机器学习和深度学习面试题

1、SVM 的损失函数是什么?

合页损失函数:

$$L\big(y\cdot(w\cdot x+b)\big)=\max\{0,1-y(w\cdot x+b)\}$$

SVM 的损失函数就是合页损失函数加上正则化项:

$$\sum_{i}^{N}[1-y_i\cdot(w\cdot x_i+b)]_+ +\lambda\|\omega\|^2$$

其中下标"+"表示以下取正值的函数，我们用 z 表示中括号中的部分:

$$z_+=\begin{cases} z, z>0 \\ 0, z\leq=0 \end{cases}$$

也就是说，数据点如果被正确分类，损失为 0，如果没有被正确分类，损失为 z。

2、深度学习权重初始化的方法?

(1) 高斯分布初始化：参数从一个固定均值(比如 0)和固定方差(比如 0.01)的高斯分布进行随机初始化。

(2) 均匀分布初始化：在一个给定的区间[-r,r]内采用均匀分布来初始化参数。超参数 r 的设置可以按照神经元的连接数量进行自适应的调整。

(3) Xavier 初始化：就是尽可能的让输入和输出服从相同的分布，这样就能够避免后面层的激活函数的输出值趋向于 0。

(4) He initialization：何恺明提出了一种针对 ReLU 的初始化方法。

(5) 随机初始化+BN：BN 减小了网络对初始值尺度的依赖，使用较小的标准差初始化即可。

参考来源:

【机器学习】权重初始化

神经网络中的权重初始化常用方法

深度学习中神经网络的几种权重初始化方法

3、Gini 指数

$$Gini(p)=\sum_{k=1}^{k}p_k(1-p_k)=1-\sum_{k=1}^{k}p_k^2$$

对于数据集: $Gini(D)=1-\sum_{k=1}^{k}(\frac{C_k}{D})^2$,

按属性 A 划分: $Gini(D,A)=\sum_{j=1}^{V}\frac{|D_j|}{|D|}Gini(D_j)$

4、ID3 和 C4.5 算法

ID3 算法：

假设 D 为训练集，m 中类别，D 中元组的期望信息熵为：

$$Info(D) = \sum_{i=1}^{m} p_i log_2(p_i)$$

p_i为每个类别的占比，如果以属性 A 划分 D 中的元组，并且将 D 划分为 V 组：

$$Info_A(D) = \sum_{j=1}^{v} \frac{D_j}{|D|} \times info(D_j)$$

则信息增益为:$Gain(A) = Info(D) - Info_A(D)$，最终选择信息增益最大的属性作为每一步的划分属性。

C4.5 算法：利用信息增益率，将信息增益规范化

分裂信息熵为：$split\ info_A D = -\sum_{j=1}^{V} \frac{D_j}{|D|}$

则信息增益率：

$$GainRation(A) = \frac{Gain(A)}{split\ info_A D}$$

决策树剪枝：

1. 前剪枝：（1）节点中样本为同一类（2）特征值不足返回多类（3）某分支没有值返回父节点多类（4）样本数小于阈值
2. 后剪枝：（1）错误率降低剪枝（2）悲观错误剪枝（3）代价复杂度剪枝

5、分类模型的评价指标

	预测结果	
真实情况	正例	反例
正例	TP（真正例）	FN（假反例）
反例	FP（假正例）	TN（真反例

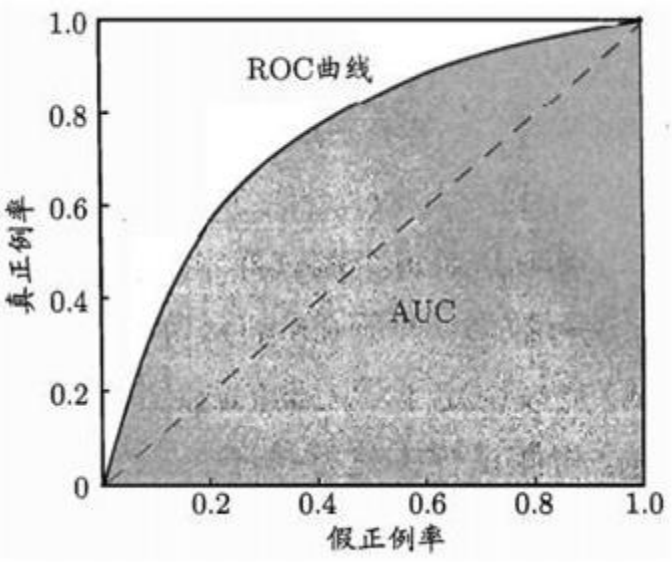
查准率（精准率）：Precision = TP / (TP+FP);

查全率（召回率）：Recall = TP / (TP+FN);

正确率（准确率）：Accuracy = (TP+TN) / (TP+FP+TN+FN)

F 值（F1-scores）：Precision 和 Recall 加权调和平均数，并假设两者一样重要。

F1-score = (2Recall*Precision) / (Recall + Precision)



真正例率，True Positive Rate：TPR = TP/ (TP+FN)
假正例率， False Postive Rate：FPR = FP/(TN+FP)

参考来源：[分类模型评价指标]

6、回归模型的评价标准

（1） SSE（误差平方和） $SSE = \sum (y - \hat{y})^2$

（2） R-square（决定系数） $R^2 = 1 - \frac{\sum (y-\hat{y})^2}{\sum (y-\bar{y})^2}$

（3） Adjusted R-square（矫正决定系数） $R^2_{adjusted} = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$

其中，y：实际值，$\hat{y}$：预测值，$\bar{y}$：平均值，n 为样本数量，p 为特征数量

7、梯度消失和梯度爆炸

神经网络基于反向传播，指导深度网络权值更新优化，BP 算法基于梯度下降策略，以目标负梯度方向对参数调整，反向传播基于的是链式求导法则。如果导数小于 1，那么随着层数的增多，梯度的更新量会以指数形式衰减，结果就是越靠近输出层的网络层参数更新比较正常，而靠近输入层的网络层参数可能基本就不更新。这就是梯度消失而如果导数值大于 1，那么由于链式法则的连乘，梯度更新量是会成指数级增长的。这就是梯度爆炸。

梯度消失出现的原因经常是因为网络层次过深，以及激活函数选择不当，比如 sigmoid 函数。梯度爆炸出现的原因也是网络层次过深，或者权值初始化值太大。

综合来看，这两种梯度问题产生的原因可以归结为网络深度，以及反向传播带来的遗留问题。

8、梯度消失和爆炸如何解决

（1） 预训练加微调

（2） 使用不同的激活函数
（3） 使用 BatchNormalization
（4） 使用残差结构
（5） 使用 LSTM

9、梯度爆炸的后果极其解决方法

（1）梯度爆炸导致学习模型无法从训练数据获得更新
（2）模型不稳定，导致更新过程中的损失出现显著变化
（3）训练过程中，模型损失变为 NaN.

