Diversity Networks:
Neural Network Compression Using Determinantal Point Processes

Zelda Mariet and Suvrit Sra

04/14/2019

Outline

Outline

Stakes and purpose

- Neural Network compression (saving memory)

Stakes and purpose

- Neural Network compression (saving memory)
- Find a method to know which neurons are important (have the main information about the network)

Stakes and purpose

- Neural Network compression (saving memory)
- Find a method to know which neurons are important (have the main information about the network)
- Understand Determinantal Point Processes

Outline

What is a DPP ?

- DPP are probabilistic models of negative correlations, and offer algorithms for sampling, inference... Recently, a new use : Machine Learning
- A DPP is a distribution over subsets $\mathbf{Y}$ of a fixed set of N items (example : distribution of the subset of sentences in a document).
- The binary variables (each items) are negatively correlated $(P(b \in \mathbf{Y} | a \in \mathbf{Y}) < P(b \in \mathbf{Y}))$
- Those negative correlation are derived from a kernel, that defines a global measure of similarity, such as more similar items are less likely to co-occur.
- The more diverse a subset is, the more probable it is

Example

Example : Document Summarization

Example 1 : We consider a document with N sentences. We would like to summarize the document with a few sentences.

Each sentence covers information about the document (each sentence can be represented by vector in the concept space).

For this summarization, we could use a probability measure for each subsets of sentences, with an anticorrelation when 2 sentences are similar, in the space of concepts $\rightarrow$ **This is a DPP**.

Outline

Definitions

Point Process (definition)

A point process $\mathcal{P}$ on a ground set $\mathcal{Y}$ is a probability measure over finite subsets of $\mathcal{Y}$

For instance, Poisson Process : $\mathcal{P}(\{t_1, t_2, t_3\})$ [Over a continuous set].

Now, we will consider only **discrete set** : $\mathcal{Y} = \{0, 1...., N\}$ without loss of generality. A PP is a probability measure over $2^{|\mathcal{Y}|}$.

Determinantal Point Process (definition)

$\mathcal{P}$ is a DPP if $\mathcal{P}$ is a Point Process, and if, when $\mathbf{Y}$ is a random subset according to $\mathcal{P}$, we have :

$$\forall A \subset \mathcal{Y}, \mathcal{P}(A \subset \mathbf{Y}) = det(K_A) \tag{1}$$

for some kernel matrix K (size $N \times N$). K_A is the restriction of K to A.

Some Properties (1)

K is call the **marginal kernel**. Some properties :

- $\forall A \subset \mathcal{Y}$, $I \geq det(K_A) \geq 0$, because it's a probability (every *det* must be in $[0,1]$.
- Thus, K is positive semidefinite. All its eigenvalues λ_n are real, and non negative.
- We can write :

$$\forall A \subset \mathcal{Y}, \quad K_A = \sum_{i \in A} \lambda_i v_i v_i^T \tag{2}$$

- Compute the probability with the eigen values :

$$\forall A \subset \mathcal{Y}, \quad \mathcal{P}(A \subset \mathbf{Y}) = det(K_A) = \prod_{i \in A} \lambda_i \tag{3}$$

Some Properties (2)

- If A={i}, $P(\{i\} \subset \mathbf{Y}) = K_{ii}$
- If A={i,j},

$$P(A \subset \mathbf{Y}) = K_{ii}K_{jj} - K_{ij}^2 = P(i \in \mathbf{Y})P(j \in \mathbf{Y}) - K_{ij}^2 \qquad (4)$$

  . This is why the DPP implies negative correlation ! The off diagonal terms define the anti-correlation. A diagonal kernel is an independent PP.

- The previous formula show the implied diversification. If $K_{ij} = \sqrt{K_{ii}K_{jj}}$, then i and j are perfectly similar, and the 2 items do not co-occur.

Outline

L-ensembles (1)

For the purpose of modeling real data, it is useful to restrict slightly the class of DPPs by focusing on L-ensemble

L-ensembles (definition)

L a real symmetric matrix indexed by elements of $\mathcal{Y}$. A PP is *L-ensemble* if :

$$P_L(\mathbf{Y} = Y) \propto det(L_Y) \tag{5}$$

- This definition is slightly different from the DPP definition (Inclusion vs equality).
- As for K, L must be **positive semidefinite**. But the eigen values of L may not be less than one : any positive semidefinite L defines a L-ensemble.
- **Can we find the normalization** to compute $P_L(\mathbf{Y} = Y)$?

