} h_{\mu K}(z) := \Omega(z)^T \text{proj}_{\Delta^{K-1}}(\frac{z_{[k]}}{\mu})$$

Where,$z_{[k]}$ denotes the vector composed of the K largest entries of z and $\Omega(z) : R \to \{0,1\}^{K \times m}$ defines their extraction, i.e.,

$$\Omega_K(z) = (e_{j1}^T, ..., e_{jk}^T) \in \{0,1\}^{K \times m}$$

where,$j1, ..., jk$ satisfy ,

$$z_{j1} \geq ... \geq z_{jk}$$

such that $z_{[k]} = \Omega_K(z) z$ A surrogate of the ℓ_2^2 smoothing is then given by

$$f_{\mu K} := h_{\mu K} \circ g$$

and

$$\tilde{\nabla} f_{\mu K} := \nabla g(w) \tilde{\nabla} h_{\mu K}(g(w))$$

the top-K strategy provides a computationally efficient heuristic to smooth the structural hinge loss. Though we do not have theoretical guarantees using this surrogate, experiments presented show its efficiency and its robustness to the choice of K.

6 Catalyst with smoothing

Catalyst [20] introduced a generic scheme to solve nonconvex optimization problems using gradient-based algorithms originally designed for minimizing convex functions. Even though these methods may originally require convexity to operate, the proposed approach allows one to use them without assuming any knowledge about the convexity of the objective. In general, the scheme is guaranteed to produce a stationary point with a worst-case efficiency typical of first-order methods, and when the objective turns out to be convex, it automatically accelerates in the sense of Nesterov and achieves near-optimal convergence rate [**?**] in function values.

The Catalyst approach considers at each outer iteration a regularized objective centered around the current iterate [15]. The algorithm proceeds by performing approximate proximal point steps, starting from a point z with a step-size $1/\kappa$ and allows us to computes the minimizer of

$$min_{w \in R^m} F(w) + \frac{\kappa}{2} \|w - z\|_2^2$$

While solving this subproblem might be as hard as the original problem, we only require an approximate solution returned by a given optimization method $\mathcal{M}$. The Catalyst approach is then aninexact accelerated proximal point algorithm that carefully mixes approximate proximal point steps with the extrapolation scheme of Nesterov [19] .

[21] extend the Catalyst approach to non-smooth optimization problems by performing adaptives smoothing in the outer-loop and adjusting the level of accuracy accordingly in the inner-loop. To this end we define as a smooth approximation of the objective F

$$F_{\mu w,k}(w, z) = \frac{1}{n} \sum_{i=1}^{n} f_{\mu_k,w}^{(i)}(w) + \frac{\lambda}{2} \|w\|_2^2$$

and ,

$$F_{\mu w,k}(w, z) = \frac{1}{n} \sum_{i=1}^{n} f_{\mu_k,w}^{(i)}(w) + \frac{\lambda}{2} \|w\|_2^2 + \frac{\kappa}{2} \|w - z\|_2^2$$

A smooth and regularized approximation of the objective centered around a given point $z \in R^d$. While the original Catalyst algorithm considered a fixed regularization term κ,[21] vary κ and μ along the iterations.This enables us to get adaptive smoothing strategies .

For Smoothable objective F, with h simple, smoothing function w , linearly convergent algorithm M, non-negative and non-increasing sequence of smoothing parameters $\mu_k, k \geq 1$, positive and non-decreasing sequence of regularization parameters $\kappa_k, k \geq 1$, non-negative sequence of relative target accuracies $\sigma_k, k \geq 1$ and, initial point $w_0 \in \alpha_0 \in (0,1)$, and time horizon K.

