西瓜书复习笔记03

- **线性模型：**
  - 如何判断线性模型：
    主要是看一个乘法式子中自变量x前的系数w，如果w只影响一个x，那么此模型为线性模型。
    或者判断决策边界线是否为直线。
- **线性回归：**
  - 什么是线性回归：
    线性回归试图学得一个线性模型尽可能的准确预测输出值。
  - 特征处理：
    - 连续值不作处理。
    - 离散值可以通过连续化变为连续值。
      （‘高’，‘中’，‘矮’可转化为{1.0， 0.5， 0.0}）
  - 均方误差：

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\left(f\left(\boldsymbol{x}_i\right)-y_i\right)^2$$

  - 最小二乘法：
    基于均方误差最小化来进行模型求解，试图找到一条线使所有样本到直线上的欧式距离之和最小。
  - 最小化损失函数：

$$\arg\min_{(w,b)}\sum_{i=1}^{m}\left(y_i-wx_i-b\right)^2$$

  - 求解过程：
    - 一元线性回归可以将E分别与w和b求导，然后令两个式子为0，可得到w和b的最有闭式解。（最小二乘法）
    - 多元线性回归，同样可以使用最小二乘法，对每个wi求偏导，使所有式子为0，最后解矩阵方程组。但高维数据很难计算。（最小二乘法）
    - 多元线性回归求最小化损失函数最常用的是梯度下降法。
- **梯度下降：**
  - 什么是梯度下降：
    在求解损失函数的最小值时，可以通过梯度下降法来一步步的迭代求解，得到最小化的损失函数和模型参数值。
    沿着负梯度方向搜索最优解。

- 梯度下降流程：
  初始化损失函数的参数w，根据梯度下降的步长更新参数w，重复这个过程一直到无法更新（梯度为0），获得最优解。
- 最优解：
  梯度下降法不一定能求得全局最优解，有可能是一个局部最优解。但是如果损失函数是凸函数那么结果一定是最优解。
- 迭代更新：

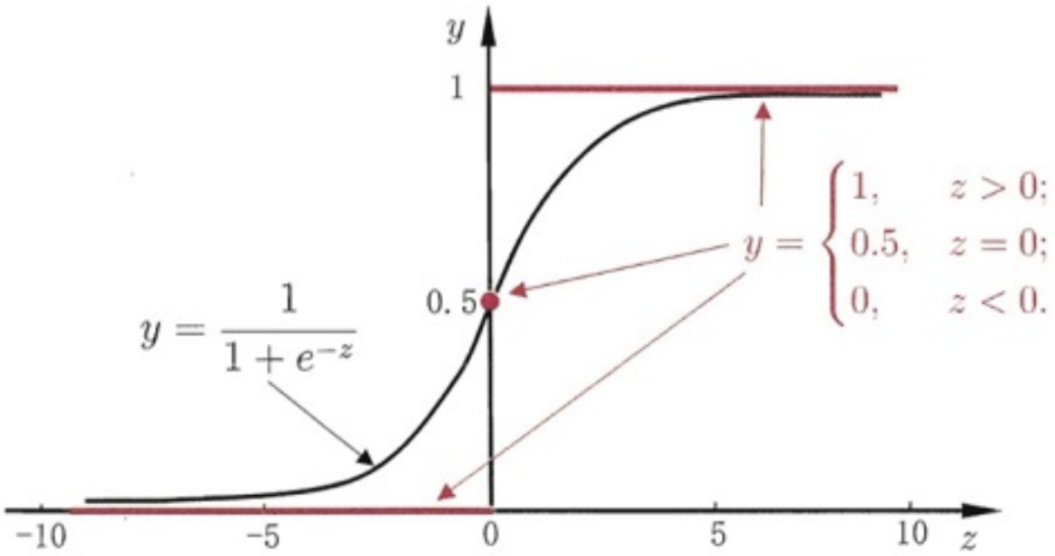
$$w_{new} = w_t - \alpha \nabla E(w,b) = w - a\frac{\partial E(w,b)}{\partial w}$$

