Learning Structured Models for Segmentation of 2D and 3D Imagery

Aurélien Lucchi[1] Pablo Márquez-Neila[2] Carlos Becker[2]
Yunpeng Li[2] Kevin Smith[3]
Graham Knott[4] Pascal Fua[2]
[1] Department of Computer Science, ETH, Zürich, Switzerland
[2] Computer Vision Laboratory, EPFL, Lausanne, Switzerland
[3]Biozentrum, University of Basel, Switzerland
[4] Interdisciplinary Center for Electron Microscopy, EPFL, Lausanne, Switzerland

***Index Terms*—Image processing, computer vision, electron microscopy, image segmentation, kernel methods, mitochondria, statistical machine learning, structured prediction, segmentation, superpixels, supervoxels.**

***Abstract*—Efficient and accurate segmentation of cellular structures in microscopic data is an essential task in medical imaging. Many state-of-the-art approaches to image segmentation use structured models whose parameters must be carefully chosen for optimal performance. A popular choice is to learn them using a large-margin framework and more specifically structured support vector machines (SSVM). Although SSVMs are appealing, they suffer from certain limitations. First, they are restricted in practice to linear kernels because the more powerful non-linear kernels cause the learning to become prohibitively expensive. Second, they require iteratively finding the most violated constraints, which is often intractable for the loopy graphical models used in image segmentation. This requires approximation that can lead to reduced quality of learning.**

In this article, we propose three novel techniques to overcome these limitations. We first introduce a method to "kernelize" the features so that a linear SSVM framework can leverage the power of non-linear kernels without incurring much additional computational cost. Moreover, we employ a working set of constraints to increase the reliability of approximate subgradient methods and introduce a new way to select a suitable step size at each iteration.

We demonstrate the strength of our approach on both 2D and 3D electron microscopic (EM) image data and show consistent performance improvement over state-of-the-art approaches.

I. Introduction

Semantic segmentation of 2D images and 3D image stacks is a fundamental medical image processing task. Graphical models, such as conditional random fields (CRF) [6], [25] are widely used for this purpose because they capture the inter-dependency between nearby pixels by minimizing an energy function, or equivalently maximizing a score function, which depends on both local data evidence and spatial consistency of the output. These models, however, typically involve a large number of parameters that must be carefully chosen to achieve good performance and learning them efficiently is a challenging task. It is particularly daunting when dealing with large datasets such as the one depicted by Fig. 1, which modern imaging devices can now routinely produce.

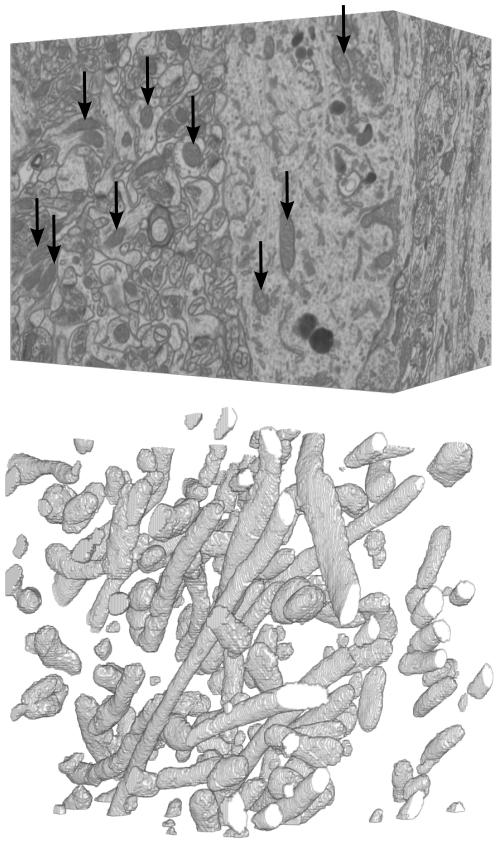


Fig. 1. A nearly isotropic stack of neural tissue acquired from the CA1 hippocampus using EM microscopy. **Top.** This stack is made of 1065 2048×1536 slices and its resolution is 5 nanometer in all three directions. This represents more than 3 billion voxels for a volume whose largest dimension is in the order of 10 μm. The black arrows point towards mitochondria, which we use to train our CRF-based segmentation algorithm. **Bottom.** 3D rendering of the mitochondria found in a $1024 \times 1024 \times 1024$ subvolume.

The maximum-margin framework [52] has gained much popularity in recent years as a means to do this. As an

This work was supported in part by the EU ERC Grant MicroNano.

alternative to maximum likelihood and maximum *a posteriori* learning, it makes unnecessary the difficult task of estimating the partition function and can be used in conjunction with many different performance metrics. The structured support vector machine (SSVM) [55] is a particularly successful variant of this approach, in which the learning objective is to minimize a regularized hinge loss caused by the violation of a set of soft margin constraints.

Although SSVMs have made the learning task easier than before, they are, nevertheless, not without their limitations. Though highly efficient when the CRF energy function is linear, they can become prohibitively expensive when non-linear kernels are involved, especially in the case of kernels computed over the graph nodes. This is because the number of kernel evaluations required, and hence the training time, scales quadratically with respect to the number of pixels or nodes in the CRF. This quickly makes the problem unmanageable for even moderate size datasets. Thus, it is usually impractical to directly apply non-linear kernels in an SSVM framework, even though they are often more powerful than their linear counterpart.

Moreover, optimizing the SSVM objective function can be challenging regardless of the linearity of the CRF energy. It is usually done iteratively using either the SSVM cutting plane algorithm [55] or by solving the equivalent unconstrained optimization problem using subgradient based methods [40], [32], [35], [58]. Either way, the *most violated constraint* must be found at every iteration. This means finding the labeling that maximizes the margin-sensitive hinge loss [55] and yields a valid cutting plane or true subgradient of the objective. This, however, is intractable for the loopy graphical models used for image segmentation. Although approximate maximizers obtained by approximate inference, such as belief propagation [34] and graph cuts [7] can be used as substitutes, the approximation are usually so imprecise that they adversely impact the learning process. In the worst case, an unsatisfactory constraint can cause the cutting plane algorithm to prematurely terminate if it does not have a higher hinge loss than previous constraints. It can also induce erratic behavior in subgradient-based methods when the implied descent direction is too far away from any true subgradient.

Another difficulty with subgradient-based methods is that they rely heavily on choosing an appropriate *step size*, that is, the distance by which to advance in the negative subgradient direction at each iteration. Many algorithms rely on preset, non-adaptive, and usually decreasing sequences of values. In theory, they provide theoretical asymptotic convergence guarantees. However, in practice, the inflexible nature of fixed-sequence step sizes often causes unsuitable values to be used, which negatively impacts the quality of the solutions obtained in finite time.

These phenomena make the learning process more susceptible to failure and limit the predictive power of the model. In this work, we develop and combine three new techniques to overcome these limitations.

- **Kernelized Features.** We formulate a two-stage training procedure that lets us combine the strength of SSVM and non-linear kernels in an efficient manner. To this end, we first use a regular non-linear non-structured SVM to create a set of support vectors from the training data. We then use it to transform our initial feature vectors into *kernelized features*, that is, the kernel product of the input feature vectors and the support vectors. We then train a linear SSVM using the kernelized features instead of the original ones, which lets us leverage the power of non-linear kernels without the computational burden of non-linear SSVM training.
- **Working Set Approach to Computing Subgradients.** We introduce an approximate subgradient method that uses a working set of constraints to make learning more robust. It is specifically designed to minimize the margin-sensitive hinge loss in the SSVM formulation when the most violated constraints and hence the resulting subgradients are not exact. We use a whole *working set of constraints*, unlike existing subgradient approaches that only consider the most recent one. This makes our method more reliable even when the subgradients estimated from individual constraints are noisy. In cases where fast training is important, it also allows us to estimate the subgradients by randomly sampling labelings instead of explicitly looking for the most violated constraints. This speeds up training at only a small loss in classification accuracy.
- **Adaptive Step Size Selection.** We propose a new approach to setting the step sizes adaptively at each iteration by exploiting knowledge about the objective function that we have accumulated during earlier iterations. More specifically, we use the working set of previously found constraints to compute the next step size. Our key observation is that previous constraints can be used to calculate a lower bound on the learning objective, which is informative for guiding the optimization. Our method preserves the asymptotic convergence properties of the sub-gradient descent (SGD), has a low computational overhead, and consistently yields improved solutions.

We introduced the first two techniques separately in [29] and [28] and the third one is new. We show that these three techniques can be combined and demonstrate that the resulting approach boosts CRF performance on both 2D and 3D datasets.

The remainder of the article is organized as follows. We discuss prior work in Section II and provide the background on the large-margin framework and corresponding learning techniques in Section III. Our three contributions are introduced in the three following sections, our kernelized features in Section IV, our working set approach in Section V and our adaptive step size selection in Section VI. We present our experimental results in Section VII.

II. Related work

For tasks such as segmentation, the labels of neighboring pixels tend to be highly correlated and modeling such correlations is often key to achieving high accuracy. As a result, structured prediction has emerged as a highly useful framework for tackling this class of problems. In this section,

we first briefly review existing versions of this framework and then discuss current approaches to learn their parameters and adjusting step-sizes during the optimization.

A. Structured Prediction and Non-linear Kernels

Structured prediction methods such as conditional random fields (CRFs) [25] have been widely applied to problems with structured outputs. While traditional classifiers, such as decision trees and SVMs independently map each data instance to a single label, structured prediction methods take into account the correlations between labels. This is critical for tasks such as image segmentation, where such correlations are strong between nearby pixels. Learning CRF models using large-margin methods has rapidly gained popularity in recent years, largely due to the fact that they are more objective-driven and do not involve the daunting partition function that can render maximum-likelihood approaches intractable in CRFs with loopy graph structures. Compared with earlier approaches including the max-margin Markov network [52], the structured support vector machine (SSVM) [55] is especially appealing because of its expressiveness and flexibility, and has since been successfully applied to many computer vision tasks, such as in [50], [36], [27], among many others.

SSVMs, however, require the CRF energy function to be expressible in a linear form, which in turn places the same restriction on all the unary and spatial terms due to the additive nature of CRF energies. While, in principle, the linear function can be defined in some high dimensional or possibly infinite-dimensional space, reproducing this kernel Hilbert space through the use of non-linear kernels is often infeasible in practice given that the number of kernel evaluations grows quadratically with the model size and must be optimized in the dual space. Though SSVM learning techniques based on sampled cuts [59], [44] have alleviated this problem to some extent, they sacrifice performance for speed. Even with this trade-off, they are still much slower than linear SSVMs [59]. Moreover, earlier implementations of these techniques were intended for use in conjunction with the cutting-plane method to speed up training of regular non-linear SVMs. This results in a *multivariate* output space in the SSVM formulation, which is analogous to a CRF without edges, and thus considerably simpler than the models typically used for image segmentation.

