LECTURE 19: Radial Basis Functions

- Introduction to RBFs
- Input-output mapping
- Hybrid training procedures
- Gram-Schmidt orthogonalization
- Orthogonal Least Squares

Introduction

- **The previous two lectures have focused on projective neural networks**
  - In perceptron-type networks, the activation of hidden units is based on the **dot product** between the input vector and a weight vector
  - In this lecture we will look at RBFs, networks where the activation of hidden units is based on the **distance** between the input vector and a prototype vector
- **Radial Basis Functions have a number of interesting properties**
  - There exists strong connections to a number of scientific disciplines
    - These include function approximation, regularization theory, density estimation and interpolation in the presence of noise [Bishop, 1995]
  - RBFs allow for a straightforward interpretation of the internal representation produced by the hidden layer
  - RBFs have training algorithms that are significantly faster than those for MLPs
    - And, as we will see today, most of these algorithms have already been presented in previous lectures!

Exact function interpolation

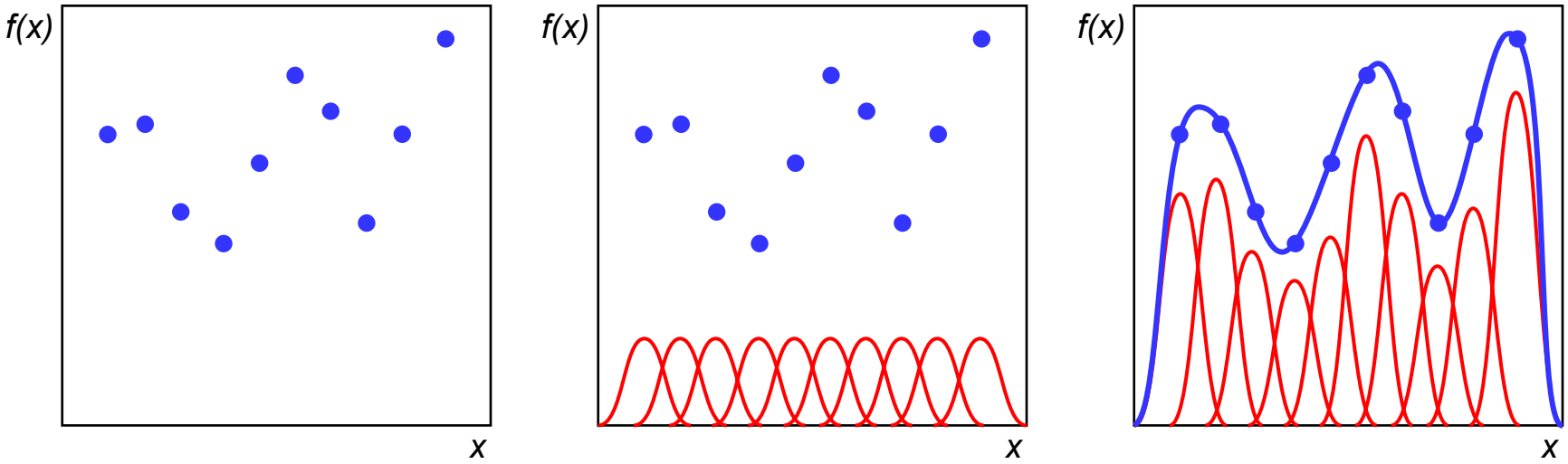
- **RBFs have their origins in techniques for performing exact function interpolation [Bishop, 1995]**
  - These techniques place a basis function at each of the training examples

$$h(x) = \sum_{k=1}^{N} w_k \varphi\left(\left\| x - x^{(n)} \right\|\right) = \Phi w$$

    - and compute the coefficients w_k so that the "mixture model" has zero error at those examples

$$h\left(x^{(i)}\right) = \sum_{k=1}^{N} w_k \varphi\left(\left\| x - x^{(n)} \right\|\right) = t^{(i} \Leftrightarrow w = \Phi^{-1} t$$

  - Can you draw any connections to other techniques we have already discussed?



Radial Basis Functions

- **Radial Basis Functions are feed-forward networks consisting of**
  - A hidden layer of radial kernels and
  - An output layer of linear neurons
- **The two layers in an RBF carry entirely different roles [Haykin, 1999]**
  - The hidden layer performs a non-linear transformation of input space
    - The resulting hidden space is typically of higher dimensionality than the input space
  - The output layer performs linear regression to predict the desired targets
- **Why use a non-linear transformation followed by a linear one?**
  - Cover's theorem on the separability of patterns
    - *"A complex pattern-classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space"*
  - As we will see in a few lectures, this very same argument is at the core of Support Vector Machines. RBFs are indeed one of the kernel functions most commonly used in SVMs!

