Random Projection Layer for Neural Networks

Diego Marron
Universitat Politecnica de Catalunya, Spain
dmarron@ac.upc.edu

Jesse Read
Aalto University and HIIT,Finland
jesse.read@aalto.fi

Albert Bifet
Télécom ParisTech, Paris, France
albert.bifet@telecom-paristech.fr

Nacho Navarro
Universitat Politecnica de Catalunya, Spain
nacho@ac.upc.edu

1. Motivation

Real-time is a key area of data-mining research, as the number of applications in this area grows.

At the same time, deep learning is becoming increasingly popular. However the use of deep learning on streams is not clear due to the hyper-parameters selection (which makes it harder to provide and of-the-shelf method), and the potential concept-drifting.

In this work we study the application of random neurons to neural works, and see how it helps to improve the detection of concept drift without the need of training them.

We extend ELMs to work with other activations functions, and investigate their application on GPUs to classification in data streams.

2. Contributions

We study the application of random features in the form of random projection layer for neural networks to investigate and compare performance against current state-of-the-art methods.

We use GPUs to implement and evaluate our random projection layer, obtaining powerful predictive performance.

We got encouraging positive results in the empirical evaluation on a collection of real-world datasets. This suggests that using GPUs for data stream mining is a promising research topic for obtaining new fast and adaptive machine learning methodologies.

We highlight important issues, and shed light on promising future directions in approaches to data-stream classification.

3. Activation Functions

Standard Sigmoid:

$$\sigma(a_k) = \frac{1}{1+e^{-a_k}}$$

Where $a_k = W_k$ x input, is the k-th activation function and W is the weight Matrix.

ReLU: and incremental ReLU:

$$z_k = f(a_k) = max(0, a_k)$$

$$z_k = f(a_k) = max(\bar{a}_k, a_k)$$

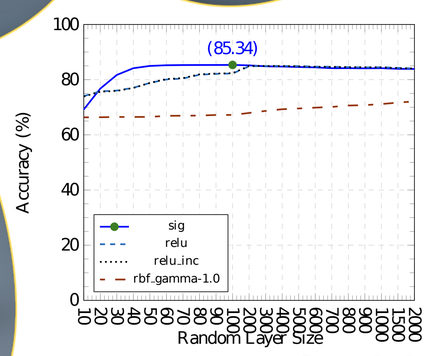
Where the threshold is 0 or the current attribute mean value.

Radial Basis Function:

$$\phi(x) = e^{-\gamma(x-c_i)^2}$$

Using euclidean distance to a random center.

Gammas value used:
1) 0.001
2) 0.01
3) 0.1
4) 1.0
5) 0.0



Activation	Random Neurons	μ	η	Accuracy(%)
SIG	100	0.3	0.11	85.33
ReLu	400	0.3	0.01	84.95
ReLu inc	200	0.3	0.01	84.97
RBF γ=0.001	2000	0.7	1.01	72.13
RBF γ=0.01	2000	0.7	1.01	72.13
RBF γ=0.1	2000	0.7	1.01	72.13
RBF γ=1.0	2000	0.7	1.01	72.13
RBF γ=10.0	2000	0.7	1.01	72.13

Electicity Dataset Evaluation

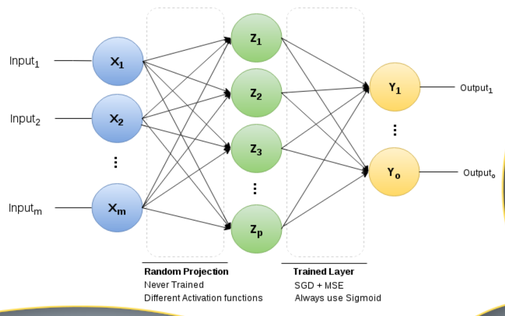
8 attributes, 45312 instances

Randon Init Strategy

Activation	Weight Matrix	Bias Vector
RBF	mean=0.0 and std=1.0	gamma
Sigmoid	mean=0.0 and std=0.9	mean=0.0 and std=0.2
ReLu	mean=0.0 and std=1.0	mean=0.0 and std=0.1
ReLu Inc	mean=0.0 and std=1.0	0.0

Evaluation

2 Layers Feed-forward fully connected netwrok



Comparison

Accuracy (%)

	State of the art				Our method
Dataset	kNN	HT	HT-kNN	LB-HT	GPU
ELEC	78.4 (5)	79.2 (4)	82.5 (3)	89.8 (1)	85.33 (2)
COVT	92.2 (2)	80.3 (5)	91.2 (4)	91.7 (3)	94.59 (1)
SUSY	67.5 (5)	78.2 (2)	77.2 (4)	78.7 (1)	77.63 (3)
avg rank	4	3.67	3.67	1.67	2