Kernel Approximation is another way to improve SSVM training efficiency by seeking a lower, finite-dimensional representation of the kernel-induced feature map that lies in a higher or infinite dimensional space. This can be achieved by random sampling from the typically infinite-dimensional feature map [39], [3] whose analytical form can be obtained using Fourier analysis when the kernel is homogeneous or stationary [56]. While promising results have been shown for specific additive kernels [31], [56], it is less clear how this approach generalizes to non-additive kernels, such as the Gaussian RBF that is more difficult to approximate. Alternatively, the *locally* linear SVM [24] can be used to simulate a non-linear decision boundary. However, this does not yield a *globally* linear function and, thus, does not fit into the SSVM framework. Moreover kernel approximation typically introduces additional tuning parameters such as the number of samples, which often present a performance-speed trade-off for which there are no well-defined tuning criteria.

Finally, it is worth pointing out the difference between our approach and the recently proposed *structural kernels*, which also perform structured prediction using non-linear kernels, but in a very different setting. In [44], [4], the kernels are defined on the overall output space, that is, the entire CRF, to exploit image-level "structural" information such as shape and color. While this serves to bias local labels and is useful for segmenting large dominant objects from the background, it often requires training data that completely characterizes the possible object configurations, such as binary masks. Multiple objects or objects whose pose are not represented in the trained model will cause the approach to fail. Our approach, on the other hand, uses regular kernels defined as products between a set of support vectors and the feature vectors extracted from individual nodes. This has the effect of making it more "local", and thus not susceptible to such failures.

B. Learning methods for structured models

Maximum margin learning of CRFs was first formulated in the max-margin Markov networks (M^3N) [52], whose objective is to minimize a margin-sensitive hinge loss between the ground-truth labeling and all other labelings for each training example. This is especially appealing for learning CRFs with loopy structures, due to its more objective-driven nature and the fact that it completely bypasses the partition function, which presents a major challenge to maximum likelihood based approaches. Nevertheless, the number of constraints in the resulting quadratic program (QP) is exponential in the size of the graph, making the optimization a non-trivial problem. In M^3N this is handled by rewriting the QP dual in terms of a polynomial number of marginal variables, which can then be solved by a coordinate descent method analogous to the sequential minimal optimization (SMO) [37]. However, solving such a QP is not tractable for loopy CRFs with high tree widths that are often needed in many computer vision tasks. In fact, even solving it approximately can become overwhelmingly expensive on large graphs.

Structured SVMs (SSVM) [55] optimize the same kind of objective as M^3N, while allowing for a more general class of loss functions. It employs a cutting plane algorithm to iteratively solve a series of increasingly larger QPs, which makes learning more scalable. However, the cutting plane algorithm requires the computation of the most violated constraints, namely the labeling that maximizes the hinge loss [55]. This involves performing the *loss augmented inference* [51], which is intractable on loopy CRFs. Though approximate constraints can be used [14], they make the cutting plane algorithm susceptible to premature termination and, in the worst case, can lead to catastrophic failure. This problem is considered in [50], which limits the optimization to a smaller parameter space where exact inference can be performed with graph cuts. However the addition of the hard submodularity constraints makes the resulting QP more expensive to solve, especially when the dimensionality of the feature space is also high.

In order to speed up the computation, a caching strategy was proposed in [19]. Instead of performing the loss augmented inference at every iteration, the algorithm first tries to construct a sufficiently violated constraint from the cache. While its use of a cache bears certain apparent similarity to the working set method described in this article, the goal of caching in [19] is to decrease the number of calls to the separation oracle while our approach aims at increasing the robustness of approximate subgradient methods.

An alternative to solving the quadratic program deterministically is to employ a stochastic gradient or subgradient method. This class of algorithms has been studied extensively for non-structured prediction problems [46]. In the context of structured prediction, learning can be achieved by finding a convex-concave saddle-point and solving it with a dual extra-gradient method [53]. In [40] max-margin learning is solved as an unconstrained optimization problem and subgradients are used to approximate the gradient in the resulting non-differentiable problem. This method trades optimality for a lower complexity, making it more suitable for large-scale problems. The approach of [32] proposes a perceptron-like algorithm based on an update whose expectation is close to the gradient of the true expected loss. However, the soundness of these methods heavily depends on the assumption that a valid subgradient is obtained at each iteration. Hence they become much less reliable when the subgradients are noisy due to inexact inference, as is the case for loopy CRFs.

The recently proposed SampleRank [57] avoids performing inference altogether during learning. Instead, it samples labelings at random using Markov chain Monte Carlo (MCMC). At each step, parameters are updated with respect to a pair of sampled labelings. Hence, unlike our method, it solves a problem that is substantially different from the original max-margin formulation. Though achieving notable speed improvement, the method does not in fact optimize the actual hinge loss but rather a loose upper bound on it.

C. Adaptive step-size

The performance of SGD largely depends on the ability to choose appropriate step sizes. They often have to be carefully tuned for the specific data at hand because inappropriate choices can easily lead to inferior solutions and slow convergence.

Several strategies for automatically choosing them are discussed in [5]. Most methods use preset step size sequences that are determined *offline* and are not adaptive [42], [46]. Typical choices include simple diminishing sequences and Polyak step sizes [38], the latter being suitable only when the optimal function value is known or can be estimated, and the evaluation of the objective function is possible and *easy*.

Nevertheless, a number of adaptive or "online" schemes have been studied in the literature and typically involve a line search to decide how far to move along a given descent direction. Exact searches are usually expensive and tend to be replaced by approximate searches methods based on Wolfe's sufficient conditions for convergence, where step sizes depend on the current point and the current search direction. Gain adaptation methods such as stochastic meta-descent (SMD) also accelerate convergence by using second-order information to adjust the gradient step sizes [43], but require being able to efficiently compute the Hessian. Another online approach is the margin infused relaxed algorithm (MIRA) [10] that uses the notion of margin to update its solution. It attempts to keep the current solution and the solution at the next iteration as close as possible while fully satisfying the margin requirement.

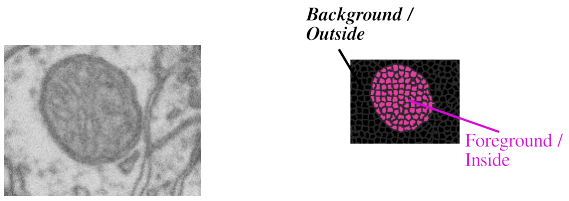
Choosing step sizes adaptively for subgradient based structured learning is also gaining attention. For example, the MIRA algorithm [10] was extended to structured outputs in [11]. Recently a stochastic coordinate ascent algorithm for solving structured SVM based on the Frank-Wolfe algorithm was developed in [23]. This approach automatically computes the step sizes in closed-form that corresponds to the result of line search in the dual space. The bundle approach to structured prediction [15], [26], [54] generalizes the cutting plane one by estimating a piecewise linear lower bound from cutting planes computed as objective function subgradients. A proximal function or a regularizer is commonly used to prevent over-large iteration steps. In fact, the bundle method of [54] generalizes SSVM [55] when the objective function includes a quadratic regularizer. Like both the cutting plane and bundle methods, ours exploits all the previously accumulated constraints. However, unlike these methods, it only requires a 1D line search in the subgradient direction, which can be efficiently done. Note that the soundness of most of the aforementioned subgradient methods depends heavily on the assumption that a valid subgradient is obtained at each iteration. Hence they become much less reliable when the subgradients are noisy due to inexact inference, as is the case for loopy CRFs. Our method is not exempt from this problem but we found empirically that using a working set increases the reliability of our subgradients.

III. Max-Margin Learning of CRFs

In this section, we begin by describing the standard CRF model [25], [49] for segmentation. We then discuss how to learn its parameters within the SSVM framework, with specific attention to expressing the CRF model in the required linear form for effective learning. We will use it in Sec. IV in conjunction with our kernelized features, thus enabling us to leverage the power of non-linear kernels while the SSVM remains linear. In Section V-B, we will introduce our approach to computing the subgradients required to learn its parameters.

A. CRF for Image Segmentation

As a standard preprocessing step, we first perform a preliminary over-segmentation of our input image into superpixels or supervoxels [1], such as those depicted by Fig. 2. The CRF $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is thus defined so that each node $i \in \mathcal{V}$ corresponds to a superpixel and there is an edge $(i, j) \in \mathcal{E}$ between two nodes i and j if the corresponding superpixels are adjacent in the image. Let $X = \{x_i\}_{i \in \mathcal{V}}$ be an input example, such as the image or features associated to it and $Y = \{y_i\}_{i \in \mathcal{V}}$ the corresponding labeling of the CRF which assigns a class label y_i to each node.



(a) Input image (b) Ground truth for the CRF

Fig. 2. The input image (a slice through a 3D volume) is shown in (a). Figure (b) shows the ground truth labeling of the CRF used in our method. The pink and black colors correspond to the foreground and background classes while the gray contours are the boundaries of the SLIC supervoxels.

The score function associated with the CRF can then be written as

$$S_{\mathbf{w}}(X,Y) = \sum_{i\in\mathcal{V}} D_i(x_i,y_i) + \sum_{(i,j)\in\mathcal{E}} V_{ij}(y_i,y_j), \quad (1)$$

where D_i is the unary data term and V_{ij} is the pairwise spatial term.

Both D_i and V_{ij} depend on the observed data and the CRF parameters $\mathbf{w}$, in addition to the labeling Y. The negated score is also commonly referred to as the "energy" in the literature and we will use the two terms interchangeably. The inferred optimal labeling is simply the one that minimizes it, that is,

$$Y^* = \arg\max_{Y\in\mathcal{Y}} S_{\mathbf{w}}(X,Y)\ , \quad (2)$$

where $\mathcal{Y}$ denotes the set of all possible labelings. In the remainder of this section, we will omit the input X, a constant with respect to the maximization in Eq. 2, from the notation $S_{\mathbf{w}}(X,Y)$ for the sake of conciseness. While the exact minimization of the score function is generally intractable on loopy CRFs, good approximate solutions can be found efficiently using techniques such as graph cuts [9] and belief propagation [34]. In our case, we use graph cuts when the score function is submodular [22] and belief propagation otherwise.

B. Discriminative Learning

Discriminative learning uses the labeled training data to learn the CRF parameters so that the inferred labeling of the CRF is "close" to that of the ground truth, defined as yielding a low *loss*. More specifically, given a set of N training examples $\mathcal{D} = ((X^1,Y^1),\ldots,(X^N,Y^N))$ where $X^i \in \mathcal{X}$ is an input example and $Y^i \in \mathcal{Y}$ is the associated labeling, the learning task consists in finding model parameters $\mathbf{w}$ that achieve low empirical loss subject to some regularization. In other words, we seek

$$\begin{aligned}\mathbf{w}^* &= \arg\min_{\mathbf{w}} \mathcal{L}(\mathcal{D},\mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{(X^n,Y^n)\in\mathcal{D}} l(X^n,Y^n,\mathbf{w}) + R(\mathbf{w}), \quad (3)\end{aligned}$$

where $R(\mathbf{w})$ is the regularizer (such as the L2 norm of $\mathbf{w}$) that helps prevent overfitting and l is the surrogate loss function, a quantity that is usually related to and often defines an upper bound on the training error. Note that the definition of the surrogate loss l depends on the score function $S_{\mathbf{w}}$, since the goal of learning is to make the maximizer of $S_{\mathbf{w}}$ a desirable output for the given input. The most common choice of l is the hinge loss, as used in [52], [55] and defined as

$$l(Y^n,Y,\mathbf{w}) = [S_{\mathbf{w}}(Y) + \Delta(Y^n,Y) - S_{\mathbf{w}}(Y^n)]_+, \quad (4)$$

where $[v]_+ = \max(0,v)$ and the task loss Δ measures the closeness of any inferred labeling Y to the ground truth labeling Y^n.

