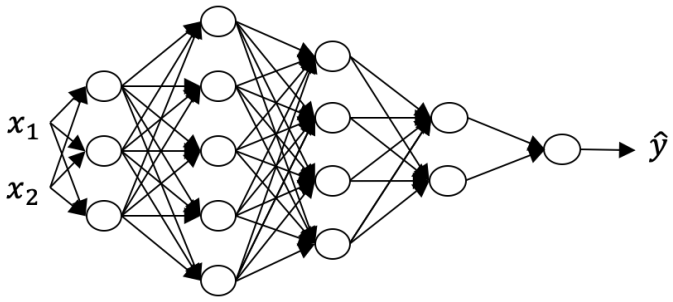
深层神经网络

深层神经网络概述

- 深层神经网络是指隐藏层超过**两层**的神经网络



符号定义

- 我们使用 `L` 来定义神经网络的层数（不包含输入层）
- `n` 表示每一层的神经元数量集合
  - `n[0]` 表示输入层的维数
  - `n[L]` 表示输出层的维数
- `g` 表示每一层的激活函数
- `z` 表示每一层的线性输出
  - `Z` 表示向量化后的线性输出
- `w` 和 `b` 表示每一层线性输出的对应参数
  - `W` 和 `B` 表示向量化后的参数
- `a` 表示每一层的激活输出
  - `a[0]` 表示输出，`a[L]` 表示输出
  - `A` 表示向量化后的激活输出

深层网络中的前向传播

- 对于单个输入，前向传播的伪代码如下：

```
z[l] = W[l]a[l-1] + b[l]
a[l] = g[l](z[l])
```

- 对于 `m` 个输入（向量化），前向传播的伪代码如下：

```
Z[l] = W[l]A[l-1] + B[l]
A[l] = g[l](Z[l])
```

- 我们无法对整个前向传播使用向量化，需要使用 for 循环（即每一层要分开计算）

维数的确认

- 我们需要确保各个向量的维数能够匹配
  - 这里用 `l` 表示当前是第几层
- `w[l]` 和 `dw[l]` 的维数：`(n[l], n[l-1])`
  - `W[l]` 和 `dW[l]` 的维数：`(n[l], n[l-1])`
- `b[l]` 和 `db[l]` 的维数：`(n[l], 1)`
  - `B[l]` 和 `dB[l]` 的维数：`(n[l], m)`
- `z[l]` 和 `a[l]` 的维数：`(n[l], 1)`
  - `Z[l]` 和 `A[l]` 的维数：`(n[l], m)`
  - `dZ[l]` 和 `dA[l]` 的维数：`(n[l], m)`

为什么要进行深层表示?

- 我们可以从两个角度解释为什么使用多个隐藏层：
  - 多个隐藏层可以将问题从简单到复杂进行拆分，先考虑简单的特征，再逐步变得复杂，最终实现预期的效果
  - 电路理论表明越少的层数需要的单元数呈指数级上升，对神经网络来说也是如此，
    - 对于一个复杂任务来说，层数越少每一层所要包含的神经元数量会爆炸式增长

深层神经网络的模块

- 深层神经网络一般包含前向传播与反向传播两个模块
  - 前向传播模块得到代价函数
  - 后向传播模块计算各层参数的梯度
  - 最后通过梯度下降来更新参数，进行学习
- 在实际实现中，我们需要通过缓存将前向传播中的某些参数传递到反向传播中，帮助进行梯度的计算

前向传播模块

- 向量化后的伪代码如下：

```
Input  A[l-1]
Z[l] = W[l]A[l-1] + B[l]
A[l] = g[l](Z[l])
Output A[l], cache(Z[l], W[l], B[l])
```

反向传播模块

- 向量化后的伪代码如下：

```
Input dA[l], Caches
dZ[l] = dA[l] * g'[l](Z[l])
dW[l] = (1/m) * np.dot(dZ[l], A[l-1].T)
dB[l] = (1/m) * np.sum(dZ[l], axis=1, keepdims=True)
dA[l-1] = np.dot(W[l].T, dZ[l])
Output dA[l-1], dW[l], dB[l]
```

- 最后一层 `dA` 的求解基于代价函数得出
  - 注意计算应去除 `1/m` 这一项，防止重复计算

参数与超参数

- 在神经网络中，参数主要指 `w` 和 `b`
- 超参数指影响参数选择的参数，例如：
  - 学习速率
  - 迭代次数
  - 隐藏层层数
  - 隐藏层单元数
  - 激励函数的选择
- 深度学习是一个经验主义的过程，随着外界条件的不断变化，需要进行多次的实验来确定最佳的超参数与参数

深层神经网络与大脑的关系

- 神经网络的单个逻辑单元与实际的神经元在**结构**上有一些相似
- 但大脑的工作原理目前还是未知的，所以无法进行进一步比较