Efficient Large-Scale Structured Learning

Steve Branson
Caltech
Pasadena, CA 91125
sbranson@caltech.edu

Oscar Beijbom
UC San Diego
La Jolla, CA 92093
obeijbom@cs.ucsd.edu

Serge Belongie
UC San Diego
La Jolla, CA 92093
sjb@cs.ucsd.edu

Abstract

We introduce an algorithm, SVM-IS, for structured SVM learning that is computationally scalable to very large datasets and complex structural representations. We show that structured learning is at least as fast–and often much faster–than methods based on binary classification for problems such as deformable part models, object detection, and multiclass classification, while achieving accuracies that are at least as good. Our method allows problem-specific structural knowledge to be exploited for faster optimization by integrating with a user-defined importance sampling function. We demonstrate fast train times on two challenging largescale datasets for two very different problems: ImageNet for multiclass classification and CUB-200-2011 for deformable part model training. Our method is shown to be 10-50 times faster than $\text{SVM}^{\text{struct}}$ *for cost-sensitive multiclass classification while being about as fast as the fastest 1-vs-all methods for multiclass classification. For deformable part model training, it is shown to be 50-1000 times faster than methods based on* $\text{SVM}^{\text{struct}}$*, mining hard negatives, and Pegasos-style stochastic gradient descent. Source code of our method is publicly available.*

1. Introduction

Binary classification algorithms have been the cornerstone of machine learning methods over the last twenty years. It is common practice to map computer vision problems into binary classification problems, which can then be solved using black box optimization algorithms. For example, sliding window object detection problems and deformable part models can be trained by drawing random bounding box locations or part location assignments as negative examples and multiclass classification problems can be solved by combining multiple 1-vs-rest binary classification problems. The popularity of binary classification perhaps has more to do with a decomposition of labor between researchers in machine learning and applied fields like computer vision rather than any inherent theoretical properties; binary classification is an easy to define, non-application-specific problem to study.

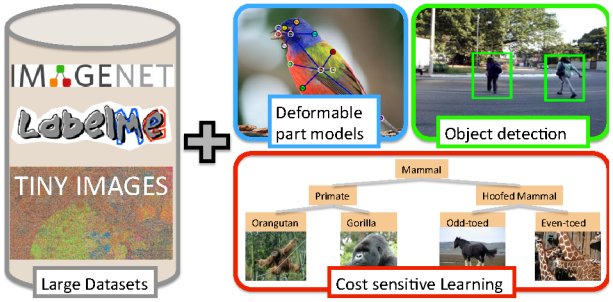


Figure 1. We introduce an efficient optimization algorithm that makes structured learning computationally tractable on very large datasets, significantly reducing train time for deformable part models, object detection, and cost-sensitive multiclass learning

It is our goal in this paper to introduce faster, customized optimization algorithms for common computer vision problems like object detection, deformable part models, and cost sensitive multiclass learning. Improved computational efficiency is obtained by exploiting structural properties of the problem that are discarded by binary classification methods. For example, for multiclass classification, we know that all output classes are mutually exclusive. For object detection, we know all possible bounding box locations occur on a 2D grid. For part-based detection, we know the spatial relationship between parts. The improved computational efficiency comes from concentrating processing on output labels (*e.g.*, bounding box locations) that have greater training error. The potential performance gains for structured learning become larger & larger as the problem becomes less and less like binary classification– it is a small gain for multiclass classification, a sizable gain for object detection, and a very significant gain for deformable part models.

At the same time, structured learning algorithms such as structured SVMs [25] are often associated with being slow to train in practice. For example, when applied to sliding window object detection, the algorithm used in $\text{SVM}^{\text{struct}}$ [25] requires running a sliding window detector on each training image every time the detection model is updated. In this setting, binary classification methods can end

up being faster by heuristically randomly sampling bounding boxes without absorbing the cost of firing the detector.

In this paper, we propose two changes that allow structured SVMs to be at least as fast as their binary SVM counterparts for problems such as object detection, deformable part models, and multiclass classification. First, we apply ideas from online sub-gradient methods [21, 19] and sequential dual optimization algorithms [11, 16], which are inclusive of the fastest algorithms for training linear SVMs and often significantly faster than the cutting plane algorithm used by $\mathrm{SVM}^{\mathrm{struct}}$ (a popular solver for structured SVMs). Second, we allow problem-specific knowledge to be injected into the optimization algorithm by incorporating a user-defined importance sampling function. Here, our optimization algorithm takes an update step with respect to a set of intelligently selected output labels (*e.g.*, a set of bounding boxes in a particular image that have high training error with respect to the current detector). This change allows the method to be a superset of techniques used by conventional structured learning methods and commonly used heuristics for mapping problems into binary classification problems, with additional parameters to explore to tailor optimization to a particular application.

Our main contributions are as follows: 1) We introduce SVM-IS, a fast structured SVM solver that is easy to apply to novel problems. 2) We show how our solver can be used to create faster learning algorithms for cost-sensitive multiclass SVMs, object detection, and deformable part models.

We demonstrate our results on ImageNet and CUB-200-2011, two challenging, large scale datasets for multiclass classification and deformable part model training, and demonstrate reductions in train time for deformable part model training and cost-sensitive multiclass SVM learning by a factor of ~ 100 for deformable part models, and ~ 20 for cost-sensitive multiclass classification.

2. Background and Related Work

Structured SVMs: Structured SVMs [23, 25, 24] provide a method for training a system to predict a multidimensional structured output, such as a bounding box, set of part locations, or segmentation. They minimize a convex upper bound on a customizable loss function. Structured SVMs are a superset of SVM-based learning algorithms, which includes many of the most popular and highest performing algorithms for object detection [12, 7, 3], and multiclass classification [17, 9]. They have been applied to training object detectors using more appropriate loss functions [2], multiclass SVMs with customizable class confusion costs [5], and deformable part models [27, 4, 20].

