Contextual Classification with Functional Max-Margin Markov Networks

Daniel Munoz J. Andrew Bagnell Nicolas Vandapel Martial Hebert
The Robotics Institute
Carnegie Mellon University
{dmunoz, dbagnell, vandapel, hebert}@ri.cmu.edu

Abstract

We address the problem of label assignment in computer vision: given a novel 3-D or 2-D scene, we wish to assign a unique label to every site (voxel, pixel, superpixel, etc.). To this end, the Markov Random Field framework has proven to be a model of choice as it uses contextual information to yield improved classification results over locally independent classifiers. In this work we adapt a functional gradient approach for learning high-dimensional parameters of random fields in order to perform discrete, multi-label classification. With this approach we can learn robust models involving high-order interactions better than the previously used learning method. We validate the approach in the context of point cloud classification and improve the state of the art. In addition, we successfully demonstrate the generality of the approach on the challenging vision problem of recovering 3-D geometric surfaces from images.

1. Introduction

We consider the task of label assignment in which we wish to assign a unique label to each site in a domain; we define a site as the primitive entity in a domain. In computer vision applications, pixels or superpixels often serve as the sites in image applications, and points or voxels are used when working with 3-D laser data. In this paper, we approach this problem as a supervised learning problem where given hand-labeled data, we extract various types of features that incorporate multiple levels of contextual information and then discriminatively train a model that accounts for the contextual interaction among sites. To model these interactions, we use a Conditional Random Field (CRF) [14] variant. The contribution of this paper is to adapt a recent functional gradient technique [20] for learning such random fields. As our experimental section will show, this approach can learn robust models involving high-order interactions more accurately than the previous learning method.

Functional gradient learning has recently been used to learn binary score functions to estimate the quality of inputs for footstep and grasp planning [18] and path planning [22]. In this work, we adapt these functional gradient techniques to learn the potential functions of Markov Random Fields for discrete multi-label contextual classification. We validate the effectiveness of the approach on two applications. First, we improve the state of the art on 3-D point cloud classification, a task known to work well with random fields [1]. In the second, we illustrate the generality of the model in a task where random fields with all potentials jointly learned from features have not traditionally been used: recovering 3-D surface from images as described by Hoiem *et al.* [8]. We show that the model learned with the functional gradient technique improves significantly over the previously used learning method and is comparable with the approach used in [8]. In Figure 1, we present example classification results from these two applications[1].

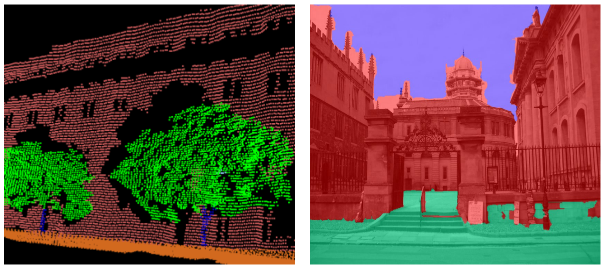
Figure 1. Example results from our two applications: 3-D point cloud classification (left) and geometric surface estimation (right).

CRFs are a popular tool in computer vision [13] for propagating contextual information among neighboring sites. In contrast to maximum a posteriori (MAP) learning for CRFs, we focus on max-margin learning, *i.e.* Max-Margin Markov Networks (M^{3}Ns), as proposed by Taskar *et al.* [26]. Instead of using a parametric gradient method, which has been demonstrated to be the current state of the art learning method for one of our applications [19], we use the functional gradient algorithm to learn the M^{3}N model. The al-

[1]This paper is best viewed in color. The following color code is used through the paper. **3-D Point Clouds:** orange = *ground*, green = *vegetation*, dark-blue = *tree-trunks/poles*, sky-blue = *wire*, red = *facade*. **Geometric Surface:** turquoise = *ground*, red = *vertical surface*, purple = *sky*.

gorithm is simple to implement, needs no quadratic or linear program solver, and its low memory requirements enable learning over large datasets. Our experiments show that the techniques we present here are viable for accurately learning large models with as many as 1,000 parameters.

Unlike MAP learning, max-margin learning optimizes a different convex objective for structured prediction problems. Similar to MAP learning, max-margin learning also requires performing inference when computing the gradient; however, as we will review, only the MAP labeling is required. We believe it has not been definitely established which learning objective is the best to optimize when using approximate inference as reported results are comparable [29]. However, the presented max-margin approach follows the suggestion of Kumar *et al.* [12] to use the same approximate inferences during both training and testing. Since max-margin learning does not require computation of marginals, we can learn by using efficient graph-cut inference, which are typically faster while as accurate as belief propagation and variants [23].

Related work. Our approach uses a functional gradient boosting technique to learn M^{3}Ns with cliques of many variables. This technique is interpreted as boosting over the space of the random field's potential functions that we would like to learn. From the perspective of learning CRFs, this approach is similar to the works of Dietterich *et al.* [3], Torralba *et al.* [27], and Liao *et al.* [15]. In [3], the authors consider a form of gradient tree boosting [6] for only 1-D CRFs. In [27], the authors perform rounds of boosting of weak learners that first learn to classify locally "easier" things and then focus on larger scale structures that are harder to classify. And in [15], the authors use another form of boosting for feature selection and model parameter learning in CRFs. The drawback with these three approaches is that they all rely on the marginal probabilities or a form of belief/confidence from inference at each stage during learning; hence they are intractable for learning random fields with high-order interactions. Additionally, Szummer *et al.* [24] and Franc and Savchynskyy [5] investigate optimizing the pairwise M^{3}N criterion using cutting-plane techniques which require the use of a quadratic program solver. Following [19, 16], we use gradient-based techniques to optimize our unconstrained objective. This paper is also related to our work on performing classification with a specific high-order M^{3}N onboard of a robot [17]; however, we now investigate an improved learning approach, robust potential interactions, and a new application.

