UNIVERSITÀ PARTHENOPE

DiST

DIPARTIMENTO DI SCIENZE E TECNOLOGIE

Machine Learning (part II)

Radial Basis Functions Neural Network

Angelo Ciaramella

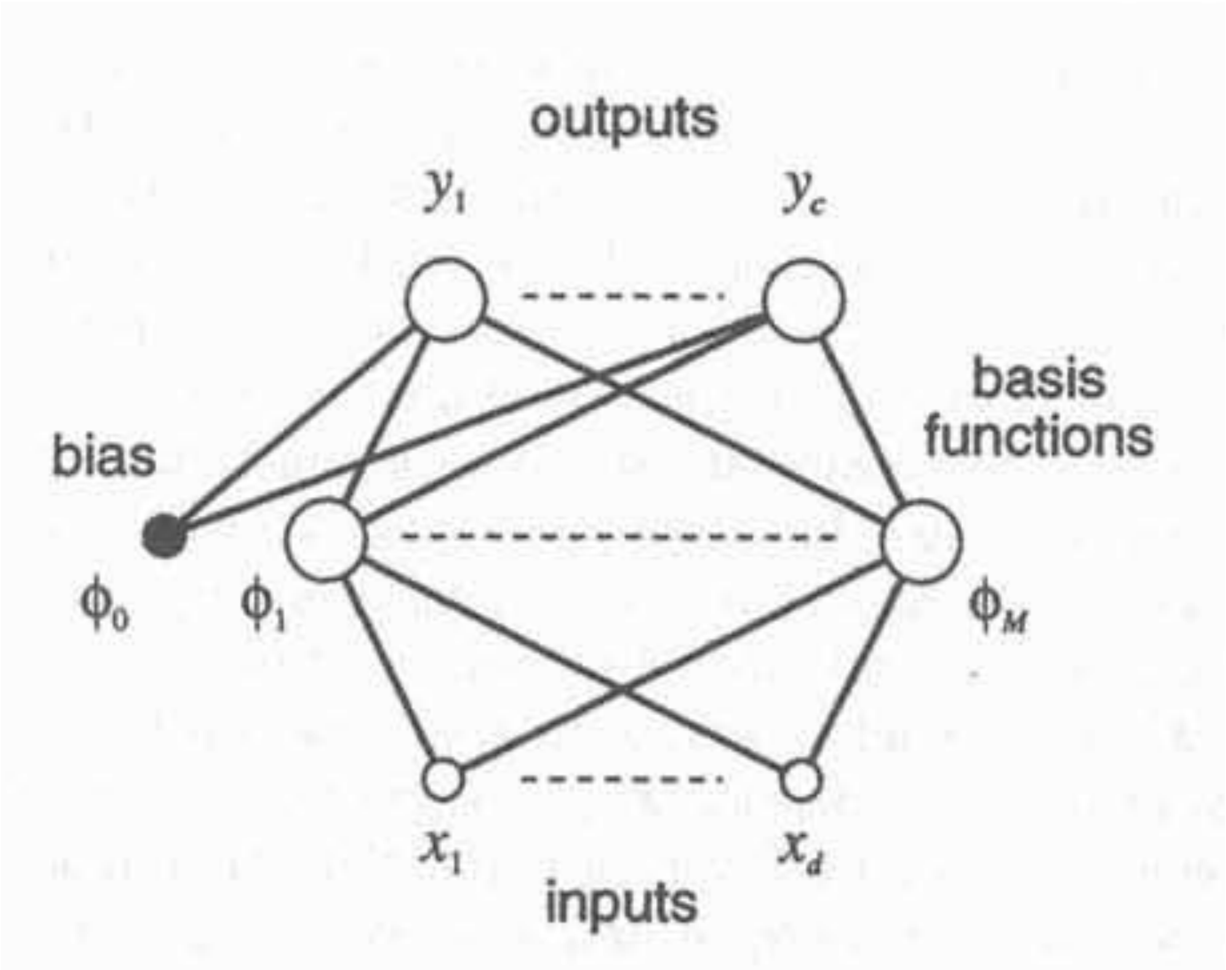
Introduction

- Regression
  - regression models based on linear combinations of fixed basis functions

- Radial Basis Functions find *f* such that

$$f(\mathbf{x}_n) = t_n \qquad f(\mathbf{x}) = \sum_{n=1}^{N} w_n h(\|\mathbf{x} - \mathbf{x}_n\|)$$

