DPP-Nets (Draft)

Max Paulus

August 1, 2017

Setting

Consider a training instance $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} = \{x_t\}_{t=1}^{l}$ is a ground set of size l (e.g. an input text sequence) and $y \in [0,1]^m$ is an m-dimensional target vector (e.g. a sentiment vector). We define a DPP over the input text with l-by-l kernel matrix $\mathbf{L} = \mathbf{V}\mathbf{V}^\top$, [1] where row $\mathbf{v}_i$ of $\mathbf{V}$ corresponds to x_i of $\{x_t\}_{t=1}^{l}$. The DPP should assign high probability to small subsets of words $\mathbf{x}^* = \{x_t\}_{t \in A}$ (where A is a small set of indices), such that predictions $\hat{y} = g(\mathbf{x}^*, \phi)$ made based only on $\mathbf{x}^*$ are relatively good. Our goal is to learn the parameters ϕ for prediction and more difficultly a good kernel matrix for the DPP, i.e. a mapping from each ground set to a kernel matrix $\mathbf{L} = \mathbf{V}\mathbf{V}^\top$. For this purpose, we consider a mapping $\mathbf{v}_i = f(x_i, \theta)$ (e.g. a neural network), whose parameters θ we attempt to learn.[2] Importantly, we do not observe good subsets $\mathbf{x}^*$ and must learn θ without supervision.

Learning

We may learn the parameters ϕ and θ by stochastic gradient descent. Let $C(y, \hat{y})$ be a cost function. Since our prediction pipeline involves a stochastic component (namely which subset $\mathbf{x}^*$ is sampled and used for prediction), we minimise the expected cost

[1] Remember any PSD matrix is the Gramian matrix of a set of vectors

[2] This imposes an implicit constraint on the kernel matrix L. Namely, for two words x_i and x_j appearing both in two different grounds sets $\mathbf{x}_1$ and $\mathbf{x}_2$, the probability assigned to the subset $\{x_i, x_j\}$ by the DPP corresponding to $\mathbf{x}_1$ will be proportional to the probability assigned to the same subset by the DPP corresponding to $\mathbf{x}_2$. This is a modest constraint however, which facilitates learning across grounds sets of different size and composition. The assumption would be relaxed by a direct mapping from ground set to the square root (Cholesky decomposition) of the corresponding kernel matrix $\mathbf{V} = f(\mathbf{x}, \theta)$

with respect to the parameters θ and ϕ. For a single training instance $(\mathbf{x}, \mathbf{y})$, this is

$$\min_{\theta,\phi} \mathop{\mathbb{E}}_{\mathbf{x}^* \sim L_{\theta,\mathbf{x}}} [C(y, g(\mathbf{x}^*, \phi))] \tag{1}$$

The gradient with respect to ϕ is thus given by:

$$\frac{\partial}{\partial \phi} \mathop{\mathbb{E}}_{\mathbf{x}^* \sim L_{\theta,\mathbf{x}}} [C(y, g(\mathbf{x}^*, \phi))] = \mathop{\mathbb{E}}_{\mathbf{x}^* \sim L_{\theta,\mathbf{x}}} [\frac{\partial C}{\partial g(\mathbf{x}^*, \phi)} \frac{\partial g(\mathbf{x}^*, \phi)}{\partial \phi}] \tag{2}$$

The gradient with respect to θ is slightly more complicated and given by:

$$\frac{\partial}{\partial \theta} \mathop{\mathbb{E}}_{\mathbf{x}^* \sim L_{\theta,\mathbf{x}}} [C(y, g(\mathbf{x}^*, \phi))] = \mathop{\mathbb{E}}_{\mathbf{x}^* \sim L_{\theta,\mathbf{x}}} [C(y, g(\mathbf{x}^*, \phi)) \frac{\partial}{\partial \theta} (\log(\det(L_{[\mathbf{x}^*]})) - \log(\det(L+I)))] \tag{3}$$

where

$$\frac{\partial}{\partial \theta} \log(\det(L_{[\mathbf{x}^*]})) = \sum_{i \in |\mathbf{x}^*|} \sum_j \frac{\partial \log(\det(L_{[\mathbf{x}^*]}))}{\partial v_{ij}} \frac{\partial v_{ij}}{\partial \theta} \tag{4}$$

$$\frac{\partial \log(\det(L_{[\mathbf{x}^*]}))}{\partial v_{ij}} = \operatorname{tr}\left(L_{[\mathbf{x}^*]}^{-1} \frac{\partial L_{[\mathbf{x}^*]}}{\partial v_{ij}} \right) \tag{5}$$

and

$$\frac{\partial}{\partial \theta} \log(\det(L + I)) = \sum_i \sum_j \frac{\partial \log(\det(L + I))}{\partial v_{ij}} \frac{\partial v_{ij}}{\partial \theta} \tag{6}$$

$$\frac{\partial \log(\det(L + I))}{\partial v_{ij}} = \operatorname{tr}\left(L^{-1} \frac{\partial (L + I)}{\partial v_{ij}} \right) \tag{7}$$