The remainder of this paper is as follows. First, we present a brief review of CRFs and M3Ns in Sections 2 and 3, respectively. We will use this as a basis to explain our adaptation of the functional gradient boosting algorithm for M^{3}Ns in Section 4. Finally, we demonstrate the effectiveness of our approach in Section 5 on the applications of 3-D point cloud classification and geometric surface estimation.

2. Conditional random fields

We are interested in multi-label problems for computer vision applications where each site Y_i can take on K possible labels: $Y_i \in \{\ell_1, \ldots, \ell_K\}$. Denoting by $\mathbf{Y} = \{Y_i\}_{i=1}^N$ the set of label assignments and by $\mathbf{X}$ the data variables, a CRF is a probabilistic graphical model that defines the joint-conditional distribution $P(\mathbf{Y}|\mathbf{X})$. This distribution is represented as an undirected graph over nodes $\mathbf{Y}$, in which the edges indicate interactions between variables. In general, the variables $\mathbf{Y}_c = \{\mathbf{Y}_i\}_{i\in c}$ in each clique $c \in C$ of the graph are associated with a potential function $\phi_c(\mathbf{Y}_c)$ that measures the score or compatibility of an assignment $\mathbf{y}_c$. In our problem, the potentials are dependent on the extracted features $\mathbf{x}_c \in \mathbb{R}^d$ that describe clique c and the model parameters (weights) $\mathbf{w} \in \mathbb{R}^n$ we wish to learn.

We review these concepts first in terms of log-linear CRF models, but we will discuss learning more general potential functions in Section 4. Log-linear potential functions take the form $\log \phi_c(\mathbf{x_c}, \mathbf{y_c}; \mathbf{w}) = \mathbf{w}^T \mathbf{f}(\mathbf{x}_c, \mathbf{y}_c)$, where $\mathbf{f}(\cdot, \cdot) \in \mathbb{R}^n$ is a vector-valued feature function. Defining $\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{c\in C} \mathbf{f}(\mathbf{x}_c, \mathbf{y}_c)$, the distribution is given by

$$P(\mathbf{y}|\mathbf{x}; \mathbf{w}) = \frac{1}{Z} \exp(\mathbf{w}^T \mathbf{f}(\mathbf{x}, \mathbf{y})) = \frac{1}{Z} \exp(\Phi(\mathbf{x}, \mathbf{y}; \mathbf{w})), \tag{1}$$

where Z is the normalizing partition function given by $Z = \sum_{\mathbf{y}} \prod_{c\in C} \phi_c(\mathbf{x}_c, \mathbf{y}_c; \mathbf{w})$.

3. Max-margin markov networks

We review the M^{3}N learning framework, specifically we examine a type of M^{3}N called Associative Markov Networks (AMNs) [25]. These models are attractive as approximate inference can be performed efficiently with graph-cuts. We will use this model in our experiments and their formulation will help guide the functional gradient derivation in Section 4.

3.1. Pairwise associative markov networks

An AMN is a generalization of the Pott's model which favors all nodes in a clique to be assigned the same label. In the pairwise model, the model parameters consist of two types of weight vectors that are specific to a group of cliques and are *shared* within each group's potential functions. One type is shared by all the node potentials and another type is shared by all the edge potentials $\mathbf{w} = [\mathbf{w}_n; \mathbf{w}_e] \in \mathbb{R}^n$, where the subscripts n and e denote "node" and "edge" to specify the respective *clique-type*. Each clique-type weight vector is a concatenation of different weight vectors for each label $\{l_k\}_{i=1}^K$, for instance $\mathbf{w}_n = [\mathbf{w}_n^1; \ldots; \mathbf{w}_n^K]$. The potentials are defined in terms of extracted feature vectors from the data that describe the cliques (nodes and edges) that constitute the random field: $\mathbf{x}_i$ and $\mathbf{x}_{ij}$ represent the features that describe node i and edge (i,j), respectively. Then, the node potentials are defined as

$$\log \phi_n(\mathbf{x}_i, y_i = l_k; \mathbf{w}) = \mathbf{w}^T \mathbf{f}_n(\mathbf{x}_i, l_k) = \mathbf{w}_n^k \cdot \mathbf{x}_i,$$

where l_k is the label assigned to node i. The feature function $\mathbf{f}_n$ simply returns a vector with the clique (respectively, node) features aligned in the dimensions of the weights corresponding to the clique-type (respectively, node-type) and of the specified label (l_k). Similarly, the edge potentials are defined as

$$\begin{aligned}\log \phi_e(\mathbf{x_{ij}}, y_i = l_k, y_j = l_l; \mathbf{w}) &= \mathbf{w}^T \mathbf{f}_e(\mathbf{x}_{ij}, l_k, l_l) \\ &= \begin{cases} \mathbf{w}_e^k \cdot \mathbf{x}_{ij} & , l_k = l_l \\ 0 & , \text{o/w}, \end{cases}\end{aligned}$$

where l_k and l_l are the labels of nodes i and j. By constraining the edge potentials to be non-negative, certain submodularity constraints are satisfied and enables efficient inference through the α-expansion graph-cut method [2].

