UFME7K-15-M IAS Lecture 5 21/2/2018

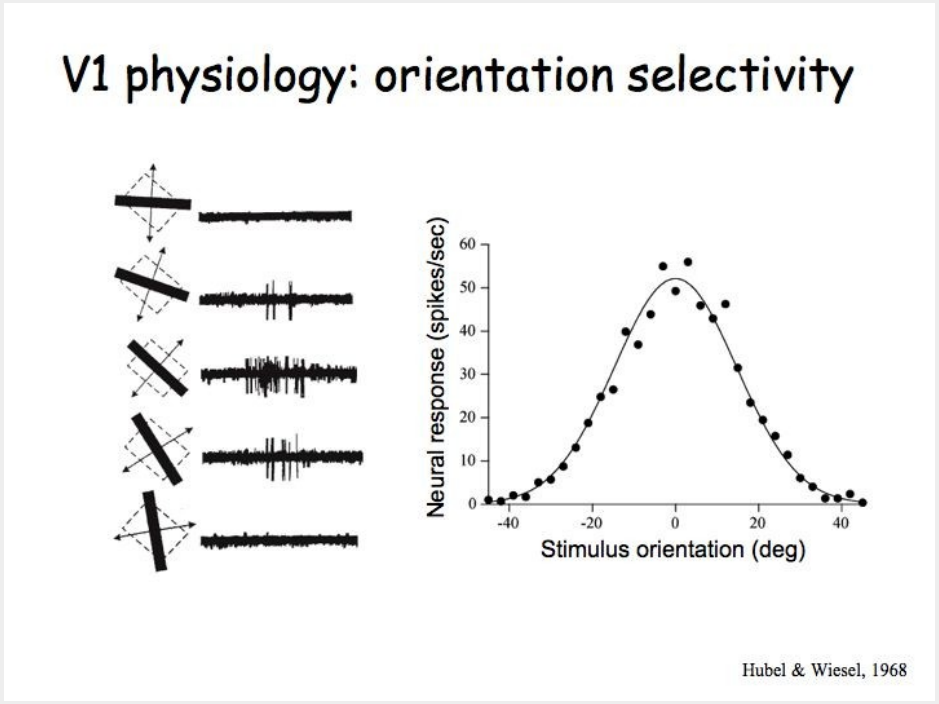
Radial Basis Function (RBF) Networks

Dr J. Charles Sullivan

Localised Receptive Fields

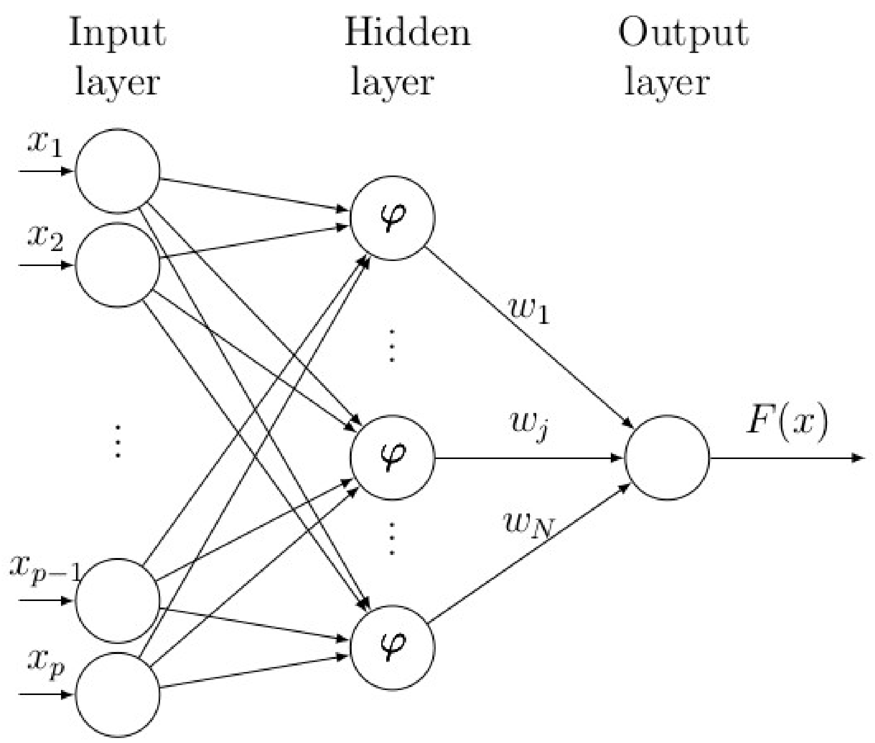
- Locally tuned, overlapping receptive fields have been observed in several areas of the cerebral cortex, most notably in the visual cortex, the auditory cortex and the somatosensory cortex
  - e.g. orientation selective cells in visual cortex
- Moody and Darken (1988) proposed a network structure using biologically inspired local receptive fields to perform function mappings
  - Also motivated by the slow training of MLP networks
- Similar methods have been applied in interpolation and approximation theory
  - e.g. Broomhead and Lowe (1988)

