L19: 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
- training algorithms for RBFs are significantly faster than those for MLPs
  - And, as we will see today, most of these algorithms have already been presented in previous lectures!

[Bishop, 1995]

Exact function interpolation

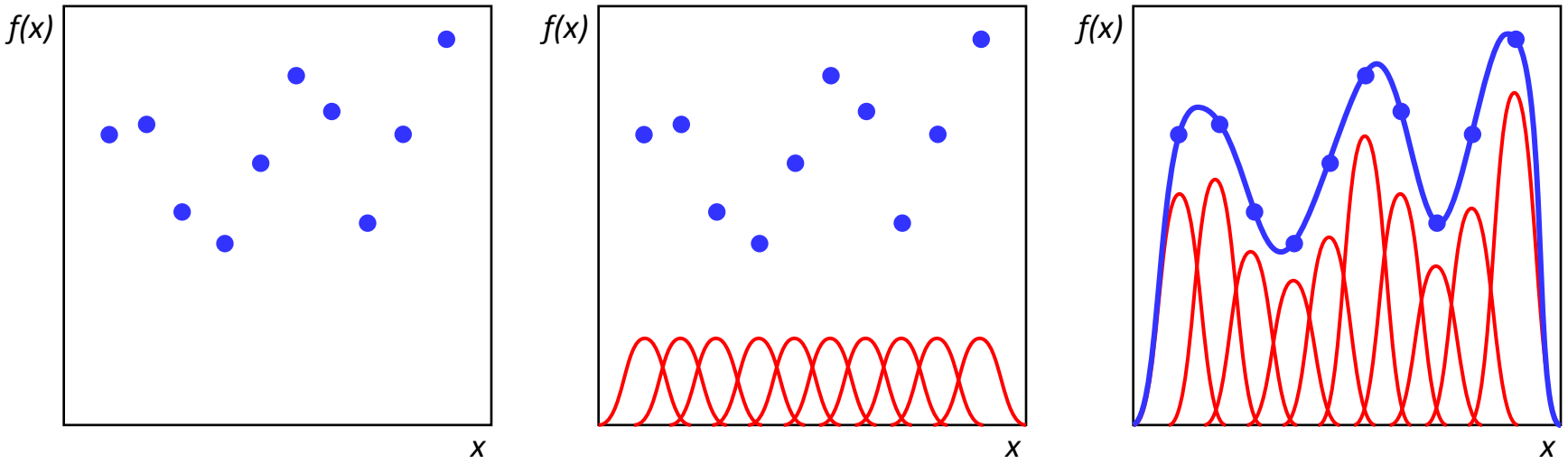
RBFs have their origins in techniques for performing exact function interpolation [Bishop, 1995]

- These techniques place a basis function at each training example

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

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

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



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 RBF layers 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

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

  - 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]

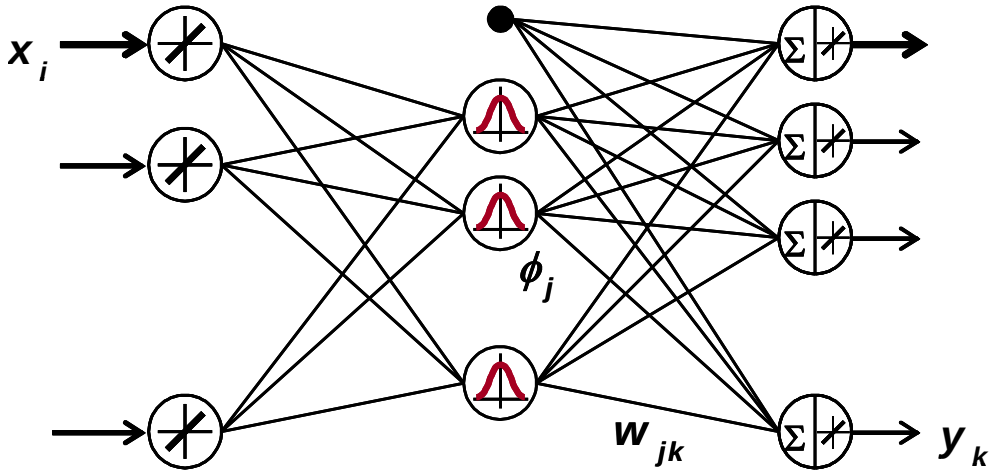
Hidden-to-output mapping

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 k-means (L15)
- 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 GMM using EM (L14)
- 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, H-O weights are computed so as to minimize the MSE error at the output

