集成学习

Round V 罗磊

上海机器学习研讨会

2016 年 5 月 7 日

为什么使用集成学习

It Works!

- 从臭皮匠到诸葛亮
- 从精英到智库
- 随机森林、boosting、dropout
- netflix prize, kddcup, imagenet, kaggle...

学习目标：

- 知道各个模型是怎么做的
- 了解集成学习为什么能够work

目录

提纲

最小二乘的偏倚方差分解

对模型进行误差分析：

$L = (y(x) - t)^2$

$E(L) = \int\int (y(x) - t)^2 p(x, t) dxdt$

根据Euler-Lagrange公式

https://en.wikipedia.org/wiki/Euler-Lagrange_equation

$2\int (y(x) - t)p(x, t)dt = 0$

得：$y(x) = \int tp(t|x)dt = E_t(t|x)$

最小二乘的偏倚方差分解(续)

$h(x) = E(t|x) = \int tp(t|x)dt$

$$
\begin{aligned}
E(L) &= \iint (y(x)-t)^2 p(x,t)dxdt \\
&= \int \{y(x)-h(x)\}^2 p(x)dx + \\
&\quad \iint \{h(x)-t\}^2 p(x,t)dxdt
\end{aligned}
$$

说明

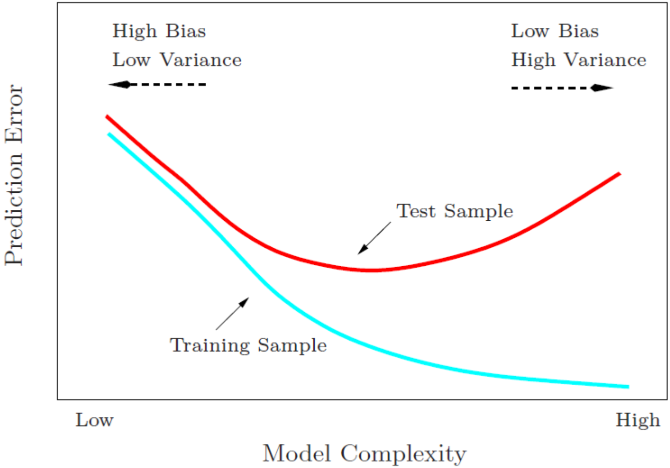
第二部分与模型无关，反映的是数据本身的问题，模型优化的只能是第一部分

最小二乘的偏倚方差分解(续)

$$
\begin{aligned}
& \{y(x;D) - h(x)\}^2 \\
= \ & \{y(x;D) - E_D[y(x;D)] + E_D[y(x;D)] - h(x)\}^2 \\
= \ & \{y(x;D) - E_D[y(x;D)]\}^2 + \{E_D[y(x;D)] - h(x)\}^2 + \\
& 2\{y(x;D) - E_D[y(x;D)]\}\{E_D[y(x;D)] - h(x)\}
\end{aligned}
$$

$$
\begin{aligned}
& E_D\{y(x;D) - h(x)\}^2 \\
= \ & \{E_D[y(x;D)] - h(x)\}^2 + E_D\{y(x;D) - E_D[y(x;D)]\}^2 \\
= \ & (bias)^2 + variance
\end{aligned}
$$

最小二乘的偏倚方差分解(续)



提纲

Bagging

算法：Bagging

输入：训练集D，基学习算法ζ，训练轮数T

for t = 1,2,...T do

从训练集D中随机选取m个样本，组成新的子集合D_t

训练模型$h_t = \zeta(D_t)$

end for

输出：$H(x) = argmax_{y \in Y} \sum_{t=1}^{T} \mathbb{I}(h(x) = y))$

Bagging降低错误的方差

$$
\begin{aligned}
& E_x[y(x) - h^\star(x)]^2 \\
= \quad & E_x[y(x) - H_{bagging}(x) + H_{bagging}(x) - h^\star(x)]^2 \\
= \quad & E_x[y(x) - H_{bagging}(x)]^2 + E[H_{bagging}(x) - h^\star(x)]^2 \\
\geqslant \quad & E_x[y(x) - H_{bagging}(x)]^2
\end{aligned}
$$

