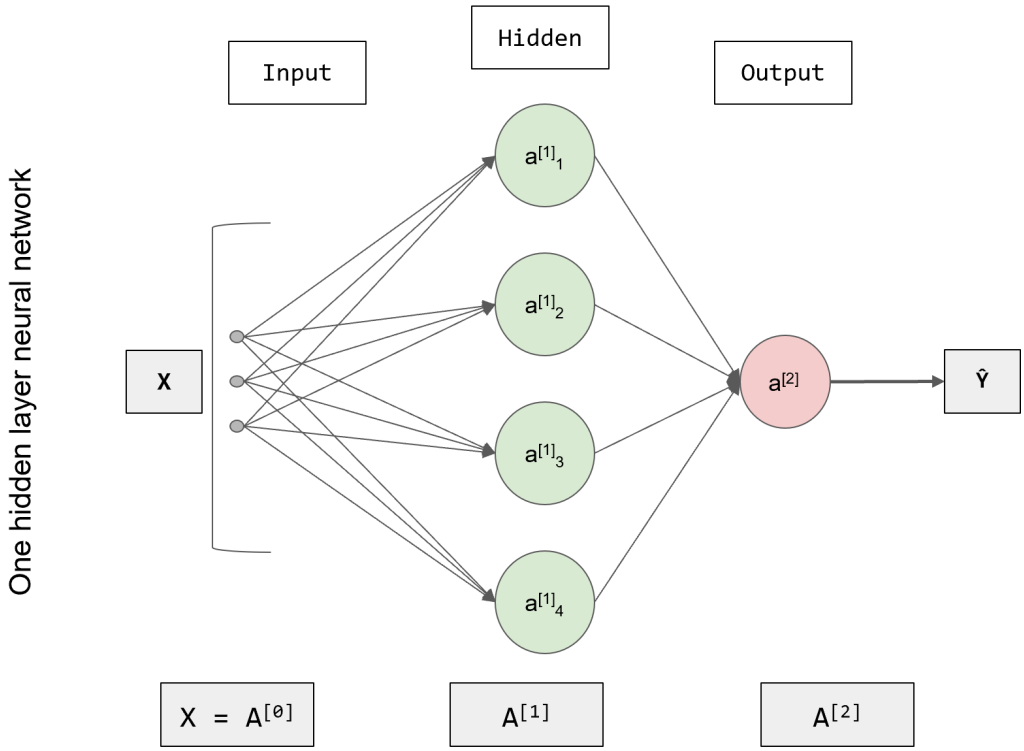
神经网络与深度学习笔记（六）L_2正则化

前言



前面提到过高方差问题主要的两种方式：

- 获取更多的数据去训练。然而这种方式局限在于，数据并不是总是很容易获得的或者数据获取的代价很大。
- 正则化。这就是这篇文章需要来讨论的主题。

最小化代价函数正则化

使用 L_2 正则化的最小化代价函数：

$$min_{(w,b)}\jmath(w,b)=\frac{1}{m}\sum_{i=1}^{m}\jmath(\hat{y}^{(i)},y^{(i)})+\frac{\lambda}{2m}\|w\|_2^2 \tag{1}$$

其中，λ 称为正则化参数。在编程过程中，常常把 λ 写为`lambd`，以防止和`Python`语言中的关键字`lamba`起冲突。这一参数通常需要调优，使用开发集或者交叉验证集来验证。

那么：

$$\|w\|_2^2 \tag{2}$$

到底是啥意思？

实际上，$\|w\|_2^2$ 被称为 L_2 正则化，又称为参数 w 的欧几里得范数，L_2 范数等

$$\|w\|_2^2=\sum_{j=1}^{n_x}w_j^2=w^Tw \tag{3}$$

既然有 L_2 范数，那么 L_1 范数是啥？

L_1 范数即为：

$$\|w\|_1 = |w_1|+|w_2|+|w_3|+\cdots|w_n| \tag{4}$$

与此对应的 L_1 正则化

$$\sum_{j=1}^{n_x}|w_j|= \|w\|_1 \tag{5}$$

使用 L_1 正则化的最小代价函数为：

$$min_{(w,b)}\jmath(w,b) = \frac{1}{m}\sum_{i=1}^{m}\jmath(\hat{y}^{(i)},y^{(i)}) + \frac{\lambda}{2m}\|w\|_1 \tag{6}$$