C. Max-margin Formulation

The max-margin approach is a specific instance of discriminative learning, where parameter learning is formulated as a quadratic program (QP) with soft margin constraints [55]

$$\begin{aligned}&\min_{\mathbf{w},\boldsymbol{\xi}\geq\mathbf{0}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_{n=1}^{N}\xi_n \quad (5)\\ &\text{s.t. } \forall n : S_{\mathbf{w}}(Y^n) \geq \max_{Y\in\mathcal{Y}_n}(S_{\mathbf{w}}(Y) + \Delta(Y^n,Y)) - \xi_n,\end{aligned}$$

where $\mathcal{Y}_n$ is the set of all possible labelings for example n, the constant C controls the trade-off between margin and training error.

The QP can be converted to an unconstrained optimization problem by incorporating the soft constraints directly into the objective function, yielding

$$\begin{aligned}&\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \quad (6)\\ &\min_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_{n=1}^{N}[S_{\mathbf{w}}(Y^*) + \Delta(Y^n,Y^*) - S_{\mathbf{w}}(Y^n)]_+,\end{aligned}$$

where

$$Y^* = \arg\max_{Y\in\mathcal{Y}_n}(S_{\mathbf{w}}(Y) + \Delta(Y^n,Y)) \quad (7)$$

is the most violated constraint of all possible labelings for example n given current parameters $\mathbf{w}$.

Finding the optimal labeling Y^* as described in Eq. 7 is a key challenge named *loss-augmented inference*.

D. Linearizing the CRF Score Function

Since the SSVM operates by solving a quadratic program (QP), all the constraints in Eq. 5 must be linear [55]. This requires that the score function $S_{\mathbf{w}}$ be expressible as an inner product between the parameter vector and a feature map. Since the score is the sum of individual unary and pairwise terms, this implies that D_i and V_{ij} also must be expressible as

$$D_i(y_i) = \left\langle \mathbf{w}^D, \psi_i^D(y_i)\right\rangle \quad (8)$$

and

$$V_{ij}(y_i,y_j) = \left\langle \mathbf{w}^V, \psi_{ij}^V(y_i,y_j)\right\rangle, \quad (9)$$

where $\psi_i^D(y_i)$ and $\psi_{ij}^V(y_i,y_j)$ [1] are feature maps dependent on both the observed data and the labels, and where

$$\mathbf{w} = ((\mathbf{w}^D)^T,(\mathbf{w}^V)^T)^T \quad (10)$$

[1] Again, we omit the input x_i from the notation for conciseness since it is a constant.

is the vector of parameters that define the functions D_i and V_{ij} respectively.

If we let $\Psi^D(Y) = \sum_{i\in\mathcal{V}} \psi_i^D(y_i)$ and $\Psi^V(Y) = \sum_{(i,j)\in\mathcal{E}} \psi_{ij}^V(y_i, y_j)$, then $\Psi(Y) = (\Psi^D(Y)^T, \Psi^V(Y)^T)^T$ allowing the CRF score to now be written linearly as

$$S_{\mathbf{w}}(Y) = \langle \mathbf{w}, \Psi(Y) \rangle . \tag{11}$$

Let $\mathbf{x}_i$ be a feature vector associated with node i extracted from the observed data. We can define the data feature map as

$$\psi_i^D(y_i) = (I(y_i = 1)\mathbf{x}_i^T, \ldots, I(y_i = K)\mathbf{x}_i^T)^T, \tag{12}$$

where K is the number of possible labels, i.e., $y_i \in \{1, \ldots, K\}$. If we write $\mathbf{w}^D = ((\mathbf{w}_1^D)^T, \ldots, (\mathbf{w}_K^D)^T)^T$, the unary term becomes the inner product

$$D_i(y_i) = \langle \mathbf{w}_{y_i}^D, \mathbf{x}_i \rangle , \tag{13}$$

which represents the score of node i taking on label y_i. Similarly, if we define the pairwise feature map as

$$\psi_{ij}^V(y_i, y_j) = (I(y_i = a, y_j = b))_{(a,b)\in\{1,\ldots,K\}^2} \tag{14}$$

where $I(a)$ is the indicator function that takes value 1 if the predicate a is true and 0 otherwise. Given the corresponding parameters $\mathbf{w}^V = (w_{ab})_{(a,b)\in\{1,\ldots,K\}^2}$, the pairwise term

$$V_{ij}(y_i, y_j) = w_{y_i y_j} \tag{15}$$

reflects the transition cost between nodes i and j from label y_i to label y_j. Although the above definition depends only on the labels y_i and y_j, the pairwise term can, in fact, be made data-aware, as in [48], [27]. For instance, it can be made gradient-adaptive by including parameters for each discretized gradient level, as this is the case for the experiments presented in Section. VII.

IV. Kernel-transformed Features

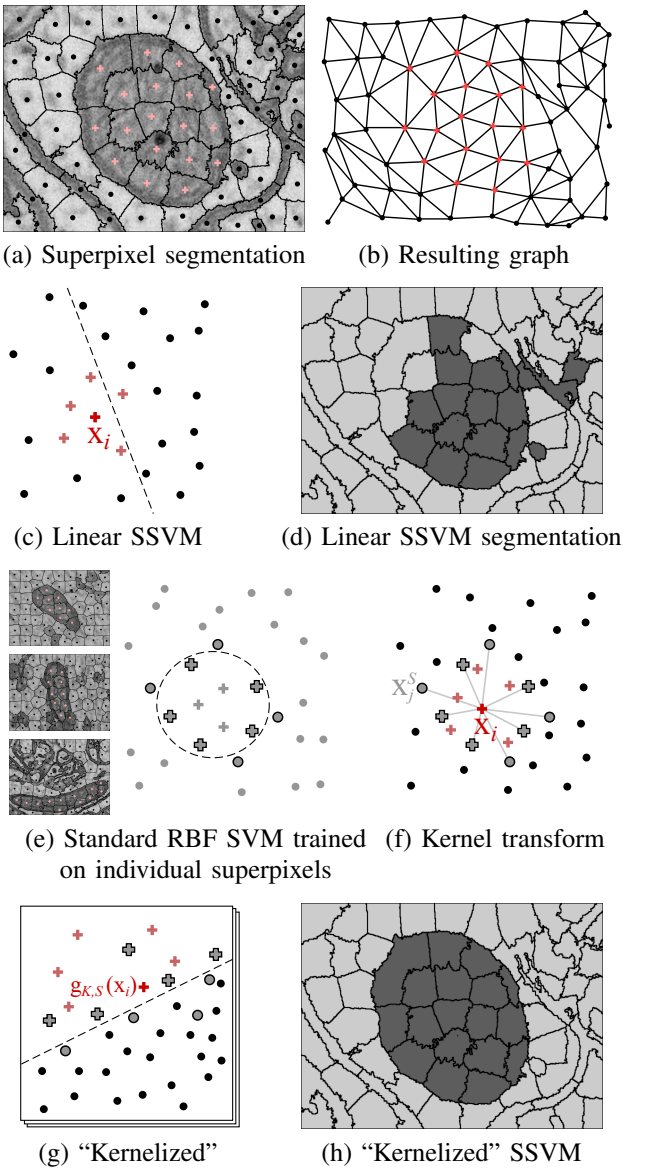
As discussed above, standard SSVMs require a score function that is linear in both parameters and features. This constitutes a major limitation, since unary terms based on non-linear SVMs are often more powerful and produce better results [16], [27]. It should be possible, in principle, to learn non-linear unary terms within the SSVM framework by implicitly defining $\mathbf{w}$ and $\mathbf{x}_i$ of Eq. 13 in a high-dimensional space through kernels. However, since computations have to be done in the dual space, this would involve quadratic number of kernel products to compute the inner products making such an approach prohibitively expensive for large CRFs.

Our approach aiming at circumventing this problem is illustrated on Fig. 3. It starts from the observation that a non-linear binary (+1/-1 label) SVM classifier always takes the form

$$\text{Score}(\mathbf{x}) = \sum_j \alpha_j y_j^S K(\mathbf{x}_j^S, \mathbf{x}) , \tag{16}$$

where $\mathbf{x}_j^S \in S$ are the support vectors with corresponding labels y_j^S. Extended to multi-class labels ($y_i \in \{1, \ldots, K\}$) for a general unary term, it becomes

$$D_i(y_i) = \sum_j \alpha_{y_i,j} c(y_j^S, y_i) K(\mathbf{x}_j^S, \mathbf{x}_i) , \tag{17}$$



(a) Superpixel segmentation (b) Resulting graph

(c) Linear SSVM (d) Linear SSVM segmentation

(e) Standard RBF SVM trained on individual superpixels (f) Kernel transform

(g) "Kernelized" (h) "Kernelized" SSVM

Fig. 3. Kernelized features. *(a)* A superpixel over-segmentation of an image. The + and • in the middle of each denotes either foreground or background. *(b)* In the graph used to construct the CRF, each node corresponds to a superpixel and each and edge indicate adjacency in the image. In 3D biomedical volumes, the superpixels become supervoxels and graph becomes a 3D grid. *(c)* An illustration of the feature space. Each point represents a feature vector extracted from a superpixel. Because it is not linearly separable, the standard SSVM gives a poor segmentation result in *(d)*. *(e)* To address this, we train a non-structured kernel SVM on individual superpixels to obtain a set of support vectors, indicated by outlined points. *(f)* Kernel-transformed features $\mathbf{g}_{K,S}(\mathbf{x}_i)$ are obtained for each feature vector x_i from the kernel products of x_i and the support vectors. *(g)* Data in the $|S|$-dimensional "kernelized" feature space is linearly separable, and can be used to train a linear SSVM. *(h)* The improved segmentation result.

where $c(y_j^S, y_i)$ is 1 if $y_i = y_j^S$ and -1 otherwise. Note that, although the function is non-linear in the input features $\mathbf{x}_i$, because of the non-linear kernel K, it is linear in the kernel products $K(\mathbf{x}_j^S, \mathbf{x}_i)$.