如何解决?

（1）重新设计网络模型
（2）使用 Relu 激活函数
（3）使用 LSTM
（4）使用梯度剪切（当梯度大于某个值时，强制变小）
（5）使用权重正则化

10、共轭梯度法和梯度下降法区别

在 N 维优化问题中，每次沿一个方向优化得到极小值，后面再沿其他方向求极小值的时候，不会影响前面已经得到的沿那些方向上的极小值，所以理论上对 N 个方向都求出极小值就得到了 N 维问题的极小值。这组方向由于两两共轭，所以就叫他共轭方向法。

梯度下降法每次都直接选取当前点的梯度方向，所以就有可能按下葫芦浮起瓢：这次求出的极小值点在之前搜索过的方向上又不是极小值了，这样就导致收敛速度比较慢甚至不收敛

共轭方向:

设 A 是 n×n 对称正定矩阵，若有两个 n 维向量 P 和 Q，满足，PAQ=0，则称向量 P 和 Q 是关于 A 共轭的，或称 P、Q 是 A 共轭方向。

11、机器学习中常用的优化方法

（1）梯度下降法

用当前位置负梯度方向作为搜索方向，因为该方向为当前位置的最快下降方向，越接近目标值，步长越小，前进越慢。

$$h(\theta) = \sum_{j=0}^{n} \theta_j x_j$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (y^i - h_0(x^i))^2$$

(a)批量梯度下降法:

$$\frac{\partial J(\theta)}{\partial(\theta_j)} = -\frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

$$\theta_j' = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

(b)随机梯度下降（SGD），上述 m=1

（2）牛顿法

(a)牛顿迭代法：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

优点：二阶收敛，收敛速度快；

缺点：牛顿法是一种迭代算法，每一步都需要求解目标函数的 Hessian 矩阵的逆矩阵，计算比较复杂。

(b)拟牛顿法：改善牛顿法每次需要求解复杂的 Hessian 矩阵的逆矩阵的缺陷，它使用正定矩阵来近似 Hessian 矩阵的逆，从而简化了运算的复杂度。

（3）共轭梯度法（共轭梯度法详细推导分析）

共轭梯度法是介于最速下降法与牛顿法之间的一个方法，它仅需利用一阶导数信息，但克服了最速下降法收敛慢的缺点，又避免了牛顿法需要存储和计算 Hesse 矩阵并求逆的缺点。

（4）启发式优化算法

（5）拉格朗日乘数法

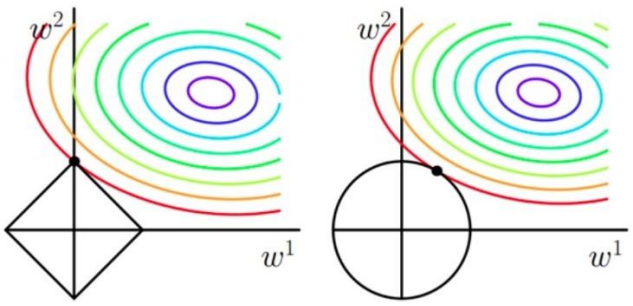
参考来源：机器学习中常用的优化方法

12、正则化

正则化解释：特征过多会过拟合，为了防止过拟合会选择重要特征，删掉次要的特征，实际上却希望利用这些特征，就可以以添加正则项来约束特征变量，使权重很小，又不至于影响过大。

L1 正则表示各个参数的绝对值之和

L2 正则表示各个参数平方和的开方值



L1 正则的解具有稀疏性，可用于特征选择（拉普拉斯分布）

L2 正则的解比较小，抗扰动能力强（高斯分布）

13、数据样本不均衡处理

（1）对大类样本进行欠采样，即删除部分样本
（2）对小类样本进行过采样，即添加部分样本
（3）考虑随机采样与非随机采样两种方法
（4）考虑对各类别尝试不同的采样比例

14、Softmax 公式，Softmax Loss

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{T} e^{a_k}}$$

Loss $L = -\sum_{j=1}^{T} y_j \log(S_j)$ ———真实标签为 1，其他为 0———> $L = -\log(S_j)$

例如：真实标签为 y=[0,0,0,1,0]，模型输出结果概率（softmax 输出）P=[0.1,0.15,0.05,0.6,0.1]，loss = -log(0.6)

15、损失函数，代价函数，目标函数以及常见损失函数

损失函数：作用于单个样本，用来表达样本误差
代价函数：作用于训练接，整个样本平均误差
目标函数：代价函数加上权重约束，例如

$$\min(\theta) = \frac{1}{n}\sum_{i=1}^{n} L\big(y_i, f(x_i;\theta)\big) + \lambda\phi(\theta)$$

