Learning Structural Element Patch Models With Hierarchical Palettes: Supplementary material

Jeroen Chua
University of Toronto
Canada
jeroen@psi.utoronto.ca

Inmar Givoni
University of Toronto
Canada
inmar@psi.utoronto.ca

Ryan Adams
Harvard University
United States
rpa@seas.harvard.edu

Brendan Frey
University of Toronto
Canada
frey@psi.utoronto.ca

1. Variational inference and learning

We derive the variational inference updates in this section. We use these updates as an E-step in a variational EM framework that is guaranteed to increase a lower bound on the data likelihood. The variational inference updates for our Q-distributions are given below.

$$Q^*(s_{mj}) \propto \sum_{u_{mg}} \Big(Q(u_{mg}) \prod_n \Big(\sum_{r_{mjn}} \Big(P(r_{mjn} \mid s_{mj}) \sum_{g_{mjn}} \Big(P(g_{mjn} \mid r_{mjn}) \times P(\vec{x}_{mjn} \mid g_{mjn}, u_{mg}) \Big) \Big) \Big) \Big). \quad (1)$$

The proportionality constant is given by $\sum_{s_{mj}} Q^*(s_{mj})$. To compute $Q^*(s_{mj})$, we evaluate each pixel under all $U \times G$ pairings of the U universal colors and G global colors, which is tractable. We use a similar derivation to obtain

$$Q^*(r_{mjn} \mid s_{mj}) \propto P(r_{mjn} \mid s_{mj}) \sum_{g_{mjn}} \Big(P(g_{mjn} \mid r_{mjn}) \times \sum_{u_{mg}} \Big(P(u_{mg}) P(x_{mjn} \mid g_{mjn}, u_{mg}) \Big) \Big), \quad (2)$$

and

$$Q^*(g_{mjn} \mid r_{mjn}) \propto P(g_{mjn} \mid r_{mjn}) \times \sum_{u_{mg}} \Big(P(u_{mt}) P(x_{mjn} \mid g_{mjn}, u_{mg}) \Big). \quad (3)$$

We let $Q(\mathbf{u}) = \prod_{mg} Q(u_{mg})$ noting that this is a further approximation since, due to explaining away, $Q(u_{mg}) \forall g \in \{1...G\}$ is not independent conditioned on observing the data. We obtain

$$Q^*(u_{mg}) \propto P(u_{mg}) \prod_j \Big(\sum_{s_{mj}} \Big(P(s_{mj}) \prod_n \Big(\sum_{r_{mjn}} \Big(P(r_{mjn} \mid s_{mj}) \times P(g \mid r_{mjn}) P(x_{mjn} \mid g, u_{mg}) \Big) \Big) \Big) \Big). \quad (4)$$

With the Q-distribution updates given as above, we can write down the M-Step for our variational EM framework. Due to space restrictions, we denote the variable pair mj by ν. The palette updates, namely the updates for $\vec{\mu}_u$ and $(\sigma^2)^h_{mg}$ $\forall u, m, g, h$ are given in the paper. The remaining M-Step updates are:

$$\pi_{ns}(r) = \frac{\sum_\nu Q(s_\nu = s) Q(r_{\nu n} = r \mid s_\nu = s)}{\sum_\nu Q(s_\nu = s)} \quad (5)$$

$$\alpha_{\nu g}(r) = \frac{\sum_k Q(s_\nu = s) \sum_n Q(r_{\nu n} = r \mid s_\nu = s) Q(g_{\nu n} = g \mid r_{\nu n} = r)}{\sum_s Q(s_\nu = s) \sum_n Q(r_{\nu n} = r \mid s_\nu = s)} \quad (6)$$

$$\beta_u = \frac{\sum_{mg} Q(u_{mg} = u)}{\sum_{mgu} Q(u_{mg} = u)} \quad (7)$$

The updates can be thought of as updating preferences for different hidden variable settings. For example, Eq. 5 operates by summing over how much each pixel prefers each region-shapelet combination in each image and each patch and normalizing, and Eq. 7 operates by summing over how often each universal color is used in each image by each image color and normalizing.

