

INT301 W9

Radial-basis function networks

Introduction

神经网络的可以看作是一个曲线拟合 (curve-fitting) 或函数近似 (function approximation) 问题。

对于 curve-fitting (approximation in high-dimensional spaces) 问题，可以等价于找到 interpolating surface (插值平面) in the space。这叫 **Curve-fitting or interpolation problem**。

注：实际问题中我们往往得不到一个函数关系具体的表达式，但往往可以通过观测等手段得到其在一些点处的函数值，这时，就需要用到插值的思想，通过对某些点已知的函数 (basis function) 值构造插值函数，用插值函数代替原函数来研究问题。

Radial-Basis Functions

Radial-basis functions (径向基函数) 可以按照以下形式构建插值函数 F :

$$F(\mathbf{x}) = \sum_{i=1}^N w_i \phi(\|\mathbf{x} - \mathbf{x}_i\|)$$

$\phi(\|\mathbf{x} - \mathbf{x}_i\|)$ 是一组非线性径向基函数 (nonlinear radial-basis functions), \mathbf{x}_i 是这些函数的中心, $\|\bullet\|$ 是 Euclidean norm。

A **radial basis function (RBF)** is a real-valued function whose value depends only on the distance from the origin

$$\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$$

Alternative form: depends on the distance from a **center \mathbf{c}**

$$\phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$$

Any function that satisfies the property is a radial basis function

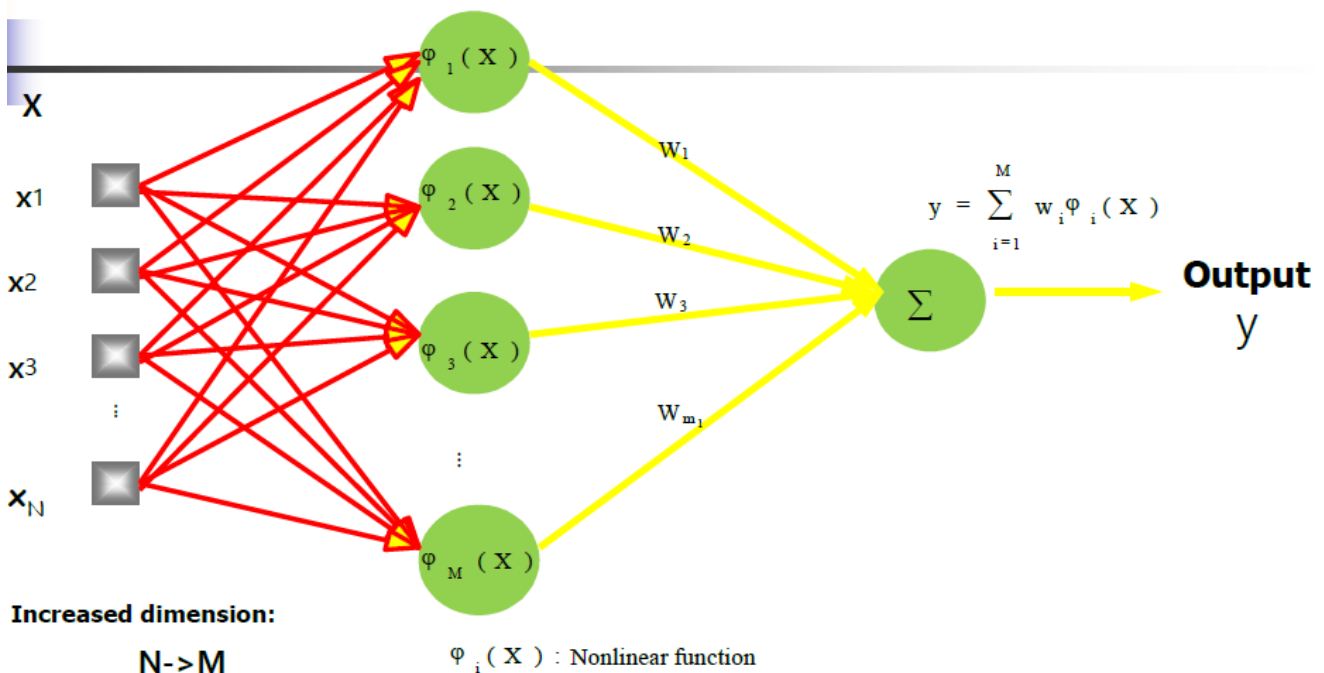
Regularization Networks

Regularization network 将插值函数 F 建模为多元高斯函数的线性叠加 (linear superposition of multivariate Gaussian functions, 其中 Gaussian functions 是常见的 basis function), 其中叠加的数量等于 input examples 的数量 N :

$$F(x) = \sum_{i=1}^N w_i \exp \left(-\|x - x_i\|^2 / 2\sigma_i^2 \right)$$

这是一个通用近似器 (**universal approximator**), 给定足够多的 units, 它可以任意地近似紧集 (compact set) 上的任何多元连续函数。

RBF Network Structure



An RBF network architecture consists of 3 layers:

- input layer: which passes the example vectors to the next layers