3.2. Subgradient learning

Given labeled training data $C = (\mathbf{x}, \widehat{\mathbf{y}})$, we wish to discriminatively learn the weights $\mathbf{w}$. Ratliff *et al.* [19] provided an unconstrained formulation of the max-margin criteria described by Taskar *et al.* [25] that enables much more efficient optimization. Instead of solving a convex program, we use the subgradient method to minimize the convex function $\mathcal{O}(\mathbf{w}) =$

$$\min_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + \max_{\mathbf{y}}(\Phi(\mathbf{x}, \mathbf{y}; \mathbf{w}) + \mathcal{L}(\mathbf{y}, \widehat{\mathbf{y}})) - \Phi(\mathbf{x}, \widehat{\mathbf{y}}; \mathbf{w}), \tag{2}$$

where λ is a regularization term and $\mathcal{L}(\mathbf{y}, \widehat{\mathbf{y}})$ is a loss function that acts as the margin between any labeling ($\mathbf{y}$) and the true labeling ($\widehat{\mathbf{y}}$). Typically for structured prediction (and in our applications) the Hamming distance between the two labellings is used, as it does not affect submodularity.

Equation 2 is non-differentiable but its subgradients $\mathbf{g_w} \in \partial\mathcal{O}(\mathbf{w})$ can be written as $\mathbf{g_w} = \lambda \mathbf{w} + \mathbf{f}(\mathbf{x}, \mathbf{y}^*) - \mathbf{f}(\mathbf{x}, \widehat{\mathbf{y}})$, where $\mathbf{y}^* = \arg\max_{\mathbf{y}}(\Phi(\mathbf{x}, \mathbf{y}; \mathbf{w}) + \mathcal{L}(\mathbf{y}, \widehat{\mathbf{y}}))$ and is estimated through inference. A solution is obtained through descent after T iterations with decaying step size $\alpha_t = c/\sqrt{t}$ (for some positive c) using the update rule $\mathbf{w}_{t+1} \leftarrow \mathcal{P}[\mathbf{w}_t - \alpha \mathbf{g_w}]$. The projection operator $\mathcal{P}[]$ projects negative edge weights to zero in order to satisfy the submodularity constraints.

3.3. Beyond pairwise potentials

In Kohli *et al.* [10], the authors show how to optimize with graph-cut inference a potential function, called the P^n Potts model, that encompasses many variables. This model is of the same associative potential form, defined over large cliques. That is, it assigns a high potential value if all nodes in the clique are labeled the same and zero otherwise. Following the authors' convention, we define our high-order cliques as sets of points resulting from a clustering or segmentation algorithm.

In our related work [17], we show how to incorporate high-order parameters with the subgradient method. In this work, we empirically demonstrate that functional gradient learning is better suited for learning high-order interaction models than the subgradient method. Quantitative and qualitative comparisons will be presented in Section 5.

Generalizing [17], we define a *clique-set* C_i as a group of cliques that shares the same potentials. We associate the resulting cliques (segments) from one or more segmentations to a particular clique-set. Hence, the cliques' orders in a clique-set are not necessarily homogeneous. A random field is then composed of S clique-sets, $C = \{C_i\}_i^S$, and its overall potential is $\Phi(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{i=1}^S \sum_{c \in C_i} \log \phi_{C_i}(\mathbf{x}_c, \mathbf{y}_c)$, where

$$\begin{aligned}\log \phi_{C_i}(\mathbf{x}_c, \mathbf{y}_c) &= \mathbf{w}^T \mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c) &(3)\\ &= \begin{cases} \mathbf{w}_{C_i}^k \cdot \mathbf{x}_c & , y_i = l_k \forall i \in c \\ 0 & , \text{o/w}. \end{cases} &(4)\end{aligned}$$

For example, $S = 2$ in the simple model of Section 3.1.

4. Functional gradient boosting

We discuss our contribution of learning M^{3}Ns using functional gradient boosting as described by Ratliff *et al.* [18], now in the context of multi-label structured prediction.

4.1. Intuition

Before proceeding with the functional gradient derivation, it is useful to examine the subgradient update in order to provide intuition on the new approach. Consider the update for a pairwise AMN objective with no regularization ($\lambda = 0$) and unit step-size ($\alpha_t = 1$). The update adds the negative subgradient to the model parameters: $\mathbf{w} \leftarrow \mathbf{w} + (-\mathbf{g_w})$, where $-\mathbf{g_w} =$

$$\sum_{i=1}^{N} \mathbf{f}_n(\mathbf{x}_i, \widehat{y_i}) - \mathbf{f}_n(\mathbf{x}_i, y_i^*) + \sum_{ij \in E} \mathbf{f}_e(\mathbf{x}_{ij}, \widehat{\mathbf{y}_{ij}}) - \mathbf{f}_e(\mathbf{x}_{ij}, \mathbf{y}_{ij}^*). \tag{5}$$

Consider the first summation term. In words, when node i is truly l_k but incorrectly inferred as l_l, the update directly adds $\mathbf{x}_i$ to the parameters $\mathbf{w}_n^k$ and subtracts $\mathbf{x}_i$ from $\mathbf{w}_n^l$. If the node was correctly classified, then the two terms cancel and do not affect the subgradient. The second summation similarly performs the same update over the edge clique-set's parameters ($\mathbf{w}_e$) with the edge features $\mathbf{x}_{ij}$. Note that with associative potentials, all nodes in the clique must be labeled the same to have a non-zero feature function vector $\mathbf{f}_e$ and, hence, contribute to the subgradient. This increase/decrease in the weights directly corresponds to an increase/decrease in how the potential values will change.

