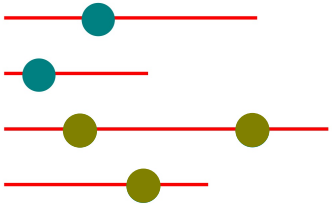
Protein Fold Recognition with Recurrent Kernel Networks

Dexiong Chen[1] Laurent Jacob[2] Julien Mairal[1]

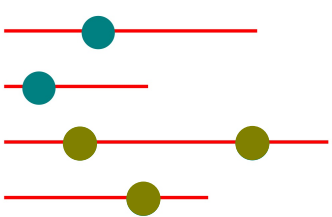
[1]Inria Grenoble [2]CNRS/LBBE Lyon

MLCB 2019, Vancouver

Sequence modeling as a supervised learning problem



Sequence modeling as a supervised learning problem



- Biological sequences $\mathbf{x}_1, \dots \mathbf{x}_n \in \mathcal{X}$ and their associated labels $y_1, \dots, y_n$.
- Goal: learning a **predictive** and **interpretable** function $f : \mathcal{X} \to \mathbb{R}$

$$\min_{f \in \mathcal{F}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))}_{\text{empirical risk, data fit}} + \underbrace{\mu \Omega(f)}_{\text{regularization}}$$

- How do we define the functional space $\mathcal{F}$?

Convolutional kernel networks

Using a string kernel to define $\mathcal{F}$ [Chen et al., 2019]

$$K_{\text{CKN}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{x}'|} K_0(\underbrace{\mathbf{x}[i : i+k]}_{\text{one k-mer}}, \mathbf{x}'[j : j+k])$$

- Kernel methods map data to a high- or infinite-dimensional Hilbert space $\mathcal{F}$ (RKHS). Predictive models f in $\mathcal{F}$ are **linear** forms: $f(\mathbf{x}) = \langle f, \varphi(\mathbf{x}) \rangle_{\mathcal{F}}$.
- Example:

$$\mathbf{x}[i : i+5] := \text{TTGAG} \mapsto \begin{array}{c} \text{A} \\ \text{T} \\ \text{C} \\ \text{G} \end{array} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

[Leslie et al., 2002, 2004]

Convolutional kernel networks

Using a string kernel to define $\mathcal{F}$ [Chen et al., 2019]

$$K_{\text{CKN}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{x}'|} K_0(\underbrace{\mathbf{x}[i : i+k]}_{\text{one k-mer}}, \mathbf{x}'[j : j+k])$$

- Kernel methods map data to a high- or infinite-dimensional Hilbert space $\mathcal{F}$ (RKHS). Predictive models f in $\mathcal{F}$ are **linear** forms: $f(\mathbf{x}) = \langle f, \varphi(\mathbf{x}) \rangle_{\mathcal{F}}$.
- K_0 is a Gaussian kernel over **one-hot** representations of k-mers (in $\mathbb{R}^{k \times d}$).
- A continuous **relaxation** of the mismatch kernel.
- $\varphi(\mathbf{x}) := \sum_{i=1}^{|\mathbf{x}|} \varphi_0(\mathbf{x}[i : i+k])$ with $\varphi_0 : z \mapsto e^{-\alpha/2\|z - \cdot\|^2}$ the kernel mapping associated with K_0.

[Leslie et al., 2002, 2004]

Mixing kernel methods with CNNs

Kernel method

- Rich **infinite-dimensional** models may be learned.
- **Regularization** is natural

$$|f(\mathbf{x}) - f(\mathbf{x}')| \leq \|f\|_{\mathcal{F}} \|\varphi(x) - \varphi(x')\|_{\mathcal{F}}$$

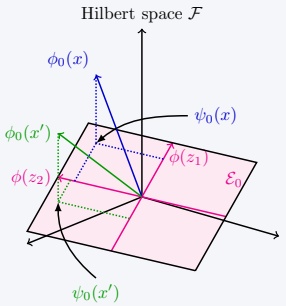
- Representation and classifier learning are **decoupled**.
- **Scalability** limitation.

Mixing kernels with CNNs using approximation

- **Scalable**, **task-adaptive** representations and **data-efficient**.
- No **tricks** (DropOut, batch normalization), parameter-free **initialization**.
- Two ways of learning: **Nyström** and **end-to-end** learning with back-propagation.

