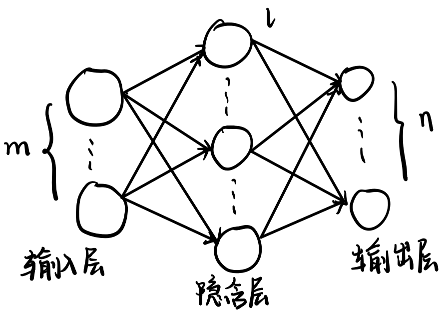
**5.1** 试述将线性函数 $f(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$ 用作神经元激活函数的缺陷.

如果激活函数是一个线性函数，那么无论多少层网络，都可以表示为一层线性网络.

比如一个2层神经网络，一个隐含层和一个输出层，设输入为 $x_i$




隐含层和输出层的激活函数都为线性函数 $f(x)=x$

则隐含层输出 $\hat{O}_j^1 = f(\sum_{i=1}^{m} W_{ij}^1 x_i + b_j^1)$

输出层输出 $\hat{O}_j^2 = f(\sum_{i=1}^{l} W_{ij}^2 \hat{O}_j^1 + b_j^2)$

那么此时可只用1层激活函数 $f(x)=x$ 将上面2层神经网络的输出表达出来

理想中的激活函数是阶跃函数，但阶跃函数非连续，在0处不可导

在0周围变化急剧的 sigmoid 函数满足需要.

线性激活函数无法完全模拟阶跃函数且线性函数在定义域内变换情况相同.

---

- 讨论 $\frac{\exp(x_i)}{\sum_{j=1}^{C}\exp(x_j)}$ 和 $\log\sum_{j=1}^{C}\exp(x_j)$ 的数值溢出问题

当 $x$ 很大时，$\exp(x)$ 的结果可能发生溢出而显示 NaN

可以设 $x_j$ $(1\le j\le C)$ 的最大值为 $x^*$，可以如下处理

$$\frac{\exp(x_i)}{\sum_{j=1}^{C}\exp(x_j)} = \frac{\exp(x_i)\exp(-x^*)}{\exp(-x^*)\sum_{j=1}^{C}\exp(x_j)} = \frac{\exp(x_i-x^*)}{\sum_{j=1}^{C}\exp(x_j-x^*)}$$

这样可以避免数值上溢问题

同样地，对于第二个函数，可作如下处理：

$$\log\sum_{j=1}^{C}\exp(x_j) = \log\left[\exp(x^*)\sum_{j=1}^{C}\exp(x_j-x^*)\right]$$

$$= x^* + \log\sum_{j=1}^{C}\exp(x_j-x^*)$$

- 计算 $\frac{\exp(x_i)}{\sum_{j=1}^{C}\exp(x_j)}$ 和 $\log\frac{\exp(x_i)}{\sum_{j=1}^{C}\exp(x_j)}$ 关于向量 $\boldsymbol{x}=[x_1,\cdots,x_C]$ 的梯度

令 $f(x)=\frac{e^{x_i}}{\sum_{j=1}^{C}e^{x_j}}$，则

若 $k\neq i$，$\frac{\partial f(x)}{\partial x_k}=\frac{-e^{x_i}\cdot e^{x_k}}{\left(\sum_{j=1}^{C}e^{x_j}\right)^2}=\frac{-e^{(x_i+x_k)}}{\left(\sum_{j=1}^{C}e^{x_j}\right)^2}$

若 $k=i$，$\frac{\partial f(x)}{\partial x_k}=\frac{e^{x_k}\left(\sum_{j=1}^{C}e^{x_j}-e^{x_k}\right)}{\left(\sum_{j=1}^{C}e^{x_j}\right)^2}=\frac{e^{x_k}\left(\sum_{j=1,j\neq k}^{C}e^{x_j}\right)}{\left(\sum_{j=1}^{C}e^{x_j}\right)^2}$

