

提升方法

NAMI LAB

陆思远

Content

- Boosting
- AdaBoost
- 提升树
- XGBoost

Boosting

➤ 基本思想：对于一个复杂任务，将多个专家的判断进行适当的综合所得出的判断，要比其中任何一个专家单独的判断好

➤ 概率近似正确（PAC）

$$P(R(f) < \varepsilon) \geq 1 - \delta$$

泛化误差

置信度



近似正确

Boosting

- 对于一个概念 c ，如果存在一个算法，对于任何 $0 < \varepsilon, \delta < 1$ ，能够给出一个分类器 $h(x)$ ，使得 h 的错误率小于 ε 的概率：

$$P(\text{err}(h) < \varepsilon) \geq 1 - \delta$$

且算法在 n, ε, δ 的多项式时间内完成，则称 c 是PAC可学习的（或强可学习的）

- 弱可学习：一个概念（类），如果存在一个多项式的学习算法可以学习它，学习正确率仅比随机猜测好 $\varepsilon \geq \frac{1}{2} - \gamma$ （ γ 可以看成是一个很小的正数）
- 一个概念是弱可学习的当且仅当其是强可学习的(强可学习和弱可学习等价)

Boosting

- 从弱学习算法出发，反复学习，得到一系列弱分类器（基本分类器），组合这些分类器，构成一个强分类器
- 改变训练数据的概率分布（训练数据的权值分布）
- 1、每一轮如何改变训练数据的权值或概率分布
 - 提高前一轮弱分类器错误分类样本的权值，降低正确分类样本的权值
- 2、将弱分类器组合成一个强分类器
 - 加权多数表决

AdaBoost

$$T = \{(x_1, y_1), \dots (x_N, y_N)\} \quad x_i \in \mathcal{X} \in \mathbf{R}^n, y_i \in \{-1, +1\}$$

- ① 初始化训练数据的权值分布

$$D_1 = (w_{11}, \dots, w_{1N}), w_{1i} = \frac{1}{N}$$

- ② 对于 $m = 1, 2, \dots, M$

使用具有权值分布的 D_m 的训练数据集学习，得到基本分类器 $G_m(x)$

计算 $G_m(x)$ 在训练集上的分类误差率

$$e_m = \sum_{i=1}^N P(G_m(x_i) \neq y_i) = \sum_{i=1}^N w_{mi} I(G_m(x_i) \neq y_i) = \sum_{G_m(x_i) \neq y_i} w_{mi}$$

AdaBoost

- ③ 计算 $G_m(x)$ 的系数（在最终分类器的重要性）

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m} \implies e_m \leq \frac{1}{2} \implies \alpha_m \geq 0$$

- ④ 更新训练集的权值分布

$$D_{m+1} = (w_{m+1,1}, \dots, w_{m+1,N})$$
$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)) \quad \left\{ \begin{array}{l} w_{m+1,i} = \frac{w_{mi}}{Z_m} e^{-\alpha_m}, G_m(x_i) = y_i \\ w_{m+1,i} = \frac{w_{mi}}{Z_m} e^{\alpha_m}, G_m(x_i) \neq y_i \end{array} \right.$$


$$Z_m = \sum_{i=1}^N w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$

- ⑤ 构造基本分类器的线性组合，得到最终分类器

$$f(x) = \sum_{m=1}^m \alpha_m G_m(x)$$


AdaBoost训练误差分析

$$\frac{1}{N} \sum_{i=1}^N I(G_m(x_i) \neq y_i) \leq \frac{1}{N} \sum_i \exp(-y_i f(x_i)) = \prod_m Z_m$$



$$\begin{aligned} G_m(x_i) &\neq y_i \\ \Rightarrow y_i f(x_i) &< 0 \\ \Rightarrow \exp(-y_i f(x_i)) &\geq 1 \end{aligned}$$

$$\begin{aligned} w_{m+1,i} Z_m &= w_{mi} \exp(-\alpha_m y_i G_m(x_i)) \\ \Rightarrow w_{mi} &= \frac{w_{m+1,i} Z_m}{\exp(-\alpha_m y_i G_m(x_i))} \end{aligned}$$



