Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

Boosting Methods

博士研究生　　刘晓双

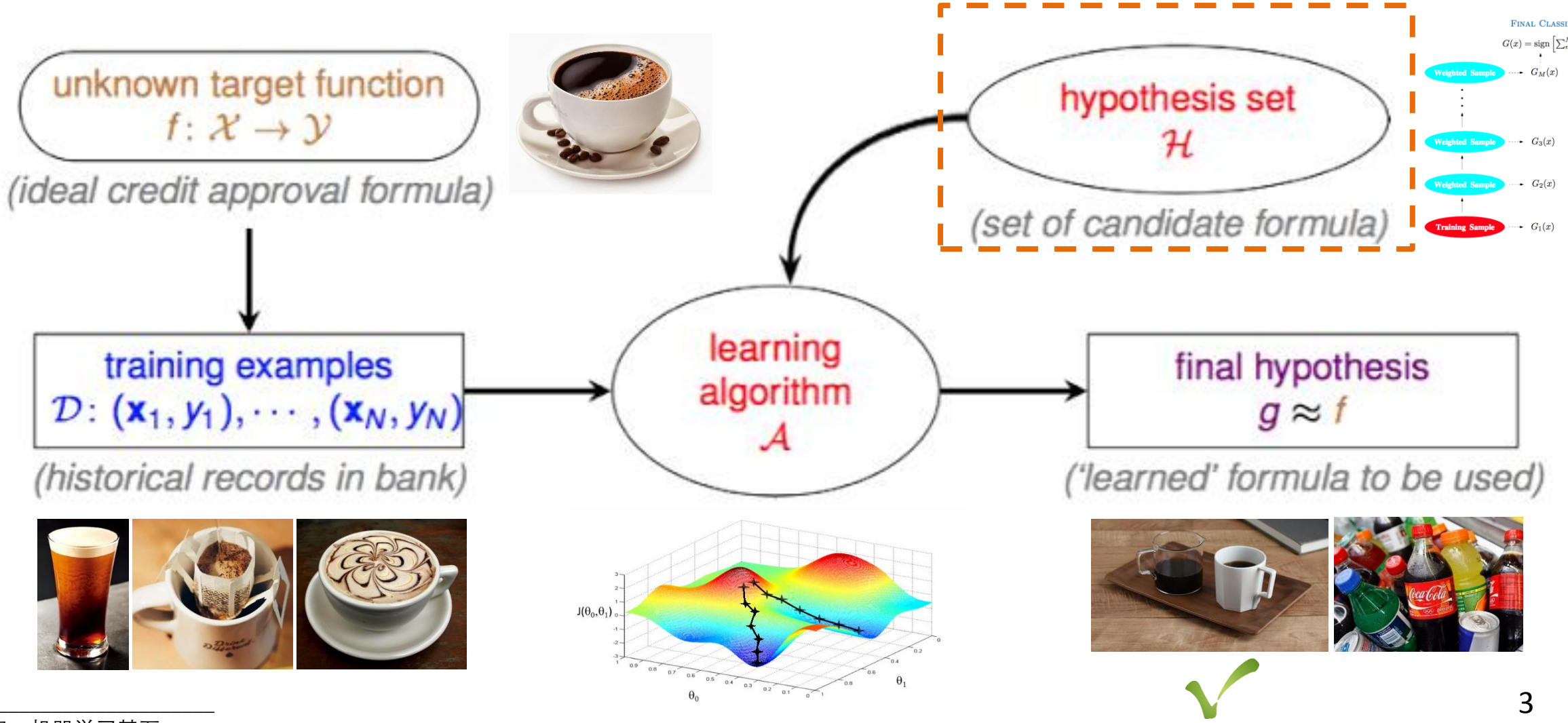
2017年09月10日

内容提要

- 背景简介
- 基本概念
- 算法原理
- 优劣分析
- 应用总结

背景简介

- 机器学习基础架构[1]



[1] 林轩田，机器学习基石

基本概念

- 集成学习（Ensemble learning）
  - In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

