Learning Message-Passing Inference Machines for Structured Prediction

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

Abstract

Nearly every structured prediction problem in computer vision requires approximate inference due to large and complex dependencies among output labels. While graphical models provide a clean separation between modeling and inference, learning these models with approximate inference is not well understood. Furthermore, even if a good model is learned, predictions are often inaccurate due to approximations. In this work, instead of performing inference over a graphical model, we instead consider the inference procedure as a composition of predictors. Specifically, we focus on message-passing algorithms, such as Belief Propagation, and show how they can be viewed as procedures that sequentially predict label distributions at each node over a graph. Given labeled graphs, we can then train the sequence of predictors to output the correct labelings. The result no longer corresponds to a graphical model but simply defines an inference procedure, with strong theoretical properties, that can be used to classify new graphs. We demonstrate the scalability and efficacy of our approach on 3D point cloud classification and 3D surface estimation from single images.

1. Introduction

Probabilistic graphical models, such as Conditional Random Fields (CRFs) [11], have proven to be a remarkably successful tool for structured prediction that, in principle, provide a clean separation between modeling and inference. However, exact inference for problems in computer vision (e.g., Fig. 1) is often intractable due to a large number of dependent output variables (e.g., one for each (super)pixel). In order to cope with this problem, inference inevitably relies on approximate methods: Monte-Carlo, loopy belief propagation, graph-cuts, and variational methods [16, 2, 23]. Unfortunately, learning these models with approximate inference is not well understood [9, 6]. Additionally, it has been observed that it is important to tie the graphical model to the specific approximate inference procedure used at test time to obtain better predictions [10, 22].

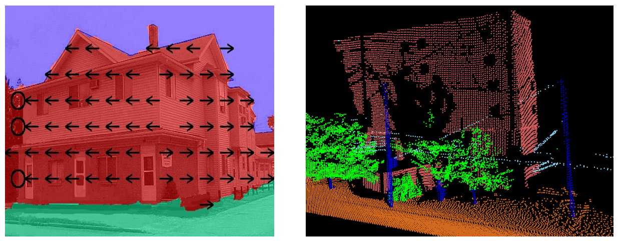
Figure 1: Applications of structured prediction in computer vision. Left: 3D surface layout estimation. Right: 3D point cloud classification.

When the learned graphical model is tied to the inference procedure, the graphical model is not necessarily a good probabilistic model of the data but simply a parametrization of the inference procedure that yields the best predictions within the "class" of inference procedures considered. This raises an important question: if the ultimate goal is to obtain the best predictions, then why is the inference procedure optimized indirectly by learning a graphical model? Perhaps it is possible to optimize the inference procedure more directly, without building an explicit probabilistic model over the data.

Some recent approaches [4, 21] eschew the probabilistic graphical model entirely with notable successes. However, we would ideally like to have the best of both worlds: the proven success of error-correcting iterative decoding methods along with a tight relationship between learning and inference. To enable this combination, we propose an alternate view of the approximate inference process as a long sequence of computational modules to optimize [1] such that the sequence results in correct predictions. We focus on message-passing inference procedures, such as Belief Propagation, which compute marginal distributions over output variables by iteratively visiting all nodes in the graph and passing messages to neighbors which consist of "cavity marginals", i.e., a series of marginals with the effect of each neighbor removed. Message-passing inference can be viewed as a function applied iteratively to each variable that takes as input local observations/features and local computations on the graph (messages) and provides as output the intermediate messages/marginals. Hence, such a procedure can be trained directly by training a predictor which predicts a current variable's marginal[1] given local features and a subset of neighbors' cavity marginals. By training such a predictor, there is no need to have a probabilistic graphical model of the data, and there need not be any probabilistic model that corresponds to the computations performed by the predictor. The inference procedure is instead thought of as a black box function that is trained to yield correct predictions. This is analogous to many discriminative learning methods; it may be easier to simply discriminate between classes than build a generative probabilistic model of them.

There are a number of advantages to doing message-passing inference as a sequence of predictions. Considering different classes of predictors allows one to obtain entirely different classes of inference procedures that perform different approximations. The level of approximation and computational complexity of the inference can be controlled in part by considering more or less complex classes of predictors. This allows one to naturally trade-off accuracy versus speed of inference in real-time settings. Furthermore, in contrast with most approaches to learning inference, we are able to provide rigorous reduction-style guarantees [18] on the performance of the resulting inference procedure.