Instead of clique potentials that are linear in the features, we now assume potentials as more general functions of features. That is, we define $\mathbf{\Psi}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{S} \sum_{c \in C_i} \psi_{C_i}(\mathbf{x_c}, \mathbf{y_c})$ as our general clique potential functions. With functional gradient boosting, the goal is to search the space of potential functions that best models the given training data. Intuitively, the functional gradient will encompass the same type of effect as before. Instead of directly modifying the parameters with the features at each step t, we now *fit a function* $h_t()$ to the features that accordingly increases/decreases the potentials. The following will derive the potential functions as a composition of the functions in the form of $\psi_{C_i}(\mathbf{x}_c, \mathbf{y}_c) = \sum_t \alpha_t h_t(\mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c))$.

4.2. Derivation

The AMN objective functional is defined as

$$\mathcal{O}[\Psi] = \max_{\mathbf{y}}(\Psi(\mathbf{x}, \mathbf{y}) + \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})) - \Psi(\mathbf{x}, \hat{\mathbf{y}}). \quad (6)$$

Using the tools in [18, 20], the negative functional gradient of Equation 6 is defined as

$$-\nabla_f \mathcal{O}[\Psi] = \sum_{i=1}^{S} \sum_{c \in C_i} \delta_{\mathbf{f}_{C_i}(\mathbf{x}_c, \widehat{\mathbf{y}_c})} - \delta_{\mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c^*)}, \quad (7)$$

where δ is the dirac delta at the point of evaluation and ∇_f is the functional gradient operator.

In the space of potential functions, the negative functional gradient is positive at clique feature locations where cliques are assigned their ground truth label and negative at the locations where they are assigned their inferred label. If the ground truth and inferred label for a clique agree, the impulses cancel as with the parametric gradient. As a form of boosting, we incrementally augment our learned potential function with the function h_t^* from a predefined class of functions $\mathcal{H}$ that best correlates with the negative functional gradient. That is, we add a function to our clique potentials that maximizes its inner product with the negative functional gradient based on the model's misclassifications

$$\begin{aligned} h_t^* &= \arg\max_{h_t \in \mathcal{H}} \langle h_t, -\nabla_f \mathcal{O}[\Psi] \rangle \\ &= \arg\max_{h_t \in \mathcal{H}} \sum_{i=1}^{S} \sum_{c \in C_i} \langle h_t, \delta_{\mathbf{f}_{C_i}(\mathbf{x}_c, \widehat{\mathbf{y}_c})} \rangle \text{-} \langle h_t, \delta_{\mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c^*)} \rangle \\ &= \arg\max_{h_t \in \mathcal{H}} \sum_{i=1}^{S} \sum_{c \in C_i} h_t(\mathbf{f}_{C_i}(\mathbf{x}_c, \widehat{\mathbf{y}_c})) \text{-} h_t(\mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c^*)). \end{aligned}$$

In general, this last step trains a predefined class of binary classifiers or regressors with +1 and -1 targets. Similarly to the parametric gradient, the update rule increases and decreases the potential values by now fitting functions to the ground truth (+1) and inferred (-1) feature locations, respectively. After T iterations, the clique potentials are then of the form $\psi_{C_i}(\mathbf{x_c}, \mathbf{y_c}) = \sum_t^T \alpha_t h_t(\mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c))$ as proposed. The entire learning algorithm and its sub-procedure is given in Algorithms 1 and 2.

Any simple (non-linear) learner (decision trees, etc.) could be used to fit the functional gradient direction. Because we are potentially interested in classification onboard of a robot (as in [17]), we chose linear regression in order to compute the potential values more efficiently than with a non-linear regressor. This approach requires 1 instead of T computations to evaluate the potential functions since we cumulatively update the the model over time: $\psi_{C_i}(\mathbf{x_c}, \mathbf{y_c}) = \sum_t \alpha_t h_t(\mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c)) = \sum_t \alpha_t \mathbf{h_t}^T \mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c) = \mathbf{w}^T \mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c)$.

A variant of this algorithm is to perform *exponentiated* functional gradient descent, as described in [18]. When evaluating the functional, these potentials are of the form $\psi_{C_i}(\mathbf{x}_c, \mathbf{y}_c) = \exp(\sum_t \alpha_t h_t(\mathbf{f}_{C_i}(\mathbf{x}_c, \mathbf{y}_c)))$. Briefly, this update places a different prior over the space of functions considered and encourages functions emphasizing few regions in feature space and little everywhere else. In our experiments, we found the exponentiated version works better in applications with a large number of features.

4.3. Learning robust potentials

So far we have only addressed learning models with associative clique potentials. In many applications, it is difficult to obtain a group of sites (*i.e.* a clique) that contain homogeneous labels. In the above framework, clique features will be discarded and not used in learning, even if 1 out of a clique of 50 pixels contains a different ground truth/inferred label. Kohli *et al.* [11] remedy this problem and extend the P^n Potts model into a robust version that enables a portion of the nodes to disagree with the mode clique label, yet still be solved with graphcut inference. The robustness of a clique c is defined as the number of nodes Q_c that disagree with the mode label, with the constraint that it is fewer than half the total nodes $2Q_c < |c|$. The clique's potential value is then proportional to the number of disagreeing nodes.

Since the features from cliques with inhomogeneous labels are not as representative of a class as features from cliques with homogeneous labels, we do not treat the two cases in the same way. Instead of regressing to the target value of 1 during learning, we regress to the value of the proportion of the nodes with the mode label in the clique. By doing this, we place less importance on the cliques with mixed labels but the information is used and not ignored during learning. We parameterize the robustness of cliques $c \in C_i$ by q_i where $Q_c = q_i|c|$. Algorithm 2 presents this modification.