Fast Solvers For Large Scale Linear SVMs: Linear kernel SVMs have been shown to have dramatically better computational properties than non-linear kernel SVMs for both training and testing. Whereas non-linear kernel SVMs train in time that is at least quadratic in the number of training examples, solvers such as Liblinear [11] and $\mathrm{SVM}^{\mathrm{perf}}$ [14] train in linear time, or in time that does not even depend on the size of the training set (at least in expectation) [21]. Among fast linear SVM solvers, online sub-gradient algorithms [21] and sequential dual algorithms [11] are faster than methods that train in batch.

Fast Solvers For Large Scale Structured SVMs: The basic convex optimization methods used by the above algorithms are general to many convex optimization problems such as structured SVMs [19, 6, 25]; however, there is a comparative scarcity of publicly available fast methods for structured SVMs. Kakade and Shalev-Shwartz provided a template algorithm for developing fast optimization algorithms for novel strongly convex optimization problems and a theoretical framework for studying their statistical convergence properties [15]. Analysis of our algorithm is based on this template. Guzman et al. [13] recently introduced a cutting-plane-based structured SVM solver that incorporates a similar idea to our importance sampling routine. Our method is different in that it is based on extending sequential dual and sub-gradient algorithms (which are often much faster than cutting plane algorithms).

3. Algorithm and Approach

We first briefly discuss notation and the problem definition, then introduce our algorithms in Sections 3.1-3.2.

Structured Learning: Let X be an input example and $Y = y_1 ... y_O$ be a multidimensional structured output defining its ground truth label. For example, for deformable part models, X is an image and each $y_p \in Y$ defines the location of a part in the image. A structured prediction function predicts the label $\arg\max_Y s(X, Y)$ with highest score:

$$s(X, Y) \quad = \quad \langle \mathbf{w}, \Psi(X, Y) \rangle \tag{1}$$

where $s(X, Y)$ is a score measuring the goodness of a particular label Y, $\Psi(X, Y)$ is a feature space extracted with respect to label Y (*e.g.*, features extracted around part locations $y_1 ... y_O$), and $\mathbf{w}$ is a learned vector of weights. Let $\Delta(Y, Y_i)$ be a customizable loss function associated with predicting label Y when the true label is Y_i. For example, $\Delta(Y, Y_i)$ can be a measure of the overlap between predicted part locations and their ground truth locations. Structured SVM learning minimizes the training error:

$$F_n(\mathbf{w}) = \sum_{i=1}^{n} f(\mathbf{w}; Z_i) \tag{2}$$

$$f(\mathbf{w}; Z_i) = \frac{\lambda}{2} \|\mathbf{w}\|^2 + \ell(\mathbf{w}, Z_i) \tag{3}$$

$$\ell(\mathbf{w}, Z_i) = \max_Y \left(s(X_i, Y) + \Delta(Y, Y_i) \right) - s(X_i, Y_i) \tag{4}$$

where each $Z_i = (X_i, Y_i)$ is a training example, λ is a regularization constant. The function $\ell(\mathbf{w}, Z_i)$ is an upper

bound on $\Delta(Y, Y_i)$ that is convex in $\mathbf{w}$. Let $\bar{Y}_i$ be the value of Y that maximizes $\ell(\mathbf{w}, Z_i)$:

$$\bar{Y}_i \quad = \quad \arg\max_Y \left(s(X_i, Y) + \Delta(Y, Y_i)\right) \tag{5}$$

Solving Eq 5 resembles a prediction problem (Eq 1) and is the primary computation for many structured learning optimization algorithms[1]. The set of problems that are appropriate for structured SVMs is limited to problems and loss functions for which Eq 5 is efficiently solvable.

Dual Problem: Eq 2 can be represented by its equivalent dual problem

$$\max_{\alpha_1 \ldots \alpha_n} \mathcal{D}_n(\alpha_1 \ldots \alpha_n) = \min_{\mathbf{w}} F_n(\mathbf{w}) \tag{6}$$

The dual objective is often useful for deriving optimization algorithms and theoretical guarantees. The problem $\max_{\alpha_1,\ldots,\alpha_n} \mathcal{D}_n(\alpha_1 \ldots \alpha_n)$ can be written as (see [25])

$$\max_{\alpha_1 \ldots \alpha_n} -\frac{1}{2\lambda n} \left\| \sum_{i,Y} \alpha_i(Y) v_i(Y) \right\|^2 + \sum_{i,Y} \alpha_i(Y) \Delta(Y, Y_i)$$
$$\text{s.t., } \forall_i \left(\forall_Y, \alpha_i(Y) \geq 0, \quad \sum_Y \alpha_i(Y) \leq 1 \right) \tag{7}$$

where each $\mathbf{v}_i(Y)$ is a feature vector

$$\mathbf{v}_i(Y) \quad = \quad \Psi(X_i, Y) - \Psi(X_i, Y_i) \tag{8}$$

that is weighted by a scalar $\alpha_i(Y)$. For example, for sliding window based object detection, $\mathbf{v}_i(Y)$ is a feature vector for every possible bounding box Y in image X_i. The bounding box feature vectors $\mathbf{v}_i(Y)$ that have non-zero weights $\alpha_i(Y)$ correspond to bounding boxes in the margin (*e.g.*, have non-zero hinge loss) and can be thought of as "support vectors". Although this is a dauntingly large number of parameters to learn (a parameter $\alpha_i(Y)$ for every bounding box in every image), for various ϵ-accurate approximate optimization algorithms, the number of non-zero $\alpha_i(Y)$ parameters will be independent of the size of structured output space (the number of bounding boxes per image).