Training such a predictor, however, is non-trivial as the interdependencies in the sequence of predictions make global optimization difficult. Building from success in deep learning, a first key technique we use is to leverage information local to modules to aid learning [1]. Because each module's prediction in the sequence corresponds to the computation of a particular variable's marginal, we exploit this information and try to make these intermediate inference steps match the *ideal* output in our training data (i.e., a marginal with probability 1 to the correct class). To provide good guarantees and performance in practice in this non-i.i.d. setting (as predictions are interdependent), we also leverage key iterative training methods developed in prior work for imitation learning and structured prediction [17, 18, 4]. These techniques allow us to iteratively train probabilistic predictors that predict the ideal variable marginals *under the distribution of inputs the learned predictors induce during inference*. Optionally, we may refine performance using an optimization procedure such as back-propagation through the sequence of predictions in order to improve the overall objective (i.e., minimize loss of the final marginals).

In the next section, we first review message-passing inference for graphical models. We then present our approach for training message-passing inference procedures in Sec. 3. In Sec. 4, we demonstrate the efficacy of our proposed approach by demonstrating state-of-the-art results on a large 3D point cloud classification task as well as in estimating geometry from a single image.

[1]For lack of a better term, we will use marginal throughout to mean a distribution over one variable's labels.

2. Graphical Models and Message-Passing

Graphical models provide a natural way of encoding spatial dependencies and interactions between neighboring sites (pixels, superpixels, segments, etc.) in many computer vision applications such as scene labeling (Fig. 1). A graphical model represents a joint (conditional) distribution over labelings of each site (node), via a factor graph (a bipartite graph between output variables and factors) defined by a set of variable nodes (sites) V, a set of factor nodes (potentials) F and a set of edges E between them:

$$P(Y|X) \propto \prod_{f \in F} \phi_f(x_f, y_f),$$

where X, Y are the vectors of all observed features and output labels respectively, x_f the features related to factor F and y_f the vector of labels for each node connected to factor F. A typical graphical model will have node potentials (factors connected to a single variable node) and pairwise potentials (factors connected between 2 nodes). It is also possible to consider higher order potentials by having a factor connecting many nodes (e.g., cluster/segment potentials as in [14]). Training a graphical model is achieved by optimizing the potentials ϕ_f on an objective function (e.g., margin, pseudo-likelihood, etc.) defined over training data. To classify a new scene, an (approximate) inference procedure estimates the most likely joint label assignment or marginals over labels at each node.

Loopy Belief Propagation (BP) [16] is perhaps the canonical message-passing algorithm for performing (approximate) inference in graphical models. Let N_v be the set of factors connected to variable v, N_v^{-f} the set of factors connected to v except factor f, N_f the set of variables connected to factor f and N_f^{-v} the set of variables connected to f except variable v. At a variable $v \in V$, BP sends a message m_{vf} to each factor f in N_v:

$$m_{vf}(y_v) \propto \prod_{f' \in N_v^{-f}} m_{f'v}(y_v),$$

where $m_{vf}(y_v)$ denotes the value of the message for assignment y_v to variable v. At a factor $f \in F$, BP sends a message m_{fv} to each variable v in N_f:

$$m_{fv}(y_v) \propto \sum_{y'_f | y'_v = y_v} \phi_f(y'_f, x_f) \prod_{v' \in N_f^{-v}} m_{v'f}(y'_{v'}),$$

where y'_f is an assignment to all variables v' connected to f, $y'_{v'}$ is the particular assignment to v' (in y'_f), and ϕ_f is the potential function associated to factor f which depends on

y'_f and potentially other observed features x_f (e.g., in the CRF). Finally the marginal of variable v is obtained as:

$$P(v = y_v) \propto \prod_{f \in N_v} m_{fv}(y_v).$$

The messages in BP can be sent synchronously (i.e., all messages over the graph are computed before they are sent to their neighbors) or asynchronously (i.e., by sending the message to the neighbor immediately). When proceeding asynchronously, BP usually starts at a random variable node, with messages initialized uniformly, and then proceeds iteratively through the factor graph by visiting variables and factors in a breath-first-search manner (forward and then in backward/reverse order) several times or until convergence. The final marginals at each variable are computed using the last equation. Asynchronous message passing often allows faster convergence and methods such as Residual BP [5] have been developed to achieve still faster convergence by prioritizing the messages to compute.