  下一时刻的权重等于这一时刻的权重减去学习率乘以损失函数对wi的梯度。所有权值W是一起更新的。
- 学习率：
  α的取值范围(0,1]。
  如果学习率太小，增加收敛的迭代次数，使系统的算力负荷。
  如果学习率太大，那么可能会在最小值旁边震荡，无法收敛。
- 如何去确定学习率：
  - 网格搜索：设定迭代次数，设定几个参数比较结果，但计算量大。
  - 梯度限制：设定大量迭代，当梯度向量变的小于某个值时停止。
- 梯度下降的方向：
  朝最优点方向，也就是斜率的反方向，也就是负梯方向更新。
- 初始值怎么选择：
  离最优点越近越好，减少收敛的迭代次数。
- 几种梯度下降的方法：
  - 批量随机下降：
    每一次迭代时使用所有样本来进行梯度的更新。
    优点：更准确的朝极值所在方向。
    缺点：算力负荷。
  - 随机梯度下降：
    每次迭代使用一个样本来对参数进行更新。在局部最优点梯度仍然不为0，可以跳出局部最小继续搜索。
    优点：训练速度快。
    缺点：准确度下降，可能收敛到局部最优。
  - 小批量梯度下降：
    每次迭代 使用batch_size个样本来对参数进行更新。
- 跳出局部最优点的方法：
  - 模拟退火
  - 神经网络中初始化不同的参数，取其中误差最小的解作为最终参数。

        - 随机梯度下降。
    - 梯度下降的应用：
      深度学习的优化算法，说白了就是梯度下降。每次的参数更新有两种方式。
      第一种，遍历全部数据集算一次损失函数，然后算函数对各个参数的梯度，更新梯度。这种方法每更新一次参数都要把数据集里的所有样本都看一遍，计算量开销大，计算速度慢，不支持在线学习，这称为Batch gradient descent，批梯度下降。
      另一种，每看一个数据就算一下损失函数，然后求梯度更新参数，这个称为随机梯度下降，stochastic gradient descent。这个方法速度比较快，但是收敛性能不太好，可能在最优点附近晃来晃去，hit不到最优点。两次参数的更新也有可能互相抵消掉，造成目标函数震荡的比较剧烈。
      为了克服两种方法的缺点，现在一般采用的是一种折中手段，mini-batch gradient decent，小批的梯度下降，这种方法把数据分为若干个批，按批来更新参数，这样，一个批中的一组数据共同决定了本次梯度的方向，下降起来就不容易跑偏，减少了随机性。另一方面因为批的样本数与整个数据集相比小了很多，计算量也不是很大。
      现在用的优化器SGD是stochastic gradient descent的缩写，但不代表是一个样本就更新一回，还是基于mini-batch的。
      那 batch epoch iteration代表什么呢?
      （1）batchsize：批大小。在深度学习中，一般采用SGD训练，即每次训练在训练集中取batchsize个样本训练；
      （2）iteration：1个iteration等于使用batchsize个样本训练一次；
      （3）epoch：1个epoch等于使用训练集中的全部样本训练一次，通俗的讲epoch的值就是整个数据集被轮几次。
      比如训练集有500个样本，batchsize = 10 ，那么训练完整个样本集：iteration=50，epoch=1.
      batch: 深度学习每一次参数的更新所需要损失函数并不是由一个数据获得的，而是由一组数据加权得到的，这一组数据的数量就是batchsize。
      batchsize最大是样本总数N，此时就是Full batch learning；最小是1，即每次只训练一个样本，这就是在线学习（Online Learning）。当我们分批学习时，每次使用过全部训练数据完成一次Forword运算以及一次BP运算，成为完成了一次epoch。
- **逻辑回归（LR）：**
    - 什么是逻辑回归：
      对于二分类问题，逻辑回归算法就是在样本数据中寻找一个超平面，然后把样本数据分割成不同的类别，并且能对新数据进行分类。
    - 输出标签：
      二分类问题，标签为{0,1}
    - 特征处理：
        - 连续值离散化。
        - 离散值不作处理。
    - 为什么要对特征进行离散化：
      将连续值离散为一系列0、1特征交给逻辑回归模型

- 非线性：
  逻辑回归属于广义线性模型，表达能力受限，将特征离散化为N个后，每个变量都有单独的权重，相当于引入非线性，加大拟合能力。
- 鲁棒性：
  对异常数据有特别强的鲁棒性。例如身高大于例如1.7m为1，那么异常数据5m就不会对模型造成很大干扰。
- 速度快：
  稀疏向量计算内积速度快。

- Sigmoid函数：



  - Sigmoid函数良好的性质：
    Sigmoid可以将样本点映射到（0，1）区间内
    Sigmoid函数连续可导

- 手推LR：
  - 构造假设函数：
    h(x)表示样本预测为正例的概率：

$$h_w(x) = \frac{1}{1+e^{-w^\top x}}$$

    有：

$$\begin{cases} p(y=1|x) = h_w(x) \\ p(y=0|x) = 1 - h_w(x) \end{cases}$$

    合并：

$$P(y|x) = (h_w(x))^y \cdot (1-h_w(x))^{1-y}$$

- 构造损失函数
  服从伯努利分布，用极大似然估计w和b
  极大似然：

$$L(w)=\prod_{i=0}^{m}P\left(y_i|x_i\right)=\left(h_w\left(x_i\right)\right)^{y_i}\cdot\left(1-h_w\left(x_i\right)\right)^{1-y_i}$$