Convolutional kernel networks (Nyström approximation)

Nyström approximation



$\mathcal{E}_0 = \text{span}(\varphi_0(z_1), \ldots, \varphi_0(z_q))$

- **Finite-dimensional** projection of the kernel map: given a set of anchor points $Z := (z_1, \ldots, z_q)$, we project $\varphi_0(x)$ for any k-mer x orthogonally onto $\mathcal{E}_0$ such that $K_0(x, x') \approx \langle \psi_0(x), \psi_0(x') \rangle_{\mathbb{R}^q}$.
- An approximate feature map of a sequence $\mathbf{x}$ is

$$\psi(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|} \psi_0(\mathbf{x}[i : i+k]) \in \mathbb{R}^q$$

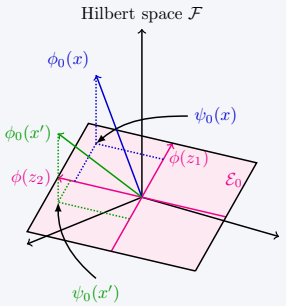
- Then solve the **linear** classification problem

$$\min_{\mathbf{w} \in \mathbb{R}^p} \sum_{i=1}^{n} L(\mathbf{w}^\top \psi(\mathbf{x}_i), y_i) + \mu \|\mathbf{w}\|^2.$$

[Williams and Seeger, 2001, Zhang et al., 2008]

Convolutional kernel networks (end-to-end kernel learning)

Nyström approximation and end-to-end training



$\mathcal{E}_0 = \text{span}(\varphi_0(z_1), \ldots, \varphi_0(z_q))$

- **Finite-dimensional** projection of the kernel map: given a set of anchor points $Z := (z_1, \ldots, z_q)$, we project $\varphi_0(x)$ for any k-mer x orthogonally onto $\mathcal{E}_0$ such that $K_0(x, x') \approx \langle \psi_0(x), \psi_0(x') \rangle_{\mathbb{R}^q}$.
- An approximate feature map of a sequence $\mathbf{x}$ is

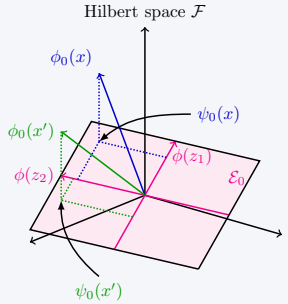
$$\psi(\mathbf{x}) = \sum_{i=1}^{|\mathbf{x}|} \psi_0(\mathbf{x}[i : i+k]) \in \mathbb{R}^q$$

- Then solve

$$\min_{\mathbf{w} \in \mathbb{R}^p,\, Z} \sum_{i=1}^{n} L(\mathbf{w}^\top \psi(\mathbf{x}_i), y_i) + \mu \|\mathbf{w}\|^2.$$

Convolutional kernel networks (end-to-end kernel learning)

Nyström approximation and end-to-end training



$\mathcal{E}_0 = \text{span}(\varphi_0(z_1), \ldots, \varphi_0(z_q))$

- Then solve

$$\min_{\mathbf{w} \in \mathbb{R}^p,\, Z} \sum_{i=1}^{n} L(\mathbf{w}^\top \psi(\mathbf{x}_i), y_i) + \mu \|\mathbf{w}\|^2.$$

- CKN kernels only take contiguous k-mers into account.
- Limitation: unable to capture **gapped motifs** (e.g. useful to model genetic insertions).

From k-mers to gapped k-mers

Gap-allowed k-mers

- For a sequence $\mathbf{x} = x_1 \dots x_n \in \mathcal{X}$ of length n, for a sequence of ordered indices $\mathbf{i} \in \mathcal{I}(k, n)$, we define a k-substring as:

$$\mathbf{x}[\mathbf{i}] = x_{i_1} x_{i_2} \dots x_{i_k}.$$

- The length of the gaps in the substring is

$$\text{gaps}(\mathbf{i}) = \text{number of gaps in the indices.}$$

- Example: $\mathbf{x} = \text{BAARACADACRB}$

$$\mathbf{i} = (4, 5, 8, 9, 11) \quad \mathbf{x}[\mathbf{i}] = \text{RADAR} \quad \text{gaps}(\mathbf{i}) = 3$$

Recurrent kernel networks

Comparing all the k-mers between a pair of sequences

$$K_{\text{CKN}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{x}'|} K_0\left(\mathbf{x}[i : i+k], \mathbf{x}'[j : j+k]\right)$$