2.1. Understanding Message Passing as Sequential Probabilistic Classification

By definition of $P(v = y_v)$, the message m_{vf} can be interpreted as the marginal of variable v when the factor f (and its influence) is removed from the graph. This is often referred as the cavity method in statistical mechanics [3] and m_{vf} are known as cavity marginals. By expanding the definition of m_{vf}, we can see that it may depend only on the messages $m_{v'f'}$ sent by all variables v' connected to v by a factor $f' \neq f$:

$$m_{vf}(y_v) \propto \prod_{f' \in N_v^{-f}} \sum_{y'_{f'} | y'_v = y_v} \phi_{f'}(y'_{f'}, x_{f'}) \prod_{v' \in N_{f'}^{-v}} m_{v'f'}(y'_{v'}). \quad (1)$$

Hence the messages m_{vf} leaving a variable v toward a factor f in BP can be thought as the classification of the current variable v (marginal distribution over classes) using the cavity marginals $m_{v'f'}$ sent by variables v' connected to v through a factor $f' \neq f$. In this view, BP is iteratively classifying the variables in the graph by performing a sequence of classifications (marginals) for each message leaving a variable. The final marginals $P(v = y_v)$ are then obtained by classifying v using all messages from all variables v' connected to v through some factor $f \in N_v$.

An example of how BP unrolls to a sequence of interdependent local classifications is shown in Fig. 2 for a simple graph. In this view, the job of the predictor is not only to emulate the computation going on during BP at variable nodes, but also emulate the computations going on at all the factors connected to the variable which it is not sending the message to, as shown in Fig. 3. During inference BP effectively employs a probabilistic predictor that has the form in Equation 1, where the inputs are the messages $m'_{v'f'}$

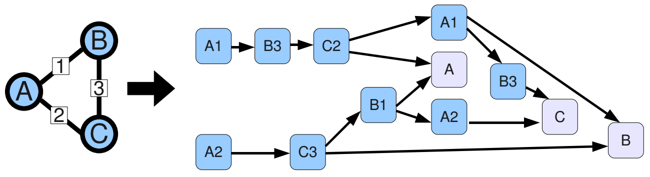

Figure 2: Depiction of how BP unrolls into a sequence of predictions for 3 passes on the graph on the left with 3 variables (A,B,C) and 3 factors (1,2,3), starting at A. Sequence of predictions on the right, where e.g., A1 denotes the prediction (message) of A sent to factor 1, while the output (final marginals) are in gray and denoted by the corresponding variable letter. Input arrows indicate the previous outputs that are used in the computation of each message.

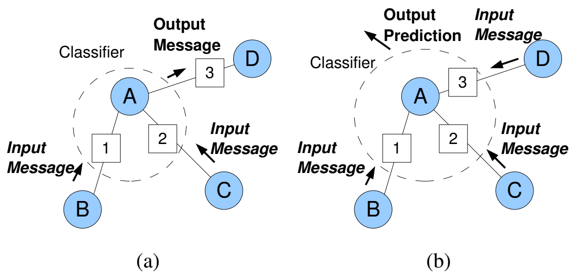

Figure 3: Depiction of the computations that the predictor represents in BP for (a) a message to a neighboring factor and (b) the final marginal of a variable outputed by BP.

and local observed features $x_{f'}$. Training graphical models can be understood as training a message-passing algorithm with a particular class of predictors defined by Equation 1, which have as parameters the potential functions ϕ_f. Under this general view, there is no reason to restrict attention to only predictors of the form of Equation 1. We now have the possibility of using different classes of predictors (e.g., Logistic Regression, Boosted Trees, Random Forests, etc.) whose inductive bias may more efficiently represent interactions between neighboring variables or in some cases be more compact and faster to compute, which is important in real-time settings.

Many other techniques for approximate inference have been framed in message-passing form. Tree-Weighted BP [23] and convergent variants follow a similar pattern to BP as described above but change the specific form of messages sent to provide stronger performance and convergence guarantees. These can also be interpreted as performing a sequence of probabilistic classifications, but using a different form of predictors. The classical "mean-field" (and more generally variational methods [15]) method is easily framed as a simpler message passing strategy where, instead of cavity marginals, algorithms pass around marginals or expected sufficient statistics which is usually more efficient but obtains lower performance then cavity message passing [15]. We also consider training such mean-field inference approach in the experiments.

3. Learning Message-Passing Inference

Training the cavity marginals' predictors in the deep inference network described above remains a non-trivial task. As we saw in Fig. 2, the sequence of predictions forms a large network where a predictor is applied at each node in this network, similarly to a deep neural network. In general, minimizing the loss of the output of such a network is difficult since it is a non-convex optimization problem, because the outputs of previous classifications are used as input for following classifications. However here there are several differences that make this training an easier problem than training general networks. First, the number of parameters is small as we assume that the same predictor is used at every node in this large network. Additionally, we can exploit local sources of information (i.e., the variables target labels) to train the "hidden layer" nodes of this network. Because each node corresponds to the computation of a particular variable's marginal, we can always try to make these marginals match the ideal output in our training data (i.e., a marginal with probability 1 to the correct class).

