IFT6760A Project: Detailed procedure of Stoudenmire 2018

Tayssir Doghri **Adel Nabli** **Bhairav Mehta**

1 The problem

We have $\{\mathbf{x_1}, ..., \mathbf{x_{N_T}}\}$ a dataset of N_T vectors $\in \mathbb{R}^N$ *(images of size* $\sqrt{N} \times \sqrt{N}$*)* and a feature map

$$\begin{array}{rccc} \Phi & : & \mathbb{R}^N & \rightarrow & \mathbb{R}^{2^N} \\ & & \mathbf{x} & \mapsto & \Phi(\mathbf{x}) \end{array} \tag{1}$$

Here, Φ is defined as a sequence of local feature maps. In fact, if $\forall j \in [\![1, N_T]\!]$ we define $\mathbf{x_j} = [x_j^1, ..., x_j^N]^\top$ as a sequence of pixels, we have that $\Phi(\mathbf{x_j})$ is the vectorization of the following sequence of outer product of N local feature maps:

$$\Phi(\mathbf{x_j}) = \Phi_1(x_j^1)\Phi_2(x_j^2)...\Phi_N(x_j^N) \tag{2}$$

with, $\forall i \in [\![1, N]\!]$, Φ_i the same feature map for all pixels:

$$\begin{array}{rccc} \Phi_i & : & \mathbb{R} & \rightarrow & \mathbb{R}^2 \\ & & x_j^i & \mapsto & \Phi_i(x_j^i) \end{array} \tag{3}$$

Then, if we define the matrix $\mathbf{\Phi} \in \mathbb{R}^{2^N \times N_T}$ the concatenation of all the feature maps for all our dataset, we can say that what we are really interested in is finding $\mathbf{U}$ the matrix that diagonalizes the feature space covariance matrix $\rho = \mathbf{\Phi}\mathbf{\Phi}^\top \in \mathbb{R}^{2^N \times 2^N}$.But, we can also define ρ as:

$$\rho = \sum_{j=1}^{N_T} \left(... \left(\overbrace{\left(\underbrace{\Phi_1(x_j^1)\Phi_1(x_j^1)^\top}_{\in \mathbb{R}^{2\times 2}} \right) \otimes \left(\Phi_2(x_j^2)\Phi_2(x_j^2)^\top \right)}^{\in \mathbb{R}^{4\times 4}} \right) \otimes ... \right) \otimes \left(\Phi_N(x_j^N)\Phi_N(x_j^N)^\top \right) \tag{4}$$

with $\otimes$ the kronecker product for matrices. As diagonalizing a $\mathbb{R}^{2^N \times 2^N}$ is intractable, we have to find an approximation, which is the purpose of the procedure defined in the following section.

Note: *To simplify the notations and as the local feature maps are the same for all pixels, we will use* $\phi_i^j := \Phi_i(x_j^i)$ *from now on.*

2 The procedure

To approximate $\mathbf{U}$, first we define a precision ϵ and then execute the following incremental procedure:

Preprint. Work in progress.

- **Step 1:** define $N/2$ local covariance matrices $\in \mathbb{R}^4$:

$$\rho_{12}^{(1)} = \sum_{j=1}^{N_T} \left(\left(\Phi_1^j \Phi_1^{j\top} \right) \otimes \left(\Phi_2^j \Phi_2^{j\top} \right) \right) \times Tr\left(\Phi_3^j \Phi_3^{j\top} \right) \times ... \times Tr\left(\Phi_N^j \Phi_N^{j\top} \right)$$

$$\rho_{34}^{(1)} = \sum_{j=1}^{N_T} Tr\left(\Phi_1^j \Phi_1^{j\top} \right) \times Tr\left(\Phi_2^j \Phi_2^{j\top} \right) \left(\left(\Phi_3^j \Phi_3^{j\top} \right) \otimes \left(\Phi_4^j \Phi_4^{j\top} \right) \right) \times Tr\left(\Phi_5^j \Phi_5^{j\top} \right) \times ... \times Tr\left(\Phi_N^j \Phi_N^{j\top} \right)$$

$$\vdots$$

$$\rho_{N-1N}^{(1)} = \sum_{j=1}^{N_T} Tr\left(\Phi_1^j \Phi_1^{j\top} \right) \times Tr\left(\Phi_2^j \Phi_2^{j\top} \right) \times ... \times \left(\left(\Phi_{N-1}^j \Phi_{N-1}^{j\top} \right) \otimes \left(\Phi_N^j \Phi_N^{j\top} \right) \right)$$