Algorithm 1 Functional Gradient for Learning M^{3}N

```
Inputs: Labeled training data: $C = \{C_i\}_{i=1}^{S} = (\mathbf{x}, \widehat{\mathbf{y}}) = \{(\mathbf{x}_c, \widehat{\mathbf{y}}_c)\}_{c \in C}$, Possible training labels: $\{\ell_i\}_{i=1}^{K}$, Step size: $\alpha_t$, Number of iterations: $T$
Output: Learned M$^3$N model: $\Psi(\cdot, \cdot)$
for $t = 1 \dots T$ do
  $\mathbf{y}^* = \arg\max_{\mathbf{y}} \Psi(\mathbf{x}, \mathbf{y}) + \mathcal{L}(\mathbf{y}, \widehat{\mathbf{y}})$

  Initialize $\mathcal{D}^+ = \mathcal{D}^- = \emptyset$
  for $C_i \in \{C_1, \dots, C_S\}$ do
    for $c \in C_i$ do
      $\widehat{\ell_{mode}} = \mathtt{mode}(\widehat{\mathbf{y}}_c)$
      $\ell^*_{mode} = \mathtt{mode}(\mathbf{y}^*_c)$
      if $\widehat{\ell_{mode}} \neq \ell^*_{mode}$ then
        $\widehat{g} = \mathtt{calcFuncGradContribution}(\widehat{\ell_{mode}}, \widehat{\mathbf{y}}_c)$
        $g^* = \mathtt{calcFuncGradContribution}(\ell^*_{mode}, \mathbf{y}^*_c)$
        if $\widehat{g} > 0$ then
          $\mathcal{D}^+ \leftarrow \mathcal{D}^+ \cup \{(\mathbf{f}_{C_i}(\mathbf{x}_c, \widehat{\ell_{\mathbf{mode}}}),\ +\widehat{g})\}$
        end if
        if $g^* > 0$ then
          $\mathcal{D}^- \leftarrow \mathcal{D}^- \cup \{(\mathbf{f}_{C_i}(\mathbf{x}_c, \ell^*_{\mathbf{mode}}),\ -g^*)\}$
        end if
      end if
    end for
  end for

  $h_t \leftarrow \mathtt{trainRegressor}(\mathcal{D}^+, \mathcal{D}^-)$
  $\Psi \leftarrow \Psi + \alpha_t h_t$    (or $\Psi \leftarrow \Psi \cdot \exp(\alpha_t h_t)$)
end for
return $\Psi$
```

Algorithm 2 $\mathtt{calcFuncGradContribution}(\ell, \mathbf{y}_c)$

```
Inputs: Label of interest: $\ell$, List of labels of nodes in a clique: $\mathbf{y}_c = \{y_i\}_{i \in c}$
Output: $[0, 1]$ value indicating how much to contribute to the functional gradient, depending on the interaction model and how many nodes match $\ell$.
Define $p = \sum_{i \in c} \delta(y_i = \ell)$    // Count the nodes labeled $\ell$
if Pott's model then
  $(p = |c|)$ ? return $1$ : return $0$
end if
if Robust Pott's model then
  $Q = q|c|$    // q is a robust truncation parameter
  $(p > (|c| - Q))$ ? return $\frac{p}{|c|}$ : return $0$
end if
```

5. Experiments

We compare the performances of M^{3}N models trained with the subgradient method and functional gradient boosting on two vision problems: 1) 3-D point cloud classification from laser range finders and 2) 3-D geometric surface estimation from images [8]. With the first experiment we improve the state-of-the-art for 3-D point classification, an application known to work well with AMNs. In the second experiment, we demonstrate the generality of the approach to an application where joint learning of the random field's potentials has not been typically done.

For both experiments, $T = 300$ and we estimated the learning parameters (step size and regularization) for the algorithms by classifying a validation dataset and picking the parameters that resulted in the best precision rate for the worst performing label.

5.1. 3-D point cloud classification

AMNs have been demonstrated to work well on the task of 3-D point cloud classification [1, 28, 16]. We created a dataset of 1.3 million hand-labeled points that were collected from a laser scanner; we split the data into sets of $100,000$ points to facilitate a validation set and graph-cut inference. We plan to publicly release this dataset as a benchmark. The classification task is to assign each 3-D point a label from the set $\{$*wire, vegetation, ground, pole/tree-trunk, facade*$\}$.

Features. We implemented the features described in Munoz *et al.* [17]. These features describe the local point cloud topology around a point by assigning a saliency spectral feature describing the degree of scatter, linearity, and planarity of the neighboring points within a fixed radius. Direction features that capture the orientation of these locally estimated geometric objects are also used. The final type of feature uses a 2.5-D elevation map to estimate a point's distance off the ground.

Random field structure. We create models using pairwise and high-order interactions. Each model uses the pairwise model described in [16] as a base configuration; we then add the elevation features to each clique-set's feature set. For the high-order models the cliques are defined to be the resulting clusters from two k-means segmentations [9] over each nodes' spectral features and locations. K-means segmentation provides a fast method to cluster scenes with over 100,000 points at a time. We experimentally chose $k_1 = 0.026N$ and $k_2 = 0.042N$ by visual inspection of the training data with the objective of producing relatively compact clusters containing one label. The clique features are the same type as computed for the nodes, except now the neighborhood volume is defined within a 1.0 m radius from the clique centroid.

Models. We compare the performances of three AMN models: a pairwise model trained with the parametric subgradient method (`PAIR-SUB`), and two models with high-order cliques trained with the parametric (`HOC-SUB`) and functional (`HOC-FUNC`) gradient techniques. The `HOC-FUNC` model was trained using the non-exponentiated version of the algorithm. We found that this produces slightly better results than the exponentiated version and attribute this to the relatively small number of features (at most 11 dimensions) used to train the individual regressors.