说明

通过bagging的方法，可以降低模型的方差，提升模型效果

模型的多样化

$$\begin{aligned} h_m(x) &= y(x) + \xi_m(x) \\ E[\{h_m(x) - y(x)\}^2] &= E[\xi_m(x)^2] \end{aligned}$$

M个模型的平均误差为：

$$E_{average} = \frac{1}{M}\sum_{m=1}^{M} E[\xi_m(x)^2]$$

$$\begin{aligned} E_{bagging} &= E_x[\left\{\frac{1}{M}\sum_{m=1}^{M}(h_m(x) - y(x))\right\}^2] \\ &= E_x[\left\{\frac{1}{M}\sum_{m=1}^{M}\xi_m(x)\right\}^2] \end{aligned}$$

模型多样化（续）

假设:

$$
\begin{aligned}
E_x[\xi_m(x)] &= 0 \\
E_x[\xi_m(x)\xi_l(x)] &= 0 \qquad l \neq m
\end{aligned}
$$

有：

$$E_{bagging} = \frac{1}{M} E_{average}$$

说明

$\frac{1}{M}$是理想情况，但模型相关性越低越接近理想情况

关于bagging的思考

- 为什么bagging的都是弱学习器（如决策树)?
- 随机森林为什么有效?
- 为什么不容易过拟合?

提纲

偏倚和方差

bagging

boosting

dropout

References

Boosting

- Boosting表示的是一系列方法
- 周老师书中有adboost的详细推导
- 这里介绍Gradient Boosting Machine

从数值优化说起

损失函数：

$$L(f)=\sum_{i=1}^{N}L(y_i,f_i(x))$$

使用数值优化方法优化上式，得最优解：

$$\widehat{f}=argmin_f L(f)$$

优化方法上采用迭代优化的方法：

$$f_m=\sum_{i=1}^{m}h_i$$

其中f_m是迭代进行到第m轮的模型，h_i是第i轮迭代的模型变化增量

从数值优化说起（续）

常见的优化方法：梯度下降

$$h_m = -\rho_m g_m$$

梯度：

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

步长：

$$\rho_m = argmin_{\rho} L(y, f_{m-1} - \rho g_m)$$

更新模型：

$$f_m = f_{m-1} - \rho_m g_m$$

gradient boosting

函数空间上的数值优化

- 把几何空间的概念换成函数空间
- 上述流程中，$f_m = \sum_{i=1}^{m} h_i$，h_i从模型变化量变成了一个弱学习器(tree)
- 在几何空间中$f_m = f_{m-1} - \rho_m g_m$，在函数空间中$f_m = f_{m-1} + \rho_m Baselearner_m$

说明

用学习器逼近梯度（$BaseLearning_m$逼近$-g_m$）

用学习器逼近梯度

梯度：

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

以最小二乘为例：

$$L = \frac{1}{2}\sum_{i=1}^{N}(y_i - f_{m-1}(x_i))^2$$

对函数求导，负梯度为：

$$g_{im} = y_i - f_{m-1}(x_i)$$

$BaseLearner_m$的训练目标为(x_1, g_{1m}), (x_2, g_{2m}),... (x_n, g_{nm})

$$\Theta_m = argmin_{\Theta}\sum_{i=1}^{N}(-g_{im} - T(x_i; \Theta))^2$$

Gradient Boosting Machine

算法：Gradient Boosting Machine

输入：训练集D，基学习算法ζ，训练轮数T

1)在训练集D上训练初始化$f_0 = \zeta$

2)for t = 1,2,...T do

a)for i = 1,2,...N

$$g_{it} = y_i - f_{t-1}(x_i)$$

b)训练新的ζ_t拟合(x_i, g_{it})

c)设置步长

$$\rho_t = argmin_\rho L(y, f_{t-1} + \rho\zeta_t)$$

d)更新模型$f_t(x) = f_{t-1}(x) + \rho_t\zeta_t$

end for

输出：$f(x) = f_T(x)$

开源实现

XGBoost

- 支持随机森林
- 支持Gradient Boosting Machine
- https://github.com/dmlc/xgboost
- 随机梯度下降暗含的是泰勒一阶展开，xgboost是二阶展开的实现
- 自定义目标函数扩展方便