(a) 0-1 损失函数:

$$L(y, f(x)) = \begin{cases} 1, y \neq f(x) \\ 0, y = f(x) \end{cases}$$

(b) 平方损失函数:

$$L(y, f(x)) = (y - f(x))^2$$

(c) 绝对值损失函数:

$$L(y, f(x)) = |y - f(x)|$$

(d) Hinge 损失（合页损失）函数:

$$L(w, b) = \max\{1 - yf(x), 0\}$$

y=+1 或-1，f(x) = wx + b，当 SVM 为线性核。

(e) 对数损失函数:

$$L(y, p(y|x)) = -logP(y|x) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

(f) 指数损失函数:

$$L(y, f(x) = e^{-yf(x)}$$

(g) 交叉熵损失函数:

$$L(y, f(x)) = -\sum_{j=1}^{T} y_j \log f(x_j)$$

(h) Huber 损失函数：

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & otherwise \end{cases}$$

16、bagging 和 boosting 的区别

Bagging：

(1) 从原始样本中抽取训练集，使用 bootstrapping 方法抽取 n 个训练样本，共进行 K 轮抽取，得到 K 个训练集。

(2) 每次使用一个训练集得到一个模型，K 个训练集得到 K 个模型。

(3) 对分类问题，将上述 K 个模型采用投票的方式得到分类结果；对于回归问题，计算上述模型的均值作为结果。

Boosting：将弱分类器组成强分类器的过程

(1) 通过提高那些在前一轮被弱分类器分错样例权值，减小前一轮分对样例的权值，使得分类器对误分数据有较好的效果。

(2) 通过加法模型将若分类器进行线性组合，例如 AdaBoost 通过加权多数表决的方式，即增大错误率小的分类器权值，同时减小错误率大的分类器权值。

两个区别：

(1) 样本选择：bagging 有放回选取，各轮之间独立；boosting 训练集不变，每个样例在分类器中权重相等

(2) 样例权重：bagging 使用均匀取样，每个样例权重相等；boosting 根据错误率不断调整权重，错误率越大权重越大

(3) 预测函数：bagging 预测函数权重相等；bagging 对分类误差小的分类器会有更大的权重

(4) 并行计算：bagging 各个预测函数可以并行生成；boosting 各个预测函数只能顺序生成，因为后一个模型参数需要前一轮模型结果

(a) Bagging+决策树 = 随机森林

(b) AdaBoost+决策树 = 提升树

(c) Gradient Boosting+决策树 = GBDT

17、GBDT 算法

(1) 初始化弱学习器

$$f_0(x) = argmin_c \sum_{i=1}^{N} L(y_i, c)$$

(2) 对 m=1,2,…,M 有：

(a) 对每样本 i=1,2,…,N，计算负梯度，即残差

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$$

（b）将上步得到的残差作为样本的新真实值，并将数据（x_i, r_{im}），$i = 1,2,\cdots,N$作为下棵树的训练数据，得到一颗新的回归树$f_m(x)$其对应的叶子节点区域为$R_{jm}, j = 1,2,\cdots,J$。其中 J 为回归树 t 的叶子节点个数

（c）对于叶子区域$j = 1,2,\cdots,J$计算最佳拟合值

$$\gamma_{jm} = \underbrace{\arg min}_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$

（d）更新强学习器

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm})$$

（3）得到最终学习器

$$f(x) = f_M(x) = f_0(x) + \sum_{m=1}^{M} \sum_{j=1}^{J} \gamma_{jm} I(x \in R_{jm})$$

参考链接：[GBDT 算法原理以及实例理解]

二元 GBDT 分类　损失函数：

$$L\big(y, f(x)\big) = \log\left(1 + \exp\left(-yf(x)\right)\right)$$

多元 GBDT 分类　损失函数

$$L\big(y, f(x)\big) = -\sum_{k=1}^{K} \mathrm{y}_k \log\big(P_k(x)\big), P_k(x) = \frac{\exp\left(f_k(x)\right)}{\sum_{k=1}^{K} \exp\left(f(x)\right)}$$

GBDT 与 XGBDT 区别：

（1）损失函数用泰勒展开二项逼近，而不像 GBDT 是一阶导数

（2）对树结构进行了正则约束，防止模型过度复杂

（3）节点分裂方式不同，GBDT 用基尼系数，XGBDT 经过优化推到后得到

18、一个完整的机器学习项目流程

（1）抽象成数学问题，目标是回归还是分类还是聚类

（2）获取数据，数据要有代表性：对分类问题，数据偏斜不能过于严重，对数据量级进行评估，内存消耗程度，训练过程中内存能否放下，如果放不下考虑改进或者降维或者分布式

（3）特征预处理与特征选择：归一化、离散化、因子化、缺失值处理、去除共线性

特征选择：

（a）filter 方法：方差法；相关系数；卡方检验；互信息法

（b）wrapper 方法：根据预测效果评分“指定”特征子集

（c）embedded 法：将特征选择和学期起的训练融为一体，学习器自动选择特征，如决策树的信息增益、信息增益率、基尼系数

（4）训练模型与调优　调整模型的（超）参数

（5）模型诊断　模型过拟合、欠拟合；交叉验证；绘制学习曲线；观察误差样本分析是参数问题还是算法选择问题，特征还是数据？

（6）上线运行　跟工程实现相关性比较大，线上运行效果决定模型成败，不单纯包括其准确度，误差等情况，还包括其运行的速度（时间复杂度），资源消耗程度（空间复杂度），稳定性等

19、Batch Normalization 思想

深层神经网络在做非线性变换前的激活输入值（即 f=Wx+b，x 为输入值）随着网络深度加深或者训练过程中，其分布逐渐发生偏移或者变动，之所以训练收敛慢，一般是整体分布逐渐往非线性函数的取值区间上下两端靠近（对于 sigmoid 来说，意味着激活输入值 Wx+b 是最大的负值或正值）导致反向传播时低层神经网络梯度消失，BN 就是把输入分布强制拉回到均值为 0 方差为 1 的标准正太分布，使得非线性变换函数落入梯度变化比较敏感的区域，以避免梯度消失问题。

参考链接：【深度学习】批归一化（Batch Normalization）

20、判别式模型和生成式模型

（1）判断方法：通过数据直接学习决策函数 Y=f(x)，或者由条件概率 P(Y|X)作为预测模型，即判别模型

（2）生成方法：由数据学习联合概率密度分布函数 P(X,Y)，然后求出条件概率分布 P(Y|X)作为预测的模型，即生成模型

判别式模型不能得到生成式模型，由生成式模型可以得到判别式模型

判别式模型：KNN，SVM，决策树，线性回归，LR，Boosting，线性判别分析（LDA），条件随机场，传统神经网络

生成式模型：朴素贝叶斯，隐马尔科夫模型，高斯混合模型，受限玻尔兹曼机

21、卷积、卷积神经网络的三大特点

卷积数学表达是，f(x)，g(x)为 R1 上两个可积函数，作积分：

$$\int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau$$

神经网络中卷积，对图像（不同数据窗口）和滤波矩阵（一组固定权重，因为每个神经元的多个权重固定，又可以看成一个恒定滤波器 filter）作内积。

作用：对一局部区域进行卷积得到这个局部区域的特征值传入下层大大提升神经网络提取特征能力，还减小了数据大小。

卷积神经网络的三个特点：

（1）局部感知：即网络部分连通，每个神经元只与上一层的部分神经元相连，只感知局部，而不是整幅图像。（滑窗实现）

（2）权值共享：从一个局部区域学习到的信息，应用到图像的其它地方去。即用一个相同的卷积核去卷积整幅图像，相当于对图像做一个全图滤波。

（3）池化：比如 max pooling，它是取一个区域的最大值。因此当图像发生平移、缩放、旋转等较小的变化时，依然很有可能在同一位置取到最大值，与变化前的响应相同，由此实现了仿射不变性。average pooling 同理，发生较小的仿射变化后，均值可能依然不变。保留主要特征同时减小参数（降维）和计算量，防止过拟合，提高模型泛化能力。

