目录

[1 理论知识 3](#_Toc37190651)

[2 Normalization 4](#_Toc37190652)

[2.1 介绍 4](#_Toc37190653)

[2.2 基本思想 4](#_Toc37190654)

[2.3 Batch Norm 5](#_Toc37190655)

[2.4 Layer Norm 5](#_Toc37190656)

[3 优化器 7](#_Toc37190657)

[3.1 理论介绍 7](#_Toc37190658)

[3.2 梯度下降法(Gradient Descent) 7](#_Toc37190659)

[3.3 动量优化法 8](#_Toc37190660)

[3.3.1 Momentum 8](#_Toc37190661)

[3.3.2 NAG 8](#_Toc37190662)

[3.4 自适应学习率优化算法 8](#_Toc37190663)

[3.4.1 AdaGrad 8](#_Toc37190664)

[3.4.2 Adadelta 9](#_Toc37190665)

[3.4.3 RMSprop 9](#_Toc37190666)

[3.4.4 Adam 9](#_Toc37190667)

[4 损失函数 10](#_Toc37190668)

[4.1 理论介绍 10](#_Toc37190669)

[4.2 回归问题损失函数 10](#_Toc37190670)

[4.2.1 平/均方误差(L2 loss，MSE) 10](#_Toc37190671)

[4.2.2 绝对值误差(L1 loss，MAE) 10](#_Toc37190672)

[4.2.3 Huber损失函数 10](#_Toc37190673)

[4.3 分类问题损失函数 10](#_Toc37190674)

[4.3.1 交叉熵(Cross Entropy) 10](#_Toc37190675)

[4.3.2 Hinge Loss 11](#_Toc37190676)

[5 激活函数 12](#_Toc37190677)

[5.1 Sigmod 12](#_Toc37190678)

[5.2 Tanh 12](#_Toc37190679)

[5.3 Relu 13](#_Toc37190680)

[5.4 Leaky ReLU 13](#_Toc37190681)

[6 评价指标 15](#_Toc37190682)

[6.1 回归问题 15](#_Toc37190683)

[6.2 分类问题 15](#_Toc37190684)

[7 Dropout 18](#_Toc37190685)

[7.1 介绍 18](#_Toc37190686)

[7.2 实现原理 18](#_Toc37190687)

[8 HMM和CRF对比 19](#_Toc37190688)

[8.1 HMM 19](#_Toc37190689)

[8.2 CRF 20](#_Toc37190690)

[9 正则化 22](#_Toc37190691)

[9.1.1 公式 22](#_Toc37190692)

[9.1.2 L1和L2特点 22](#_Toc37190693)

[10 集成学习 24](#_Toc37190694)

[10.1 Adaboost 24](#_Toc37190695)

[10.2 GBDT 25](#_Toc37190696)

[10.3 XGBoost 25](#_Toc37190697)

[10.3.1 目标函数推导与树结构的选择 26](#_Toc37190698)

[10.3.2 切分点选择算法-近似算法 27](#_Toc37190699)

[10.3.3 稀疏特征的处理 27](#_Toc37190700)

[10.4 Lightgbm 28](#_Toc37190701)

[11 传统模型 29](#_Toc37190702)

[11.1 CART(分类和回归树) 29](#_Toc37190703)

[11.1.1 回归树 29](#_Toc37190704)

[11.1.2 分类树 29](#_Toc37190705)

[12 SVM 30](#_Toc37190706)

[12.1 基本概念 30](#_Toc37190707)

[12.2 线性不可分情况下的软间隔最大化 31](#_Toc37190708)

[12.3 合页损失函数 31](#_Toc37190709)

[12.4 核函数 31](#_Toc37190710)

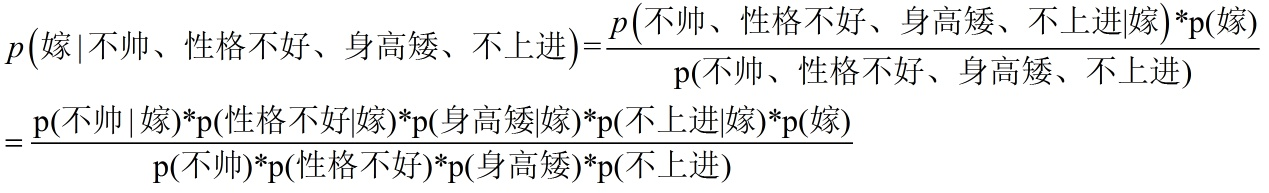
[13 残差网络 32](#_Toc37190711)

[13.1 深模型存在的问题 32](#_Toc37190712)

[13.2 残差网络如何解决 32](#_Toc37190713)

[14 主题模型 33](#_Toc37190714)