If we define $\mathbf{g}_{K,S}(\mathbf{x}_i)$ as the vector of kernel products

$$\mathbf{g}_{K,S}(\mathbf{x}_i) = (K(\mathbf{x}_1^S, \mathbf{x}_i), \ldots, K(\mathbf{x}_{|S|}^S, \mathbf{x}_i))^T, \tag{18}$$

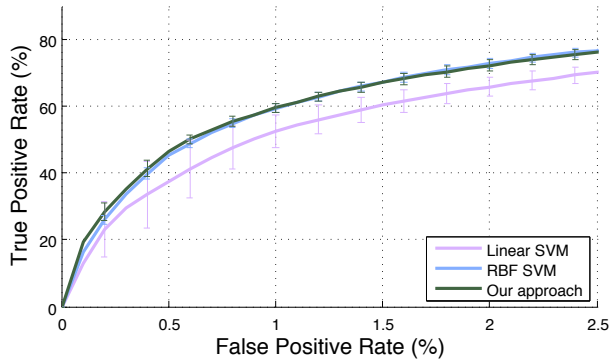


Fig. 4. Comparison of a linear SVM, RBF-SVM, and a linear SVM trained on the feature vectors described in Section VII-C and kernelized using the support vectors of an RBF-SVM. For classification of individual samples (ignoring structure), training a linear SVM on kernelized feature vectors yields a similar performance to a standard RBF-SVM. Error bars indicate standard deviation over 10 experiments on the hippocampus dataset with different samples randomly drawn from the training set.

and $\mathbf{w'}_{y_i}$ as their coefficients

$$\mathbf{w'}_{y_i} = (\alpha_{y_i,1} c(y_1^S, y_i), \cdots, \alpha_{y_i,|S|} c(y_{|S|}^S, y_i))^T, \quad (19)$$

then the unary term can be re-expressed as

$$D_i(y_i) = \langle \mathbf{w'}_{y_i}, \mathbf{g}_{K,S}(\mathbf{x}_i) \rangle, \quad (20)$$

which is of the finite-dimensional linear form needed for learning within the SSVM framework, as discussed in the previous section.

This suggests a simple 2-step learning approach to incorporating kernels into an SSVM, illustrated in Fig. 3. First, we train a standard non-structured, non-linear kernel SVM to obtain a set of support vectors. These vectors are then used to create a set of *kernel-transformed*, or "kernelized", feature vectors $\mathbf{g}_{K,S}(\mathbf{x}_i)$, which are used to train the linear SSVM.

Although our formulation is not equivalent to a non-linear kernel SSVM[2], nor does it intend to approximate a non-linear SSVM as kernel approximation methods do [39], [31], [56], it does produce models with the same functional form as those learned using a kernel SSVM. Most importantly, the resulting models perform well in practice as we will show later.

To demonstrate the principle that a linear SVM trained using kernel-transformed features performs similarly to a non-linear SVM that uses the same kernel, we conducted a simple proof-of-concept experiment for a classification task with a linear SVM classifying each sample independently while ignoring structure. We compare the performance of a standard linear SVM, an SVM trained with an RBF kernel, and a simplification of our approach in which kernelized feature vectors – obtained using the support vectors of the RBF-SVM – are used to train a standard linear SVM. The results in Fig. 4 support our intuition: So long as we transform the original feature vector using the right set of support vectors, learning the new coefficients under a different objective function (i.e., as primal instead of dual variables) yields performance similar to a non-linear kernel SVM.

[2]The parameters $\mathbf{w}'$ now correspond to the primal variables of a linear SSVM instead of the dual variables of a non-linear SSVM.

V. Working Sets

We begin with a review of the stochastic subgradient method in the context of structured learning. We then introduce a novel technique to make the stochastic subgradient method more robust to approximation errors in the computation of subgradients using working sets of previously computed constraints.

A. The Stochastic Subgradient Method

The objective function of Eq. 6 can be minimized via a stochastic subgradient method [40], [32]. This class of methods iteratively computes and steps in the opposite direction of a subgradient vector with respect to an example (X^n, Y^n) chosen by picking an index $n \in \{1 \dots N\}$ uniformly at random. This implies finding, at each step, the subgradient of the function

$$f(Y^n, Y^*, \mathbf{w}) = l(Y^n, Y^*, \mathbf{w}) + \frac{\lambda}{2} ||\mathbf{w}||^2. \quad (21)$$

A subgradient of the convex function $f : \mathcal{W} \to \mathbb{R}$ at $\mathbf{w}$ is defined as a vector $\mathbf{g}$, such that

$$\forall \mathbf{w}' \in \mathcal{W}, \mathbf{g}^T(\mathbf{w}' - \mathbf{w}) \leq f(\mathbf{w}') - f(\mathbf{w}). \quad (22)$$

The set of all subgradients at $\mathbf{w}$ is called the *subdifferential* at $\mathbf{w}$ and is denoted $\partial f(\mathbf{w})$. The subdifferential is always a non-empty convex compact set.

For the hinge loss, a valid subgradient $\mathbf{g}(Y^n, Y^*, \mathbf{w})$ with respect to the parameter $\mathbf{w}$ can always be computed as:

$$\frac{\partial f(Y^n, Y^*, \mathbf{w})}{\partial \mathbf{w}} = \psi(Y^*) - \psi(Y^n) + \lambda \mathbf{w}. \quad (23)$$

This results in a simple algorithm that iteratively computes and steps in the direction of the negative subgradient as described in Algorithm 1. In order to guarantee convergence, the step size $\eta^{(t)}$ needs to satisfy the following conditions:

$$\lim_{T \to +\infty} \sum_{t=1}^{T} \eta^{(t)} = \infty \quad \text{and} \lim_{T \to +\infty} \sum_{t=1}^{T} (\eta^{(t)})^2 < \infty. \quad (24)$$

Algorithm 1 SGD + inference

INPUTS :
  $\mathcal{D}$: Training set of N examples.
  β : Learning rate parameter.
  $\mathbf{w}^{(1)}$: Arbitrary initial values, e.g., 0.
OUTPUT : $\mathbf{w}^{(T+1)}$
for $t = 1 \dots T$ **do**
  Pick some example (X^n, Y^n) from $\mathcal{D}$
  $Y^* = \arg\max_{Y \in \mathcal{Y}_n} (S_{\mathbf{w}}(Y) + \Delta(Y^n, Y))$
  $\eta^{(t)} \leftarrow \frac{\beta}{t}$
  $\mathbf{g}^{(t)} \leftarrow \frac{\partial f(Y^n, Y, \mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}}$
  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}^{(t)}$
end for

For loopy CRFs, however, true subgradients of the hinge loss cannot always be obtained due to the intractability of loss-augmented inference, namely finding the Y^* for each example. This can lead to erratic behavior due to noisy subgradient estimates and loss of performance.

B. The Subgradient Method With Working Sets

Our algorithm aims at increasing the reliability of subgradient methods by using working sets of constraints, denoted $\mathcal{A}^n$, for learning loopy CRFs where exact inference is intractable.

We first outline a batch version of our method in Algorithm 2 that uses subgradients computed as an average over the working sets of constraints. Later on, we will present a more efficient variant of this algorithm that uses atomic updates for every constraint in the working set. The batch version first solves the loss-augmented inference to find a constraint Y^* and add it to the working set $\mathcal{A}^n$. It then steps in the opposite direction of the approximate subgradient computed as an average over the set of violated constraints belonging to $\mathcal{A}^n$.

Algorithm 2 Working sets + inference – batch version

INPUTS :
  $\mathcal{D}$: Training set of N examples.
  β : Learning rate parameter.
  $\mathbf{w}^{(1)}$: Arbitrary initial values, e.g., 0.
OUTPUT : $\mathbf{w}^{(T+1)}$
Initialized $\mathcal{A}^n \leftarrow \emptyset$ for each $n = 1 \dots N$
for $t = 1 \dots T$ **do**
  Pick some example (X^n, Y^n) from $\mathcal{D}$
  $Y^* = \arg\max_{Y \in \mathcal{Y}_n} (S_{\mathbf{w}}(Y) + \Delta(Y^n, Y))$
  $\mathcal{A}^n \leftarrow \mathcal{A}^n \cup \{Y^*\}$
  $\mathcal{A}^{n'} \leftarrow \{Y \in \mathcal{A}^n \mid l(Y, Y^n, \mathbf{w}^{(t)}) > 0\}$
  $\eta^{(t)} \leftarrow \frac{\beta}{t}$
  $\mathbf{g}^{(t)} \leftarrow \frac{1}{|\mathcal{A}^{n'}|} \sum_{Y \in \mathcal{A}^{n'}} \frac{\partial f(Y^n, Y, \mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}}$
  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}^{(t)}$
end for

Hence unlike dual averaging methods [35], [58] that aggregate over all previous subgradients, our algorithm only considers the subset of active, namely violated, constraints when computing the parameter updates. Therefore all subgradients are computed with respect to the parameters at the *current* iteration, as opposed to using their historical values. We will show that this produces better results in the experiments reported in Section VII.

We now analyze the convergence properties of the algorithm presented in Algorithm 2. Although finding true subgradients as defined in Eq. 22 cannot be guaranteed for loopy CRFs, interesting results can still be obtained even if one can only find an approximate ϵ-subgradient $\mathbf{g}$, as defined in [47]:

$$\forall \mathbf{w}' : \mathbf{g}^T(\mathbf{w} - \mathbf{w}') \geq f(\mathbf{w}) - f(\mathbf{w}') - \epsilon \tag{25}$$

The convergence properties of ϵ-subgradient methods were studied in [41], [47], [40]. The "regret" or loss of the parameter vector $\mathbf{w}$ can be bounded by

$$\mathbb{E}\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|_2^2 \leq \frac{G^2}{\lambda^2 t} + \frac{\epsilon}{\lambda}, \tag{26}$$

where G is a constant satisfying the condition $||\mathbf{g}||^2 \leq G^2$ and $\lambda = \frac{1}{C}$. The re-derivation of this proof using our notations is given in the appendix.

Given that the choice of the step size satisfies Eq. 24, we can see that the first term on the right side of Eq. 26 goes to 0 so stochastic ϵ-subgradient methods converge to a certain distance ϵ to the optimum. The key to improving convergence is thus to obtain more accurate ϵ-subgradients, and we show below how this could be achieved through the use of working sets.

Let $\mathbf{g}_1, \dots, \mathbf{g}_m \in \mathbb{R}^d$ be the approximate subgradients of $\mathcal{L}$ with respect to example (X^n, Y^n), for the labelings in the working set $\mathcal{A}^n$ that still violates the margin constraint at a given iteration. Assume that each $\mathbf{g}_i \in \mathbb{R}^d$ comes from some distribution with mean $\boldsymbol{\mu}_i \in \partial\mathcal{L}(\mathbf{w})$ and bounded variance.

Let $\boldsymbol{\delta}_i = \mathbf{g}_i - \boldsymbol{\mu}_i$ be the difference between approximate ϵ-subgradient $\mathbf{g}_i$ and true ϵ-subgradient $\boldsymbol{\mu}_i$, and assume that all $\boldsymbol{\delta}_i$ are independent of one another. Note that, by definition, each $\boldsymbol{\delta}_i$ has zero expectation and hence their average $\bar{\boldsymbol{\delta}} = \frac{1}{m}\sum \boldsymbol{\delta}_i = \frac{1}{m}\sum \mathbf{g}_i - \frac{1}{m}\sum \boldsymbol{\mu}_i$.