Results. We trained our models on a point cloud with $37,000$ points and determined the learning parameters by classifying a validation point cloud of $100,000$ points. Our test set consisted of 1.3 million points. The overall accuracies of the `HOC-FUNC`, `HOC-SUB` and `PAIR-SUB` models are respectively $\mathbf{97.2}\%$, 96.1%, and 95.7%. Though the overall accuracy improvement is small, this can be attributed to the law of diminishing returns and the overwhelming number of *vegetation, ground, facade* labels present in the data. However, Table 1 shows that there is a clear precision rate improvement with `HOC-FUNC` over the previous learning algorithm `PAIR-SUB` for the smaller labels *wire, pole/tree-trunk*. The *Macro-average Modified* [4] precision rate provides a metric to account for unbalanced classes when evaluating performance. Using the values in Table 1, the rates for the `HOC-FUNC`, `HOC-SUB` and `PAIR-SUB` models are respectively $\mathbf{71.5}\%$, 64.3%, 63.4%, demonstrating that the functional gradient optimized the learning objective better than subgradient method with the high-order model. In addition, there is noticeable improved recall rate with `HOC-FUNC` for the *vegetation* label without `HOC-FUNC` sacrificing performance with the other labels. Figure 2 qualitatively demonstrates this behavior of `HOC-FUNC` producing improved classifications in challenging areas.

In principle, one would expect the robust potentials to improve performance as there will always be segments spanning multiple objects. However, in this application we found that the number of such regions is small compared to the number of regions with noisy data, as evident in Figure 2. The robust potentials caused a decrease in performance because they allowed the noisy points to maintain the wrong labels. The next experiment will demonstrate how robust potentials can be used to improve classification when there are many more segments containing inhomogeneous labels.

5.2. 3-D surface estimation

We applied this contextual learning approach to the problem of recovering 3-D geometric surfaces from from images, *i.e.*, determining whether a pixel belongs on one of three classes: *Ground, Vertical (object standing on the ground), Sky*. We use the Geometric Context Dataset provided by Hoiem *et al.* [8], where the overall distribution of pixels per class is: *Ground* $= 31.4\%$, *Vertical* $= 48.0\%$, *Sky* $= 20.6\%$. In [8], the authors extract features from overlapping segments in the image and average the independent classifications from boosted decision tree classifiers. In this experiment we compare the method used in [8] with the M^{3}N model using the two learning techniques. We note that recent work from Saxena *et al.* [21], have also used random fields to extract 3-D information from single images. However, our model is not best suited in that scenario since we perform discrete label classification and they are estimating continuous distances.

Features. In [8], the authors first group pixels into superpixels to use as the sites in their classification. They then perform 15 different segmentations: by increments of 5 segments between 5 and 50, and then by increments of 10 segments until 100. Each segment is composed of a group of superpixels. For each superpixel and segment, they extract 50 and 94 features, respectively, which capture location, shape, color, texture, and perspective. We implemented these same features.

Random field structure. In the M^{3}N model, the nodes represent superpixels and the segments are the cliques. As the sizes of the segments from segmentation-5 will differ with those from segmentation-100, the types of computed features will also vary with respect to the segment size. We model how these features vary per clique by creating different potentials based on segment size. In addition to the nodes, we defined three clique-sets. Clique-set 1 (C_1) contains the cliques from the five largest segmentations. Clique-set 2 (C_2) contains the cliques from the middle five segmentations. Clique-set 3 (C_3) contains the cliques from the five smallest segmentations. Hence, the overall number of parameters we learn for this model is $996 = 3 \times 50 + 3(94 \times 3)$.

Models. We evaluate three different M^{3}N models and compare with the original learning algorithm of [8]. The first model is learned using the subgradient method with Potts model interactions (`POTTS-HOC-SUB`). The second model uses the functional gradient learning with Potts model interactions (`POTTS-HOC-FUNC`). Because many cliques in this framework will contain multiple labels, we train the final model using functional gradient learning with Robust Potts interactions (`ROBUST-HOC-FUNC`). We define the robustness of the clique-sets corresponding to the five largest (C_1) and five second largest (C_2) segments to allow 20% ($q_1 = 0.2$) and 10% ($q_2 = 0.1$), respectively, of the nodes within a clique to disagree with the mode label. We use a Potts model ($q_3 = 0$) for the clique-set with the smallest segments (C_3). In contrast to the experiment in Section 5.1, because the number of clique features is an order of magnitude larger than in the point cloud application, we found that we get better performance by training the functional gradient models with the exponentiated version of the algorithm. Finally, we make a small modification to the functional gradient algorithm. Since the size of the segment can be an indicator of the quality of features extracted, we weight the clique features during regression proportional to the number of pixels contained in the clique.

Results. In [8], the authors evaluate their algorithm by performing five-fold classification on a dataset of 250 images where 200 random images are chosen for training and the remaining 50 are used for evaluation. Since we need to determine the learning rate for our algorithm, we validate on a random 50 image subset and train on the remaining 150 images. In Table 2 (row-normalized), we give the quantitative classification results of the above three models. We immediately see a clear improvement in performance across all classes using the new `POTTS-HOC-FUNC` model over the previous `POTTS-HOC-SUB` model. By incorporating

