Andrew Ng ML学习总结

**概述**

断断续续，一个月的时间，把吴老师的机器学习视频教程看完，收获很多，从一无所知到概念的理解、公式的推导、算法的探究等等，可以说基本上算是快要入门了，接下来将是继续学习。当然，在学习的过程中需要不断的总结、实践和提升，所以，接下来我将对所学到的知识进行简单总结，以加强自我学习。

什么是**机器学习**

机器学习是人工智能的核心，它在数据挖掘、计算机视觉、自然语言处理等领域有着广泛的应用。机器学习可以分为**监督学习**和**无监督学习**。

**监督学习**

监督学习是指：利用一组已知类别的样本调整分类器的参数，使其达到所要求性能的过程。例如，线性回归、逻辑回归、神经网络等算法都属于监督学习。

**线性回归算法**

线性回归根据其特征数，可分为一元线性回归和多元线性回归，这里以通用模型进行讲解。线性回归的**假设函数**模型如下：

其中，x为**特征值**，θ为我们的参数也叫**权重**，x\_0=1。经过推导和演变，可以得到我们的代价函数（方差）：

我们的**梯度下降**公式如下：

(for j=0 and j=m)

其中𝛼为**学习率**，它决定算法收敛的快慢，但取值过大可能造成无法收敛。通过迭代次数与J(θ)函数的可视化，可以查看梯度下降是否正确**收敛**。

根据多元微分法，我们可得到：

这样就可求得θ的值，进而得到我们的算法模型。

在梯度下降中，我们可以进行**特征缩放**，将我们的特征值缩放到一个相近的范围，如[-1,1]，这样我们的梯度下降会更快的收敛（特征缩放不能用于x\_0，因为它是我们假设的常量1）。**均值归一化**是一个不错的特征缩放方式。其公式为： x\_i = (x\_i – 平均值) / (最大值 – 最小值)

除了梯度下降法求解θ的值，我们还可以用**正规方程法**直接求解θ。至于什么时候选择梯度下降？什么时候选择正规方程求解？可根据我们的训练集大小来定夺，如果数据量过大，比如超过10000，那么应该就要考虑使用梯度下降来求解，如果数据量较小，那么选择方程法直接求解也是一个不错的选择。

**逻辑回归算法**

当我们要预测的值为一些离散的值时（如0、1的分类问题）线性回归就显得不适用，这时我们就需要开发一个新的算法——**逻辑回归**。其算法推导过程如下：

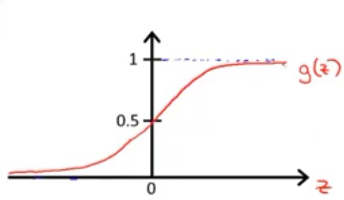
我们希望假设函数，根据线性回归的模型：

我们经过加工如下：

其中，g为**凸函数**，定义如下：

最后得到我们的假设函数如下：

我们的假设函数取值范围为[0,1]，如下：



在这个假设函数中，我们可以得到决策边界，来对我们的训练集进行分类，它是假设函数的属性，通过可视化，我们可以清晰的看到决策边界将训练集进行了分类。

逻辑回归的代价函数如下：

此模型是由统计学中极大似然法得来，且是凸性质的。

同样为了拟合代价函数，得到J(θ)最小值时参数θ，我们运用梯度下降公式：

从规则来看线性回归和逻辑回归基本相同，但实际上两者是不同的，区别在于假设函数：

特征缩放同样适用于逻辑回归算法。

**正则化**

如果我们的算法具有高方差，则可能出现**过度拟合**的情况；如果我们的算法出现高偏差，则可能为**欠拟合**。

当出现过拟合时我们有2种方法进行处理：

1. 人工检查变量清单，看哪些变量更为重要，哪些应该保留，哪些应该舍弃；
2. 模型选择算法，这种算法可自动选择哪些变量应该保留，哪些应该舍弃；

这种减少特征变量的方法可以有效避免过拟合，但缺点就是舍弃了一些信息，可能导致我们算法的结果不是很理想。

正则化可以有效避免过拟合问题，其思路是尽可能的使参数θ变小，这样我们拟合的线条就是更加平滑，所以需要给代价函数加一个**惩罚项**：

其中lambda为平衡参数，这样我们拟合出来的函数就会更加平滑。（默认情况下惩罚项不应用于θ\_0）

线性回归中的正则化后的代价函数如下：

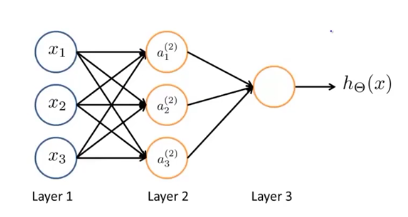
我们在执行梯度下降时的公式如下：

逻辑回归中的正则化后的代价函数如下：

它的梯度下降执行与线性回归基本相同。

**神经网络**

我们输入x\_1,x\_2,x\_3，到神经元，再输出h\_θ(x)函数，其中x\_0为偏置单元或偏置神经元，如下图所示：



如上图所示，我们的输入单元为x\_1,x\_2,x\_3，图层1是**输入层**，图层3是**输出层**，中间层是**隐藏层**。另外可以显式添加x\_0和a\_1(2)两个偏置单元。

