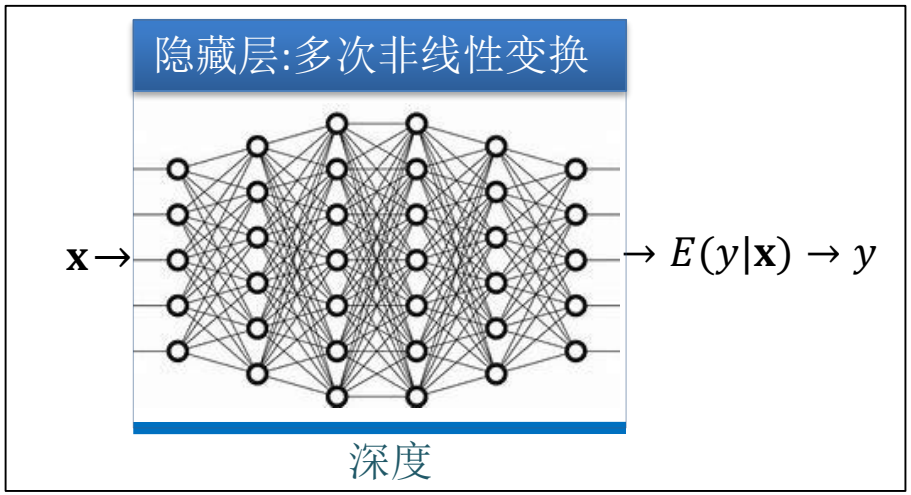
第十六讲 非线性回归

2024.1.5



预备：分段线性

多项式回归将x映射到高维，

$$x \to \mathbf{h} = (x, \cdots, x^q),$$

在q维线性空间$C(x, \cdots, x^q)$中线性组合

$$E(y|x) = a + bx + cx^2 + \cdots + dx^q = a + \boldsymbol{\beta}^{\top}\mathbf{h}$$

关于$\mathbf{h}$线性，关于x非线性。

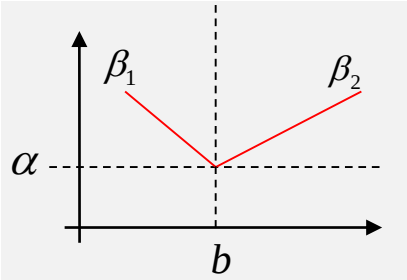
神经网络也是估计非线性回归函数，它将自变量向更高维空间m次映射，得到的$\mathbf{h}_m$与响应变量y建立线性回归模型。

两段线性

实轴上的两段线性函数（连续）：

$$y=\begin{cases}\alpha_1+\beta_1 x, & 若x\le b\\ \alpha_2+\beta_2 x, & 若x>b\end{cases}=\alpha-\beta_1(x-b)_-+\beta_2(x-b)_+$$

左端平坦$\beta_1=0$时：$y=\alpha+\beta_2(x-b)_+$



转变点b称为偏置(bias)

验证：由连续性$\alpha_1+\beta_1 b=\alpha_2+\beta_2 b$

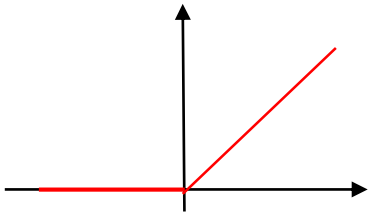
$\Rightarrow \alpha_1=\alpha_2+\beta_2 b-\beta_1 b$

$$\Rightarrow y=\begin{cases}\alpha+\beta_1(x-b), & 若x\le b\\ \alpha+\beta_2(x-b), & 若x>b\end{cases},$$

其中$\alpha=\alpha_2+\beta_2 b$,

正部函数也被称为ReLU (Rectified Linear Unit)激活函数：

$$\mathrm{ReLU}(u)=u_+=\max(0,u)=u1_{(u>0)}=\begin{cases}u & u\ge 0\\ 0 & u<0\end{cases}$$

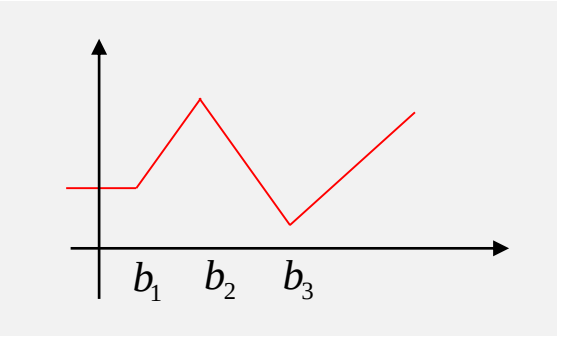


分段线性

在$b_1,\ldots,b_q$处转折的分段线性函数

$$y = f(x) = \beta_0 + \beta_1(x-b_1)_+ + \ldots + \beta_q(x-b_q)_+$$

$$= \begin{cases} \beta_0 & x < b_1 \\ \beta_0 + \beta_1(x-b_1) & b_1 \le x < b_2 \\ \beta_0 + \beta_1(x-b_1) + \beta_2(x-b_2) & b_2 \le x < b_3 \\ \ldots \end{cases}$$



将x映射/变换到高维（q维线性空间的基函数）：

$$x \in R \to \mathbf{h} = \begin{pmatrix} (x-b_1)_+ \\ \vdots \\ (x-b_q)_+ \end{pmatrix} \in R^q,$$

则分段线性函数$f(x) = \beta_0 + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{h}$是$\mathbf{h}$的线性函数，但不是$x$的线性函数。