- hidden layer: applies a non-linear transformation function to the inputs, and expands them in the usually very high-dimensional hidden space
- output layer: applies a linear transformation to the given vector

RBF Network: Rationale

非线性函数可以将困难的非线性插值问题转换为涉及线性映射的更简单的问题。这就是为什么 RBF network 被设计出来。RBF 执行从输入空间到隐藏空间的非线性映射, 然后执行从隐藏空间到输出空间的线性映射。

Function Mapping by RBF Networks

RBF network 将插值函数 F 建模为有限的多元高斯函数的线性叠加：

$$F(\mathbf{x}) = \sum_{i=1}^n w_i \exp \left(-\frac{\|\mathbf{x} - \mathbf{t}_i\|^2}{2\sigma_i^2} \right)$$

其中： w_i 是权重，basis function 的数量 n 少于 training set 中点的数量 (i.e. $n < N$)，basis function 的中心 \mathbf{t}_i 是提前确定的。

RBF Learning

Weights can be computed by the **pseudo-inverse** method.

For an example (x_i, d_i) , consider the output of the network

$$o(x_i) = w_1 \varphi_1(\|x_i - t_1\|) + \dots + w_n \varphi_n(\|x_i - t_n\|)$$


We would like $o(x_i) = d_i$ for each example, that is

$$w_1 \varphi_1(\|x_i - t_1\|) + \dots + w_n \varphi_n(\|x_i - t_n\|) = d_i$$

This can be re-written in **matrix form** for one example

$$[\varphi_1(\|x_i - t_1\|) \cdots \varphi_n(\|x_i - t_n\|)] \cdot [w_1 \cdots w_n]^T = d_i$$

注： x_i 相当于 training example, d_i 相当于 label



$$\begin{bmatrix} \varphi_1(\|x_1 - t_1\|) \dots \varphi_n(\|x_1 - t_n\|) \\ \dots \\ \varphi_1(\|x_N - t_1\|) \dots \varphi_n(\|x_N - t_n\|) \end{bmatrix} [w_1 \dots w_n]^T = [d_1 \dots d_N]^T$$

for all the examples at the same time.

Let $\Phi = \begin{bmatrix} \varphi_1(\|x_1 - t_1\|) & \dots & \varphi_n(\|x_1 - t_n\|) \\ \dots & \dots & \dots \\ \varphi_1(\|x_N - t_1\|) & \dots & \varphi_n(\|x_N - t_n\|) \end{bmatrix}$

then we can write $\Phi \begin{bmatrix} w_1 \\ \dots \\ w_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \dots \\ d_N \end{bmatrix}$