L-ensembles (2)

Normalization theorem (1)

L a real positive semidefinite matrix indexed by elements of $\mathbf{Y}$

$$\forall A \subset \mathcal{Y}, \sum_{A \subset Y \subset \mathbf{Y}} det(L_Y) = det(L + I_{\bar{A}}) \quad (6)$$

where $I_{\bar{A}}$ is the diagonal matrix with ones in the diagonal positions corresponding to elements of $\bar{A}$, and zeros everywhere else.

Normalization theorem (2)

For a L-ensemble $\mathcal{P}$ on a set discrete set $\mathcal{Y}$, we have :

$$\forall Y \subset \mathcal{Y}, P_L(\mathbf{Y} = Y) = \frac{det(L_Y)}{\sum_{Y \subset \mathbf{Y}} det(L_Y)} = \frac{det(L_Y)}{det(L + I)} \quad (7)$$

This is the definition of DPP in the paper. Are L-ensemble DPP ?

L-ensembles (3)

Theorem : L-ensemble is a DPP

An L-ensemble is a DPP, and it's marginal kernel is :

$$K = L(L+I)^{-1} = I - (L+I)^{-1} \tag{8}$$

- **Note** : K, the marginal kernel, can be computed from an eigendecomposition of $L = \sum_{n \in \mathcal{Y}} \lambda_n v_n v_n^T$:

$$K = \sum_{n \in \mathcal{Y}} \frac{\lambda_n}{\lambda_n + 1} v_n v_n^T$$

- **When is a DPP an L-ensembles ?** Only if $(I - K)$ is inversible. We can prove that it is the case if and only if the empty set has a strictly positive probability.
- Modulo this, $L-ensemble$ and DPP are equivalent. L models atomic probabilities of each subset, which offers an appealing use. Most of work on DPP focus on L-ensemble.

Outline

Geometric Interpretation (1)

- **In a** $L - ensemble$**, L is positive semidefinite**. Thus, $\exists B$ s.a $L = B^T B$. [B are "the features vector" in the kernel space]
- (**Recall :** For a matrix $A = (v_1, .., v_n)$, $det(A) = Vol(P)$ where P is the parallelepiped spanned by $(v1..vn)$)
- **Parallelepiped** : $P_L(Y) = det(L_Y) = Vol^2(\{B_i\}, i \in Y)$, which represents the $|Y|$-dimension volume of the parallelepiped spanned by the columns of B, corresponding to elements in Y.
- **If** $(B_i)_{i \in Y}$ **is not linearly independent** , then $P_L(Y) = det(L_Y) = 0$. Moreover, orthogonal features are more likely to co-occur.

Each probability is a mix of **covering information** (size of the vector B_i), and **diversity** (similar B_i are less probable to co-occur because the volume is small).

Geometric Interpretation (2)

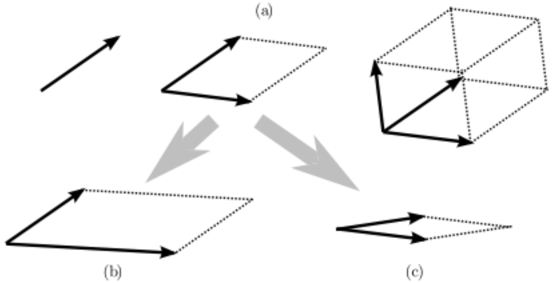


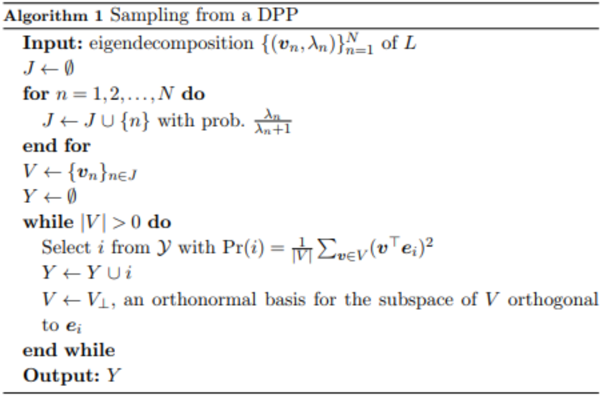
Figure: A geometric view of DPPs: each vector corresponds to an element of Y. (a) The probability of a subset Y is the square of the volume spanned by its associated feature vectors. (b) As the magnitude of an item's feature vector increases, so do the probabilities of sets containing that item. (c) As the similarity between two items increases, the probabilities of sets containing both of them decreas

Geometric Interpretation (3)

Thus, DPP are elegant models to enforce diverse and accurate sampling (or, think DPP as a Point Process with some repulsion : cf Fermions).

