Report on Structured Forests for Fast Edge Detection

Ans Munir
(MSDS20033)
ITU Lahore

1. Introduction

Edge detection is a key pre-processing step in several important computer vision tasks such as object recognition, active contours and segmentation. Old approaches to detect edges compute color gradient magnitude and then perform non-maximum suppression. However a lot of visually important edges do not correspond to the above color gradient method including texture edges.

The edges of local patch many times have well-known patterns like straight line or T-junctions. For this characteristics recently structured learning approaches has been applied. A good approach for edge detection is generalized structure learning. This approach uses the inherit structure present in the edge patches and it is also very computationally efficient. In order to capture the structured information, Random Forest framework has been used. The problem of edge detection has been formulated as given the input image patches, the goal is to predict the local segmentation masks. A novel approach to learn decision trees has been used that takes the advantage of structured labels to decide the splitting at each branch of the tree. Every forest outputs a patch of the edge pixel labels that are aggregated from all the image in order to compute final edge map.

2. Random Decision Forests

A decision tree $f_t(x)$ classifies a sample $x \in X$ by iteratively branching left or right down the tree until it reach to leaf node. Specially each node j in the tree is associated with the binary $split function$:

$$h(x, \theta_j) \in 0, 1$$

with parameters θ_j. If $h(x, \theta_j) = 0$ node j sends x left otherwise right and this process will terminate at the leaf node. If there is an input x then the output of the tree would be predicting score at the leaf reached by x which can be a target label $y \in Y$ or distribution over labels Y.

A decision forest is an ensemble of T independent trees f_t. If a sample x is given then prediction $f_t(x)$ from the set of trees are combined with the help of $ensemblemodel$ in the form of single output. Common choices for ensembles models are majority voting for classification and averaging for regression. However we can also use more complex ensemble models.

3. Training Decision Trees

Every tree is trained recursively and independently. For a given node j and training set $S_j \subset X * Y$, the goal is to find parameters $_j$ of the split function $h(x, \theta_j)$ that result in a 'good' split of the data. This requires defining an information gain criterion of the form:

$$I_j = I(S_j, S_j^L, S_j^R)$$

where $S_j^L = \{(x, y) \in S_j | h(x, \theta_j) = 0\}, S_j^R = S_j / S_j^L$. Splitting parameters $_j$ are chosen to maximize the information gain I_j ; training then proceeds recursively on the left node with data S_j^L and similarly for the right node.

4. Randomness and Optimality

Individual decision trees shows high variance and more prone to over-fit. Decision forests make this better by training multiple de-correlated trees and combining their output. Therefore during training the main focus is to achieve the enough diversity in the trees.

Diversity of trees can be achieved either by randomly sub-sampling the data that is used to train each $tree$ randomly sub-sampling the features and splits used to train each $node$.

5. Structured Random Forests

Training random forests with structured labels creates two main challenges. First, structured output spaces are often high dimensional and complex. Thus scoring numerous candidate splits directly over structured labels may be prohibitively expensive. Second, and more important, information gain over structured labels may not be well defined.

Given the discrete labels C, information gain calculated directly and efficiently over C can serve as a proxy for the information gain over the structured labels Y. By mapping the structured labels to discrete labels prior to training each node we can leverage existing random forest training procedures to learn structured random forests effectively.

6. Ensemble Model

There is a need to define how to combine a set of n labels $y_1 ... y_n \in Y$ into a single prediction both for training (to associate labels with nodes) and testing (to merge multiple predictions). Like before, m dimensional mapping Π_θ has been sampled and computed $z_i = \Pi_\theta(y_i)$ for each i. We select the label y_k whose z_k is the medoid, i.e. the z_k that minimizes the sum of distances to all other z_i.

The ensemble model depends on m and the selected mapping Π_θ. However, there is only need to compute the medoid for small n (either for training a leaf node or merging the output of multiple trees), so having a coarse distance metric suffices to select a representative element y_k.

7. Edge Detection

It has been assumed that a set of segmented training images are given in which the boundaries between the segments correspond to contours Given an image patch, its annotation can be specified either as a segmentation mask indicating segment membership for each pixel (defined up to a permutation) or a binary edge map. Former has been denoted by $y \in Y = Z^{d \times d}$ and later has been denoted by $y' \in Y' = \{0, 1\}^{d \times d}$, where d indicates patch width. An edge map y' can always be trivially derived from segmentation mask y, but not vice versa.