Hence our general strategy for optimizing the message-passing procedure will be to first use the local information to train a predictor (or sequence of predictors) that predicts the ideal variable marginals (messages) under the distribution of inputs it encounters during the inference process. We refer to this step as local training. For cases where we train a differentiable predictor, we can use the local training procedure to obtain a good starting point, and seek to optimize the global non-convex objective (i.e., minimize logistic loss only on the final marginals) using a descent procedure (i.e., back-propagation through this large network). We refer to this second step as global training. The local training step is still non-trivial as it corresponds to a non-i.i.d. supervised learning problem, i.e., previous classifications in the network influence the future inputs that the predictor is evaluated on. As all statistical learning approaches assume i.i.d. data, it has been shown that typical supervised learning approaches have poor performance guarantees in this setting. Fortunately, recent work [17, 18, 4] have presented iterative training algorithms that can provide good guarantees. We leverage these techniques below and present how these techniques can be used in our setting.

3.1. Local Training for Synchronous Message-Passing

In synchronous message-passing, messages from nodes to their neighbors are only sent once all messages at each node have been computed. Our goal is to train a predictor that performs well under the distribution of inputs (features and messages) induced by the predictors used at previous passes of inference. A strategy that we analyzed previously in [17] to optimize such a network of modules is to simply train a sequence of predictors, rather than a single predictor, and train the predictors in sequence starting from the first one. At iteration n, the previously learned predictors can be used to generate inputs for training the n^{th} predictor in the sequence. This guarantees that each predictor is trained under the distribution of inputs it expects to see at test time.

```
**for** $n = 1$ **to** $N$ **do**
    Use $h_{1:n-1}$ to perform synchronous message-passing
    on training graphs up to pass $n$.
    Get dataset $D_n$ of inputs encountered at pass $n$, with
    the ideal marginals as target.
    Train $h_n$ on $D_n$ to minimize a loss (e.g., logistic).
**end for**
**Return** the sequence of predictors $h_{1:N}$.
```

Algorithm 3.1: Forward Training Algorithm for Learning Synchronous Message-Passing.

In our synchronous message-passing scenario, this leads to learning a different predictor for each inference pass. The first predictor is trained to predict the ideal marginal at each node given no information (uniform distribution messages) from their neighbors. This predictor can then be used to perform a first pass of inference on all the nodes. The algorithm then iterates until a predictor for each inference pass has been trained. At the n^{th} iteration, the predictor is trained to predict the ideal node marginals at the n^{th} inference pass, given the neighbors' messages obtained after applying the previously learned $n-1$ predictors on the training graphs (scenes) for $n-1$ inference passes (Algorithm 3.1).

In [17, 18], we have showed that this forward training procedure guarantees that the expected sum of the loss in the sequence is bounded by $N\bar{\epsilon}$, where $\bar{\epsilon}$ is the average true loss of the learned predictors $h_{1:N}$. In our scenario, we are only concerned with the loss at the last inference pass. Unfortunately, applying naively this guarantee would tell us that the expected loss at the last pass is bounded by $N\bar{\epsilon}$ (e.g., in the worst case where all the loss occurs at the last pass) and would suggest that *fewer* inference passes is better (making N small). However, for convex loss functions, such as the logistic loss, simply averaging the output node marginals at each pass, and using those average marginals as final output, guarantees[2] achieving loss no worse than $\bar{\epsilon}$. Hence, using those average marginals as final output enables using an arbitrary number of passes to ensure we can effectively find the best decoding.

[2]If f is convex and $\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p_i$, then $f(\bar{p}) \leq \frac{1}{N}\sum_{i=1}^{N} f(p_i)$.

Some recent work is related to our approach. [19] demonstrates that constrained simple classification can provide good performance in NLP applications. The technique of [21] can be understood as using forward training on a synchronous message passing using only marginals, similar to mean-field inference. Similarly, from our point of view, [13] implements a "half-pass" of hierarchical mean-field message passing by descending once down a hierarchy making contextual predictions. We demonstrate in our experiments the benefits of enabling more general (BP-style) message passing.