RBF NN



Architecture of RBF NN

RBF NN

- Outputs

$$y_k = \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}) + w_{k0}$$

- For the case of Gaussian functions

$$\phi_j(\mathbf{x}) = exp\left(-\frac{\left\|x - \mu_j\right\|^2}{2\sigma_j^2}\right)$$

RBF NN

- Result
  - Best approximation - in the set of approximating functions there is one function which has minimum approximating errore for any given function to be approsimated (Girosi and Poggio, 1990)
  - This property is not shared by MLP

RBF NN training

- Basis functions
  - Unsupervised learning (e.g., clustering)

- Network mapping

$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x})$$

- Matrix notation

$$y_k(\mathbf{x}) = \mathbf{W\Phi}$$

RBF NN training

- Sum-of-squares error

$$E = \frac{1}{2}\sum_{n}\sum_{k}\{y_k(\mathbf{x}^n) - t_k^n\}^2$$

- Differentiating with respect w_{kj} and setting the derivative to zero

$$\sum_{n}\{y_k(\mathbf{x}^n) - t_k^n\}\phi_j^n = 0$$

- Matrix notation

$$(\Phi^T\Phi)\mathbf{W}^T = \Phi^{\mathrm{T}}\mathbf{T}$$

RBF NN training

- Providing the matrix is non-singular we may invert

$$\mathbf{W}^{T} = \boldsymbol{\Phi}^{\dagger}\mathbf{T}$$

- Pseudo-inverse

$$\boldsymbol{\Phi}^{\dagger} = \left(\boldsymbol{\Phi}^{T}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{T}$$

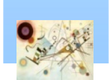
- Pseudoinverse property

$$\boldsymbol{\Phi}^{\dagger}\boldsymbol{\Phi} = \mathbf{I}$$

  - If the matrix $\boldsymbol{\Phi}^{T}\boldsymbol{\Phi}$ is not singular equation does not have a unique solution
  - In practical, use a SVD (Singular Value Decomposition) methodology

Properties and Depth

- Multi-Layer Neural Networks
  - all units are heaviside
    - Lippmann 1987; Lonstaff and Cross 1987
    - Single layer of weights
      - Hyperplane decision boundary
    - Two layer of weights
      - Decision boundary which sorround a signle convex region of the input space
    - Three layer of weights
      - Can generate arbitrary decision regisons, non-convex and disjoint



Properties and Depth

- Multi-Layer Neural Networks
  - all units are sigmoidal
    - Two layer of weights
      - Can approximate arbitrarly well any functional (one-one or many-one) continuos mapping from one finite-dimensional space to another, provided the number M of hidden units is sufficiently large
      - Universal Approximatin Theorem (Hornik et al., 1989; Cybenko, 1989) – Feedforward Net with linear output layer and at least one hidden layer with «squashing» activation function (e.g., logistic) can approximate any Borel measurable function from one-finite dimensional space to another provided the number M of hidden units is sufficiently large
      - Corollary (Classification) – Networks with sigmoidal non-linearities and two layers of weights can approximate any decision boundary to arbitrary accuracy (universal non-linear discriminant functions)
    - Three layer of weights
      - Can approximate, to arbitrary accuracy, any smooth mapping

Properties and Depth

- RBF NN
  - Best approximation - in the set of approximating functions there is one function which has minimum approximating error for any given function to be approsimated (Girosi and Poggio, 1990)
  - This property is not shared by MLP

Properties and Depth

- Feedforward NN layer may be infeasibly large and may fail to learn and generalize correctly (Barron, 1993)
- Functions representable with a deep rectifier net can require an exponential number of hidden units with a shallow (one hidden layer) network (Montufar et al., 2014)
  - Representation learning point
  - Better generalization (empirically)