  对数极大似然：

$$\ln L(w)=\sum_{i=0}^{m}y_i\ln\left(h_w\left(x_i\right)\right)+\left(1-y_i\right)\ln\left(1-h_w\left(x_i\right)\right)$$

  求极大似然，等价于最小化：

$$\text{loss}(w)=-\frac{1}{m}\ln L(w)$$

  损失函数：

$$\text{loss}(w)=-\frac{1}{m}\sum_{i=0}^{m}y_i\ln\left(h_w\left(x_i\right)\right)+\left(1-y_i\right)\ln\left(1-h_w\left(x_i\right)\right)$$

  最终：

$$loss(w)=-\frac{1}{m}y_i\ln\left(1+e^{-\hat{y}_i}\right)+\left(1-y_i\right)\ln\left(1+e^{\hat{y}_i}\right)$$

- 损失函数优化
  求解：

$$\boldsymbol{w}^*=\arg\min_{\boldsymbol{w}}\ell(w)$$

  最小化损失函数：
  - 梯度下降（同上）
  - 牛顿法:

$$w^{t+1}=w^t-\left(\frac{\partial^2\ell(w)}{\partial w\partial w^{\mathrm{T}}}\right)^{-1}\frac{\partial\ell(w)}{\partial w}$$

正则化：

- L1正则:

$$\min_w \text{loss}(w) + \alpha\|w\|_1$$

- L2正则:

$$\min_w \text{loss}(w) + \alpha\|w\|_2^2$$

- 为什么逻辑回归加了Sigmoid依然是线性模型，而BP神经网络加了Sigmoid之后变成了非线性模型：
  LR中的Sigmoid函数只是将实值转化为0/1值（只是映射到0-1区间之内）。
  然而BP神经网络中的激活函数确实可以带来非线性。神经网络的每一个节点都是一个LR模型，下一层的变量x可能受上一层的权重影响，因此模型呈现非线性。
  https://www.cnblogs.com/toone/p/8574294.html

- **线性判别分析（LDA）：**
  又称Fisher判别分析
  - 什么是LDA：
    给定一个训练集，将所有的样本投射到一条直线上，同类的尽可能近，异类的尽可能远。当新样本进行分类时，同样投射到这条线上，根据位置判定类别
- **多分类：**
  - OVO（One vs One）：
    有N个类别，不同类别两两训练一个分类器，一共训练c_N^2个分类器。在预测时，将样本交给所有分类器，通过投票法获得最终结果。
  - OVA(One vs All)：
    有N个类别，取第i种类别为正例，其他为反例，一共训练N个分类器。在预测时，将样本交给所有分类器，选择其中概率最大的作为最终结果。
  - Softmax回归：
    - 什么是softmax：
      在多分类任务中，softmax函数可以把输出的值归一化为概率。
    - 样本标签为A_iSoftmax的值：

$$S_i = \frac{e^{A_i}}{\sum_j e^{A_i}}$$

    - Softmax回归模型的假设函数：

$$P(y=i|x;\theta) = \frac{e^{w_i^T x}}{\sum_{j=1}^K e^{w_j^T x}}$$

    - Softmax回归模型的损失函数：

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}\left[\sum_{j=1}^{K} I\left\{y^{(i)}=j\right\}\ln\frac{e^{w_j^T x^{(i)}}}{\sum_{l=1}^{K} e^{w_l^T x^{(i)}}}\right]$$

- **类别不平衡问题：**
  - 欠采样：
    - 直接删除
      删除多数类别的样本，构成1:1的正负样本集
      缺点：容易造成信息丢失
    - 集成思想：
      有6000个数据，5000个负例，1000个正例，把5000个样本分为5份，每份负样例与正样例构成1:1的样本集，使用集成方法训练5个模型，最终再投票决定输出。
      缺点：计算负荷
  - 过采样：
    - 直接抽样：
      在少数类别的样本集中做bootstrap抽样，构成1:1的正负样本集
      缺点：简单复制容易过拟合
    - SMOTE-Regular：
      合成少数类样本。首先使用KNN技术，计算每个少数类样本的K个临近，从K个邻近中随机选择N个样本进行随机插值。构造新的少数类样本集（最好形成正负1:1）。最后新的样本集与原数据进行合成。
  - 阈值移动：
    不对数据集做任何处理，正常的预测，决策过程嵌入阈值
    原决策规则：

$$\frac{y}{1-y} > 1$$

    嵌入后决策规则：

$$\frac{y}{1-y} > \frac{m^+}{m^-}$$

    其中m^+表示样本集中正例的数目m^-表示负例数目
- **正则化：**