The relationship between the dual parameters α and primal parameters $\mathbf{w}$ is

$$\mathbf{w}(\alpha) = -\frac{1}{\lambda n} \sum_{i,Y} \alpha_i(Y) v_i(Y) \tag{9}$$

3.1. Multi Sample Online Dual Ascent Algorithm

We consider two different related dual coordinate ascent optimization algorithms, which work by sequentially processing examples and approximately maximizing the dual objective $\mathcal{D}(\cdot)$ with respect to the parameters α_i of one example i at a time. Fast algorithms for linear SVMs such as Pegasos and Liblinear can be understood as variants of these algorithms; whereas our algorithm has additional choices that make it more appropriate for tweaking optimization speed for a particular application.

The first algorithm, shown in Algorithm 1 processes one example $i(t)$ in each timestep t. Let $F_t(\mathbf{w})$ and $\mathcal{D}_t(\alpha_1, ..., \alpha_t)$ correspond to an online version of the primal and dual structured SVM objectives, where in contrast to Eq 2 error is accumulated over each processed example:

$$F_t(\mathbf{w}) = \quad \sum_{s=1}^{t} f(\mathbf{w}; Z_{i(s)}) \tag{10}$$

Note that if each $i(s)$ is selected independently at random, the objectives have equivalent expected values: $\mathbb{E}[F_t(\mathbf{w})/t] = \mathbb{E}[F_n(\mathbf{w})/n]$. In each timestep, the algorithm solves for weights on each sample, with the objective of maximally increasing the online dual objective

$$\Delta\mathcal{D}_t(\alpha_t) = \mathcal{D}_t(\alpha_1, ..., \alpha_t) - \mathcal{D}_{t-1}(\alpha_1, ..., \alpha_{t-1}) \tag{11}$$

Examining Eq 7-9, this is equivalent to the QP problem:

$$\max_{\alpha_t} \frac{1}{2} - \alpha_t^T Q \alpha_t + \ell_t^T \alpha_t + c, \quad \text{s.t. } \alpha_t^j \geq 0, \quad \sum_j \alpha_t^j \leq 1 \tag{12}$$

where $\ell_t = [\ell_t^1, ..., \ell_t^K]$ and $\alpha_t = [\alpha_t^1, ..., \alpha_t^K]$ are vectors, c is a constant, Q is a $K \times K$ matrix with elements

$$Q_{jk} \quad = \quad \frac{\left\langle \mathbf{v}_t^j, \mathbf{v}_t^k \right\rangle}{\lambda t}, \quad c = \frac{\lambda(t-1)\|w^{t-1}\|^2}{2t} \tag{13}$$

and α_t^j, $\mathbf{v}_t^j$, and ℓ_t^j are shorthand for per sample weights $\alpha_t(\bar{Y}_t^j)$, features $\mathbf{v}_t(\bar{Y}_t^j)$, and loss $\langle \mathbf{w}^{t-1}, \mathbf{v}_t^j \rangle + \Delta(\bar{Y}_t^j, Y_t)$, respectively. Using Eq 9, this will result in an update of the model weights $\mathbf{w}^t$ according to:

$$\mathbf{w}^t \leftarrow \frac{t-1}{t} \mathbf{w}^{t-1} - \frac{1}{\lambda t} \sum_{j=1}^{K} \alpha_t^j \mathbf{v}_t^j \tag{14}$$

Let R be a bound on the magnitude of the feature space $\|\Psi(X, Y)\| \leq R$:

Theorem 3.1 *Algorithm 1 obtains an ϵ-accurate solution in at most $\tilde{O}(\frac{R^2}{\lambda\epsilon})$ iterations in expectation, for any subroutine* IMPORTANCESAMPLE *that includes $\bar{Y}_t^1 = \bar{Y}_i$ (Eq 5) and any approximate solver on line 7 that is at least as good as the one obtained by setting $\alpha_t^1 = 1$.*

This theorem is a direct consequence of Theorem 1 from Kakade and Shalev-Shwartz [15]. Note that this bound does not depend directly on the size of the training set. The proof is based on the observation that choosing $\alpha_t^1 = 1$ increases $\mathcal{D}_t$ by a predictable amount (see supplementary material).

[1]Note that a sub-gradient $\nabla\ell(\mathbf{w}, Z_i)$ of $\ell(\mathbf{w}, Z_i)$ can be computed from is computable from $\bar{Y}_i$ as $\ell(\mathbf{w}, Z_i) = \Psi(X_i, \bar{Y}_i) - \Psi(X_i, Y_i)$

Algorithm 1 SVM-IS-Online

1: Initialize to zero: $\mathbf{w}^0$
2: **for** $t = 1...$ **do**
3: Choose an example i (randomly)
4: $\bar{Y}_t^1 ... \bar{Y}_t^K \leftarrow \text{IMPORTANCESAMPLE}(X_i, Y_i, \mathbf{w}^{t-1})$
5: $\forall_j,\ \mathbf{v}_t^j \leftarrow \Psi(X_i, \bar{Y}_i^j) - \Psi(X_i, Y_i)$
$\ell_t^j \leftarrow \langle \mathbf{w}^{t-1}, \mathbf{v}_t^j \rangle + \Delta(\bar{Y}_i^j, Y_i)$
6: Define $\Delta\mathcal{D}_t := -\frac{1}{2}\alpha^T Q \alpha + \ell_t^T \alpha + c$
$Q_{jk} := \frac{\langle \mathbf{v}_t^j, \mathbf{v}_t^k \rangle}{\lambda t}, \quad c = \frac{\lambda(t-1)\|w^{t-1}\|^2}{2t}$
7: Approx. solve $\alpha_t \leftarrow \arg\max_\alpha \Delta\mathcal{D}_t(\alpha)$
s.t. $\forall_j,\ \alpha_t^j \geq 0,\ \sum_j \alpha_t^j \leq 1$
8: $\mathbf{w}^t \leftarrow \frac{t-1}{t}\mathbf{w}^{t-1} - \frac{\sum_{j=1}^K \alpha_t^j \mathbf{v}_t^j}{\lambda t}$
9: **end for**

Optimizing Line 7: A better solution to Eq 12 (by incorporating $\alpha_t^2 ... \alpha_t^K$) will converge at least as quickly, with a larger increase yielding faster convergence. Due to space limitations, we discuss two efficient algorithms for solving Eq 12 in the supplementary material. The runtime for both such algorithms is roughly equal to the time needed to compute one dot product $\langle \mathbf{w}^{t-1}, \mathbf{v}_t(\bar{Y}_t^j) \rangle$ per sample j. The first algorithm is a general purpose one-pass approximate algorithm, whereas the 2nd is an exact algorithm that applies to multiclass classification problems.