参考链接：CNN 初步认识（局部感知、权值共享）

22、SVM 的原理

SVM，二分类模型，目的是找一个超平面对样本进行分割，分割原则是间隔最大化，最终转化为一个凸二次规划问题求解。

（1）当训练样本可分时，通过硬间隔最大化，学习一个线性可分支持向量机
（2）当训练样本近似线性可分，通过软间隔最大化，学习一个线性支持向量机
（3）当样本不可分时，通过核技巧和软间隔最大化，学习一个非线性支持向量机

核函数：线性核，多项式核函数，高斯核函数

凸优化问题——>构造拉格朗日函数—KTT 条件—>对偶问题（求解更高效）—SMO 算法—>求解

为什么转为对偶问题？

原问题如果不是二次凸优化（往往实现就是如此）比较难解，复杂麻烦，而对偶问题则简单很多，就是线性凸优化问题。实际上无论原问题是什么形式，对偶问题总是个凸优化问题，其极值是唯一的。

KTT 条件：定义不等式下约束的拉格朗日函数 L,

$$L(x,\lambda,u)=f(x)+\sum_{j=1}^{p}\lambda_j h_j(x)+\sum_{k=1}^{q}\mu_k g_k(x)$$

$f(x)$为原函数，$h_j(x)$为等式约束，$g_k(x)$为不等式约束
若求解上述优化问题，必须满足下述条件（KTT 条件）

$$\begin{cases}\left.\dfrac{\partial L}{\partial x}\right|_{x=x^*}=0\\ \lambda_j\neq 0\\ u_k\geq 0\\ u_k g_k(x^*)=0\\ h_j(x^*)=0, j=1,2,\cdots,p\\ g_k(x^*)\leq 0, k=1,2,\cdots,p\end{cases}$$

参考链接：
支持向量机（SVM）从入门到放弃再到掌握

23、多分类问题如何解决

（1）一对一：将 N 个类别两两配对，产生 N(N-1)/2 个二分类，测试阶段样本同时交给所有分类器，投票产生结果
（2）一对多：每次将一个例作为正例，其他的作为反例，训练 N 个分类器，测试时如果只有一个分类器，预测为正类，则对应类别为最终结果，如果有多个，一般选置信度大的
（3）多对多，若干类作为正类，若干类作为反类

24、常用距离度量表达式

（1）欧几里德距离：

$$dist(i,j)=\sqrt{|x_{i1}-x_{j1}|^2+|x_{i2}-x_{j2}|^2+\cdots|x_{ip}-x_{jp}|^2}$$

（2）曼哈顿距离：

$$dist(i,j)=\sqrt{|x_{i1}-x_{j1}|^2+|x_{i2}-x_{j2}|^2+\cdots|x_{ip}-x_{jp}|^2}$$

（3）闵可夫斯基距离：

$$\sqrt[q]{|x_{i1}-x_{j1}|^q+|x_{i2}-x_{j2}|^q+\cdots|x_{ip}-x_{jp}|^q}$$

25、LR 和 SVM 的区别和联系

联系：

（1）LR 和 SVM 都可以处理分类问题，且一般用于线性二分类问题

（2）两个方法都可以增加正则项，如 L1，L2 等

区别：

（1）LR 为参数模型，SVM 为非参数模型

（2）目标函数，逻辑回归采用 logistic loss

$$L(\hat{y},y)=-ylog(\hat{y})-(1-y)\log(1-\hat{y})$$

$\hat{y}$表示训练样本下预测$y=1$的概率

$$\begin{cases} if\ y=1: P(y|x)=\hat{y} \\ if\ y=0: P(y|x)=1-\hat{y} \end{cases}$$

SVM 采用的 hinge loss+正则项

$$\sum_{i}^{N}[1-y_i\cdot(w\cdot x_i+b)]_+ +\lambda\|\omega\|^2$$

（3）LR 对异常值敏感，SVM 对异常值不敏感

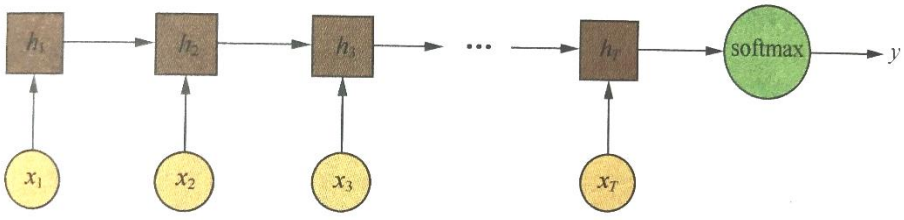
（4）计算复杂度不同，对于海量数据，SVM 较低，LR 效率比较高

26、Dropout 为何可以防止过拟合?