When v_i are low-dimensional vectors, then (5) and (7) can be calculated efficiently, because the matrix inversion of a low-dimensional matrix is feasible. [2]. This then gives:

$$\operatorname{tr}\left(L_{[\mathbf{x}^*]}^{-1} \frac{\partial L_{[\mathbf{x}^*]}}{\partial v_{ij}} \right) = \left(L_{[\mathbf{x}^*]}^{-1} \right)_{i,\bullet}^{\top} \left(V_{[\mathbf{x}^*]} \right)_{\bullet,j} + \sum_r \left(L_{[\mathbf{x}^*]}^{-1} \right)_{r,i} v_{rj} \tag{8}$$

and

$$\operatorname{tr}\left(L^{-1} \frac{\partial (L + I)}{\partial v_{ij}} \right) = \left(I - V(I + V^\top V)^{-1} V^\top \right)_{i,\bullet}^{\top} (V)_{\bullet,j} + \sum_r \left(I - V(I + V^\top V)^{-1} V^\top \right)_{r,i} v_{rj} \tag{9}$$

As is well-known in the reinforcement learning literature, the stochastic gradient with respect to θ may have large variance. The reason is that we are optimising the sampling procedure by drawing (noisy) samples from the current distribution and only evaluate their associated cost and the score function for them, but don't know how the expected cost will change when another (unobserved) sample is made more likely.

Control Variates

A common estimation strategy to combat the high variance of the stochastic gradient is to employ a control variate. Generally, when estimating $m = \mathbb{E}\,Y$ from noisy samples of $Y = h(X)$, a good control variate Z is a variable that is correlated with Y, has low variance and whose expectation can be computed. Z then reveals some information about whether the simple sample average of $h(Y)$ is likely to over- or underestimate m and gives rise to a correction, which may reduce the variance substantially. Specifically,

$$\hat{m} = \frac{\Sigma_{i=1}^{n} Y_i - c^*(Z_i - \mathrm{E}(Z))}{n} \tag{10}$$

$$c^* = \frac{\mathrm{Cov}(Y, Z)}{\mathrm{Var}(Z)} \tag{11}$$

and c^* may be estimated empirically.

In the setting of unsupervised learning of a DPP, this amounts to finding a variable Z that is correlated with $Y = C(y, g(\mathbf{x}^*, \phi))\frac{\partial}{\partial\theta}(\log(\det(L_{[\mathbf{x}^*]})) - \log(\det(L+I)))$ and whose expectation can be evaluated. An obvious choice is to use the score function itself, i.e.

$$Z = \frac{\partial}{\partial\theta}(\log(\det(L_{[\mathbf{x}^*]})) - \log(\det(L+I))) \tag{12}$$

When the score is used for Z, c^* is known as the baseline. This approach is used in a different thread of research for variational inference. [5] It is further adapted in [4], where c^* is implemented as a neural network that takes $\mathbf{x}$ as an input, i.e. they use an input-dependent baseline. This seems particularly important, in the variational inference setting, where the learning signal is a log-ratio of probability densities, and thus may take very different values for very common or very uncommon $\mathbf{x}$. The use here seems somewhat limited to the case of observing somewhat unusual combinations of $\mathbf{x}$ and $\mathbf{y}$ and learning to reweight their influence on the gradient signal.

Next Step: Code an implementation on the beer recommendations with simple score function control variate. Explore other possible control variates.

Talk about

- talk about write-up; i.e. present; talk about things that may appear unclear to supervisor

  - presentation is independent of rank of V, paper uses between 15-76 dimension; we're probs looking at dim < 300, seems reasonable
  - talk about the subtlety with the DPP mapping, implicit constraint, good parameterization?
  -

- talk about coding implementation
  - will use beer recommendation set, will use same cost function for now, will start coding this week.
- talk about more structured search for control variates
  - Areas of DPP literature that may be useful (approximations/ for matrice properties, i.e. trace/ determinants, but how is this helpful? Always needs to be a random quantity; it should rather be something from statistics, that is related to the score function and can be efficiently computed for DPPS, i.e. using properties of matrices

Control Variates - Guided Search

- An input-dependent baseline may depend on both $\mathbf{y}$ and $\mathbf{x}$
- check applicability of muprop or rebar and other ideas of that stochastic neuron setting.
- search for random quantities beyond the score.