即
$$\frac{\partial f(x)}{\partial x_k}=\begin{cases}\frac{-e^{(x_i+x_k)}}{\left(\sum_{j=1}^{C}e^{x_j}\right)^2}, & k\neq i\\ \frac{e^{x_k}\left(\sum_{j=1,j\neq k}^{C}e^{x_j}\right)}{\left(\sum_{j=1}^{C}e^{x_j}\right)^2}, & k=i\end{cases}$$

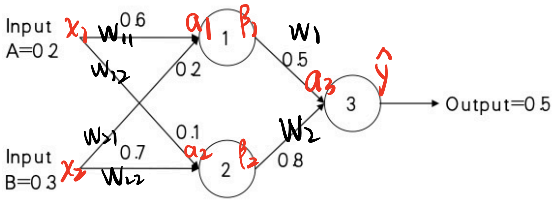
同理令 $g(x)=\log\frac{e^{x_i}}{\sum_{j=1}^{C}e^{x_j}}$，则

若 $k\neq i$，$\frac{\partial g(x)}{\partial x_k}=\frac{\sum_{j=1}^{C}e^{x_j}}{e^{x_i}}\cdot\frac{-e^{(x_i+x_k)}}{\left(\sum_{j=1}^{C}e^{x_j}\right)^2}=-\frac{e^{x_k}}{\sum_{j=1}^{C}e^{x_j}}$

若 $k=i$，$\frac{\partial g(x)}{\partial x_k}=\frac{\sum_{j=1}^{C}e^{x_j}}{e^{x_k}}\cdot\frac{e^{x_k}\left(\sum_{j=1,j\neq k}^{C}e^{x_j}\right)}{\left(\sum_{j=1}^{C}e^{x_j}\right)^2}=1-\frac{e^{x_k}}{\sum_{j=1}^{C}e^{x_j}}$

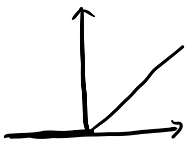
即
$$\frac{\partial g(x)}{\partial x_k}=\begin{cases}-\frac{e^{x_k}}{\sum_{j=1}^{C}e^{x_j}}, & k\neq i\\ 1-\frac{e^{x_k}}{\sum_{j=1}^{C}e^{x_j}}, & k=i\end{cases}$$

- 考虑如下简单网络，假设激活函数为ReLU，用平方损失$\frac{1}{2}(y-\hat{y})^2$计算误差，请用BP算法更新一次所有参数（学习率为1），给出更新后的参数值（给出详细计算过程），并计算给定输入值x=(0.2,0.3)时初始时和更新后的输出值，检查参数更新是否降低了平方损失值.




激活函数 ReLU = max(0, x)

$RELU'(x) = I(x>0)$



$E_k = \frac{1}{2}\sum_{j=1}^{2}(\hat{y}_j^k - y_j^k)^2$ , $\eta = 1$

该网络各参数为：$W_{11} = 0.6$ , $W_{12} = 0.1$ , $W_{21} = 0.2$ , $W_{22} = 0.7$ , $W_1 = 0.5$ , $W_2 = 0.8$

$a_1 = 0.2 \times 0.6 + 0.3 \times 0.2 = 0.18 \xrightarrow{ReLU} \beta_1 = ReLU(a_1) = 0.18$

$a_2 = 0.2 \times 0.1 + 0.3 \times 0.7 = 0.23 \xrightarrow{ReLU} \beta_2 = ReLU(a_2) = 0.23$

$a_3 = 0.18 \times 0.5 + 0.23 \times 0.8 = 0.274 \xrightarrow{ReLU} \hat{y} = ReLU(a_3) = 0.274$