[Lodhi et al., 2002, Lei et al., 2017]

Recurrent kernel networks

Comparing all the gapped k-mers between a pair of sequences

$$K_{\mathrm{RKN}}(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{i} \in \mathcal{I}(k, |\mathbf{x}|)} \sum_{\mathbf{j} \in \mathcal{I}(k, |\mathbf{x}'|)} \lambda^{\mathrm{gaps}(\mathbf{i})} \lambda^{\mathrm{gaps}(\mathbf{j})} K_0\left(\mathbf{x}[\mathbf{i}], \mathbf{x}'[\mathbf{j}]\right)$$

- Larger set of partial patterns (i.e. gapped k-mers) is taken into account. $\lambda^{\mathrm{gaps}(\mathbf{i})}$ penalizes the gaps.
- $\varphi(\mathbf{x}) = \sum_{\mathbf{i} \in \mathcal{I}(k, |\mathbf{x}|)} \lambda^{\mathrm{gaps}(\mathbf{i})} \varphi_0(\mathbf{x}[\mathbf{i}])$.
- A continuous relaxation of substring kernel.

[Lodhi et al., 2002, Lei et al., 2017]

Approximation and recursive computation of RKN

Approximate feature map of RKN kernel

The approximate feature map of K_{RKN} via Nyström approximation is

$$\psi(\mathbf{x}) = \sum_{\mathbf{i} \in \mathcal{I}(k,t)} \lambda^{\text{gaps}(\mathbf{i})} \psi_0(\mathbf{x}[\mathbf{i}]),$$

- Exhaustive enumeration of all substrings can be **exponentially** costly.
- But the sum can be computed fast using **dynamic programming** [Lodhi et al., 2002, Lei et al., 2017].
- Leads to a **particular** recurrent neural network with a kernel interpretation.

Results

Protein fold classification on SCOP 2.06 [Hou et al., 2017] (multi-class classification, using more informative sequence features including PSSM, secondary structure and solvent accessibility)

Method	♯Params	Accuracy		Level-stratified accuracy (top1/top5)		
		top 1	top 5	family	superfamily	fold
PSI-BLAST	-	84.53	86.48	82.20/84.50	86.90/88.40	18.90/35.100
DeepSF	920k	73.00	90.25	75.87/91.77	72.23/90.08	51.35/67.57
CKN (128 filters)	211k	76.30	92.17	83.30/94.22	74.03/91.83	43.78/67.03
CKN (512 filters)	843k	84.11	94.29	**90.24/95.77**	82.33/94.20	45.41/69.19
RKN (128 filters)	211k	77.82	92.89	76.91/93.13	78.56/92.98	60.54/83.78
RKN (512 filters)	843k	**85.29**	**94.95**	84.31/94.80	**85.99/95.22**	**71.35/84.86**

Note: More experiments with statistical tests have been conducted in our paper.
[Hou et al., 2017, Chen et al., 2019]

Availability

Our code in Pytorch is freely available at

`https://gitlab.inria.fr/dchen/CKN-seq`

`https://github.com/claying/RKN`

References I

D. Chen, L. Jacob, and J. Mairal. Biological sequence modeling with convolutional kernel networks. *Bioinformatics*, 35(18):3294–3302, 02 2019.

S. Hochreiter, M. Heusel, and K. Obermayer. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736, 2007.

J. Hou, B. Adhikari, and J. Cheng. DeepSF: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 12 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx780. URL `https://doi.org/10.1093/bioinformatics/btx780`.

T. Lei, W. Jin, R. Barzilay, and T. Jaakkola. Deriving neural architectures from sequence and graph kernels. In *International Conference on Machine Learning (ICML)*, 2017.

C. Leslie, E. Eskin, J. Weston, and W. Noble. Mismatch String Kernels for SVM Protein Classification. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2003. URL `http://www.cs.columbia.edu/~cleslie/papers/mismatch-short.pdf`.

C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, volume 7, pages 566–575. Hawaii, USA, 2002.

C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

References II

L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of computational biology*, 10(6):857–868, 2003.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research (JMLR)*, 2:419–444, 2002.

J.-P. Vert, H. Saigo, and T. Akutsu. Convolution and local alignment kernels. *Kernel methods in computational biology*, pages 131–154, 2004.