1. DNN(p,q,m)模型结构

深度神经网络DNN(deep neural network)是深度学习的核心预测方法。我们仅考虑最基本的前馈网络。

自变量/特征: $\mathbf{x} \in R^p$ ， 响应/类别标号： $y \in R^1$

目标：估计非线性回归函数$f_\theta(\mathbf{x}) = E(y|\mathbf{x})$ ， 预测y

深度神经网络DNN

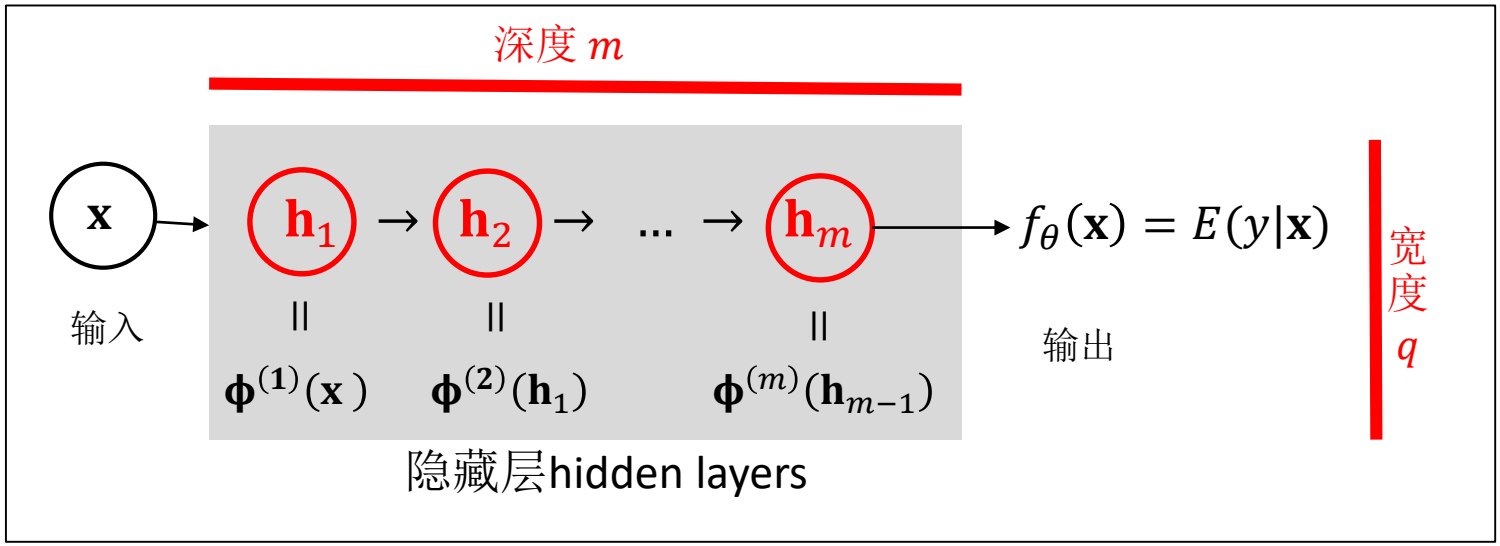
DNN(p,q,m)： p: 输入， q:宽度/神经元个数， m：深度/层数

将自变量(特征) $\mathbf{x}$向前/前馈进行m次非线性复合变换：

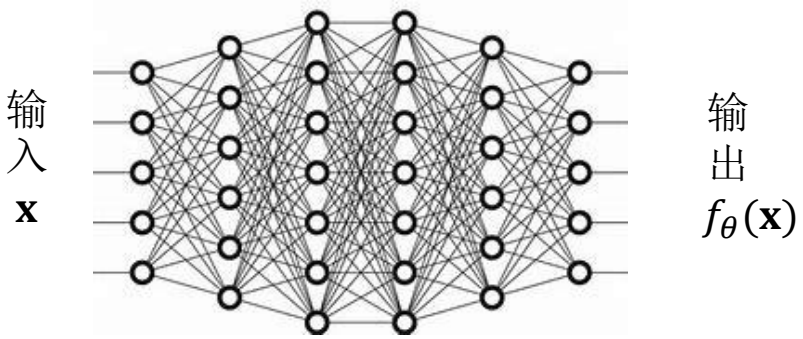
$$\mathbf{h}_m = \boldsymbol{\phi}^{(m)}\left(\boldsymbol{\phi}^{(m-1)}\left(\cdots \boldsymbol{\phi}^{(1)}(\mathbf{x})\right)\right)$$

并建立线性回归$E(y|\mathbf{x}) = f_\theta(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{h}_m$ 或其它回归模型。

DNN图示



DNN通常展示向量的分量，每个节点代表一个分量，称为神经元。



DNN = 自变量复合变换 + 线性模型（或logistic模型等）

❑ **复合**：对输入$\mathbf{x}$进行递归非线性变换：$\mathbf{h}_k = \boldsymbol{\phi}^{(k)}(\mathbf{h}_{k-1}), k = 1,2,\ldots,m,$

$$\mathbf{h}_0 = \mathbf{x},\ \mathbf{h}_m = \boldsymbol{\phi}^{(m)}\left(\boldsymbol{\phi}^{(m-1)}\left(\cdots \boldsymbol{\phi}^{(1)}(\mathbf{x})\right)\right)$$

常用仿射+ReLU变换：$\mathbf{h}_k = \boldsymbol{\phi}^{(k)}(\mathbf{h}_{k-1}) = (A_k\mathbf{h}_{k-1} + \mathbf{b}_k)_+$

A_k: 参数矩阵，$\mathbf{b}_k$:偏置参数(bias)，$(\bullet)_+$: ReLU非线性激活

参数个数约为 $O(mq^2 + pq)$.

$$\mathbf{h}_m = (A_m \cdots (A_2(A_1\mathbf{x} + \mathbf{b}_1)_+ + \mathbf{b}_2)_+ \cdots + \mathbf{b}_m)_+$$

❑ **回归**：$E(y|\mathbf{x}) = \varphi(\mathbf{h}_m) \triangleq f_\theta(\mathbf{x})$，比如

- ❖ 回归：$\varphi(\mathbf{h}_m) = \beta_0 + \boldsymbol{\beta}^\top\mathbf{h}_m \approx y$
- ❖ 分类：$\varphi(\mathbf{h}_m) = 1/(1+\exp(-\beta_0 - \boldsymbol{\beta}^T\mathbf{h}_m))$

单层神经网络与传统统计模型

响应y,自变量$\mathbf{x} \in R^p \rightarrow \mathbf{h} = (h_1,...,h_q)^{\mathsf{T}} = \Phi(\mathbf{x})$，$h_k = \phi_k(\mathbf{x}, \boldsymbol{\theta}_k)$

假设回归模型(单层神经网络).

$$E(y \mid \mathbf{x}) = \beta_0 + \sum_{k=1}^{q} \beta_k h_k, \ \mathbf{x} \in R^p$$

其中变换ϕ_k含有未知参数$\boldsymbol{\theta}_k, k = 1,...,q$,构成$q$隐藏层的神经元。

- 若$h_k = \phi_k(\mathbf{x}, \boldsymbol{\theta}_k) = \phi_k(\boldsymbol{\theta}_k{}^{\mathsf{T}}\mathbf{x})$，称为投影追踪(projection pursuit).
- 若ϕ_k取为幂次函数x^k，则为多项式回归。
- 若$\phi_k = x_k^{\lambda}$,则为线性模型中自变量Box - Cox变换。
- 若$h_k = \phi_k(\mathbf{x}, \boldsymbol{\theta}_k) = (A_k\mathbf{x} + \mathbf{b}_k)_+$，DNN最常用的激活。

它们都可以称作神经网络

2. 宽度q

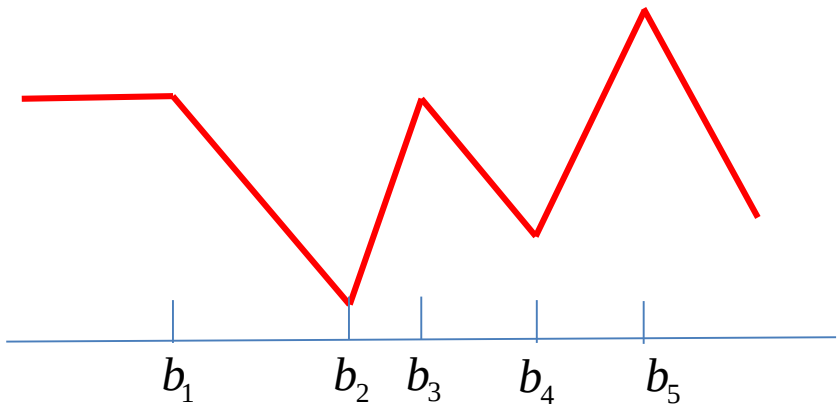
宽度即神经元的个数，也即偏置参数的个数q。

这里我们暂时不考虑一般DNN，主要考虑一元函数的非线性逼近（即$DNN(1,q,1)$： 输入为实数，单个隐藏层的情形）。

一元输入$(p=1)$情形：q个基函数$h_k=(x-b_k)_+$组合生成多段连续线性函数(转折点为偏置$b_1,...,b_q$)