# 理论知识

1. 正则化
2. 防止过拟合
   1. 正则-化
   2. Dropout
   3. K-fold
   4. 数据集扩充，尽量拟合真实分布
   5. 设定评估标准(以验证集效果为准)，及时停止
3. 防止梯度消失
4. W2V分布式softmax、下采样
5. 数据扩充方法
   1. 心电图：
      1. 上下颠倒，所有值取负数
      2. 左右整体平移
      3. 加上噪音，生成在1上下浮动的噪音向量Noise,与原始数据进行点积
6. 朴素贝叶斯分类
   1. 贝叶斯公式：
   2. 应用：，右边的概率可以通过统计得到
   3. 朴素贝叶斯：
   4. 
7. 负梯度方向是函数下降最快的方向
   1. 梯度就是一组偏导数组成的向量，指向一定方向
   2. 在一个点，朝每个方向都有一个方向导数
   3. 方向导数最大时的方向就是梯度方向
8. 偏差与方差
   1. 偏差：模型输出与标签值之间的误差，体现模型的拟合能力
   2. 方差：模型的泛化能力，方差度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响
9. 残差网络更深层次的理解
10. 变分自编码器
11. GAN
12. Attention相比于其他模型的优点

# Normalization

## 介绍

目的：使得数据变成独立同分布(白化)

独立：去除数据之间的相关性

同分布：使得所有特征具有相同的均值和方差

问题：如何去除数据之间的相关性？

现有研究只是使其符合某种分布规律，暂未实现独立同分布

独立同分布的好处：

1. 加速训练，降低数据落在饱和区间的概率(梯度消失)
2. 缓解Internal Covariate Shift(ICF)问题
   1. 神经网络层数较多，底层参数的更新会导致后面层输入变化，一层层叠加之后，高层的输入数据变化会非常剧烈，导致高层需要不断的适应底层的变化，难以学习。
   2. 原来每一层之间相互影响比较严重，对每层输入的数据进行独立同分布操作之后，即使之前层参数更新了，对后面层而言依旧是符合独立同分布的数据，变化不大，层与层之间的依赖性降低。

## 基本思想

Norm的基本思想是在数据传递给下一层神经元之前对数据进行**平移和变换**，使其符合某种分布，常用公式如下：

其中

：输入X的均值

：输入X的标准差

：再缩放参数

：再平移参数

Y满足均值为b，方差为的分布

说明：

1. 为什么不直接使用：(0-1分布) ？
   1. 假如使用这种方式，每一层输入的都是0-1分布的数据，前面层无论怎么调整对后面层影响都不大，使得前面层在做无用功，使得整体模型表达能力下降
2. 再缩放参数g和再平移参数b都是可以学习的参数，会随着模型不断训练，所以每一层输入的数据经过Norm之后的分布都各不相同，这样能充分尊重和利用底层神经元的学习成果。

## Batch Norm

公式：

特点：

1. 避免除数为0
2. 对一批(batch size)计算均值和标准差来代替整体的均值和标准差，降低计算复杂度。
3. 每个数据都是由不同特征组成的向量，均值和标准差也是相同维度的向量，代表着在每个特征维度上求得均值和标准差。
4. 使用批均值和标准差替代整体，要求每个batch的数据分布差别不大，都比较贴近整体的分布，否则每个batch适合的分布都不同，会增加训练难度。因此要充分shuffle，并且batch较大比较好。

问题：BN需要保存每个batch的均值和方差，为什么？

1. 在训练时保存每个批次的均值和方差，训练完毕后就可以使用他们计算出整个训练集的均值和方差，用于预测，因为预测时是没有batch要求的。

## Layer Norm

1. BN的一种改进，不再受限于batch的大小。
2. LN是对每一层所有的神经元的输入进行norm，对某一层网络而言，所有神经元的输入都使用相同的均值和标准差进行归一化。但对于不同的数据，其计算的均值和标准差不同。
3. 也就是对于每一层的每个输入，计算其所有特征值(神经元)的均值和方差，然后对其每个特征值(神经元)进行相同的归一化操作。
4. 因为LN是对每一个数据的不同特征求的均值和方差，所以不依赖于batch，预测时也不受影响。

# 优化器

## 理论介绍

优化器的目标：寻找合适的参数，使得损失函数最小

常用的有梯度下降法(Gradient Descent,GD)、动量优化法、自适应学习率，这些都使用的是一阶导数，牛顿优化方法使用的是二阶导数，比如L-BFGS

## 梯度下降法(Gradient Descent)

公式：

1. :学习率，调控参数更新的幅度(步长)
2. ：损失函数对于的导数，也就是梯度，有两个作用：
   1. 确定参数更新方向
   2. 确定更新步长，导数绝对值越大，距离最小值越远，需要调整的幅度就越大

常见的GD方法：

1. 批量梯度下降法(BGD)
   1. 计算全部数据的导数，求均值更新参数
   2. 缺点：收敛速度慢，大数据集计算量太大
2. 随机梯度下降法(SGD)
   1. 每次选取一个样本更新参数
   2. 缺点：容易进入局部最优解，模型效果下降
3. 小批量梯度下降法(mini-batch GD)
   1. 每次计算一个小批次，更新参数
   2. 默认的GD算法