术语：a\_i(j)表示图层j的第i个激活项；θ(j)表示图层j到图层j+1映射的权重矩阵。

继续看上图，我们可以推导：

接下来我们将其向量化：

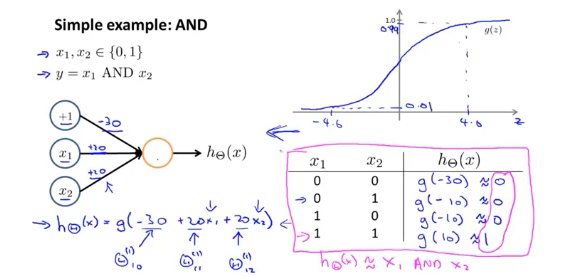
=> 添加

=>

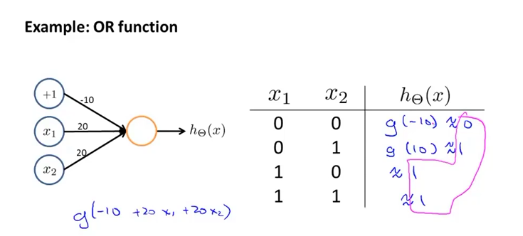
=>

这样的推导方式称为**前向传播**。

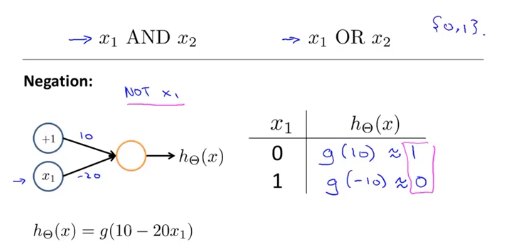
神经网络简单应用的实例如下：



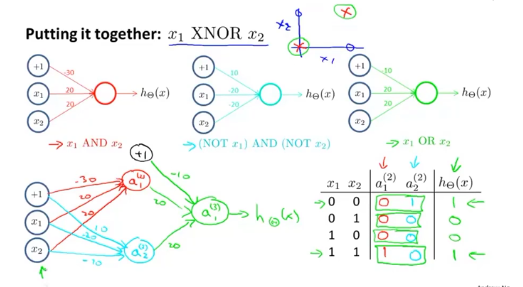
X1&X2



X1|X2



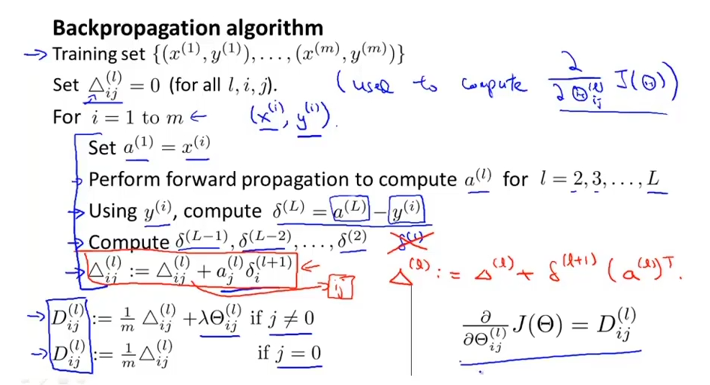
~X



X1 NOR X2

神经网络的代价函数如下：

神经网络反向传播算法：



**应用机器学习的建议**

当我们在调试学习算法的过程中，我们可以有以下尝试的方法：

* 获取更多的样本数据
* 将特征值缩放
* 获取更多特征
* 组合多项式特征
* 增加或减小lambda值

但上述方法未必能很好的解决问题，我们在评估假设函数时，可以将训练样本随机分成70%训练集和30%测试集，这样我们通过训练集学习算法，再用测试集计算误差。

或者我们可以将样本数据分为60%训练集、20%验证集（交叉验证集）和20%测试集，先训练集学习算法，再用验证集对算法进行评价，最后用测试集检查算法的泛化能力。

如果训练误差和交叉验证误差都很高，那么算法应该存在高偏差，即欠拟合；如果训练误差小，但交叉验证误差较大，那么算法存在高方差，即过拟合。

正则化可以很好的解决偏差或方差问题。

可视化学习曲线可以很直观的看出高偏差和高方差的现象。

当您在研究一个机器学习算法是，最好的做饭是先实现一个简单粗暴的算法，尽管它不太理想或者效果很差，一旦有了一个算法雏形之后我们再对它进行优化改进，看看它所造成的错误，通过误差分析看看出现了什么错误，然后决定优化方法。

当我们的训练集样本数据的正样本数与负样本数比例非常高，而我们预测y=0的概率非常好，但实际是算法欺骗了我们，我们称之为**偏斜类**，计算查准率和召回率可以很好的检查此类问题，**查准率**=真阳性数/预测阳性数，**召回率**=真阳性数/实际阳性数，查准率和召回率越高说明算法越好。

**支持向量机**

**无监督学习**

**K-means**