First, we initialize $z_0 = w_0$ and we repeat for K times the following process :

Using $\mathcal{M}$ with z_{k-1} as the starting point, find

$$w_k \approx argmin_{w \in R^d} F_{\mu_k,w,\kappa}(w, z_{k-1})(w, z_{k-1})$$

Where,

$$F_{\mu_k,w,\kappa}(w, z_{k-1}) = \frac{1}{n} \sum_{i=1}^{n} f_{\mu_k,w}^{(i)}(w) + \frac{\lambda}{2} \|w\|_2^2 + \frac{\kappa_k}{2} \|w - z\|_2^2$$

such that

$$F_{\mu_k,w,\kappa}(w, z_{k-1}) - min_w F_{\mu_k,w,\kappa}(w, z_{k-1}) \leq \frac{\mu_k \kappa_k}{2} \parallel w - z_{k-1} \parallel_2^2$$

Then we solve for α_k,

$$\alpha_k^2(\kappa_{k+1} + \lambda) = (1 - \alpha_k)\alpha_k^2(\kappa_k + \lambda) + \alpha_k \lambda$$

Finally, we set

$$z_k = w_k + \beta_k(w_k - w_{k-1})$$

Where,

$$\frac{\alpha_{k-1}(1-\alpha_{k-1})(\kappa_k+\lambda)}{\alpha_{k-1}^2(\kappa_k+\lambda)+\alpha_k(\kappa_{k+1}+\lambda)}$$

In view of the strong convexity of $F_{\mu_k,w,\kappa}(.,z_{k-1})$, the stopping criterion for the subproblem can be checked by looking at the gradient of $F_{\mu_k,w,\kappa}(.,z_{k-1})$. As it is smooth and strongly convex, the maximal number of iterations to satisfy the stopping criterion can also be derived.

The global complexity of the algorithm then depends on the choice of optimization method $\mathcal{M}$. [21] summarize in Table 1 the total complexity for different strategies when SVRG [13] is used as $\mathcal{M}$, resulting in an algorithm called SC-SVRG in the remainder of the paper. Note that the adaptive smoothing schemes (2, 4) do not match the rate obtained by a fixed smoothing (1, 3). A standard doubling trick can easily fix this. Yet we choose to use an adaptive smoothing scheme, easier to use and working well in practice .

Scheme	μ_k	κ_k	σ_k	$E[N]$
1	$\frac{\epsilon}{D}$	$\frac{aD}{\epsilon n} - \lambda$	$\sqrt{\frac{\lambda\epsilon n}{aD}}$	$n + \sqrt{\frac{aDn}{\lambda\epsilon}}$
2	μc^k	λ	c'	$n + \frac{a}{\lambda\epsilon}\frac{\Delta F_0 + \mu D}{\mu}$
3	$\frac{\epsilon}{D}$	$\frac{aD}{\epsilon n}$	$\frac{1}{K^2}$	$n\sqrt{\frac{\Delta F_0}{\epsilon}} + \frac{\sqrt{aDn}\Delta_0}{\epsilon}$
4	$\frac{\mu}{k}$	$\kappa_0 k$	$\frac{1}{k^2}$	$\frac{\hat{\Delta}_0}{\epsilon}(n + \frac{a}{\mu\kappa_0})$

Table 1: Summary of global complexity of SC-SVRG, i.e., Algorithm 1 with SVRG as the inner solver for various parameter settings. We show $3[N]$, the expected total number of SVRG iterations required to obtain an accuracy ϵ , up to constants and factors logarithmic in problem parameters. We denote $\Delta F_0 := F(w_0) - F^*$ and $\Delta_0 = \|w_0 - w^*\|_2$. Constants a and D are defined so that $-D \leq w(u) \leq 0$ for all u Δ and each $f_{\mu w}^{(i)}$ is $a/\mu - smooth$ for $i \in [n]$.

When a smooth score function is not linear in w, [21] proposed to use the prox-linear algorithm [4],[6]. At each step, the latter linearizes the mapping around the current iterate w_k, resulting in a convex model $w \to h(w_k + \nabla g(w_k)^T(w - w_k))$ of $h \circ g$ around w_k. The overall convex model of the objective F with an additional proximal term is then minimized. The next iterate is given by

$$w_{k+1} = argmin_{w \in R^d} \frac{1}{n}\sum_{i=1}^{n} h(w_k + \nabla g(w_k)^T(w - w_k)) + \frac{\lambda}{2}\|w\|_2^2 + \frac{1}{2\gamma}\|w - w_k\|^2$$

where $g^{(i)}$ is the mapping associated with the i-th sample and $\gamma > 0$ is the parameter of the proximal term.

