Lecture 14：Radial Basis Function Network

> 课件链接：Hsuan-Tien Lin - radial basis function network
>
> **Radial Basis Function Network(径向基函数网络)**
>
> - RBF Network Hypothesis：径向基函数网络的假说形式
> - RBF Network Learning：径向基函数网络的学习过程
> - k-Means Algorithm：k均值演算法
> - k-Means and RBF Network in Action：k均值与径向基函数网络的实践

一. RBF Network Hypothesis：径向基函数网络的假说形式

回顾：Gaussian SVM——基于高斯核函数的支撑向量机

$$g_{SVM}(\mathbf{x}) = sign\left(\sum_{SV}\alpha_n y_n exp(-\gamma||\mathbf{x}-\mathbf{x}_n||^2)+b\right)$$

Gaussian SVM的实质是在一个无线多维的空间内找一个large-margin的边界作为最优的分离超平面。从结果上看，Gaussian SVM是一堆高斯函数的线性组合，组合系数是$\alpha_n y_n$，而每个高斯函数的中心均为一个样本点(支撑向量)。

之前提到，高斯核又被称为径向基函数核，即Radial Basis Function Kernel。其中的radial是指函数值只与某个距离有关，该距离即输入样本点$\mathbf{x}$与中心点$\mathbf{x}_n$的距离；basis function是指将要把这些函数进行线性组合，在Gaussian SVM中组合系数为$\alpha_n y_n$。

另一种看待Gaussian SVM回传函数的方式

令：

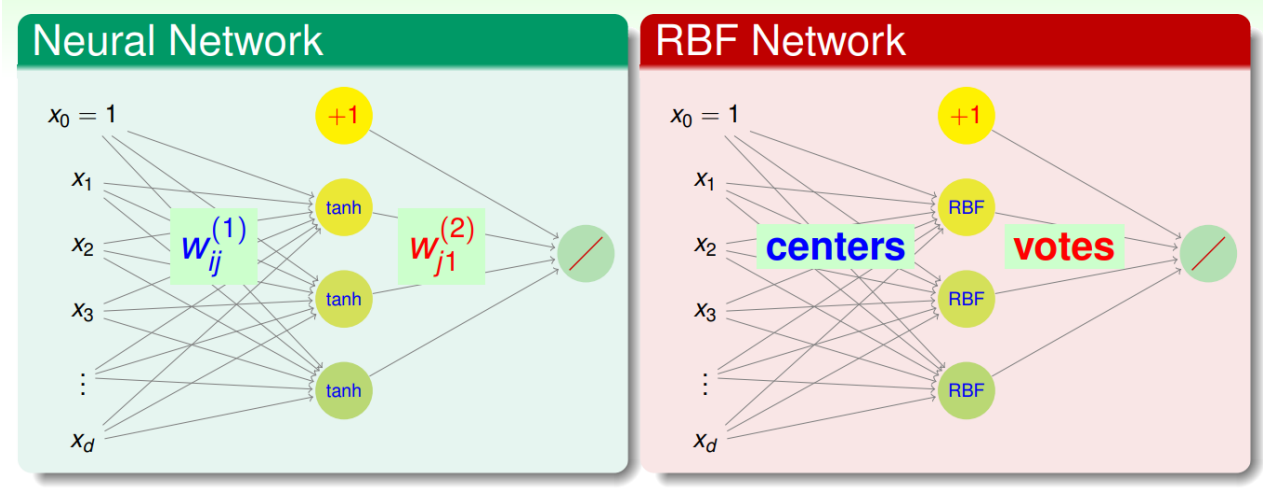
$$g_n(\mathbf{x}) = y_n exp(-\gamma||\mathbf{x}-\mathbf{x}_n||^2)$$