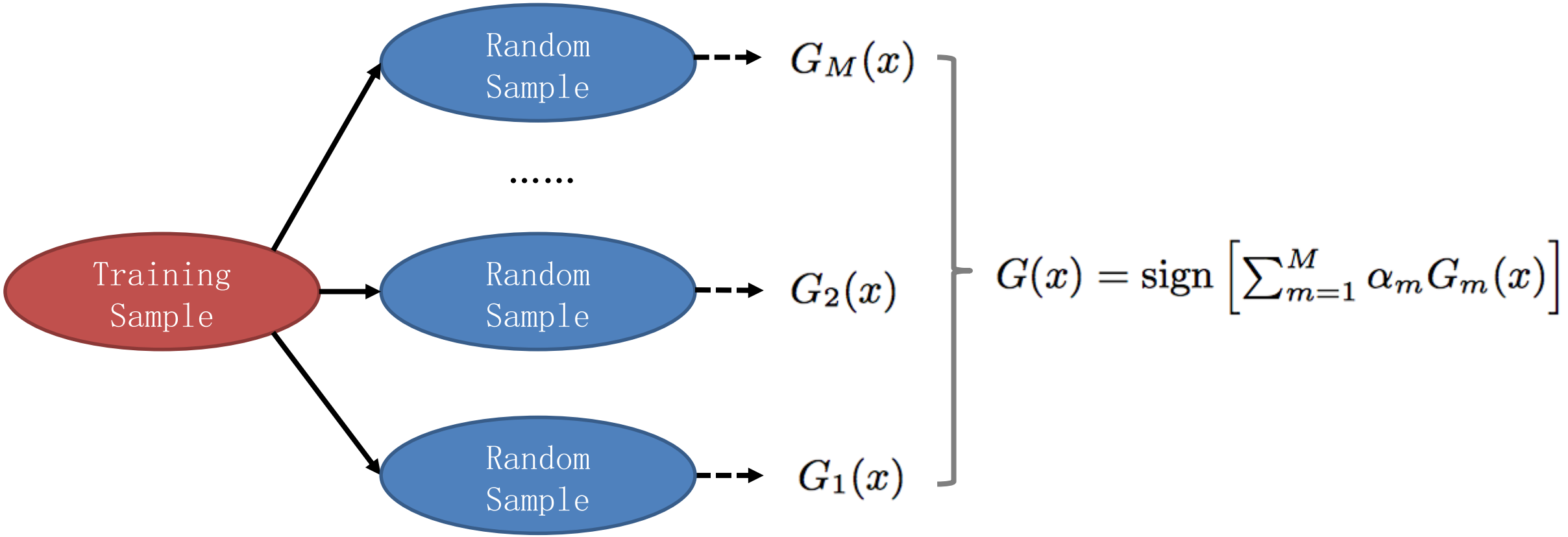
  - Bayes optimal classifier
    - all the hypotheses
  - Bayesian parameter averaging
  - Bayesian model combination
  - Bucket of models
  - Stacking

  - Bootstrap aggregating (bagging)
    - trains each model using a randomly drawn subset of the training set
  - Boosting

Wikipedia，Ensemble learning

基本概念

- Bagging



基本概念

- 集成学习（Ensemble learning）
  - In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

  - Bayes optimal classifier
    - all the hypotheses
  - Bayesian parameter averaging
  - Bayesian model combination
  - Bucket of models
  - Stacking

  - Bootstrap aggregating (bagging)
    - trains each model using a randomly drawn subset of the training set
  - Boosting