$$f_\theta(x)=\beta_0+\beta_1(x-b_1)_+ +....+\beta_q(x-b_q)_+,\ x\in R^1$$

通用逼近定理证明了只要q最够大，$f_\theta(x)$可逼近任何连续函数。

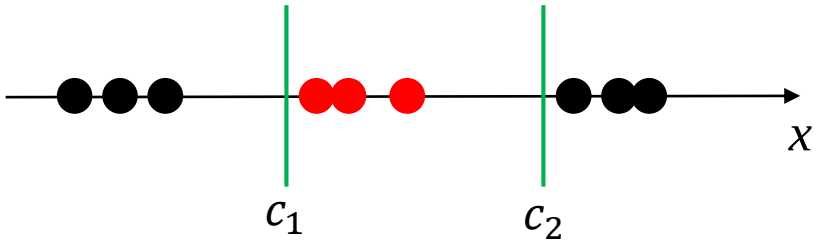


$f_\theta(x), x\in R^1$
$q=5$

线性可分

如果数轴上两类数据点（红、黑）可用一个线性函数 $x = b$分开，即由$x < b$ 和 $x > b$ 判别两类，那么只需要1个神经元（一个偏置或一个阈值），此时是线性可分的。

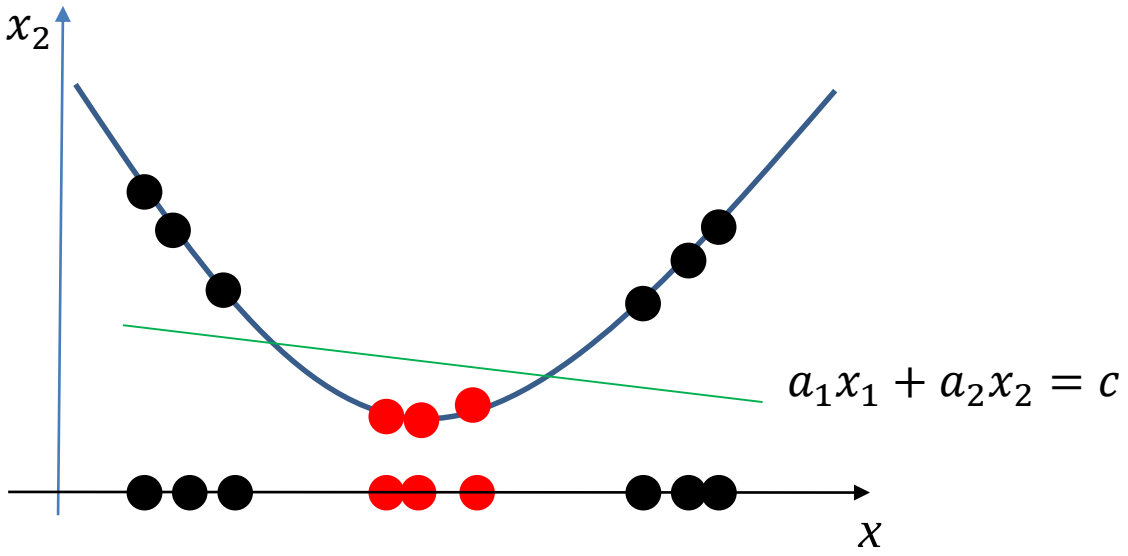
如果一维数轴上两类数据点（红、黑）不能用一个分割点分开$(x = c)$，称为是线性不可分的（下图），但可用2个分割点分开$(x = c_1, x = c_2)$。



向高维映射

将 x 映射到二维空间：

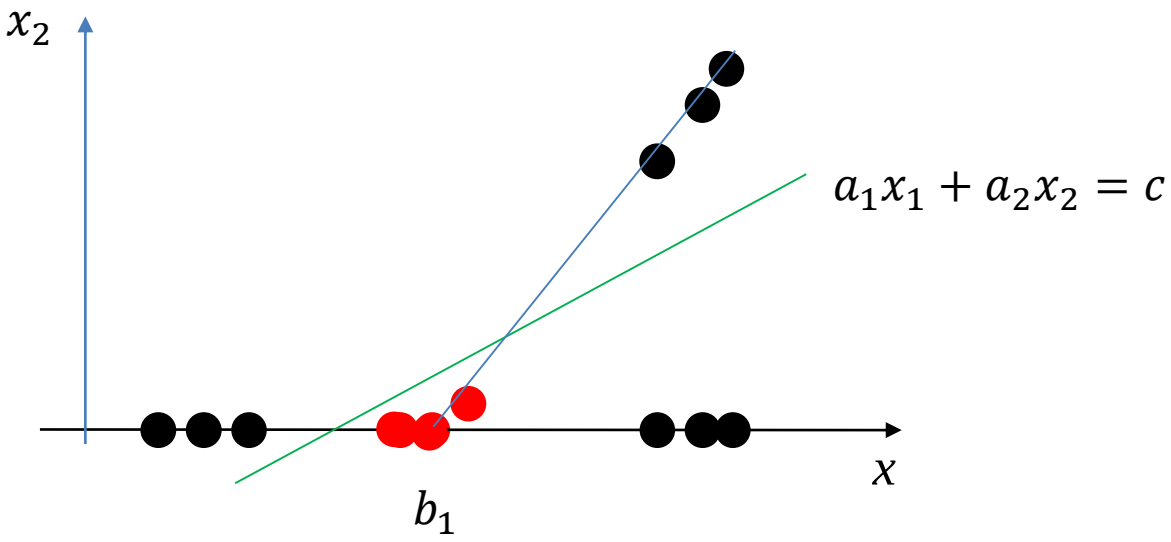
$$x \rightarrow (x_1, x_2) = (x, x^2)$$



在二维空间中直线$a_1x_1 + a_2x_2 = c$可将两类分开（线性可分）。这是支持向量机的通常做法。

ReLU有同样效果，将 x 映射到二维空间：

$$x \to (x_1, x_2) = (x, (x - b_1)_+)$$



在二维空间中直线$a_1x_1 + a_2x_2 = c$可将两类分开（线性可分）

2.2. DNN(1,q,1)：一元函数的分段线性逼近算法

过拟合

命题：假设数据为(x_i, y_i)，$x_i \in R^1, y_i \in R^1$，$i = 1,2,...,n$，

假设$x_1 < ... < x_n$，则存在$q, b_1, ..., b_q, \beta_0, \beta_1, ..., \beta_q$使得：

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 (x_i - b_1)_+ - .... - \beta_q (x_i - b_q)_+)^2 = 0$$