Therefore, using Hoeffding's inequality [17] and the union bound, we can show that the average error $\bar{\boldsymbol{\delta}}$ concentrates around its expectation, i.e., 0 in this case, as the number of violated constraints in the working set m increases:

$$\Pr\left(||\bar{\boldsymbol{\delta}}|| \geq r\right) \leq 2d \exp\left(\frac{-mr^2}{2G^2}\right). \tag{27}$$

The convexity of the subdifferential $\partial\mathcal{L}(\mathbf{w})$ implies that $\bar{\boldsymbol{\mu}} = \frac{1}{m}\sum_i \boldsymbol{\mu}_i \in \partial\mathcal{L}(\mathbf{w})$. Therefore the probability of $\mathbf{g}^{(t)} \triangleq \frac{1}{m}\sum \mathbf{g}_i$ being more than a distance r away from any true subgradient is bounded by Eq. 27 as well.

Algorithm 3 Working sets + sampling

INPUTS :
  $\mathcal{D}$: Training set of N examples.
  $\mathcal{Q}$: MCMC walker.
  β : Learning rate parameter.
  $\mathbf{w}^{(1)}$: Arbitrary initial values, e.g., 0.
OUTPUT : $\mathbf{w}^{(T+1)}$
Initialized $\mathcal{A}^n \leftarrow \emptyset$ for each $n = 1 \dots N$
for $t = 1 \dots T$ **do**
  Pick some example (X^n, Y^n) from $\mathcal{D}$
  Sample Y^* according to $\mathcal{Q}(w^{(t)}, Y^n)$
  $\mathcal{A}^n \leftarrow \mathcal{A} \cup \{Y^*\}$
  $\mathcal{A}^{n'} \leftarrow \{Y \in \mathcal{A}^n \mid l(Y, Y^n, \mathbf{w}^{(t)}) > 0\}$
  $\mathbf{z} \leftarrow \mathbf{w}^{(t)}$
  for $Y \in \mathcal{A}^{n'}$ **do**
    $\mathbf{g}^{(t)} \leftarrow \frac{1}{|\mathcal{A}^{n'}|} \frac{\partial f(Y^n, Y, \mathbf{z})}{\partial \mathbf{w}^{(t)}}$
    $\eta^{(t)} \leftarrow \frac{\beta}{t}$ or $\eta^{(t)} \leftarrow$ Line_search$(\mathbf{g}^{(t)})$
    $\mathbf{z} \leftarrow \mathbf{z} - \eta^{(t)} \mathbf{g}^{(t)}$ * atomic update *
  end for
  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{z}$
end for

Given that the assumption of independence between all $\boldsymbol{\delta}_i$ might not always hold, we experimented with an adaptation of the batch version that replaces the standard update of Algorithm 2 with a sequence of atomic updates that has been shown to improve the rate of convergence [57]. We also introduce a second modification to Algorithm 2 where, instead

Algorithm 4 Working sets + inference

INPUTS :
  $\mathcal{D}$: Training set of N examples.
  $\mathbf{w}^{(1)}$: Arbitrary initial values, e.g., 0.
OUTPUT : $\mathbf{w}^{(T+1)}$
Initialized $\mathcal{A}^n \leftarrow \emptyset$ for each $n = 1 \dots N$
for $t = 1 \dots T$ **do**
  Pick some example (X^n, Y^n) from $\mathcal{D}$
  $Y^* = \arg\max_{Y \in \mathcal{Y}_n} (S_{\mathbf{w}}(Y) + \Delta(Y^n, Y))$
  $\mathcal{A}^n \leftarrow \mathcal{A} \cup \{Y^*\}$
  $\mathcal{A}^{n'} \leftarrow \{Y \in \mathcal{A}^n \mid l(Y, Y^n, \mathbf{w}^{(t)}) > 0\}$
  $\mathbf{z} \leftarrow \mathbf{w}^{(t)}$
  for $Y \in \mathcal{A}^{n'}$ **do**
    $\mathbf{g}^{(t)} \leftarrow \frac{1}{|\mathcal{A}^{n'}|} \frac{\partial f(Y^n, Y, \mathbf{z})}{\partial \mathbf{w}^{(t)}}$
    $\eta^{(t)} \leftarrow \frac{\beta}{t}$ or $\eta^{(t)} \leftarrow$ Line_search$(\mathbf{g}^{(t)})$
    $\mathbf{z} \leftarrow \mathbf{z} - \eta^{(t)} \mathbf{g}^{(t)}$ * atomic update *
  end for
  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{z}$
end for

of stepping by an arbitrarily fixed distance in the subgradient direction, we choose the step size using a line search so as to best satisfy all currently active constraints, as explained in Section VI. The resulting method is outlined in Algorithm 4.

In addition, the analysis presented in Section V-B suggests that it is possible to use a sampling method instead of the loss-augmented inference to obtain new constraints, and under similar assumptions the average subgradient $\bar{g}$ still converges to a valid subgradient. Based on this observation, we propose an adaptation of Algorithm 2 that uses sampling instead of solving the loss-augmented inference. This adaptation described in Algorithm 3 generates new constraints using an MCMC walker denoted $\mathcal{Q}$ similar to the one described in [57].

VI. Adaptive Step Size Selection

We aim at providing a systematic way to pick the step size by which to advance in the descent direction by minimizing a regularized loss. Instead of stepping by an arbitrarily fixed distance, we want to choose the step size $\eta^{(t)}$ that minimizes an objective function $h_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta)$ that depends also on the current value of the parameters $\mathbf{w}^{(t)}$ as well as the working set $\mathcal{A}^n$ that contains all the constraints for example n generated at the previous iterations, i.e.,

$$\eta^{(t)} \leftarrow \arg\min_{\eta} h_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta). \tag{28}$$

More specifically, we would like to choose $\eta^{(t)}$ to ensure that none of the previous constraints in the working set $\mathcal{A}^n$ are highly violated at time $t+1$, namely after the subgradient update $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}^{(t)}$. We thus define the function $h_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta)$ as

$$\begin{aligned} h_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta) &= \max_{Y \in \mathcal{A}^n} \; l(Y^n, Y, \mathbf{w}_\eta^{(t+1)}) + \frac{\lambda}{2} \left|\left| \mathbf{w}_\eta^{(t+1)} \right|\right|^2 \\ &= \max_{Y \in \mathcal{A}^n} \left[\left\langle \mathbf{w}_\eta^{(t+1)}, \delta\psi_n(Y) \right\rangle + \Delta(Y^n, Y) \right]_+ + \frac{\lambda}{2} \left|\left| \mathbf{w}_\eta^{(t+1)} \right|\right|^2 \end{aligned} \tag{29}$$

where $\mathbf{w}_\eta^{(t+1)}$ and $\delta\psi_n(Y)$ are shorthand notations:

$$\begin{aligned} \mathbf{w}_\eta^{(t+1)} &= \mathbf{w}^{(t)} - \eta \mathbf{g}^{(t)} \\ \delta\psi_n(Y) &= \psi(Y) - \psi(Y^n), \end{aligned} \tag{30}$$

recalling that vector $\psi(Y)$ is the feature map for labeling Y.

The intuition is that $h_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta)$ is a lower bound on the SSVM objective at $\mathbf{w}_\eta^{(t+1)}$ with respect to example n, i.e., $f(Y^n, Y^*_{\mathbf{w}}, \mathbf{w})$ (Eq. 21), and therefore an informative heuristic for the line search in the negative subgradient direction.

To simplify Eq. 29, we can remove the floor function $[\,\cdot\,]_+$ by adding an extra (trivial) constraint Y^n to $\mathcal{A}^n$ since by definition $l(Y^n, Y^n, \cdot) \equiv 0$ and $\delta\psi_n(Y^n) = \mathbf{0}$. Hence

$$\begin{aligned} &h_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta) \\ &= \max_{Y \in \mathcal{A}^n_+} \left\langle \mathbf{w}_\eta^{(t+1)}, \delta\psi_n(Y) \right\rangle + \Delta(Y^n, Y) + \frac{\lambda}{2} \left|\left| \mathbf{w}_\eta^{(t+1)} \right|\right|^2, \end{aligned} \tag{31}$$

where $\mathcal{A}^n_+ = \mathcal{A}^n \cup \{Y^n\}$.

Since h is not the same quantity as the SSVM objective function, minimizing h may occasionally produce excessively large step sizes, which impede convergence. To avoid this, we include a quadratic penalty on the magnitude by which $\mathbf{w}$ can move in space, that is, $\left|\left| \mathbf{w}_\eta^{(t+1)} - \mathbf{w}^{(t)} \right|\right| = \left|\left| \mathbf{g}^{(t)} \right|\right| \eta$. This term is also known as a prox-function (i.e., proximity control function) which prevents overly large steps in the iterates and is commonly used in machine learning [21], [10]. This gives us the regularized line search objective function

$$h^R_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta) = h_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta) + \rho^{(t)} \left|\left| \mathbf{g}^{(t)} \right|\right|^2 \eta^2, \tag{32}$$

where $\rho^{(t)}$ is a regularization constant. We choose $\rho^{(t)} = Ct$ where C is a constant value that can be determined by cross-validation. Finally we choose the step size $\eta^{(t)}$ that minimizes h^R, i.e.,

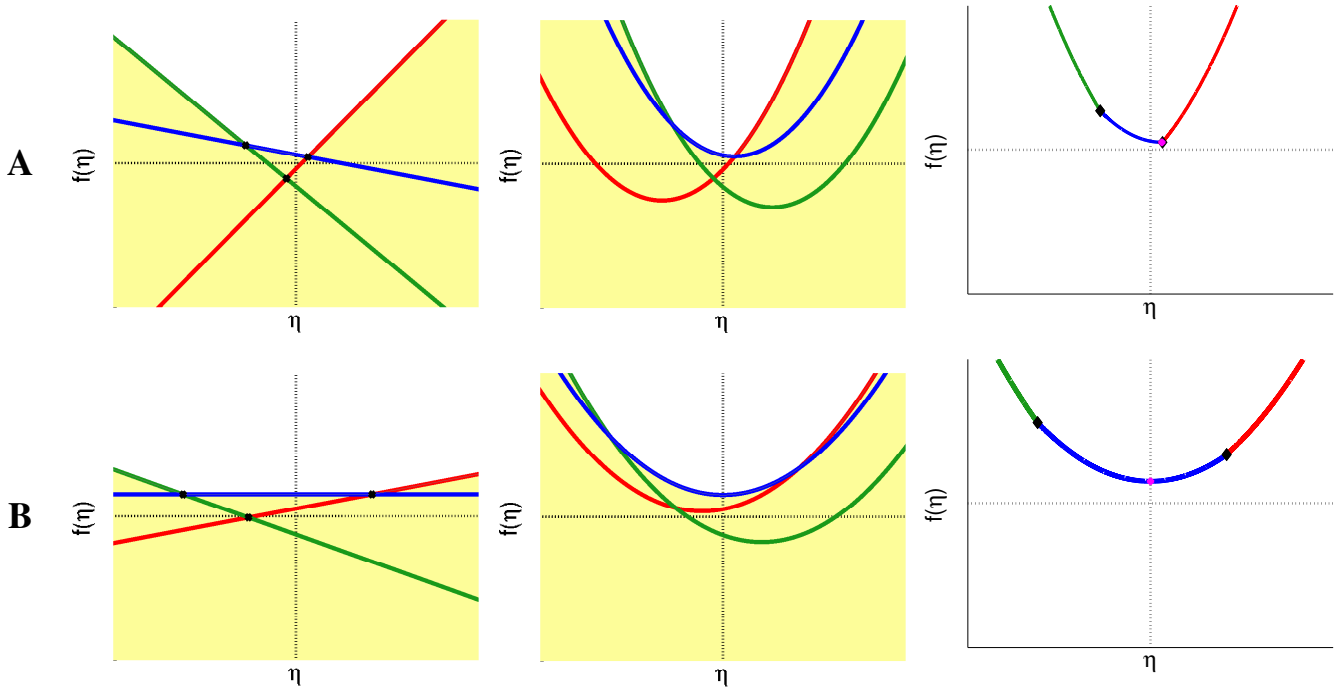
$$\eta^{(t)} \leftarrow \arg\min_{\eta} h^R_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta). \tag{33}$$