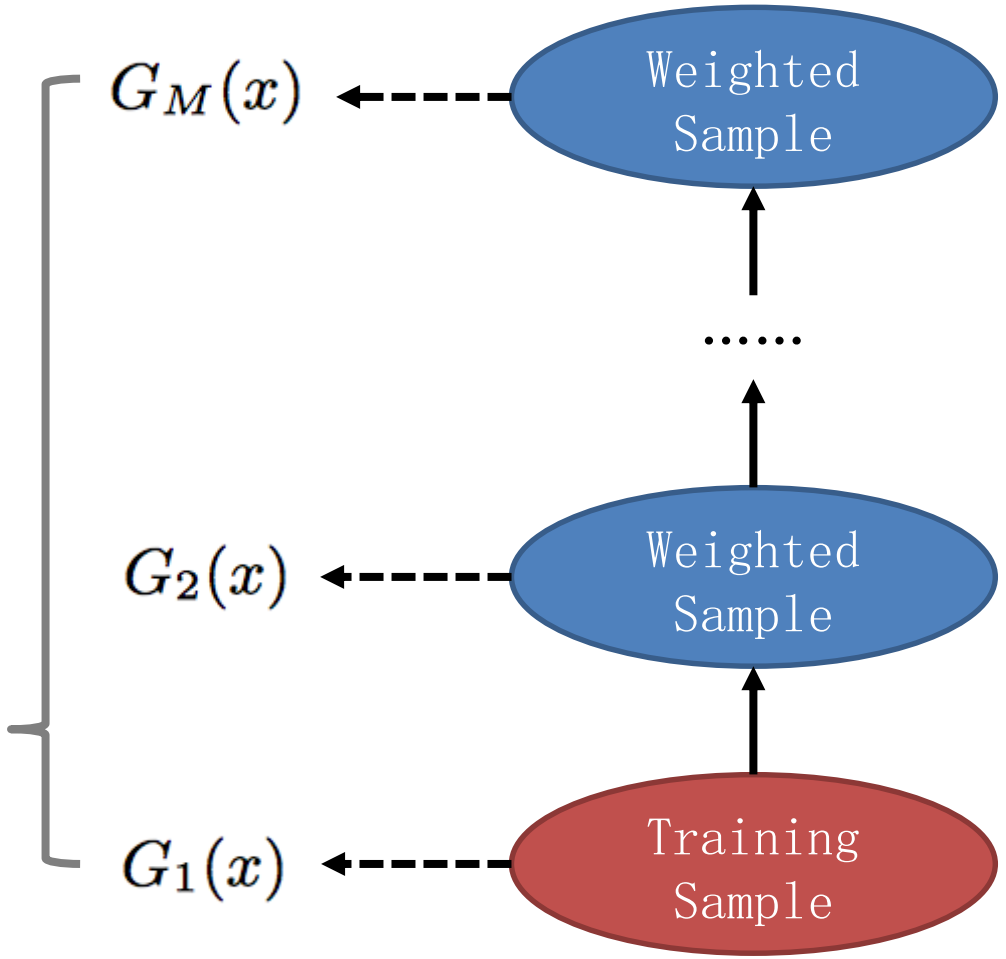
Wikipedia，Ensemble learning

算法原理

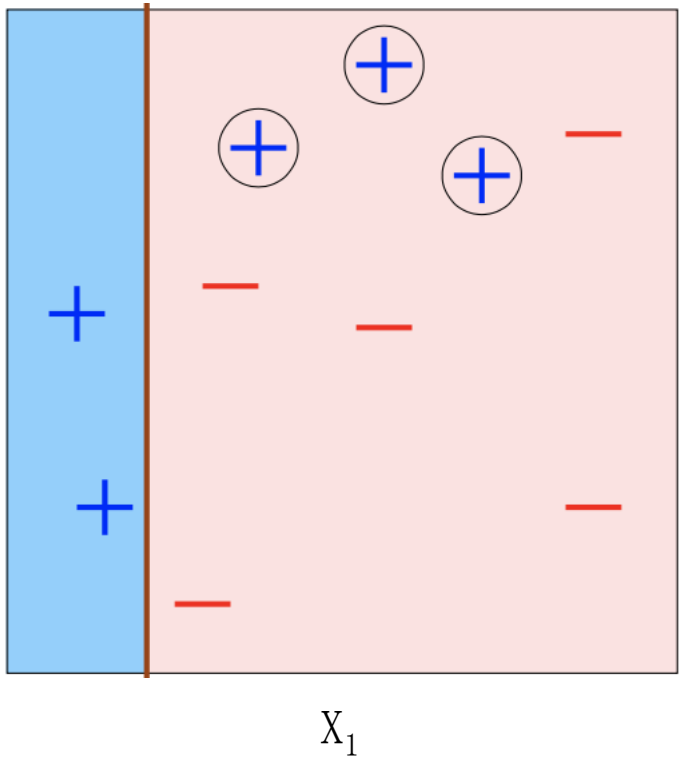
算法原理

- Boosting
  - Incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models misclassified.