## 动量优化法

参数更新时，梯度连续多次在相同的方向上，那么参数更新幅度变大些；如果梯度方向改变，参数更新幅度变小些。使用之前的梯度来平滑这一次计算出的梯度。

### Momentum

公式：

1. ：动量，保存着之前梯度的累积信息
2. ：动量因子

### NAG

公式：

1. 思想：既然下次参数更新会到达的位置附近，那为什么不直接计算该位置的梯度，来平滑这次的参数更新，即计算处的梯度。相当于提前知道下次的更新方向。

## 自适应学习率优化算法

自适应的调整学习率大小，提高训练速度

### AdaGrad

公式：

1. r是梯度的不断累积值，初始值为0，训练一段时间后，r变大，学习率降低。
2. 缺点：仍需人工设定初始学习率，后期r较大，停止学习。

### Adadelta

### RMSprop

### Adam

结合Momentum和RMSprop，累积动量的同时调整学习步长

偏差校正：

偏差校正原因：一般

刚开始时,一开始的梯度会非常大。加上偏差校正可以减少一开始时对参数和的依赖性。

# 损失函数

## 理论介绍

1. 损失函数用于计算评估模型预测值与真实标签的差异。
2. Loss function针对的是单一数据的计算结果，cost function指的是整个数据集的平均损失，也就是loss function的均值，优化策略优化的是cost function。