- Φ is the $n \times n$ design matrix of basis functions

$$\phi = \exp\left(-\frac{\|x - t_i\|^2}{2\sigma_i^2}\right)$$

$$\Phi = \begin{bmatrix} \varphi_1(\|x_1 - t_1\|) & \dots & \varphi_n(\|x_1 - t_n\|) \\ \dots & \dots & \dots \\ \varphi_1(\|x_N - t_1\|) & \dots & \varphi_n(\|x_N - t_n\|) \end{bmatrix} \quad \Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \dots & \phi_{2n} \\ \dots & \dots & \dots & \dots \\ \phi_{N1} & \phi_{N2} & \dots & \phi_{Nn} \end{bmatrix}$$

$$\Phi W = \Phi \begin{bmatrix} w_1 \\ \dots \\ w_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \dots \\ d_N \end{bmatrix} = d$$

The unknown weights can be found by solving the following linear matrix equation using pseudo-inverse:

$$W = (\Phi^T \Phi)^{-1} \Phi^T d$$

where: W is the weight vector, d is the response vector, and Φ is the **design matrix**

虽然这里让 W 和 Φ 以及 d 挂钩了，但更新权重的时候只更新 W ，和其他的没关系。

注: Φ^+ 是 Φ 的 pseudo-inverse (伪逆), 即 $\Phi^+ \times \Phi = 1$

If Φ^+ is the pseudo-inverse of the matrix Φ we obtain the weights using the following formula

$$[w_1 \dots w_n]^T = \Phi^+ [d_1 \dots d_N]^T$$

Pseudo inverse

Φ has a unique pseudo-inverse Φ^+ satisfying the conditions:

$$\Phi \Phi^+ \Phi = \Phi$$

$$\Phi^+ \Phi \Phi^+ = \Phi^+$$

$$\Phi^+ \Phi = (\Phi^+ \Phi)^T$$

$$\Phi \Phi^+ = (\Phi \Phi^+)^T$$

RBF Training Algorithm

- **Initialization:** Examples $\{(\mathbf{x}_e, d_e)\}_{e=1}^N$
- **Determine:**
 - the network structure with a number n of basis functions $\phi_i, i=1,2,\dots,n$
 - the basis function centers $\mathbf{t}_i, i=1,2,\dots,n$
 - e.g.: using ***k-means clustering*** algorithm
 - the basis function variances $\sigma_i^2, i=1,2,\dots,n$
- **Compute:**
 - the outputs from each example with the Gaussian basis functions:
$$\phi_{ei} = \frac{\exp(-\|\mathbf{x}_e - \mathbf{t}_i\|^2)}{2\sigma_i^2}$$

// at each i -th hidden unit where $i=1,2,\dots,n; e=1,2,\dots,N$
 - the correlation matrix: $\Phi^T \Phi$ and pseudo inverse $(\Phi^T \Phi)^{-1}$
 - the vector: $\Phi^T d$
 - the weights: $W = (\Phi^T \Phi)^{-1} \Phi^T d$

注：径向基函数的中心 \mathbf{t}_i 和方差 σ_i^2 可以与输出权重一起训练，但这需要很长时间。因此，它们通常是事先确定的。

Implementing RBF Networks

XOR Example

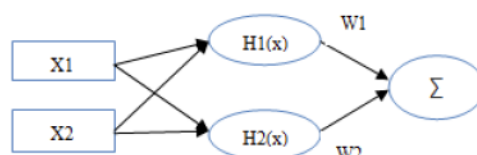
Consider the XOR example:

(x_1, x_2)	$ $	Y

$(0, 0)$	$ $	0
$(0, 1)$	$ $	1
$(1, 0)$	$ $	1
$(1, 1)$	$ $	0

- The XOR example can be learned by an RBF network with the following structure
 - two hidden nodes
 - one output node

$$F(x) = \sum_{i=1}^2 w_i \exp\left(-\frac{\|\mathbf{x} - \mathbf{t}_i\|^2}{2\sigma_i^2}\right)$$



- **Solution:** Training of this RBF to learn the XOR examples involves the following steps

- **Initialization:**

- the network structure has two basis functions ϕ_i

$$\phi_i = \exp(-||\mathbf{x} - \mathbf{x}_i||^2)$$
- choose the centers: $\mathbf{x}_1 = (1,1), \mathbf{x}_2 = (0,0), 2\sigma_i^2 = 1$