2. Learning the shapelet library

The variational EM algorithm presented in the paper provides a way to learn a shapelet library. However, we

have found that a slight variation on this algorithm generates, on average, shapelet libraries with a higher data likelihood on the training set. We have found that learning shapelets in increasing order of the number of regions they contain results in a shapelet library with a higher likelihood than learning all shapelets simultaneously. Specifically, to learn the shapelet library, we run the variational EM algorithm described earlier $\hat{R}-1$ times, successively learning shapelets which contain more regions. During each EM run, numbered $k=[1...\hat{R}-1]$, we re-infer the parameters $\alpha, \beta, \mu, (\sigma^2)$ from a random initialization. For the parameters π, we use a random initialization for the first EM run, and use the π parameters inferred from the previous EM run for all subsequent runs. We modify the M-Step update in Eq. 5 to be:

$$\pi_{ns}(r) = \frac{\sum_\nu Q(s_\nu = s) Q(r_{\nu n} = r \mid s_\nu = s)}{\sum_\nu Q(s_\nu = s)} \times \delta_{R_s-(k+1)} + \pi_{ns}^{old}(r) \times (1 - \delta_{R_s-(k+1)}) \quad (8)$$

where δ is the Kronecker delta function, and $\pi_{ns}^{old}(r)$ is the $\pi_{ns}(r)$ parameter from the previous EM iteration. That is, for the k-th variational EM run, we update the elements of π that govern shapelets with $k+1$ regions, keeping all other elements of π fixed. Note that there is only one shapelet with one region, so it does not need to be learned. This modified algorithm is still a valid variational EM algorithm as it still increases a lower bound on the marginal likelihood of the training data. Note that this algorithm is only used to learn the shapelet library parameters; at inference time, we proceed with inference as outlined in the main paper, and in Sec. 1 of the supplementary material.

3. Implementation details

For learning, it is infeasible to store the entire training set, and all of the parameters, in GPU memory. For this reason, instead of looking at all patches within each training image, we choose a random subset of patches from each training image to use for learning. We only require enough patches per image to estimate the color palettes, and enough total patches to estimate the shapelet parameters. In practice, we found that selecting a subset of 20% of patches from each training image to use during learning is sufficient. We have not observed significant differences in quantitative results when using up to 50% of patches from each training image. We use the framework of [1] and turn on GPU computation. For inference, we do not select a subset of patches from an image- we run inference using all patches in the image. This is computationally feasible since at inference time, we fix the library of shapelets and the set of universal colors, which makes all images conditionally independent. Therefore, we can do inference one image at a time, and so only have to hold a single image's parameters in GPU memory at a time. We note that because we randomly select a subset of patches from training images during learning, we must run inference on training images in addition to test images, but using all patches in the image this time. For learning, we use a fixed number of 120 iterations for each of the $\hat{R}-1$ variational EM runs, and for inference on an image, we use a fixed number of 50 iterations for the single variational EM run. To help avoid local minima issues associated with the variances of the Gaussian observation models, (σ^2), collapsing to small values prematurely, we fix the variances to be large for the first 10 iterations of inference and learning. We also scale the input data to be in the range $[0...1]$, and clip the values of (σ^2) to be > 0.008 to prevent the Gaussians from collapsing to a single data point. Lastly, for computational efficiency, we consider universal colors on a grid over the color space. Specifically, we define the universal colors to be the centres of the U bins formed by dividing each of the H color channels, which we normalize to be in the range $[0...1]$, into $\sqrt[H]{U}$ bins. For RGB space ($H=3$) and $U=125$, each color channel is divided in 5 bins, and the color centres are described by an 3-dimensional in RGB space. During inference, to assign each of the G global colors to one of the U universal colors, we allowed each image to learn its own set of G colours without regard to the universal colors, and then assigned each of the global colors to its nearest universal color for the last iteration of variational EM.

References

[1] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.