## 回归问题损失函数

### 平/均方误差(L2 loss，MSE)

优点：二次方程，没有局部最小值，只有最小值，容易收敛

缺点：有平方，对异常值鲁棒性较低。次数越高，计算结果就越与较大的值有关，而忽略较小的值

### 绝对值误差(L1 loss，MAE)

优点：对异常值鲁棒

缺点：数学方程中处理绝对或模数运算符并不容易

### Huber损失函数

结合MSE和MAE两者的特点，误差较小时使用二次方程，较大时使用一次方程。

## 分类问题损失函数

### 交叉熵(Cross Entropy)

熵的公式：

1. 是某一类数据所占比例，也可以说是概率
2. 对于概率分布而言，熵越大，分布越不稳定，我们的目标就在于最小化熵

对于二分类模型，输出1个概率值，我们可以定义：

1. 为模型输出的1的概率，为模型输出的0的概率，为真实的标签值
2. 则：
3. 合并两种情况
4. 取对数，加负号就得到损失函数的形式。

对于多分类问题，假设有k个类，模型输出k个类的概率值，真实标签里每个类别对应的值为，则

### Hinge Loss

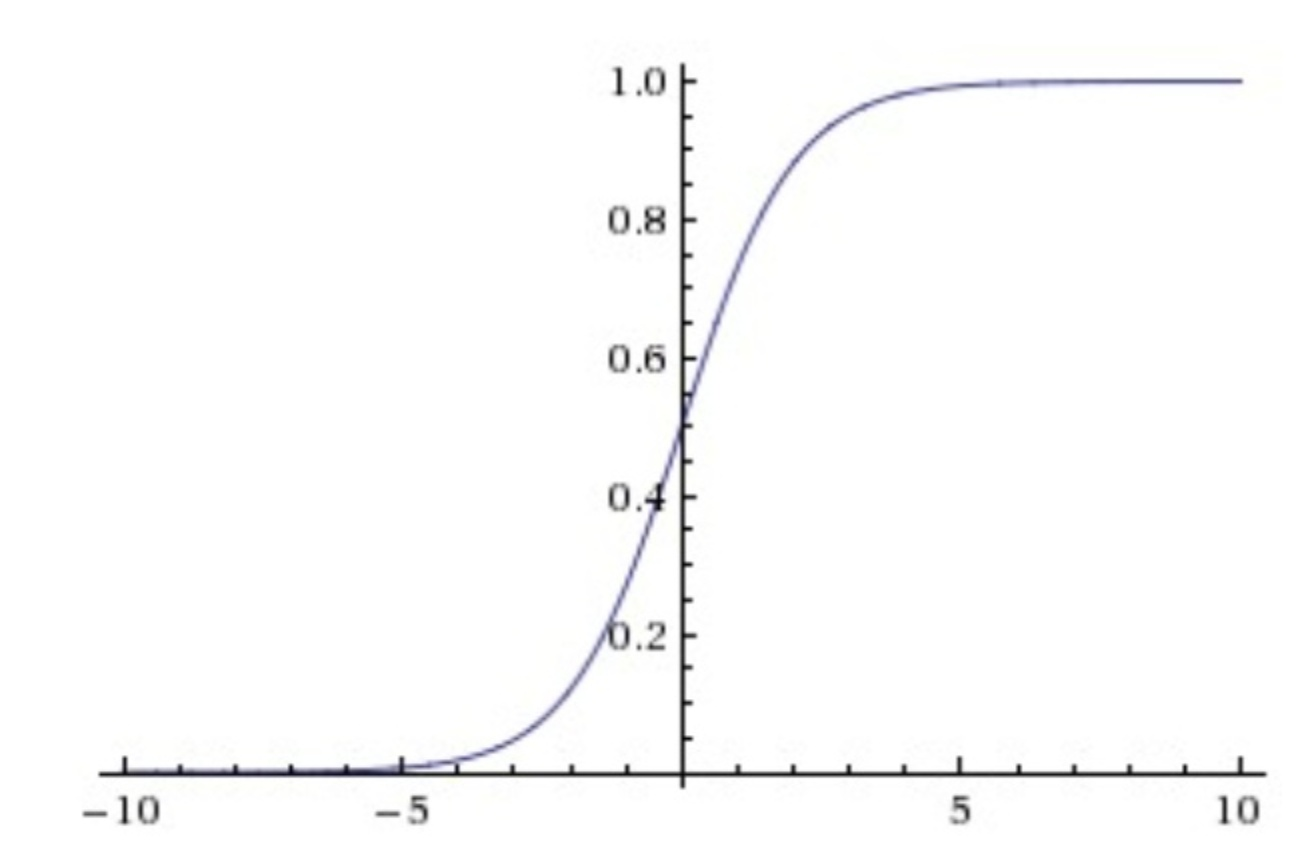
专用于二分类，标签值为1/-1：

当时，会有很大的损失，也会有较小的损失，对置信度不高的情况也加以惩罚调整。时也是一样的规律。

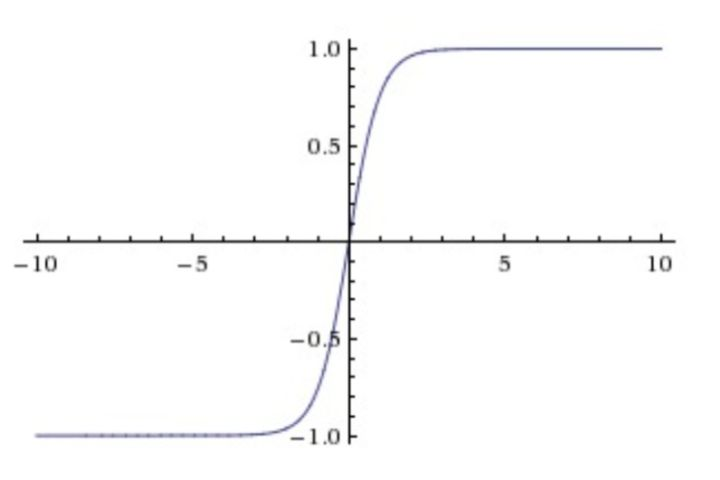
# 激活函数

增加模型的表征能力，线性模型表达能力较弱

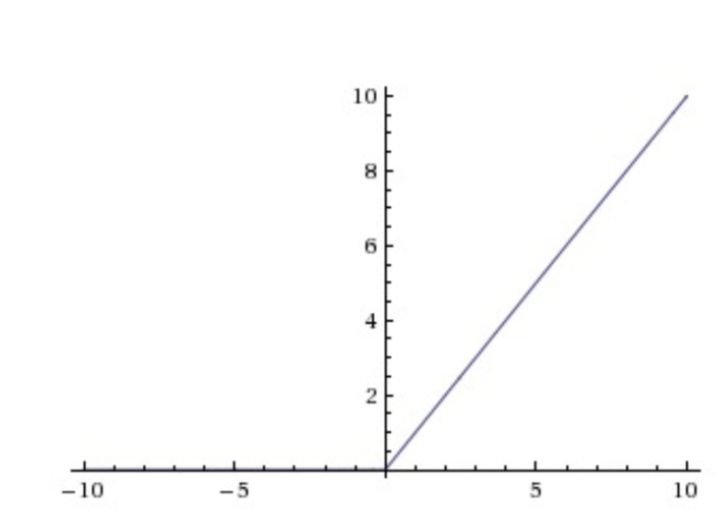
## Sigmod

1. 公式：
   1. 
2. 取值范围：0~1，导数范围0~1/4
3. 缺点
   1. 饱和区间太大，容易梯度消失
4. 常用于最终的分类层，将输出映射到0~1

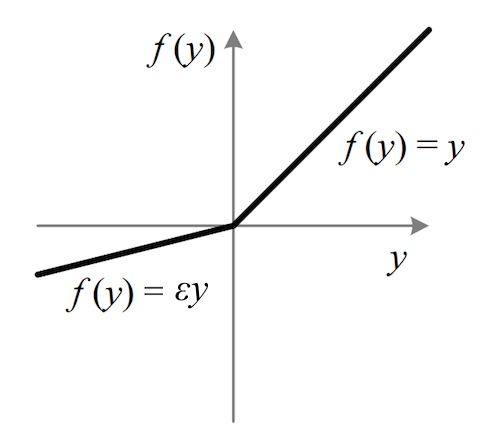
## Tanh

1. 公式：
   1. 
2. 取值范围-1~1，导数范围0~1
3. 相比sigmod，输出是0中心，效果会好些
4. 缺点
   1. 饱和区间太大，容易梯度消失
5. LSTM中使用tanh不用relu?
   1. 可以用，效果还会更好
   2. CNN、DNN这些深度网络，梯度消失主要原因在于激活函数的导数累积造成的，所以使用Relu会有很好的改进效果
   3. RNN中梯度消失主要是参数累积造成的，和激活函数关系不大，所以可以用tanh，也可以用Relu

