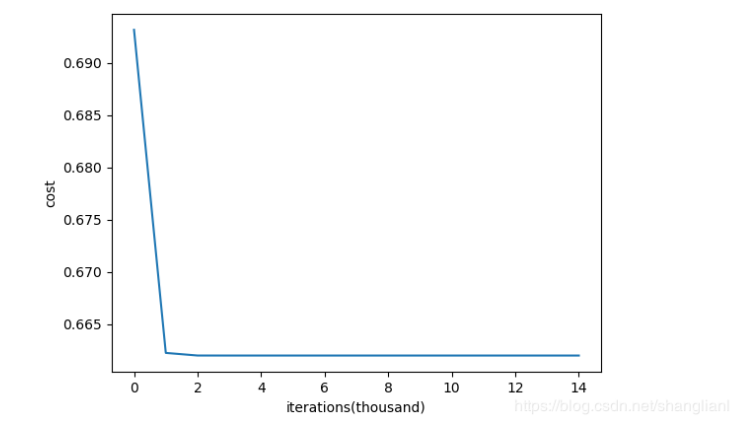
权重初始化

1. 全部初始化为零

实现代码：

W = np.zeros(input\_layer\_neurons, hidden\_layer\_neurons)

将权重全部初始化为零，每一层所学到的参数一样，因为梯度一样，所以在反响传播过程中，每一层神经元相同。所以会导致代价函数在一开始明显下降，一段时间后停止下降。



1. 初始化为相同的随机数：

将权重w初始化为相同的随机数与全部初始化为零的效果一样，会出现Symmetry problem

实现代码：

W = np.zeros(input\_layer\_neurons, hidden\_layer\_neurons)\*T

随机初始化可以打破对称。在随机初始化后，每个神经元可以继续学习其输入的不同功能。

1. 初始化为较小的随机数

权重参数随机初始化为服从均值为零和方差为1的高斯分布函数

对于含有个输入和个输出的全连接层：

standard\_normal：

standard\_uniform:

实现代码：

W = np.random.randn(input\_layer\_neurons, hidden\_layer\_neurons)\*0.01

随着时间增加，前向传播时，方差开始减少，梯度也开始向零靠近，会导致梯度消失。当激活函数使用sigmoid时，梯度接近0.5；当激活函数使用tanh时，梯度接近0

1. 初始化为较大的随机数

实现代码：

W = np.random.randn(input\_layer\_neurons, hidden\_layer\_neurons)

反向传播时，倒数趋于零，梯度也会消失。此外，权重较大且当输入也很大时，如果使用sigmoid做激活函数，会使输出趋向于0和1，会导致更多问题。

1. Xavier/Glorot Initialization（适用于激活函数是softsign和tanh）

条件：正向传播时，激活值的方差保持不变；反向传播时，关于状态值的梯度的方差保持不变，方差：

初始化方法：（具体推导过程：<https://blog.csdn.net/VictoriaW/article/details/73000632>、<https://blog.csdn.net/freeyy1314/article/details/85029599>）

xavier\_normal：

xavier\_uniform：

代码实现：

W = np.random.rand（（x\_dim，y\_dim））\* np.sqrt（1 /（ni + no））

注：ni是输入单元的数量，no是该层的输出单元的数量。

限制：在推导过程中假设激活函数在零点附近接近线性函数，且激活值关于0对称，而sigmoid、relu函数不满足这些假设。

参考资料：

1. Understanding the difficulty of training deep feedforward neural networks Xavier Glorot, Yoshua Bengio ; PMLR 9:249–256
2. MSRA/He initialization（适用于激活函数relu）

条件：正向传播时，状态值的方差保持不变；反向传播时，关于激活值的梯度的方差保持不变。

初始化方法：（具体推导过程：<https://blog.csdn.net/VictoriaW/article/details/73166752>、）

he\_normal：

he\_uniform：

代码实现：

W = np.random.randn（layer\_size [1]，layer\_size [L-1]）\* np.sqrt（1 / layer\_size [L-1]）

W = np.random.randn（layer\_size [1]，layer\_size [L-1]）\* np.sqrt（2 /（layer\_size [L-1] + layer\_size [1]））