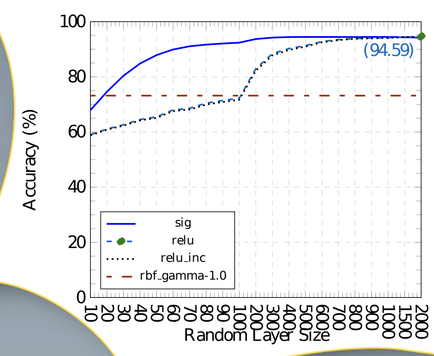
Running Time (s)

	State of the art				Our method
Dataset	kNN	HT	HT-kNN	LB-HT	GPU
ELEC	14 (4)	1 (1)	19 (5)	10 (3)	1.2 (2)
COVT	605 (4)	19 (1)	727 (5)	220 (3)	32 (2)
SUSY	1464 (4)	45 (1)	1428 (4)	520 (3)	172 (2)
avg rank	4	1	4.67	3	2

Covertype Dataset Evaluation

54 attributes, 581012 instances

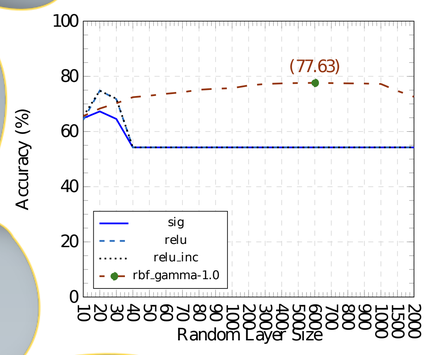
Activation	Random Neurons	μ	η	Accuracy(%)
SIG	1000	0.4	0.11	94.45
ReLu	2000	0.4	0.01	94.59
ReLu inc	2000	0.4	0.01	94.58
RBF γ=0.001	90	0.9	1.01	73.18
RBF γ=0.01	90	0.9	1.01	73.18
RBF γ=0.1	90	0.5	1.01	73.18
RBF γ=1.0	90	0.8	1.01	73.18
RBF γ=10.0	90	1.0	1.01	73.18



SUSY Dataset Evaluation

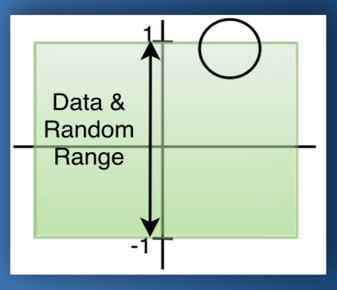
8 attributes, 5000000 instances

Activation	Random Neurons	μ	η	Accuracy(%)
SIG	20	1	0.61	67.28
ReLu	20	1	0.61	74.84
ReLu inc	20	1	0.91	74.80
RBF γ=0.001	600	1	0.71	77.63
RBF γ=0.01	600	1	0.71	77.63
RBF γ=0.1	600	1	0.71	77.63
RBF γ=1.0	600	1	0.71	77.63
RBF γ=10.0	600	1	0.71	77.63

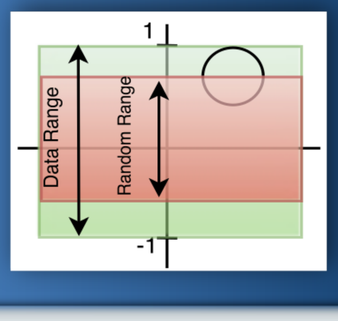


4. Random Initialization

Typically the random space range is the same as input data:



This causes a part of the neuron influence zone located at the edges (or close to it) to be wasted. A more efficient use of the random space:



5. Evaluation Configuration

We implent our experiments using NVIDIA CUDA 7.0 and Tesla K40. The evaluation is done using prequential learnining, and a two layers fully-connected network. Random neurons are never trained, only the last layer.

The time of the GPU evaluation includes the overhead of launching and synchronizing the kernels with the CPU. We cross-validated many configurations and random initialization strategies.

6. Conclusion & Future work

We achieve competitive performance against state-of-the-art methods.

Neural networks are very sensitive to hyper-parameters and initial state, this makes hard to provide and of the shelf method.

We further plan to reduce step by step the number of hyper-parameters, starting by automatically add/prune neurons.

We also noticed there is an overhead on training only one instance at a time on the GPU in order to build the incremental model, we plan to solve this in next studies.

Acknowledgements

NVIDIA GPU CENTER OF EXCELLENCE
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación
UPC UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
GOBIERNO DE ESPAÑA MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

References

G. Huang, What are extreme learning machines? filling the gap between frank rosenblatt's dream and john von neumann's puzzle, Cognitive Computation 7 (3) (2015)

G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, Neural Networks, IEEE Transactions on 17 (4) (2006) 8