（1）取平均的作用。用了 dropout 之后，相当于训练了很多个只有半数隐层单元的神经网络，每一个这样的网络都可以给出一个分类结果，大部分网络都可以给出正确分类，可以看作为集成大量深层神经网络的使用 bagging 方法

（2）减少神经元之间复杂的共线性。因为 dropout 使两个神经元不一定每次都在一个子网络结构中出现，基于此权值更新不再依赖固定关系的隐含节点的共同作用，阻止了某些特征仅仅在其他特征下才有效的情况，迫使网络学习更加鲁棒性的特征

27、循环神经网络



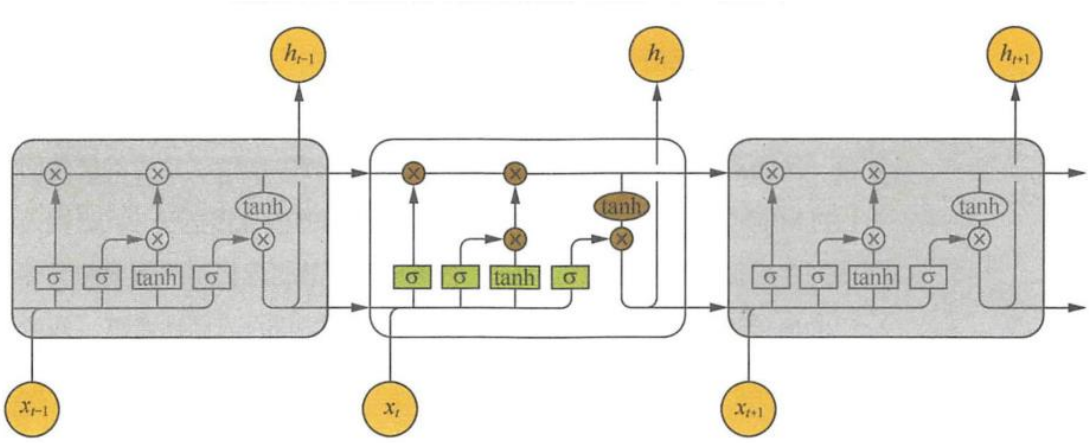
其中第 t 层隐含状态h_t编码了序列前 t 个输入信息，可以通过当前x_t和上一层神经网络的状态h_{t-1}计算得到，最后一层的状态h_T编码了整个序列的信息。

$$net_t=Ux_t+wh_t$$
$$h_t=f(net_t)$$
$$y=g(Vh_T)$$

f,g 为激活函数，U 为输入层到隐含层的权重矩阵，w 为隐含层从上一时刻到下一时刻状态转移权重矩阵，在在文本分类任务中，f 可以取 Tanh 函数或者 ReLU 函数，g 可以采用 Softmax 函数。

28、长短期记忆网络 LSTM

LSTM 是循环神经网络的最知名和成功的扩展，RNN 有梯度消失和梯度爆炸的问题，学习能力有限，在实际中往往达不到效果，LSTM 可以对有价值的信息进行长期记忆，从而减小循环神经网络的难度。



LSTM 仍然是基于x_t和h_{t-1}计算，只不过内部增加了更精心设计，加入了输入门i_t，遗忘门f_t以及输出门o_t三个门和一个内部记忆单元c_t

（1）输入门控制当前计算新状态以多大程度更新到记忆单元中

（2）遗忘门控制前一步记忆单元中的信息有多大程度被遗忘

（3）输出门控制当前输出有多大程度取决于当前的记忆单元

第 t 步的更新公式为：

$$i_t = \sigma(w_i x_t + U_i h_{t-1} + b_i)$$
$$f_t = \sigma(w_f x_t + U_f h_{t-1} + b_f)$$
$$o_t = \sigma(w_o x_t + U_o h_{t-1} + b_o)$$
$$\tilde{c}_t = tanh(w_c x_t + U_c h_{t-1})$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$h_t = O_t \odot \tanh(c_t)$$

其中⊙表示矩阵中对应元素相乘，i_t是通过x_t和上一步隐含层输出h_{t-1}，再经过σ激活函数得到。

29、超参数调优方法

（1）网格搜索，通过查找搜索范围内所有点来确定最优值，如果采用较大的搜索范围以及较小的步长，网格搜索有很大概率找到全局最优，但十分消耗计算资源、时间。一般先用较大的搜索范围和较大的步长，来寻找最优值可能的位置，然后缩小搜索范围和步长，来寻找最优值

（2）随机搜索，在搜索范围内随机选取样本点，理论是如果样本集足够大，那么通过随机采样也能大概率地找到全局最优值或近似值

（3）贝叶斯优化算法，贝叶斯优化算法充分利用之前的信息，通过对目标函数的形状学习，找到使目标向全局最优值提升的参数，具体为，首先根据先验分布，假设一个搜索函数，然后每次用一个新的采样点来测试目标函数时，利用这个信息更新目标函数的

先验分布，最后算法测试由后验分布给出全局最优值最可能出现的位置

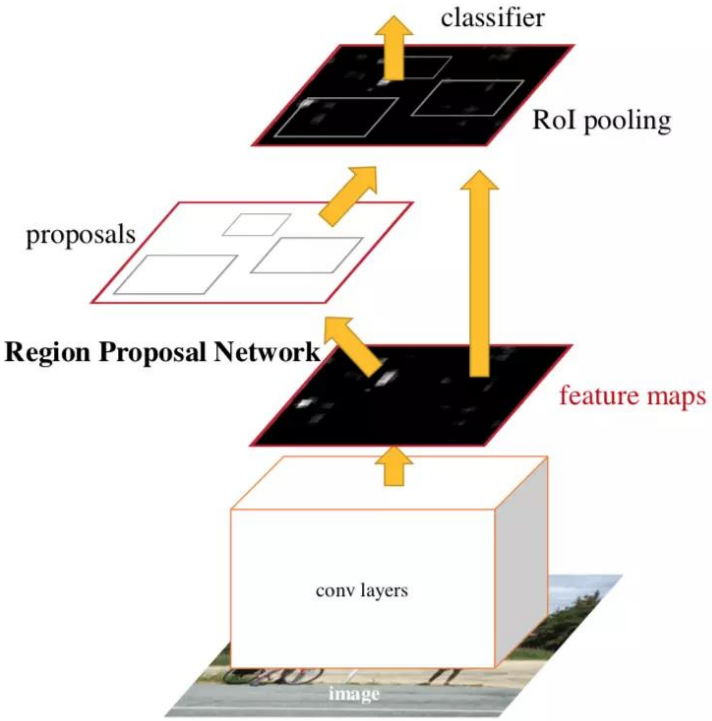
30、faster-cnn 框架

（1）使用 VGG16 或者其他成熟图片分类模型提取图片的特征（feature map）

（2）将图片喂入 RPN（region proposal network）网络得到 proposals（含第一次回归）

（3）将上步的结果特征和 proposals 喂入 ROI pooling 层得到综合的 proposals 特征

（4）根据 proposals 特征预测物体的 bounding box 和物体的类别（含第二次回归）



参考链接：

ROI pooling 解释

Faster R-CNN 文章详细解读

33、1×1 卷积的作用

（1）降维或升维，例如一张 500×500 且厚度 depth 为 100 的图片在 20 个 filter 上做 1*1 卷积，那么结果为 500*500*20

