Auto-context and Its Application to High-level Vision Tasks

Zhuowen Tu
Lab of Neuro Imaging, University of California, Los Angeles
ztu@loni.ucla.edu

Abstract

The notion of using context information for solving high-level vision problems has been increasingly realized in the field. However, how to learn an effective and efficient context model, together with the image appearance, remains mostly unknown. The current literature using Markov Random Fields (MRFs) and Conditional Random Fields (CRFs) often involves specific algorithm design, in which the modeling and computing stages are studied in isolation. In this paper, we propose an auto-context algorithm. Given a set of training images and their corresponding label maps, we first learn a classifier on local image patches. The discriminative probability (or classification confidence) maps by the learned classifier are then used as context information, in addition to the original image patches, to train a new classifier. The algorithm then iterates to approach the ground truth. Auto-context learns an integrated low-level and context model, and is very general and easy to implement. Under nearly the identical parameter setting in the training, we apply the algorithm on three challenging vision applications: object segmentation, human body configuration, and scene region labeling. It typically takes about 30 ∼ 70 seconds to run the algorithm in testing. Moreover, the scope of the proposed algorithm goes beyond high-level vision. It has the potential to be used for a wide variety of problems of multi-variate labeling.

1. Introduction

It has been noted that context and high-level information plays a vital role in object and scene understanding [2, 22]. Yet, a principled way of learning an effective and efficient context model, together with the image appearance, is not available. Context comes into a variety of forms and it can be referred to as Gestalt laws in middle level knowledge, intra-object configuration, and inter-object relationship. For example, a clear horse face may suggest the location of its tail and legs, which are often occluded or not easy to identify. The strong appearance of a car might well suggest the existence of a road, and vice versa [8].

From the Bayesian statistics point view, context is repre-
rectly in a supervised approach. Like in the BP algorithm [30], the goal is to learn/compute the marginals of the posterior, which we also call classification maps for the rest of this paper. Each training image comes with a label map in which every pixel is assigned with a label of interest. A classifier is first trained to classify each pixel. There are two types of features for the classifier to choose from: (1) image features computed on the local image patches centered at the current pixel (we use image patches of fixed size 11 × 11 in this paper), and (2) context information on the classification maps. Initially, the classification maps are uniform, and thus, the context features are not selected by the first classifier. The trained classifier then produces new classification maps which are used to train another classifier. The algorithm iterates to approach the ground truth until convergence. In testing, the algorithm follows the same procedure by applying the sequence of learned classifiers to compute the posterior marginals.

The auto-context algorithm integrates the image appearances (observed data) together with the context information by learning a series of classifiers. Unlike many the energy minimization algorithms where the modeling and computing stages are separated, auto-context uses the same procedures in the two (this is a property of many classifiers since they get close-form solutions). The difference between the results in training and testing is the generalization error of the trained classifiers. Therefore, there is no explicit energy to minimize in auto-context. This alleviates the burden in searching for the optimal solution. Auto-context uses deterministic procedures, but it carries the uncertainties without the need of making any hard decisions. This makes the auto-context algorithm significantly faster than most the existing energy minimization algorithms. Compared to MRFs, CRFs, auto-context no longer works on a fixed neighborhood structure. Each pixel (sample) can have support from a large number of neighbors, either short or long range. It is up to the learning algorithm to select and fuse them. The classifiers in different stages may choose different supporting neighbors to either enhance or suppress the current probability towards the ground truth. Also, the appearance (likelihood) and prior (context and shape) are directly combined in an implicit way and the balance between the two is naturally handled.

Two pieces of work directly related to auto-context are: Boosted Random Fields (BRFs) [22] and SpatialBoost [1], which both use boosting algorithm to combine the contextual information. However, both the algorithms use contextual beliefs as weak learner in the boosting algorithm, which is time-consuming to update. Possibly due to this reason, SpatialBoost was only illustrated on an interactive segmentation task. BRFs learns the message update rules in the belief propagation algorithm, and the main focus of SpatialBoost is to propose an extended algorithm for the boosting algorithm. The auto-context is a general algorithm and the classifier of choice is not limited to boosting algorithms. It directly targets the posterior through iterative steps, resulting in a simpler and more efficient algorithm than BRFs and SpatialBoost. Under nearly the same set of parameters in training, we demonstrate several applications using the auto-context, which are not available in [22, 1].