- **Step 2:** For each local covariance matrix, compute its truncated eigenvalue decomposition and keep the local $\mathbf{U}_{kk+1}^{*(1)}$. Example with $\rho_{12}^{(1)}$:

$$\rho_{12}^{(1)} = \underbrace{\mathbf{U}_{12}^{(1)}}_{\in \mathbb{R}^{4\times 4}} \underbrace{\mathbf{P}_{12}^{(1)}}_{\in \mathbb{R}^{4\times 4}} \underbrace{\mathbf{U}_{12}^{(1)\top}}_{\in \mathbb{R}^{4\times 4}}$$

$$\rho_{12}^{(1)} \simeq \underbrace{\mathbf{U}_{12}^{*(1)}}_{\in \mathbb{R}^{4\times r_{12}}} \underbrace{\mathbf{P}_{12}^{*(1)}}_{\in \mathbb{R}^{r_{12}\times r_{12}}} \underbrace{\mathbf{U}_{12}^{*(1)\top}}_{\in \mathbb{R}^{r_{12}\times 4}}$$

with r_{12} defined as the number of eigenvalues to keep so that the truncation error is less than ϵ. If we suppose that the eigenvalues are ordered in **decreasing order** in $\mathbf{P}_{12}^{(1)}$, then the truncation error E is:

$$E = \frac{\sum_{i=r_{12}}^{4} p_{ii}}{Tr(\rho_{12}^{(1)})} < \epsilon \tag{5}$$

- **Step 3:** From the Φ_i^j and the $\mathbf{U}_{kk+1}^{*(1)\top}$ we obtain $N/2$ new local feature maps. To understand the formula to apply, let's focus on the case of $\rho_{12}^{(1)}$. We have that $\rho_{12}^{(1)}$ is the covariance matrix of the local feature space of dimension $2 \times 2 = 4$ spanned by the Φ_1^j and Φ_2^j. To find the new local feature map, we then apply $\mathbf{U}_{12}^{*(1)\top}$ to all the $\Phi_1^j \otimes \Phi_2^j$ in the vector form. We can think of that as the generalization of the kronecker product to vectors, or as the vectorization of $\Phi_1^j \Phi_2^{j\top}$:

$$\forall j \in [\![1, N_T]\!],\ {\Phi_1^j}_{new} = \mathbf{U}_{12}^{*(1)\top} vec\left(\Phi_1^j \Phi_2^{j\top} \right)$$

In more details, if we have $\Phi_1^j = \begin{pmatrix} \phi_{11}^j \\ \phi_{12}^j \end{pmatrix}$ and $\Phi_2^j = \begin{pmatrix} \phi_{21}^j \\ \phi_{22}^j \end{pmatrix}$, we have:

$${\Phi_1^j}_{new} = \mathbf{U}_{12}^{*(1)\top} \begin{pmatrix} \phi_{11}^j \times \phi_{21}^j \\ \phi_{11}^j \times \phi_{22}^j \\ \phi_{12}^j \times \phi_{21}^j \\ \phi_{12}^j \times \phi_{22}^j \end{pmatrix} \in \mathbb{R}^{r_{12}}$$

- **Step 4:** Repeat steps 1-2-3 until $N_{top} = 2$. The two output dimensions are noted t_1 and t_2.
- **Step 5:** We can define $\tilde{\mathbf{\Phi}}^{t_1 t_2}(\mathbf{x}) = \sum_{s_1 \dots s_N} \boldsymbol{\mathcal{U}}_{s_1 \dots s_N}^{t_1 t_2} \mathbf{\Phi}(\mathbf{x})$ and use the $\mathbf{x}$ from our training set to learn *(in a supervised way)* the top tensor w on the task we want to perform: $f(\mathbf{x}) = \sum_{t_1 t_2} w_{t_1 t_2} \tilde{\mathbf{\Phi}}^{t_1 t_2}(\mathbf{x})$. Again here, if we note $t = t_1 \times t_2$, instead of seeing $\tilde{\mathbf{\Phi}}^{t_1 t_2}$ as an order 2 tensor, we could vectorize it as a t dimensional vector :

$$\forall \mathbf{x},\ \tilde{\mathbf{\Phi}}^{t_1 t_2}(\mathbf{x}) = vec\left({\Phi_1}_{N_{top}}(\mathbf{x}) {\Phi_2^\top}_{N_{top}}(\mathbf{x}) \right) \in \mathbb{R}^t$$

and then training for w means training a linear regression to find the weights $\mathbf{W} \in \mathbb{R}^{l \times t}$ with l the output dimension for our problem *(in a multiclass classification setting, l would be equal to the number of classes)*.