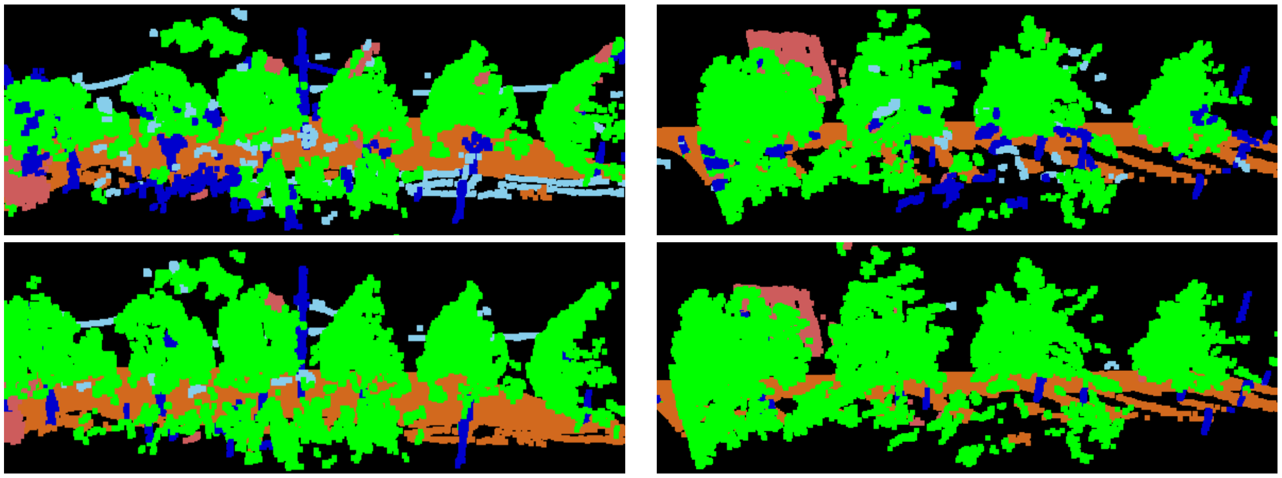
Figure 2. Two classification comparisons for 3-D point cloud classification: `HOC-SUB` (top row) and `HOC-FUNC` (bottom row).

		Inferred label ↓					Recall		
	HOC-FUNC	vegetation	wire	pole/tree trunk	ground	facade	HOC-FUNC	HOC-SUB	PAIR-SUB
True label $\longrightarrow$	vegetation	240861	885	4620	3475	8101	**0.93**	0.88	0.87
	wire	202	3126	135	0	16	**0.90**	**0.90**	0.89
	pole/tree trunk	789	0	6150	222	391	0.81	0.80	**0.83**
	ground	1408	0	2592	926515	654	$>$ **0.99**	0.99	0.99
	facade	1302	2231	9973	175	96147	0.88	**0.89**	0.88
Precision	HOC-FUNC	**0.99**	**0.50**	**0.26**	$>$ **0.99**	**0.91**			
	HOC-SUB	**0.99**	0.26	0.22	$>$ **0.99**	0.88			
	PAIR-SUB	0.98	0.26	0.18	$>$ **0.99**	0.90			

Table 1. Classification confusion matrix over 1.3 million point cloud using functional gradient with high-order interactions (`HOC-FUNC`). Precision and recall values for `HOC-SUB` and `PAIR-SUB` models are also provided.

the robust potentials (`ROBUST-HOC-FUNC`), we can further improve the performance of each class, especially the preservation of the smallest class *Sky*, and achieve comparable accuracy with [8]. Visual comparisons of the four models are presented in Figure 3. While we do not improve upon the results of [8], it is worth noting that our model uses linear hypotheses in comparison to [8] who has the benefits of nonlinear hypotheses; investigating non-linear regressors in our contextual model is ongoing work.

5.3. Discussion

At first glance it may appear that this improvement is solely due faster convergence since both models use linear potentials. However, we expect a difference in optima because they optimize the objective over difference spaces and with different regularization. A way to see that is to note that the parametric objective (Equation 2) is not invariant to rescaling a single feature [7]. Were we to scale a random feature dimension by 1,000, that scaled feature would essentially determine the margin and the parametric optima would converge to a different solution, which is not the equivalent rescaled solution (and most likely worse). However, since the functional gradient parameterization is terms of the regressors and not the features, the solution is manifestly invariant to the scale.

6. Conclusion

In this paper, we adapted the functional gradient technique to learn M^3N models in order to perform discrete, multi-label classification. With this formulation, we demonstrated with two distinct applications that we can learn improved high-order models than with the previous learning method. Additionally, we showed how the model can be incorporated with robust potentials to preserve less dominant labels. Future work will investigate using non-linear potentials.

Acknowledgments

Prepared through collaborative participation in the Robotics consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-209912. The first author was partially supported by a Siebel Scholars Fellowship. We thank N. Ratliff, J.-F. Lalonde, J. Huang, and the reviewers for their comments, and S. K. Divvala for help with the geometric surface experiments.

References

[1] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3-d scan data. In *CVPR*, 2005.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *T-PAMI*, 23(1), 2001.