Input-to-hidden mapping

- **Each hidden neuron in an RBF is tuned to respond to a rather local region of feature space by means of a radially symmetric function**
  - Activation of a hidden unit is determined by the DISTANCE between the input vector *x* and a prototype vector μ

$$\varphi_j(x) = f\left(\left\|x - \mu_j\right\|\right)$$

- **Choice of radial basis**
  - Although several forms of radial basis may be used, Gaussian kernels are most commonly used
    - The Gaussian kernel may have a full-covariance structure, which requires D(D+3)/2 parameters to be learned

$$\varphi_j(x) = \exp\left[-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)\right]$$

    - or a diagonal structure, with only (D+1) independent parameters

$$\varphi_j(x) = \exp\left[-\frac{\|x-\mu_j\|^2}{2\sigma_j^2}\right]$$

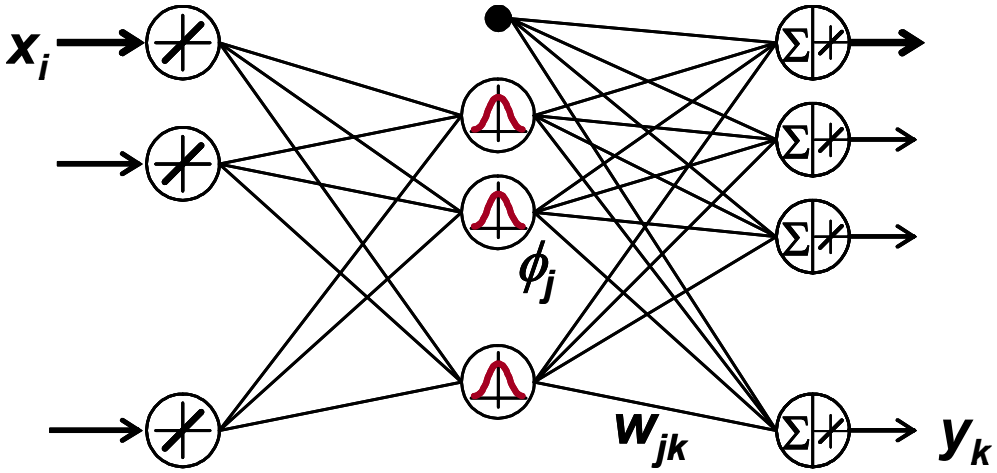
    - In practice, a trade-off exists between using a small number of basis with many parameters or a larger number of less flexible functions [Bishop, 1995]

Hidden-to-output mapping

- **Output units form linear combinations of the hidden-unit activations to predict the output variable(s)**
  - The activation of an output unit is determined by the DOT-PRODUCT between the hidden activation vector ϕ and the weight vector w

$$y_k = \sum_{j=1}^{N_H} w_{jk} \varphi_j \left(\left| x - \mu_j \right| \right) + w_{0k}$$

    - For convenience, an additional basis function ϕ_0 with a constant activation of 1 can be used to absorb the bias term w_{0k}



Hybrid training

- **RBFs are commonly trained following a hybrid procedure that operates in two stages or time scales [Haykin, 1999]**
  - **Unsupervised selection of RBF centers**
    - RBF centers are selected so as to match the distribution of training examples in the input feature space
    - This is the critical step in training, normally performed in a slow iterative manner
    - Fortunately, a number of strategies presented in previous lectures can be used to solve this problem
  - **Supervised computation of output vectors**
    - Hidden-to-output weight vectors are determined so as to minimize the sum-squared error between the RBF outputs and the desired targets
    - Since the outputs are linear, the optimal weights can be computed using fast, linear matrix inversion

Unsupervised selection of RBF centers

- **Random selection of centers**
  - The simplest approach is to randomly select a number of training examples as RBF centers
    - This method has the advantage of being very fast, but the network will likely require an excessive number of centers
  - Once the center positions have been selected, the spread parameters σ_j can be estimated, for instance, from the average distance between neighboring centers
- **Clustering**
  - Alternatively, RBF centers may be obtained with a clustering procedure such as the k-means algorithm (Lecture 15)
  - The spread parameters can be computed as before, or from the sample covariance of the examples of each cluster
- **Density estimation**
  - The position of the RB centers may also be obtained by modeling the feature space density with a Gaussian Mixture Model using the Expectation-Maximization algorithm (Lecture 14)
  - The spread parameters for each center are automatically obtained from the covariance matrices of the corresponding Gaussian components