即完美拟合、过拟合。

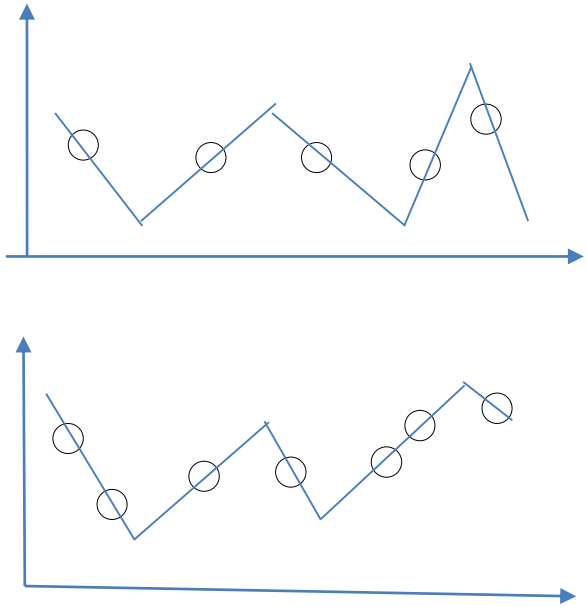
证明：取$q = n-1$, 任意取定$b's$:

$$x_1 < b_1 < x_2 < b_2 < x_3 < ... < b_{n-1} < x_n,$$

则方程组：

$$\begin{cases} y_1 = \beta_0 \\ y_2 = \beta_0 + \beta_1 (x_2 - b_1) \\ y_3 = \beta_0 + \beta_1 (x_3 - b_1) + \beta_2 (x_3 - b_2) \\ ... \\ y_n = \beta_0 + \beta_1 (x_n - b_1) + ... + \beta_{n-1} (x_n - b_{n-1}) \end{cases}$$

可解得$\beta_0 = y_1, \beta_1 = (y_2 - \beta_0)/(x_2 - b_1), ...$。注意最优解不唯一。



梯度下降法

二元数据:$(y_i, x_i), 1 \le i \le n$, 极小化误差平方和

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} (\varepsilon_i)^2 = \sum_{i=1}^{n} \left[y_i - \beta_0 - \beta_1 (x_i - b_1)_+ - \ldots - \beta_q (x_i - b_q)_+ \right]^2$$

梯度: $\nabla L = \partial L / \partial \boldsymbol{\theta} = \begin{pmatrix} \partial L / \partial \beta_0 \\ \partial L / \partial \beta_k \\ \partial L / \partial b_k \end{pmatrix} = \begin{pmatrix} -\sum \varepsilon_i \\ -\sum \varepsilon_i (x_i - b_k)_+ \\ \beta_k \sum \varepsilon_i 1_{(x_i > b_k)} \end{pmatrix}$

梯度下降法: $\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta} - \eta \nabla L(\boldsymbol{\theta})$, 学习率 $\eta \sim 1/n$

$$\beta_0^{\,new} = \beta_0 + \eta \sum \varepsilon_i$$

$$\beta_k^{\,new} = \beta_k + \eta \sum \varepsilon_i (x_i - b_k)_+$$

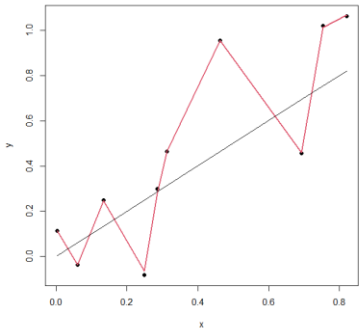
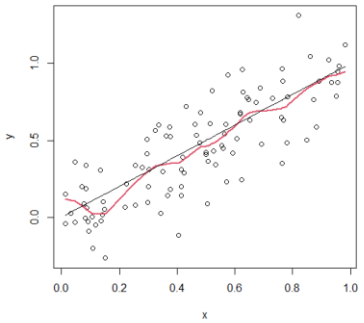
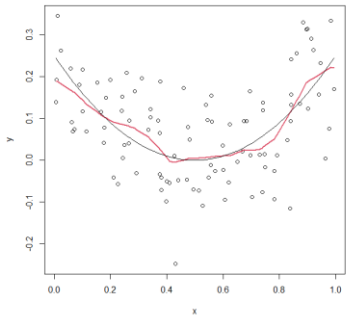
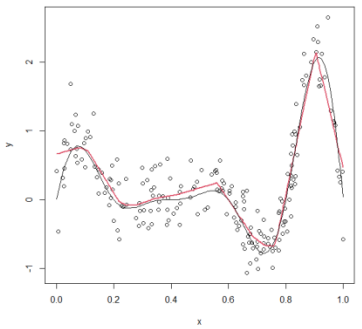
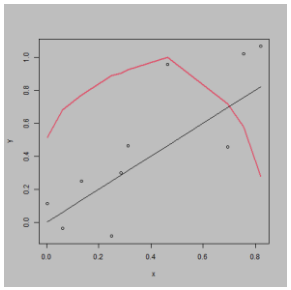
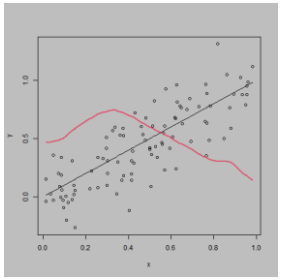
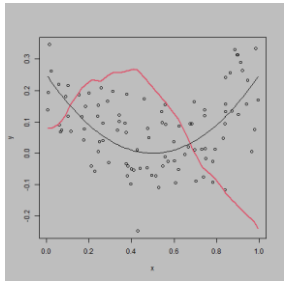
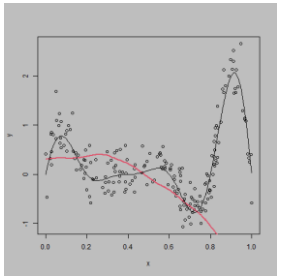
$$b_k^{\,new} = b_k - \eta \beta_k \sum \varepsilon_i 1_{(x_i > b_k)}$$

R function: nn1, neural net for 1-dim input

dnn1q1=function(x, y, q=100, eta=0.01/length(x),…)

模型$y_i = g(x_i) + \varepsilon_i, i = 1, \ldots, n$，其中$g$是真实的均值函数（下图中黑线），红线是DNN(1,q,1)梯度下降得到的拟合。

$q \geq n = 10$	$q \geq n \gg 100$		

实验观察到的现象：$(1) n < 10$时有可能过拟合 $(2) n > 10$时拟合良好，但几乎不会出现过拟合，

梯度下降算法为什么难以收敛到全局最优（完美拟合）？
简单来说，太多局部最优解（参见下面的说明）。

梯度下降收敛于$\boldsymbol{\theta}$满足$\nabla L(\boldsymbol{\theta})=0$:

$$\nabla L=\begin{pmatrix}\partial L/\partial\beta_0\\ \partial L/\partial\beta_k\\ \partial L/\partial b_k\end{pmatrix}=\begin{pmatrix}-\sum\varepsilon_i\\ -\sum\varepsilon_i x_i 1_{(x_i>b_k)}+b_k\sum\varepsilon_i 1_{(x_i>b_k)}\\ \beta_k\sum\varepsilon_i 1_{(x_i>b_k)}\end{pmatrix}=0\ \Leftrightarrow\ \begin{pmatrix}\sum\varepsilon_i\\ \sum\varepsilon_i x_i 1_{(x_i>b_k)}\\ \sum\varepsilon_i 1_{(x_i>b_k)}\end{pmatrix}=0,$$