## Relu

1. 公式：
   1. 
2. 梯度：
3. 缺点：
   1. 神经元坏死，当某个神经元的输入变为负数时，该神经元就再也无法学习
   2. 失去了非线性映射能力，在深网络中没关系，在浅层网络中会使模型表征能力下降
4. 优点：
   1. 梯度饱和区间较小，只在输入为负数的情况下饱和
   2. 导数计算简单，就两种可能

## Leaky ReLU

1. 公式：，当x<0时，存在一个很小的梯度
   1. 
2. 弥补Relu负数情况下导数为0，不再学习的缺点

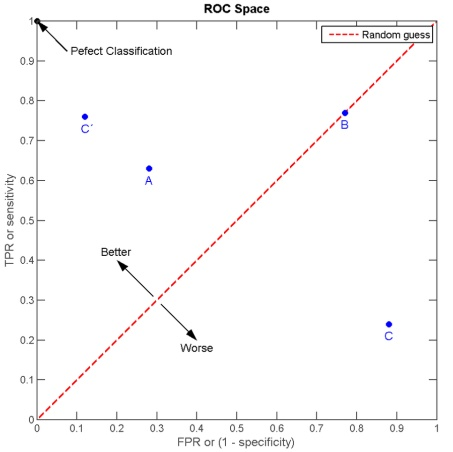
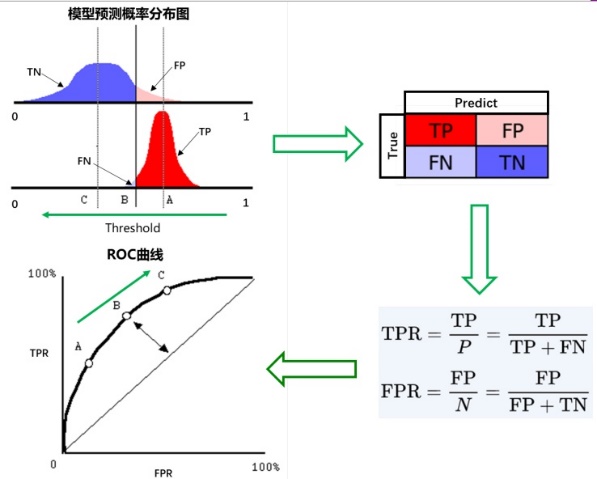
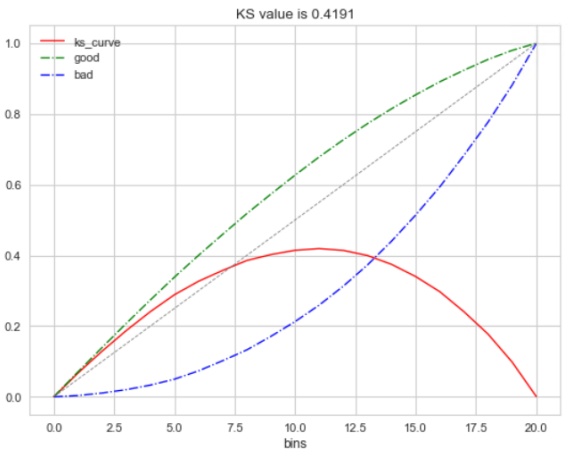
# 评价指标

## 回归问题

回归问题输出的是具体的数值，常见指标有：

1. 平均绝对值误差MAE(Mean Absolute Error)
2. 均方误差MSE(Mean Squared Error)
3. 均方根误差RMSE

## 分类问题

1. 精度Accuracy
   1. 预测正确的样本占总样本比例
2. 混淆矩阵
   1. T代表预测正确，F预测错误；P代表预测结果为正例，N代表预测结果为负例
   2. 
3. 准确率Precision
   1. 预测结果为正的样本中的正确率
4. 召回率Recall
   1. 真实标签为正的样本被预测正确的概率
   2. 综合衡量正确率和召回率
5. ROC和AUC
   1. ROC是一个曲线，X轴是FPR，y轴是TPR
   2. FPR是负样本被预测错误的概率
   3. TPR是正样本被预测正确的概率，即Recall
   4. 理论上来说，Y越大，效果越好
      1. 
   5. AUC是ROC曲线覆盖面积，AUC面积越大，模型效果越好
      1. 
   6. ROC和AUC都与样本比例无关，XY值都是根据单一类样本算出来的，所以比较适合不均衡数据的评估。
6. KS
   1. 以阈值为X轴，TPR和FPR为Y轴
   2. 我们希望TPR越大越好，FPR越小越好，所以KS=TPR-FPR越大越好
   3. 

