Radial Basis Function Network for Multi-task Learning

Xuejun Liao and Lawrence Carin

{xjliao,lcarin}@ee.duke.edu

Department of Electrical and Computer Engineering

Duke University

Durham, NC 27708-0291, USA

Oveview

- Multi-task learning
- Multi-task radial basis function (RBF) networks
  - Network topology
  - Mathematical formulation
- Supervised learning of multi-task RBF networks
- Active learning of multi-task RBF networks
- Experimental results
- Summary

Multi-task Learning

- Solving multiple related tasks simultaneously under a unified representation
- Making each task easier to solve by transferring what is learned from one task to another correlated task
- Particularly useful when each individual task has scarce training data
- Multi-task learning becomes superfluous when
  - the tasks are identical - in this case, we simply pool them together
  - the tasks are independent - then, there is no correlation to utilize and we learn each task separately

Multi-task RBF Network

- Topology

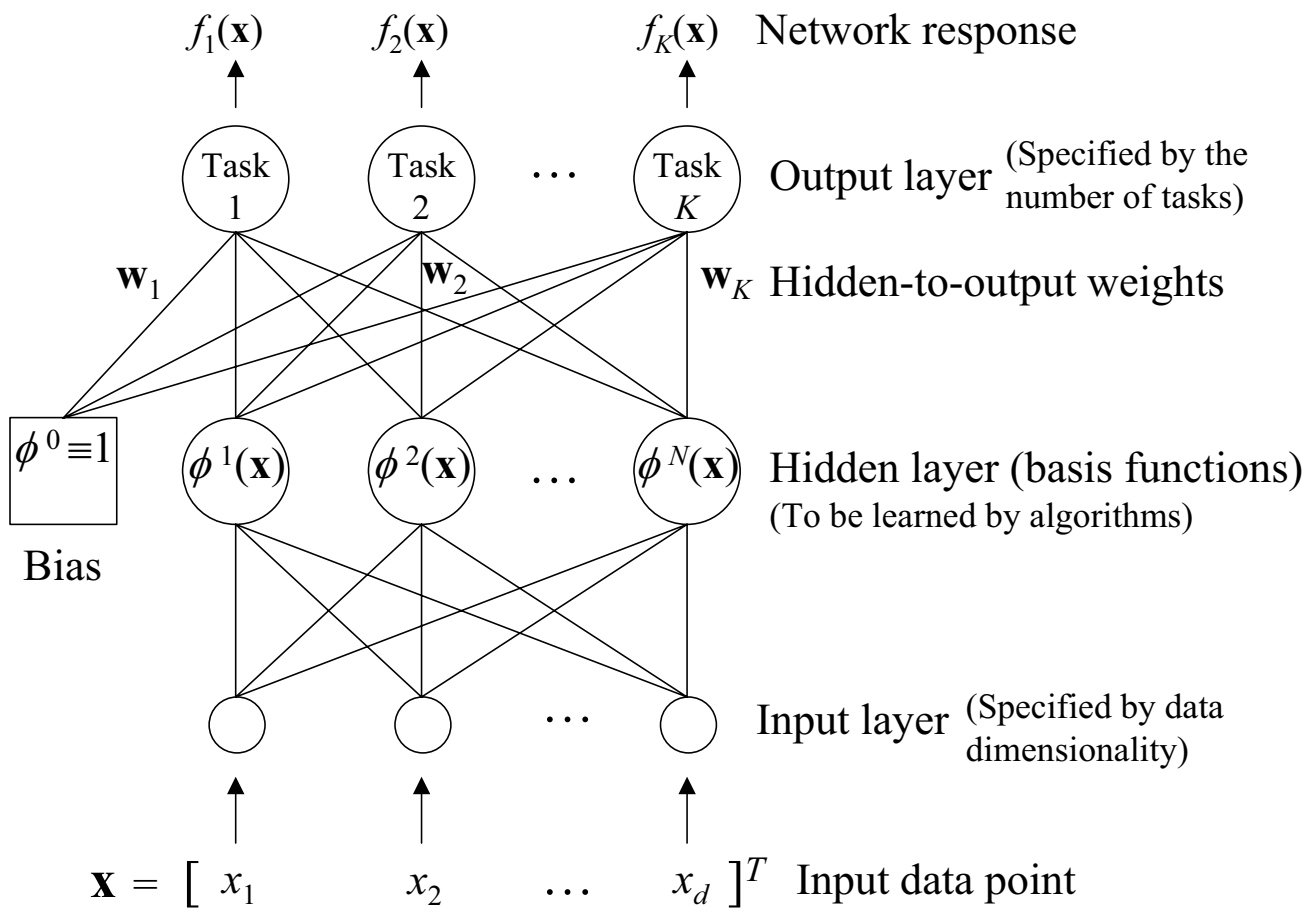

Figure 1: Each of the output nodes represents a unique task. Each task has its own hidden-to-output weights but all the tasks share the same hidden nodes. The activation of hidden node n is characterized by a basis function $\phi_n(\mathbf{x}) = \phi(||\mathbf{x} - \mathbf{c}_n||, \sigma_n)$. A typical choice of ϕ is $\phi(z, \sigma) = \exp(-\frac{z^2}{2\sigma^2})$, which gives the Gaussian RBF.

Multi-task RBF Network (cont'd)

- Mathematical formulation

Denote $\mathbf{w}_k = [w_{0k}, w_{1k}, \cdots, w_{Nk}]^T$ as the weights connecting hidden nodes to the k-th output node. The output for the k-th task, in response to input $\mathbf{x}$, takes the form

$$f_k(\mathbf{x}) = \mathbf{w}_k^T \boldsymbol{\phi}(\mathbf{x}) \tag{1}$$

where $\boldsymbol{\phi}(\mathbf{x}) = \left[\phi^0(\mathbf{x}), \phi^1(\mathbf{x}), \ldots, \phi^N(\mathbf{x})\right]^T$ is a column containing $N+1$ basis functions with $\phi^0(\mathbf{x}) \equiv 1$ a dummy basis accounting for the bias in Figure 1.

Supervised Learning