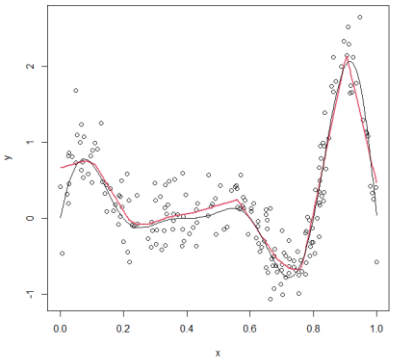
$\Leftrightarrow$ 在所有区间$(b_k,b_{k+1}]$上$(b_1\le\ldots\le b_q)$

$$\sum_{x_i\in(b_k,b_{k+1}]}\varepsilon_i=0,\qquad\sum_{x_i\in(b_k,b_{k+1}]}x_i\varepsilon_i=0$$

即在每个划分区间上等价于最小二乘法

任意给定的区间$(b_k,b_{k+1}]$划分内拟合直线回归，得到的LS估计$\hat{\beta}_k$即是局部最优点，满足$\nabla L=0$，注意其中b's任意给定即可。

但实践证明，梯度下降算法收敛到的局部最优解能捕捉到函数的主要的拐点特征（如图） 。

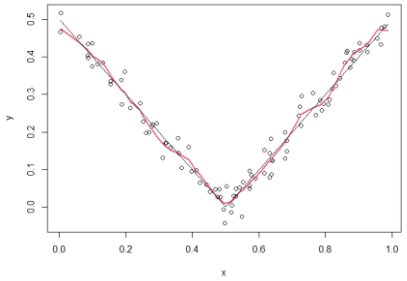


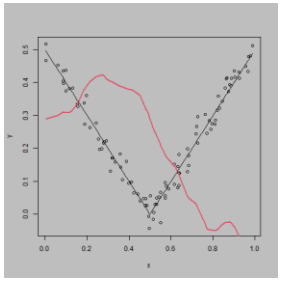
证明或否定：假设$g(x) = E(y|x)$是真正的未知回归函数（黑线），假设g具有某种光滑性，比如 $\mathrm{g} \in C(0,1)$, $\hat{f}$是$\mathrm{DNN}(1, q, 1)$梯度下降的局部最优拟合（红线），则

$$P\left(\max_{f \in C(0,1)} \left\|\hat{f} - g\right\| < \delta\right) > 1 - \varepsilon$$

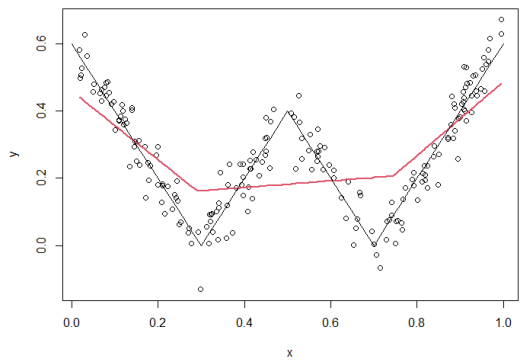
证明或否定：假设x_0是g的拐点，则一定存在某个偏置的梯度下降得到的解$\hat{b}_k$充分接近x_0

$$P\left(\max_{f \in C(0,1)} \min_{k=1,\ldots,q} \left\|\hat{b}_k - x_0\right\| < \delta\right) > 1 - \varepsilon$$

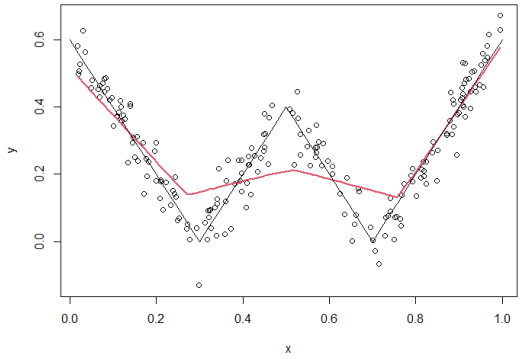




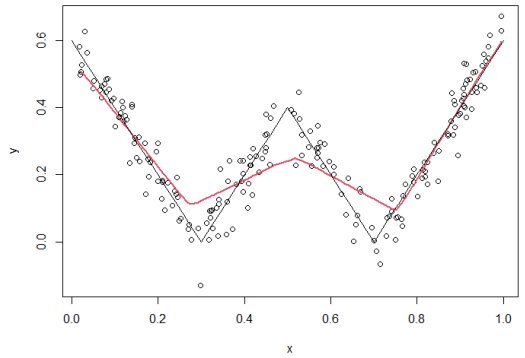
$q = 5$



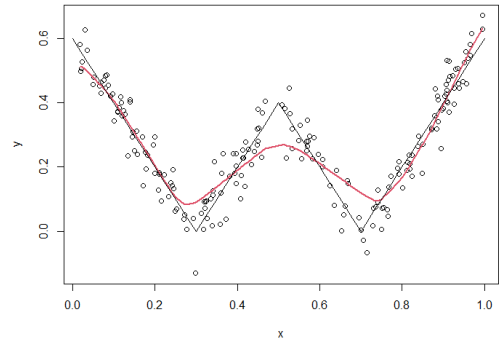
$q = 20$



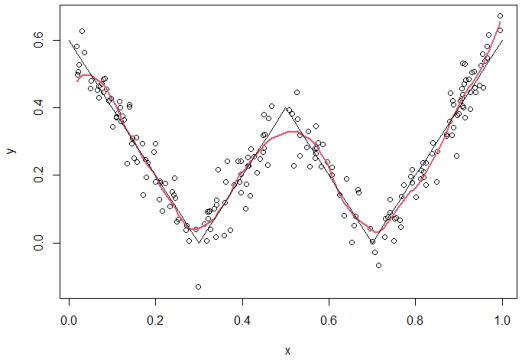
$q = 50$



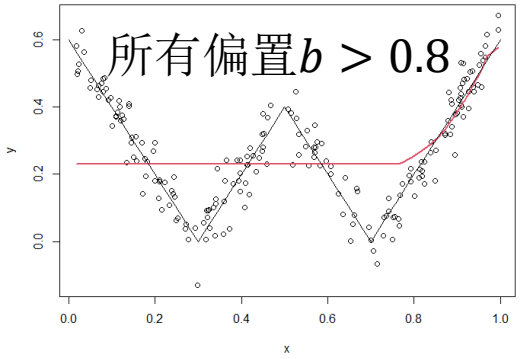
$q = 500$



$q = 2000$



$q = 4000$



q足够大时，能捕捉非线性特征（尤其是拐点），但不会过拟合。

q过大时，效果变差，分割点堆积在一起，不能有效划分实数轴。原因？

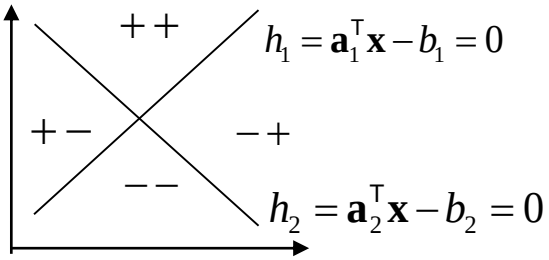
2.3 DNN(p,q,1)：p元函数的分片线性逼近

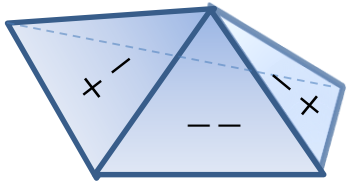
多元分段线性

例.输入$\mathbf{x}\in R^2$，$q=2$，均值函数/回归函数：

$$f(x,\boldsymbol{\theta})=\beta_0+\beta_1(\mathbf{a}_1^{\mathsf{T}}\mathbf{x}-b_1)_+ +\beta_2(\mathbf{a}_2^{\mathsf{T}}\mathbf{x}-b_2)_+$$

