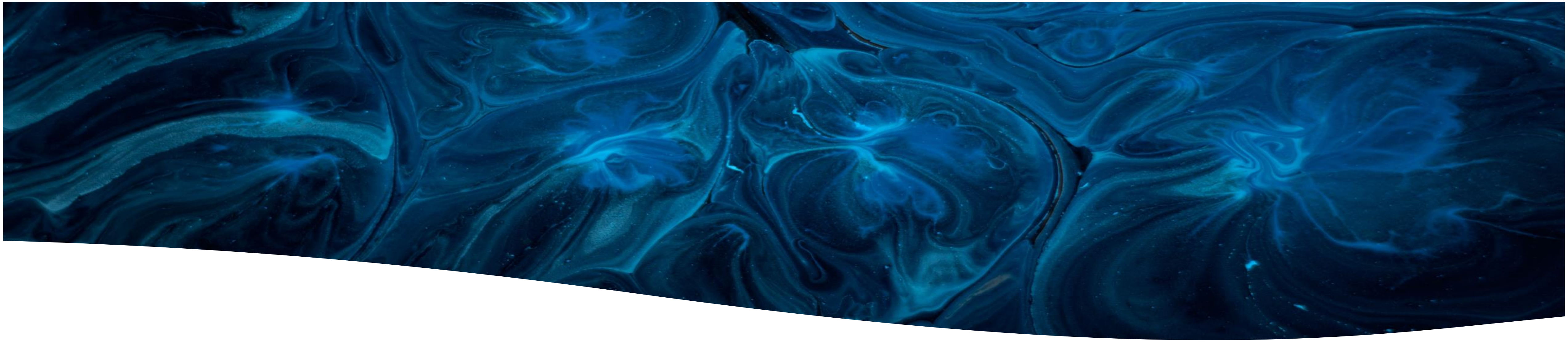


Kernel Based Non-Parametric Activation Functions for Neural Networks [1]

S. Scardapane et al.

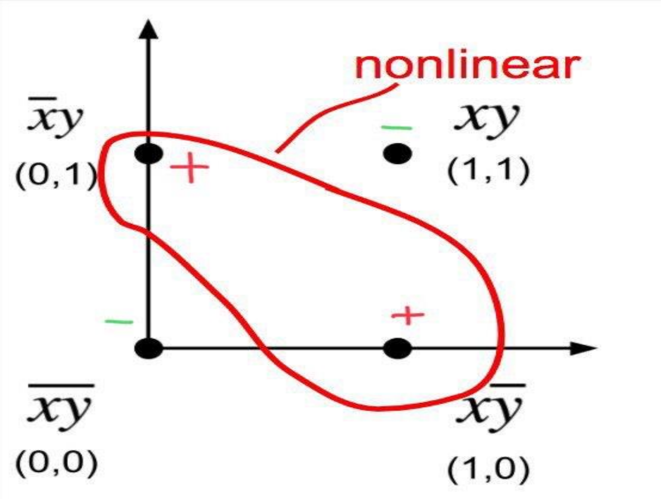
Author: Federico Peconi

Project: Review and result's reproduction of the homonymous paper

Course: Neural Networks for Data Science Applications, Msc in Data Science @Sapienza University of Rome

Activation Functions

$$XOR(x,y) := (x \vee y) \wedge (\overline{x} \vee \overline{y})$$



- As the name suggests, a linear model *cannot* capture non-linear relations between features. E.g. XOR

- To avoid that the composition of two linear models stays linear we need to introduce *non-linearity*

$$\psi(Wx)$$

- Any continuous function can be approximated by some $\psi(Wx)$ [2]

- Common examples of ψ are: *tanh, ReLU, ELU,* ..

Parametric Activation Functions

- Nowadays, the activation functions used inside neural networks are mostly *fixed functions* specifically chosen with respect to the task, architecture or other hyperparameters

- To promote more *flexibility*, several authors proposed the concept of *trainable* activation functions, two main types of such functions take shape:

  1. **Parametric Activation Functions:** - *gen-tanh, PReLU, PELU, SReLU, ..*[3,4,5,6]
     - *Few #parameters, limited flexibility*
  2. **Non-Parametric Activation Functions:** - *APL, Maxout Networks, Spline Activation Functions*[7,8,9]
     - In theory unbounded #parameters, model a large number of shapes

- However, *none* of these approaches has gained wide acceptance in practice

KAF - Overview

- Authors propose a novel non-parametric function:

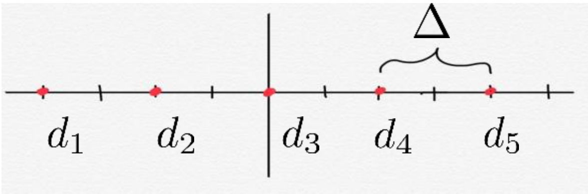
$$\psi(Wx) = Kaf(s) := \sum_{i=1}^{D} a_i k(s, d_i)$$

- Where:
  - $k(\cdot,\cdot) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is a 1-dimensional kernel function
  - D is the user-defined *size* of the dictionary and $\{d_i\}$ are fixed *dictionary elements*
  - $\{a_i\}$ are called *mixing-coefficients* and are the actual *trainable* parameters

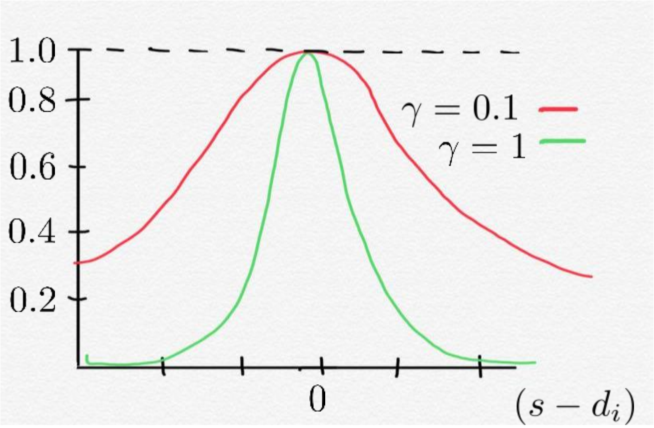
- *BPP* easy to compute: $\frac{\partial Kaf(s)}{\partial a_i} = k(s, d_i)$

- Kernel Function, Parameters initialization ?

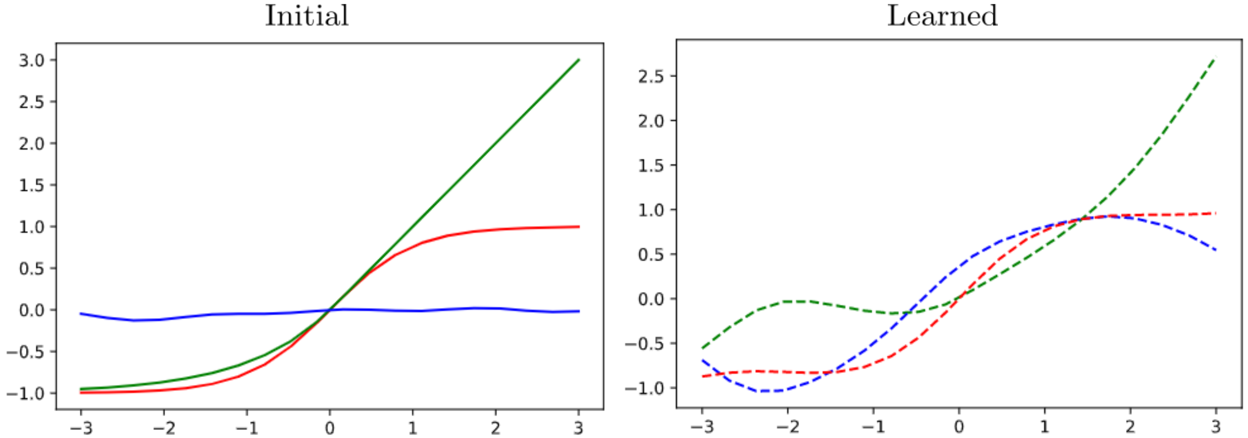
KAF - Design



- Dictionary elements uniformly sampled around 0 with *step size* Δ
- *Gaussian 1D kernel*: $k(s, d_i) = \exp\{-\gamma(s - d_i)^2\}$
  - Heuristic: $\gamma := \frac{1}{6\Delta^2}$
  - Guarantees *local* behavior of mixing coefficients w.r.t. the overall function



- Mixing coefficients are initialized either:
  - *Randomly* from a normal distribution
  - With *ridge-regression* if we want to approx. f : $f(\mathbf{d}) = \mathbf{t} \implies \mathbf{a} = (\mathbf{K} - \varepsilon\mathbf{I})^{-1}\mathbf{t}$
- Such KAF satisfies:
  - *Smoothness* over the entire domain
  - *Linear* number of trainable parameters
  - Admits any *regularization* technique



Experiment - Set Up

- Features normalized between 0 and 1.