# Dropout

## 介绍

目的：缓解过拟合情况

原因：

1. 增加了随机性，每次都在训练不同的网络
2. 减少了不同层神经元之间的复杂共适应性

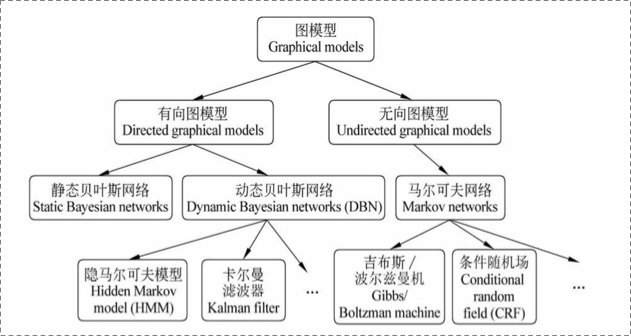
## 实现原理

未加dropout层：

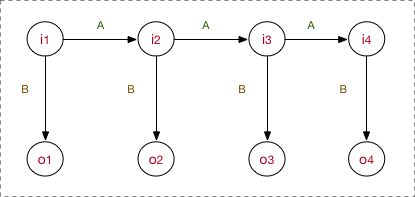
增加dropout层：

1. 使用分布函数随机生成0或1，对输入神经元的数据进行调整：清0或者不变
2. 未加dropout之前，梯度与有关
3. 加了dropout，梯度为，梯度为0就相当于不更新该神经元

# HMM和CRF对比

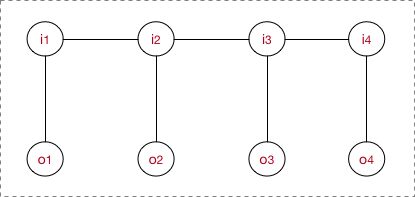


## HMM



1. HMM 5要素
   1. 观测序列，观测集合M
   2. 隐藏状态序列，隐藏状态集合N
   3. 状态转移矩阵
   4. 观测概率矩阵
   5. 初始隐藏状态分布
2. HMM是有向概率图模型
   1. 模型预测目标
   2. 两者的区别
3. HMM是生成式模型
   1. 在训练时，输入的是观测序列，模型通过最大化来学习A、B、的参数
   2. 求的是联合分布
   3. 使用时，对于一个给定的观测序列，可以先利用A、B、计算和，然后通过贝叶斯公式计算，进而从中选择最可能的。
4. HMM训练过程？
   1. 初始化A、B、，不断迭代更新

## CRF



1. CRF是无向图模型，满足马尔科夫假设的随机场
   1. 是一个复杂函数，用于将后面的值归一化到0~1
   2. T个状态，K个特征函数，每个状态的概率都是这K个特征函数的加权和，是每个特征函数的权重
   3. 无向图模型，每个状态都要考虑到前后状态，满足马尔科夫假设，所以只考虑相邻的前后状态
2. CRF是判别式模型
   1. 模型训练时，计算的是的各个特征函数的权重
3. 特征函数一般分为两部分
   1. 是转移特征，是状态特征(观测概率特征)
4. CRF训练过程
   1. 定义特征函数
   2. 训练模型，确定特征函数权重

# 正则化

### 公式

原来的损失函数：

L1正则化(lasso回归)：

L2正则化(岭回归)：

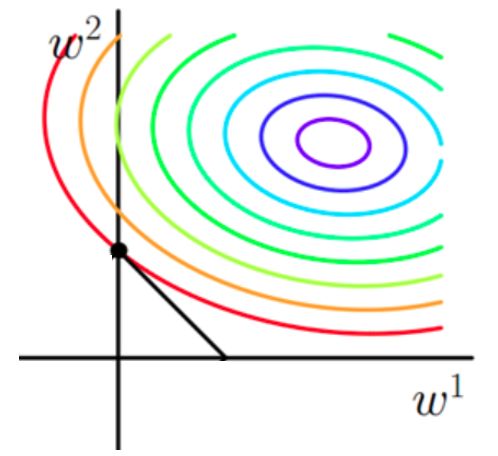
L1和L2能够实现过拟合的原因：

1. 结构风险最小化：在经验风险最小化的基础上（也就是训练误差最小化），尽可能采用简单的模型，以此提高泛化预测精度。
2. L1会使得部分参数变为0，参数变得稀疏；L2会使得参数接近于0，变得平滑；都会使得模型结构变得简单。