To compute the minimum of h^R, we first observe that, by expanding $\mathbf{w}_\eta^{(t+1)}$ and after rearranging terms, Eq. 32 can be explicitly written as

$$\begin{aligned} &h^R_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta) = \\ &\max_{Y \in \mathcal{A}^n_+} \left(\frac{\lambda}{2} + \rho^{(t)} \right) \left|\left| \mathbf{g}^{(t)} \right|\right|^2 \eta^2 - \left\langle \mathbf{g}^{(t)}, \delta\psi_n(Y) + \lambda \mathbf{w}^{(t)} \right\rangle \eta \\ &+ \left(\frac{\lambda}{2} \left|\left| \mathbf{w}^{(t)} \right|\right|^2 + \left\langle \mathbf{w}^{(t)}, \delta\psi_n(Y) \right\rangle + \Delta(Y^n, Y) \right). \end{aligned} \tag{34}$$

It is easy to see that h^R is the upper envelope of a set of quadratic functions of η. In fact, since the quadratic term does not depend on the specific constraint Y, the quadratic upper envelope can be obtained easily by computing the corresponding linear upper envelope, for which there are simple $O(m \log m)$-time algorithms [18] where m is the number of constraints being considered. The resulting envelope is then shifted vertically by the quadratic term, as shown in Figure 5.

Let $Y_1, \dots, Y_m \in \mathcal{A}^n_+$ be the constraints that correspond to segments on the upper envelope and $\eta_1, \dots, \eta_{m-1}$ be their intersections, both sorted left-to-right (visualized in Figure 5),



(a) Set of linear functions (b) Set of quadratic functions (c) Quadratic upper envelope

Fig. 5. Graphical illustration of the optimization problem posed by Eq. 34 for three random constraints shown in different colors. The two rows illustrate the two different cases explained in the main text: **A**) the minimum lies at the intersection itself, **B**) the minimum lies on the interior of one of the adjacent segments. In (a) and (b), the areas below the upper envelope are shaded in yellow. The solid lines in (a) represent the linear part of h^R, while those in (b) and (c) represent the full objective. Figure (c) shows the upper envelope over the three constraints with the optimal η shown in magenta.

given by the upper envelope. Let $\eta_0 = -\infty$ and $\eta_m = \infty$ for notational convenience, so that $[\eta_{i-1}, \eta_i]$ is the interval bounding the i-th segment of the upper envelope. The minimum for each segment i can be computed in closed form as the minimum of the parabola subject to the bounds of the interval, i.e.,

$$\eta_i^* = \max\left(\eta_{i-1}, \min\left(\eta_i, \frac{\left\langle \mathbf{g}^{(t)}, \delta\psi_n(Y_i) + \lambda \mathbf{w}^{(t)} \right\rangle}{\left(\lambda + 2\rho^{(t)}\right)\left|\left|\mathbf{g}^{(t)}\right|\right|^2}\right)\right). \tag{35}$$

Let $h^*(\eta_i^*)$ denote its respective vertical values on the parabola, which is equal to $h^R_{\mathcal{A}^n, \mathbf{w}^{(t)}}(\eta_i^*)$ since by definition the segment of parabola i between η_{i-1} and η_i is part of the upper envelope. Then $\eta^{(t)}$ is simply the η_i^* that results in the smallest value of h^*,

$$\eta^{(t)} = \underset{\eta_i^* \in \{\eta_1^*, \ldots, \eta_m^*\}}{\arg\min} \; h^*(\eta_i^*). \tag{36}$$

In practice, we need only consider the two quadratic segments on either side of the lowest intersection point, as shown in Fig. 5. The minimum must lie on either the intersection itself (case A) or the interior of one of the adjacent segments (case B), due to strong convexity and monotonicity of the upper envelope on either side of minimum. This helps to further reduce computational cost.

As outlined in Algorithm 4, the optimal step size $\eta^{(t)}$ for each violated constraint in $\mathcal{A}^n$ is computed using Eq. 33 – 36 and used to decide how far to move in the opposite direction of the corresponding approximate subgradient $\mathbf{g}^{(t)}$. In order to ensure convergence, we include a hard lower bound on the step-size in the algorithm to prevent it to converge to a point which is not optimal. In the supplementary material we show that the step size $\eta^{(t)}$ resulting for this procedure satisfies the conditions of convergence stated in Eq. 24.

VII. Experimental Results

In this section, we first introduce baselines against which we compare our approach. We then present our results on 2D and 3D data and discuss the length of the training times involved.

A. Multiple Versions of our Approach and Baselines

We will compare three versions of our approach

- **Working sets + sampling**. Compute each subgradient using the working set of constraints where instead of performing inference we use MCMC to sample constraints from a distribution targeting the loss-augmented score. We use a simple decreasing learning rate.
- **Working sets + inference**. Compute each subgradient using the working set of constraints where each constraint is computed by performing loss-augmented inference using graph-cuts or belief-propagation. We also use a simple decreasing learning rate.
- **Working sets + inference + autostep**. This is similar to **Working sets + inference** but the decreasing learning rate is replaced by the adaptive step sizes of Section VI.

against the following six baselines

- **SVM.** A linear SVM that classifies each sample independently without using a CRF.
- **SSVM.** The cutting plane algorithm introduced in [55] and discussed in Section II.
- **SampleRank.** The method proposed in [57] and also discussed in Section II.
- **FW-Struct.** The Frank-Wolfe algorithm for the L2-Loss SSVM [23]. We used the version without averaging known-as BCFW.
- **SGD + inference.** Loss-augmented inference using graph-cuts or belief-propagation. This is the subgradient descent (SGD) formulation of [40]. We try different values for the step size $\eta^{(t)} = \frac{\beta}{t}$ and use cross-validation to determine the best β in the interval $[10^{-5}, \ldots, 10^{-10}]$.
- **SGD + sampling.** Instead of performing inference, use MCMC to sample constraints from a distribution targeting the loss-augmented score. This is equivalent to the method referred to as "SampleRank SVM" in [57].

In addition to these methods, we also experimented with averaging all past subgradients [35], [58], which did not produce competitive results for our biomedical tasks. We therefore did not include them in our comparison charts.

The loss-augmented inference of Eq. 7 required by the structured methods was implemented using the maxflow library of [8] and the Loopy Belief Propagation implementation of [33].

In all cases, we used cross-validation on the training set to determine the regularization constants. All the methods were run for a sufficiently high number of iterations $T = 1000$ to ensure that they reach convergence. Because they involve randomization, the results for **Working sets + sampling** and **SampleRank** were averaged over 5 runs.

The task loss Δ of Eq. 5 is the per-superpixel weighted loss $\Delta(Y^n, Y) = \sum_{i \in \mathcal{V}} \mathcal{I}(y_i \neq y_i^n) r(y_i^n)$. The term $r(y_i^n)$ in the loss function weighs errors for a given class inversely proportional to the frequency at which it appears in the training data.

B. 2D Data

We performed segmentation of photon receptor cells in 2D retinal images. We use the publicly available UCSB retinal dataset, [3] which consists of 50 laser scanning confocal images of normal and 3-day detached feline retinas. The images were annotated by three different experts and the performance is measured by the *Jaccard index* commonly used for image segmentation [12]. The Jaccard index is the ratio of the areas of the intersection between what has been segmented and the ground truth, and of their union. It is written as

$$\text{Jaccard index} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive} + \text{False Negative}} .$$

The segmentation process begins by over-segmenting the images using SLIC superpixels [1]. For each superpixel, we extract a feature vector that captures texture information using 8×8 co-occurrence matrices and 10-bin intensity histograms.

[3] http://www.bioimage.ucsb.edu/research/biosegmentation

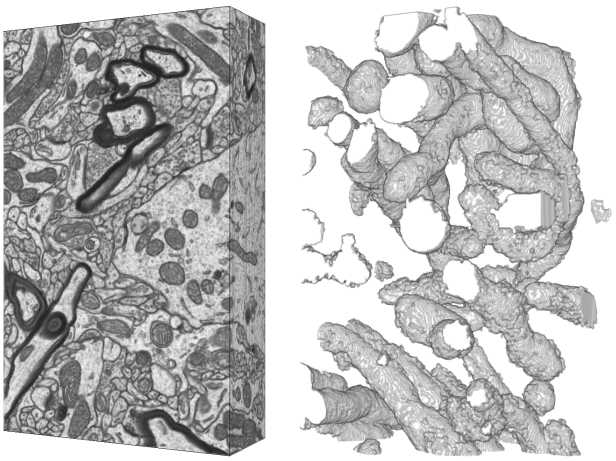


Fig. 7. **Left.** A second isotropic stack of neural tissue, similar to the one show in Fig. 1 but this time acquired from the striatum. This stack is made of $318\ 872 \times 1536$ slices with around 6-nanometer resolution in all three directions. **Right.** 3D rendering of the mitochondria found in a $450 \times 711 \times 318$ subvolume.

These feature vectors are used to train each baseline method, as well as our model. In a second set of experiments, we also transform the features using the kernel method described in Section IV. The original feature vectors are 74-dimensional and are thus mapped to a higher dimensional space. Example segmentations are shown in Fig. 6 and quantitative results are provided in Table I.

In the first line of this table, we compare our results against those of the six baselines using standard features. Working sets clearly improve performance over that of the baselines when using either **Working sets + inference** or **Working sets + sampling**. Among the two, **Working sets + inference** yields the best results, at the cost of a higher-computational load as discussed in Section VII-D. Finally, also incorporating the adaptive step sizes, as in **Working sets + inference + autostep**, further improves performance. In the second line of the table, we use kernelized features instead of standard ones and observe exactly the same trends. We also see a performance increase due to the kernelized features by themselves.