Outline

DPP sampling : Algorithm



Algorithm 1 Sampling from a DPP

```
Input: eigendecomposition $\{(\boldsymbol{v}_n, \lambda_n)\}_{n=1}^{N}$ of $L$
$J \leftarrow \emptyset$
for $n = 1, 2, \ldots, N$ do
    $J \leftarrow J \cup \{n\}$ with prob. $\frac{\lambda_n}{\lambda_n + 1}$
end for
$V \leftarrow \{\boldsymbol{v}_n\}_{n \in J}$
$Y \leftarrow \emptyset$
while $|V| > 0$ do
    Select $i$ from $\mathcal{Y}$ with $\Pr(i) = \frac{1}{|V|} \sum_{\boldsymbol{v} \in V} (\boldsymbol{v}^\top \boldsymbol{e}_i)^2$
    $Y \leftarrow Y \cup i$
    $V \leftarrow V_\perp$, an orthonormal basis for the subspace of $V$ orthogonal
    to $\boldsymbol{e}_i$
end while
Output: $Y$
```

Figure: DPP sampling Algorithm

Outline

k-DPP : Definition

- A DPP assigns a probability to every subset of $\mathcal{Y}$. But in many cases, we want to have a control of the cardinality of $Y \subset \mathcal{Y}$.
- We will (like most of the time), restrict our study to L-ensembles DPP.

k-DPP (definition)

A k-DPP on a discrete set $\mathcal{Y} = \{0, ...N\}$ is a distribution over all subsets $Y \subset \mathcal{Y}$. Thus, $\mathcal{P}_L^k$ is obtained by conditionning a standard DPP on the vent that the set Y has cardinality k :

$$\forall Y \subset \mathcal{Y} \text{ such as} |Y| = k, \mathcal{P}_L^K(\mathbf{Y} = Y) = \frac{det(L_Y)}{\sum_{|Y'|=k} det(L_{Y'})} \tag{9}$$

Outline

k-DPP : a few comments

This change has significant implications :

- A k-DPP is **NOT** a DPP (exepct for k=0,1,N-1), because it does not assign any positive probabilities to numerous subset (if you try to construct it, there is a problem)
- **Normalization computation ?** :

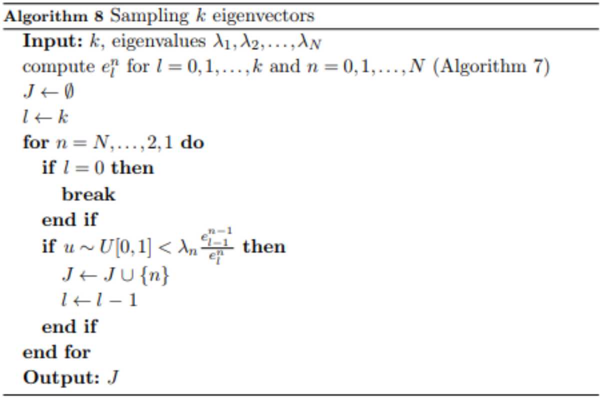
Proposition (k-DPP Normalization)

$$Z_k = \sum_{|Y'|=k} det(L_{Y'}) = \sum_{J \subset \mathcal{Y}, |J|=k} \prod_{n \in J} \lambda_n$$

- Now we have an explicit formulation for normalization, we can design an algorithm similar to DPP to sample a k-DPP.