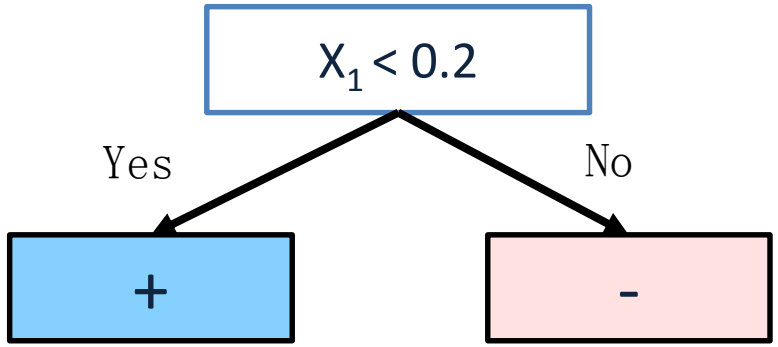
$$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$



Boosting - Toy Example

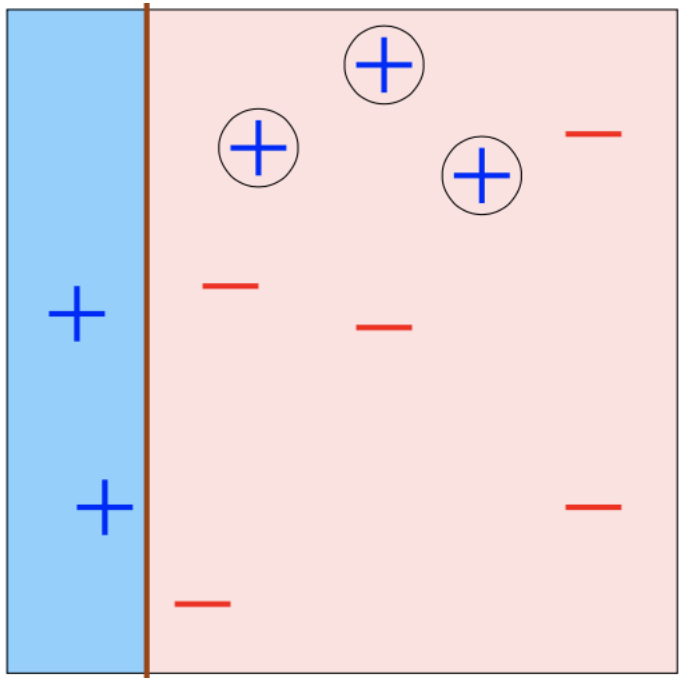


- Decision Stump



Boosting - Toy Example

h_1



$\epsilon_1 = 0.30$

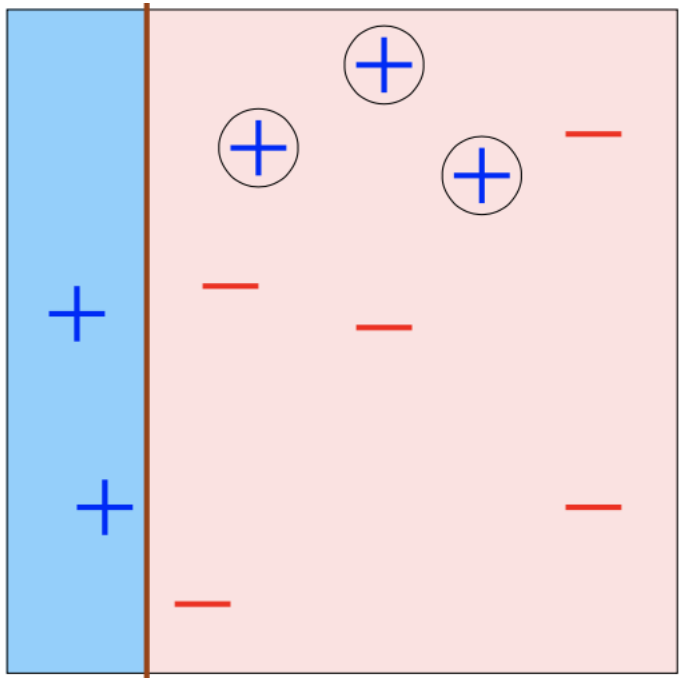
$\alpha_1 = 0.42$

$$\epsilon(h_t) = \frac{1}{m}\sum_{i=1}^{m}\delta(h_t(x_i) \neq y_i)$$

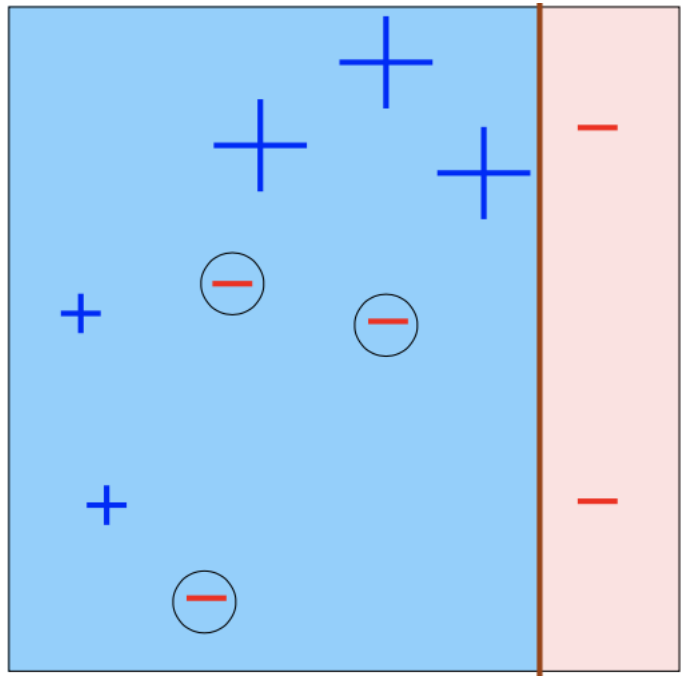
$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}\right)$$