Heuristics for choosing samples: Examining Eq 12, given a sample set $\bar{Y}_t^1 ... \bar{Y}_t^{j-1}$, the utility of a new sample $\bar{Y}_t^j$ is proportional to $\alpha_t(\bar{Y}_t^j)\left(-\frac{1}{2}\sum_{k=1}^j \alpha_t(\bar{Y}_t^k) Q_{jk} + \ell_t^j\right)$. This will be high if $\ell_t^j = s(X_t, \bar{Y}_t^j) + \Delta(\bar{Y}_t^j, Y_t) - s(X_t, Y_t)$ is high (the loss associated with sample $\bar{Y}_t^j$) and each Q_{ij} is low. Thus in general we should favor samples with high loss that are as uncorrelated as possible ($\mathbf{Q}$ is as close to a diagonal matrix as possible).

Detecting Convergence: Convergence is detected based on the primal dual gap $F(\mathbf{w}) - \mathcal{D}(\alpha)$ [2] is less than ϵ. Note that in Algorithms 1-2 $\mathcal{D}(\alpha)$ can be efficiently computed by summing $\Delta\mathcal{D}$ in each iteration. Rather than explicitly compute $\frac{F(\mathbf{w})}{n}$, we approximate it by the accumulated loss $\frac{\sum_{s=1}^t f(\mathbf{w}^{s-1}; Z_{i(s)})}{t}$, which exceeds $\frac{F(\mathbf{w})}{n}$ if the primal objective tends to be decreasing.

3.2. Multi-Sample Dual Ascent Algorithm

A similar algorithm, depicted in Algorithm 2, also sequentially processes examples. It differs from Algorithm 1 primarily with respect to what happens when an example is processed multiple times. On later iterations, it optimizes a change $\Delta\alpha$ to the parameters α_i for example i:

$$\Delta\mathcal{D}_n(\Delta\alpha_i) = \mathcal{D}_n(\alpha_1, ..., \alpha_i + \Delta\alpha_i, ..., \alpha_n) - \mathcal{D}_n(\alpha_1, ..., \alpha_n) \quad (15)$$

[2] This upperbounds the training error, because by weak duality, for any α, $\frac{\mathcal{D}(\alpha) \leq \min_{\mathbf{w}} F(\mathbf{w})}{n}$

Algorithm 2 SVM-IS

1: Initialize to zero: $\mathbf{w}^0, t, \left\{\bar{\mathbf{v}}_i, \bar{\ell}_i, \bar{\alpha}_i\right\}_{i=1}^n$
2: **for** $t = 1...$ **do**
3: Choose an example i (randomly)
4: $\bar{Y}_t^1 ... \bar{Y}_t^K \leftarrow \text{IMPORTANCESAMPLE}(X_i, Y_i, \mathbf{w}^{t-1})$
5: Define $\mathbf{v}_t := [\mathbf{v}_t^1 ... \mathbf{v}_t^K, \bar{\mathbf{v}}_i]$, $\ell_t := [\ell_t^1 ... \ell_t^K, \bar{\ell}_i]$
$\Delta\alpha := [\alpha_t^1 ... \alpha_t^K, \Delta\bar{\alpha}_i]$, $\alpha_t := [\alpha_t^1 ... \alpha_t^K, \bar{\alpha}_i + \Delta\bar{\alpha}_i]$
$\Delta\mathcal{D}_n := -\frac{1}{2}\Delta\alpha^T Q \Delta\alpha + \ell_t^T \Delta\alpha, \quad Q_{jk} := \frac{\langle \mathbf{v}_t^j, \mathbf{v}_t^k \rangle}{\lambda n}$
6: Approx. solve $\Delta\alpha_t \leftarrow \arg\max_{\Delta\alpha} \Delta\mathcal{D}_n(\Delta\alpha)$
s.t. $\forall_{j=1}^{K+1},\ \alpha_t^j \geq 0,\ \sum_j \alpha_t^j \leq 1$
7: $\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \frac{\sum_{j=1}^{K+1} \Delta\alpha_t^j \mathbf{v}_t^j}{\lambda n}$
8: Merge samples: set $\bar{\mathbf{v}}_i$ and $\bar{\ell}_i$ to the average of $\mathbf{v}_t$ and ℓ_t weighted by α_t, and set $\bar{\alpha}_i$ to the sum of α_t
9: **end for**

While pseudo-code is provided in Algorithm 2, we discuss derivation and properties in the supplementary material. By the arguments presented in [22], it shares the same worst case convergence rate as Algorithm 1, typically with faster convergence when making multiple passes through the data.

3.3. Comparison to Other Algorithms

In this section, we briefly mention various possible optimization algorithms for structured SVMs (all of which are evaluated in our experimental results).

Cutting Plane Algorithm: $\text{SVM}^{\text{struct}}$–a popular software package for structured SVMs–optimizes Eq 2 using a cutting plane algorithm. An n-slack variety of $\text{SVM}^{\text{struct}}$ solves Eq 5 for all n training examples before solving a QP problem, whereas a 1-slack variety solves a QP problem after each time Eq 5 is solved for a particular example. The n-slack method is asymptotically slower (by a factor of n) than the 1-slack method and our algorithm in terms of the number of times the inference algorithm will be invoked. For the 1-slack method, solving the QP problem can become intractibly slow.