Error： $E = \frac{1}{2}(y-\hat{y})^2 = 0.0256$

计算梯度项：

$$\frac{\partial E}{\partial w_{11}} = -(y-\hat{y})\,ReLU'(a_3)\cdot w_1 \cdot ReLU'(a_1)\,x_1 = -0.0226$$

$$\frac{\partial E}{\partial w_{12}} = -(y-\hat{y})\,ReLU'(a_3)\cdot w_1 \cdot ReLU'(a_2)\,x_1 = -0.0226$$

$$\frac{\partial E}{\partial w_{21}} = -(y-\hat{y})\,ReLU'(a_3)\,w_2 \cdot ReLU'(a_1)\,x_2 = -0.0542$$

$$\frac{\partial E}{\partial w_{22}} = -(y-\hat{y})\,ReLU'(a_3)\cdot w_2 \cdot ReLU'(a_2)\,x_2 = -0.0542$$

$$\frac{\partial E}{\partial w_1} = -(y-\hat{y})\,ReLU'(a_3)\,\beta_1 = -0.0407$$

$$\frac{\partial E}{\partial w_2} = -(y-\hat{y})\,ReLU'(a_3)\,\beta_2 = -0.0520$$

由于学习率为 $\eta=1$，更新参数：

$$w_{11} \leftarrow w_{11} - \eta\frac{\partial E}{\partial w_{11}} = 0.6226$$

$$w_{12} \leftarrow w_{12} - \eta\frac{\partial E}{\partial w_{12}} = 0.1226$$

$$w_{1} \leftarrow w_{1} - \eta\frac{\partial E}{\partial w_{1}} = 0.5407$$

$$a_1 = w_{11}x_1 + w_{21}x_2 = 0.2008$$

$$a_2 = w_{12}x_1 + w_{22}x_2 = 0.2508$$

$$a_3 = w_1\beta_1 + w_2\beta_2 = 0.301$$

$$w_{21} \leftarrow w_{21} - \eta\frac{\partial E}{\partial w_{21}} = 0.2542$$

$$w_{22} \leftarrow w_{22} - \eta\frac{\partial E}{\partial w_{22}} = 0.7542$$

$$w_{2} = w_{2} - \eta\frac{\partial E}{\partial w_{2}} \quad 0.8520$$

$$\beta_1 = \mathrm{ReLU}(a_1) = 0.2008$$

$$\beta_2 = \mathrm{ReLU}(a_2) = 0.2508$$

$$\hat{y} = \mathrm{ReLU}(a_3) = 0.301$$

$$E = \frac{1}{2}(y-\hat{y})^2 = 0.0198 < 0.0256$$ 平方损失下降了

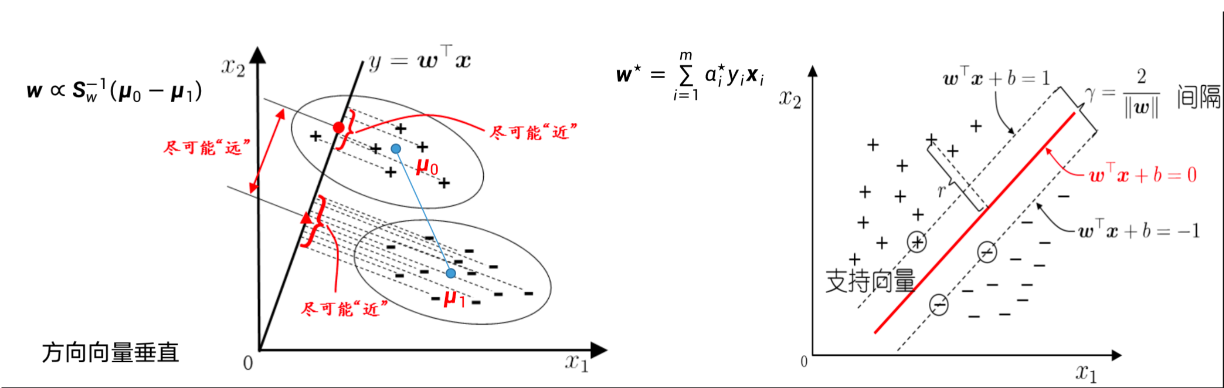
---

**6.4** 试讨论线性判别分析与线性核支持向量机在何种条件下等价.

线性判别分析能够解决多分类问题，而SVM只能解决二分类问题

线性判别分析能将数据以同类样例间低方差，不同样例中心之间大间隔来投射到一条直线上，但是如果样本线性不可分，那么线性判别分析就不能有效进行，支持向量机也是.