Localised Receptive Fields in Primary Visual Cortex



- data from Hubel and Wiesel's early experiments on V1.
- The dashed rectantangles on the left indicate a V1 neuron's receptive field. The superimposed lines are the stimuli that were used
- For each stimulus (each line orientation), Hubel and Wiesel recorded how many action potentials were produced by this neuron
- The middle row are examples of the electrophysiological recordings for each of the corresponding stimulus orientations.
- The diagonal orientation (the middle row) evoked the greatest response (largest number of action potentials)
- The graph on the right shows the result after a number of measurements like these for each of a bunch of lines.
- There is a peak response for one particular orientation and weaker responses for other orientations, falling to zero when the line orientaton is about 40 degrees away from the neurons favorite (preferred) orientation).
- From http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/V1/lgn-V1.html

Architecture of a simple RBF Network



Activation Function

Activation of a hidden unit is determined by the DISTANCE between the input vector x and prototype vector μ

$$a_i = \varphi_i(\vec{x}) = \varphi_i(\|\vec{x} - \vec{\mu}_i\|),\ i = 1,2,\ldots,H$$

Where $\varphi_i()$ Is typically a Gaussian function :

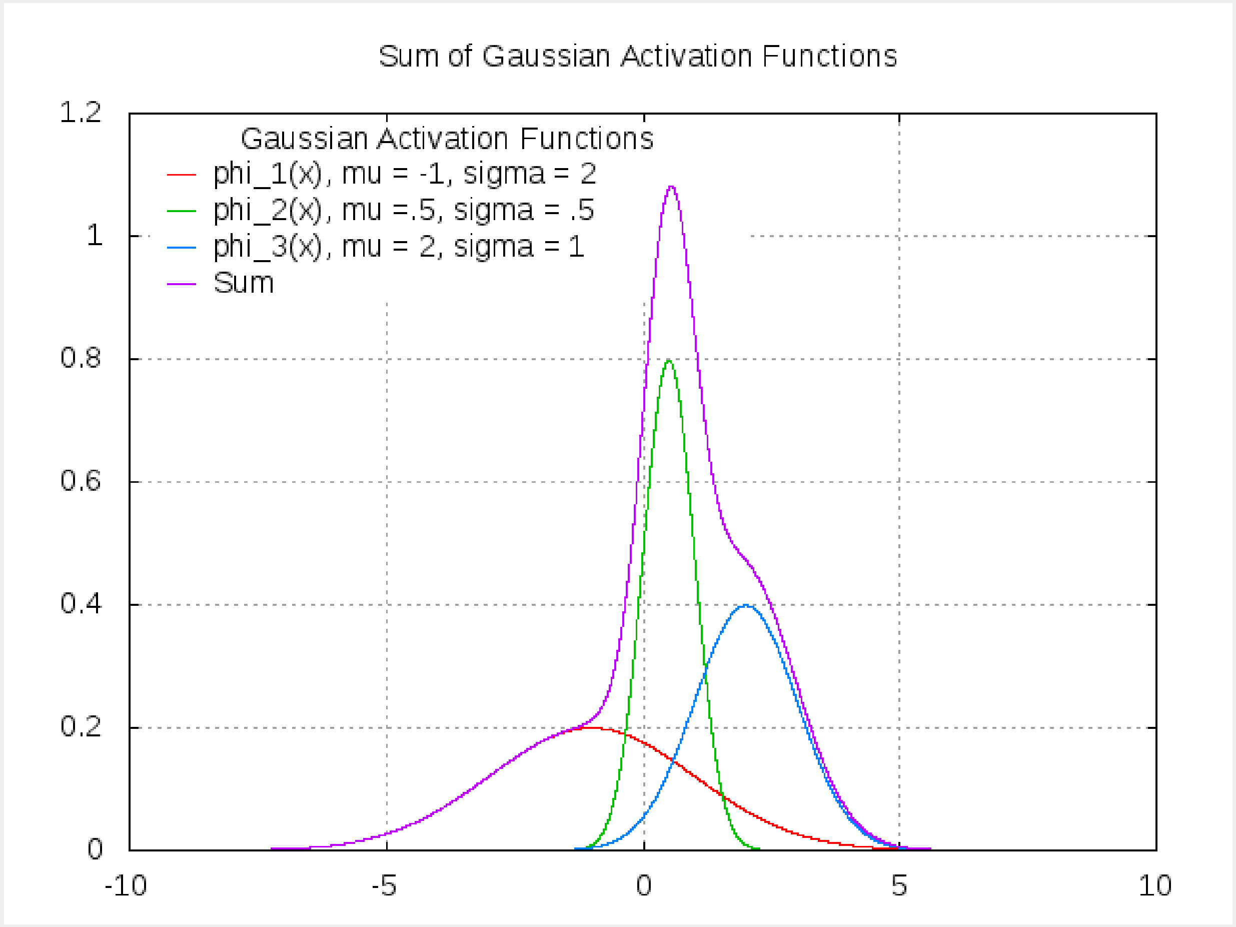
$$\varphi_i(\vec{x}) = \exp\left(\frac{-\|\vec{x} - \vec{\mu}_i\|^2}{2\sigma_i^2}\right)$$