然而使用 L_1 正则化的效果并不是太明显，主要是因为使用后会导致 w 稀疏，w 矢量中会有很多 0，虽然模型会有一定的压缩但是效果不大。

这就是为什么通常使用 L_2 正则化而不是 L_1 正则化的原因。

那么， b 参数是否可以正则化呢？比如： $\frac{\lambda}{2m}b^2$

答案也是效果不大。

因为参数实际上大多数集中在 w 中，而不是 b ，即使对 b 进行了正则化，b 对模型的影响效果也不是太大

在神经网络中的 L_2 正则化

$$\jmath(w^{[1]},b^{[1]},\cdots w^{[l]},b^{[l]}) = \frac{1}{m}\sum_{i=1}^{m}L(\hat{y}^{(i)},y^{(i)}) + \frac{\lambda}{2m}\sum_{i=1}^{l}\|w^{[l]}\|^2 \tag{7}$$

$$\|w^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l]}}\sum_{j=1}^{n^{[l-1]}}(w_{ij}^{[l]})^2 \tag{8}$$

$$w^{[l]}:(n^{[l]},n^{[l-1]}) \tag{9}$$

其中矩阵的 $\|w^{[l]}\|_F^2 = \sum_{i=1}^{n^{[l]}}\sum_{j=1}^{n^{[l-1]}}(w_{ij}^{[l]})^2$ ，表示矩阵中元素平方和。又称为弗罗贝尼乌斯范数（Frobenius norm），这里就不叫 L_2 范数了。

在梯度下降的过程中,$dw^{[l]}$ ，$w^{[l]}$ 也会变化

$$dw^{[l]} = dz^{[l]} * a^{[l-1]} + \frac{\lambda}{m}w^{[l]} \tag{10}$$

$$w^{[l]} = w^{[l]} - \alpha dw^{[l]} \tag{11}$$

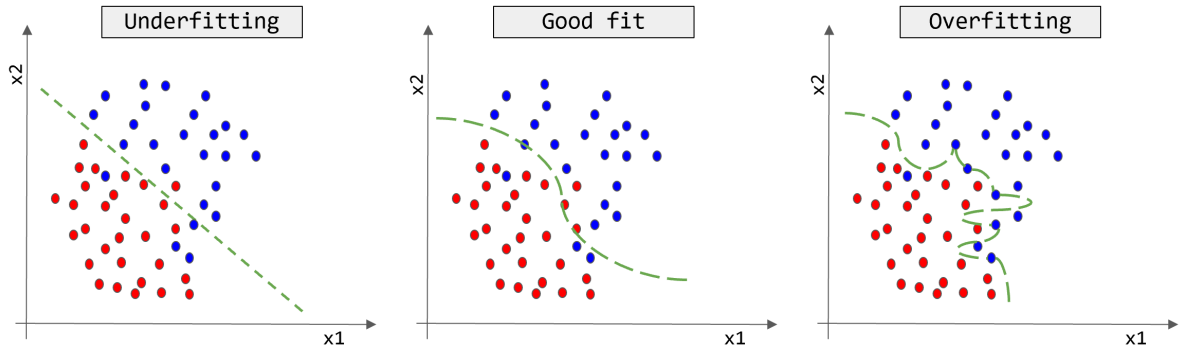
$$= w^{[l]} - \alpha * (dz^{[l]} * a^{[l-1]} + \frac{\lambda}{m}w^{[l]}) \tag{12}$$

$$= w^{[l]} - \frac{\alpha\lambda}{m}w^{[l]} - \alpha(dz^{[l]} * a^{[l-1]}) \tag{13}$$

$$= w^{[l]}(1 - \frac{\alpha\lambda}{m}) - \alpha(dz^{[l]} * a^{[l-1]}) \tag{14}$$