XOR

$(x_1, x_2) \mid Y$	
$(0, 0) \mid 0$	
$(0, 1) \mid 1$	
$(1, 0) \mid 1$	
$(1, 1) \mid 0$	

注：因为是二分类，因此选两个 centers。

- **Training:**

$$\begin{aligned}\phi_{11} &= \exp(-((0-1)^2 + (0-1)^2)) = \exp(-2) = 0.1353 \\ \phi_{21} &= \exp(-((0-1)^2 + (1-1)^2)) = \exp(-1) = 0.3678 \\ \phi_{31} &= \exp(-((1-1)^2 + (0-1)^2)) = \exp(-1) = 0.3678 \\ \phi_{41} &= \exp(-((1-1)^2 + (1-1)^2)) = \exp(0) = 1 \\ \phi_{12} &= \exp(-((0-0)^2 + (0-0)^2)) = \exp(0) = 1 \\ \phi_{22} &= \exp(-((0-0)^2 + (1-0)^2)) = \exp(-1) = 0.3678 \\ \phi_{32} &= \exp(-((1-0)^2 + (0-0)^2)) = \exp(-1) = 0.3678\end{aligned}$$

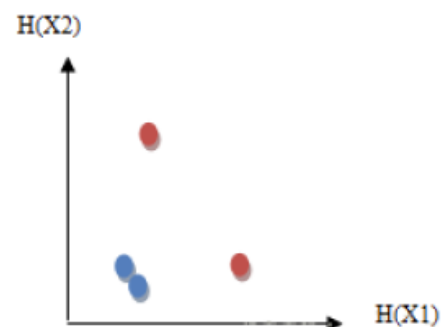
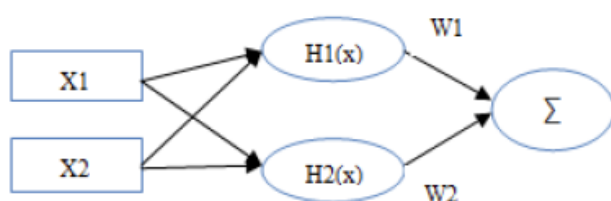
$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{Nn} \end{bmatrix} \quad \Phi = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \\ \phi_{31} & \phi_{32} \\ \phi_{41} & \phi_{42} \end{bmatrix} = \begin{bmatrix} 0.1353 & 1 \\ 0.3678 & 0.3678 \\ 0.3678 & 0.3678 \\ 1 & 0.1353 \end{bmatrix}$$

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \\ \phi_{31} & \phi_{32} \\ \phi_{41} & \phi_{42} \end{bmatrix} = \begin{bmatrix} H1(x) & H2(x) \\ 0.1353 & 1 \\ 0.3678 & 0.3678 \\ 0.3678 & 0.3678 \\ 1 & 0.1353 \end{bmatrix}$$

XOR

$(x_1, x_2) \mid Y$	

$(0, 0) \mid 0$	
$(0, 1) \mid 1$	
$(1, 0) \mid 1$	
$(1, 1) \mid 0$	



Finding RBF Centers

Random Selection

最简单的方法是从给定的 training example 中随机选择中心。

K-means Clustering

使用 K 均值聚类算法选择中心时，假设相似的输入会产生相似的输出，我们将这些相似的输入对捆绑在一起形成聚类。K 均值聚类仅将 RBF 中心放置在输入空间中存在大量训练示例的区域中。

Cluster

Cluster: 数据的集合，同一集群中元素彼此相似，不同集群中的元素不同。

Cluster analysis: 无监督学习，根据数据中发现的特征查找数据之间的相似性，并将相似的数据对象分组到聚类中。

A good clustering method will produce high quality clusters with:

- high intra-class similarity
similar to one another within the same cluster
- low inter-class similarity
dissimilar to the objects in other clusters