$$=\begin{cases}\beta_0 & \mathbf{a}_1^{\mathsf{T}}\mathbf{x}<b_1,\ \mathbf{a}_2^{\mathsf{T}}\mathbf{x}<b_2\\ \beta_0+\beta_1(\mathbf{a}_1^{\mathsf{T}}\mathbf{x}-b_1) & \mathbf{a}_1^{\mathsf{T}}\mathbf{x}\geq b_1,\ \mathbf{a}_2^{\mathsf{T}}\mathbf{x}<b_2\\ \beta_0+\beta_2(\mathbf{a}_2^{\mathsf{T}}\mathbf{x}-b_2) & \mathbf{a}_1^{\mathsf{T}}\mathbf{x}<b_1,\ \mathbf{a}_2^{\mathsf{T}}\mathbf{x}\geq b_2\\ \beta_0+\beta_1(\mathbf{a}_1^{\mathsf{T}}\mathbf{x}-b_1)+\beta_2(\mathbf{a}_2^{\mathsf{T}}\mathbf{x}-b_2) & \mathbf{a}_1^{\mathsf{T}}\mathbf{x}\geq b_1,\ \mathbf{a}_2^{\mathsf{T}}\mathbf{x}\geq b_2\end{cases}$$





二元分段函数: $q=2$ 个偏置将平面划分称4个区域，4个区域上的线性函数拼接成二元分段函数。

DNN(p,q,1)：$O(pq)$个参数，$q>p$个偏置，$O(q^p)$个划分

一般地，$\mathbf{x}\in R^p$,q个偏置：

$$f(\mathbf{x},\boldsymbol{\theta})=\beta_0+\beta_1(\mathbf{a}_1^{\mathsf{T}}\mathbf{x}-b_1)_+ +\beta_2(\mathbf{a}_2^{\mathsf{T}}\mathbf{x}-b_2)_+ +\ldots+\beta_q(\mathbf{a}_q^{\mathsf{T}}\mathbf{x}-b_q)_+$$

q个偏置（q个超平面）将空间至多划分为$r(q,p)$个块（Strang2019）：

$$r(q,p)=\binom{q}{0}+\binom{q}{1}+\ldots+\binom{q}{p}=\begin{cases}O(2^q), & q\le p\\ O(q^p), & q>p\end{cases}$$

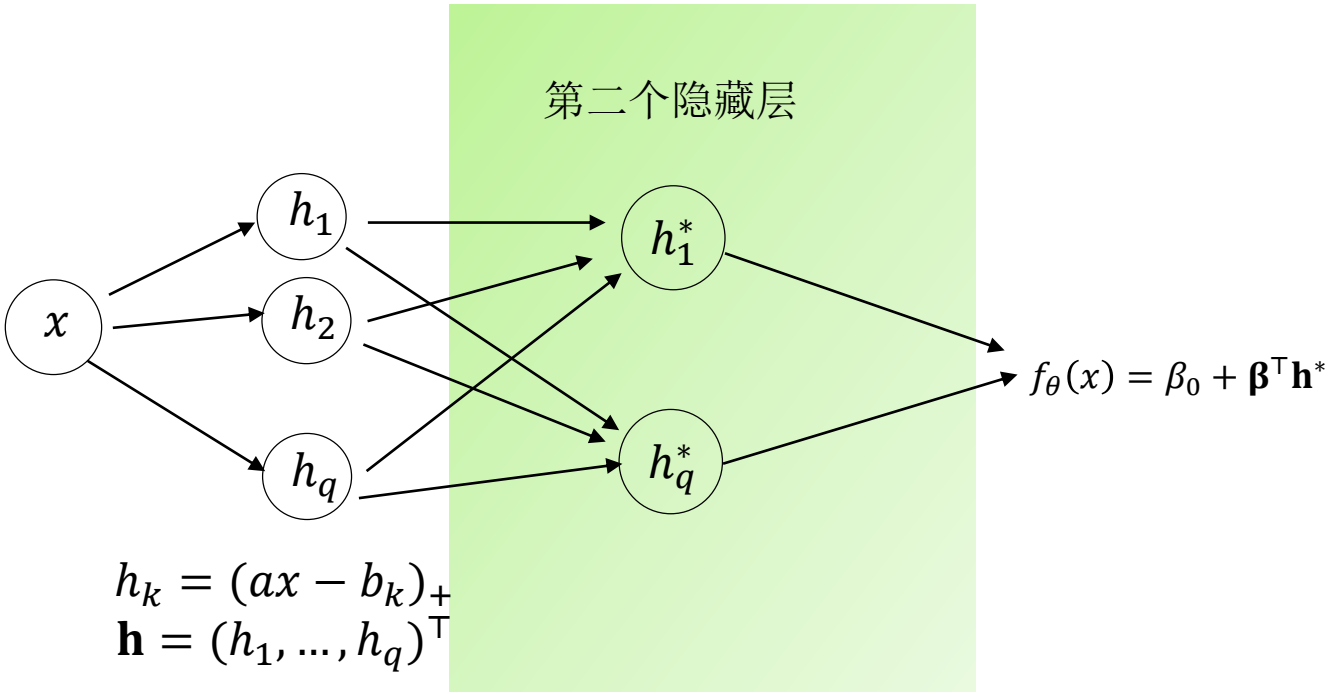
在每块上，f 线性。

记$A_{q\times p}=\begin{pmatrix}\mathbf{a}_1^{\mathsf{T}}\\ \vdots\\ \mathbf{a}_q^{\mathsf{T}}\end{pmatrix}$，$\mathbf{b}_{q\times 1}=\begin{pmatrix}b_1\\ \vdots\\ b_q\end{pmatrix}$，$\mathbf{h}_{q\times 1}=(A\mathbf{x}-\mathbf{b})_+$，$\boldsymbol{\beta}=\begin{pmatrix}\beta_1\\ \vdots\\ \beta_q\end{pmatrix}$，

所有参数$\boldsymbol{\theta}=\{A,\mathbf{b},\boldsymbol{\beta},\beta_0\}$，共$pq+2q+1$个参数。

3. 深度m

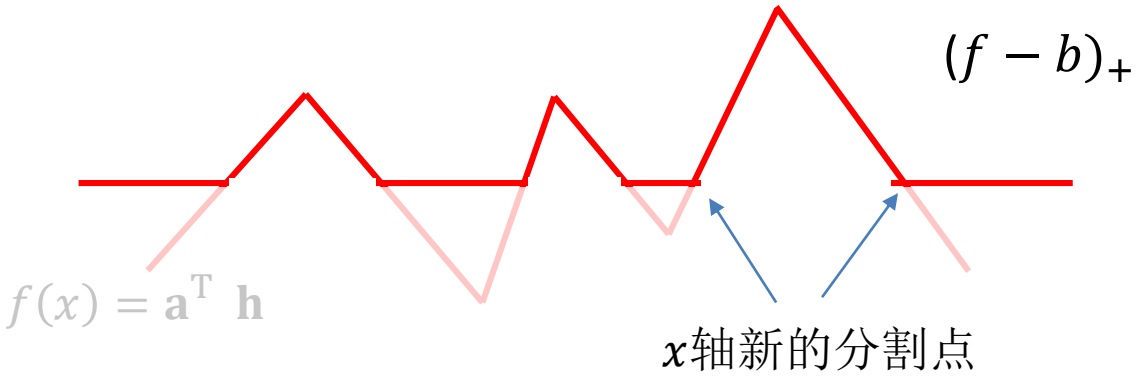
两层神经网络：对第一层的输出再次应用非线性变换，特别地，仿射+ReLU激活（仍以一元输入为例）