Properties and Depth

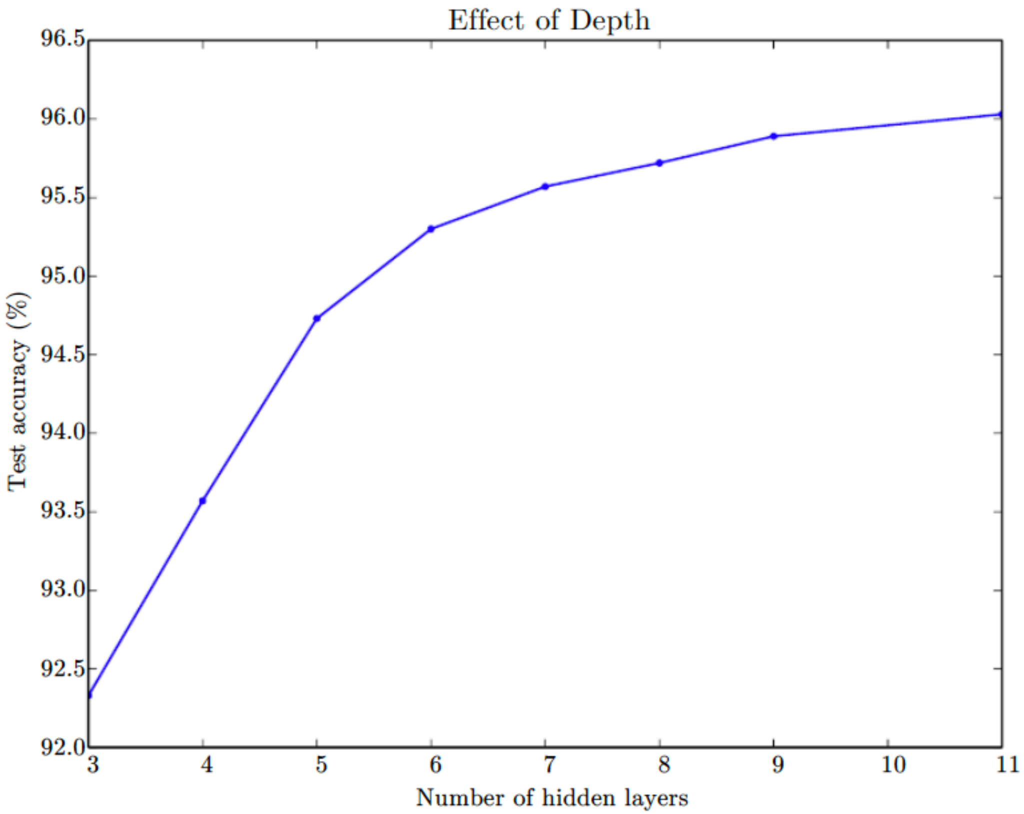


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See Fig. 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

Properties and Depth

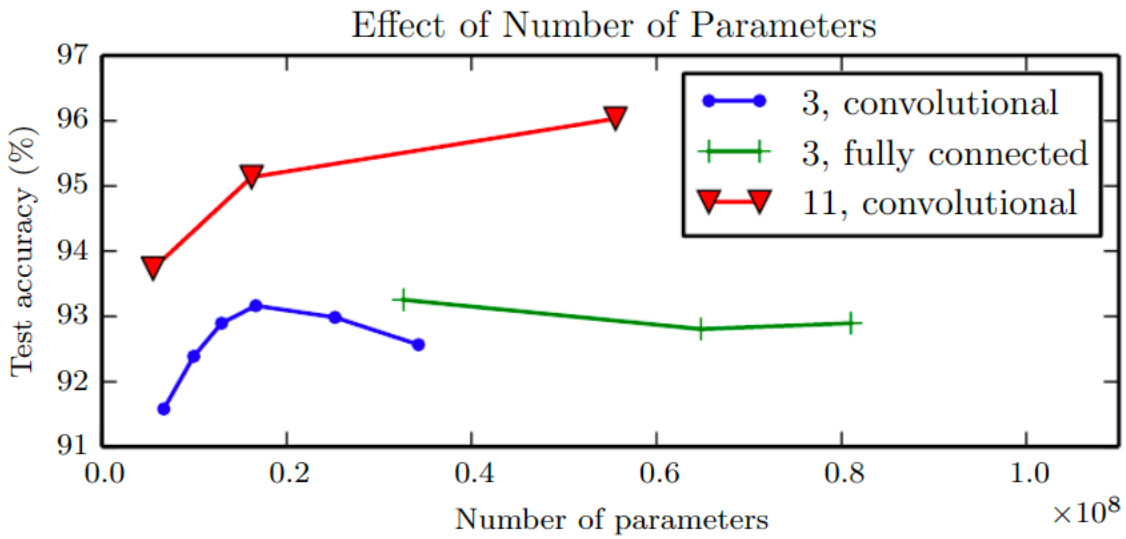


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from Goodfellow *et al.* (2014d) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance. The legend indicates the depth of network used to make each curve and whether the curve represents variation in the size of the convolutional or the fully connected layers. We observe that shallow models in this context overfit at around 20 million parameters while deep ones can benefit from having over 60 million. This suggests that using a deep model expresses a useful preference over the space of functions the model can learn. Specifically, it expresses a belief that the function should consist of many simpler functions composed together. This could result either in learning a representation that is composed in turn of simpler representations (e.g., corners defined in terms of edges) or in learning a program with sequentially dependent steps (e.g., first locate a set of objects, then segment them from each other, then recognize them).