Stochastic Sub-Gradient Descent: Stochastic gradient descent (SGD) is a very simple algorithm with surprisingly good theoretical guarantees (for certain implementations). SGD iterates over each example sequentially, taking an update step:

$$\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - \eta_t \nabla f(\mathbf{w}^{t-1}; Z_t) \quad (16)$$

where $\nabla f(\mathbf{w}^{t-1}; Z_t)$ is the sub-gradient of Eq 3. A Pegasos-like update [21] uses a decaying step size $\eta_t = \frac{1}{\lambda t}$ followed by a projective step[3]. Applied to structured SVMs [19], this simple algorithm has as good asymptotic properties as the 1-slack $\text{SVM}^{\text{struct}}$ algorithm in terms of the number of times inference will be invoked (and asymptotically better properties than the n-slack algorithm), and

[3] $\mathbf{w}^t \leftarrow \frac{\min\left(1/\sqrt{\lambda}, \|\mathbf{w}^t\|\right)}{\|\mathbf{w}^t\|}\mathbf{w}^t$

doesn't suffer from slowness of solving a QP of growing size. One can verify that SGD is equivalent to Algorithm 1 with $K = 1$, $\bar{Y}_t^1 = \bar{Y}$, and choosing an update $\alpha_t^1 = 1$, meeting the criteria of Theorem 3.1. In practice, it converges more slowly, is more dependent on the regularization parameter, and has no easy way to test for convergence.

4. Applications

In this section, we show how Algorithm 1-2 can be applied to a variety of popular learning problems, including cost-sensitive multiclass SVMs, object detection, and deformable part models. The general template for applying our method to new problems is:

1. Implement a feature extraction routine $\Psi(X, Y)$
2. Implement a routine to solve Eq 5: $\bar{Y}_t = \arg\max_Y \left(s(X_t, Y) + \Delta(Y, Y_t)\right)$, which is similar to an inference problem
3. Choose a sample set $\bar{Y}_t^1, ..., \bar{Y}_t^K$ that includes $\bar{Y}_t$. Favor samples with high loss $s(X_t, \bar{Y}_t^j) + \Delta(\bar{Y}_t^j, Y_t)$, and where $\langle \mathbf{v}_t(\bar{Y}_t^i), \mathbf{v}_t(\bar{Y}_t^j) \rangle$ tends to be small for $i \neq j$

4.1. Cost-Sensitive Multiclass SVMs

Given a training set of image-label pairs $\{(X_i, Y_i)\}_{i=1}^n$ where $Y_i \in 1...C$ is a class label, a cost-sensitive multiclass SVM solves the optimization problem:

$$F_n^C(\mathbf{w}) = n\left(\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^n \epsilon_i\right) \tag{17}$$
$$\text{s.t.}, \forall_{i,c}, \langle \mathbf{w}_c, \phi(X_i)\rangle + \Delta^C(c, Y_i) \leq \langle \mathbf{w}_{Y_i}, \phi(X_i)\rangle + \epsilon_i$$

where $\phi(X)$ is a feature vector of length d, each $\mathbf{w}_c$ is a vector of weights for class c, $\mathbf{w} = [\mathbf{w}_1, ..., \mathbf{w}_C]$ concatenates the weights for all classes, and $\Delta^C(c, Y_i)$ is a confusion cost associated with predicting class c when the true label is Y_i. As in [25], this problem is solvable using a structured SVM, where $\Psi(X, Y)$ concatenates features for each class:

$$\Psi^C(X, Y) = [\psi_1(X, Y), ..., \psi_C(X, Y)] \tag{18}$$
$$\psi_c(X, Y) = \begin{cases} \phi(X) & \text{if } Y = c \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{19}$$

Choosing samples: For this problem, we choose a dense sample set that includes every possible class label $1, ..., C$. In the supplementary material, we derive a fast exact solver for Eq 12. The time of this update step is roughly equal to the time to evaluate one dot product $\langle \mathbf{w}, \Psi^C(X, Y)\rangle$.

4.2. Sliding Window Object Detection

A sliding window object detector can be trained using a structured SVM [2], where $Y = \{\text{x}, \text{y}, \text{scale}\}$ encodes a bounding box and $\Psi^B(X, Y)$ is a vector of features extracted at Y. Let $\Delta^B(Y, Y_i)$ be an arbitrary loss function associated with predicting bounding box Y when the true bounding box is Y_i. The sliding window detector can be trained by optimizing the structured SVM objective (Eq 2). Let $\mathbf{\Delta}_i^B$ encode all values of $\Delta^B(Y, Y_i)$ into an array, with $\mathbf{\Delta}_i^B[Y] = \Delta^B(Y, Y_i)$. Similarly, let $\mathbf{M}_i$ be an array of sliding window responses, such that $\mathbf{M_i}[Y] = \langle \mathbf{w}, \Psi(X_i, Y)\rangle$, and let $\mathbf{L_i} = \mathbf{M_i} + \mathbf{\Delta}_i^B$. Note that

$$\begin{aligned} \bar{Y}_i &= \arg\max_Y \left(\langle \mathbf{w}, \Psi(X_i, Y)\rangle + \Delta(Y, Y_i)\right) \\ &= \arg\max_{Y'} \mathbf{L}_i[Y'] \end{aligned} \tag{20}$$

Choosing samples: We choose a sparse sample set of bounding boxes $\bar{Y}_t^1 ... \bar{Y}_t^K$, that are the result of running the greedy non-maximal suppression technique used in [10]. This method greedily selects the bounding box with highest loss $\bar{Y}^j = \arg\max_Y \mathbf{L}_i[Y]$, then suppresses all overlapping bounding boxes by setting $\mathbf{L}_i[Y] \leftarrow -\infty$ for all Y that overlap with $\bar{Y}^j$. The motivation behind this sampling technique is that overlapping bounding boxes will tend to have more correlated features, such that $\langle \Psi(X_t, \bar{Y}_t^j), \Psi(X_t, \bar{Y}_t^i)\rangle$ is more likely to be high.