而当两类样本线性可分时，且处理二分类问题时等价.



**6.6** 试析 SVM 对噪声敏感的原因.

SVM的决策只依赖于少量的支持向量，若噪声样本出现在支持向量中，容易对决策造成影响，所以SVM对噪声敏感.

---

**6.9** 试使用核技巧推广对率回归, 产生“核对率回归”.

核对率回归模型： $l(\beta) = \sum_{i=1}^{m}\left(-y_i \beta^T \hat{x}_i + \log(1+e^{\beta^T \hat{x}_i})\right)$

SVM模型： $\min_{w,b} \frac{1}{2}\|w\|^2 \quad s.t. \quad y_i(w^T x_i + b) \geq 1, \quad i=1,\cdots,m$

软间支持向量机： $\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} l_{0/1}\left(y_i(w^T\phi(x_i)+b)-1\right)$

令 $y_i(w^T\phi(x_i)+b) = z$

使用对率损失： $l_{log}(z) = \log(1+e^{-z}) = \log\left(\frac{1+e^{z}}{e^{z}}\right) = \log(1+e^{z}) - z$

则可改写为： $\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\left(-z + \log(1+e^{z})\right)$

$$h(x) = w^T\phi(x_i) = \sum_{i=1}^{m} a_i K(x, x_i)$$

支持向量回归的对偶问题如下，

$$\max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}} g(\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\alpha_i-\hat{\alpha}_i)(\alpha_j-\hat{\alpha}_j)\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)+\sum_{i=1}^{m}(y_i(\hat{\alpha}_i-\alpha_i)-\epsilon(\hat{\alpha}_i+\alpha_i))$$

$$s.t.\ C \geqslant \boldsymbol{\alpha},\ \hat{\boldsymbol{\alpha}} \geqslant 0 \text{ and } \textstyle\sum_{i=1}^{m}(\alpha_i-\hat{\alpha}_i)=0$$

请将该问题转化为类似于如下标准型的形式（$\boldsymbol{u},\boldsymbol{v},\boldsymbol{K}$均已知），

$$\max_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^\top\boldsymbol{v}-\frac{1}{2}\boldsymbol{\alpha}^\top\boldsymbol{K}\boldsymbol{\alpha}$$

$$s.t.\ C \geqslant \boldsymbol{\alpha} \geqslant 0 \text{ and } \boldsymbol{\alpha}^\top\boldsymbol{u}=0$$

例如在软间隔SVM中$\boldsymbol{v}=\boldsymbol{1}$, $\boldsymbol{u}=\boldsymbol{y}$, $\boldsymbol{K}[i,j]=y_iy_j\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)$.

令 $a^* = \begin{pmatrix} a \\ \hat{a} \end{pmatrix}$，则

$$\sum_{i=1}^{m}\sum_{j=1}^{m}(a_i-\hat{a}_i)(a_j-\hat{a}_j)K(x_i,x_j)$$

$$=\sum_{i=1}^{m}\sum_{j=1}^{m} a_ia_jk_{ij}-\hat{a}_ia_jk_{ij}-a_i\hat{a}_jk_{ij}+\hat{a}_i\hat{a}_jk_{ij}$$

$$=(a^*)^T\begin{bmatrix} k & -k \\ -k & k \end{bmatrix}a^*$$

令 $v=\begin{pmatrix} -y-\varepsilon \\ y-\varepsilon \end{pmatrix}$，则有 $\sum_{i=1}^{m}(y_i(\hat{a}_i-a_i)-\varepsilon(\hat{a}_i+a_i))=(a^*)^Tv$

因此原式可化简为：

$$\max_{a^*} g(a^*) = (a^*)^Tv-\frac{1}{2}(a^*)^TKa^*$$

$$s.t.\ C \geqslant a^* \geqslant 0,\ (a^*)^Tv=0$$

这里 $K=\begin{bmatrix} k & -k \\ -k & k \end{bmatrix}$