A feed forward way of including context with appearance was proposed in [28] for object detection. However, their method is not to iteratively learn a posterior. More importantly, their findings lead to the conclusion that the help from context is negligible (unless the image quality is really poor). Our experimental results in Fig. (3.a) suggest it differently. One possible reason might be that the image segmentation/parsing task is different from the object detection problem focused in [28]. Others [16] also showed that explicit context information improves region segmentation/ labeling results greatly.

Compared to the traditional Bayesian approach for image understanding [25], auto-context is much easier to train and it avoids heavy algorithm design. It is significantly faster than many the existing algorithms in this domain. Compared to the algorithms using context [16, 8, 26], it learns an integrated model. There is no hard decision made in the intermediate stages and uncertainties are all carried through posterior marginals.

We demonstrate the auto-context algorithm on challenging high-level vision tasks for three well known datasets: horse segmentation in the Wizemann dataset [3], human body configuration in the Berkeley dataset [13], and scene region labeling in the MSRC dataset [19]. The proposed algorithm is general and very easy to implement. Its scope goes beyond high-level vision tasks. Indeed, it has the potential to be used for many problems for multi-variate labeling where the joint statistics needs to be modeled.

2. Problem formulation

In this section, we give the problem formulation for the auto-context model and briefly discuss some related algorithms.

2.1. Objective

Let the data vector be \( X = (x_1, ..., x_n) \). In the case of 2D image, \( X = (x_{i,j}), (i, j) \in \Lambda \) where \( \Lambda \) denotes the image lattice. For notational clarity, we do not distinguish the two and call the both ‘image’. In training, each image \( X \) comes with a ground truth \( Y = (y_1, ..., y_n) \) where \( y_i \in \{1, K\} \) is the label of interest for each pixel \( i \). The training set is then \( S = \{(Y_j, X_j), j = 1 \ldots m \} \) where \( m \) denotes the number of training images. The Bayes rule says \( p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)} \), where \( p(X|Y) \) and \( p(Y) \) are the likelihood and prior respectively. Often, we look for the optimal solution
maximizing a posterior (MAP)
\[ Y^* = \arg \max p(Y|X) = \arg \max p(X|Y)p(Y). \]

As mentioned before, the main difficulties for the MAP framework come from two aspects. (1) **modeling**: it is very hard to learn accurate \( p(X|Y) \) and \( p(Y) \) for real-world applications. Both of them have high complexity and usually do not follow independent identical distribution (i.i.d.). (2) **computing**: The combination of the \( p(X|Y) \) and \( p(Y) \) is often non-regular. Besides many recent advances made in optimization and energy minimization [21], a general and immediate solution yet remains out of reach.

Instead of decomposing \( p(Y|X) \) into \( p(X|Y) \) and \( p(Y) \), we study the posterior directly. Moreover, we look at the optimization and energy minimization \[21\], a general and often non-regular. Besides many recent advances made in optimization and energy minimization [21], a general and immediate solution yet remains out of reach.

To better approximate the marginals in eqn. (1) by including a large number of context information, we propose an auto-context model. As mentioned above, a traditional classifier can learn a classification model based on local image patches, which now we call
\[ \mathcal{P}^{(0)} = (p_1^{(0)}, \ldots, p_n^{(0)}) \]
where \( p_i^{(0)} \) is the posterior marginal for each pixel \( i \) learned by eqn. (2). We construct a new training set
\[ S_1 = \{(y_{ji}, X_j(N_i)), j = 1..m, i = 1..n\} \]
where \( \mathcal{P}_j^{(0)}(i) \) is the classification map for the \( j \)th training image centered at pixel \( i \). We train a new classifier, not only on the features from the image patch \( X_j(N_i) \), but also on the probabilities, \( \mathcal{P}_j^{(0)}(i) \), of a large number context pixels. These pixels can be either near or very far from \( i \), and Fig. (1) shows an illustration. It is up to the learning algorithm to select and fuse important supporting context pixels, together with features about image appearance. Once a new classifier is learned, the algorithm repeats the same procedure until it converges. The algorithm iteratively updates the marginal distribution to approach
\[ p^{(n)}(y_i|X(N_i), \mathcal{P}^{(n-1)}) \rightarrow p(y_i|X) = \int p(y_i, y_{-i}|X)dy_{-i}. \]