$$\begin{aligned} \frac{1}{N} \sum_i \exp(-y_i f(x_i)) &= \frac{1}{N} \sum_i \exp(-\sum_{m=1}^M \alpha_m y_i G_m(x_i)) \\ &= \sum_i w_{1i} \prod_{m=1}^M \exp(-\alpha_m y_i G_m(x_i)) \\ &= Z_1 \sum_i w_{2i} \prod_{m=2}^M \exp(-\alpha_m y_i G_m(x_i)) \\ &= \dots \\ &= Z_1 Z_2 \dots Z_{M-1} \sum_i w_{Mi} \exp(-\alpha_M y_i G_M(x_i)) \\ &= \prod_{m=1}^M Z_m \end{aligned}$$


$w_{1i} = \frac{1}{N}$

AdaBoost训练误差分析(二分类)

$$\prod_{m=1}^M Z_m = \prod_{m=1}^M [2\sqrt{e_m(1-e_m)}] = \prod_{m=1}^M \sqrt{(1-4\gamma_m^2)} \leq \exp(-2 \sum_{m=1}^M \gamma_m^2)$$

$(\gamma_m = \frac{1}{2} - e_m)$

$$\begin{aligned} Z_m &= \sum_{i=1}^N w_{mi} \exp(-\alpha_m y_i G_m(x_i)) = \sum_{y_i=G_m(x_i)} w_{mi} e^{-\alpha_m} + \sum_{y_i \neq G_m(x_i)} w_{mi} e^{\alpha_m} \\ &= (1-e_m)e^{-\alpha_m} + e_m e^{\alpha_m} = 2\sqrt{e_m(1-e_m)} = \sqrt{1-4\gamma_m^2} \end{aligned}$$


$$e^{\alpha_m} = e^{\frac{1}{2} \log \frac{1-e_m}{e_m}} = \sqrt{\frac{1-e_m}{e_m}}$$

AdaBoost训练误差分析(二分类)

泰勒展开式($x = 0$):

$$e^x = 1 + x + x^2$$

$$\sqrt{1-x} = 1 - \frac{x}{2} - \frac{x^2}{4}$$

$$\sqrt{1-x} \leq e^{-\frac{x}{2}}$$

$$\prod_{m=1}^M \sqrt{1-4\gamma_m^2} \leq \exp(-2 \sum_{m=1}^M \gamma_m^2)$$

$$\sqrt{1-4\gamma_m^2} \leq \exp(-2\gamma_m^2)$$

$$1-4\gamma_m^2 \leq \exp(-4\gamma_m^2)$$

$$4\gamma_m^2 \in [0,1]$$

$$1-x \leq \exp(-x) \quad x \in [0,1]$$

求导利用单调性可以判断

AdaBoost训练误差分析(二分类)

➤推论:

若存在 $\gamma > 0$, 且对于所以 m 有 $\gamma_m \geq \gamma$

$$\frac{1}{N} \sum_{i=1}^N I(G_m(x_i) \neq y_i) \leq \exp(-2M\gamma^2)$$

➤训练误差随着 M 增长以指数速率下降

前向分步算法

➤ 加法模型

$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$$

$b(x; \gamma_m)$ 基函数, γ_m 基函数的参数, β_m 基函数的系数

$$\min_{\beta_m, \gamma_m} \sum_{i=1}^N L(y_i, \sum_{m=1}^M \beta_m b(x_i; \gamma_m))$$

➤ 前向分步算法(forward stagewise algorithm)

每一步只学习一个基函数及其参数, 逐步逼近优化目标函数

$$\min_{\beta, \gamma} \sum_{i=1}^N L(y_i, \beta b(x_i; \gamma))$$

前向分步算法

初始化 $f_0(x) = 0$

极小化损失函数

$$(\beta_m, \gamma_m) = \underset{\beta, \gamma}{\operatorname{argmin}} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$$