- *Validation set* is chosen *randomly* from trainset and used as metric evaluation during training. If no relevant improvement happens after 3 epochs, *early stop* is applied.

- All the networks use a *softmax* in their output layer and the loss function is *cross-entropy* to which we add a small l_2 - regularization factor.

- Weights of linear layers are initialized using the *Uniform He* strategy.

- *Computing platform*: 8th-gen Intel Core i7-8750H CPU, 16GB DDR4 of RAM and exploiting the CUDA capabilities of an NVIDIA GeForce GTX 1050 Ti Max-Q with 4GB of memory. OS is Ubuntu 19.10.

- *TensorFlow v2.1.0*, KAF implemented as a Keras Layer

Experiment – KFNN, SUSY

- *SUSY dataset*: binary classification problem in high energy particle physics [10]
  - Last $500 \cdot 10^3$ records used for test, another $500 \cdot 10^3$ for validation, $500 \cdot 10^6$ dataset size
- Models:
  - *Feed-forward NN*: 5 Dense layers of 300 neurons, ReLU activation and Dropout on the last two layers (p=0.5)
  - *Feed-forward NN*: **2** hidden Dense layers of 300 neurons, **KAF** (both random and ridge init.) activation and Dropout on the last two layers (p=0.5)

Activation Function	Testing AUC	#Trainable Params
$ReLU$	0.8729	367201
$\mathbf{KAF}$	$\mathbf{0.8720}$	$\mathbf{108301}$
$KAF-ELU$	0.8715	108301

Experiment – KCNN, CIFAR10

- *CIFAR10 dataset*: multi label image classification[11]
  - 60000 32×32 colour images in 10 classes

- Block:
  - *conv2D* with 150 filters, filter size 5×5, stride of 1
  - *MaxPooling2D* 3×3 w/ stride of 2
  - Dropout $p = 0.25$

- Models:
  - *CNN* with 4 blocks, *ELU*
  - *KAF-CNN* **2** blocks, *rand*. initialized
  - *KAF-CNN* **2** blocks, *ELU-ridge* initialized

Activation Function	Testing Accuracy	#Trainable Params
ELU	0.7474	1700860
$\mathbf{KAF}$	$\mathbf{0.7485}$	$\mathbf{653600}$
$KAF-ELU$	0.7285	653600

Considerations

- All the details regarding the implementation: https://github.com/arbiter1elegantiae/kaf-nets

- KAFs combine different *beneficial properties* for non-parametric activation functions, whereas introducing only a *linear* number of additional parameters

- They're *effectiveness* is supported with different experiments including CNNs and FNNs

- Future works may focus on:
  - The study of other design choices: *different kernels*, better *heuristics*
  - How well do KAFs perform when extended to different scenarios like *RNNs or Adversarial Attacks*

References

- [1]: Scardapane, S., Van Vaerenbergh, S., Totaro, S. and Uncini, A., 2019. Kafnets: Kernel-based non-parametric activation functions for neural networks. Neural Networks, 110, pp.19-32.
- [2]: Hornik, K., Stinchcombe, M. and White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5), pp.359-366.
- [3]: Chen, C.-T., Chang, W.-D., 1996. A feedforward neural network with function shape autotuning. Neural Networks 9 (4), 627–641.
- [4]: He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proc. IEEE International Conference on Computer Vision (ICCV). pp. 1026– 1034.
- [5]: Trottier, L., Gigu`ere, P., Chaib-draa, B., 2016. Parametric exponential linear unit for deep convolutional neural networks.
- [6]: Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., Yan, S., 2016. Deep learning with S-shaped rectified linear activation units. In: Proc. Thirtieth AAAI Conference on Artificial Intelligence. pp. 1–7.
- [7]: Agostinelli, F., Hoffman, M., Sadowski, P., Baldi, P., 2014. Learning activation functions to improve deep neural networks.
- [8]: Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y., 2013. Maxout networks. In: Proc. 30th International Conference on Machine Learning (ICML). pp. 1–9.
- [9]: Vecci, L., Piazza, F., Uncini, A., 1998. Learning and approximation capabilities of adaptive spline activation function neural networks. Neural Networks 11 (2), 259–270.
- [10]: Baldi, P., Sadowski, P., Whiteson, D., 2014. Searching for exotic particles in high-energy physics with deep learning. Nature communications 5, 4308
- [11]: Cs.toronto.edu. 2020. CIFAR-10 And CIFAR-100 Datasets. [online] Available at: <https://www.cs.toronto.edu/~kriz/cifar.html> [Accessed 20 April 2020].

Thank You