Computationally Efficient Radial Basis Function

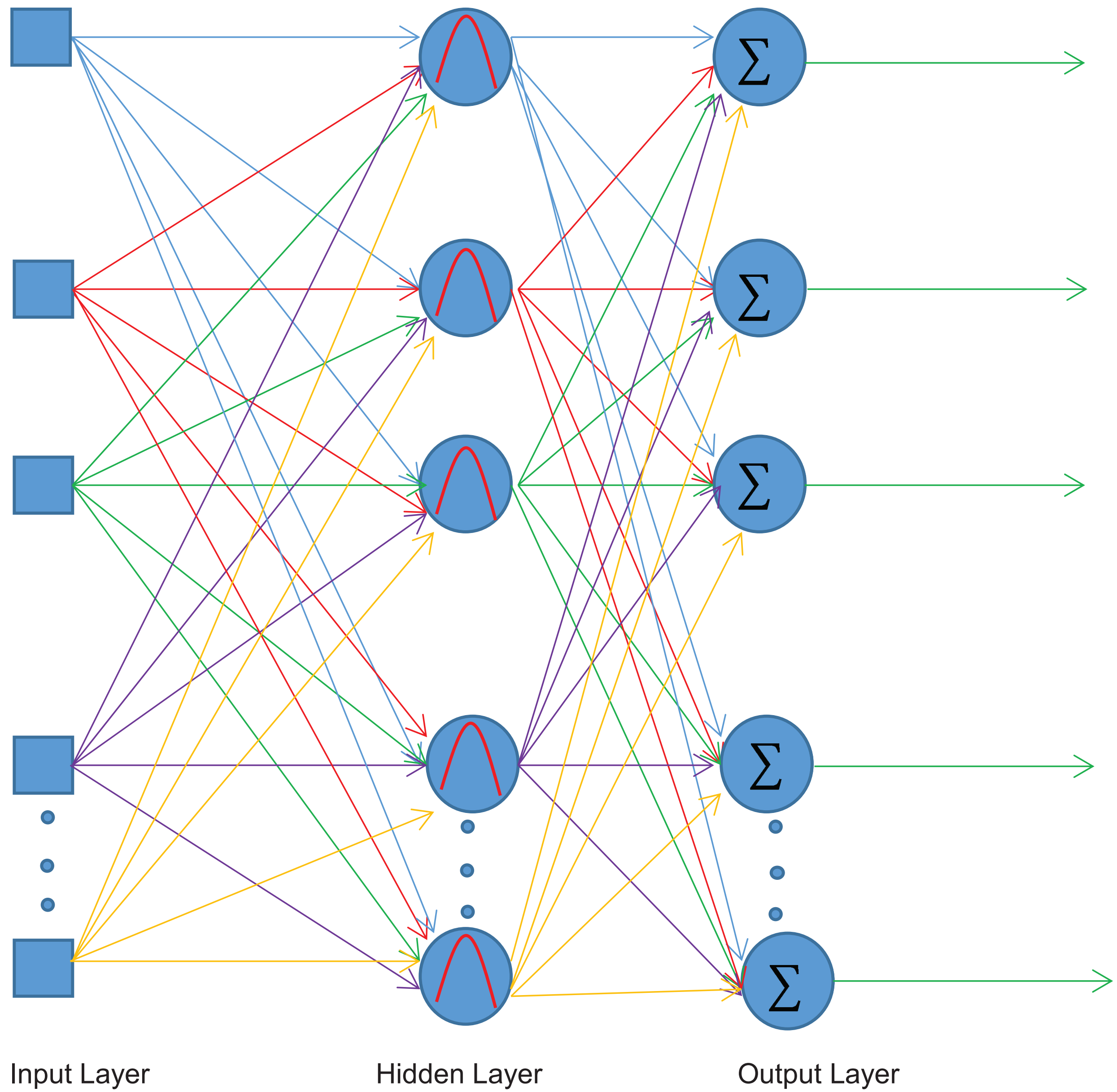
Ms. Adedamola Wuraola, Dr. Nitish Patel

December 2018

Introduction and Motivation

- Hardware ANN
  - Training
  - Inference
- FPGA based, maybe ASIC based (neither Neuromorphic nor GPU/TPU styled)