### L1和L2特点

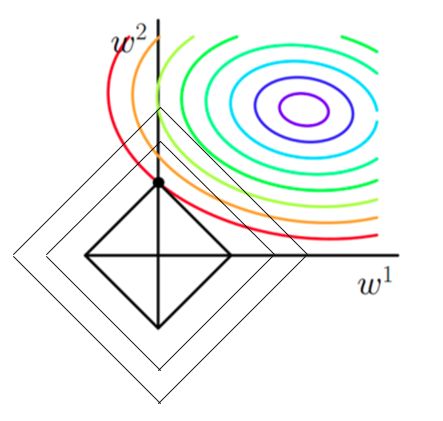
假设有两个参数，L的曲线如下图，

1. 越靠近中心点L的值越小
2. 等高线上的L值是相同的



的曲线是一个菱形，如下图所示

1. 菱形越往外F越大
2. 菱形边上的点F相同

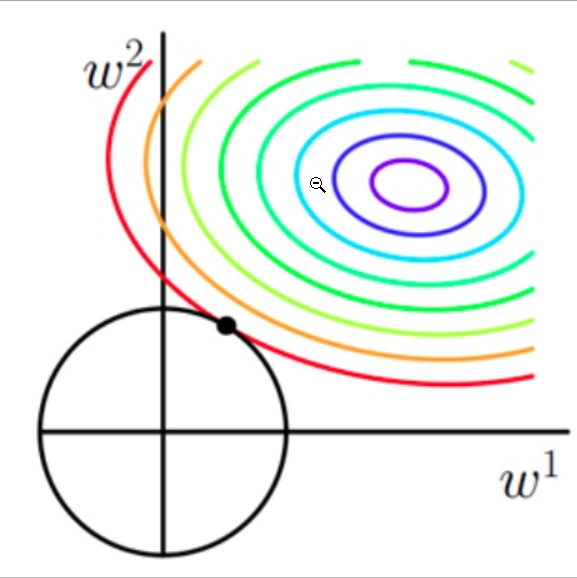


对于而言，要最小化，就得找到合适的L和F。

1. 假设合适的F已知，则F对应菱形与L曲线等高线的切点是最优的参数取值
2. 因为F值是固定的，除了切点以外的其他与F右上边相交的点的L都比切点大，因为他们的等高线都靠外。
3. 而这个切点通常是坐标轴上的点，也就是使得某个参数为0的点

通过上述说明，可知，L1正则会使得某些参数为0，也就是**使得参数变得稀疏**。

L2也和L1类似，不过F的形状变为了圆形，而圆形与L曲线的切点通常接近坐标轴，但不易在坐标轴上，所以L2正则化会使得参数接近0，比较**平滑**。



# 集成学习

集成学习分为两种：

1. Bagging，并行模式，对数据进行抽样训练弱分类器，弱分类器投票：随机森林
2. Boosting，串行模式，弱分类器之间有前后关联，共同组合成强分类器：Adaboost、GBDT、Xgboost、
3. Stacking，先用特征训练弱分类器，然后将特征和弱分类器的结果作为新的特征再训练其他的集成学习模型

## Adaboost

自适应增强模型，基本思路是前一个基本分类器被错误分类的样本的权值会增大，而正确分类的样本的权值会减小，并再次用来训练下一个基本分类器。达到停止条件后再对所有弱分类器进行加权，组合成最终的强分类器。

1. 初始化数据权重
2. 对于
   1. 训练一个基本(弱)分类器
   2. 计算该基本分类器的误差，即错误分类数据的权重和
   3. 计算该基本分类器的权重
   4. 更新数据权重
   5. **弱分类器的训练就是根据误差决定的，每个切分点的选择都是使误差最小化**
3. 这样就得到了m个基本分类器以及其对应的权重，最终的强分类器

缺点：

1. 只适合处理二分类问题，才能计算误差率
2. 对异常值敏感

特殊说明：

1. 数据权重只是用来计算该基本分类器的权重，并没有影响到的训练

## GBDT

与Adaboost相比的优点：

1. 可以处理回归问题
2. 加入了正则化，减少了噪音的影响，减少过拟合

用之前弱分类器结果和预测值之间的差值作为下个弱分类器(决策树)的学习目标。

1. 初始化回归树
2. 对于
   1. 计算残差
   2. 使用数据训练回归树
   3. 更新
3. 最终组合成的强分类器

## XGBoost

是GBDT的一种优化与实现，也是一种加法模型。

1. 使用二阶导数，并加入了正则化
2. 采用近似算法寻找切分点

### 目标函数推导与树结构的选择

XGBoost推导与GBDT相似：

1. ，引入了正则化，表示模型的复杂度
   1. 表示在第m个分类器中，输入对应的预测值
   2. 表示叶子节点j的值
2. 在训练第m个分类器时，前m-1个树已经确定，只考虑当前分类器的偏差与模型复杂度
   1. 因为前m-1个树已经确定，所以和都是确定常量
   2. 二阶展开为
   3. 其中都是确定常量
   4. 去掉常量后，
3. 将上述公式进行整理，对第m个树而言，其目标函数：
   1. 上述公式是针对每个数据计算的，转换成针对每个叶子节点进行计算，得到下面公式
   2. (进行简化)
4. 对于第m个树，我们已经知道其目标函数，那么我们可以求得其**目标函数的最优值**，代表了这棵树的最优性能，用来衡量一个树结构的好坏。此时的**最优性能并不是树的真实性能**。
   1. 导数为0时，最优情况下的
   2. 最优情况下的
5. **上述公式只是求某个树结构的最优性能，用来选择树结构，并不是说树的节点值就是**
6. 有了上述公式，我们就可以衡量每个切分点切分前后的性能差，从而选择最优的切分点