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

- Now, since the hidden activation patterns Φ are fixed, the optimum weight vector W can be obtained directly from the conventional pseudo-inverse solution (L17)

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

Drawbacks of unsupervised center selection

Hybrid RBF training procedures have one major disadvantage

- 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

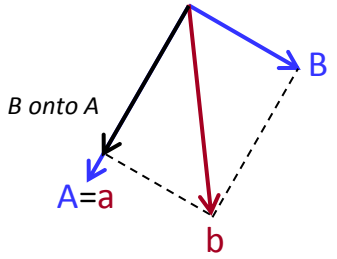
1) Start with N candidate centers and $M = 0$ centers
2) For each of the $k = N - M$ remaining candidates
    a) Add the k-th center to the existing M centers
    b) Compute the pseudo-inverse solution
    c) Compute the resulting SSE at the output
3) Choose the k-th candidate that yields lowest SSE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

- This implementation is, however, very inefficient since the pseudo-inverse $W = (\Phi'\Phi)^{-1}\Phi'T$ 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 $Q'Q$ becomes diagonal

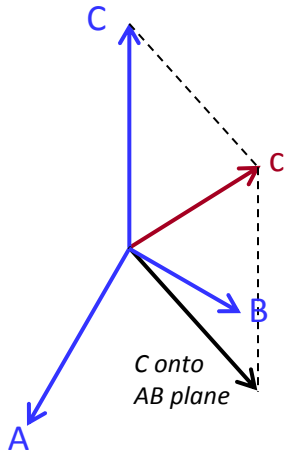
Gram-Schmidt orthogonalization

Procedure

- Assume that we have three (independent) vectors a, b and c, from which we wish to construct three orthogonal vectors A, B and C
- 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}



[Strang, 1998]

– For M basis vectors $\Phi = (\varphi_1, \varphi_2 \dots \varphi_M)$, Gram-Schmidt generalizes to

$$\left.\begin{aligned} q_1 &= \varphi_1 \\ \alpha_{ik} &= \frac{q_i' \varphi_k}{q_i' q_i} \quad 1 \le \mathrm{i} < k \\ q_k &= \varphi_k - \sum_{i=1}^{k-1} \alpha_{ik} q_i \end{aligned}\right\} k = 2 \dots M \qquad Equation\ (\boldsymbol{EQ1})$$

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

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

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

Geometric interpretation of the pseudo-inverse

Assume dataset $X = \{(x^{(1}, t^{(1}), (x^{(2}, t^{(2}) \dots (x^{(N}, t^{(N})\}$

- 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

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

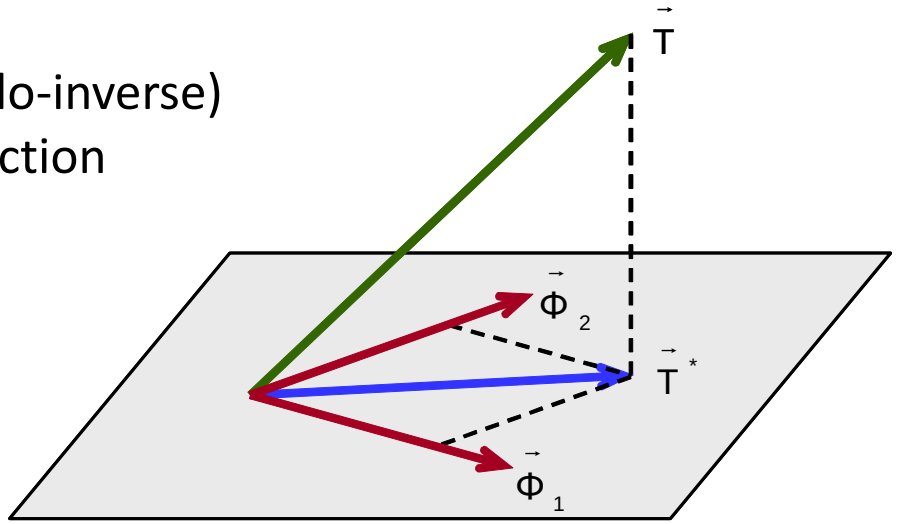
  - where E is the vector of prediction errors, whose sum square we seek to minimize
- Notice that the activation of a particular radial basis for all the training examples $\Phi_{\mathrm{k}} = \left[\varphi_k^{(1}, \varphi_k^{(2} \dots \varphi_k^{(N}\right]'$ can be treated as a vector
- And the desired target $\mathrm{T} = \left[t^{(1}, t^{(2} \dots t^{(N}\right]'$ can also be treated as a vector

