第 13 章　Boosting 算法

13.6　梯度提升算法

梯度提升算法[25]是 boosting 算法的另一种实现，是最速下降法与广义加法模型的结合。训练时用最速下降法逐步优化损失函数，得到每一个弱学习器及其系数。与 AdaBoost 算法类似，强学习器也是多个弱学习器的加权组合，不同的是训练时没有为样本构造权重，而是用损失函数对强学习器（强学习器的输出值）的梯度值构造样本的标签值，训练每一个弱学习器，从而保证每增加一个弱学习器可以导致损失函数值下降。

13.6.1　梯度提升框架

梯度提升算法也采用了加法模型，训练时逐步最小化损失函数。在第 2.2.1 节介绍了最速下降法的原理，优化变量沿着负梯度方向迭代时目标函数值下降。梯度提升算法将强学习器对训练样本的输出值作为损失函数的优化变量，对其求导，得到训练当前弱学习器时的拟合目标，沿着这个方向，损失函数值下降。与 AdaBoost 算法类似，梯度提升算法依次优化弱学习器的参数和系数，训练第 m 个弱学习器时优化的目标为

$$\left(\mathbf{a}_m,\beta_m\right)=\arg\min_{\mathbf{a},\beta}\sum_{i=1}^{N}L\left(y_i,F_{m-1}\left(\mathbf{x}_i\right)+\beta h\left(\mathbf{x}_i;\mathbf{a}\right)\right)$$

其中 $\left(\mathbf{x}_i,y_i\right)$ 为训练样本向量，N 为训练样本数，L 为损失函数，$F_{m-1}\left(\mathbf{x}\right)$ 为之前迭代得到的强学习器，$h\left(\mathbf{x};\mathbf{a}\right)$ 为当前要训练的弱学习器，$\mathbf{a}$ 是弱学习器的参数，β 为弱学习器的系数。训练得到弱学习器之后，更新强学习器的预测函数

$$F_m\left(\mathbf{x}\right)=F_{m-1}\left(\mathbf{x}\right)+\beta_m h\left(\mathbf{x};\mathbf{a}_m\right)$$

接下来重复这一过程，完成所有弱学习器及其系数的训练。弱学习器参数的最优解为

$$\mathbf{a}_m=\arg\min_{\mathbf{a},\beta}\sum_{i=1}^{N}\left[-g_m\left(\mathbf{x}_i\right)-\beta h\left(\mathbf{x}_i;\mathbf{a}\right)\right]^2$$

即用当前的弱学习器来拟合损失函数对强学习器的负梯度值，该梯度值为

$$g_m\left(\mathbf{x}\right)=\mathrm{E}_y\left[\frac{\partial L\left(y,F\left(\mathbf{x}\right)\right)}{\partial F\left(\mathbf{x}\right)}\right]_{F\left(\mathbf{x}\right)=F_{m-1}\left(\mathbf{x}\right)}$$

即将 $F\left(\mathbf{x}\right)$ 当做损失函数的变量，对其求导，取在 $F_{m-1}\left(\mathbf{x}\right)$ 处的导数值。这个梯度值是对所有训练样本梯度的数学期望。然后执行最速下降法中的直线搜索，确定弱学习器的系数，

即求解如下优化问题

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \rho h\left(\mathbf{x}_i; \mathbf{a}_m\right)\right)$$

接下来更新强学习器，将当前训练得到的弱学习器加入其中

$$F_m\left(\mathbf{x}\right) = F_{m-1}\left(\mathbf{x}\right) + \rho_m h\left(\mathbf{x}; \mathbf{a}_m\right)$$

根据第 2.2.1 节已经证明的结果，沿着负梯度方向迭代，函数值下降，因此如果每个弱学习器拟合在当前点处的负梯度值，将其加入强学习器中之后损失函数值下降。强学习器在每次迭代（加上当前弱学习器的输出值）后的输出值构成的序列，与梯度下降法中优化变量的迭代序列一致。由此得到通用的梯度提升算法框架。

初始化强学习器：$F_0\left(\mathbf{x}\right) = \arg\min_{\rho} \sum_{i=1}^{N} L\left(y_i, \rho\right)$

循环，对 $m = 1, ..., M$

计算梯度值：$y_i^{'} = -\left[\frac{\partial L\left(y_i, F\left(\mathbf{x}_i\right)\right)}{\partial F\left(\mathbf{x}_i\right)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1, ..., N$