将第一层的输出 $\mathbf{h}$ 线性组合，偏置后激活：

$h_k^* = (\mathbf{a}_k^\top \mathbf{h} - b_k^*)_+$，$\mathbf{h}^* = (h_1^*, \ldots, h_q^*)^\top$

第一层输出$\mathbf{h}$的线性组合$f(x) = \mathbf{a}^{\mathrm{T}}\ \mathbf{h}$是一个$q$个转折点的分段线性函数，对其再应用仿射+ ReLU激活，则得到~$q/2$个新的分割点（下图）



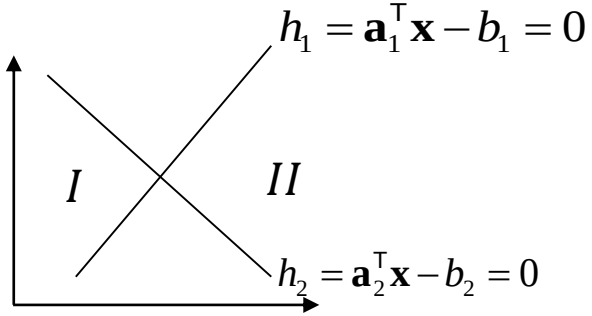
- 对第一层输出应用r次仿射+ReLU激活，大约得到$\frac{q}{2} \times r$个分割点，即两层神经网络使用$q + r$个偏置得到~$O(qr)$ 段的分段线性函数。
- 新增加的分割点在原分割点之间插补，避免了分割点堆积在一起的现象。
- 重复上述步骤迭代m层，即DNN$(p = 1, q, m)$：每层q个神经元，mq个偏置可生成$O(q^m)$段的分段线性函数。

例2. DNN($p=2, q=2,1$)的两个神经元记为(h_1, h_2)，它们将平面划分成4块。在此基础上 再次对(h_1, h_2)做同样的变换，得到第二层的2个神经元：

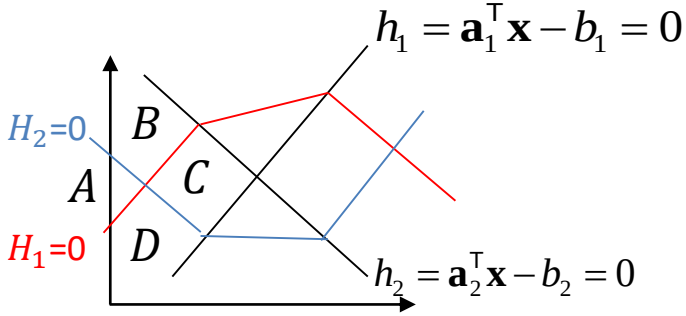
$$H_1 = (c_1 h_1 + c_2 h_2 - c)_+$$

$$H_2 = (d_1 h_1 + d_2 h_2 - d)_+$$

平面被被2个神经元H_1, H_2进一步切成8块，而且它们都是原有4块的进一步划分。类比单层$q=4$的神经网络，平面被划分为11块，所以两层网络在平面划分方面更有效率。



单层DNN(2,2,1)



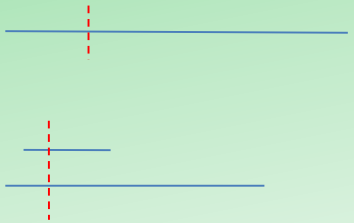
两层DNN(2,2,2)

第二层H_1和H_2将第一层得到的划分区域I又进一步划分成四个更小的区域A, B, C, D。同样区域II也被划分成4块。

深度可以高效划分、合理划分。

一般地，DNN(p,q,m)共mq个偏置，$O(mq^2+pq)$个参数，可生成$O(q^{mp})$段。

类比折叠剪绳：
一条绳子先剪q次（q个偏置），把$q+1$段折叠在一起（线性组合），再剪q次，共得到$\sim q^2$个线段 。剪mq次可产生$\sim q^m$段。



偏置产生分割划分，较多的分割保证分段函数形式足够丰富，而梯度下降法阻止过拟合。

但一层内过多的偏置（过大的q）导致得到的偏置解拥挤在一起，不能有效划分输入空间。增加层数/深度，可以用较少的偏置参数得到更多的划分，而且是以插补方式增加偏置，防止偏置解拥挤（平凡解）。

梯度下降法

损失函数

$$L(\boldsymbol{\theta}) = \sum(y_i - f(\mathbf{x}_i, \boldsymbol{\theta}))^2$$
$$= \sum(y_i - [\beta_0 + \boldsymbol{\beta}^\top \boldsymbol{\phi}^{(m)}\left(\boldsymbol{\phi}^{(m-1)}\left(\cdots \boldsymbol{\phi}^{(1)}(\mathbf{x})\right)\right)])^2$$

梯度下降解法：$\boldsymbol{\theta}^{\text{new}} = \boldsymbol{\theta}^{\text{old}} - \eta \nabla L(\boldsymbol{\theta}^{\text{old}})$，$\eta$：学习率

学习率以及迭代停止阈值的选取都颇为关键

梯度下降法包含f的函数值和梯度

- 前馈计算f函数值：从输入开始逐层计算函数值计算

$$f(\mathbf{x}_i, \boldsymbol{\theta}) = \beta_0 + \boldsymbol{\beta}^\top \boldsymbol{\phi}^{(m)}\left(\boldsymbol{\phi}^{(m-1)}\left(\cdots \boldsymbol{\phi}^{(1)}(\mathbf{x})\right)\right)$$
$$\mathbf{x} \to \mathbf{h}_1 = \boldsymbol{\phi}^{(1)}(\mathbf{x}) \to \mathbf{h}_2 = \boldsymbol{\phi}^{(2)}(\mathbf{h}_1)$$

- 反向传播法计算梯度∇f：从最后一层开始反向计算梯度（下页）

$$\frac{\partial f}{\partial \boldsymbol{\theta}_k} = \frac{\partial f}{\partial \mathbf{h}_m}\frac{\partial \mathbf{h}_m}{\partial \mathbf{h}_{m-1}} \cdots \frac{\partial \mathbf{h}_{k+1}}{\partial \mathbf{h}_k}\frac{\partial \mathbf{h}_k}{\partial \boldsymbol{\theta}_k}$$