Supervised training of output weights

- **Once the RBF centers have been selected, hidden-to-output weights are computed so as to minimize the MSE error at the output**

$$W = \underset{W}{\operatorname{argmin}} \left\{ \sum_{n=1}^{N} \sum_{k=1}^{N_O} \left(t_k^{(n} - \sum_{j=1}^{N_H} w_{jk} \varphi_j \left(x^{(n} \right) \right) \right\} = \underset{W}{\operatorname{argmin}} \left\| T - \Phi W \right\|$$

  - Now, since the hidden activation patterns Φ are fixed, the optimum weight vector W can be obtained directly from the conventional pseudo-inverse solution (Lecture 17)

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

Drawbacks of unsupervised center selection

- **Hybrid RBF training procedures have one major disadvantage**
  - The selection of RBF centers is not guided by the MSE objective function
  - RBF centers that are representative of the feature space density are not guaranteed to capture the structure that carries discriminatory information
    - To some extent, this is a similar argument to that of signal-representation (PCA) versus signal-classification (LDA) in dimensionality reduction
- **To avoid this problem, fully-supervised algorithms can also be used for RBF training**
  - Orthogonal Least Squares (OLS) is the most widely used method, and will be covered next
  - Other approaches have also been proposed [Haykin, 1999]

Introduction to Orthogonal Least Squares

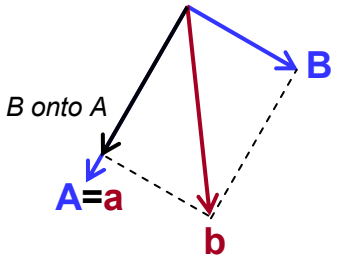
- **OLS is a forward stepwise regression procedure**
  - Starting from a large pool of candidate centers (e.g., training examples), OLS sequentially selects the center that results in the largest reduction of sum-square-error at the output
    - A simple implementation of this idea is to perform sequential forward selection directly on the radial-basis internal representation

> *1) Start with N candidate centers and M=0 centers*
> *2) For each of the k=N-M remaining candidates*
> *a) Add the k-th center to the existing M centers*
> *b) Compute the pseudo-inverse solution*
> *c) Compute the resulting SSE at the output*
> *3) Choose the k-th candidate that yields lowest SSE*
> *4) Set M=M+1*
> *5) Go to 2*

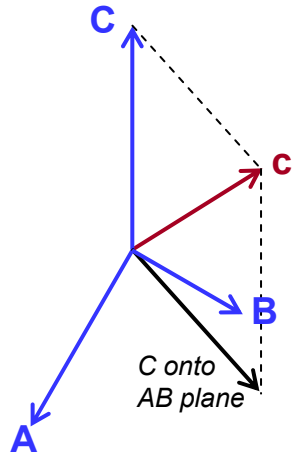
  - This implementation is, however, very inefficient since the pseudo-inverse $W=(\Phi^t\Phi)^{-1}\Phi^tT$ needs to be computed N-M times at step M in the selection process
    - Instead, OLS constructs a set of orthogonal vectors Q for the space spanned by the candidate centers
    - In this orthogonal subspace, computation of the pseudo-inverse is effectively avoided since Q^tQ becomes diagonal

Gram-Schmidt orthogonalization (1)

- **Assume that we have three (independent) vectors a, b and c, from which we wish to construct three orthogonal vectors A, B and C**
- **The Gram-Schmidt orthogonalization procedure operates as follows**
  - Start with A=a
    - This gives the first direction
  - The second direction must be perpendicular to A
    - Start with B=b and subtract its projection along A
    - This leaves the perpendicular part, which is the orthogonal vector B



  - The third direction must be perpendicular to A and B
    - Start with C=c and subtract its projections along A and B
  - The resulting vectors {A,B,C} are orthogonal and span the same space as {a,b,c}



From [Strang, 1998]

Gram-Schmidt orthogonalization (2)

- **For M basis vectors $\Phi=\{\varphi_1,\varphi_2,\ldots,\varphi_M\}$, Gram-Schmidt generalizes to**

$$\left.\begin{aligned} q_1 &= \varphi_1 \\ \alpha_{ik} &= \frac{q_i^T \varphi_k}{q_i^T q_i} \quad 1 \le i < k \\ q_k &= \varphi_k - \sum_{i=1}^{k-1} \alpha_{ik} q_i \end{aligned}\right\} k = 2,\ldots,M$$