Outputs are formed by weighted linear sum

$$y_k(x) = \sum_{i=1}^{M} w_{ki} \varphi_i(x)$$

- If we know the correct values at the training (target) points,
  - Optimal weights can be determined by inverting the phi matrix
  - or by Least Mean Squares

Addition of Gaussians



How to determine positions of centres of receptive fields?

- Could put one at each training point
  - exact interpolation
  - very fast (matrix inversion)
  - but computationally expensive if a lot of data
  - data may not be uniformly sampled
  - available in MATLAB as **`newrbe`**

- Can use unsupervised learning algorithms
  - e.g. k-means clustering

- A simple approach is to start with a small number, uniformly spread over input range
  - add more nodes iteratively until a performance criterion is satisfied
  - this is what MATLAB **`newrb`** does

Training

- The weights in the output sum have to be determined
  - Often a simple LMS (Least Mean Squares) algorithm is used
  - Much faster than gradient descent (cf MLPs)
  - Since the output sum is linear, it is not prone to local minima
- Typically learning also involves determining the centres (and possibly also widths) of the receptive fields
  - Can use unsupervised learning (e.g. clustering)
  - Note that for multi-dimensional problems, the widths could also be different in each dimension
  - However, this makes it a non-linear problem

Least Squares Learning

When applied to supervised learning with linear models the least squares principle leads to a particularly easy optimisation problem.

If the model is

$$f(\vec{x}) = \sum_{j=1}^{m} w_j \varphi_j(\vec{x})$$

and the training set is

$$\{(\vec{x}_i, \hat{y}_i)\}_{i=1}^{p}$$

then the least squares method is to

minimise the *sum-squared-error*

$$S = \sum_{i=1}^{p} (\hat{y}_i - f(\vec{x}_i))^2$$

with respect to the weights of the model.

Regularisation

If a weight penalty term is added to the sum-squared-error,

as is the case with ridge regression (regularisation),

then the following *cost function* is minimised

$$C = \sum_{i=1}^{p} \left(\hat{y}_i - f\left(\vec{x}_i\right)\right)^2 + \sum_{j=1}^{m} \lambda_j w_j^2$$

where the $\left\{\lambda_j\right\}_{j=1}^{m}$

are regularisation parameters

Comparison with the MLP

- Architecturally similar to MLP but RBFs usually have only one hidden layer
- RBF neurons have local (cf. MLP global) coverage
- RBFs are usually much faster than MLPs to train and often faster to evaluate
  - But may need more training data to achieve comparable accuracy
- Not so good at interpolating over “gaps” in the data, or extrapolating
- Very prone to the “curse of dimensionality”

MATLAB newrb Function

`MATLAB function newrb` creates a Radial Basis Function Network by iteratively adding neurons to the hidden layer until it meets the specified mean squared error goal.