Boosting - Toy Example

h_1

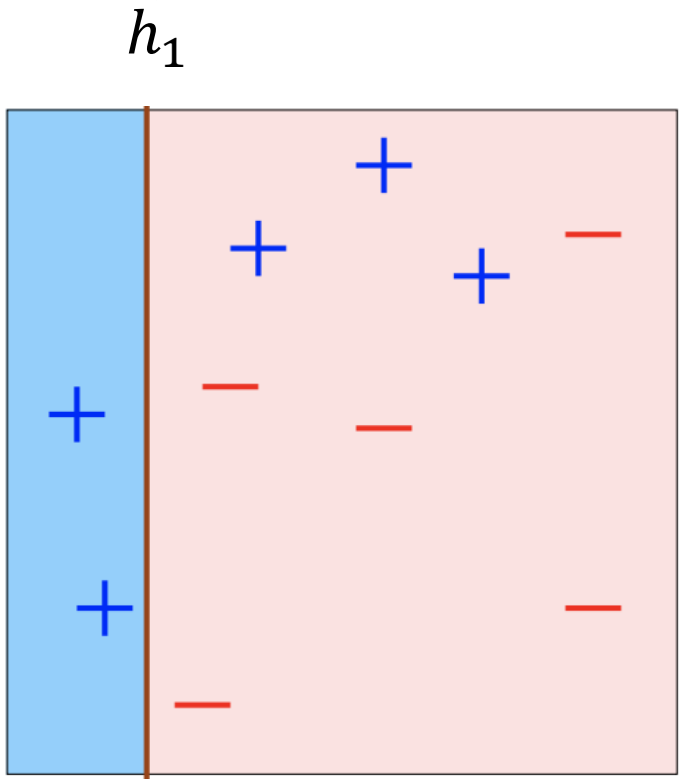


h_2



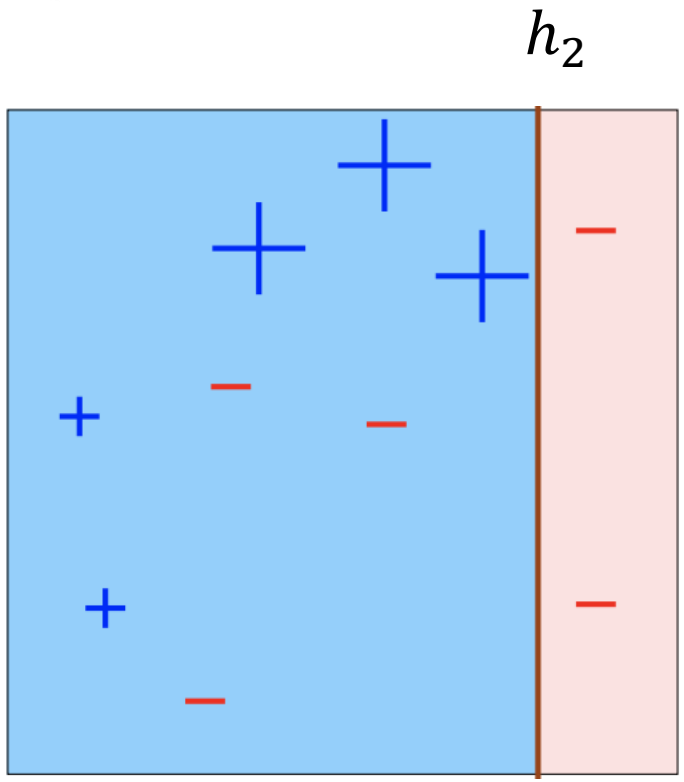
$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(i)\delta(h(\mathbf{x}_i) \neq y_i)$$