RBF Networks



RBF Kernels

RBF Kernel	Expression
Gaussian	$f(x) = \exp^{-(\beta x)^2}$
Multiquadric	$f(x) = \sqrt{1 + (\beta x)^2}$
Inverse Multiquadric	$f(x) = \frac{1}{\sqrt{1+(\beta x)^2}}$
Thin-plate Spline	$f(x) = x^2 \log(\beta x^2)$
C^4 Matern	$f(x) = exp^{-\beta x} \cdot (3 + 3\beta x + \beta x)^2$
Approximate Gaussian	$f(x) = \frac{1}{1+(\beta x)^2}$
Approximate Gaussian	$f(x) = \frac{1}{1+(\beta x)^4}$

Disadvantages

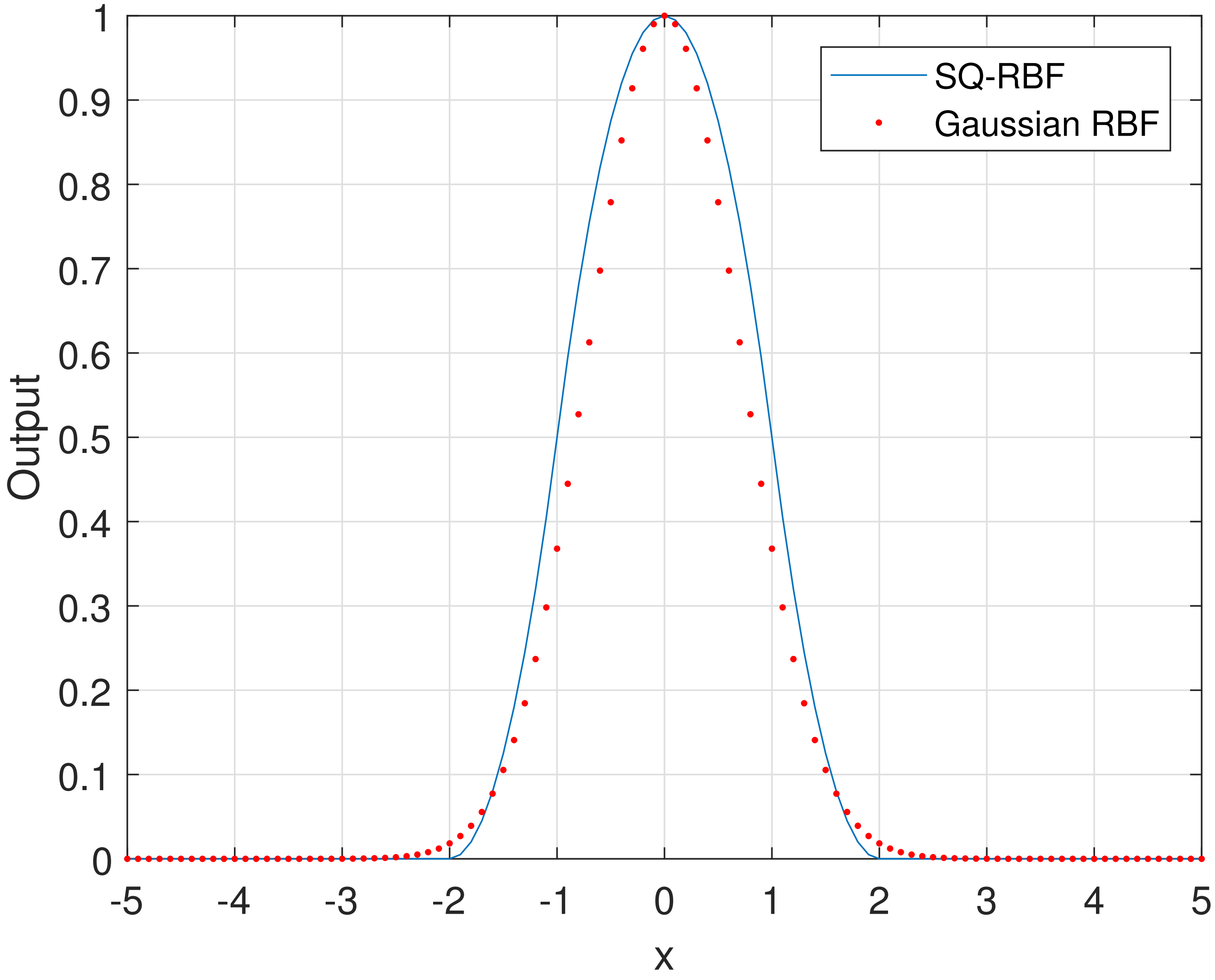
- Computationally Expensive due to Exponent Term
- Computationally Expensive due to Square root