训练弱学习器：$\mathbf{a}_m = \arg\min_{\mathbf{a}, \beta} \sum_{i=1}^{N} \left[y_i^{'} - \beta h\left(\mathbf{x}_i; \mathbf{a}\right)\right]^2$

直线搜索，确定系数：$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \rho h\left(\mathbf{x}_i; \mathbf{a}_m\right)\right)$

更新强学习器：$F_m\left(\mathbf{x}\right) = F_{m-1}\left(\mathbf{x}\right) + \rho_m h\left(\mathbf{x}; \mathbf{a}_m\right)$

结束循环

这是一个通用的算法，接下来将其用于不同的损失函数，得到各种具体的梯度提升算法，解决分类和回归问题。

13.6.2 回归问题

对于回归问题，常用的是平方误差损失函数，定义为

$$L\left(y, F\right) = \frac{\left(y - F\right)^2}{2}$$

其中 y 为训练样本标签值，F 为强学习器对训练样本的预测输出值。损失函数对强学习器预测函数的导数为

$$\frac{\partial L\left(y, F\right)}{\partial F} = F - y$$

由此得到训练样本的标签值（称为伪标签）为

$$y_i^{'} = y_i - F_{m-1}\left(\mathbf{x}_i\right)$$

这种情况下弱学习器拟合的是强学习器在当前的残差。由此得到训练算法的流程如下。

将强学习器初始化为样本标签值的均值：$F_0\left(\mathbf{x}\right) = \overline{y}$

循环对 $m = 1,...,M$

计算训练样本伪标签值：$y_i^{'} = y_i - F_{m-1}\left(\mathbf{x}_i\right), i = 1,...,N$

训练弱学习器和系数：$\left(\mathbf{a}_m, \rho_m\right) = \arg\min_{\mathbf{a},\rho} \sum_{i=1}^{N}\left[y_i^{'} - \rho h\left(\mathbf{x}_i;\mathbf{a}\right)\right]$

更新强学习器：$F_m\left(\mathbf{x}\right) = F_{m-1}\left(\mathbf{x}\right) + \rho_m h\left(\mathbf{x};\mathbf{a}_m\right)$

结束循环

除平方误差损失函数之外，回归问题还可以使用绝对值误差，Huber 损失函数。

13.6.3 分类问题

对于二分类问题，一般使用 logistic 回归的对数似然函数作为损失函数，与第 11.1.2 节定义的相同

$$L\left(y,F\right) = \log\left(1 + \exp\left(-yF\right)\right), y \in \left\{-1,1\right\}$$

强分类器的预测函数为对数似然比

$$F\left(\mathbf{x}\right) = \ln\frac{p\left(y=+1\middle|\mathbf{x}\right)}{p\left(y=-1\middle|\mathbf{x}\right)}$$

样本伪标签值为

$$y_i^{'} = -\left[\frac{\partial L\left(y_i, F\left(\mathbf{x}_i\right)\right)}{\partial F\left(\mathbf{x}_i\right)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} = \frac{y_i}{1+\exp\left(y_i F_{m-1}\left(\mathbf{x}_i\right)\right)}$$

直线搜索变为

$$\rho_m = \arg\min_{\rho}\sum_{i=1}^{N}\log\left(1+\exp\left(-y_i\left(F_{m-1}\left(\mathbf{x}_i\right)+\rho h\left(\mathbf{x}_i;\mathbf{a}_m\right)\right)\right)\right)$$

如果使用决策树作为弱学习器，它定义的是分段常数函数，每个叶子节点输出一个固定值，假设第 m 次迭代时决策树的第 l 个叶子节点值为 h_{lm}，对叶子节点的直线搜索为

$$\rho_{lm} = \arg\min_{\rho}\sum_{\mathbf{x}_i\in R_{lm}}\log\left(1+\exp\left(-y_i\left(F_{m-1}\left(\mathbf{x}_i\right)+\rho h_{lm}\right)\right)\right)$$

其中 R_{lm} 为第 m 次迭代时决策树的第 l 个叶子节点定义的区域，落入该区域的训练样本 $\mathbf{x}_i$ 的预测值都是这个叶子节点的值 h_{lm}。将弱学习器的系数与决策树叶子节点的值融合，上