`net = newrb(P,T,goal,spread,MN,DF)` takes the following arguments:-

P	R-by-Q matrix of Q input vectors
T	S-by-Q matrix of Q target class vectors
goal	Mean squared error goal (default = 0.0)
spread	Spread of radial basis functions (default = 1.0)
MN	Maximum number of neurons (default is Q)
DF	Number of neurons to add between displays (default = 25)

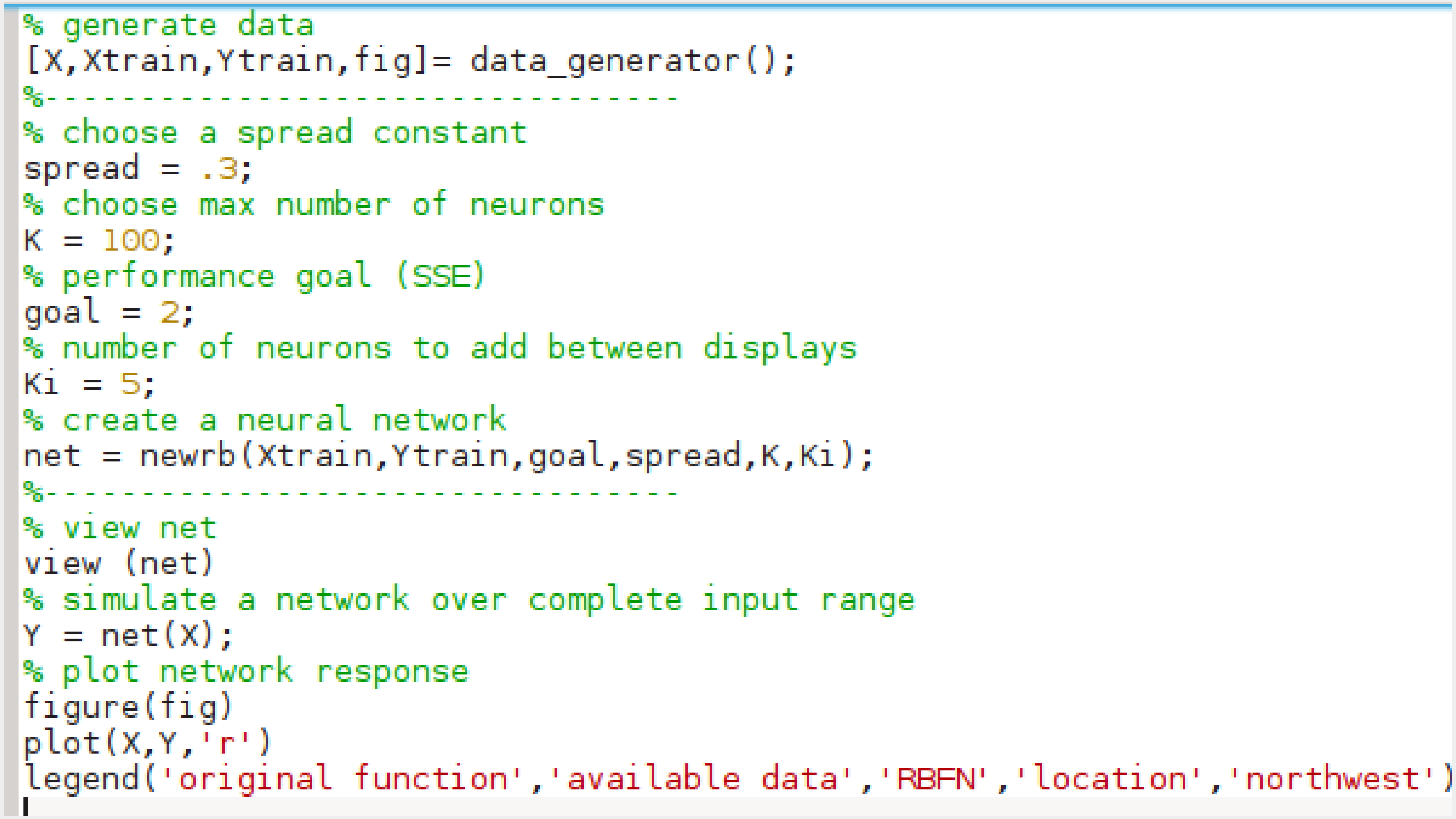
The larger `spread` is, the smoother the function approximation.