7 Experiments

In this section, we study the experimental behavior of the proposed algorithms on visual object localization structured prediction. We compare the performance of various optimization algorithms based on the number of calls to a smooth inference oracle. Moreover, following literature for algorithms based on SVRG [13], we exclude the cost of computing the full gradients.

Visual object localization :
The task consists in predicting the spatial location of the visual object in an image. We follow the methodology outlined in [10] to construct $\phi(x, y)$.

[10] solves the CNN localization problem by performing calculations in the Using Region Recognition paradigm, which has been successfully applied to object detection and semantic segmentation. At the time of testing, the method generates approximately 2,000 class-independent candidate regions for the input image. [10] used CNN to extract a fixed-length feature vector from each candidate region, and then uses a specific class of linear SVM to classify each of the regions. They use a simple technique (affine image warping) to calculate a fixed-size CNN input for each candidate region, regardless of the shape of the region. Figure 3 gives an overview of the method and highlights some of the results. Since the system combines candidate regions and CNNs, they abbreviate the method to R-CNN: Region with CNN features.

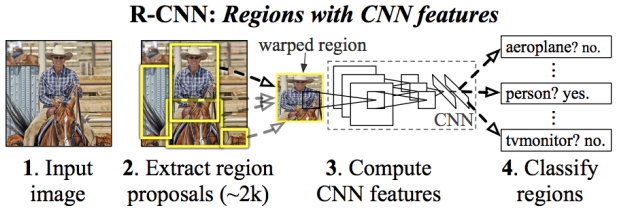


Figure 3: Overview of the object detection system Our system (1) receives an input image, (2) extracts approximately 2,000 candidate regions from bottom to top, (3) uses a large CNN to calculate features for each candidate region, and then (4) Use a specific type of linear SVM to classify each region.

The second challenge in detection is that the labeled data is small and the amount currently available is not sufficient to train a large CNN. The conventional solution to this problem is to use unsupervised pre-training followed by supervised fine-tuning. The second main contribution of this paper is to show supervised pre-training on a large auxiliary data set (ILSVRC), and then fine-tuning in a specific domain on a small data set (PASCAL). This method is used when the data is insufficient. Learn effective examples of high-capacity CNNs. In this experiments, fine-tuning for detection tasks improved mAP performance by 8 percentage points. After fine-tuning, the system achieved mAP to 54% on VOC2010, while the mAP of the highly adjusted HOG-based DPM was 33%. The concurrent work done by Donahue et al showed that Krizhevsky's CNN can be used as a black box (without fine-tuning) to perform feature extraction and produce excellent performance in several recognition tasks, including scenes Classification, fine-grained sub-classification, and domain adaptation.

The target detection system includes three modules. The first generation of class-independent candidate regions. These candidate regions define the set of candidate detections available for our detector. The second module is a large convolutional neural network that extracts fixed-length feature vectors from each region. The third module is some specific categories of linear SVMs.

Candidate area. Various recent papers provide methods for generating class-independent candidate regions. Examples include: objectness [1], selective search [29], class-independent candidates [7], CPMC, multi-scale combined grouping, and Ciresan et al. To detect mitotic cells by applying CNN to regularly spaced square tailoring, This is a special case of candidate regions. R-CNN is agnostic to specific candidate region methods. They use selective search to achieve control comparison with previous detection work.

Feature extraction. R-CNN used the Caffe implementation of CNN described by Krizhevsky et al. To extract 4096-dimensional feature vectors from each candidate region. The feature is calculated by spreading the 227×227 RGB image minus the average through 5 convolutional layers and the predecessors of two fully connected layers. In order to calculate the feature information of the candidate frame, we must first convert the image data into a fixed size, which is compatible with the extracted CNN. For irregular areas, we take a simple approach, regardless of his aspect ratio .