式也可以写成

$$\gamma_{lm}=\rho_{lm}h_{lm}=\arg\min_{\gamma}\sum_{\mathbf{x}_i\in R_{lm}}\log\left(1+\exp\left(-y_i\left(F_{m-1}\left(\mathbf{x}_i\right)+\gamma\right)\right)\right)$$

该问题无法得到公式解，类似于 LogitBoost，采用牛顿法近似求解，利用 logistic 函数的导数公式，最优解为

$$\gamma_{lm}=\sum_{\mathbf{x}_i\in R_{lm}}y_i^{'}\Big/\sum_{\mathbf{x}_i\in R_{lm}}\left|y_i^{'}\right|\left(1-\left|y_i^{'}\right|\right)$$

由此得到训练算法的流程如下。

初始化强学习器：$F_0\left(\mathbf{x}\right)=\log\dfrac{1+\overline{y}}{1-\overline{y}}$

循环对 $m=1,...M$

计算训练样本伪标签值：$y_i^{'}=\dfrac{y_i}{1+\exp\left(y_iF_{m-1}\left(\mathbf{x}_i\right)\right)}$

用 $\left\{\mathbf{x}_i,y_i^{'}\right\}$ 训练决策树，第 l 个叶子节点定义的区域为 $\left\{R_{lm}\right\}$

直线搜索确定叶子节点值：$\gamma_{lm}=\sum_{\mathbf{x}_i\in R_{lm}}y_i^{'}\Big/\sum_{\mathbf{x}_i\in R_{lm}}\left|y_i^{'}\right|\left(1-\left|y_i^{'}\right|\right)$

更新强学习器：$F_m\left(\mathbf{x}\right)=F_{m-1}\left(\mathbf{x}\right)+\gamma_{lm}1\left(\mathbf{x}\in R_{lm}\right)$

结束循环

预测时直接用强学习器得到输出值。

对于多分类问题，使用 softmax 回归的做法，损失函数为交叉熵，与第 11.4 节的定义相同。如果有 K 个类，则每个弱学习器由 K 棵决策树构成，根据它们预测样本属于每个类的概率。单个样本的损失函数定义为

$$L\left(\left\{y_k,F_k\left(\mathbf{x}\right)\right\}\right)=-\sum_{k=1}^{K}y_k\log p_k\left(\mathbf{x}\right)$$

训练样本的标签值采用 one-hot 编码。弱学习器预测样本属于每个类 k 的概率

$$p_k\left(\mathbf{x}\right)=p\left(y_k=1|\mathbf{x}\right)$$

此概率值的计算公式为

$$p_k\left(\mathbf{x}\right)=\frac{\exp\left(F_k\left(\mathbf{x}\right)\right)}{\sum_{l=1}^{K}\exp\left(F_l\left(\mathbf{x}\right)\right)}$$

对上式取对数，可以得到

$$F_k\left(\mathbf{x}\right)=\log p_k\left(\mathbf{x}\right)-\frac{1}{K}\sum_{l=1}^{K}\log p_l\left(\mathbf{x}\right)$$

这就是强学习器要拟合的目标。将预测函数代入交叉熵的定义并求导，可以得到伪标签

值为

$$y_{ik}^{'}=-\left[\frac{\partial L\left(\left\{y_{ij},F_j\left(\mathbf{x}_i\right)\right\}\right)}{\partial F_k\left(\mathbf{x}_i\right)}\right]_{\left\{F_j(\mathbf{x})=F_{j,m-1}(\mathbf{x})\right\}}=y_{jk}-p_k\left(\mathbf{x}_i\right)$$

每次迭代时训练 K 棵决策树，每棵决策树预测的目标是 $\left\{y_{ik}-p_k\left(\mathbf{x}_i\right)\right\}$。在第 m 次迭代时产生 K 棵决策树，每棵决策树有 N_L 个叶子节点，第 k 棵树的第 l 个叶子节点定义的区域为 $\left\{R_{klm}\right\}$，该叶子节点的直线搜索为

$$\gamma_{klm}=\arg\min_{\gamma}\sum_{\mathbf{x}_i\in R_{klm}}\phi_k\left(y_{ik},F_{k,m-1}\left(\mathbf{x}_i\right)+\gamma\right)$$