- Assuming K tasks, each associated with a data set $\mathcal{D}_k = \{(\mathbf{x}_{1k}, y_{1k}), \cdots, (\mathbf{x}_{J_k k}, y_{J_k k})\}$, where y_{ik} is the target (desired output) of $\mathbf{x}_{ik}$.
- We are interested in learning the functions $f_k(\mathbf{x})$ for the K tasks, based on minimizing the squared error

$$e(\boldsymbol{\phi}, \mathbf{w}) = \sum_{k=1}^{K} \left\{ \sum_{i=1}^{J_k} \left(\mathbf{w}_k^T \boldsymbol{\phi}_{ik} - y_{ik} \right)^2 + \rho \, ||\mathbf{w}_k||^2 \right\} \quad (2)$$

where $\boldsymbol{\phi}_{ik} = \boldsymbol{\phi}(\mathbf{x}_{ik})$ for notational simplicity. The regularization terms $\rho \, ||\mathbf{w}_k||^2$, $k = 1, \cdots, K$, are used to prevent singularity of the $\mathbf{A}$ matrices defined in (3), with ρ a small positive number.

Supervised Learning (cont'd)

- For fixed ϕ's, the $\mathbf{w}$'s are solved as

$$\mathbf{w}_k = \mathbf{A}_k^{-1} \sum_{i=1}^{J_k} y_{ik} \boldsymbol{\phi}_{ik} \quad \text{and} \quad \mathbf{A}_k = \sum_{i=1}^{J_k} \boldsymbol{\phi}_{ik} \boldsymbol{\phi}_{ik}^T + \rho \mathbf{I} \quad (3)$$

  for $k = 1, \cdots, K$

- The basis functions ϕ are obtained by minimizing

$$e(\boldsymbol{\phi}) = \sum_{k=1}^{K} \sum_{i=1}^{J_k} \left(y_{ik}^2 - y_{ik} \mathbf{w}_k^T \boldsymbol{\phi}_{ik} \right) \quad (4)$$

  which is obtained by substituting (3) into (2). Note $e(\boldsymbol{\phi})$ is a functional of ϕ only, as $\mathbf{w}_k$'s are related to ϕ by (3).

Supervised Learning (cont'd)

- Recalling that ϕ_{ik} is an abbreviation of $\boldsymbol{\phi}(\mathbf{x}_{ik}) = \left[1, \phi^1(\mathbf{x}_{ik}), \ldots, \phi^N(\mathbf{x}_{ik})\right]^T$, this amounts to determining N, the number of basis functions, and the functional form of each basis function $\phi^n(\cdot)$, $n = 1, \ldots, N$
- Consider the candidate functions

$$\{\phi^{nm}(\mathbf{x}) = \phi(||\mathbf{x}-\mathbf{x}_{nm}||, \sigma) \ : \ n = 1, \cdots, J_m, \ m = 1, \cdots, K\}$$

- We learn the RBF network structure by selecting $\boldsymbol{\phi}(\cdot)$ from these candidate functions such that $e(\boldsymbol{\phi})$ in (4) is minimized. The following theorem tells us how to perform the selection in a sequential way, the proof of which is given in the Appendix of the paper.