C. K. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

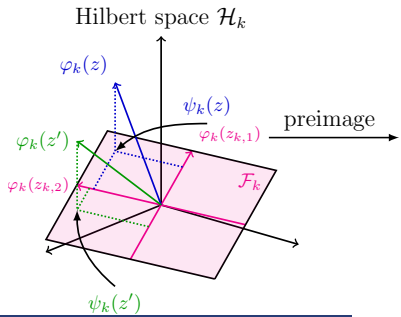
K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nyström low-rank approximation and error analysis. In *International Conference on Machine Learning (ICML)*, 2008.

Motifs in CKN

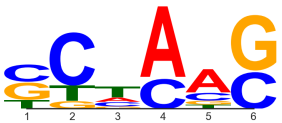
- Find the preimage of each filter at the last layer, by optimizing

$$\min_{\mathbf{y} \in \mathcal{M}} \|\varphi_k(P_k \mathbf{y}_{k-1}) - \varphi_k(z_{k,i})\|^2_{\mathcal{H}_k},$$

  where $\mathcal{M}$ is the appropriate set of motifs.
- Projection onto the simplex induces sparsity thus more informative motif.



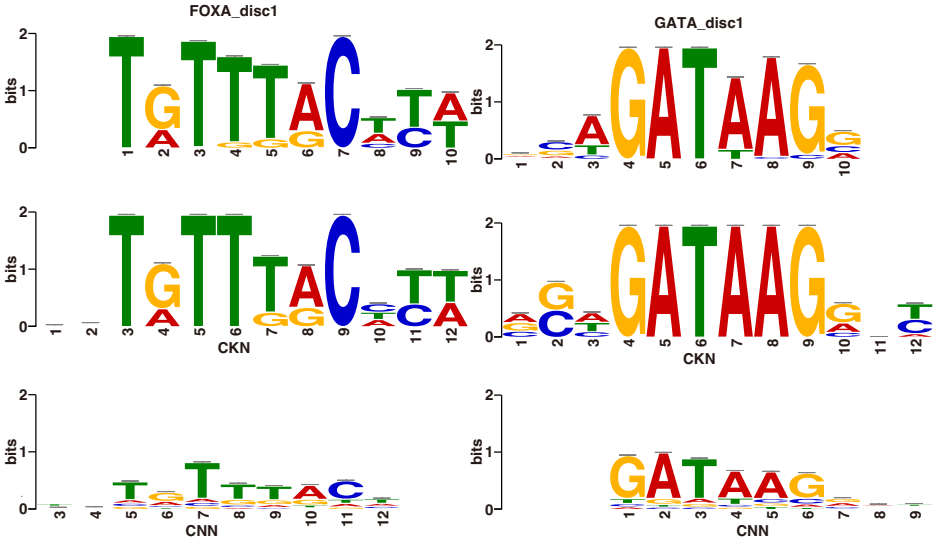
preimage



A	0	0	0.2	0.7	0.6	0
C	0.4	0.7	0.2	0.3	0.2	0.4
G	0.4	0.1	0.1	0	0.1	0.6
T	0.2	0.2	0.5	0	0.1	0

motif associated with $\varphi_k(z_{k,1})$

Logos



A feature map of RKN

A feature vector of $\mathbf{x}$ for RKN is a mixture of Gaussians centered at $\mathbf{x[i]}$, weighted by the corresponding penalization $\lambda^{\mathbf{gaps(i)}}$.