C. 3D Data

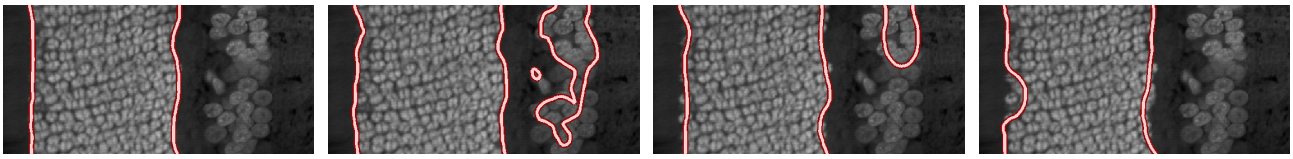
Here we test our approach for the purpose of mitochondria segmentation from a 3D electron microscopy dataset. The segmentation of mitochondria and other organelles is a critical step for providing new insights into brain functionality, and has received a lot of interest from the computer vision and medical imaging communities [2], [20]. We use both the large and publicly available hippocampus stack [4] depicted in Fig. 1 and a second stack from the striatum and depicted in Fig. 7.

We first over-segment the volume into supervoxels [1]. For each one, we extract a feature vector that captures local shape and texture information using Ray descriptors [30], intensity histograms and the following features computed at

[4] http://cvlab.epfl.ch/data/em

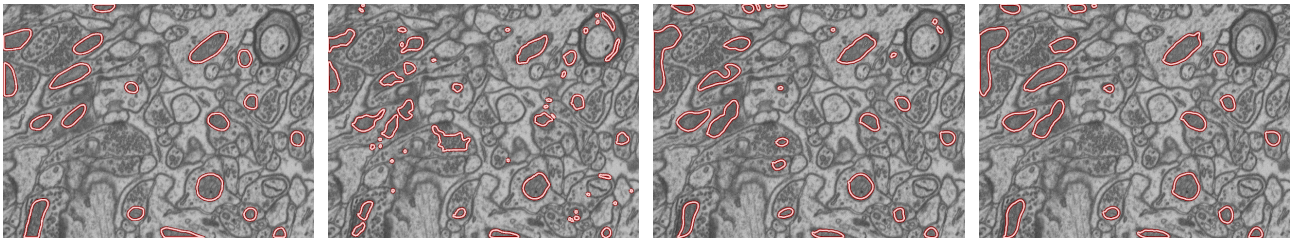
	SVM	SSVM [55]	Sample-Rank [57]	FW-Struct [23]	SGD + sampling	SGD + inference [40]	Working set + sampling	Working set + inference	Autostep
Original features	81.1%	89.4%	82.6%	89.4%	83.0%	87.2%	86.0%	89.8%	**90.2%**
Kernelized features	83.0%	90.1%	82.9%	90.3%	83.9%	87.6%	86.4%	91.4%	**91.8%**

TABLE I
SEGMENTATION PERFORMANCE FOR THE PHOTON RECEPTOR DATASET MEASURED IN TERMS OF THE JACCARD INDEX. THE LAST COLUMN REFERS TO THE METHOD **Working sets + inference + autostep**.



Ground truth SVM SGD + inference [40] Working set + inference

Fig. 6. Segmentation outlines obtained on one image of the photon receptor dataset using some of methods we tested overlaid in red.



Ground truth SVM SGD + inference [40] Working set + inference

Fig. 8. Outlines of mitochondria volumes obtained by different methods overlaid in red on a specific slice of the hippocampus dataset of Fig. 1. Our results are closer to the ground truth than the others.

	SVM	Lucchi [30]	SSVM [55]	Sample-Rank [57]	FW-Struct [23]	SGD + sampling	SGD + inference [40]	Working set sampling	Working set inference	Autostep
Original features	78.6%	80.0%	90.5%	82.6%	92.1%	85.4%	88.7%	87.1%	91.8%	**92.9%**
Kernelized features	82.3%	-	92.7%	83.3%	92.7%	87.4%	89.2%	88.1%	94.4%	**94.8%**

Hippocampus

	SVM	Lucchi [30]	SSVM [55]	Sample Rank [57]	FW-Struct [23]	SGD + sampling	SGD + inference [40]	Working set sampling	Working set inference	Autostep
Original features	81.5%	74.0%	90.0%	84.2%	88.8%	81.7%	84.8%	86.5%	90.1%	**90.7%**
Kernelized features	87.6%	-	90.6%	89.6%	90.5%	87.9%	88.1%	90.2%	90.6%	**92.1%**

Striatum

TABLE II
SEGMENTATION PERFORMANCE MEASURED IN TERMS OF THE JACCARD INDEX FOR THE TWO EM DATASETS. THE LAST COLUMN REFERS TO THE METHOD **Working sets + inference + autostep**.

five different scales: gradient magnitude, Laplacian of Gaussian and eigenvalues of the Hessian matrix, eigenvalues of the structure tensor. These feature vectors are used to train each baseline method, as well as our model. In a second set of experiments, we also transform the features using the kernel method described in Section IV. The original feature vectors are 140-dimensional and are thus mapped to a higher dimensional space. We trained a non-structured kernel SVM using the 140-dimensional standard features extracted from $N = 40000$ randomly sampled supervoxels. This yields a set of 4223 support vectors for the striatum dataset and 4568 for the hippocampus that are then used to compute the transformed features.

In the case of the hippocampus dataset, due to the difficulty in obtaining labels for such large volumes, we performed our experiments on two subvolumes containing $1024 \times 768 \times 165$ voxels. The first subvolume was used to train the various methods while the second one was used for testing. Each subvolume contains $\sim$13K supervoxels. The resulting graphs have $\sim$91K edges. The same kind of splitting was also done

for the striatum dataset, which resulted in CRFs of similar sizes.

The mitochondria we find are depicted as 3D volumes in Figs. 1 and 7. Their outlines are overlaid in Fig. 8 on a single slice, along with those obtained using competing methods.

Performance is again measured by comparing 3D volumes against ground-truth ones in terms of the Jaccard index described in Section VII-B, but using volumes instead of areas. The corresponding results are provided in Table II. For [30], we use is a non-linear RBF-SVM classifier. The numbers exhibit the same trends as those discussed at the end of Section VII-B.

For the sake of completeness, we have also tested a very recent mitochondria segmentation method [45] that does *not* rely on structured learning. Instead, it trains a cascade of classifiers at different scales and has been shown to outperform earlier algorithms based on Neural Networks, SVMs, and Random Forests on EM imagery. For the hippocampus and striatum datasets, it yields 83.8% and 83.5% Jaccard indices, which is much lower than what we obtain. The time required to train this method on our data (72605s) is also significantly longer than all those in Table III, and discussed in the next section.[5]

D. Training Cost

The time and memory required to train a classifier are important considerations for any Machine Learning approach. Concerning the memory requirements, the working set used by our approach does not lead to a significant increase in memory as we only need to store the feature maps, which are generally much more compact than the full labelings of the CRF.

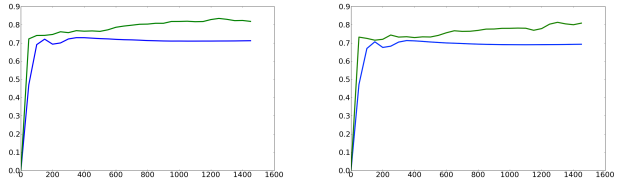
While the linear SSVM is fast to train thanks to the use of kernelized features, the training of the first non-linear SVM used to transform the features can still potentially incur quadratic cost. Fortunately, it can be kept in check using known techniques such as randomly sampling the data or iteratively mining for hard examples [13]. However, these techniques cannot be used to directly speed up the SSVM though because they treat pixels as independent examples and disregard the structure of the graph.

We conducted a run-time analysis of the standard subgradient method of [40] against the three versions of our algorithm discussed in this section. Table III gives the training time for each method for the same 1000 iterations, which is enough for all methods to have converged. Note, however, that Fig. 10 shows that **Working sets + inference + autostep** requires only about 200 iterations to converge. So, even though each iteration is a little slower due to the overhead attributable to maintaining the working set and doing a line-search, **Working sets + inference + autostep** yields in practice better solutions faster. Table III also shows that **Working sets + sampling** is much faster than solving the loss-augmented inference to find the most violated constraint.

[5]We used the publicly available Matlab implementation.

	EM	**Photon receptor**
SSVM [55]	43250s	1113s
SampleRank [57]	2524s	382s
SGD + Sampling	2481s	354s
Working sets + sampling	2619s (+5.5%)	370s (+4.5%)
SGD + inference [40]	5315s	438s
Working sets + inference	5842s (+9.9%)	492s (+12.3%)
Working sets + inference + autostep	6142s (+15.6%)	510s (+16.3%)

TABLE III
TRAINING TIME REQUIRED TO REACH CONVERGENCE ($T = 100$ ITERATIONS FOR SSVM AND $T = 1000$ ITERATIONS FOR ALL OTHER METHODS) ON THE HIPPOCAMPUS EM DATASET. THE THREE VERSIONS OF OUR APPROACH DISCUSSED AT THE BEGINNING OF SECTION VII ARE LABELED IN BOLD. THE SLOWDOWN RESULTING FROM USING THE WORKING SET AND THE ADAPTIVE STEP SIZES IS SHOWN IN PARENTHESES.



(a) Training set, EM *(b) Test set, EM*

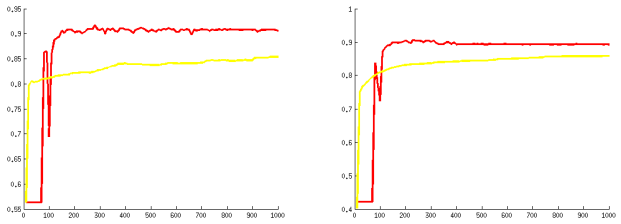
Fig. 9. Learning curves showing the training and test scores (Jaccard index) on the Hippocampus dataset as a function of the number of iterations t. We report results for the sampling method with and without working set in green and blue respectively.

E. Generalization Error

The evolution of the training scores and test scores as a function of the number of iterations is shown in Figs. 10 and 9. The parameters for the methods reported in these figures were found using cross-validation to avoid over-fitting. We have seen that all these methods could reach higher scores on the training set using different sets of parameters. **Working sets + inference + autostep** achieved the highest score on the training set and had to be regularized by increasing the value of λ (increasing $\rho^{(t)}$ also lead to a better generalization error). The curves in Fig. 9 clearly show that the working set of constraints produces much higher score on both the training and test sets. Fig. 10 shows that **Working sets + inference + autostep** significantly outperforms **SGD** on the training set as well as the test set. As shown in Table IV, it also outperforms **SSVM** on the training set.

	Hippocampus	**Striatum**	**Photon receptor**
SSVM	90.9%	86.3%	86.3%
Autostep	93.8%	90.6%	89.2%

TABLE IV
TRAINING SCORES ACHIEVED BY **SSVM** AND **Working sets + inference + autostep** USING THE ORIGINAL FEATURES ON ALL THREE DATASETS.



(a) Training set, EM *(b) Test set, EM*

Fig. 10. Learning curves showing the training and test scores, expressed in terms of the Jaccard index, on the Striatum dataset as a function of the number of iterations t. We report results for **Working sets + inference + autostep** and **SGD + inference** in red and yellow respectively.