更新

$$f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$$

加法模型

$$f(x) = f_M(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$$

前向分步算法与AdaBoost

➤ AdaBoost 是模型为加法模型，损失函数为指数函数，学习算法为前向分步算法

$$f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$$

$$(\alpha_m, G_m(x)) = \underset{\alpha, G}{\operatorname{argmin}} \sum_{i=1}^N \exp[-y_i(f_{m-1}(x_i) + \alpha G(x_i))]$$

$$(\alpha_m, G_m(x)) = \underset{\alpha, G}{\operatorname{argmin}} \sum_{i=1}^N \bar{w}_{mi} \exp[-y_i \alpha G(x_i)]$$

$\bar{w}_{mi} = \exp[-y_i f_{m-1}(x_i)]$ 与 α, G 无关

$$G_m^*(x) = \underset{G}{\operatorname{argmin}} \sum_{i=1}^N \bar{w}_{mi} I(y_i \neq G(x_i))$$

前向分步算法与AdaBoost

$$\begin{aligned} & \sum_{i=1}^N \bar{w}_{mi} \exp[-y_i \alpha G(x_i)] \\ &= \sum_{y_i=G_m(x_i)} \bar{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G_m(x_i)} \bar{w}_{mi} e^{\alpha} \quad \xrightarrow{\text{求导}} \quad \alpha_m^* = \frac{1}{2} \log \frac{1 - e_m}{e_m} \\ &= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^N \bar{w}_{mi} I(y_i \neq G(x_i)) \quad e_m \text{ 分类误差率} \\ &+ e^{-\alpha} \sum_{i=1}^N \bar{w}_{mi} \quad e_m = \frac{\sum_{i=1}^N \bar{w}_{mi} I(y_i \neq G(x_i))}{\sum_{i=1}^N \bar{w}_{mi}} = \sum_{i=1}^N w_{mi} I(G_m(x_i) \neq y_i) \end{aligned}$$

$$\left. \begin{aligned} f_m(x) &= f_{m-1}(x) + \alpha_m G_m(x) \\ \bar{w}_{mi} &= \exp[-y_i f_{m-1}(x_i)] \end{aligned} \right\} \bar{w}_{m+1,i} = \bar{w}_{m,i} \exp(-y_i \alpha_m G_m(x))$$

提升树

➤ 以决策树（二叉树）为基函数的提升方法

➤ 决策树桩：由根节点直接连接两个叶节点的简单决策树

$$f_M(x) = \sum_{m=1}^M T(x; \Theta_m)$$

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$$

➤ 回归树：将输入空间 χ 划分为 J 个互不相交的区域 R_1, \dots, R_J ，并在每个区域上确定输出的常量 c_j

$$T(x; \Theta_m) = \sum_{j=1}^J c_j I(x \in R_j)$$

J 是回归树的复杂度即叶节点的个数

提升树

➤平方误差损失函数：

$$\begin{aligned} &L(y, f_{m-1}(x_i) + T(x_i; \Theta_m)) \\ &= [y - f_{m-1}(x_i) - T(x_i; \Theta_m)]^2 \\ &= [r - T(x_i; \Theta_m)]^2 \end{aligned}$$

➤当前模型拟合数据的残差： $r = y - f_{m-1}(x_i)$

每一步只需拟合当前所求得的模型的残差，就可以学习到下一步的模型，从而实现回归树的学习

梯度提升(Gradient Boosting)

➤ 利用损失函数的负梯度在当前模型的值作为回归问题提升树算法中的残差的近似值

$$r \approx - \left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)} \right]_{f(x)=f_{m-1}(x)}$$

初始化: $f_0(x) = \operatorname{argmin}_c \sum_{i=1}^N L(y_i, c)$

对 r_{mi} 拟合一个回归树, 得到第 m 棵树的叶节点区域 R_{mj}

$$\left. \begin{aligned} c_{mj} &= \operatorname{argmin}_c \sum_{x_i \in R_{mj}} L(y_i, f_{m-1}(x_i) + c) \\ f_m(x) &= f_{m-1}(x) + \sum_{j=1}^J c_{mj} I(x \in R_{mj}) \end{aligned} \right\} f_M(x) = \sum_{m=1}^M \sum_{j=1}^J c_{mj} I(x \in R_{mj})$$

GBDT(Gradient Boosting Decision Tree)