Given an image and an object of interest, the task is to localize the object in the given image, i.e., determine the best bounding box around the object. A related, but harder task is object detection, which requires identifying and localizing any number of objects of interest, if any, in the image.

Here, we restrict our selves to pure localization with a single instance of each object. Given an image $x \in X$ of size $n_1 \times n_2$, the label $y \in Y$ is a bounding box, where $Y(x)$ is the set of all bounding boxes in an image of size $n_1 \times n_2$. Note that $Y(x) = O(n_1^2 n_2^2)$.

Dataset : We used Blood Cell Images Dataset from Kaggle. This dataset contains 12,500 augmented images of blood cells with accompanying cell type labels . There are approximately 3,000 images for each of 4 different cell types grouped into 4 different folders (according to cell type). The cell types are Eosinophil, Lymphocyte, Monocyte, and Neutrophil. This dataset is accompanied by an additional dataset containing the original 410 images as well as two additional subtype labels (WBC vs WBC) and also bounding boxes for each cell in each of these 410 images. We focus on the "Lymphocyte" and we consider all images with only a single occurrence of the object, and train anindependent model for each class.
The figure 4 illustrates a randomly drawn images from our dataset.

Evaluation Methodology. We used the formulation based on R-CNN approach [10] and we consider a pre-trained

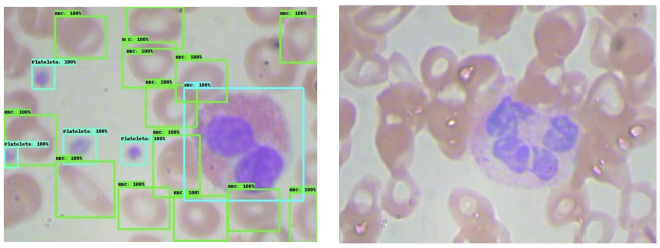


Figure 4: (a) In the left an example of the labeled cell image,(b) In the right a randomly drawn image from our dataset.

AlexNet Krizhevsky et al.[14] on ImageNet Russakovsky et al [24].

For a proposed box y, we consider a patch $x \mid_y$ of image x cropped to box y, and rescale it to 64×64 and pass it through AlexNet. We take the output of **conv4** penultimate convolutional layer of size $3 \times 3 \times 256$ as the feature map $\Phi(x, y)$ and we define the score ϕ as

$$\phi(x, y, w) =< w, \Phi(x, y) >$$

.

Loss Function : IoU metric [8] is used to measure the quality of localization. Given bounding boxes y, y', the IoU is defined as the ratio of the intersection of the bounding boxes to the union :

$$\text{IoU}(y, y') = \frac{\text{Area}(y) \cap \text{Area}(y')}{\text{Area}(y) \cup \text{Area}(y')}$$

We then use the $1 - \text{IoU}$ loss defined as $\ell(y, y') = 1 - \text{IoU}(y, y')$

Inference : For a given input image x, we follow the R-CNN approach [10], and use selective search [29] to prune the search space. In particular, for an image x, we use the selective search implementation provided by Open CV [3] and take the top 1000 candidates returned to be the set $\hat{\mathcal{Y}}(x)$, which we use as a proxy for $\mathcal{Y}$. The max oracle and the top-K oracle are then implemented as exhaustive searches over this reduced set $\hat{\mathcal{Y}}(x)$.

The experiments use K =10 , where the running time of the top-K oracle is independent of K and the performance of the tested algorithms is robust to the value of K.

Hyper-parameter tuning : The experiments compare various convex stochastic and incremental optimization methods for structured prediction. Some algorithms require tuning one or more hyperparameters such as the learning rate. We use grid search to find the best choice of the hyperparameters, a wide range of hyperparameter values achieved nearly equal performance in terms of the best localization accuracy, also known as CorLoc (for correct localization) over the given time horizon. So we choose the value of the hyperparameter that achieves the best objective

function value within a given iteration budget.

All experiments are repeated ten times, and prediction performance averaged over ten trials is reported.