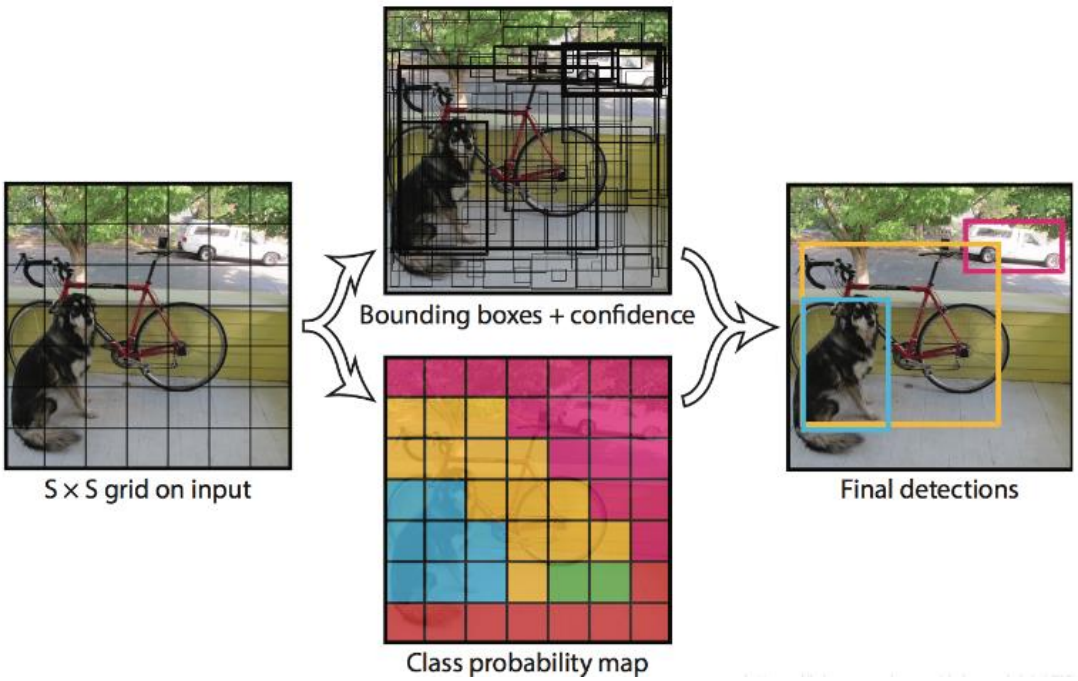
（2）加入非线性，卷积层之后经过激励层，1*1 的卷积在前一层上添加了非线性激励，提升了网络的表达力

34、yolov1、yolov2、yolov3 区别

Yolov1: YOLO v1 的核心思想在于将目标检测作为回归问题解决 ，YOLO v1 首先会把原始图片放缩到 448×448 的尺寸，放缩到这个尺寸是为了后面整除来的方便。然后将图片划分成 SxS 个区域，如果一个对象的中心落在某个单元格上，那么这个单元格负责预测这个物体。每个单元格需要预测 B 个边界框（bbox）值(bbox 值包括坐标和宽高)，同时为每个 bbox 值预测一个置信度(confidence scores)，此后以每个单元格为单位进行预测分析。

$$confidence = P(Object) \times IOU_{pred}^{truth}$$

每个边界框对应于 5 个输出，分别是 x，y，w，h 和置信度。其中 x，y 代表边界框的中心离开其所在网格单元格边界的偏移。w，h 代表边界框真实宽高相对于整幅图像的比例。x，y，w，h 这几个参数都已经被限制到了区间[0,1]上。除此以外，每个单元格还产生 C 个条件概率。注意，我们不管 B 的大小，每个单元格只产生一组这样的概率。



在 test 的非极大值抑制阶段，对于每个边界框，按照下式衡量该框是否应该予以保留。

$$confidence \times P(Class_i \mid Object) = P(Class_i) \times IOU_{pred}^{truth}$$

YOLO v1 全部使用了均方差（mean squared error）作为损失（loss）函数。由三部分组成：坐标误差、IOU 误差和分类误差。其损失函数如下：

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Yolov2：引入了 anchor，同时使用 k-means 方法对 anchor 数量进行讨论，在精度和速度之间折中，修改了网络结构，去掉全连接层，改成全卷积结构。在训练时引入 world tree，将检测和分类问题做成了统一的框架，并提出一种层次性联合训练方法，将 imagenet 分类集合 coco 检测数据集同时对模型训练。

相比于 yolov1：

（1）Batch Normalization：

- v1 中大量用了 BN（Batch Normalization）,同时在定位层 dropout
- v2 中取消了 dropout，均采用了 BN 归一化处理，使模型更稳定收敛更快

（2）高分辨率分类器：

- v1 中使用 224 x 224 预训练，448 x 448 用于网络监测
- v2 中以 448 x 448 的分辨率微调最初的分类网络

3.Anchor Boxes：

- 预测 bbox 的偏移，使用卷积代替 FC
- 输入大小为：416 x 416
- max pooling 下采样（32 倍下采样，输出得到 13x13 特征图）
- 预测超过 1000 个
- mAP 降低，recall 显著提高

使用 Anchor Boxes 来预测 Bounding Boxes。作者去掉了网络中一个池化层，这让卷积层的输出能有更高的分辨率。收缩网络让其运行在 416×416 而不是 448×448。由于图片中的物体都倾向于出现在图片的中心位置，特别是那种比较大的物体，所以有一个单独位于物体中心的位置用于预测这些物体。YOLO 的卷积层采用 32 这个值来下采样图片，所以通过选择 416×416 用作输入尺寸最终能输出一个 13×13 的特征图。使用 Anchor Box 会让精确度稍微下降，但用了它能让 YOLO 能预测出大于一千个框，同时 recall 达到 88%，mAP 达到 69.2%。