Supervised Learning (cont'd)

Theorem 1 *Let* $\boldsymbol{\phi}(\mathbf{x}) = [1, \phi^1(\mathbf{x}), \ldots, \phi^N(\mathbf{x})]^T$ *and* $\phi^{N+1}(\mathbf{x})$ *be a single basis function. Let the* $\mathbf{A}$ *matrices associated with* $\boldsymbol{\phi}$ *and* $[\boldsymbol{\phi}, \phi^{N+1}]^T$ *be non-degenerate. Then*

$$\begin{aligned} \delta e(\boldsymbol{\phi}, \phi^{N+1}) &= e(\boldsymbol{\phi}) - e([\boldsymbol{\phi}, \phi^{N+1}]^T) \\ &= \sum_{k=1}^{K} \left(\mathbf{c}_k^T \mathbf{w}_k - \sum_{i=1}^{J_k} y_{ik} \phi_{ik}^{N+1} \right)^2 q_k^{-1} \geq 0 \end{aligned} \tag{5}$$

where $\phi_{ik}^{N+1} = \phi^{N+1}(\boldsymbol{\phi}_{ik})$,

$$\left.\begin{aligned} \mathbf{c}_k &= \sum_{i=1}^{J_k} \boldsymbol{\phi}_{ik} \phi_{ik}^{N+1} \\ d_k &= \sum_{i=1}^{J_k} (\phi_{ik}^{N+1})^2 + \rho \\ q_k &= d_k - \mathbf{c}_k^T \mathbf{A}_k^{-1} \mathbf{c}_k \end{aligned}\right\} \tag{6}$$

and the $\mathbf{w}_k$*'s and* $\mathbf{A}_k$*'s are as given in (3).*

Supervised Learning (cont'd)

- Because $\delta e(\boldsymbol{\phi}, \phi^{N+1}) \geq 0$, adding ϕ^{N+1} to $\boldsymbol{\phi}$ generally makes the squared error decrease.
- The decrease $\delta e(\boldsymbol{\phi}, \phi^{N+1})$ depends on ϕ^{N+1}. By sequentially selecting basis functions that bring the maximum error reduction, we achieve the goal of maximizing $e(\boldsymbol{\phi})$. The details of the learning algorithm are summarized in Table 1.

Supervised Learning (cont'd)

Table 1: Learning Algorithm of Multi-Task RBF Network

Input: $\{(\mathbf{x}_{1k}, y_{2k}), \cdots, (\mathbf{x}_{J_k,k}, y_{J_k,k})\}_{k=1:K}$, $\phi(\cdot, \sigma)$, σ, and ρ; Output: $\boldsymbol{\phi}(\cdot)$ and $\{\mathbf{w}_k\}_{k=1}^K$.