Too large a spread means a lot of neurons are required to fit a fast-changing function.

Too small a spread means many neurons are required to fit a smooth function, and the network might not generalize well.

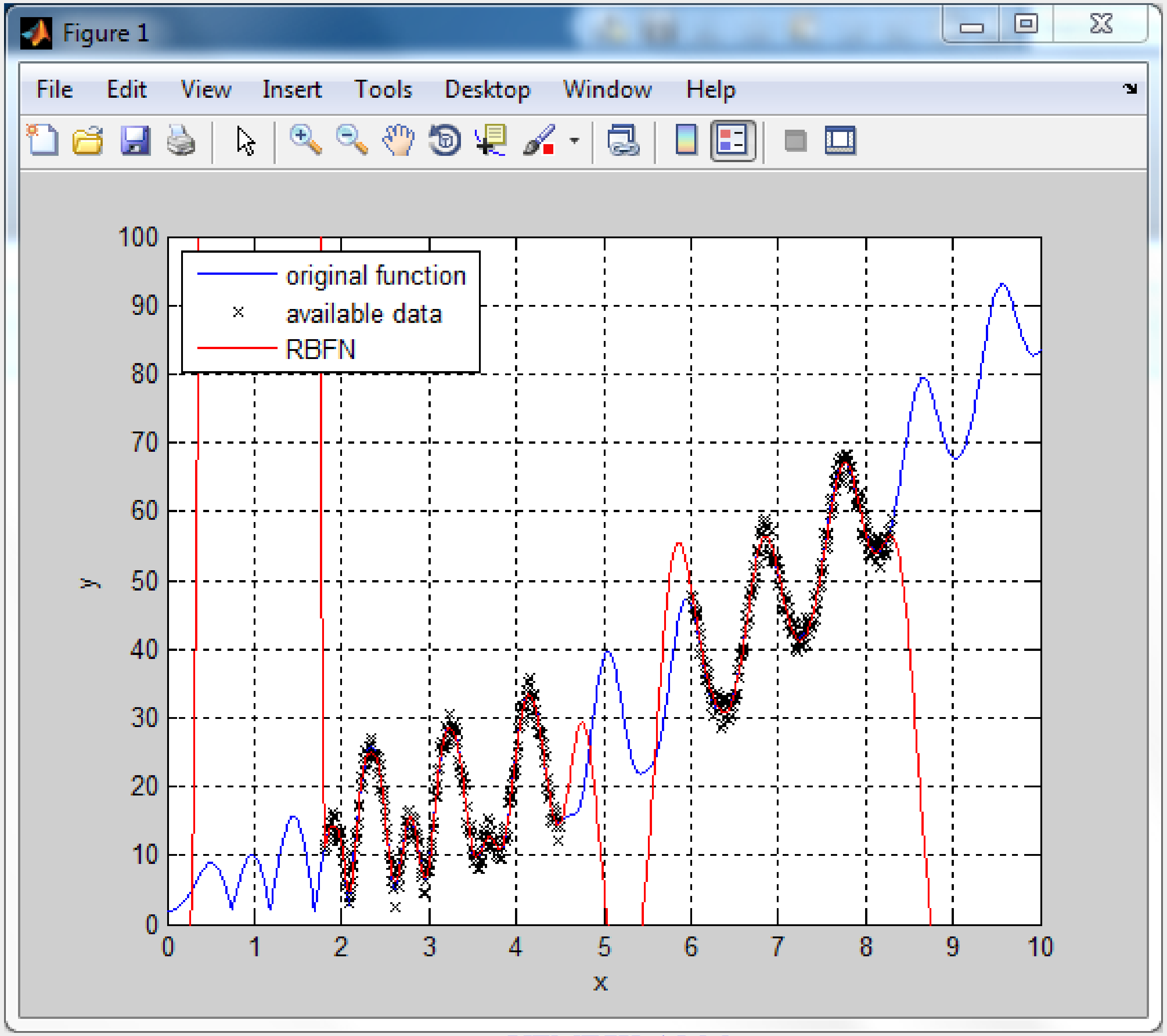
Call `newrb` with different spreads to find the best value for a given problem.