Findings

- Right upon initialization: The DPP does pretty well, returns most of the items most of the time. This is because average subset size is high, but also because orthogonality is roughly preserved. (Hence, interesting to check with non-orthogonal input once the other thing is working). The embedding also looks roughly as it looked after training as a result. But we want sharper edges in the DPP (train longer?) It also suggests that most of the convergence comes from training the predictor whose initial predictions are really bad.
- Suprisingly difficult to train the predictor on the entire bag of words. Very surprisingly.
-

Determinantal Point Processes

A discrete point process $\mathcal{P}$ on a ground set $\mathcal{Y}$ of cardinality N is a probability measure on its power set $2^{\mathcal{Y}}$. Hence, a random sample from $\mathcal{P}$ might be any proper or improper subset of $\mathcal{Y}$. Let $\mathbf{Y}$ be such a random sample. Then, $\mathcal{P}$ is called a *determinantal point process*, if we have for every $A \subseteq \mathcal{Y}$,

$$\mathcal{P}(A \subseteq \mathbf{Y}) = \det(K_A) \tag{13}$$

for some real, symmetric $N \times N$ matrix K, which is known as the marginal kernel. By K_A, we denote the sub-matrix $[K_{ij}]_{i,j \in A}$, which restricts K to the entries indexed by elements in A. For $\mathcal{P}$ to be a valid probability measure, we define $\det(K_\emptyset) = 1$ and require all eigenvalues of K to be bounded between 0 and 1. For this reason, most practical applications restrict the class of DPPs to *L-ensembles* [1]. Through the choice of any positive semi-definite matrix, they directly specify the probability associated with each subset of $\mathcal{Y}$ as

$$\mathcal{P}(\mathbf{Y} = Y) = \frac{\det(L_Y)}{\det(L + I)} \tag{14}$$

This identifies $\mathcal{P}$ as a valid probability measure, because $\Sigma_{Y \subseteq (Y)} \det(L_Y) = \det(L+I)$ [3]. An *L-ensemble* defined by L gives rise to a DPP with marginal kernel

$$K = L(L + I)^{-1} = I - (L + I)^{-1} \tag{15}$$

Regularization

Denote by $|\mathbf{Y}|$ the cardinality of a random subset drawn from a DPP. It is easily seen that $\mathbb{E}[|\mathbf{Y}|]$ is given by the sum of the marginal probabilities of the singletons of $\mathcal{Y}$ and hence,

$$\mathbb{E}[|\mathbf{Y}|] = \mathrm{tr}(K) \tag{16}$$

When we parameterise a DPP through $L = EE^\top$, we can re-write this expression using the singular value decomposition of $E = USV^\top$,

$$\begin{aligned}
\mathbb{E}[|\mathbf{Y}|] &= \text{tr}(L(L+I)^{-1}) \\
&= \text{tr}(US^2U^\top(US^2U^\top + I)^{-1}) \\
&= \text{tr}(US^2U^\top(U(S^2+I)^{-1}U^\top)) \\
&= \text{tr}(US^2(S^2+I)^{-1}U^\top) \\
&= \text{tr}(S^2(S^2+I)^{-1}) \\
&= \sum_{i=1}^{N} \frac{s_i^2}{s_i^2+1}
\end{aligned}$$

This allows us to directly regularise the

References

[1] A. Borodin and E. M. Rains. Eynard Mehta Theorem, Schur Process, and their Pfaffian Analogs. *Journal of Statistical Physics*, 121:291–317, November 2005.

[2] M. Gartrell, U. Paquet, and N. Koenigstein. Low-Rank Factorization of Determinantal Point Processes for Recommendation. *ArXiv e-prints*, February 2016.

[3] Alex Kulesza. *Learning with Determinantal Point Processes*. PhD thesis, University of Pennsylvania, 2012.

[4] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. *CoRR*, abs/1402.0030, 2014.

[5] R. Ranganath, S. Gerrish, and D. M. Blei. Black Box Variational Inference. *ArXiv e-prints*, December 2014.