那么Gaussian SVM可以写作：

$$g_{SVM}(\mathbf{x}) = sign\left(\sum_{SV}\alpha_n g_n(\mathbf{x})+b\right)$$

即Gaussian SVM可以看做是线性聚合——linear aggregation，聚合的元素是一个个小g——radial hypotheses。每一个小g都依据一个样本点(支撑向量)定义，其含义是：**根据输入样本与支撑向量样本点的相似性(距离)决定"投票票数"的多少**。例如，某个小g是定义在y为-1的某个支撑向量上的；如果输入样本x与该支撑向量的距离很小，那么高斯函数那一项就会比较大，那么就会给y，即-1，乘到比较大的正数，导致该小g的输出是一个很负的负数，也可以理解为该g投了非常偏向-1类别的票——依据是小g认为该输入样本与其背后的支撑向量样本距离比较近，因此"应该很相似"。

综上，Gaussian SVM可以看做是一堆radial hypotheses的线性聚合。而**径向基函数(RBF)网络也是一堆radial hypotheses的线性聚合。**



由上图可以看出，RBF网络与神经网络的输出层是一样的，均为线性聚合。但它们的**隐藏层有着显著的区别**：神经网络隐藏层的每个单元在：①做内积②通过转换函数如tanh；而RBF网络的每个单元在：①计算距离②通过径向基函数如Gaussian。

RBF网络的hypothesis

$$h(\mathbf{x}) = Output\left(\sum_{m=1}^{M} \beta_m RBF(\mathbf{x}, \mu_m) + b\right)$$

b可以暂时略去。因此，需要决定的参数有：

- （M：有多少个单元）；
- （RBF：径向基函数的选择）；
- （Output：取决于面向的问题）；
- μ_m：每个单元的中心(center)；
- β_m：每个单元的投票权重。

用上述hypothesis的观点看Gaussian SVM：

- M是支撑向量的数量；
- RBF是高斯函数；
- Output是sign，因为在做二元分类问题；
- 每个单元的中心即为每个支撑向量；
- 每个单元的投票权重为$\alpha_n y_n$。

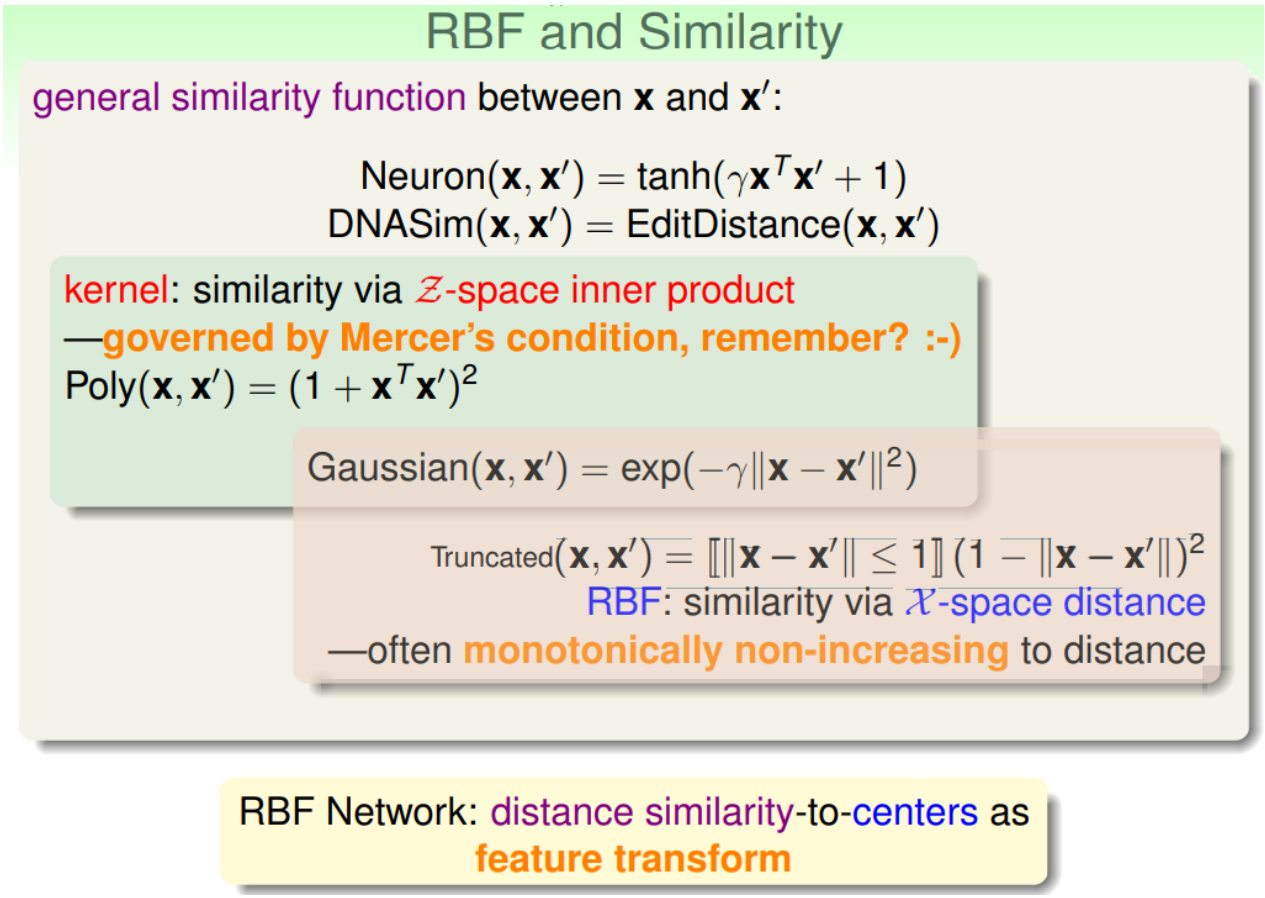
学习RBF网络的过程，就是在RBF、Output给定的情形下，决定μ_m与β_m。