（4）细粒度特征：

- 添加 pass through layer， 把浅层特征图（26x26）连接到深层特征图
- Resnet 中的 identity mapping
- 输入大小为：416 x 416
- 把 26x26x512 的特征图叠加成 13x13x2048 的特征图，与深层特征图相连接，增加细粒度特征
- 性能获得 1%的提升

（5）多尺度训练：

- 每隔几次迭代后就会微调网络的输入尺寸
- 输入图像尺寸：320,352，…，608
- 尺度的变化

Yolov2 依赖于 Darknet-19：

- 主要使用 3 x 3 卷积
- pooling 之后 channel 数增加
- global average pooling
- 用 1 x 1 卷积压缩特征
- batch normalization

Yolov3:

相比于 yolov2：

- loss 不同：作者 v3 替换了 v2 的 softmax loss 变成 logistic loss，而且每个 ground truth 只匹配一个先验框。
- anchor bbox prior 不同：v2 作者用了 5 个 anchor，一个折衷的选择，所以 v3 用了 9 个 anchor，提高了 IOU。
- detection 的策略不同：v2 只有一个 detection，v3 一下变成了 3 个，分别是一个下采样的，feature map 为 13*13，还有 2 个上采样的 eltwise sum，feature map 为 26*26, 52*52, 也就是说 v3 的 416 版本已经用到了 52 的 feature map，而 v2 把多尺度考虑到训练的 data 采样上，最后也只是用到了 13 的 feature map，这应该是对小目标影响最大的地方。
- backbone 不同：这和上一点是有关系的，v2 的 darknet-19 变成了 v3 的 darknet-53，为啥呢？就是需要上采样啊，卷积层的数量自然就多了，另外作者还是用了一连串的 3*3、1*1 卷积，3*3 的卷积增加 channel，而 1*1 的卷积在于压缩 3*3 卷积后的特征表示，这波操作很具有实用性，一增一减，效果较好

参考链接：

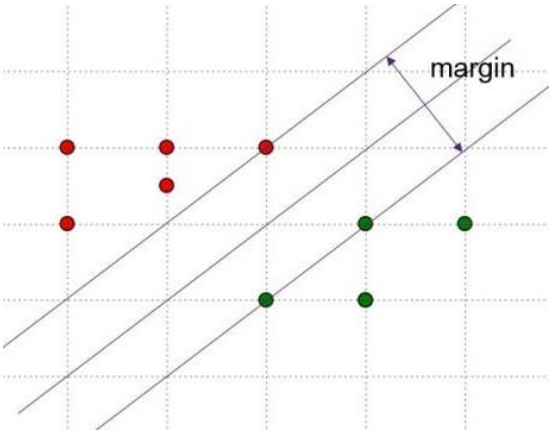
[目标检测：yolov1、yolov2、yolov3 的对比理解]

[从 YOLOv1 到 YOLOv3，目标检测的进化之路]

[yolo 类检测算法解析——yolo v3]

35、SVM 推导

SVM 主要思想是通过寻找使得 Margin 最大的超平面，该超平面为两条平行边界超平面的中心。



假设分类超平面为$w^Tx+b=0$，两个平行的边界超平面分别为：

$$\begin{cases} w^Tx+b=l \\ w^Tx+b=-l \end{cases}$$

因为$\frac{wT}{l}x+\frac{b}{l}=0$与$w^Tx+b=0$是同一个超平面，所以等式两边同除以 l，并进行变量替换后，边界超可以定义成如下形式：

$$\begin{cases} w^Tx+b=1 \\ w^Tx+b=-1 \end{cases}$$

样本被两个边界超平面分隔开，假设对于标签$y_i=1$的样本，$w^Tx+b\geq 1$，对于标

签$y_i = -1$的样本，$w^T x + b \leq -1$，两种情况可以统一：

$$y_i(w^T x_i + b) \geq 1$$

边界平面$w^T x + b = 1$上任意一点到超分类超平面$w^T x + b = 0$的距离为：$d = \frac{1}{||\omega||}$，

margin 为其二倍，等于$\frac{2}{||\omega||}$，SVM 的求解过程就是求解$max_w \frac{2}{||\omega||}$，等同于$min_w \frac{1}{2}||\omega||^2$。

至此，我们将 SVM 转化为带有条件约束的最优化问题：

$$\begin{cases} min_w \frac{1}{2}||\omega||^2 \\ y_i(w^T x_i + b) \geq 1 \end{cases}$$

SVM 的求解思路：

对于上面带有约束条件的凸优化问题，构造如下拉格朗日函数：

$$L(\omega, b, \alpha) = \frac{1}{2}||\omega||^2 + \sum_i \alpha_i(1 - y_i(w^T x_i + b))$$

则原带有约束条件的问题可以等价为$min_{w,b} max_\alpha L(\omega, b, \alpha)$，将 min 和 max 互换顺序，得到原问题的对偶问题，$max_\alpha min_{w,b} L(\omega, b, \alpha)$，令$\theta(\alpha) = min_{w,b} L(\omega, b, \alpha)$，则对偶问题可以表示为$max_\alpha \theta(\alpha)$。

因为 SVM 原问题为凸优化问题，在 slater 条件满足时，原问题与对偶问题等价，可以通过求解对偶问题得到原问题的最优解（因为 KKT 条件为强对偶的必要条件，则此时 KKT 条件必然也是满足的）

求解$\theta(\alpha)$，令$L(\omega, b, \alpha)$对 w,b 的偏导为 0：

$$\frac{\partial L(w, b, \alpha)}{\partial w} = 0$$
$$\frac{\partial L(w, b, \alpha)}{\partial b} = 0$$

可以得到：

$$w = \sum_i \alpha_i y_i x_i$$
$$\sum_i \alpha_i y_i = 0$$

把 w, b 带入$L(\omega, b, \alpha)$可得到$\theta(\alpha)$如下：

$$\theta(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

对偶问题最终可以表示如下：

$$\begin{cases} max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \alpha_i \geq 0 \\ \sum_i \alpha_i y_i = 0 \end{cases}$$

其中$\sum_i \alpha_i y_i = 0$是$L(\omega, b, \alpha)$对 b 偏导为 0 的约束，$\alpha_i \geq 0$是拉格朗日因子本身满足的条件。