Outline

k-DPP algorithm



Algorithm 8 Sampling k eigenvectors

```
**Input:** $k$, eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$
compute $e_l^n$ for $l = 0, 1, \ldots, k$ and $n = 0, 1, \ldots, N$ (Algorithm 7)
$J \leftarrow \emptyset$
$l \leftarrow k$
**for** $n = N, \ldots, 2, 1$ **do**
    **if** $l = 0$ **then**
        **break**
    **end if**
    **if** $u \sim U[0,1] < \lambda_n \frac{e_{l-1}^{n-1}}{e_l^n}$ **then**
        $J \leftarrow J \cup \{n\}$
        $l \leftarrow l - 1$
    **end if**
**end for**
**Output:** $J$
```

Figure: k-DPP sampling algorithm

Outline

Definition, Notation (1)

We consider a neural network training. We define :

- $\tau = (t_1, ...t_T)$: training data
- n_l : number of neurons in layer l
- a^l_{ij} : activation of the ith neuron on input t_j of the layer l (output)
- $v^l_i = (a^l_{i1}....a^l_{iT})$: activation vector of the ith neuron of the layer l (size of the training data)
- $w^l_{i,j}$ are the weights from the neuron i of layer $(l-1)$ to neuron j of the layer l.
- **Hypothesis :** All the information of the neurons are in the v^l_i

Why DPP for NN compresion ?

To enforce **diversity** in layer l, we must determine which neurons are computing **redundant information**. Doing so requires finding a maximal subset of independent activation vectors in a layer and retaining only the corresponding neurons. The vector size is very large, so the $v_1 ... v_{n_l}$ are likely linearly independent. We need DPP to select a **diverse subset**.

Definition, Notation (2)

We represent an example below. The compression procedure will have to steps (DIVNET) :

1- k-DPP to sample k neurons in a layer
2- Reweighting procedure, to avoid important consequences on the next layer.

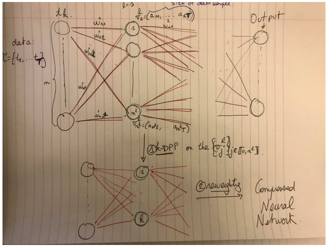


Figure: DIVNET on the first layer for a fully connected neural network

Outline

Constructing the DPP kernel

Steps :

1. **Choice of Kernel** :A lot of potential kernel. Here : Gaussian RBF kernel.
$$\forall i,j \in \mathcal{Y} L''_{i,j} = exp(-\beta ||v_i - v_j||^2)$$
with $\beta = 10/\tau$
2. **Perturbation** :To ensure strict positive definiteness of the kernel matrix L' : $L' = L'' + \epsilon I$ $(\epsilon = 0.001)$
3. **"Quasi-KDPP"** : Let's take $k \in \{0, ...N\}$. The authors claim they are doing a k-DPP by rescaling L, in order to change the expected cardinality of **Y** : $\mathbf{E}(|Y|) = Tr(L(I + L)^{-1})$.
$$L = \gamma L' \text{ where } \gamma = \frac{k(n_l - k')}{k'(n_l - k)} \tag{10}$$
with $k' = \mathbf{E}_{L'}(|Y|)$
But, this is not a real k-DPP ! I don't know why the authors did not compute a real k-DPP (and call this procedure a k-DPP).

Outline

Re-weighting procedure : Fusing redundant neurons (1)

- Idea : we should try to preserve the total information of a given layer with a fusing procedure.
- We want to minimize the consequences of deleting neurons on the next layer $l+1$ for each neuron j:

$$\underset{\bar{w}_j^{l+1\prime}}{\operatorname{argmin}}(||\sum_{i=0}^{k} \bar{w}_{ij}^{l+1\prime} v_i^l - \sum_{i=0}^{n_l} w_{ij}^{l+1} v_i^l||) = \operatorname{argmin}(||\sum_{i}^{k} \delta_{ij}^{l+1} v_i^l - \sum_{i=k+1}^{n_l} w_{ij}^{l+1} v_i^l||)$$

where $\delta_{i,j}^{l+1}$ and $\bar{w}_{ij}^{l+1}$ are respectively the weights differences and the updated weights.

Re-weighting procedure : Fusing redundant neurons (2)

Thus, the final reweighting procedure corresponds to a **projection of each deleted neurons into the space spanned by the selected neurons** v_i^l. The weights are computed according to this projection.