关于相似性(Similarity)的解释

在SVM相关章节中我们提到，kernel其实在描述两个样本点的相似性，这种描述是通过"偷吃步"的方法计算Z空间中的内积完成的。但并不是所有描述两个样本点相似性的函数都是合法的kernel——合法的kernel需要满足Mercer's condition。

在刚刚，我们又接触了一种描述两个样本点的相似性的工具——RBF，即径向基函数。不同于kernel通过计算Z空间内积的方式，RBF透过X空间的距离(的函数)来衡量两个样本点的相似性。因为距离越近，往往相似性越大，因此RBF函数往往是关于距离单调递减的。

当然，衡量相似性的方式不只有kernel与RBF，还有其他的一些函数。它们的关系如下图所示。其中，高斯函数既属于kernel类，也属于RBF类：



二. RBF Network Learning：径向基函数网络的学习过程

完全RBF网络：Full RBF Network

$$h(\mathbf{x}) = Output\left(\sum_{m=1}^{M} \beta_m RBF(\mathbf{x}, \mu_m)\right)$$

完全RBF网络是指M=N且$\mu_m = \mathbf{x}_m$，即分别以每一个训练样本点为center构造RBF函数。这种做法的含义是：每一个训练样本点都会对输入的预测产生影响，影响的效力用β_m体现。例如，我们考虑所有训练样本的相似性的投票权相同，即取$\beta_m = 1 \cdot y_m$，对于二元分类问题：

$$g_{uniform}(\mathbf{x}) = sign\left(\sum_{m=1}^{N} y_m exp(-\gamma||\mathbf{x} - \mathbf{x}_m||^2)\right)$$

对于该"均匀"的RBF网络，由于每个训练样本的投票权相同，因此离输入$\mathbf{x}$最近的那个点很有可能就直接决定了预测结果——**因为离输入最近的点RBF函数值最大，而高斯函数又是衰退很快的函数**。因此，我们可能不需要计算上式中的求和项，而是直接找到离输入$\mathbf{x}$最近的那**一个**训练样本点，然后用它的标签作为预测结果：

$$g_{nbor}(\mathbf{x}) = y_m \ such\ that\ \mathbf{x}\ closest\ to\ \mathbf{x}_m$$

这样寻找"最近邻"的方法，称为**nearest neighbor model**。最近邻方法也可以进行延伸，得到**k近邻模型**——找离输入点最近的k个训练样本，然后做均匀投票动作。近邻模型是一种**偷懒 (lazy)** 的方法，似乎并没有进行训练，只是在进行预测的时候才开始计算每个训练样本与输入点的相似性(距离)。因此，训练不费时间，测试很费时间。