1. **For** $m = 1:K$, **For** $n = 1:J_m$, **For** $k = 1:K$, **For** $i = 1:J_k$
   Compute $\widehat{\phi}_{ik}^{nm} = \phi(||\mathbf{x}_{nm} - \mathbf{x}_{ik}||, \sigma)$;
2. **Let** $N = 0$, $\boldsymbol{\phi}(\cdot) = 1$, $e_0 = \sum_{k=1}^K \left[\sum_{i=1}^{J_k} y_{ik}^2 - (J_k + \rho)^{-1}(\sum_{i=1}^{J_k} y_{ik})^2\right]$;
   For $k = 1:K$, compute $\mathbf{A}_k = J_k + \rho$, $\mathbf{w}_k = (J_k + \rho)^{-1}\sum_{i=1}^{J_k} y_{ik}$;
3. **For** $m = 1:K$, **For** $n = 1:J_m$
   If $\widehat{\phi}^{nm}$ is not marked as "deleted"
   For $k = 1:K$, compute
   $\mathbf{c}_k = \sum_{i=1}^{J_k} \boldsymbol{\phi}_{ik}\widehat{\phi}_{ik}^{nm}$, $q_k = \sum_{i=1}^{J_k}(\widehat{\phi}_{ik}^{nm})^2 + \rho - \mathbf{c}_k^T \mathbf{A}_k^{-1}\mathbf{c}_k$;
   If there exists k such that $q_k = 0$, mark $\widehat{\phi}^{nm}$ as "deleted";
   else, compute $\delta e(\boldsymbol{\phi}, \widehat{\phi}^{nm})$ using (5).
4. **If** $\{\widehat{\phi}^{ik}\}_{i=1:J_k, k=1:K}$ are all marked as "deleted", go to 10.
5. **Let** $(n^*, m^*) = \arg\max_{\widehat{\phi}^{nm} \text{ not marked as "deleted"}} \delta e(\boldsymbol{\phi}, \widehat{\phi}^{nm})$; Mark $\widehat{\phi}^{n^*m^*}$ as "deleted".
6. Tune RBF parameter $\sigma_{N+1} = \arg\max_\sigma \delta e(\boldsymbol{\phi}, \phi(||\cdot - \mathbf{x}_{n^*m^*}||, \sigma))$
7. **Let** $\phi^{N+1}(\cdot) = \phi(||\cdot - \mathbf{x}_{n^*m^*}||, \sigma_{N+1})$; Update $\boldsymbol{\phi}(\cdot) \leftarrow [\boldsymbol{\phi}^T(\cdot), \phi^{N+1}(\cdot)]^T$;
8. **For** $k = 1:K$
   Compute $\mathbf{A}_k^{new}$ and $\mathbf{w}_k^{new}$ respectively by (A-1) and (A-3) in the appendix of the paper; Update $\mathbf{A}_k \leftarrow \mathbf{A}_k^{new}$, $\mathbf{w}_k \leftarrow \mathbf{w}_k^{new}$
9. **Let** $e_{N+1} = e_N - \delta e(\boldsymbol{\phi}, \phi^{N+1})$; **If** the sequence $\{e_n\}_{n=0:(N+1)}$ is converged, go to 10, **else** update $N \leftarrow N + 1$ and go back to 3.
10. **Exit** and output $\boldsymbol{\phi}(\cdot)$ and $\{\mathbf{w}_k\}_{k=1}^K$.

Active Learning

- Assume the data in D_k are initially unsupervised (only $\mathbf{x}$'s are available without access to the associated y's)
- We want to actively select a subset from $\mathcal{D}_k$ to be supervised (y's acquired) such that the resulting network generalizes well to the remaining data in $\mathcal{D}_k$.
- We first learn the basis functions ϕ from the unsupervised data
- We then select data to be supervised, based on ϕ.
- Both of these steps are based on the following theorem, the proof of which is given in the Appendix of the paper.

Active Learning (cont'd)

Theorem 2 *Let there be K tasks and the data set of the k-th task is $\mathcal{D}_k \cup \widetilde{\mathcal{D}}_k$ where $\mathcal{D}_k = \{(\mathbf{x}_{ik}, y_{ik})\}_{i=1}^{J_k}$ and $\widetilde{\mathcal{D}}_k = \{(\mathbf{x}_{ik}, y_{ik})\}_{i=J_k+1}^{J_k+\widetilde{J}_k}$. Let there be two multi-task RBF networks, whose output nodes are characterized by $f_k(\cdot)$ and $f_k^{\sim}(\cdot)$, respectively, for task $k = 1, \ldots, K$. The two networks have the same given basis functions (hidden nodes) $\boldsymbol{\phi}(\cdot) = [1, \phi^1(\cdot), \cdots, \phi^N(\cdot)]^T$, but different hidden-to-output weights. The weights of $f_k(\cdot)$ are trained with $\mathcal{D}_k \cup \widetilde{\mathcal{D}}_k$, while the weights of $f_k^{\sim}(\cdot)$ are trained using $\widetilde{\mathcal{D}}_k$. Then for $k = 1, \cdots, K$, the square errors committed on $\mathcal{D}_k$ by $f_k(\cdot)$ and $f_k^{\sim}(\cdot)$ are related by*

$$0 \leq [\det \boldsymbol{\Gamma}_k]^{-1} \leq \lambda_{max,k}^{-1} \leq \frac{\sum_{i=1}^{J_k} (y_{ik} - f_k(\mathbf{x}_{ik}))^2}{\sum_{i=1}^{J_k} (y_{ik} - f_k^{\sim}(\mathbf{x}_{ik}))^2} \leq \lambda_{min,k}^{-1} \leq 1 \qquad (7)$$