- GBDT是以决策树（CART二叉树，弱分类器）为基学习器的迭代算法
- GBDT里的决策树通常都是回归树(而非分类树，输出是连续值，可以用负梯度进行拟合)
- 优点：适用面广
- 缺点：弱分类器的串行依赖，导致难以并行训练数据

XGBoost

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

Training Loss measures how well model fit on training data

Regularization, measures complexity of model

Objective: $\sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), f_k \in \mathcal{F}$



$$\Omega(f_t) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2$$

Number of leaves

L2 norm of leaf scores

XGBoost

- **Goal** $Obj^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant$
 - Seems still complicated except for the case of square loss
- **Take Taylor expansion of the objective**
 - Recall $f(x + \Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$
 - Define $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$, $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$

$$Obj^{(t)} \simeq \sum_{i=1}^n \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) + constant$$



XGBoost

$$f_t(x) = w_{q(x)}, \quad w \in \mathbf{R}^T, q : \mathbf{R}^d \rightarrow \{1, 2, \dots, T\}$$

- Define the instance set in leaf j as $I_j = \{i | q(x_i) = j\}$
- Regroup the objective by each leaf


$$\begin{aligned} Obj^{(t)} &\simeq \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \\ &= \sum_{i=1}^n \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^T w_j^2 \\ &= \sum_{j=1}^T \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \end{aligned}$$

XGBoost

- Let us define $G_j = \sum_{i \in I_j} g_i$ $H_j = \sum_{i \in I_j} h_i$

$$\begin{aligned} Obj^{(t)} &= \sum_{j=1}^T \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \\ &= \sum_{j=1}^T \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

$$w_j^* = -\frac{G_j}{H_j + \lambda} \quad Obj = -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$$

 This measures how good a tree structure is!

XGBoost

Greedy Learning of the Tree

- In practice, we grow the tree greedily
 - Start from tree with depth 0
 - For each leaf node of the tree, try to add a split. The change of objective after adding the split is

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

the score of left child the score of right child the score of if we do not split The complexity cost by introducing additional leaf



The End

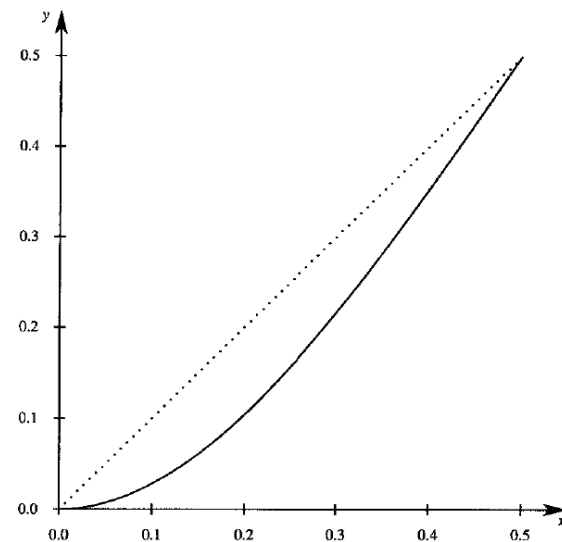
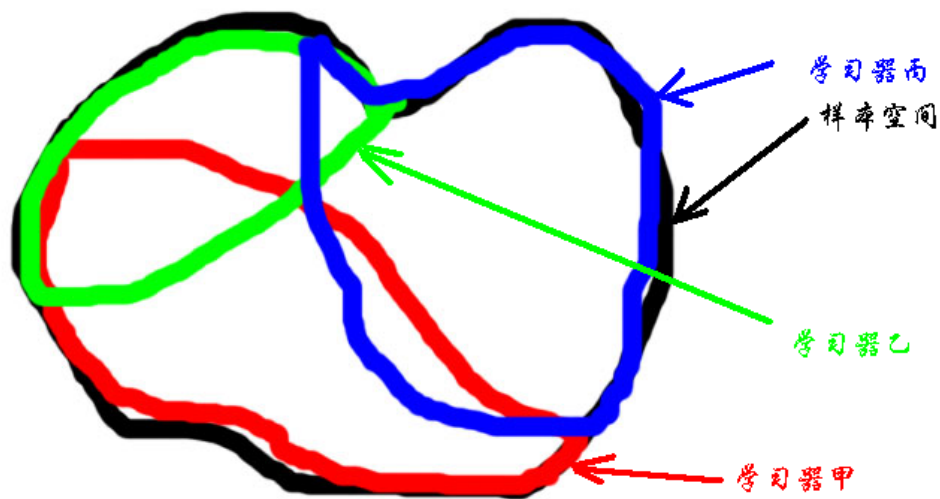
PAC