4.3. Deformable Part Model Based Detection

A Felzenszwalb-like deformable part model [12] can be trained using a structured SVM, where $Y = y_1, ..., y_P$ encodes a set of part locations, each of which is represented using a 4-tuple $y_p = \{\text{x}_p, \text{y}_p, \text{scale}_p, \text{aspect}_p\}$, where aspect_p defines a mixture component index (*e.g.*, it toggles between different appearance and spatial models for side view, frontal view, *etc.*), and parts can have tree-structured dependencies with spring costs connecting parent and child parts. Details of the mapping into structured SVMs are presented in [27, 4]. These methods concatenate appearance features and spatial offsets for all part-aspect pairs into a structured feature space $\Psi^P(X, Y)$. The label $\bar{Y}_i$ can be efficiently solved for using standard dynamic programming algorithms for pictorial structure inference. Similar to the method described in the previous section, this is implemented using a modified unary detection score $\mathbf{L_i^P} = \mathbf{M_i^P} + \mathbf{\Delta}_i^p$ for each part p, where $\mathbf{\Delta}_i^p[y_p]$ encodes the loss associated with placing part p at location y_p.

Choosing samples: We empirically evaluate two different importance sampling routines. The first is the same bounding box non-maximal suppression technique described in the previous section, applied to the score for the root of the part tree after running dynamic programming.

The second is based on the observation that two different samples $\bar{Y}_t^i$ and $\bar{Y}_t^j$ with different aspects (mixture component assignments) will necessarily satisfy $\langle \Psi(X_t, \bar{Y}_t^i), \Psi(X_t, \bar{Y}_t^j)\rangle = 0$. We create a modified version of the greedy non-maximal suppression method that favors

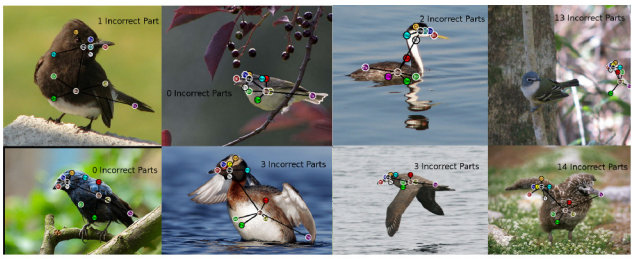


Figure 2. **Typical Part Detection Results on CUB-200-2011** with each color circle corresponding to a different semantic part. For each image, the loss by our performance metric is shown.

picking samples with diverse aspect labels. In this case, we always choose a sample $\bar{Y}_t^j$ with an aspect label that has been selected the fewest times, breaking ties by the score of the root of the part tree. This procedure is related to recent papers for finding n-best solutions for DPMs [18, 1]. Our method is more heuristic but faster, only requiring running inference (dynamic programming) once.

5. Experiments

We demonstrate generality and computational speed on two challenging, large-scale datasets: ImageNet for multi-class classification and CUB-200-2011 for pose registration and part-based detection.

5.1. Part-Based Detection

CUB-200-2011 [26] is a dataset of 11,788 images of birds labelled with 15 different part locations. We measure performance as the number of correctly predicted part locations, the same metric used in [4], where a predicted part location is considered to be correct if we correctly predict its visibility and its x,y-location is within some radius of the ground truth location (defined as within 1.5 standard deviations of where MTurker responses for that part). See Fig. 2 for example results.

Implementation Details: Empirically, we found that using a more complex pose mixture model was important for achieving good performance; however, doing so has implications of making training more computationally intensive. We used 14 different aspects (mixture components) for each of the 15 parts, and also predict part visibility by incorporating a mixture component with no appearance features. Each mixture component was represented by a 7X7 HOG template and spring costs between neighboring parts. We allowed the head and body to have different poses/aspects (*e.g.*, the head can be in left side-view while the body is in frontal view). The dimensionality of the feature space $\Psi(X_i, Y)$ was 81,902.

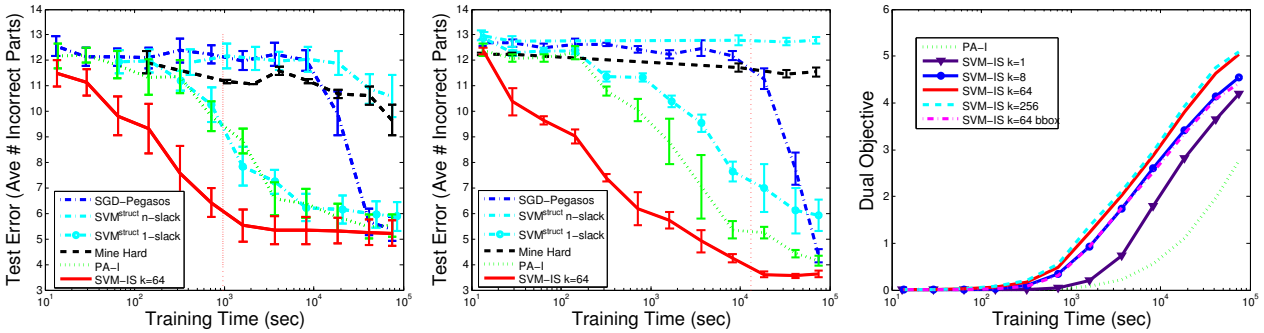
Summary of Results: Fig. 3 shows results on CUB-200-2011 for two different training set sizes $n = 400$ and $n = 5794$. For both training set sizes, we computed 5 random train/test splits and averaged results across splits. We compare training time to popular methods for object detection and deformable part model training, including the n-slack and 1-slack varieties of SVM$^{\text{struct}}$ [25] (which invokes our custom inference algorithm for part detection), and a method based on mining hard negatives, where on each round we augmented a binary training set with 100 negative examples per image. Our method was multiple orders of magnitude faster than these methods.