### 切分点选择算法-近似算法

基本思想：

1. 对每个特征，对其所有取值进行排序，选择一些分位点，作为待选切分点
2. 并行计算每个特征在每个切分点上的收益，选择最优的切分点

切分点选择：

1. 对于某个特征，重排序后，每个数据都对应一个下标r
2. 确定参数，采集下标在的特征值作为切分点
3. N是数据个数

通过加权算法确定采样点

### 稀疏特征的处理

一些数据缺少部分特征值

1. 先对非缺失数据进行划分
2. 将缺失数据全划分到左子节点计算Score
3. 将缺失数据全划分到右子节点计算Score
4. 选择最优的Score，并将缺失情况设定为指向相应的左/右子节点
5. 如果训练时没有缺失，测试时有缺失值，则缺失数据默认分到右子节点
6. Xgboost里是将特征值0和np.nan都当作了缺失值处理

## Lightgbm

# 传统模型

## CART(分类和回归树)

### 回归树

1. CART是二叉树，每个节点只有两个分支
2. 以均方误差的减少量作为切分点选择的标准
3. 在每个节点
   1. 遍历其所有特征j
      1. 遍历j的所有取值情况s
      2. 以s为切分点将数据分为两个子节点
      3. 计算两个子节点的均方误差和原节点的均方误差，计算出均方误差的减少量
      4. 选择最优的切分点s作为该特征j的最优情况
   2. 选择均方误差减少最多的一组做为该节点的切分点
4. 重复2直到停止
5. 对于每个叶节点，其所有样本的标签值的均值作为其对应的输出
6. CART树可表示为

### 分类树

1. 以基尼指数作为切分点选择标准，与熵类似
   1. ，是k类样本所占比例
2. 切分点的寻找方法与回归树类似，遍历所有特征的所有取值情况，寻找最优切分点

# SVM

## 基本概念

1. 输入数据：
2. 决策函数：
3. 超平面：，在特征空间中将样本正确分为两类
4. 函数间隔
   1. 超平面关于样本点的函数间隔
   2. 超平面关于样本集合T的函数间隔
   3. 超平面=超平面，但其函数间隔翻倍，所以对进行约束，使得，此时函数间隔就是几何间隔
5. 几何间隔
   1. 几何间隔
6. SVM的目标就是使得超平面的几何间隔最大
   1. 最大化max
      1. 同时满足： ，
   2. 等价于max
      1. 同时满足： ，
   3. 的取值不影响上述最优化问题，所以取，并且max 等价于min ，可得
      1. min
   4. 通过拉格朗日函数求解上述函数
7. 支持向量
   1. 距离超平面最近的点，也就是是得的点
   2. 支持向量所在超平面和叫做间隔边界
   3. 在决定超平面时只有支持向量在起作用，移动其他向量对模型超平面的选择不影响

## 线性不可分情况下的软间隔最大化

1. 对于样本集T，大部分样本是线性可分的，但有少量异常点线性不可分
   1. 即不满足
2. 为每个样本引入一个松弛变量，尽量避免误分类
3. SVM最优化问题就变成了使得软间隔最大化
   1. min
   2. 是惩罚项，C值大时对误分类惩罚较大，C值小时对误分类惩罚较小
4. 最优化后得到参数，确定决策函数
5. 此时的支持向量就比较复杂，分布比较乱

## 合页损失函数

SVM最优化问题

* 1. min

等价于min

其中左半部分就是合页损失函数：

1. 合页损失函数只有分类正确，且为1/-1时损失才为0，即使分类正确但确信度不高也会有损失，所以合页损失函数要求较高。

## 核函数

1. 对于非线性分布的数据集，存在一个超曲面能够将对数据集进行分类
2. 通过核函数将输入映射到一个高维空间，在高维空间内线性可分，能够找到一个超平面进行分类。

# 残差网络

## 深模型存在的问题

1. 梯度消失、爆炸
2. 网络退化

## 残差网络如何解决

1. 残差网络：
2. 加入，就相当于恒等映射
3. 展开：
4. 所以残差网络提供了从任意低层(k)传递给高层(t+1)的能力，缓解了网络退化的问题
5. 同时在计算梯度时，由原来的单一反向传播途径转为了两个反向传播途径，即使有一个途径梯度消失，还存在另外一个恒等映射途径将梯度传递回去。

# 主题模型