完全RBF网络 & 过拟合

现在我们考虑这样一个完全RBF网络：①针对基于平方误差的回归问题，直接输出分数，而不是分数的sign；②将β设置为需要最佳化的参数，而非上面直接将其给定为y的值：

$$h(\mathbf{x}) = \sum_{m=1}^{N} \beta_m RBF(\mathbf{x}, \mu_m)$$

对于上面的这个hypothesis，我们可以将其看成：

①特征转换：用每个RBF对输入$\mathbf{x}$作用，得到转换的每一个维度，

$$\mathbf{z}_n = [RBF(\mathbf{x}_n, \mathbf{x}_1), RBF(\mathbf{x}_n, \mathbf{x}_2), \cdots, RBF(\mathbf{x}_n, \mathbf{x}_N)] \in \mathbb{R}^N$$

②线性回归(如果$Z^T Z$可逆)

$$\beta = (Z^T Z)^{-1} Z^T \mathbf{y}$$

注意，$Z \in \mathbb{R}^N \times \mathbb{R}^N$的矩阵，且是一个**对称方阵**：

$$z_{ij} = RBF(\mathbf{x}_i, \mathbf{x}_j) = RBF(\mathbf{x}_j, \mathbf{x}_i) = z_{ji}$$

如果①所有的$\mathbf{x}_n$各不相同，②RBF函数是高斯函数，那么Z可逆。因此：

$$\begin{aligned} \beta^* &= (Z^T Z)^{-1} Z^T \mathbf{y} \\ &= (ZZ)^{-1} Z\mathbf{y} \\ &= Z^{-1} Z^{-1} Z\mathbf{y} \\ &= Z^{-1}\mathbf{y} \end{aligned}$$

如果用上述最佳化的RBF网络进行预测：

$$\begin{aligned} g_{RBF}(\mathbf{x}_1) &= \beta^T \mathbf{z}_1 \\ &= (Z^{-1}\mathbf{y})^T \mathbf{z}_1 \\ &= \mathbf{y}^T (Z^{-1})^T \mathbf{z}_1 \\ &= \mathbf{y}^T (Z^T)^{-1} \mathbf{z}_1 \\ &= \mathbf{y}^T Z^{-1} \mathbf{z}_1 \\ &= \mathbf{y}^T [1\ 0\ \cdots\ 0]^T \\ &= y_1 \end{aligned}$$

可见，对于每一个训练样本，该完全RBF网络能够做到100%正确的预测。因此，$E_{in} = 0$——可能会过拟合。因此可以做正则化，例如在最优化β时使用ridge regression：

$$\beta^* = (Z^T Z + \lambda I)^{-1} Z^T \mathbf{y}$$

另一种正则化方法：Fewer Centers

Gaussian SVM里，并没有使用所有训练样本点的高斯函数，仅使用了SV为中心的高斯函数。因此，考虑$M << N$，而非完全RBF网络中的$M = N$。例如，1000笔训练资料，只用其中10笔作为center构造高斯函数的线性组合。即，**用减少center数量的方式进行正则化**。因此，我们需要在所有训练资料中，找到一部分**具有代表性的资料**，用它们来构建RBF网络。这些"代表"μ_m，称为**prototypes**。那么，如果找出"好的代表"？K-means方法。

三. k-Means Algorithm：k均值演算法

现在的目标：寻找**Good Prototypes**。

在完全RBF网络中，每个训练样本点都是"代表"。但如果两个训练样本点很接近，那么它们可能可以"共用"一个"代表"。这样问题便转化为**cluster**，即分群/聚类问题。**我们需要将训练资料分群，并在每个群中选择一个合适的"代表"**：

- $\{\mathbf{x}_n\} \rightarrow S_1, S_2, \cdots, S_M$，各群的交集为空；
- 从每个S_m中选择一个"代表"μ_m；
- Hope：如果$\mathbf{x}_1$与$\mathbf{x}_2$均属于S_m，那么它们应该和该群的"代表"很接近，即：$\mu_m \approx \mathbf{x}_1 \approx \mathbf{x}_2$。