其中

$$\phi_k=-y_k\log p_k\left(\mathbf{x}\right)$$

概率值 $p_k\left(\mathbf{x}\right)$ 在前面已经定义。这个问题也没有公式解，采用牛顿法近似求解。如果用对角矩阵近似 Hessian 矩阵，可以得到

$$\gamma_{klm}=\frac{K-1}{K}\frac{\sum_{x_i\in R_{klm}}y_{ik}^{'}}{\sum_{x_i\in R_{klm}}\left|y_{ik}^{'}\right|\left(1-\left|y_{ik}^{'}\right|\right)}$$

由此得到下面的训练算法。

初始化强学习器：$F_{k0}\left(\mathbf{x}\right)=0,k=1,...,K$

循环，对 $m=1,...,M$

用当前强学习器对样本进行预测：$p_k\left(x\right)=\frac{\exp\left(F_k\left(\mathbf{x}\right)\right)}{\sum_{l=1}^{K}\exp\left(F_l\left(\mathbf{x}\right)\right)}$

循环，对 $k=1,...,K$

计算训练样本伪标签值：$y_{ik}^{'}=y_{jk}-p_k\left(\mathbf{x}_i\right)$

用 $\left\{\mathbf{x}_i,y_{ik}^{'}\right\}$ 训练决策树，叶子节点定义的区域为 $\left\{R_{klm}\right\}$

直线搜索确定叶子节点值：$\gamma_{klm}=\frac{K-1}{K}\frac{\sum_{\mathbf{x}_i\in R_{klm}}y_{ik}^{'}}{\sum_{\mathbf{x}_i\in R_{klm}}\left|y_{ik}^{'}\right|\left(1-\left|y_{ik}^{'}\right|\right)}$

更新强学习器：$F_{km}\left(\mathbf{x}\right)=F_{k,m-1}\left(\mathbf{x}\right)+\gamma_{klm}1\left(\mathbf{x}\in R_{klm}\right)$

结束循环

结束循环

预测时根据强学习器的输出值估计样本属于每一类的概率，类似 softmax 回归。

13.6.4 XGBoost

XGBoost[26]是对梯度提升算法的改进，求解损失函数极值时使用了牛顿法，将损失函数泰勒展开到二阶，另外在损失函数中加入了正则化项。训练时的目标函数由两部分构成，第一部分为梯度提升算法损失，第二部分为正则化项。损失函数定义为

$$L(\phi)=\sum_{i=1}^{n} l\left(y_i^{'}, y_i\right)+\sum_{k} \Omega\left(f_k\right)$$

其中 n 为训练样本数，l 是对单个样本的损失，假设它为凸函数，$y_i^{'}$ 为模对训练样本的预测值，y_i 为训练样本的真实标签值。正则化项定义了模型的复杂程度

$$\Omega(f)=\gamma T+\frac{1}{2}\lambda\|\mathbf{w}\|^2$$

其中 γ 和 λ 为人工设置的系数，$\mathbf{w}$ 为决策树所有叶子节点值形成的向量，T 为叶子节点数。正则化项由叶子节点数，叶子节点值向量的模平方两项构成，第一项体现了决策树结构的复杂程度，第二项体现了决策树预测值的复杂程度。定义 q 为输入向量 $\mathbf{x}$ 到决策树叶子节点编号的映射，即根据输入向量确定用决策树的第几个叶子节点值来预测它的输出值

$$i=q(\mathbf{x}): \mathbb{R}^d \rightarrow \mathbb{Z}$$

因此 q 定义了树的结构，$\mathbf{w}$ 定义了树的输出值。决策树的映射函数可以写成

$$f(\mathbf{x})=w_{q(\mathbf{x})}$$

与梯度提升算法相同，用加法模型表示强学习器。假设 $y_{i,t}^{'}$ 为第 i 个样本在第 t 次迭代时的强学习器预测值，训练时依次确定每一个弱学习器函数 f_t，加到强学习器预测函数中，即最小化如下目标函数

$$L_t=\sum_{i=1}^{n} l\left(y_i, y_{i,t-1}^{'}+f_t\left(\mathbf{x}_i\right)\right)+\Omega\left(f_t\right)$$

用贪婪法将 f_t 加入到模型中，以最小化目标函数值。对于一般的损失函数无法求得上面优化问题的公式解。采用牛顿法近似求解，对目标函数在 $y_{i,t-1}^{'}$ 点处作二阶泰勒展开后得到

$$L_t \approx \sum_{i=1}^{n}\left(l\left(y_i, y_{i,t-1}^{'}\right)+g_i f_t\left(\mathbf{x}_i\right)+\frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\right)+\Omega\left(f_t\right)$$