where $\boldsymbol{\Gamma}_k = \left[\mathbf{I} + \boldsymbol{\Phi}_k^T (\rho\, \mathbf{I} + \widetilde{\boldsymbol{\Phi}}_k \widetilde{\boldsymbol{\Phi}}_k^T)^{-1} \boldsymbol{\Phi}_k\right]^2$ with $\boldsymbol{\Phi} = \left[\boldsymbol{\phi}(\mathbf{x}_{1k}), \ldots, \boldsymbol{\phi}(\mathbf{x}_{J_k k})\right]$ and $\widetilde{\boldsymbol{\Phi}} = \left[\boldsymbol{\phi}(\mathbf{x}_{J_k+1,k}), \ldots, \boldsymbol{\phi}(\mathbf{x}_{J_k+\widetilde{J}_k,k})\right]$, and $\lambda_{max,k}$ and $\lambda_{min,k}$ are respectively the largest and smallest eigenvalues of $\boldsymbol{\Gamma}_k$.

Active Learning (cont'd)

- Specializing Theorem 2 to the case $\widetilde{J}_k = 0$, we have

Corollary 1 *Let there be K tasks and the data set of the k-th task is $\mathcal{D}_k = \{(\mathbf{x}_{ik}, y_{ik})\}_{i=1}^{J_k}$. Let the RBF network, whose output nodes are characterized by $f_k(\cdot)$ for task $k = 1, \ldots, K$, have given basis functions (hidden nodes) $\boldsymbol{\phi}(\cdot) = [1, \phi^1(\cdot), \cdots, \phi^N(\cdot)]^T$ and the hidden-to-output weights of task k be trained with $\mathcal{D}_k$. Then for $k = 1, \cdots, K$, the squared error committed on $\mathcal{D}_k$ by $f_k(\cdot)$ is bounded as*

$$0 \leq [\mathit{det}\, \boldsymbol{\Gamma}_k]^{-1} \leq \lambda_{max,k}^{-1} \leq \frac{\sum_{i=1}^{J_k} \left(y_{ik} - f_k(\mathbf{x}_{ik})\right)^2}{\sum_{i=1}^{J_k} y_{ik}^2} \leq \lambda_{min,k}^{-1} \leq 1$$

where $\boldsymbol{\Gamma}_k = \left(\mathbf{I} + \rho^{-1} \boldsymbol{\Phi}_k^T \boldsymbol{\Phi}_k\right)^2$ with $\boldsymbol{\Phi} = \left[\boldsymbol{\phi}(\mathbf{x}_{1,k}), \ldots, \boldsymbol{\phi}(\mathbf{x}_{J_k,k})\right]$, and $\lambda_{max,k}$ and $\lambda_{min,k}$ are respectively the largest and smallest eigenvalues of $\boldsymbol{\Gamma}_k$

Active Learning (cont'd)

- We are interested in selecting the basis functions ϕ that minimize the squared errors for all tasks, before seeing y's.
- By Corollary 1, this is accomplished by maximizing the eigenvalues of $\mathbf{\Gamma}_k$ for $1, \cdots, K$
- Expanding $\det \mathbf{\Gamma}_k$ as

$$\begin{aligned} \det \mathbf{\Gamma}_k &= \det\left[\left(\mathbf{I}_{J_k \times J_k} + \rho^{-1} \mathbf{\Phi}_k^T \mathbf{\Phi}_k\right)^2\right] \\ &= \left[\det(\rho \mathbf{I}_{(N+1)\times(N+1)} + \mathbf{\Phi}_k \mathbf{\Phi}_k^T)\right]^2 [\det(\rho\, \mathbf{I}_{(N+1)\times(N+1)})]^{-2} \\ &= [\det \mathbf{A}_k^2][\det(\rho\, \mathbf{I}_{(N+1)\times(N+1)})]^{-2} \end{aligned} \quad (8)$$

- Then the problem boils down to selecting ϕ to maximize $\det \mathbf{A}_k$ for all tasks simultaneously, i.e., selecting ϕ to maximize $\prod_{k=1}^{K} \det \mathbf{A}_k$.