聚类方法的质量也是通过其发现部分或全部的 hidden patterns 的能力来衡量的。

Similarity and Dissimilarity

Distances are normally used to measure the similarity or dissimilarity between two data objects。

Minkowski distance

$$d(i, j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \dots + |x_{ip} - x_{jp}|^q)}$$

where $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ are two p -dimensional data objects, q is a positive integer。

If $q = 1$, d is Manhattan distance

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

If $q = 2$, d is Euclidean distance:

$$d(i, j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2)}$$

K-means clustering algorithm

该算法将数据点划分为 K 个不相交的子集。

聚类分类标准：

- 聚类中心设置在数据的高密度区域
- 数据点被分配到与中心具有最小距离的聚类

$$J = \sum_{j=1}^K \sum_{n \in S_j} \|\mathbf{x}_n - \mathbf{c}_j\|^2$$

where

S_j : the j -th cluster containing N_j data points

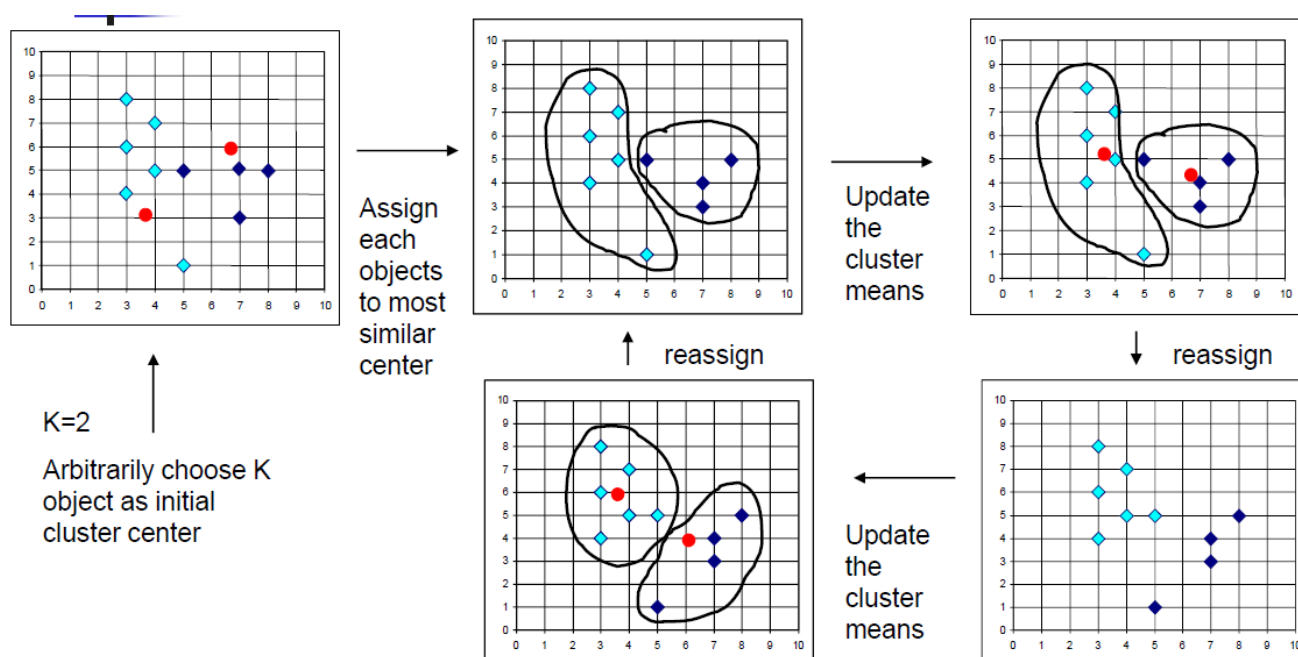
$\mathbf{c}_j = \frac{1}{N_j} \sum_{n \in S_j} \mathbf{x}_n$: the mean of the data points in cluster S_j