Analysis : In Figure 5 we compare the proposed methods performance to various convex stochastic and incremental optimization methods for structured prediction. The proposed methods converge faster in terms of training error while achieving a competitive performance in terms of the performance metric on a held-out set. In general, small values of the smoothing parameter lead to better optimization performance for Casimir-SVRG-const, where Casimir-SVRG-adapt is more robust to the choice of μ ,the adaptive smoothing strategy is more robust to the choice of the smoothing parameter, We find that Casimir-SVRG-adapt is robust to the choice of the warm start strategy while Casimir-SVRG-const is not the Prox-center choice being the best compromise between acceleration and compatibility. The method to choose the hyperparameter K results within one standard deviation between them.

8 Conclusionn

In this work, we examined the structured prediction models, and studied two methods for solving the training of structural support vector machines. We also introduced a general notion of smooth inference oracles in the context of black-box first-order optimization. Finally we illustrated the potential of the new proposed methods of incremental optimization algorithm and we compared their performance to various convex stochastic and incremental optimization methods for structured prediction, enjoying worst-case complexity bounds and demonstrating competitive performance on two real-world problems.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, Nov 2012.

[2] Amir Beck and Marc Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22, 01 2012.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] James V. Burke. Descent methods for composite nondifferentiable optimization problems. *Mathematical Programming*, 33(3):260–279, 1985.

[5] Koby Crammer, Yoram Singer, Nello Cristianini, John Shawe-Taylor, and Bob Williamson. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res*, 2, 01 2002.

[6] D. Drusvyatskiy and C. Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, 178(1):503–558, 2019.

[7] Ian Endres and Derek Hoiem. Category independent object proposals. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 575–588, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[8] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[9] G. D. Forney. The viterbi algorithm. *Proc. of the IEEE*, 61:268 – 278, March 1973.

[10] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[11] He He, Hal Daume III, and Jason M Eisner. Learning to search in branch and bound algorithms. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3293–3301. Curran Associates, Inc., 2014.

[12] Thorsten Joachims, Thomas Finley, and Chun-Nam Yu. Cutting-plane training of structural svms. *Machine Learning*, 77:27–59, 10 2009.

[13] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, page 315–323, Red Hook, NY, USA, 2013. Curran Associates Inc.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[15] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. Catalyst acceleration for first-order convex optimization: From theory to practice. *J. Mach. Learn. Res.*, 18(1):7854–7907, January 2017.

[16] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[17] Yu. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, 16(1):235–249, 2005.

[18] Yu Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, May 2005.

[19] Yurii Nesterov. A method for solving the convex programming problem with convergence rate o$(1/k^2)$.1983.

[20] Courtney Paquette, Hongzhou Lin, Dmitriy Drusvyatskiy, Julien Mairal, and Zaid Harchaoui. Catalyst for gradient-based nonconvex optimization. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 613–622, Playa Blanca, Lanzarote, Canary Islands, 09–11 Apr 2018. PMLR.