Example using newrb

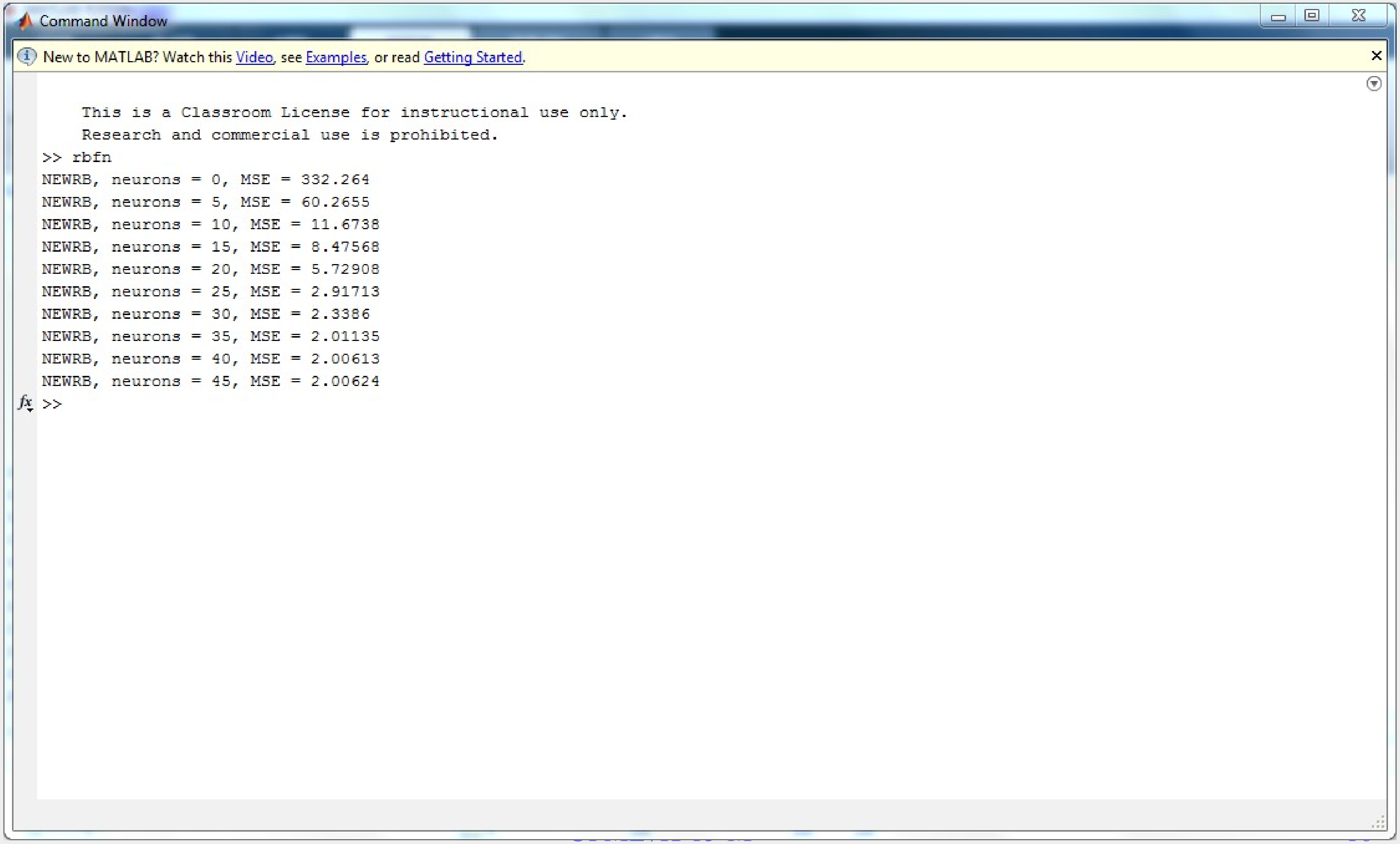


```
% generate data
[X,Xtrain,Ytrain,fig]= data_generator();
%-----------------------------------
% choose a spread constant
spread = .3;
% choose max number of neurons
K = 100;
% performance goal (SSE)
goal = 2;
% number of neurons to add between displays
Ki = 5;
% create a neural network
net = newrb(Xtrain,Ytrain,goal,spread,K,Ki);
%-----------------------------------
% view net
view (net)
% simulate a network over complete input range
Y = net(X);
% plot network response
figure(fig)
plot(X,Y,'r')
legend('original function','available data','RBFN','location','northwest')
```