因此，我们构建如下的代价函数(cluster error with squared error measure)：

$$E_{in}(S_1, \cdots, S_M;\ \mu_1, \cdots, \mu_M) = \frac{1}{N} \sum_{n=1}^{N} \sum_{m=1}^{M} I[\mathbf{x}_n \in S_m] \cdot ||\mathbf{x}_n - \mu_m||^2$$

目标是：

$$\min_{\{S_1, \cdots, S_M;\ \mu_1, \cdots, \mu_M\}} E_{in}(S_1, \cdots, S_M;\ \mu_1, \cdots, \mu_M)$$

因为该最优化问题有两组变量，一组是分群变量$S_1, \cdots, S_M$，另一组是代表变量$\mu_1, \cdots, \mu_M$。考虑**交替优化(optimize alternatingly)**：每次固定一组变量，最优化另一组，然后再固定刚刚优化过的那一组，最优化此前固定的那一组，这样交替迭代，直至收敛：

- 如果$\mu_1, \cdots, \mu_M$确定，对于每个$\mathbf{x}_n$，它有且仅有一个归属的群，且它与该群代表的距离的平方是惩罚函数的组成部分。因此，如果要使代价函数最小，每个点必须归属于离它最近的那个代表的群中。
- 如果$S_1, \cdots, S_M$确定：

$$\nabla_{\mu_m} E_{in} = -2 \sum_{n=1}^{N} I[\mathbf{x}_n \in S_m](\mathbf{x}_n - \mu_m) = 0$$

因此：

$$\mu_m = \frac{\sum_{\mathbf{x}_n \in S_m} \mathbf{x}_n}{|S_m|}$$

上式说明，在分群确定的情况下，每个群中的代表最好是该群中所有点的平均——consensus。

综上，我们得到了**k-Means演算法**。

k-Means Algorithm

1. initialize $\mu_1, \mu_2, \ldots, \mu_k$: say, as k randomly chosen $\mathbf{x}_n$
2. **alternating optimization** of E_{in}: repeatedly
   1. optimize $S_1, S_2, \ldots, S_k$:
      each $\mathbf{x}_n$ 'optimally partitioned' using its closest μ_i
   2. optimize $\mu_1, \mu_2, \ldots, \mu_k$:
      each μ_n 'optimally computed' as consensus within S_m

   until **converge**

因为交替优化的两步都是在使E_{in}更小，而E_{in}的下限是0，因此该算法一定会收敛。

RBF Network using k-Means

- 首先使用k-Means算法得到M=k个代表$\{\mu_m\}_{m=1}^{M}$；
- 构建特征转换：

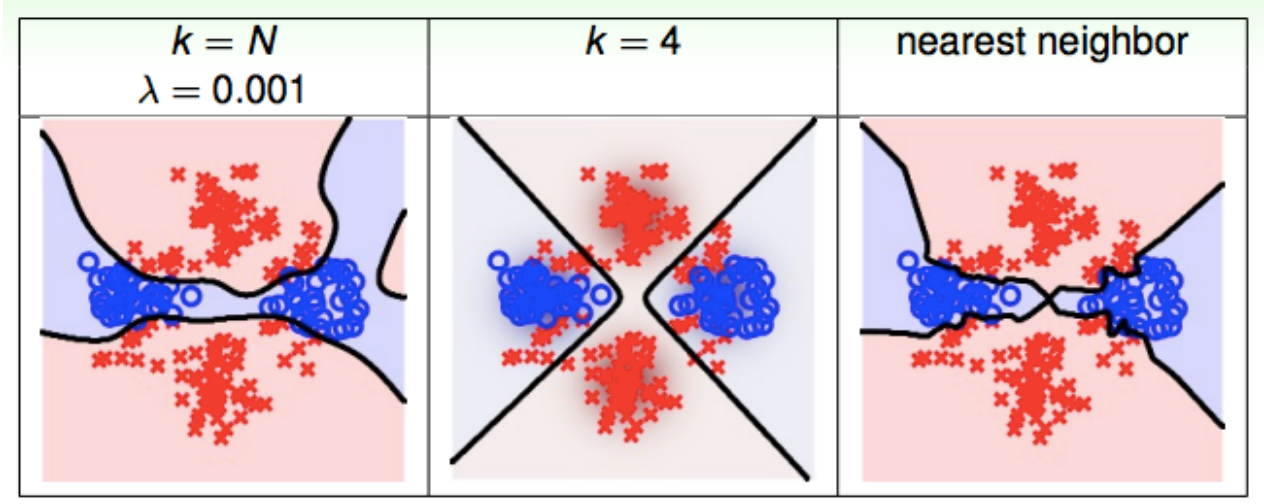
$$\Phi(\mathbf{x}) = [RBF(\mathbf{x}, \mu_1), RBF(\mathbf{x}, \mu_2), \cdots, RBF(\mathbf{x}, \mu_M)]$$