In theorem 1 we show that the algorithm is asymptotically approaching \( p(y_i|X) \) without doing explicit integration. A more direct link between the two, however, is left for future research.

In fact, even the first classifier is trained the same way as the others by giving it a probability map of uniform distribution. Since the uniform distribution is not informative at all, the context features are not selected by the first classifier. In some particular applications, e.g. medical image
segmentation, the positions of the anatomical structures are roughly known. One then can use a probability atlas as the initial $P^{(0)}$.

Given a set of training images together with their label maps, $S = \{(Y_j, X_j), j = 1..m\}$: For each image $X_j$, construct probability maps $P_j^{(0)}$ with uniform distribution on all the labels. For $t = 1, ..., T$:

- Make a training set $S_t = \{(y_{ji}, (X_j, N_i), P_j^{(t-1)}(i)), j = 1..m, i = 1..n\}$.
- Train a classifier on both image and context features extracted from $X_j(N_i)$ and $P_j^{(t-1)}(i)$) respectively.
- Use the trained classifier to compute new classification maps $P_j^{(t)}(i)$ for each training image $X_j$.

The algorithm outputs a sequence of trained classifiers for $P^{(n)}(y_i|X(N_i), P^{(n-1)}(i))$.

Figure 2. The training procedures of the auto-context algorithm.

**Theorem 1** The auto-context algorithm monotonically decreases the training error.

Proof: For notational simplicity, we consider only one image in the training data and use $X(i)$ to denote $X(N(i))$. In the AdaBoost algorithm [5], the error function is taken by $\epsilon = \sum_i e^{-y_i H(X(i))}$ for $y_i \in \{-1, +1\}$, which can be given an explanation as the log-likelihood model [4]. At different steps, 

$$\epsilon_t = - \sum_i \log p^{(t)}(y_i|X(i), P^{(t-1)}(i)), \text{ and}$$

$$\epsilon_{t-1} = - \sum_i \log p_i^{(t-1)}(y_i),$$

where

$$p^{(t)}(y_i|X(i), P^{(t-1)}(i)) = \frac{e^{F_k^{(t)}(X(i), P^{(t-1)}(i))}}{\sum_k e^{F_k^{(t)}(X(i), P^{(t-1)}(i))}}.$$  

\[ (5) \]

$F_k^{(t)}(X(i), P^{(t-1)}(i))$ includes a set of weak classifiers selected for label class $k$. It is straightforward to see that we can at least make

$$p^{(t)}(y_i|X(i), P^{(t-1)}(i)) = p_i^{(t-1)}(y_i)$$

since the equality can be easily achieved by making

$$F_k^{(t)}(X(i), P^{(t-1)}(i)) = \log p_i^{(t-1)}(k).$$

The boosting algorithm (or almost any valid classifier) choose a set of $F_k^{(t)}$ in minimizing the total error $\epsilon_t$, which should at least do better than $p_i^{(t-1)}(y_i)$ Therefore,

$$\epsilon_t \leq \epsilon_{t-1}.$$  

The convergence rate depends on the amount of error reduced $\epsilon_{t-1} - \epsilon_t$. Intuitively, the next round of classifier tries to select features both from the appearances and the previous classification maps. A trivial solution is to use the previous probability map for the classifier. This also shows that the optimal classifier is at a stable point. Of course, this requires to have the feature of its own probability in the candidate pool, which is not hard to achieve. Fig. (1) gives an illustration of the procedures of the auto-context. Several rays are shot from the current pixel and we sparsely sample the context locations (both individual pixels and windows) to use their classification probabilities as features. Each round of training will select different sets of context pixels, either short range or long-range.