Active Learning (cont'd)

- The selection proceeds in a sequential manner:
  - Suppose we have selected basis functions $\boldsymbol{\phi} = [1, \phi^1, \cdots, \phi^N]^T$, with $\mathbf{A}_k = \sum_{i=1}^{J_k} \boldsymbol{\phi}_{ik}\boldsymbol{\phi}_{ik}^T + \rho\,\mathbf{I}_{(N+1)\times(N+1)}$, $k = 1, \cdots, K$.
  - Augmenting basis functions to $[\boldsymbol{\phi}^T, \phi^{N+1}]^T$, the $\mathbf{A}$ matrices change to $\mathbf{A}_k^{new} = \sum_{i=1}^{J_k} [\boldsymbol{\phi}_{ik}^T, \phi_{ik}^{N+1}]^T [\boldsymbol{\phi}_{ik}^T, \phi_{ik}^{N+1}] + \rho\,\mathbf{I}_{(N+2)\times(N+2)}$.
  - Using the determinant formula of block matrices, we get $\prod_{k=1}^K (\det \mathbf{A}_k^{new})^{-2} = \prod_{k=1}^K (q_k \det \mathbf{A}_k)^{-2}$, where q_k is as given in (6). As $\mathbf{A}_k$'s do not depend on ϕ^{N+1}, the left-hand side is minimized by maximizing $\prod_{k=1}^K q_k^2$.
  - The selection is easily implemented by making the following two minor modifications in Table 1: (a) in step 2, compute $e_0 = \sum_{k=1}^K \ln(J_k + \rho)^{-2}$; in step 3, compute $\delta e(\boldsymbol{\phi}, \widehat{\phi}^{nm}) = \sum_{k=1}^K \ln q_k^2$. Employing the logarithm is for gaining additivity and it does not affect the maximization.

Active Learning (cont'd)

- Based on the basis functions ϕ determined above, we proceed to selecting data to be supervised.
- The selection of data is based on an iterative use of Corollary 2 as given in slide 18, which is a specialization of Theorem 2, as is Corollary 1.

Active Learning (cont'd)

Corollary 2 *Let there be K tasks and the data set of the k-th task is $\mathcal{D}_k = \{(\mathbf{x}_{ik}, y_{ik})\}_{i=1}^{J_k}$. Let there be two RBF networks, whose output nodes are characterized by $f_k(\cdot)$ and $f_k^+(\cdot)$, respectively, for task $k = 1, \ldots, K$. The two networks have the same given basis functions $\boldsymbol{\phi}(\cdot) = [1, \phi^1(\cdot), \cdots, \phi^N(\cdot)]^T$, but different hidden-to-output weights. The weights of $f_k(\cdot)$ are trained with $\mathcal{D}_k$, while the weights of $f_k^+(\cdot)$ are trained using $\mathcal{D}_k^+ = \mathcal{D}_k \cup \{(\mathbf{x}_{J_k+1,k}, y_{J_k+1,k})\}$. Then for $k = 1, \cdots, K$, the squared errors committed on $(\mathbf{x}_{J_k+1,k}, y_{J_k+1,k})$ by $f_k(\cdot)$ and $f_k^+(\cdot)$ are related by $\left[f_k^+(\mathbf{x}_{J_k+1,k}) - y_{J_k+1,k}\right]^2 = \left[\gamma(\mathbf{x}_{J_k+1,k})\right]^{-1}\left[f_k(\mathbf{x}_{J_k+1,k}) - y_{J_k+1,k}\right]^2$, where $\gamma(\mathbf{x}_{J_k+1,k}) = \left[1 + \boldsymbol{\phi}^T(\mathbf{x}_{J_k+1,k})\mathbf{A}_k^{-1}\boldsymbol{\phi}(\mathbf{x}_{J_k+1,k})\right]^2 \geq 1$ and $\mathbf{A}_k = \sum_{i=1}^{J_k}\left[\rho\mathbf{I} + \boldsymbol{\phi}(\mathbf{x}_{ik})\boldsymbol{\phi}^T(\mathbf{x}_{ik})\right]$ is the same as in (3).*

Active Learning (cont'd)

- Two observations are made from Corollary 2:
  - If $\gamma(\mathbf{x}_{J_k+1,k}) \approx 1$, seeing $y_{J_k+1,k}$ does not effect the error on $\mathbf{x}_{J_k+1,k}$, indicating $\mathcal{D}_k$ already contain sufficient information about $(\mathbf{x}_{J_k+1,k}, y_{J_k+1,k})$.
  - If $\gamma(\mathbf{x}_i) \gg 1$, seeing $y_{J_k+1,k}$ greatly decrease the error on $\mathbf{x}_{J_k+1,k}$, indicating $\mathbf{x}_{J_k+1,k}$ is significantly dissimilar (novel) to $\mathcal{D}_k$ and $\mathbf{x}_{J_k+1,k}$ must be supervised to reduce the error.