We also implemented a number of non-publicly available optimization methods that are particularly suited for large datasets, including stochastic gradient descent with a Pegasos-like update [19] (see Section 3.3) and the Online Passive Aggressive Update (PA-I) method from [6] (both were adapted to structured SVM learning). Our algorithm converged at least an order of magnitude faster than these methods. Note that training time for all methods is dominated by the time to run deformable part model inference, which shares the same implementation among all methods. Thus performance gains can be attributed to the respective optimization updates rather than to implementation details. All experiments were run on a single core 2.5GHz Xeon.

Choice of Sampling Method: Fig. 3 shows the effect of the choice of the importance sampling method and the number of samples chosen (parameter K). We see that computational efficiency kept improving as we increased K from 1 up to 256, despite the fact that each update step for $K = 256$ took approximately 20% longer than for $K = 1$. However, there was little gain from moving beyond $K = 64$. By default, we used our custom sampling method that was designed to favor samples with different aspect labels (see Section 4.3). Training for this method was about a factor of two faster than when sampling using non-maximal suppression of bounding boxes (see Section 4.2). Additional analysis is contained in Section 5.3.

5.2. Multiclass SVM

In this section we consider the task of large-scale cost-sensitive multiclass classification with a hierarchical loss function. We compare the proposed method to other SVM solvers and optimization algorithms. We also evaluate the influence of the K parameter of the importance sampling routine. ImageNet is a large image data base introduced by Deng *et al.*, comprising $> 14M$ images and $> 20k$ categories [9]. We use a subset of 200 randomly chosen categories (similar to sets RAND{A,B,C} in [8]), and the same SIFT-based bag-of-words feature representation with 1k visual words. Following Jia *et al.* [8], we consider the hierarchical loss, $C_{i,j}$, where $C_{i,j} = 0$, when $i = j$, and $C_{i,j} = h(i,j)$, the height of the lowest common ancestor, otherwise. We report results from a five fold cross validation ($4/5$ of the data was used for training, and $1/5$ for evaluation). The optimal value of $\lambda = \{10^{-3} \dots 10^{-8}\}$ was determined for each solver on a reference set. We use 1000 samples per category for the K sweep experiment and fast linear solver comparison, and 500 samples per category for the solver comparison (because we had problems running



(a) Results on 400 image training set (b) Results on 5794 image training set (c) Effect of Parameter K

Figure 3. **Part Detection Results on CUB-200-2011:** Qualitative results are shown in Fig. 2. (a-b) Results on training sets of size 400 and 5794, respectively. Our method SVM-IS converged orders of magnitude faster than popular learning methods, including the 1-slack and n-slack varieties of SVM^{struct} and a method based on mining hard examples and training a binary classifier using Liblinear. Relative gains became larger as the training set size increases. It was an order of magnitude faster than our implementation of other fast (non-publicly available) online learning algorithms applied to structured learning, including stochastic gradient descent with a Pegasos-like update [19] and the Online Passive Aggressive Update (PA-I) method from [6]. (c) A sweep though the parameter K indicate that computational efficiency kept improving as we increased K from 1 up to 256; however, there was little gain moving beyond $K = 64$. Our proposed sampling method outperforms one based on non-maximal suppression (bbox) by a factor of two.

$\text{SVM}^{\text{struct}}$ on the larger set). All experiments were run on a Dual 3.2GHz Intel Xeon with 4GB RAM.

Summary of results: Fig. 4(a) shows that the runtime of our algorithm compares favorably to state-of-the-art specialized solvers for linear SVMs (*e.g.* Liblinear). It also achieves stronger final results by optimizing the structured loss directly [4]. We compare runtimes of cost-sensitive SVM methods in Fig. 4(b), and show that SVM-IS is at least one order of magnitued faster than the only publicly availible method, $\text{SVM}^{\text{struct}}$. It is also faster than two alternative algorithms that we implemented: Stochastic Gradient Descent with a Pegasos-like update [21] and the Online Passive Aggressive Update (PA-I) method from [6]. Finally, we perform a sweep over parameter K (see Fig. 4(c)) and show that SVM-IS converges faster for higher values of K. This is expected because the update for $K = 200$ takes roughly the same amount of time as for $K = 1$, and allows us to update weights for all classes (as opposed to weights for a single class $\bar{Y}_t$).

5.3. Analysis of Results

Online/sequential vs. batch algorithms: Methods that employ updates in batches the size of the training set are slower by an asymptotic factor that scales with training set size. This includes the n-slack version of $\text{SVM}^{\text{struct}}$ (shown as a cyan curve in Fig. 4 and Fig. 3), which takes roughly 30 times longer to converge in the ImageNet experiment and 100-1000 times longer in the CUB-200-2011 experiment. The method based on mining hard negatives also processes the dataset in batch and is slow to make progress.

Single sample vs. multi sample: The multi-sample update reduces training time by a factor of approximately 5-50 compared to single sample online updates (PA-I [6] and SGD [19]) in the experiments that we have considered. This is to be expected as the level of improvement is dependent on structural properties of the problem: the more complex the output space is, the more important the importance sampling routine becomes.

Training with a single pass over the data: In Fig. 3a-b, we draw a vertical dotted red line indicating the time it takes to make one pass through the training set (where training roughly equals test time). We see that for training sets of size $n = 400$ and $n = 5000$, test error is close to saturation point with only 1-2 passes through the training set. At the same time, training error has not fully converged yet (See Fig. 3c). This is somewhat surprising but not theoretically unexpected [19, 21, 15] (see supplementary material).