根据统计机器学习理论，随机选择一个样本集合，必然对应一个分类器；构造弱分类器的过程就是构造样本集合的方法，包含三个步骤； D 是原始样本集合

- ① 从 D 中选取子集 D_1 ，根据 D_1 训练第一个分类器 C_1
- ② 抛硬币策略决定样本集 D_2
 - 正面：选取 D 中剩余的样本用 C_1 分类，一个被错分的样本加入 D_2
 - 反面：选取一个被 C_1 正确分类的样本
 - D_2 中一定在概率上有一半的样本可以被 C_1 正确分类，而另一半被 C_1 错误分类；利用 D_2 训练一个新的分类器
- ③ 构造样本集 D_3 。在 D 剩余的样本中选取样本，用 C_1 和 C_2 进行分类，若分类结果不同，则将样本加入 D_3 ，使用 D_3 训练分类器 C_3

PAC

- h 根据 h_1, h_2, h_3 采用“多数票决”来决定最后输出
- 对于任何 $0 < \varepsilon < 1/2$ ，可以用上述方式构造一个错误率不大于 ε 的分类器
假设 h_1, h_2, h_3 在任意分布上的错误率小于等于 α ，则 $h = \text{sign}(h_1 + h_2 + h_3)$ 在任意分布上的错误率小于等于 $g(\alpha) = 3\alpha^2 - 2\alpha^3$



PAC

$$p_i(v) = \Pr[h_i(v) \neq c(v)]$$

$$q(v) = \Pr[h_1(v) \neq h_2(v)]$$

$$w = \Pr[h_2(v) \neq h_1(v) = c(v)]$$

$$x = \Pr[h_1(v) = h_2(v) = c(v)]$$

$$y = \Pr[h_1(v) \neq h_2(v) = c(v)]$$

$$z = \Pr[h_1(v) \neq h_2(v) \neq c(v)]$$

$$w + x = 1 - a_1$$

$$y + z = a_1$$

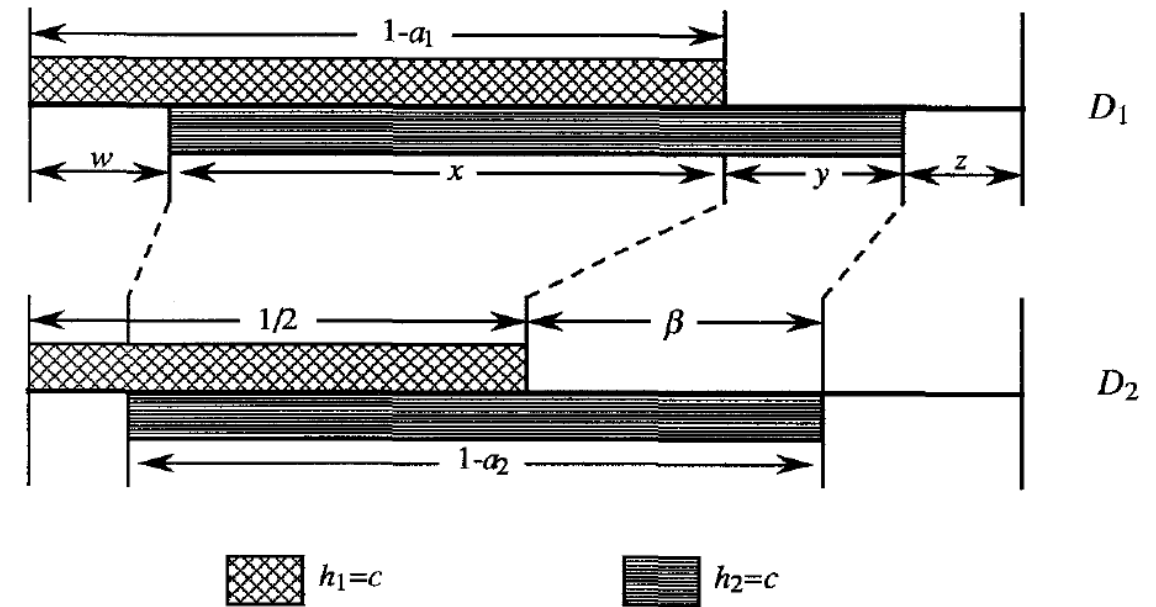


Figure 3. The distributions D_1 and D_2 .