3.2. Local Training for Asynchronous Message-Passing

In asynchronous message-passing, messages from nodes to their neighbors are sent immediately. This creates a long sequence of dependent messages that grows with the number of nodes, in addition to the number of inference passes. Hence the previous forward training procedure is impractical in this case for large graphs, as it requires training a large number of predictors. Fortunately, an iterative approach called Dataset Aggregation (DAgger) [18] that we developed in prior work can train a single predictor to produce all predictions in the sequence and still guarantees good performance on its induced distribution of inputs over the sequence. For our asynchronous message-passing setting, DAgger proceeds as follows. Initially inference is performed on the training graphs by using the ideal marginals from the training data to classify each node and generate a first training distribution of inputs. The dataset of inputs encountered during inference and target ideal marginals at each node is used to learn a first predictor. Then the process keeps iterating by using the previously learned predictor to perform inference on the training graphs and generate a new dataset of encountered inputs during inference, with the associated ideal marginals. This new dataset is aggregated to the previous one and a new predictor is trained on this aggregated dataset (i.e., containing all data collected so far over all iterations of the algorithm). This algorithm is summarized in Algorithm 3.2. [18] showed that for strongly convex losses, such as regularized logistic loss, this algorithm has the following guarantee:

Theorem 3.1. *[18] There exists a predictor h_n in the sequence $h_{1:N}$ such that $\mathbb{E}_{x \sim d_{h_n}}[\ell(x, h_n)] \leq \epsilon + \tilde{O}(\frac{1}{N})$, for $\epsilon = \text{argmin}_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{x \sim d_{h_i}}[\ell(x, h)]$.*

d_h is the inputs distribution induced by predictor h. This theorem indicates that DAgger guarantees a predictor that, when used during inference, performs nearly as well as when classifying the aggregate dataset. Again, in our case we can average the predictions made at each node over the inference passes to guarantee such final predictions would have an average loss bounded by $\epsilon + \tilde{O}(\frac{1}{N})$. To make the factor $\tilde{O}(\frac{1}{N})$ negligible when looking at the sum of loss over the whole graph, we can choose N to be on the order of the number of nodes in a graph. Though in practice, often much smaller number of iterations ($N \in [10, 20]$), is sufficient to obtain good predictors under their induced distributions.

```
Initialize $D_0 \leftarrow \emptyset$, $h_0$ to return the ideal marginal on any variable $v$ in the training graph.
**for** $n = 1$ **to** $N$ **do**
    Use $h_{n-1}$ to perform asynchonous message-passing inference on training graphs.
    Get dataset $D'_n$ of inputs encountered during inference, with their ideal marginal as target.
    Aggregate dataset: $D_n = D_{n-1} \cup D'_n$.
    Train $h_n$ on $D_n$ to minimize a loss (e.g., logistic).
**end for**
**Return** best $h_n$ on training or validation graphs.
```

Algorithm 3.2: DAgger Algorithm for Learning Asynchronous Message-Passing.

3.3. Global Training via Back-Propagation

In both synchronous and asynchronous approaches, the local training procedures provide rigorous performance bounds on the loss of the final predictions; however, they do not optimize it directly. If the predictors learned are differentiable functions, a procedure like back-propagation [12] make it possible to identify local optima of the objective (minimizing loss of the final marginals). As this optimization problem is non-convex and there are potentially many local minima, it can be crucial to initialize this descent procedure with a good starting point. The forward training and DAgger algorithms provide such an initialization. In our setting, Back-Propagation effectively uses the current predictor (or sequence of) to do inference on a training graph (forward propagation); then errors are back-propagated through the network of classification by rewinding the inference, successively computing derivatives of the output error with respect to parameters and input messages.

4. Experiments: Scene Labeling

To demonstrate the efficacy of our approach, we compared our performance to state-of-the-art algorithms on two labeling problems from publicly available datasets: (1) 3D point cloud classification from a laser scanner and (2) 3D surface layout estimation from a single image.

4.1. Datasets

3D Point Cloud Classification. We evaluate on the 3D point cloud dataset[3] used in [14]. This dataset consists of 17 full 3D laser scans (total of ∼1.6 million 3D points) of

[3]http://www.cs.cmu.edu/˜ vmr/datasets/oakland_3d/cvpr09/

an outdoor environment and contains 5 object labels: Building, Ground, Poles/Tree-Trunks, Vegetation, and Wires (see Fig. 1). We design our graph structure as in [14] and use the same features. The graph is constructed by linking each 3D point to its 5 nearest neighbors (in 3D space) and defining high-order cliques (clusters) over regions from two k-means clusterings over points. The features describe the local geometry around a point or cluster (linear, planar or scattered structure; and its orientation); as well as a 2.5-D elevation map. In [14], performance is evaluated on one fold where one scene is used for training, one for validation, and the remaining 15 are used for testing. In order to allow each method to better generalize across different scenes, we instead split the dataset into 2 folds with each fold containing 8 scans for testing and the remaining 9 scans are used for training and validation (we always keep the original training scan in both folds' training sets). We report overall performances on the 16 test scans.