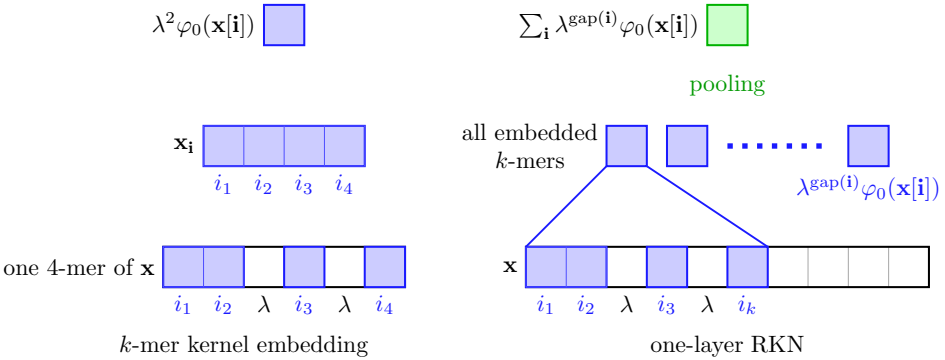


Figure: Example of K_{RKN} for $k = 4$

Computation of recurrent kernel networks

The approximate feature map of K_{RKN} via Nyström approximation is

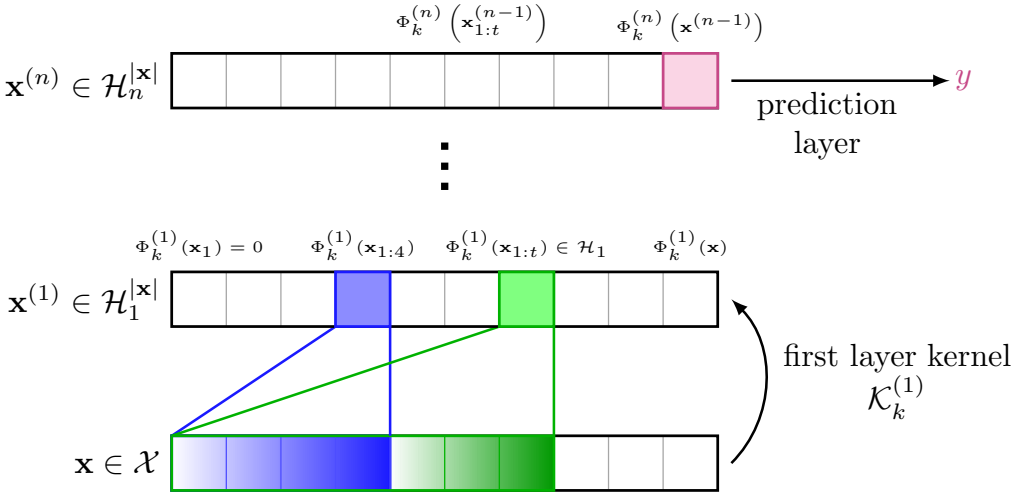
$$\psi_j(\mathbf{x}_{1:t}) = \sum_{\mathbf{i}\in\mathcal{I}(j,t)} \lambda^{\text{gaps}(\mathbf{i})}\psi_0(\mathbf{x}_{1:t}[\mathbf{i}]) = K_{Z_jZ_j}^{-1/2} \sum_{\mathbf{i}\in\mathcal{I}(j,t)} \lambda^{\text{gaps}(\mathbf{i})} K_{Z_j}(\mathbf{x}[\mathbf{i}]) := K_{Z_jZ_j}^{-1/2}\mathbf{h}_j[t],$$

for any $j \in \{1,\dots,k\}$ and $t \in \{1,\dots,|\mathbf{x}|\}$. Z_j is a matrix in $\mathbb{R}^{d\times q}$ whose i-th column is the j-th vector of z_i.
We can prove that $\mathbf{h}_j[t]$ in $\mathbb{R}^q$ obeying some recursion similar to the one used in substring kernel

$$\begin{aligned}
\mathbf{c}_j[1] &= \mathbf{h}_j[1] = 0 && 1 \le j \le k,\\
\mathbf{c}_0[t] &= 1 && 1 \le t \le |\mathbf{x}|,\\
\mathbf{c}_j[t] &= \lambda\mathbf{c}_j[t-1] + \mathbf{c}_{j-1}[t-1] \odot \kappa(Z_j^\top \mathbf{x}_t) && 1 \le j \le k,\\
\mathbf{h}_j[t] &= \mathbf{h}_j[t-1] + \mathbf{c}_{j-1}[t-1] \odot \kappa(Z_j^\top \mathbf{x}_t) && 1 \le j \le k,
\end{aligned}$$

where κ is a non-linear function $\kappa(x) = e^{\alpha(x-1)}$.

Multilayer construction of RKNs



Results

Protein fold recognition on SCOP 1.67 (widely used benchmark)

Method	pooling	one-hot		BLOSUM62	
		auROC	auROC50	auROC	auROC50
SVM-pairwise		0.724	0.359		
Mismatch		0.814	0.467		
LA-kernel		–	–	0.834	0.504
LSTM		0.830	0.566	–	–
CKN		0.837	0.572	0.866	0.621
RKN	mean	0.829	0.541	0.840	0.571
RKN	max	**0.844**	**0.587**	**0.871**	**0.629**
RKN (unsup)	mean	0.805	0.504	0.833	0.570

[Liao and Noble, 2003, Leslie et al., 2003, Vert et al., 2004, Hochreiter et al., 2007, Chen et al., 2019]