Re-weighting procedure

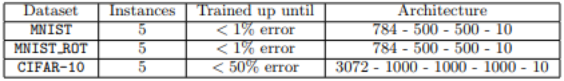
$$\forall j > k, i \leq k, \qquad \alpha^*_{i,j} = \underset{\alpha_{ij}}{\operatorname{argmin}} ||v_j^l - \sum_{i=1}^{k} \alpha_{ij} v_i^l||$$

$$\forall j > k, i \leq k \qquad \bar{w}_{ij}^{l+1} = w_{ij}^{l+1} + \sum_{r=k+1}^{n^l} \alpha^*_{i,r} w_{rj}^{l+1}$$

Outline

Datasets and hypothesis

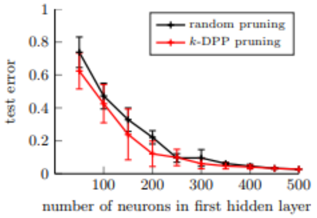
- Experiments on common datasets for neural network evaluation: MNIST (LeCun and Cortes, 2010), MNIST ROT (Larochelle et al., 2007) and CIFAR-10 (Krizhevsky, 2009).
- All networks were trained up until a certain training error threshold, using softmax activation on the output layer and sigmoids on other layers.
- Datasets :



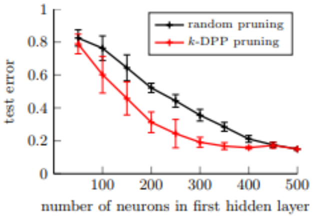
Dataset	Instances	Trained up until	Architecture
MNIST	5	$< 1\%$ error	784 - 500 - 500 - 10
MNIST_ROT	5	$< 1\%$ error	784 - 500 - 500 - 10
CIFAR-10	5	$< 50\%$ error	3072 - 1000 - 1000 - 1000 - 10

Figure: Overview of the sets of networks used in the experiments. We train each class of networks until the first iteration of backprop for which the training error reaches a predefined threshold.

Results (1)



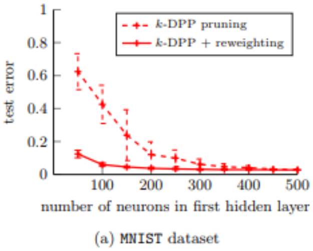
(a) MNIST dataset



(b) MNIST_ROT dataset

Figure: Comparison of random and k-DPP pruning procedures.

Results (2)



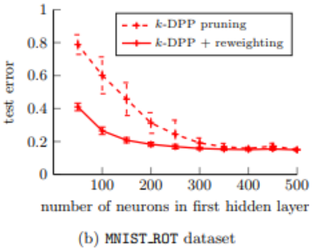
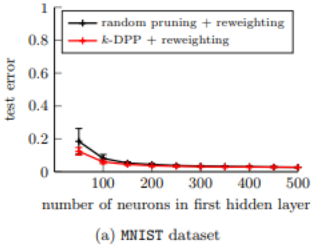


Figure: Comparison of Divnet (k-DPP + reweighting) to simple k-DPP pruning

Results (3)



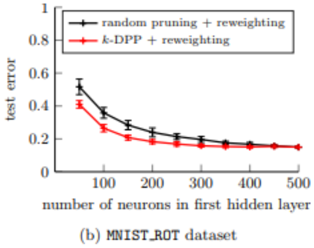
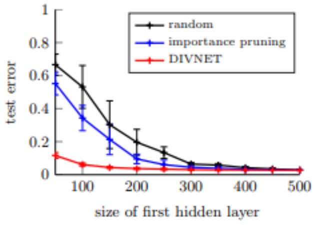
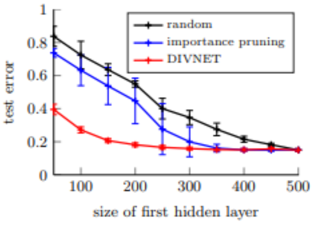


Figure: Comparison of random and k-DPP pruning when both are followed by reweighting.

Results (4)



(a) MNIST dataset



(b) MNIST_ROT dataset

Figure: Comparison of random pruning, importance pruning, and Divnet's impact on the network's performance after decreasing the number of neurons in the first hidden layer of a network.

Conclusion

- We studied the theory of DPP and an interesting application in Machine Learning
- Promising results, which can be combined with others compression techniques
- We considered slight corrections for the current version of the paper (notations, k-DPP).
- We could use different Kernel (learn a kernel ??? We could learn a DPP such as the kernel is the best to compress)
- Use for different NN (CNN etc...)
- Give some ideas to use DPP (efficient sampling, Quasi-MonteCarlo, learning, etc...)