### 2.4. Understanding auto-context

We first take a look at the Belief Propagation algorithm [15, 30] since it also works on the marginal distribution. For directed graph, BP is guaranteed to find the global optimal. For loopy graph, BP computes an approximation. For a model on a graph

$$p(Y) = \frac{1}{Z} \prod_{(i,j)} \psi(y_i, y_j) \prod_i \phi_i(y_i)$$

where $Z$ is the normalization constant, $\psi(y_i, y_j)$ is the pairwise relation between sites $i$ and $j$, and $\phi_i(x_i)$ is a unary term. The BP algorithm [30] computes the belief (marginal) $p_i(y_i)$ by

$$p_i(y_i) = \frac{1}{Z} \phi_i(y_i) \prod_{j \in N(i)} m_{ji}(y_i),$$  

\[ (6) \]

where $m_{ji}(x_i)$ are the messages from $j$ to $i$,

$$m_{ij}(y_j) \leftarrow \sum_{y_i} \phi_i(x_i) \phi_{i,j}(y_i, y_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_k).$$  

\[ (7) \]

Similarly, the auto-context algorithm updates the marginal distribution by eqn. (5). The major difference between BP and auto-context are: (1) On the graphical model, every pair of $\psi_{i,j}(y_i, y_j)$ on all possible labels need to be evaluated and integrated in eqn. (7). Therefore, BP can only work with a limited number of neighborhoods to keep the computational burden under check. For auto-context, it evaluates a sequence of learned classifiers, $F_k^{(t)}(X(i), P^{(t-1)}(i))$, which is computed based on a set of selected features. Therefore, auto-context can afford to look at a much longer range of support and it is up to the learning algorithm to select and fuse the most supportive contexts and appearance information. Also, there is no integration between the pair $y_i$ and $y_j$. (2) BP works on a fixed graph structure and the update rule is the same. auto-context learns different classifiers on different set of features at different stages, which allows it to make use of the best available set of information each time. (3) In BP, there is often separate stages to
design the graphical model, and learn \( \psi(y_i, y_j) \) and \( \phi_i(y_i) \). Auto-context is targeted to learn the posterior marginal directly and its inference stage follows the identical steps in the learning phase. The difference between the learning and test stages is the generalization error of the classifiers which can be studied by the VC dimension or margin theory [27]. However, BP has the advantage that it uses the same message passing rule for different forms of \( p_1(y_i) \) in eqn. (6), whereas auto-context requires to learn a different set of classifiers for different tasks.

A question one might ask is: “How different is it between learning a recursive model \( p^{(t)}(y_i|X, \psi^{(t-1)}(i)) \) and learning \( p(y_i|X) \) directly?” A classifier can be learned by using the entire image \( X \) rather than the image patch \( X(i) \). A major issue is that \( p(y_i|X) \) should be a marginal distribution by integrating out the other \( i \) as shown in eqn. (1). The correlation between different pixels needs to be taken into account, which is hard by learning one classifier for \( p(y_i|X) \). It also builds a feature space too big for a classifier to handle and might lead to severe overfitting. Wolf and Bileschi [28] suggested that using label context might achieve the same effect as using image appearance context, in object detection. Moreover, in both the situations where contexts were used, the improvements were small. We conducted an experiment to train a system with image appearance, instead of the probabilities, for the pixels sparsely sampled on the rays, as suggested in [28]. The results are shown in Fig. (3.a) and the conclusions are different from [28] in two aspects: (1) having appearance context even gives worse result than using features from patches only (due to overfitting); (2) label (in probabilities) contexts greatly improve the segmentation/labeling result.

There have been many algorithms along the line of using context [2, 16, 8, 22, 1]. Auto-context makes an attempt to recursively select and fuse context information, as well as appearance, in a unified framework. The first round of classifier is based purely on the local appearance. Objects with strong appearance cues often achieve high probabilities on their labels. These probabilities then start to make influence to the others, if there are strong correlations between them. Themselves are also getting support from the others. During the iterations, the algorithm learns to suppress the strong probabilities, if wrong, and improve the low ones to the target distribution. Context information not only comes from between-objects, they are also from within-objects (parts), even the parts may be far from each other spatially. Auto-context uses a very general and simple scheme by avoiding heavy manual algorithm design.