故，在梯度下降过程中，w 是逐渐变小的，所以 L_2 正则化有时称之为权重衰减

为什么 L_2 正则化可以防止过拟合，减少方差？



Train-Dev-Test vs. Model fitting

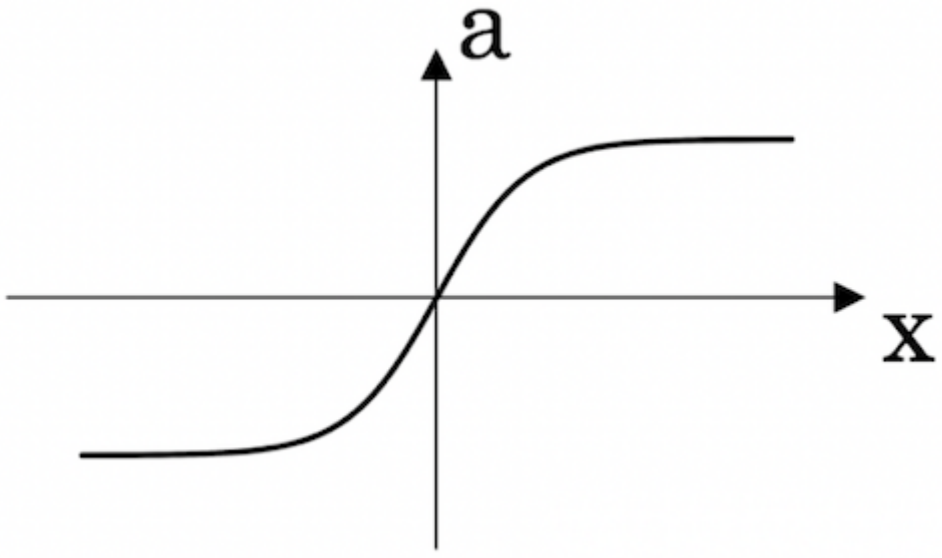
$$J(w^{[1]},b^{[1]},\cdots w^{[l]},b^{[l]})=\frac{1}{m}\sum_{i=1}^{m}L(\hat{y}^{(i)},y^{(i)})+\frac{\lambda}{2m}\sum_{i=1}^{l}\|w^{[l]}\|^2 \tag{15}$$

观察上图的公式：

当 $\lambda\uparrow$ 且 λ 足够大的时候 ——> w 就会 $\to 0$ ——> 从而将隐藏单元的影响削减 ——> 进一步使得神经网络简单化 ——> 最终使得神经网络接近逻辑回归

举个例子：

当激活函数是 $tanh(z)$ 时，

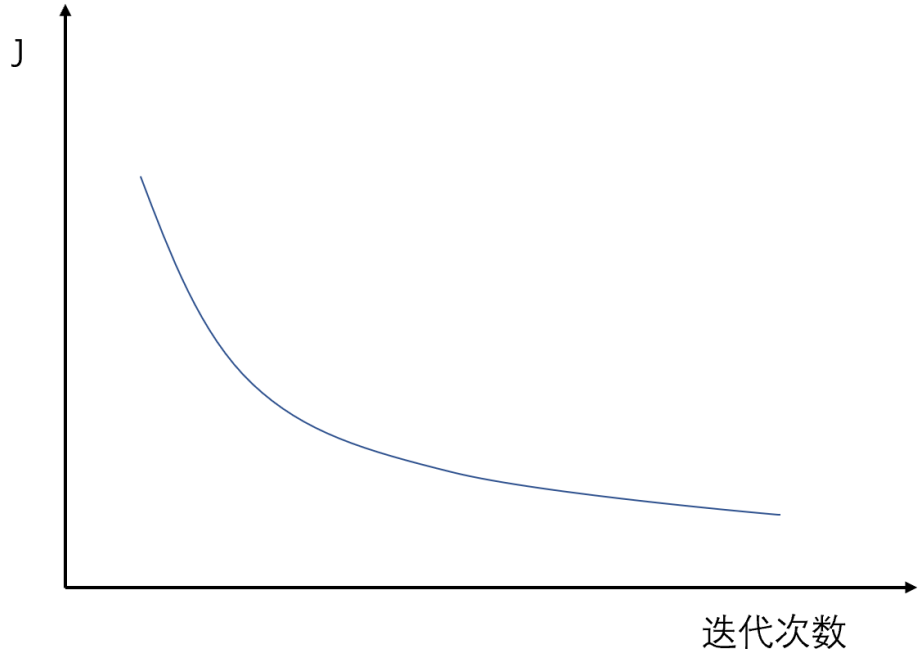


$$\lambda\uparrow\longrightarrow w^{[l]}\downarrow\longrightarrow(z^{[l]}=w^{[l]}a^{[l-1]}+b^{[l]})\downarrow\longrightarrow \text{接近线性} \tag{16}$$

因为 $w^{[l]}$ 变小会导致以 $w^{[l]}$ 为参数求出的 $z^{[l]}$ 的值变小，使得隐藏单元的影响削减，使得神经网络简单化，最后使得神经网络接近逻辑回归

就降低了方差

同时，在使用 L_2正则化时建议画出 J 关于梯度下降迭代次数的图像



你会发现使用 L_2 正则化后，随着迭代次数的增加，J 的值会逐渐减小