Mapping Function

We define a new convex kernel that makes use of a square-law and eliminates the exponential term present in Gaussian expression. We referred to this expression as SQ-RBF.

$$f(x) = \begin{cases} 1 - \frac{x^2}{2} & : x \leq 1.0 \\ 2 - \frac{(2-x)^2}{2} & : 1.0 \leq |x| < 2.0 \\ 0 & : |x| \geq 2.0 \end{cases} .$$

Mapping Function (cont)



Properties of the SQ-RBF

The function has been named the Square-Radial Basis Function (SQ-RBF) function due to its inherent square operation.

Simple Non-Linearity The square law is, arguably, the simplist non-linearity

Symmetrical and Continuous

Is this mapping comparable to other similar mappings?

- 100 networks trained: 100 weight-sets stored and reused with every experiment
- Only mapping function changed
- Performance Accuracy Criteria:
  - Number of RBF Kernels (Neurons) require to get to a specified MSE
  - Generalizability independent of the number of RBF kernels

Function Approximation Problems

SinE Function This is defined by

$$y = 0.8\exp(-0.2x)\sin(10x) \quad .$$

RBF Kernel	Training Time (seconds)	Test MSE	Number of Neurons
Gaussian	0.6189	**0.0060**	90
SQ-RBF	**0.5555**	0.0067	**84**

RBF Kernel	Training Time (seconds)	Test MSE
Gaussian	5.3898	**2.93x10e-19**
SQ-RBF	**5.1141**	6.75x10e-15

Function Approximation Problems (cont)

Inverse Cosine Function

RBF Kernel	Training Time (seconds)	Test MSE	Number of Neurons
Gaussian	0.1603	0.0213	3
SQ-RBF	**0.1588**	**0.0189**	3

RBF Kernel	Training Time (seconds)	Test MSE
Gaussian	295.28	9.7208×-8
SQ-RBF	**260.94**	**4.4768 x -9**

System Identification Problem

RBF Kernel	Training Time (seconds)	Test MSE	Number of Neurons
Gaussian	5.5499	0.0065	368
SQ-RBF	**4.7737**	**0.0162**	**342**

RBF Kernel	Training Time (seconds)	Test MSE
Gaussian	11.0813	9.09x10e-4
SQ-RBF	**10.82**	**7.49x10e-4**

Mackey-Glass Time Series

RBF Kernel	Training Time (seconds)	Test MSE	Number of Neurons
Gaussian	20.2748	**0.0140**	450
SQ-RBF	**18.6580**	0.0173	**431**

RBF Kernel	Training Time (seconds)	Test MSE
Gaussian	20.1224	0.0091
SQ-RBF	**19.5357**	**0.0060**

Conclusions

- The SQ-RBF is a simple non-linearity
- The execution time is reduced
- The SQ-RBF on an average performs better.
- The SQ-RBF results in smaller number of neurons without compromising the performance accuracy
- However, the variation in performance suggests a strong data set dependence
- Importantly, the SQ-RBF is not inferior to the well established Gaussian RBF.
- Digital circuit implementations are possible