- 在转换后的数据$\{\Phi(\mathbf{x}_n), y_n\}$上做线性模型得到最佳的$\beta$；
- 回传$g_{RBFNET}(\mathbf{x}) = LinearHypothesis(\beta, \Phi(\mathbf{x}))$。

在这里，我们又一次使用非监督式学习的方法帮助进行特征转换。

四. k-Means and RBF Network in Action：k均值与径向基函数网络的实践

- 左侧：完全RBF网络+一点点正则化；
- 中间：k=4；
- 右侧：最近邻模型。



需要注意的两点：

1. k-Means演算法对于，①群的数量k，②初始化点的位置，很敏感。
2. 完全RBF网络和最近邻模型实务上不常用，因为计算量太大。

五. Summary

- 本章从高斯SVM出发，展示了另一种看待高斯SVM的方式：高斯SVM是一系列radial hypotheses的线性组合，每一个radial hypothesis在衡量输入$\mathbf{x}$与自己的center$\mathbf{x}_m$的相似性(similarity)——rbf这种相似性衡量方式是基于计算输入$\mathbf{x}$与自己center间的距离(X空间内)。
- RBF网络本质上也是一系列radial hypotheses的线性聚合。回忆，对于linear aggregation，我们也可以将其看做是先把输入$\mathbf{x}$进行特征转换，然后再通过一个线性模型。同样的，RBF网络是把输入$\mathbf{x}$进行基于一系列RBF函数的特征转换，然后用某组系数β将转换后的结果线性组合起来。因此，我们有了如下的RBF hypothesis：

$$h(\mathbf{x}) = Output\left(\sum_{m=1}^{M} \beta_m RBF(\mathbf{x}, \mu_m)\right)$$

- 当M=N时，RBF网络又被称为完全RBF网络(Full RBF Network)。在完全RBF网络中，每一个训练样本点都会对输入$\mathbf{x}$的预测产生影响，影响的效力由组合系数β决定。
- 考虑所有训练样本(的相似性)的投票权相同，我们推出了"最近邻"模型与"k近邻"模型。这样的模型告诉我们，一个输入$\mathbf{x}$的预测，很大程度上取决于离它最近的一个训练样本点或几个训练样本点的情况。
- 然而，对于处理线性回归问题的完全RBF网络，在①训练样本各不相同，②RBF函数是高斯函数的情况下，其经验误差为0，因此具有过拟合的风险，需要进行正则化调整。一种正则化的方式是，对β进行正则化；另一种正则化的方式是，使$M << N$，即选出一系列好的"代表"μ_m——prototypes。后一种正则化方式则引出了k-均值演算法。
- 从一系列样本点中选出k个"代表"，实质是无监督学习中的聚类问题(clustering)。我们需要将训练资料分群，并在每个群中选择一个合适的"代表"。k-均值演算法的核心是"交替优化"：固定"代表"，最优化分群；固定分群，最优化代表；直至收敛。
- 因此，我们可以使用k-Means演算法从原始训练数据中找出k个"代表"，然后基于这k个"代表"，训练不完全的RBF网络。