注： x_n 是 j -th cluster 中的点， c_j 是 j -th cluster 的中心， $\sum_{n \in S_j} \|\mathbf{x}_n - \mathbf{c}_j\|^2$ 计算 j -th cluster 中所有点到中心的距离，因此 J 就是所有 cluster 中点到中心点距离的总和，因此我们的目的是最小化 J 。

Algorithm process:

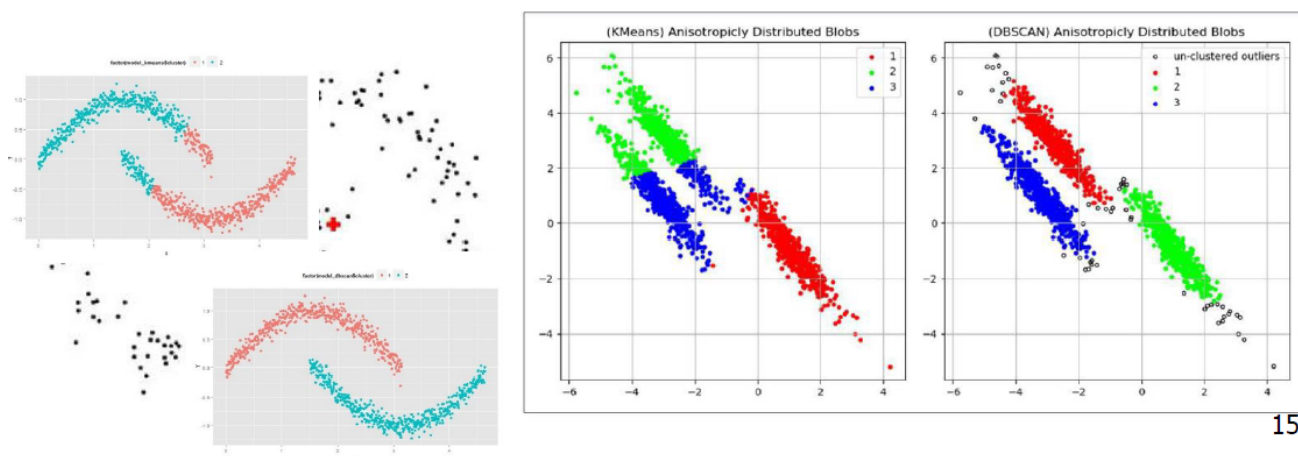
- 1, 随机选择 K 个中心点。计算 training examples 中的每个点到这 K 个中心点的距离，把每个点分到离它最近的聚类中；
- 2, 根据同一个聚类中的所有点的坐标，计算出该类的 mean，作为新的中心点；
- 3, 再计算 training examples 中的每个点到这 K 个新中心点的距离，把每个点分到离它最近的聚类中；

4, 如果第 3 步中没有出现分类的变化, 则停止算法; 否则, 回到第 2 步。



Issues

此方法不适用于具有非凸形状的聚类, 并且对噪声和异常值元素敏感。



注: 上图左边是非凸形状, 右边是噪声和异常值; 其中每个图的左边是 kmeans 算法, 右边是其他算法。

Improve k-means

Repeated k-means:

- Try several random initializations and take the best.

Better initialization:

- Use some better heuristic to allocate the initial distribution of code vectors.
 - Designing good initialization is not any easier than designing good clustering algorithm at the first place.

RBF Networks vs. MLP

Similarities

- The RBF Networks and the MLP are **layered feedforward networks** that produce **nonlinear function mappings**
- They are both proven to be **universal approximators**

Differences

- RBF 网络只有一个隐藏层，而 MLP 网络有一个或多个隐藏层，具体取决于应用任务
- MLP 的隐藏层和输出层中的节点使用相同的激活函数，而 RBF 在每个节点上使用不同的激活函数（使用不同的中心和方差）
- MLP 的隐藏层和输出层都是非线性的，而 RBF 只有隐藏层是非线性的（输出层是线性的）
- RBF 节点中的激活函数计算输入示例和中心之间的欧氏距离，而 MLP 的激活函数计算输入示例和传入权重的内积
- MLP constructs **global approximations** while RBF construct **local approximations**