Errata on the paper "End-to-End Kernel Learning with Supervised Convolutional Kernel Networks"

Julien Mairal

This note describes a minor issue in the gradient formula presented in [3], which was noticed by Dexiong Chen when we started working on [1] in 2017. When the experiments of [3] were conducted, the computation of the gradient was regularly checked numerically with finite differences without exhibiting significant discrepancy (for a reason we describe later in this note), suggesting that the effect of this mistake should be minor from a numerical point of view. Indeed, the code accompanying [3][1], which uses the right formula, reproduces closely the results published in [3].[2]

Yet, when using other tasks or datasets, using the correct formula may be important. Below, we briefly review the Nyström kernel approximation method, and show how to do back-propagation with the corresponding encoding function.

Reminder about Nyström encoding. Given data points $\mathbf{x}, \mathbf{x}'$ living in $\mathbb{R}^m$, the paper [3] considers a dot-product kernel $\kappa(\mathbf{x}^\top \mathbf{x}')$ using a smooth enough function $\kappa : \mathbb{R} \to \mathbb{R}$. Then, Nyström's approximation relies on a set of p anchor points represented as the columns of a matrix $\mathbf{Z}$ in $\mathbb{R}^{m \times p}$, and encode an input vector $\mathbf{x}$ with the formula

$$\psi(\mathbf{x}) = \kappa(\mathbf{Z}^\top \mathbf{Z})^{-\frac{1}{2}} \kappa(\mathbf{Z}^\top \mathbf{x}),$$

where, with an abuse of notation, the function κ is applied pointwise.

In [3], the anchor points $\mathbf{Z}$ are learned by back-propagation, which requires differentiating $\psi(\mathbf{x})$ with respect to $\mathbf{Z}$. This is not difficult if one knows how to differentiate with respect to the inverse square root of a positive definite matrix (which was the source of the mistake in [3]). The reason why the formula from [3] did not lead to numerical problems is that the formula was correct to a first approximation, assuming the matrix $\mathbf{A} = \kappa(\mathbf{Z}^\top \mathbf{Z})$ to be diagonally dominant. This turned out to be the case in practice in our experiments, preventing us to numerically spot the issue.

Below, we now provide the correct derivation.

[1] available here `https://gitlab.inria.fr/mairal/ckn-cudnn-matlab`

[2] Note that the code of the follow-up work [1], available here `https://gitlab.inria.fr/dchen/CKN-seq`, also uses the right formula.

Differentiating with respect to $\mathbf{A}^{-1/2}$ when $\mathbf{A}$ is symmetric p.d. First, let us differentiate with respect to the inverse matrix $\mathbf{A}^{-1}$:

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \quad \Longrightarrow \quad \mathbf{A}^{-1}d\mathbf{A} + d(\mathbf{A}^{-1})\mathbf{A} = 0 \quad \Longrightarrow \quad d(\mathbf{A}^{-1}) = -\mathbf{A}^{-1}d\mathbf{A}\mathbf{A}^{-1}.$$

Then, by applying the same (classical) trick,

$$\mathbf{A}^{-\frac{1}{2}}\mathbf{A}^{-\frac{1}{2}} = \mathbf{A}^{-1} \quad \Longrightarrow \quad d(\mathbf{A}^{-\frac{1}{2}})\mathbf{A}^{-\frac{1}{2}} + \mathbf{A}^{-\frac{1}{2}}d(\mathbf{A}^{-\frac{1}{2}}) = d(\mathbf{A}^{-1}) = -\mathbf{A}^{-1}d\mathbf{A}\mathbf{A}^{-1}.$$

Consider now the eigenvalue decomposition $\mathbf{A} = \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^\top$, where $\mathbf{U}$ is orthogonal and $\boldsymbol{\Delta}$ is diagonal with eigenvalues $\delta_1, \ldots, \delta_p$. Then, by multiplying the last relation by $\mathbf{U}^\top$ on the left and by $\mathbf{U}$ on the right.

$$\mathbf{U}^\top d(\mathbf{A}^{-\frac{1}{2}})\mathbf{U}\boldsymbol{\Delta}^{-\frac{1}{2}} + \boldsymbol{\Delta}^{-\frac{1}{2}}\mathbf{U}^\top d(\mathbf{A}^{-\frac{1}{2}})\mathbf{U} = -\boldsymbol{\Delta}^{-1}\mathbf{U}^\top d\mathbf{A}\mathbf{U}\boldsymbol{\Delta}^{-1}.$$

Note that $\boldsymbol{\Delta}$ is diagonal. By introducing the matrix $\mathbf{F}$ such that $\mathbf{F}_{kl} = \frac{1}{\sqrt{\delta_k}\sqrt{\delta_l}(\sqrt{\delta_k}+\sqrt{\delta_l})}$, it is then easy to show that

$$\mathbf{U}^\top d(\mathbf{A}^{-\frac{1}{2}})\mathbf{U} = -\mathbf{F} \circ (\mathbf{U}^\top d\mathbf{A}\mathbf{U}),$$

where $\circ$ is the Hadamard product between matrices. Then, we are left with

$$d(\mathbf{A}^{-\frac{1}{2}}) = -\mathbf{U}(\mathbf{F} \circ (\mathbf{U}^\top d\mathbf{A}\mathbf{U}))\mathbf{U}^\top.$$

When doing back-propagation, one is usually interested in computing a quantity $\bar{\mathbf{A}}$ such that given $\bar{\mathbf{B}}$ (with appropriate dimensions), we have

$$\langle \bar{\mathbf{B}}, d(\mathbf{A}^{-\frac{1}{2}})\rangle_F = \langle \bar{\mathbf{A}}, d\mathbf{A}\rangle_F,$$

see [2], for instance. Here, $\langle , \rangle_F$ denotes the Frobenius inner product. Then, it is easy to show that

$$\bar{\mathbf{A}} = -\mathbf{U}(\mathbf{F} \circ (\mathbf{U}^\top \bar{\mathbf{B}}\mathbf{U}))\mathbf{U}^\top.$$

References

[1] Dexiong Chen, Laurent Jacob, and Julien Mairal. Biological sequence modeling with convolutional kernel networks. *Bioinformatics*, 2019.

[2] Mike B Giles. Collected matrix derivative results for forward and reverse mode algorithmic differentiation. In *Advances in Automatic Differentiation*, pages 35–44. Springer, 2008.

[3] Julien Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems*, 2016.