PAC

$$D_1(v) = D(v)$$

$$D_2(v) = \frac{D(v)}{2} \left(\frac{p_1(v)}{a_1} + \frac{1-p_1(v)}{1-a_1} \right)$$

$$D_3(v) = \frac{D(v)q(v)}{w+y}$$

$$1 - a_2 = \sum_{v \in X_n} D_2(v)(1 - p_2(v)) = \frac{y}{2a_1} + \frac{z}{2(1 - a_1)}$$

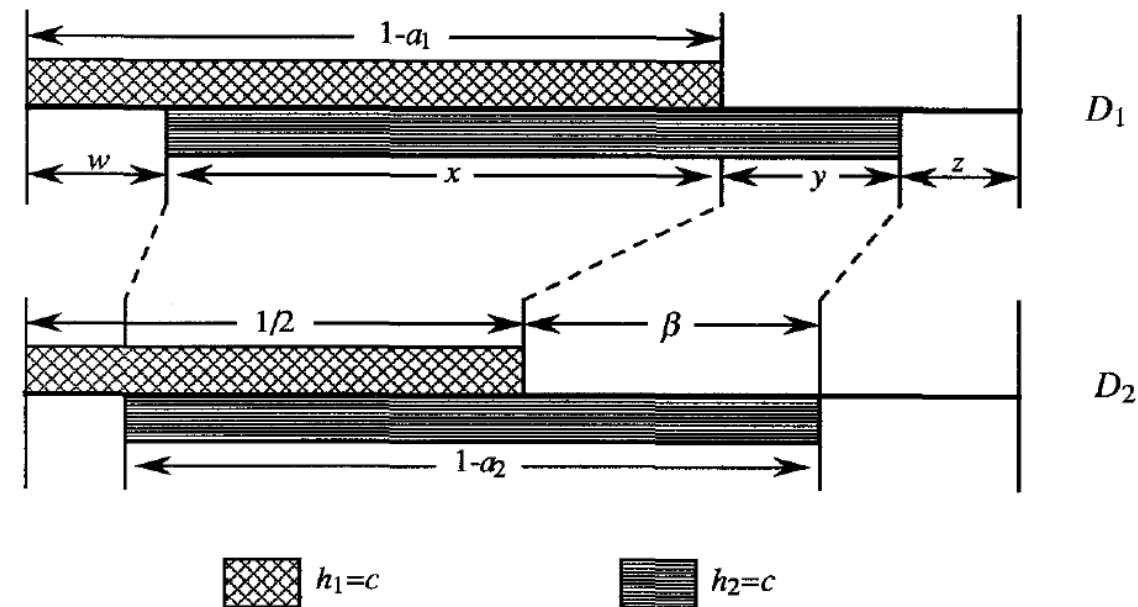


Figure 3. The distributions D_1 and D_2 .

PAC

$$\begin{aligned} & Pr[h(v) \neq c(v)] \\ &= Pr[(h_1(v) = h_2(v) \neq c(v)) \vee (h_1(v) \neq h_2(v) \wedge h_3(v) \neq c(v))] \\ &= z + a_3(w + y) \\ &\leq z + \alpha(w + y) \\ &= \alpha(2a_1 - 1)(1 - a_1) + a_1 + \frac{y(\alpha - a_1)}{a_1} \\ &\leq \alpha(2a_1 - 1)(1 - a_1) + a_1 + \frac{\alpha(\alpha - a_1)}{a_1} \\ &\leq 3\alpha^2 - 2\alpha^3 = \varepsilon \end{aligned}$$