The 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

Keeping in mind the geometric interpretation of LS and the GSO procedure, we are now ready to present the OLS algorithm

- As mentioned earlier, OLS constructs a set of orthogonal vectors Q for the space spanned by basis vectors Φ_k such that $\Phi = QA$ (A upper triangular)
- Using this orthogonal representation, the RBF solution is expressed as

$$T = \Phi W = QG$$

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

$$G = (Q'Q)^{-1}Q'T$$

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

$$g_i = \frac{q_i' T}{q_i^T q_i} \quad equation\ (\boldsymbol{EQ2})$$

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

[Chen et al., 1991]

How are basis functions selected?

- The sum of squares or energy of the target vector T is

$$T'T = \sum_{i=1}^{M} g_i^2 q_i' q_i + E'E$$

- Assuming that the mean of T has been removed, then the variance of T is

$$N^{-1}T'T = N^{-1}\sum_{i=1}^{M} g_i^2 q_i' q_i + N^{-1}E'E$$

  - The first term, $N^{-1}\sum_{i=1}^{M} g_i^2 q_i' q_i$, is the part of the desired variance explained by the regressors, whereas $N^{-1}E'E$ is the unexplained variance
  - Therefore, $N^{-1}g_i^2 q_i' q_i$ is the increment in the explained output variance achieved by adding regressor q_i, which contributes to a reduction of the error (relative to the total $T'T$) by

$$[err]_i = \frac{g_i^2 q_i' q_i}{T'T} \quad Equation\ (\boldsymbol{EQ3})$$

- 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

OLS algorithm

At the first step, for $1 \leq \mathrm{i} \leq M$, compute

- Orthogonalize vector (EQ1): $q_1^{(i)} = \varphi_i$
- Compute LS solution (EQ2): $g_1^{(i)} = {q_1^{(i)}}^T T / \left({q_1^{(i)}}' q_1^{(i)} \right)$
- Compute error reduction (EQ3): $[err]_i^{(1)} = {g_1^{(i)}}^2 {q_1^{(i)}}' q_1^{(i)} / (T^T T)$
  - and select 'regressor that yields highest reduction in error $q_1 = \underset{q_1^{(i)}}{\operatorname{argmax}} [err]_1^{(i)} = \varphi_{i_1}$

At the k-th step, for $1 \leq \mathrm{i} \leq M$, and i not already selected

- Orthogonalize vector (EQ1): $\begin{cases} \alpha_{jk}^{(i)} = q_j' \varphi_i / \left(q_j' q_j \right), 1 \leq j < k \\ q_k^{(i)} = \varphi_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} q_j' \end{cases}$
- Compute LS solution (EQ2): $g_k^{(i)} = {q_k^{(i)}}' T / \left({q_k^{(i)}}' q_k^{(i)} \right)$
- Compute error reduction (EQ3): $[err]_k^{(i)} = {g_k^{(i)}}^2 {q_k^{(i)}}' q_k^{(i)} / (T' T)$
  - and select regressor $q_k = \underset{q_k^{(i)}}{\operatorname{argmax}} [err]_k^{(i)} = \varphi_{i_k} - \sum_{j=1}^{k-1} \alpha_{jk} q_j$

Stop at iteration M if residual error falls below pre-specified tolerance ρ

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

- The regressors $\left\{ \varphi_{i_1}, \varphi_{i_2} \ldots \varphi_{i_M} \right\}$ define the final subset of RBF centers

[Chen et al., 1991]