Equation (eq1)

- where it can be shown [Strang, 1998; Chen et al., 1991] that the orthogonal set $Q=\{q_1,q_2,..,q_M\}$ is linearly related to the original set Φ by the following relationship

$$\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \varphi_1 & \varphi_2 & \cdots & \varphi_M \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ q_1 & q_2 & \cdots & q_M \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} 1 & \alpha_{12} & \cdots & \alpha_{1M} \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & \alpha_{M-1M} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad \text{or} \quad \Phi = QA$$

  - To prove this relationship notice that, at every step, φ_k is a combination of the previous orthogonal vectors $q_1,q_2,\ldots q_k$. Later q's are not involved

Geometric interpretation of the pseudo-inverse (1)

- **Assume a dataset of examples $X=\{(x^{(1},t^{(1}),(x^{(2},t^{(2}),\ldots,(x^{(N},t^{(N})\}$**
  - For convenience we will assume an RBF with a single output
    - Notice that the pseudo-inverse solution estimates each output independently anyways
  - The hidden-to-output regression can be expressed as

$$\begin{bmatrix} t^{(1} \\ t^{(2} \\ \vdots \\ t^{(N} \end{bmatrix} = \begin{bmatrix} \varphi_1^{(1} & \varphi_2^{(1} & \cdots & \varphi_M^{(1} \\ \varphi_1^{(2} & \varphi_2^{(2} & \cdots & \varphi_M^{(2} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1^{(N} & \varphi_2^{(N} & \cdots & \varphi_M^{(N} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} + \begin{bmatrix} \varepsilon^{(1} \\ \varepsilon^{(2} \\ \vdots \\ \varepsilon^{(N} \end{bmatrix} \Leftrightarrow T = \Phi W + E$$

    - where E is the vector of prediction errors, whose sum square we seek to minimize

  - Notice that the activation of a particular radial basis for all the training examples $\Phi_k = [\varphi_k^{(1},\varphi_k^{(2},\ldots\varphi_k^{(N}]^T$ can be treated as a vector

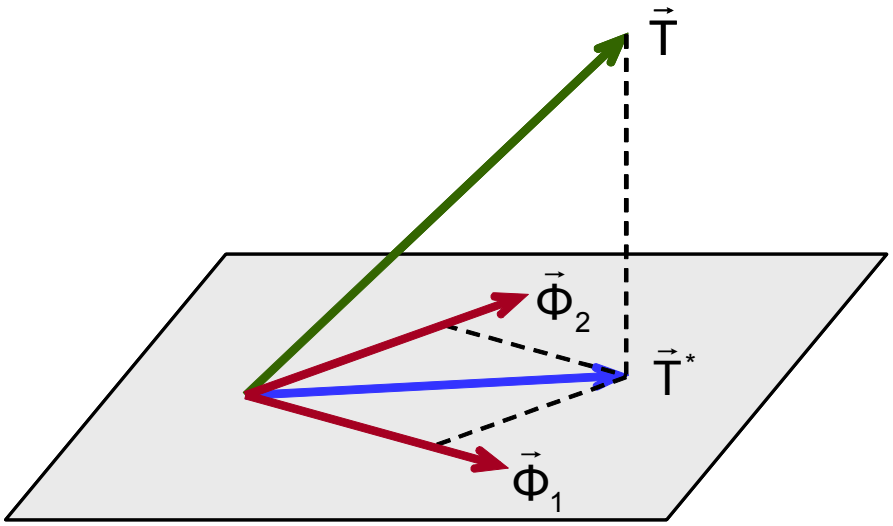
  - Notice that the desired target $T = [t^{(1},t^{(2},\ldots t^{(N}]^T$ can also be treated as a vector

Geometric interpretation of the pseudo-inverse (2)

- **The pseudo-inverse solution has a very nice geometric interpretation**
  - The linear system is attempting to express the target vector T as a linear combination of the M hidden vectors Φ_k

$$T = \sum_{k=1}^{M} w_k \Phi_k$$

  - In the case of an over-determined system, an exact solution cannot be found since the vector T lies outside of the space spanned by the vectors Φ_k
  - It can be shown [Bishop, 1995] that the least-squares (or pseudo-inverse) solution is the orthogonal projection of T onto that space