F. Further Observations

1) Approximate subgradients: Our approach, and most of the existing methods presented in the related work section, all optimize the same objective function, given in Eq. 3, and rely on the computation of the most violated constraints. When these constraints cannot be obtained exactly, as is usually the case in image segmentation, the subgradient estimates are then necessarily only approximate. Certain methods, such as [55], require these quantities to be exact, while others, such as [40], [14] and [23], may work with approximate subgradients to obtain convergence up to a certain accuracy. The experimental results suggest that using a history of constraints as well as appropriate step sizes make learning more robust to approximation errors. An in-depth theoretical analysis of the accuracy of the subgradients will undoubtedly provide further insight into the behavior of this class of methods; though this is beyond the focus of this work, which is motivated by the need to improve performance in known applications.

2) Parameter selection for Kernelized features: While in certain cases kernel methods show improved performance compared with linear methods, they also introduce tuning parameters such as the number of support vectors. The kernel method presented in Section IV is not exempt from this difficulty. The selection of the number of support vectors for this method was done by cross-validation over the regularization constant λ and by empirically selecting an appropriate number of training samples. The memory complexity of the linear SSVM is affected by the dimension of the kernelized features and is therefore another limiting factor that must be considered. We found empirically that the number of support vectors scale almost linearly with the number of training samples when fixing the regularization constant λ. In our case, we found a training set of 40,000 samples to be a good trade-off between performance and training complexity.

3) Our Approach vs **SSVM***:* Although **SSVM** takes fewer iterations to converge, we found the cost per iteration to be much higher than SGD-based methods, which is mostly due to the higher cost of the loss-augmented inference. We hypothesize that the region of the space visited by **SSVM** is highly non-submodular. One alternative to address this problem proposed in [50] is to perform the optimization over a much smaller set of parameters for which exact optimization can be performed with graph-cuts.

4) **Working sets + inference** *vs* **Working sets + inference + autostep***:* As stated earlier, all methods converged after at most 1000 iterations. The improvement due to the automatic step size selection reported in the experimental section is thus not due to a faster rate of convergence. Rather, we believe it to be attributable to the difficulty of manually setting the right step size at each iteration. We found empirically that different step size selections can occasionally yield improvements on specific datasets but we did not find a specific one that would perform well for all datasets. We thus opted for the standard decreasing scheme $\eta^{(t)} = \frac{\beta}{t}$ that gave the best results overall.

VIII. Conclusion

Understanding neural connectivity, function, and structure of the brain requires detailed 3D models. Although new imaging techniques such as FIB-SEM allow neuroscientists to visualize the brain in great detail, these crucial models are mostly still traced by hand. To eliminate this barrier to progress, new automatic segmentation techniques for identifying structures of interest in these rich datasets are required.

To address this need, we have presented a new segmentation framework that nearly achieves a human level of performance, raising the bar over previous methods, including our own earlier work. To this end, we introduced three key innovations, including a technique to leverage the power of non-linear kernels in a structured prediction framework as well as a working set based approximate subgradient method with line-search to learn graphical models when inference is challenging. Our method is particularly appealing for learning large CRFs with loops, which are common in computer vision tasks, since under these circumstances the use of working sets of constraints makes the subgradient method more robust to approximation errors and produce higher-quality solutions.

The benefits of our approach were clearly demonstrated on 2D confocal optical images and 3D EM image stacks, but it is important to note that our method is general and can be applied to imagery from any type of microscopy.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, Nov 2012.

[2] B. Andres, U. Koethe, M. Helmstaedter, W. Denk, and F. Hamprecht. Segmentation of SBFSEM Volume Data of Neural Tissue by Hierarchical Classification. In *DAGM Symposium on Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 142–152, 2008.

[3] M.-F. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79–94, Oct 2006.

[4] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. Kernelized Structural SVM Learning for Supervised Object Segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 2153–2160, 2011.

[5] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[6] J. Besag. Spatial Interaction and the Statistical Analysis of Lattice Systems. *J. Roy. Statist. Soc. B*, 36(2):192–225, 1974.

[7] Y. Boykov and M. Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *International Conference on Computer Vision*, volume 1, pages 105–112, 2001.

[8] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sep 2004.

[9] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[10] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[11] K. Crammer, R. Mcdonald, and F. Pereira. Scalable Large-Margin Online Learning for Structured Classification. Technical report, University of Pennsylvania, 2005.

[12] M. Everingham, C. W. L. Van Gool and, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge (VOC2010) Results, 2010.

[13] P. Felzenszwalb, D. Mcallester, and D. Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[14] T. Finley and T. Joachims. Training Structural SVMs When Exact Inference is Intractable. In *International Conference on Machine Learning*, pages 304–311, 2008.

[15] V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th international conference on Machine learning*, pages 320–327, 2008.

[16] J. Gonfaus, X. Boix, J. Weijer, A. Bagdanov, J. Serrat, and J. Gonzalez. Harmony Potentials for Joint Classification and Segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 3280–87, 2010.

[17] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Amer. Statistical Assoc. J.*, 58(301):13–30, March 1963.

[18] J. J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, Dec 1989.

[19] T. Joachims, T. Finley, and C.-N. Yu. Cutting-Plane Training of Structural SVMs. *Machine Learning*, 77(1):27–59, Oct 2009.

[20] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister. Large-Scale Automatic Reconstruction of Neuronal Processes from Electron Microscopy Images. *arXiv preprint*, 2013.

[21] K. C. Kiwiel. Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46(1-3):105–122, Jan 1990.

[22] V. Kolmogorov and R. Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, Feb 2004.

[23] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-Coordinate Frank-Wolfe Optimization for Structural SVMs. In *International Conference on Machine Learning*, pages 53–61, 2013.

[24] L. Ladicky and P. Torr. Locally Linear Support Vector Machines. In *International Conference on Machine Learning*, pages 985–992, 2011.

[25] J. Lafferty, A. Mccallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *International Conference on Machine Learning*, pages 282–289, 2001.

[26] Q. V. Le, A. J. Smola, and S. Vishwanathan. Bundle methods for machine learning. In *Advances in neural information processing systems*, pages 1377–1384, 2007.

[27] A. Lucchi, Y. Li, X. Boix, K. Smith, and P. Fua. Are Spatial and Global Constraints Really Necessary for Segmentation? In *International Conference on Computer Vision*, pages 9–16, 2011.

[28] A. Lucchi, Y. Li, and P. Fua. Learning for Structured Prediction Using Approximate Subgradient Descent with Working Sets. In *Conference on Computer Vision and Pattern Recognition*, pages 1987–1994, June 2013.

[29] A. Lucchi, Y. Li, K. Smith, and P. Fua. Structured Image Segmentation Using Kernelized Features. In *European Conference on Computer Vision*, pages 400–413, October 2012.

[30] A. Lucchi, K. Smith, R. Achanta, G. Knott, and P. Fua. Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks with Learned Shape Features. *IEEE Transactions on Medical Imaging*, 31(2):474–486, Feb 2012.

[31] S. Maji and A. Berg. Max-Margin Additive Classifiers for Detection. In *International Conference on Computer Vision*, pages 40–47, 2009.

[32] D. Mcallester, T. Hazan, and J. Keshet. Direct Loss Minimization for Structured Prediction. In *Advances in Neural Information Processing Systems*, pages 1594–1602, 2010.

[33] J. M. Mooij. libDAI: A Free and Open Source C++ Library for Discrete Approximate Inference in Graphical Models. *Journal of Machine Learning Research*, 11:2169–2173, Aug. 2010.

[34] K. Murphy, Y. Weiss, and M. Jordan. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

[35] Y. Nesterov. Primal-Dual Subgradient Methods for Convex Problems. *Math. Program.*, 120(1):221–259, Aug 2009.

[36] S. Nowozin, P. Gehler, and C. Lampert. On Parameter Learning in CRF-Based Approaches to Object Class Image Segmentation. In *European Conference on Computer Vision*, pages 98–111, 2010.

[37] J. Platt. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. MIT Press, 1998.

[38] B. Polyak. A General Method of Solving Extremum Problems. *Soviet Mathematics*, 8(3):593–597, 1967.

[39] A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2007.

[40] N. Ratliff, J. A. Bagnell, and M. Zinkevich. (Online) Subgradient Methods for Structured Prediction. In *International Conference on Artificial Intelligence and Statistics*, pages 380–387, 2007.

[41] S. M. Robinson. Linear Convergence of Epsilon-Subgradient Descent Methods for a Class of Convex Functions. *Mathematical Programming*, 86(1):41–50, Sep 1999.

[42] R. Samdani and D. Roth. Efficient Decomposed Learning for Structured Prediction. *International Conference on Machine Learning*, 1206.4630:217–224, 2012.

[43] N. Schraudolph. Local Gain Adaptation in Stochastic Gradient Descent. In *International Conference on Artificial Neural Networks*, pages 569–574, 1999.

[44] A. Severyn and A. Moschitti. Fast Support Vector Machines for Structural Kernels. In *European Conference on Machine Learning*, pages 175–190, 2011.

[45] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen. Image Segmentation with Cascaded Hierarchical Models and Logistic Disjunctive Normal Networks. In *International Conference on Computer Vision*, pages 2168–2175, 2013.

[46] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal Estimated Sub-Gradient Solver for SVM. *Math. Program.*, 127(1):3–30, Mar 2011.

[47] N. Shor, K. Kiwiel, and A. Ruszcayǹski. *Minimization Methods for Non-Differentiable Functions*, volume 3 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, 1985.

[48] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. *International Journal of Computer Vision*, 81(1):2–23, Jan 2009.

[49] C. Sutton and A. McCallum. Introduction to Conditional Random Fields for Relational Learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

[50] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs Using Graph Cuts. In *European Conference on Computer Vision*, pages 582–595, 2008.

[51] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning Structured Prediction Models: A Large Margin Approach. In *International Conference on Machine Learning*, pages 896–903, 2005.

[52] B. Taskar, C. Guestrin, and D. Koller. Max-Margin Markov Networks. In *Advances in Neural Information Processing Systems*, pages 25–32, 2003.

[53] B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured Prediction, Dual Extragradient and Bregman Projections. *Journal of Machine Learning Research*, 7:1627–1653, 2006.

[54] C. H. Teo, S. Vishwanthan, A. J. Smola, and Q. V. Le. Bundle methods for regularized risk minimization. *Journal of Machine Learning Research*, 11:311–365, 2010.

[55] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *International Conference on Machine Learning*, page 104, 2004.

[56] A. Vedaldi and A. Zisserman. Efficient Additive Kernels via Explicit Feature Maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):480–492, Mar 2012.

[57] M. Wick, K. Rohanimanesh, K. Bellare, A. Culotta, and A. Mccallum. Samplerank: Training Factor Graphs with Atomic Gradients. In *International Conference on Machine Learning*, pages 777–784, 2011.

[58] L. Xiao. Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization. *Journal of Machine Learning Research*, 11:2543–2596, December 2010.

[59] C.-N. Yu and T. Joachims. Training Structural SVMs with Kernels Using Sampled Cuts. In *Conference on Knowledge Discovery and Data Mining*, pages 794–802, 2008.