3D Surface Layout Estimation. We also evaluate our approach on the problem of estimating the 3D surface layout from single images, using the Geometric Context Dataset[4] from Hoiem et al. [7]. In this dataset, the problem is to assign the 3D geometric surface labels to pixels in the image (see Fig. 1). This task can be viewed as a 3-class or 7-class labeling problem. In the 3-class case, the labels are Ground/Supporting Surface, Sky, and Vertical structures (objects standing on the ground), and in the 7-class case the Vertical class is broken down into 5 subclasses: Left-perspective, Center-perspective, Right-perspective, Porous, and Solid. We consider the 7-class problem. In [7] the authors use superpixels as the basic entities to label in conjunction with 15 image segmentations. Various features are computed over the regions which capture location and shape, color, texture, and perspective. Boosted decision trees are trained per segmentation and are combined to obtain the final labeling. In our approach, we define a graph over the superpixels, create edges between adjacent superpixels, and consider the multiple segmentations as high-order cliques. We use the same respective features and 5-fold evaluation as detailed in [7].

4.2. Approaches

Conditional Random Field (CRF). One baseline is a pairwise, Pott's CRF, trained using asynchronous Loopy BP to estimate the gradient of the partition function.

Max-Margin Markov Network (M^{3}N). Another is the high-order [8], associative M^{3}N [20] model from [14]; we use their implementation[5]. We analyzed linear models optimized with the parametric subgradient method (M^{3}N-P) and with functional subgradient boosting (M^{3}N-F).

Synchronous Mean-Field Inference Machine. We used our proposed approach to train a synchronous mean-field inference machine (MFIM) using the forward training procedure (Sect. 3.1). A simple logistic regressor is used to predict the node marginals (messages) at each pass. As this is a mean-field approach, at each node we predict a new marginal using all of its neighbors' messages (i.e., no cavity method). For the 3D point cloud dataset, the feature vectors are obtained by concatenating[6] the messages with the node, edge, and cluster 3D features. For 3D surface estimation, we defined the feature vectors similarly except that the edge/cluster features are averaged together instead of being concatenated.

Asynchronous BP Inference Machine. We also train an asynchronous belief propagation inference machine. In this case, inference starts at a random node and proceeds in breadth-first-search order and alternates between forward and backward order at consecutive passes. Again a simple logistic regressor to predict the node marginals (messages). We compare 3 different approaches for optimizing it. BPIM-D: DAgger[7] from Sect. 3.2. BPIM-B: Back-Propagation starting from a 0 weight vector. BPIM-DB: Back-Propagation starting from the predictor found with DAgger (only for point cloud dataset). For both datasets, the input feature vector is constructed exactly as for the synchronous mean-field inference machine when predicting the final classification of a node. However, when sending messages to neighbors and clusters the cavity method is used (i.e., the features related to the node/cluster we are sending a message to are removed from the concatenation/average).

4.3. Results

We measure the performance of each method in terms of per site (i.e., over points in the point cloud and superpixels in the image) accuracy, the Macro-F1 score (average of the per class F1 scores), and Micro-F1 score (weighted average, according to class frequency, of the per class F1 scores). Table 1 summarizes the results for each approach and dataset.

3D Point Cloud Classification. We observe that the best approach overall is the functional gradient M^{3}N approach of [14]. We believe that for this particular dataset this is due to the use of a functional gradient method, which is less affected by the scaling of features and large class imbalance in this dataset. We can observe that when using the regular parametric subgradient M^{3}N approach, the performance is slightly worse than our inference machine approach, also optimized via parametric gradient descent. Hence using a functional gradient approach when training

[4]http://www.cs.illinois.edu/homes/dhoiem/projects/data.html

[5]http://www-2.cs.cmu.edu/~ vmr/software/software.html

[6]We concatenate up to 20 neighbors, ordered by physical distance, and append with zeros if there are less than 20 neighbors

[7]DAgger is used for 30 iterations, and the predictor that has the lowest error rate (from performing message-passing inference) on the training scenes is returned as the best one.