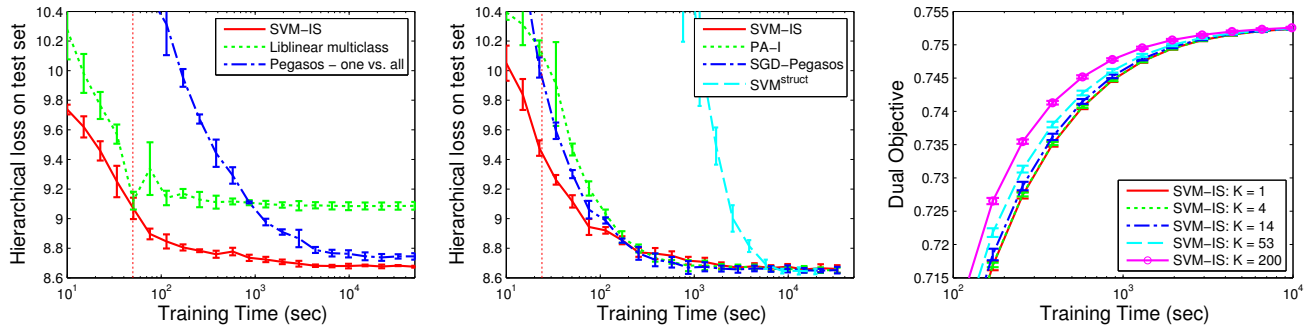
Memory usage: An additional benefit of online/sequential algorithms is that only one training example needs to be loaded into memory at the same time. We have implemented our method such that it can support datasets that are larger than the size of memory.

Drawbacks/Weaknesses: First, our method is specific to SVM-based learning algorithms (*e.g.*, it excludes potentially faster methods based on boosting or random forests). Secondly, it is less easily parallelizable than some methods based on binary classification (for example for 1-vs-all classifiers, one can train each binary classifier in parallel).

6. Conclusion

We introduced a fast structured SVM solver that is shown to be significantly faster than existing methods based

[4] Jia *et al.* [8] used a method for making a cost insensitive linear SVM cost sensitive by estimating posterior probabilities from the svm decision values. Our experience with this heuristic is that it was cumbersome, time-consuming, and unreliable, and we do not apply this step



(a) Comparison to popular fast linear SVM-solvers (b) Other methods for cost-sensitive SVMs (c) Effect of Parameter K

Figure 4. **Results on ImageNet:** (a) Our method converges at least as quickly as existing specialized solvers for linear SVMs, while incorporation of cost-sensitive learning allows us to obtain lower hierarchical loss. (b) We implemented cost-sensitive versions of Stochastic Gradient Descent with a Pegasos-like update [21] and the Online Passive Aggressive Update (PA-I) method from [6]. The proposed method converges noticeably faster than [21, 6], and is a full order of magnitude faster than the publicaly availible $\text{SVM}^{\text{struct}}$. (c) A parameter sweep of the K parameter of the importance sampling routine illustrate that it's best to update model parameters for *all* classes (K=200).

on $\text{SVM}^{\text{struct}}$, mining hard negatives, or online/sequential updates. It reduces train time by a factor of 20-1000 for cost-sensitive multiclass learning and deformable part model training on Imagenet and CUB-200-2011. In future work, we plan on running more extensive object detection experiments. We would also like to apply our algorithm to larger datasets, stronger alignment models, and other types of problems like tracking, segmentation, and attribute vocabulary learning.

Acknowledgments

The authors thank Kai Wang and Catherine Wah for helpful discussions and feedback. Funding for this work was provided by NSF Computer Vision Coral Ecology grant #ATM-0941760 and the Amazon AWS in Education program.

References

[1] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich. Diverse m-best solutions in markov random fields. In *ECCV*. 2012.

[2] M. Blaschko and C. Lampert. Learning to localize objects with structured output regression. *ECCV*, 2008.

[3] L. Bourdev and J. Malik. Poselets: Body part detectors using 3d human pose annotations. In *ICCV*, 2010.

[4] S. Branson, P. Perona, and S. Belongie. Strong supervision from weak annotation. In *ICCV*, 2011.

[5] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM*, 2004.

[6] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algs. *JMLR*, 2006.

[7] N. Dalal and B. Triggs. Hog for detection. In *CVPR*, 2005.

[8] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 categories? In *ECCV*, 2010.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale database. In *CVPR*, 2009.

[10] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. In *ICCV*, 2009.

[11] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear. *JMLR*, 2008.

[12] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, DPM. In *CVPR*, 2008.

[13] A. Guzman-Rivera, P. Kohli, and D. Batra. Faster training of structural svms with diverse m-best cutting-planes. 2013.

[14] T. Joachims. Training linear SVMs in linear time. In *SIGKD*, 2006.

[15] S. Kakade and S. Shalev-Shwartz. Mind the duality gap: Logarithmic regret algorithms. *NIPS*, 2008.

[16] S. Keerthi, S. Sundararajan, K. Chang, C. Hsieh, and C. Lin. A sequential dual method for svms. In *SIGKDD*, 2008.

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching. In *CVPR*, 2006.

[18] D. Park and D. Ramanan. N-best maximal decoders for part models. In *ICCV*, 2011.

[19] N. Ratliff. (Online) Subgradient Methods for Structured Prediction. In *AI*, 2007.

[20] B. Sapp, C. Jordan, and B. Taskar. Adaptive pose priors for pictorial structures. In *CVPR*, 2010.

[21] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient. In *ICML*, 2007.

[22] S. Shalev-Shwartz and T. Zhang. Stochastic Dual Coordinate Ascent Methods. *JMLR*, 2013.

[23] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, volume 16, page 25. MIT Press, 2004.

[24] B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction via the extragradient method. 2005.

[25] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured output. *JMLR*, 2006.

[26] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, Caltech, 2011.

[27] Y. Yang and D. Ramanan. Articulated Pose Estimation using Flexible Mixtures of Parts. In *CVPR*, 2011.