损失函数的一阶导数为

$$g_i = \frac{\partial l\left(y_i, y_{t-1}^{'}\right)}{\partial y_{t-1}^{'}}$$

与梯度提升算法相同，将之前已经训练得到的强学习器对样本的预测值当做变量求导。损失函数的二阶导数为

$$h_i = \frac{\partial^2 l\left(y_i, y_{t-1}^{'}\right)}{\partial^2 y_{t-1}^{'}}$$

去掉与 $f_t\left(\mathbf{x}_i\right)$ 无关的常数项，可以得到化简后的损失函数为

$$L_t = \sum_{i=1}^{n}\left(g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\right) + \Omega\left(f_t\right)$$

如果定义集合

$$I_j = \left\{i \middle| q\left(\mathbf{x}_i\right) = j\right\}$$

即落入第 j 个叶子节点的训练样本集合（样本下标集合）。由于每个训练样本只属于某一个叶子节点，目标函数可以拆分成对所有叶子节点损失函数之和

$$\begin{aligned} L_t &= \sum_{i=1}^{n}\left(g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2} h_i f_t^2\left(\mathbf{x}_i\right)\right) + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}\left(\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right) + \gamma T \end{aligned}$$

首先介绍叶子节点的值如何确定。如果决策树的结构即 $q\left(\mathbf{x}\right)$ 确定，根据牛顿法可以得到第 j 个叶子节点的最优值为

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}$$

这是单个叶子节点的损失函数对 w_j 求导并令导数为 0 后解方程的结果。前面已经假定对单个样本的损失函数是凸函数，因此必定是极小值。单个叶子节点的损失函数对 w_j 的一阶导数为

$$\frac{\partial}{\partial w_j}\left(\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right) w_j^2\right) = \sum_{i\in I_j} g_i + \left(\sum_{i\in I_j} h_i + \lambda\right) w_j$$

令其为 0，即可得到上面的结果。

接下来说明如何确定决策树的结构，即寻找最佳分裂。将 w_j 的最优解代入损失函数，得到只含有 q 的损失函数

$$L_t(q)=\sum_{j=1}^{T}\left(\left(\sum_{i\in I_j}g_i\right)\left(-\frac{\sum_{i\in I_j}g_i}{\sum_{i\in I_j}h_i+\lambda}\right)+\frac{1}{2}\left(\sum_{i\in I_j}h_i+\lambda\right)\left(-\frac{\sum_{i\in I_j}g_i}{\sum_{i\in I_j}h_i+\lambda}\right)^2\right)+\gamma T$$

$$=\sum_{j=1}^{T}\left(-\frac{\left(\sum_{i\in I_j}g_i\right)^2}{\sum_{i\in I_j}h_i+\lambda}+\frac{1}{2}\frac{\left(\sum_{i\in I_j}g_i\right)^2}{\sum_{i\in I_j}h_i+\lambda}\right)+\gamma T$$

$$=-\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j}g_i\right)^2}{\sum_{i\in I_j}h_i+\lambda}+\gamma T$$

此函数可以看做是对决策树结构优劣的一个度量，要求其极小值，类似于决策树寻找分裂时的不纯度指标。假设节点在分裂之前的训练样本集为I，分裂之后左子节点的训练样本集为I_L，右子节点的训练样本集为I_R。决策树每次分裂之后叶子节点数增加 1，因此上面目标函数的第二项由γT变为$\gamma(T+1)$，二者的差值为γ。将上面的目标函数取反，求$L_t(q)$的极小值等价于求$-L_t(q)$的极大值。寻找最佳分裂的目标是使得损失函数的下降值最大化

$$L_{split}=\frac{1}{2}\left(\frac{\left(\sum_{i\in I_L}g_i\right)^2}{\sum_{i\in I_L}h_i+\lambda}+\frac{\left(\sum_{i\in I_R}g_i\right)^2}{\sum_{i\in I_R}h_i+\lambda}-\frac{\left(\sum_{i\in I}g_i\right)^2}{\sum_{i\in I}h_i+\lambda}\right)-\gamma$$

除了使用不同的分裂指标，其他过程与标准的决策树训练算法相同，在第 5.3.2 节已经介绍。

XGBoost 实现时还使用了权重收缩与列采样技术，以抵抗过拟合。权重收缩的做法是在每训练出一棵决策树之后将其权重进行缩放，乘上系数η。权重收缩降低了之前的决策树的影响，为将来要生成的决策树留下了更多的空间。列采样的做法与随机森林相同，每次寻找最佳分裂时只考虑一部分随机抽取的特征分量。

参 考 文 献

[1] Freund, Y. Boosting a weak learning algorithm by majority. Information and Computation, 1995.