3. Experiments

We illustrate the auto-context algorithm on three challenging high-level vision tasks: horse segmentation, human body configuration, and scene parsing/labeling. In these three tasks, the system uses nearly an identical parameter setting, mostly generic ones such as the number of weak classifiers and the stopping criterion. The system can be used for a variety of tasks with given training images and label maps.

3.1. Horse segmentation

We use the Weizmann dataset consisting of 328 horse images [3] of gray scale. The dataset also contains manually annotated label maps. We split the dataset randomly into half for training and half for testing. The training stage follows the steps described in Fig. (2). Two main implementation issues we have not talked about are the choices of: (1) classifiers, (2) features. Boosting algorithms appear to be a natural choice for the classifier as they are able to select and fuse a set of features from a large candidate pool to approximate the target posterior marginal. However, our algorithm is not tied to any specific classifier and one can choose others such as SVM [27]. Since each pixel is a training sample, there consists of millions of positive and negative ones. It is hard to build a single node boosting algorithm to perform the classification. We adopt the probabilistic boosting tree (PBT) algorithm [23] as it learns and computes a discriminative model in a hierarchical way by

\[
p(y|X) = \sum_{l_1} p(y|l_1, X)p(l_1|X) = \sum_{l_1, \ldots, l_n} p(y|l_n, \ldots, l_1, X), \ldots, p(l_2|l_1, x)p(l_1|X),
\]

where \( p(l_i) \) is the classification model learned by boosting node in the tree. The details can be found in [23]. In our implementation of the PBT, we further improve it by using a criterion in [14] for the choice of tree branches. We choose not to discuss the details as it is not the major focus of this paper. A nice thing about PBT is that two-class and multi-class classification problems are treated in an identical way with two-class being a special case. Therefore, our algorithm handles two-class and multi-class labeling problems naturally without any change.

Image features are computed on the patch (size of 21 × 21) centered on each pixel \( i \) and we use around 8,000 including Canny edge results at a low scales (1.5), Haar filter responses, and gradients. The context features are the probability values of different pixels relative to the current pixel. For each current pixel, we shoot out many rays and sparsely choose the pixels on these rays. Fig. (1) shows an example. The features can be probability directly on these pixels or the mean probability around them, or even other high order statistics. In total, we use around 4,000 of the context features. The training algorithm starts from probability maps of uniform distribution, and then it recursively refines the maps until it converges. The first classifier does
not choose any context features as they are not informative at all. Starting from the second classifier, nearly 90% of the features selected are context features with the rest being the image features. This demonstrates the importance of using the context information in clarifying the ambiguities.
This is now a multi-class labeling problem rather than foreground-background segregation. As stated before, PBT based auto-context algorithm treats two-class and multi-class problem in an identical way. We collect around 130 images for training, and use the same set of features as in the horse segmentation problem on image patch of size $21 \times 21$ (designing some specific features for this task might further improve our result).

Fig. (6) shows the results at different stages of the auto-context on the test images in [19]. Fig. (5.b) gives the confusion matrix. As we can see, the main body, the head, the left thigh, the right thigh and the feet can be labeled robustly in most cases. The arms appear to be confused with the main body and the background. The speed on these test images are about the same as in the horse segmentation case. The existing algorithms in the problem often have a pre-segmentation stage [13, 24], which is prone to errors by the low-level segmentation algorithms. For example, the procedures described by [13, 24] can merge segmented regions into big ones but can not break them. This may cause problem where there is no clear boundary between the parts. The auto-context algorithm computes the posterior marginals for each pixel directly.

We illustrate our algorithm on gray scale images, and color images used in [13]. The results are better than those shown in [24] in which a BP algorithm was implemented. BP computes the marginals by propagating messages through local connected neighbors, whereas auto-context can take long-range context information directly. This results in significant speed improvement. Similar to the argument made in the horse segmentation case, our algorithm is more general than [13, 24]. However, a thorough comparison of the auto-context algorithm with the state of art algorithms (BP, Graph-Cuts, CRFs), in terms of both quality and speed, remains as future research.