	3D Point Clouds				3D Surface Layout		
	Accuracy	Macro-F1	Micro-F1		Accuracy	Macro-F1	Micro-F1
BPIM-D	0.9795	0.8206	0.9799		**0.6467**	**0.5971**	**0.6392**
BPIM-B	0.9728	0.6504	0.9706		0.6287	0.5705	0.6149
BPIM-DB	**0.9807**	0.8305	**0.9811**		-	-	-
MFIM	**0.9807**	**0.8355**	**0.9811**		0.6378	0.5947	0.6328
CRF	0.9750	0.8067	0.9751		0.6126	0.5369	0.5931
M^3N-F	**0.9846**	**0.8467**	**0.9850**		0.6029	0.5541	0.6001
M^3N-P	0.9803	0.8230	0.9806		-	-	-
[7]	-	-	-		**0.6424**	**0.6057**	**0.6401**

Table 1: Comparisons of overall performances on the two datasets.

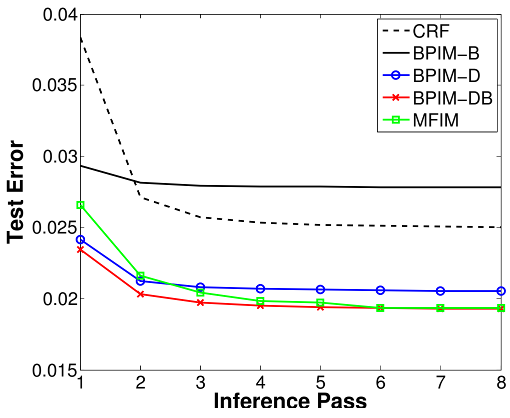

Figure 4: Average test error as a function of pass for each message-passing method on the 3D classification task.

the base predictor with our inference machine approaches could potentially lead to improved performance. Both inference machines (MFIM, BPIM-D) outperform the baseline CRF message-passing approach. Additionally, we observe that using backpropagation on the output of DAgger slightly improved performance. Without this initialization, backpropagation does not find a good solution. In this particular dataset we do not notice any advantage of the cavity method (BPIM-D) over the mean-field approach (MFIM). In Fig. 4, we observe that the error of all asynchronous message-passing approaches converge roughly after 3-4 inference passes, while the synchronous message-passing (MFIM) converges slightly slower and requires around 6 passes.

We also performed the experiment on the smaller split used in [14] (i.e., training on a single scene) and our approach (BPIM-D) obtained slightly better accuracy of 97.27% than the best approach in [14] (M^3N-F: 97.2%). However, in this case, backpropagation did not further improve the solution on the test scenes as it overfits more to a single training scene.

3D Surface Layout Estimation. In this experiment our BPIM-D approach performs slightly better than all other approaches in terms of accuracy, including the performance of the previous state-of-the-art in [7]. In terms of F1 score, [7] is slightly better. Given that [7] used a more powerful base predictor (boosted trees) than our logistic regressor, we believe we could also achieve better performance using more complex predictors. We notice here a larger difference between the BPIM and MFIM approaches, which confirms the cavity method can lead to better performance. Here the M^3N-F approach did not fare very well and all message-passing approaches outperformed it. All inference machine approaches also outperformed the baseline CRF. Fig. 6 shows a visual comparison of the M^3N-F, MFIM, BPIM-D and [7] approaches on two test images. The outputs of BPIM-D and [7] are very similar, but we can observe more significant improvements over the M^3N-F.

5. Conclusion and Future Work

We presented a novel approach to structured prediction which is simple to implement, has strong performance guarantees, and performs as well as state-of-the-art methods across multiple domains. The efforts presented here by no means represent the end of a line of research; we believe there is substantial remaining research to be done in learning inference machines. In particular, while we present simple effective approaches (Forward training, DAgger) for leveraging local information to learn a deep modular inference machine, we believe alternate techniques may be very effective at addressing this optimization. Further, while the message passing approaches we investigate here often provide outstanding performance, other methods including sampling and graph-cut based approaches are often considered state-of-the-art for other tasks. We believe the similar ideas of unrolling such procedures and using notions of global and local training may prove equally effective and are worthy of investigation. Additionally, a significant (unexplored) benefit of the approach taken here is that we can easily include features and computations not typically considered as part of the graphical model approach and still attempt to optimize overall performance; e.g., computing new features or changing the structure based on the results of partial inference.