反向传播

根据链式法则，复合函数的求导是从外到里（ANN反箭头方向）

$$\frac{\partial f}{\partial \beta_0}=1,\quad \frac{\partial f}{\partial \boldsymbol{\beta}}=h_m$$

$$\frac{\partial f}{\partial \boldsymbol{\theta}_m}=\left[\frac{\partial f}{\partial h_m}\right]\frac{\partial h_m}{\partial \boldsymbol{\theta}_m}$$

$$\frac{\partial f}{\partial \boldsymbol{\theta}_{m-1}}=\left[\left(\frac{\partial f}{\partial h_m}\right)\frac{\partial h_m}{\partial h_{m-1}}\right]\frac{\partial h_{m-1}}{\partial \boldsymbol{\theta}_{m-1}}$$

$$\frac{\partial f}{\partial \boldsymbol{\theta}_{m-2}}=\left[\left(\frac{\partial f}{\partial h_m}\frac{\partial h_m}{\partial h_{m-1}}\right)\frac{\partial h_{m-1}}{\partial h_{m-2}}\right]\frac{\partial h_{m-2}}{\partial \boldsymbol{\theta}_{m-2}}$$

$$\frac{\partial f}{\partial \boldsymbol{\theta}_{m-3}}=\left[\left(\underset{1\times q_m}{\frac{\partial f}{\partial h_m}}\underset{q_m\times q_{m-1}}{\frac{\partial h_m}{\partial h_{m-1}}}\underset{q_{m-1}\times q_{m-2}}{\frac{\partial h_{m-1}}{\partial h_{m-2}}}\right)\underset{q_{m-2}\times q_{m-3}}{\frac{\partial h_{m-2}}{\partial h_{m-3}}}\right]\frac{\partial h_{m-3}}{\partial \boldsymbol{\theta}_{m-3}},$$

.....

$\frac{\partial f}{\partial h_m}=\boldsymbol{\beta}^{\mathsf{T}}$ 为$1\times q_m$向量，其中$q_m=|h_m|$

$\left(\frac{\partial f}{\partial h_m}\right)\frac{\partial h_m}{\partial h_{m-1}}$ 为$1\times q_m$向量与$q_m\times q_{m-1}$矩阵的乘积，结果为$1\times q_{m-1}$向量，在下一步不必重新计算

[]中的乘积是一个向量，在下一个梯度中它出现在()中，不需要重新计算。BP方法计算梯度从后向前的次序计算，每次记录[]中的向量，下一步只需计算该向量与一个矩阵的乘积

参考：G.Strang (2019) Linear algebra and learning from data (Part VII).
/books/8.pdf

如果希望深入了解DNN，特别是深度和梯度下降算法，

最好的途径是：

自己写一个一元 DNN(1,q,m)函数！ 而不是读文献

这里分享一下我个人了解深度学习的过程，希望对你学习深度学习有借鉴作用:

1. 多数入门材料比较空泛或抽象，比如大多使用一般激活函数而不是具体的、容易理解的ReLU，不易深入理解。直到看到Strang书，才对深度学习有了一点认识，该书聚焦于“DNN=仿射+ReLU复合变换”的表述：

   $$\mathbf{h}_m = (A_m \cdots (A_2(A_1\mathbf{x} + \mathbf{b}_1)_+ + \mathbf{b}_2)_+ \cdots + \mathbf{b}_m)_+, \quad \varphi(\mathbf{h}_m) = \beta_0 + \boldsymbol{\beta}^\top \mathbf{h}_m \to \mathrm{E}(y|\mathbf{x}) \to y$$

   这为理解DNN提供了有力的数学工具，但该书没有提供更多直观的理解（缺少实验）。

2. 考虑单层DNN情形：$p = 1, m = 1$，此时$\varphi(\mathbf{h}_m)$作为x的函数是分段线性函数；偏置作为转折点，其重要性在分段线性函数图像中可以看到。

3. 自行编写DNN(1,q,1)程序， 实验，了解到只要偏置参数个数 q 和斜率参数beta足够多，理论上能完美拟合数据点（全局最优、过拟合），但实验结果表明梯度下降法几乎一般都收敛到局部最优（从而阻止过拟合），这或许说明收敛到局部最优是梯度下降算法和模型的优势（换言之，大模型中不应该追求收敛到全局最优）。但还不清楚为什么算法收敛到的局部最优解一般不是那些退化的平凡解，一般能较好地拟合数据，特别是能够正确捕捉到转折点（导数为0点）。

4. 基本上， q越大，拟合越好（但不过拟合），但当 q 非常大（参数过多）的时候，拟合效果又会变差，转折点/偏置估计会拥挤在自变量区间比如[0,1]区间的边界附近，甚至可能跑出区间之外，得到退化、平凡解。

5. 对单层网络的输出再次应用仿射+ReLU变换，可达到“使用较少参数得到更多划分”的效果，而且它是在第一层划分的基础上继续分割，可能不太会出现退化/拥挤现象。

6. 因为没有编写DNN(1,q,m)程序，故本次课件对深度引起的问题和优势没有更多具体的认识。所以，如果你希望对DNN有更好的理解，建议自行编写DNN(1,q,m)函数做实验。

4. 深度学习

深度学习(deep learning)方法是基于DNN的机器学习方法。 深度学习之父G.Hinton(2006)应用反向传播法解决了多层深度网络模型的可计算性问题。此后，深度学习普及应用于多种场景的不同网络结构的学习任务。

主要模型结构

基本模型架构：深度神经网络（前馈/循环/双向）。
各种应用的不同在于输入不同，以及针对特殊场景的模型结构的调整。

主要应用领域

图像识别（CV: computer vision）
语言处理（NLP: natural language processing）
语音识别（ASR： automatic speech recognition）
最新成就：alphafold2、chatGPT

深度学习框架

Pytorch, Tensorflow, keras,…

考试时间：1月14日上午8:30-10:30

- 复习范围（排除下表中的阴影部分）：课件+作业
- 考试内容、类型和难度：与作业类似。

  Lab不考，但会有与R输出有关的计算或解释（参见hw7.6）。

  带计算器，但计算不会过于复杂(矩阵求逆不超过二阶)。

课件	阅读材料	作业	上机
第一讲：回归分析简介 (9.8)	Freedman:第一章	hw1 参考答案	
第二讲：相关系数 (9.15)		hw2 参考答案	
第三讲：偏相关系数 (9.22)	Freedman《Statistics》1-2章 2、3、4讲讲义	hw3 参考答案	
第四讲：多元正态 (10.8)		hw4 参考答案	lab1
第五讲：线性回归模型 (10.13)		hw5 参考答案	
第六讲：简单线性回归模型 (10.20)		hw6 参考答案	
	对称回归		
第七讲：幂次律 (11.3)	6、7讲讲义	hw7 参考答案	
第八讲：投影 (11.10)			lab2
第九讲：欧氏空间的投影 (11.17)	8-9讲投影讲义	hw8 参考答案	
第十讲：最小二乘 (11.24)		hw9 参考答案	
第十一讲：最小二乘II (12.1)		hw10 参考答案	
第十二讲：F检验 (12.8)		hw11	
第十三讲：回归诊断 (12.15)			lab3
第十四讲：广义最小二乘 (12.22)			lab4
第十五讲：预测 (12.29)		hw12	

排除课程主页列表的阴影部分：

- 阅读材料
- lab
- 课件：第8讲(代数)、第9讲P1-21(矩阵预备知识)、第12讲P32-40(两因素anova，置信)、第14讲P9-12(IRLS)、第15讲（AIC/BIC, 贪心算法，lasso），以及课件中虚线框内容和附录。