[2] Yoav Freund, Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. computational learning theory. 1995.

[3] Freund, Y. An adaptive version of the boost by majority algorithm. In Proceedings of the Twelfth Annual Conference on Computational Learning Theory, 1999.

[4] R.Schapire. The boosting approach to machine learning: An overview. In MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, 2001.

[5] Freund Y, Schapire RE. A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780. 1999.

[6] Hastie, T. and Tibshirani, R.. Generalized Additive Models. Chapman and Hall, London. 1990.

[7] Jerome Friedman, Trevor Hastie and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. Annals of Statistics 28(2), 337–407. 2000.

[8] Buja, A., Hastie, T. and Tibshirani, R. Linear smoothers and additive models (with discussion). Ann. Statist. 17 453–555. 1989.

[9] P.Viola and M.Jones. Rapid object detection using a boosted cascade of simple features. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, 2001.

[10] Lubomir Bourdev, Jonathan Brandt. Robust Object Detection Via Soft Cascade. CVPR 2005.

[11] Rainer Lienhart, Jochen Maydt. An extended set of Haar-like features for rapid object detection. international conference on image processing. 2002.

[12] Bo Wu, Haizhou Ai, Chang Huang, Shihong Lao. Fast rotation invariant multi-view face detection based on real Adaboost. IEEE international conference on automatic face and gesture recognition. 2004.

[13] Bin Yang, Junjie Yan, Zhen Lei, Stan Z Li. Aggregate channel features for multi-view face detection. International Journal of Central Banking. 2014.

[14] Timo Ahonen, Abdenour Hadid, Matti Pietikainen. Face Description with Local Binary Patterns: Application to Face Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2006.

[15] R. Benenson, M. Mathhhias, R. Tomofte, L. Van Gool. Pedestrian detection at 100 frames per second. CVPR, 2012.

[16] M.Jones, P.Viola. Fast Multi-View Face Detection. Mitsubishi Electric Research Laboratories, Technical Report: MERL-2003-96, July 2003.

[17] Y.Ma, X.Q.Ding. Real-time rotation invariant face detection based on cost-sensitive AdaBoost. In: Proceedings of the IEEE International Conference on Image Processing. Barcelona, Spain: IEEE Computer Society, 921-924. 2003.

[18] Y.Ma, X.Q.Ding. Robust multi-view face detection and pose estimation based on cost-sensitive AdaBoost. In: Proceedings of the 6-th Asian Conference on Computer Vision. Jeju, Korea: Springer, 2004.

[19] S.Z.Li, L.Zhu, Z.Q.Zhang, A.Blake, H.J.Zhang, H.Y.Shum. Statistical learning of multi-view face detection. In: Proceedings of the 7-th European Conference on Computer Vision. Copenhagen, Denmark: Springer, 67-81. 2002.

[20] S.Z.Li, Z.Q.Zhang, H.Y.Shum, H.J.Zhang. FloatBoost learning for classification. In: Proceedings of the 16-th Annual Conference on Neural Information Processing Systems. Vancouver, Canada: MIT Press, 993-1000. 2002.

[21] S.Z.Li, L.Zhu, Z.Q.Zhang, H.J.Zhang. Learning to detect multi-view faces in real-time. In: Proceedings of the 2-nd International Conference on Development and Learning. Cambridge, MA, USA: IEEE Computer Society, 172-177. 2002.

[22] S.Z.Li, Z.Q.Zhang. FloatBoost Learning and Statistical Face Detection. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004.

[23] Yudong Cai, Kaiyan Feng, Wencong Lu, Kuochen Chou. Using LogitBoost classifier to predict protein structural classes. Journal of Theoretical Biology. 2006.

[24] Robert E Schapire, Yoram Singer. Improved boosting algorithms using confidence-rated predictions. computational learning theory. 1998.

[25] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. Annals of Statistics. 2001.

[26] Tianqi Chen, Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. knowledge discovery and

data mining. 2016.