求解上式子（利用 SMO 算法），可以得到最优的一组α_i^*，然后利用$w^* = \sum_i \alpha_i^* y_i x_i$

求解得到最优的w^*。假设x_+，x_-，分别为两个边界超平面上的点，则$w^*x_+ + b = 1$ $w^*x_+ + b = -1$，两式子相加可得：

$$b^* = \frac{w^*x_+ + w^*x_-}{2}$$

根据 KKT 条件，$\alpha^*\left(y_i(w^*x_i + b) - 1\right) = 0$,则若$\alpha^* > 0$，则$\left(y_i(w^*x_i + b)\right) - 1 = 0$, 即在$x_i$在边界分类平面上, 则求解 b 过程中的$x_+$，$x_-$可以在$\alpha^* > 0$的$x_i$中寻找。

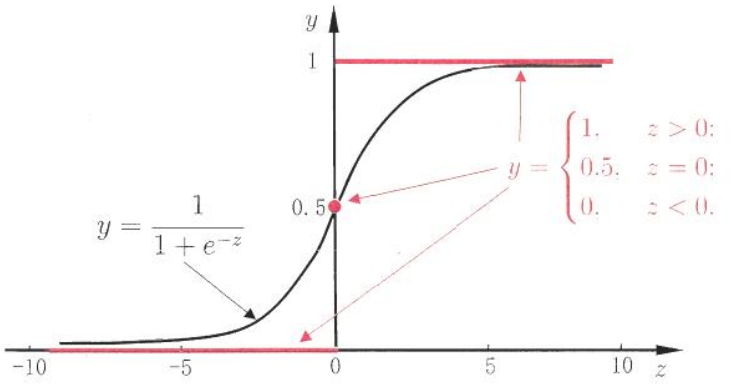
参考链接：
SVM 的推导和理解
支持向量机（SVM）从入门到放弃再到掌握

36、逻辑回归原理

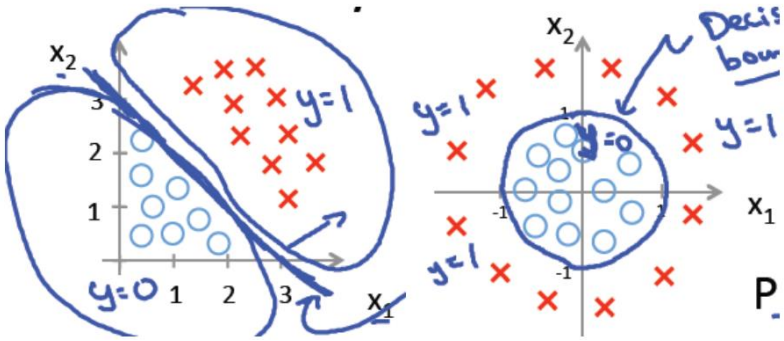
逻辑回归利用回归类似的方法来解决分类问题，假设有一个二分类问题，输出{0,1}，而线性模型的预测值 z 是实数值，我们希望找一个跃阶函数将实数 z 映射为{0,1}，这样我们可以处理分类问题。逻辑回归中使用 sigmoid 函数进行映射：

$$g(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid 函数的图像：



sigmoid 函数中的 z 就是线性函数的 z，因为 g(z)最后输出的时样本类别的概率值，所以我们可以把阈值设为 0.5，g(z)大于等于 0.5 的看作 1，小于 0.5 的看作 0，这样我们就能利用逻辑回归来处理二分类问题了。分类结果就是这样子的。



那我们现在的问题就是怎样计算得到线性函数的模型，也就是我们上文提到输出为 z 的线性模型。为了使模型能分类更准确，我们需要得出一个最优的线性模型函数。也就是下图所示的公式。如何让这个参数达到最优呢？我们就要对每个 x 找到最优的参数θ，这也是我们接下来要求解的问题。

$$\theta_0+\theta_1x_1+,\ldots,+\theta_nx_n=\sum_{i=1}^{n}\theta_ix_i=\theta^Tx$$

此时我们可以先将线性模型和 sigmoid 函数结合起来构造逻辑回归的预测函数:

$$h_\theta(x)=g(\theta^Tx)=\frac{1}{1+e^{-\theta^Tx}}$$

通常求解回归问题(也就是上面的线性问题)的常规步骤为三步:

1.寻找预测函数$h_\theta(x)$

2.构造损失函数$J(\theta)$

3.想办法使得$J(\theta)$函数最小并求得回归参数θ

构造损失函数:

我们要解决的时二分类问题，函数$h_\theta(x)$的值有特殊的含义，它表示结果取 1 的概率，因此对于输入 x 分类结果为类别 1 和类别 0 的概率分别为:

$$P(y=1|x;\theta)=h_\theta(x)$$

$$P(y=0|x;\theta)=1-h_\theta(x)$$

y（标签）要么取 0 要么取 1，这样就可以把两个类别进行整合，得到一个更直观的表达:

$$P(y|x;\theta)=(h_\theta(x))^y(1-h_\theta(x))^{1-y}$$

此时 P 就是某个样本的概率值，我们只要最大化样本的概率就可以得到最好的分类模型了。接下来我们用极大似然函数来求解样本的概率值 P:

$$L(\theta)=\prod_{i=1}^{m}P(y^i|x^i;\theta)=\prod_{i=1}^{m}(h_\theta(x^i))^{y^i}(1-h_\theta(x^i))^{1-y^i}$$

为了简化运算，等式两边都取对数，对数似然函数为:

$$l(\theta)=\log\big(L(\theta)\big)=\sum_{i=1}^{m}(y^i\log(h_\theta(x^i)))+(1-y^i)log(1-h_\theta(x^i))$$

利用梯度下降法求最小值:

$$\theta_j:=\theta_j-\alpha\frac{\delta}{\delta_{\theta_j}}J(\theta)$$

$$\frac{\delta}{\delta_{\theta_j}}J(\theta)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta\big(x^i\big)-y^i)x_j^i$$

则θ更新过程可以写成:

$$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta\big(x^i\big)-y^i)x_j^i$$

这样就可以求出最优参数θ，也就能得到最优的逻辑回归模型。

参考链接：逻辑回归原理