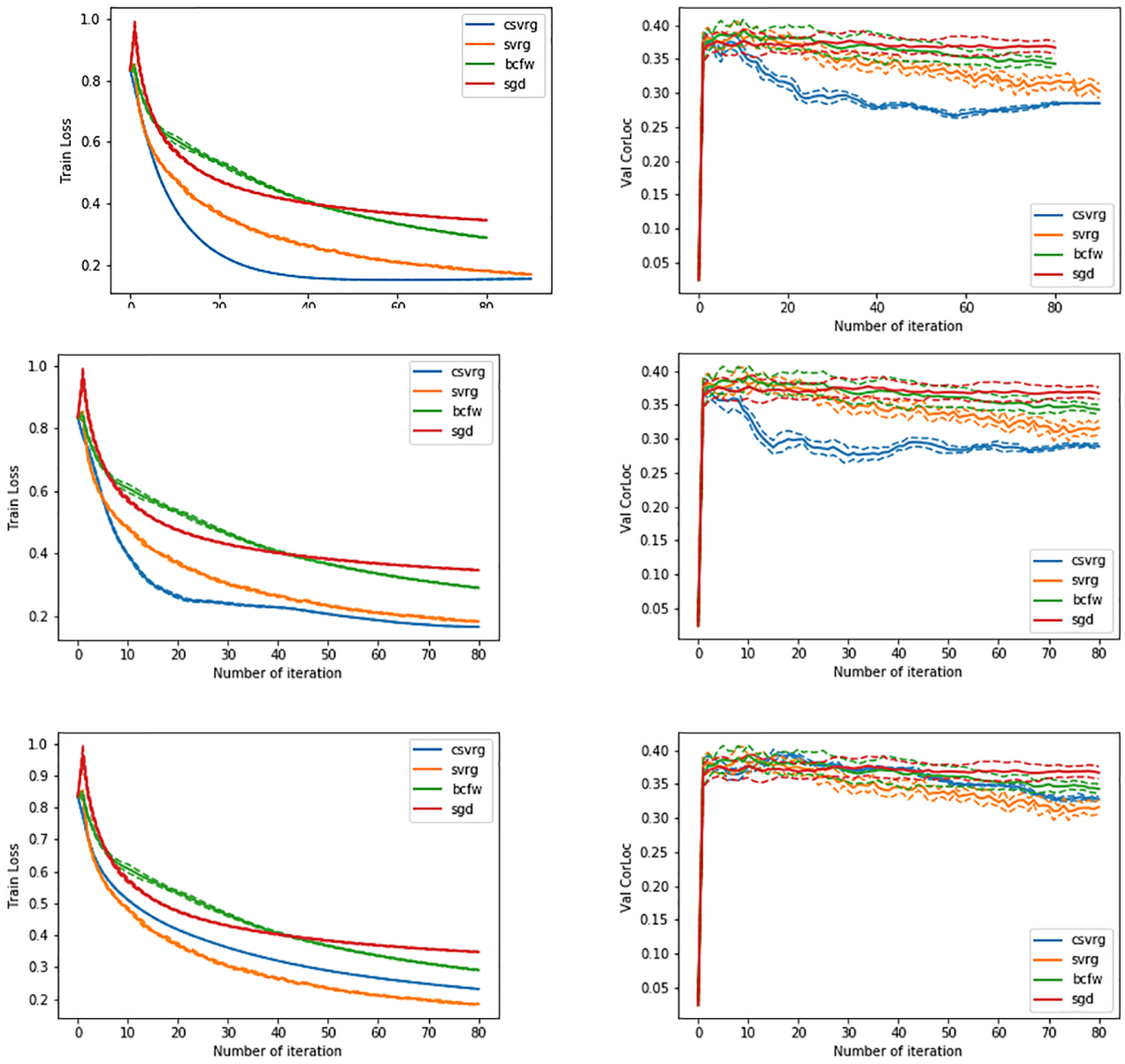


Figure 5: Comparison of the effect of Warm Start Strategies on the proposed methods performance and the various convex stochastic and incremental optimization methods for structured prediction.

[21] Krishna Pillutla, Vincent Roulet, Sham Kakade, and Zaid Harchaoui. A smoother way to train structured prediction models, 02 2019.

[22] Nathan Ratliff, J. Andrew (Drew) Bagnell, and Martin Zinkevich. (online) subgradient methods for structured prediction. In *Proceedings of Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*, March 2007.

[23] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[25] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine*

Learning, ICML '07, page 807–814, New York, NY, USA, 2007. Association for Computing Machinery.

[26] Ben Taskar, Dan Klein, Mike Collins, Daphne Koller, and Christoper Manning. Max-margin parsing. pages 1–8, 01 2004.

[27] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. *Machine Learning*, 07 2004.

[28] Jasper Uijlings, K. Sande, T. Gevers, and Arnold Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013.

[29] Jasper Uijlings, K. Sande, T. Gevers, and Arnold Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013.

[30] Alexander Vezhnevets, Vittorio Ferrari, and Joachim Buhmann. Weakly supervised structured output learning for semantic segmentation. 06 2012.

[31] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 1169–1176, New York, NY, USA, 2009. Association for Computing Machinery.