newrb: output compared to original function



newrb: Error reduction as neurons are added


Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
    This is a Classroom License for instructional use only.
    Research and commercial use is prohibited.
>> rbfn
NEWRB, neurons = 0, MSE = 332.264
NEWRB, neurons = 5, MSE = 60.2655
NEWRB, neurons = 10, MSE = 11.6738
NEWRB, neurons = 15, MSE = 8.47568
NEWRB, neurons = 20, MSE = 5.72908
NEWRB, neurons = 25, MSE = 2.91713
NEWRB, neurons = 30, MSE = 2.3386
NEWRB, neurons = 35, MSE = 2.01135
NEWRB, neurons = 40, MSE = 2.00613
NEWRB, neurons = 45, MSE = 2.00624
>>
```

Universal Approximation

- Hartman et al. (1990) give a formal proof of this property for networks with Gaussian basis functions in which the widths of the Gaussians are treated as adjustable parameters.
- A more general result was obtained by Park and Sandberg (1991) who show that, with only mild restrictions on the form of the kernel functions, the universal approximation property still holds.
- As with the corresponding proofs for multi-layer perceptron networks, these are existence proofs which rely on the availability of an arbitrarily large number of hidden units, and they do not offer practical procedures for constructing the networks.
- Nevertheless, these theorems are crucial in providing a theoretical foundation on which practical applications can be based with confidence.

Best Approximation

- Girosi and Poggio (1990) have shown that **radial basis function networks possess the property of best approximation**.

- An approximation scheme has this property if, in the set of approximating functions (i.e. the set of functions corresponding to all possible choices of the adjustable parameters) there is **one function which has minimum approximating error for any given function to be approximated**.

- They also showed that this property is **not** shared by multi-layer perceptrons.

Applications

- Radial basis functions (RBFs) have emerged as a powerful tool for scattered data approximation in high dimensional spaces
- Successfully applied in various applications including :-
  - geography and digital terrain modelling
  - data assimilation in geodesy and metrology
  - engineering design and mesh generation
  - neural networks and artificial intelligence
  - expensive function optimisation
  - mesh-free methods
  - solving partial differential equations (PDEs) on surfaces
  - post-processing of simulation and 3D surface reconstruction
  - sampling, signal processing and machine learning

Curse of Dimensionality

- If the data contains features on a small spatial scale, the RBF network requires a large number of neurons with narrow receptive fields to achieve an accurate fit
- The computational cost, and in particular, memory usage then increases steeply with increasing numbers of dimensions
- This “combinatorial explosion” also occurs with Fuzzy Inference Systems and other related approaches
- There are techniques which can be used to “prune” the numbers of neurons and also to adapt the spread of each neuron but these have an adverse effect on the learning times

Similarity to ANFIS

- Jang and Sun(1993) demonstrates the **functional equivalence** between radial basis function networks and a simplified class of fuzzy inference systems.

- Though these two models are motivated from different origins (RBFNs from physiology and fuzzy inference systems from cognitive science), they share common characteristics
  - not only in their operations on data,
  - but also in their learning process to achieve desired mappings

- Jang and Sun showed that under some minor restrictions, they are functionally equivalent;
  - the learning algorithms and the theorem on representational power for one model can be applied to the other, and vice versa.

Summary

- RBF networks are an alternative to MLP networks
- Particularly suited to curve-fitting and scattered data approximation problems
- In many applications, RBFs greatly out-perform MLPs in speed of learning
  - Can achieve similar levels of accuracy but may need more training data
- In high-dimensional problems, RBFs can be particularly prone to the “curse of dimensionality”

References

- Jang, J-SR and Sun, C-T (1993)

  Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Transactions on Neural Networks, vol 4, 1, pp 156-159

- J. Moody and C. Darken (1988) Learning with localized receptive fields. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, Proc. of the 1988 Connectionist Models Summer School. Carnegie Mellon University, Morgan Kaufmann Publishers, 1988

- Broomhead, D. S. and D. Lowe (1988).

  Multivariate functional interpolation and adaptive networks. Complex Systems 2, 321-355.

- Girosi, F. and T. Poggio (1990).

  Networks and the best approximation property. Biological Cybernetics 63 , 169-176.

- MATLAB NN Toolbox User Guide 2014b Edition. Mathworks Inc.