Further procedures are still required to explicitly extract the body parts since the auto-context algorithm only outputs probability maps. This is probably the place where more explicit shape information can be used in the Bayesian framework.

### 3.3. Scene parsing/labeling

We also applied our algorithm on the task of scene parsing/region labeling. We used the MSRC dataset [19] in which there are 591 images with 21 types of objects manually segmented and labeled (there are two additional types in the new dataset). There is a nuisance category labeled as 0. The setting for this task is similar as before, and the only difference is that we use color images in this case. Shotton et al. did not have the background model to learn the regions of 0 label, whereas it is not a problem in our case. However, to obtain a direct comparison to their result, we also exclude the 0 label both in training and testing. We use the identical training and testing images as in [19]. Fig. (7) shows some results and the confusion matrix. The results by auto-context are the marginal probabilities for each pixel belonging to a specific class. We simply assign the label with the highest probability to each pixel. The accuracy by the first stage of auto-context, classification method PBT only, achieves 50.4%. The overall pixel-wise accuracy by 4 layers of auto-context is 74.5% which is better than 72.2% reported in [19]. However, a careful reading at the confusion matrices by both the algorithms shows that our result is more consistent and the mistakes made are more “reasonable”. For example, boat is mostly confused with car and building whereas boat was mis-classified to many other classes in [19] such as water, bike, and tree.

Our algorithm is more general and easier to implement. The speed reported in [19] was 3 minutes per image whereas ours is around 70 seconds. Notice that the classification results are a bit scattered. Another level of algorithm to enforce region-based consistency is still needed. With a post-processing stage to encourage the neighboring pixels to have the same label, the accuracy improves to 77.7%.

$$Y^* = \arg \min_i -\sum_i \log p(y_i|X) + \alpha \sum_{(i,j)} \delta(y_i \neq y_j),$$

where $\alpha = 2.0$ for the results in this paper.

It is noted that almost all the algorithms we compare to, on the horse segmentation, human body configuration, and scene labeling use the context or high-level information. CRF models are indeed context based. A direct comparison to the algorithms reported on the MSRC dataset is given in table (1). [16] gave the accuracy measure on segmented regions rather than pixels with a score 68.4%. Using auto-context with a post-processing achieves 77.7%.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>TextureForest [19]</th>
<th>[29]</th>
<th>Auto-Context</th>
<th>AC+ post</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>72.2%</td>
<td>75.1%</td>
<td>74.5%</td>
<td>77.7%</td>
</tr>
</tbody>
</table>

Table 1. Comparison to other algorithms on the MSRC dataset. AC+post refers to the result by auto-context with a post-processing for smoothing and the post processing takes about 0.1 second.

### 3.4. Conclusions

In this paper, we have introduced an auto-context algorithm to learn a unified low-level and context model. We
target the posterior distribution directly, and thus, the test phase shares the same procedures as those in the training. The auto-context algorithm selects and fuses a large number of supporting contexts which allow it to perform rapid message propagation. We introduce iterative procedures into the traditional classification algorithms to refine the classification results by taking effective context information.

The proposed algorithm is very general. Under nearly a same set of parameters, we illustrate the auto-context algorithm on three challenging vision tasks. The results show to significantly improve the results by patch-based classification algorithms and demonstrate improved results over almost all the existing algorithms using CRFs and BP. It typically takes about 30 ~ 70 seconds to run the algorithm of size around $300 \times 200$. However, a full scale of comparison with various choices of the algorithm, like that conducted in [21], is needed. The scope of the auto-context model goes beyond vision applications and can be applied in other problems of multi-variate labeling in machine learning and AI.

The limitations for the auto-context model are: (1) the features on the context information are still somewhat limited and more explicit shape information is still required; (2) different auto-context models need to be trained for different applications; (3) the training time takes a bit long (a few days) for the scene parsing.

Acknowledgments ZT is funded by NIH Grant U54 RR021813 entitled Center for Computational Biology. We thank Yingnian Wu for many stimulating discussions.

References