关于Boosting未提到的问题

- Boosting正则化，Shrinkage，借用random forest的思想
- Boosting的理论基础，为什么不容易overfitting？
  - 偏倚方差
  - max margin

提纲

为什么用dropout

- 神经网络是典型的低偏倚，高方差的模型
- 神经网络计算量大(deep)，如果直接采用bagging的方法训练和测试时间都不能接受
- dropout可以避免过拟合

什么是dropout

- 假设神经网络共有n个隐藏的神经元（不管层数）
- 对于每个训练用例，每个隐藏单元都以一定概率p被丢弃
- 假设p=0.5，理论上共训练了2^n个模型，而参数个数仍维持在$O(n^2)$级别（甚至更少）

说明

dropout是非常多的模型的组合，这些模型共享参数，每个模型被训练次数很少

什么是dropout(续)

没有使用dropout的前馈神经网络：

$$\begin{aligned} z^{l+1} &= W^{l+1} y^{l} + b^{l+1} \\ y^{l+1} &= f(z^{l+1}) \end{aligned}$$

使用dropout的前馈神经网络：

$$\begin{aligned} r^{l} &\sim Bernoulli(p) \\ \widetilde{y}^{l} &= r^{l} * y^{l} \\ z^{l+1} &= W^{l+1} \widetilde{y}^{l} + b^{l+1} \\ y^{l+1} &= f(z^{l+1}) \end{aligned}$$

测试阶段的dropout

- 近似模型$W_{test}^l = pW^l$
- bagging和boosting通常比较慢，而dropout很快
- 对比随机选择K个dropout的模型用于测试阶段（效果接近当K较大时）

为什么dropout会有效

解释一：

- 消除模型参数之间的co-adaptations
- 隐含节点的参数没法因为dropout没法依赖与其他参数对自己进行修正

为什么dropout会有效(续)

解释二，先以线性回归为例：

$$||Y - XW||^2$$

引入dropout：

$$min_w E_{R \sim Bernoulli(p)}[||Y - R * XW||^2]$$

期望展开，等价于：

$$||y - pXw||^2 + p(1-p)||\Gamma w||^2，其中\Gamma = (diag(X^T X))^{1/2}$$

另 $\widetilde{w} = pw$

$$||y - X\widetilde{w}||^2 + \frac{1-p}{p}||\Gamma \widetilde{w}||^2$$

为什么dropout会有效(续)

解释二

- 一种特殊形式的L2-Norm，正则化
- 参数服从高斯分布（贝叶斯）
- 神经网络并不能推导的如此规范形式

提纲

References

- 周志华老师《机器学习》第八章
- 《Pattern Recognition and Machine Learning》第3.2节，第14章
- 《The Elements of Statistical Learning》第二版，第8.7节，第10章，第15章
- 《A Short Introduction to Boosting》Freund, Yoav Schapire, Robert E Avenue, Park
- 《Greedy function approximation: a boosting machinegradient》Friedman, JH
- 《Boosting the margin: a new explanation for the effectiveness of voting methods》Schapire, Robert E.Freund, YoavBartlett, PeterLee, Wee Sun
- 《Boosting 25 Years》ppt，Zhou, Zhi-hua
- 《Introduction to Boosted Trees》ppt, Tianqi Chen
- 《Improving neural networks with dropout》Srivastava, N
- 《Introduction to the Calculus of Variations》Peter J. Olver

完

Q & A