Active Learning (cont'd)

- Data Selection proceeds in a sequential fashion:
  - Given $\mathcal{D}_k = \{(\mathbf{x}_{ik}, y_{ik})\}_{i=1}^{J_k}$ have been selected, the data point selected next should be

$$\mathbf{x}_{J_k+1,k} = \arg \max_{\substack{i>J_k \\ k=1,\cdots,K}} \gamma(\mathbf{x}_{ik}) = \arg \max_{\substack{i>J_k \\ k=1,\cdots,K}} \left[1+\boldsymbol{\phi}^T(\mathbf{x}_{ik})\mathbf{A}_k^{-1}\boldsymbol{\phi}(\mathbf{x}_{ik})\right]^2$$

  - After $\mathbf{x}_{J_k+1,k}$ is selected, the $\mathbf{A}_k$ is updated as $\mathbf{A}_k \leftarrow \mathbf{A}_k + \boldsymbol{\phi}(\mathbf{x}_{J_k+1,k})\boldsymbol{\phi}^T(\mathbf{x}_{J_k+1,k})$, and the next selection begins.
  - As the iteration advances γ will decrease until it reaches convergence, upon which we stop selecting data.
- Finally we use (3) to compute $\mathbf{w}$ from the selected $\mathbf{x}$ and their associated targets y, completing learning of the multi-task RBF network.

Experimental Results

- We consider three types of RBF networks to learn K tasks, each with its data set $\mathcal{D}_k$:
  - In the first, which we call “one RBF network”, we let the K tasks share both basis functions ϕ (hidden nodes) and hidden-to output weights $\mathbf{w}$, thus we do not distinguish the K tasks and design a single RBF network to learn a union of them.
  - The second is the multi-task RBF network, where the K tasks share the same ϕ but each has its own $\mathbf{w}$.
  - In the third, we have K independent networks, each designed for a single task.

Experimental Results (cont'd)

- We use a school data set from the Inner London Education Authority, consisting of examination records of 15362 students from 139 secondary schools. The data are available at

  http://multilevel.ioe.ac.uk/intro/datasets.html

- The goal is to predict the exam scores of the students.
- Treating each school as a task, we perform multi-task learning.
- After converting each categorical variable to a number of binary variables, we obtain a total number of 27 input variables, i.e., $\mathbf{x} \in \mathbb{R}^{27}$.

Experimental Results (cont'd)

- Some details of the implementation:
  - The multi-task RBF network is implemented as shown in Figure 1 and trained with the learning algorithm in Table 1.
  - The "one RBF network" is implemented as a special case of Figure 1, with a single output node and trained using the union of supervised data from all 139 schools.
  - 139 independent RBF networks are designed, each having a single output node and trained for a single school.
  - The Gaussian RBF $\phi^n(\mathbf{x}) = \exp(-\frac{||\mathbf{x}-\mathbf{c}_n||^2}{2\sigma^2})$ is used, where the $\mathbf{c}_n$'s are selected from training data points and σ_n's are initialized as 20 and optimized as described in Table 1. The regularization parameter ρ is set to 10^{-6}.

Experimental Results (cont'd)

- Training and testing
  - We randomly take 75% of the 15362 data points as training (supervised) data and the remaining 25% as test data.
  - The generalization performance is measured by the squared error $(f_k(\mathbf{x}_{ik}) - y_{ik})^2$ averaged over all test data $\mathbf{x}_{ik}$ of tasks $k = 1, \cdots, K$.
  - We made 10 independent trials to randomly split the data into training and test sets and the squared error averaged over the test data of all the 139 schools and the trials are shown in Table 2, for the three types of RBF networks.

Experimental Results (cont'd)

Table 2: Squared error averaged over the test data of all 139 schools and the 10 independent trials for randomly splitting the school data into training (75%) and testing (25%) sets.

Multi-task RBF network	Independent RBF networks	One RBF network
109.89 ± 1.8167	136.41 ± 7.0081	149.48 ± 2.8093

Experimental Results (cont'd)

- Table 2 shows the multi-task RBF network outperforms the other two types of RBF networks by a considerable margin. This is attributed to:
  - The "one RBF network" ignores the difference between the tasks and the independent RBF networks ignore the tasks' correlations, therefore they both perform inferiorly.
  - The multi-task RBF network uses the shared hidden nodes (basis functions) to capture the common internal representation of the tasks and meanwhile uses the independent hidden-to-output weights to learn the statistics specific to each task.