Orthogonal Least Squares (1)

- **Keeping in mind this geometric interpretation of the least-squares solution and the Gram-Schmidt orthogonalization procedure, we are now ready to present the final implementation of OLS**
  - As mentioned earlier, OLS constructs a set of orthogonal vectors Q for the space spanned by the basis vectors Φ_k such that Φ=QA (A upper triangular)
  - Using this orthogonal representation, the RBF solution is expressed as

$$T = \Phi W = QG$$

    - and the least-squares solution for the weight vector G in the orthogonal space is

$$G = \left(Q^t Q\right)^{-1} Q^t T$$

    - Now, since Q is orthogonal, Q^tQ is then diagonal, and each component of G can be extracted independently without ever having to compute a pseudo-inverse matrix

$$g_i = \frac{q_i^t T}{q_i^t q_i}$$ ***Equation (eq2)***

    - This is precisely what makes OLS a very efficient implementation of stepwise forward regression

Orthogonal Least Squares (2)

- **How are basis functions selected?**
  - The sum of squares or energy of the target vector T is

$$T^tT = \sum_{i=1}^{M} g_i^2 q_i^t q_i + E^tE$$

  - Assuming that the mean of the desired target T has been removed, then the variance of T is given by

$$N^{-1}T^tT = N^{-1}\sum_{i=1}^{M} g_i^2 q_i^t q_i + N^{-1}E^tE$$

    - The first term, $N^{-1}\Sigma_i(g_i^2 q_i^t q_i)$, is the part of the desired variance which can be explained by the regressors, whereas $N^{-1}E^tE$ is the unexplained variance
    - Therefore, $N^{-1}g_i^2 q_i^t q_i$ is the increment in the explained output variance achieved by adding the regressor q_i, which contributes to a reduction of the error (relative to the total T^tT) by

$$[\text{err}]_i = \frac{g_i^2 q_i^t q_i}{T^tT}$$ ***Equation (eq3)***

    - **This ratio provides a simple measure that allows OLS to select a subset of regressors in a stepwise forward manner**
    - The complete algorithm is included in the next page

Orthogonal Least Squares (3)

- **At the first step, for 1≤i≤M, compute**

$$\left.\begin{aligned} q_1^{(i)} &= \varphi_i \\ g_1^{(i)} &= \left(q_1^{(i)}\right)^t T \Big/ \left(\left(q_1^{(i)}\right)^t q_1^{(i)}\right) \\ [err]_1^{(i)} &= \left(g_1^{(i)}\right)^2 \left(q_1^{(i)}\right)^t q_1^{(i)} \Big/ \left(T^t T\right) \end{aligned}\right\} \begin{aligned} &\text{Orthogonalize vector (eq1)} \\ &\text{Compute least-squares solution (eq2)} \\ &\text{Compute reduction in error (eq3)} \end{aligned}$$

  - and select the regressor that yields the highest reduction in error $q_1 = \underset{q_1^{(i)}}{\text{argmax}}\left[[err]_1^{(i)}\right] = \varphi_{i_1}$

- **At the k-th step, for 1≤i≤M, and i not already selected**

$$\left.\begin{aligned} \alpha_{jk}^{(i)} &= q_j^T \varphi_i \Big/ \left(q_j^t q_j\right), \; 1 \le j < k \\ q_k^{(i)} &= \varphi_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} q_j^t \end{aligned}\right\} \text{Orthogonalize vector (eq1)}$$

$$\left.\begin{aligned} g_k^{(i)} &= \left(q_k^{(i)}\right)^t T \Big/ \left(\left(q_k^{(i)}\right)^t q_k^{(i)}\right) \\ [err]_k^{(i)} &= \left(g_k^{(i)}\right)^2 \left(q_k^{(i)}\right)^t q_k^{(i)} \Big/ \left(T^t T\right) \end{aligned}\right\} \begin{aligned} &\text{Compute least-squares solution (eq2)} \\ &\text{Compute reduction in error (eq3)} \end{aligned}$$

  - and select the regressor with highest reduction in error $q_k = \underset{q_k^{(i)}}{\text{argmax}}\left[[err]_k^{(i)}\right] = \varphi_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} q_j$

- **Stop at iteration M if the residual error falls below some specified tolerance ρ**

$$1 - \sum_{j=1}^{M} [err]_j < \rho$$

  - The regressors $\{\phi_{i1}, \phi_{i2}, \ldots, \phi_{iM}\}$ define the final subset of RBF centers that become selected