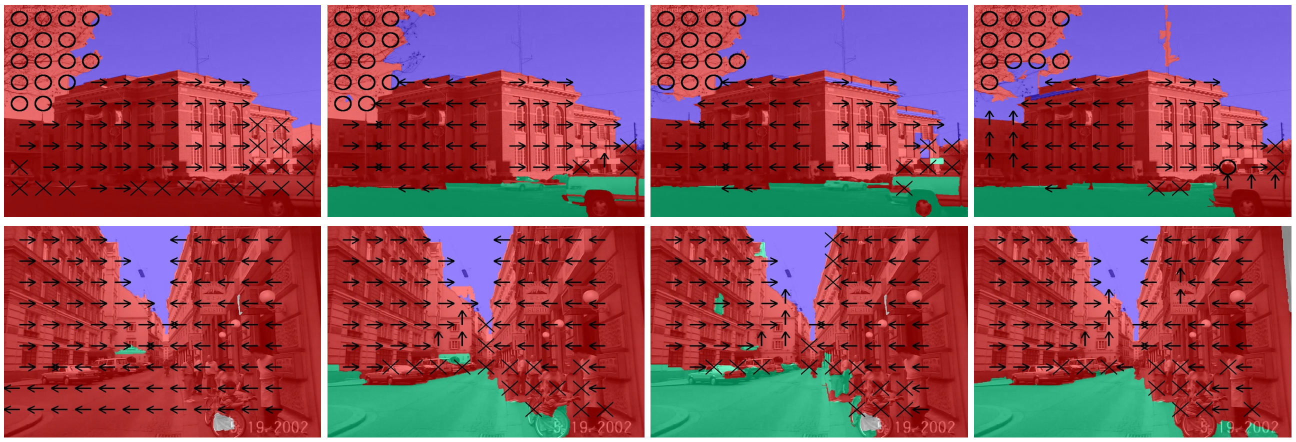

Figure 5: Estimated 3D geometric surface layouts. From left to right: M^{3}N-F, Hoeim et al. [7], BPIM-D, Ground truth.

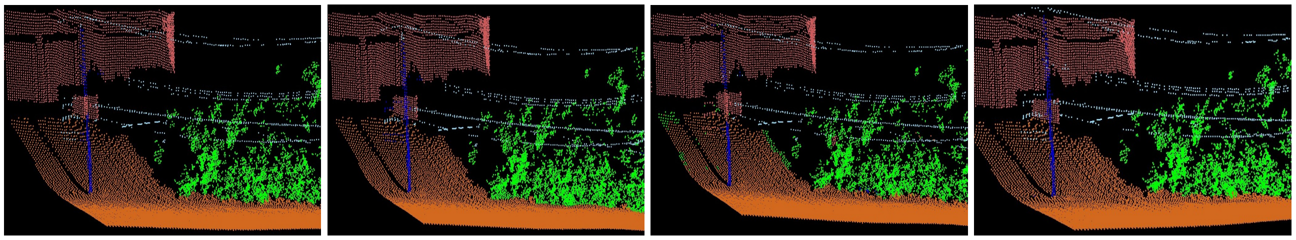

Figure 6: Estimated 3D point cloud labels. From left to right: M^{3}N-F, M^{3}N-P, MFIM, Ground truth.

Acknowledgements

This work is supported by the ONR MURI grant N00014-09-1-1052, Reasoning in Reduced Information Spaces, by the National Sciences and Engineering Research Council of Canada (NSERC), and by a QinetiQ North America Robotics Fellowship.

References

[1] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *T-PAMI*, 23(1), 1999.

[3] L. Csato, M. Opper, and O. Winther. Tap gibbs free energy, belief propagation and sparsity. In *NIPS*, 2001.

[4] H. Daumé III, J. Langford, and D. Marcu. Search-based structured prediction. *MLJ*, 75(3), 2009.

[5] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *UAI*, 2006.

[6] T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *ICML*, 2008.

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

[8] P. Kohli, L. Ladicky, and P. H. Torr. Robust higher order potentials for enforcing label consistency. *IJCV*, 82(3), 2009.

[9] A. Kulesza and F. Pereira. Structured learning with approximate inference. In *NIPS*, 2008.

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

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

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *IEEE*, 86(11), 1998.

[13] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *ECCV*, 2010.

[14] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert. Contextual classification with functional max-margin markov networks. In *CVPR*, 2009.

[15] M. Opper and D. Saad. *Advanced Mean Field methods – Theory and Practice*. MIT Press, 2000.

[16] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[17] S. Ross and J. A. Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010.

[18] S. Ross, G. J. Gordon, and J. A. Bagnell. No-Regret Reductions for Imitation Learning and Structured Prediction. In *AISTATS*, 2011.

[19] D. Roth, K. Small, and I. Titove. Sequential learning of classifiers for structured prediction problems. In *AISTATS*, 2009.

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

[21] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *T-PAMI*, 32(5), 2009.

[22] M. J. Wainwright. Estimating the “wrong” graphical model: Benefits in the computation-limited setting. *JMLR*, 7(11), 2006.

[23] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. Tree-based reparameterization for approximate estimation on loopy graphs. In *NIPS*, 2001.