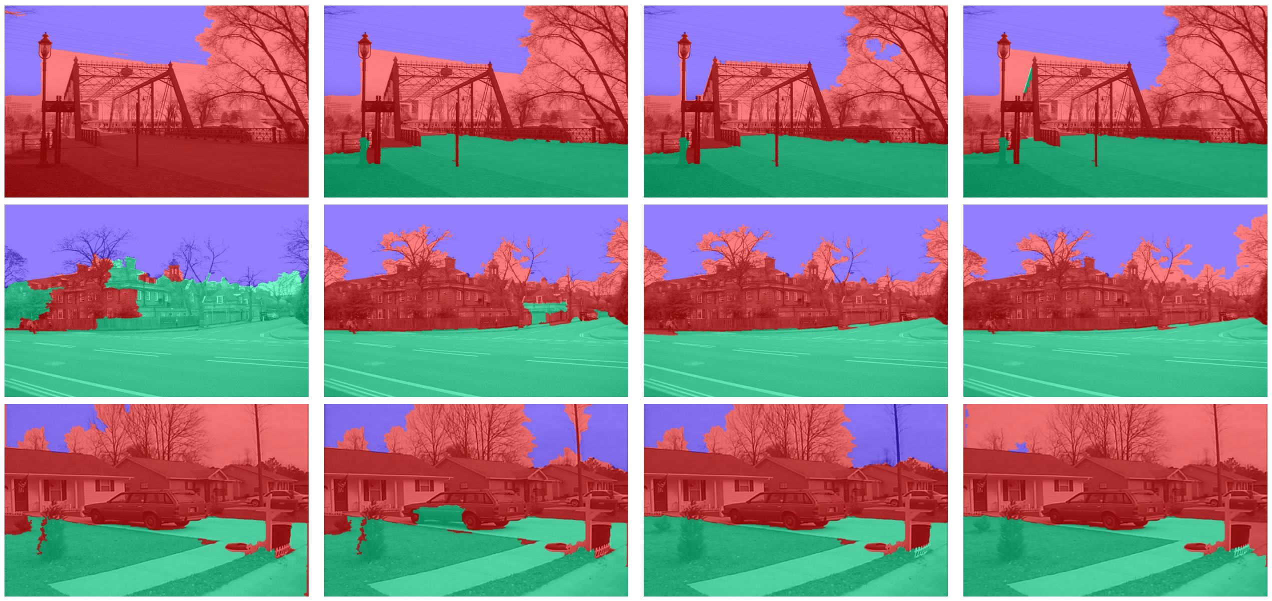

Figure 3. Three classification comparisons (one per row) for geometric surface estimation. Models used (from left to right): `POTTS-HOC-SUB`, `POTTS-HOC-FUNC`, `ROBUST-HOC-FUNC`, Hoiem *et al.* [8].

	POTTS-HOC-SUB			POTTS-HOC-FUNC			ROBUST-HOC-FUNC			Hoiem *et al.* [8]		
	Ground	Vertical	Sky	Ground	Vertical	Sky	Ground	Vertical	Sky	Ground	Vertical	Sky
Ground	0.74	0.24	0.02	0.84	0.15	0.01	**0.85**	0.14	0.01	0.83	0.16	0.00
Vertical	0.24	0.70	0.07	0.13	0.83	0.04	0.13	0.84	0.04	0.09	**0.89**	0.02
Sky	0.03	0.20	0.78	0.02	0.07	0.91	0.01	0.05	**0.94**	0.00	0.10	0.89
	Accuracy: 72.8%			Accuracy: 84.9%			Accuracy: 86.0%			Accuracy: **87.1**%		

Table 2. Overall confusion matrices (row normalized) from 5-fold cross validation comparing models for geometric surface estimation.

[3] T. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *ICML*, 2004.

[4] C. Ferri, J. Hernandez-Orallo, and M. Salido. Volume under the roc surface for multi-class problems. In *ECML*, 2003.

[5] V. Franc and B. Savchynskyy. Discriminative learning of max-sum classifiers. *JMLR*, 9(1), 2008.

[6] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[7] R. Herbrich and T. Graepel. A pac-bayesian margin bound for linear classifiers: Why svms work. In *NIPS*, 2001.

[8] D. Hoiem, A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 75(1), 2007.

[9] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *T-PAMI*, 24(7), 2002.

[10] P. Kohli, M. Kumar, and P. Torr. P3 and beyond: Solving energies with higher order cliques. In *CVPR*, 2007.

[11] P. Kohli, L. Ladicky, and P. Torr. Robust higher order potentials for enforcing label consistency. In *CVPR*, 2008.

[12] S. Kumar, J. August, and M. Hebert. Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. In *EMMCVPR*, 2005.

[13] S. Kumar and M. Hebert. Discriminative random fields. *IJCV*, 2006.

[14] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[15] L. Liao, T. Choudhury, D. Fox, and H. Kautz. Training conditional random fields using virtual evidence boosting. In *IJCAI*, 2007.

[16] D. Munoz, N. Vandapel, and M. Hebert. Directional amn for 3-d point cloud classification. In *3DPVT*, 2008.

[17] D. Munoz, N. Vandapel, and M. Hebert. Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. In *ICRA*, 2009.

[18] N. Ratliff, J. Bagnell, and S. Srinivasa. Imitation learning for locomotion and manipulation. In *Humanoids*, 2007.

[19] N. Ratliff, J. Bagnell, and M. Zinkevich. Online subgradient methods for structured prediction. In *AISTATS*, 2007.

[20] N. Ratliff, D. Bradley, J. Bagnell, and J. Chestnutt. Boosting structured prediction for imitation learning. In *NIPS*, 2007.

[21] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3-d scene structure from a single still image. *T-PAMI*, 31(5), 2008.

[22] D. Silver, J. Bagnell, and A. Stentz. High performance outdoor navigation from overhead data using imitation learning. In *RSS*, 2008.

[23] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappena, and C. Rother. A comparative study of energy minimization methods for markov random fields. *T-PAMI*, 30(6), 2007.

[24] M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. In *ECCV*, 2008.

[25] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *ICML*, 2004.

[26] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2003.

[27] A. Torralba, K. Murphy, and W. Freeman. Contextual models for object detection using boosted random fields. In *NIPS*, 2004.

[28] R. Triebel, R. Schmidt, O. M. Mozos, and W. Burgard. Instance-based amn classification for improved object recognition in 2d and 3d laser range data. In *IJCAI*, 2007.

[29] J. Yuan, J. Li, and B. Zhang. Scene understanding with discriminative structured prediction. In *CVPR*, 2008.