Experimental Results (cont'd)

- We apply active learning to split the data into training and test sets using the two-step procedure:
  - First we learn the basis functions ϕ of multi-task RBF network using all 15362 data, which are unsupervised, i.e., their targets y are not required.
  - Based on the ϕ, we then select the data $\mathbf{x}$ to be supervised, collect the y associated with them, and use them as training data to learn the hidden-to-output weights $\mathbf{w}$.
- To make the results comparable, we use the same training data to learn the other two types of RBF networks (including learning their own ϕ and $\mathbf{w}$).
- The networks are then tested on the remaining data, with the results shown in Figure 2.

Experimental Results (cont'd)

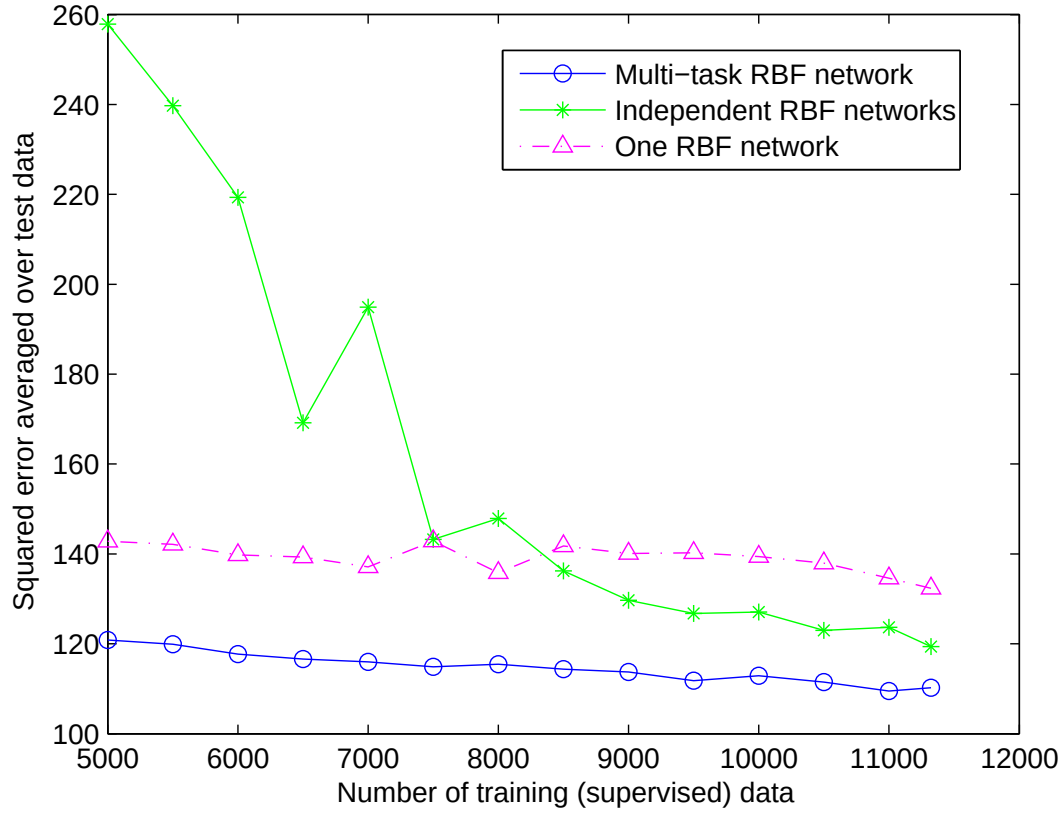

Figure 2: Squared error averaged over the test data of all 139 schools, as a function of the number of training (supervised) data. The data are split into training and test sets via active learning.

Experimental Results (cont'd)

- Discussions on Figure 2:
  - Clearly the multi-task RBF network maintains its superior performance all the way down to 5000 training data points, whereas the independent RBF networks have their performances degraded seriously as the training data diminish.
  - This demonstrates the increasing advantage of multi-task learning as the number of training data decreases.
  - The "one RBF network" seems also insensitive to the number of training data, but it ignores the inherent dissimilarity between the tasks, which makes its performance inferior.

Summary

- We have presented the structure and learning algorithms for multi-task learning with radial basis function (RBF) networks.
- Supervised learning of the network structure is achieved via simple and efficient evaluation of candidate basis functions
- Unsupervised learning of the network structure enables us to actively split the data $\mathbf{x}$ into training and test sets, without accessing the y's.
- The proposed unsupervised criteria select novel data into the training set, hence what remain in the test set are similar to those in the training set. This improves the generalization of the resulting network to the test data.