Boosting - Toy Example



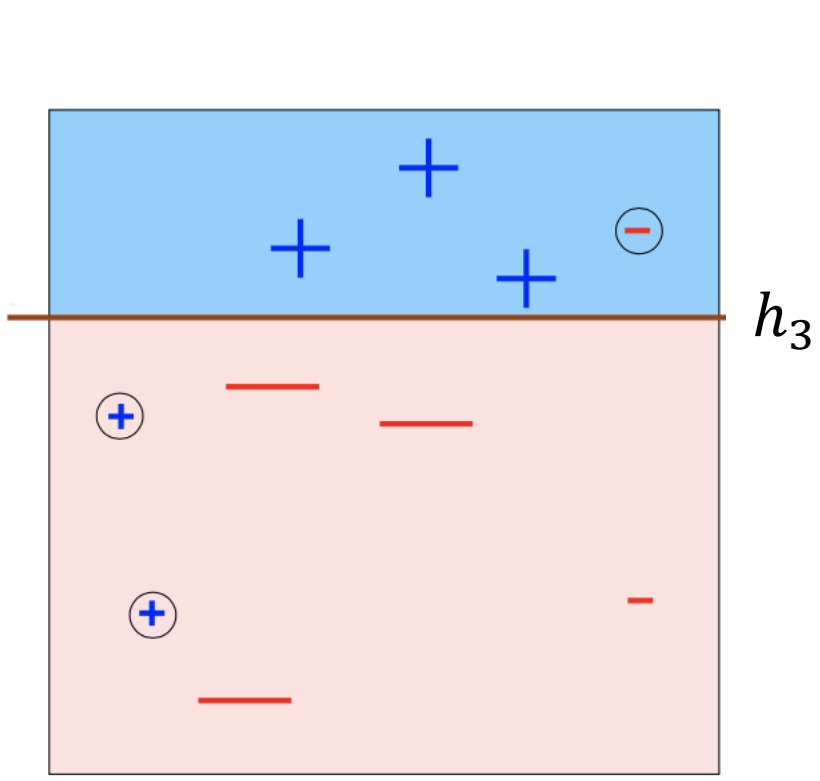
$\epsilon_1 = 0.30$

$\alpha_1 = 0.42$



$\epsilon_2 = 0.21$

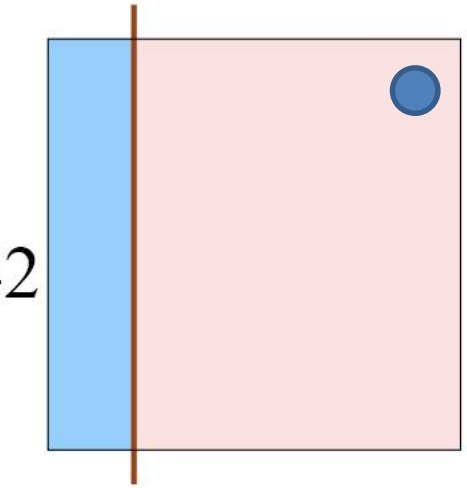
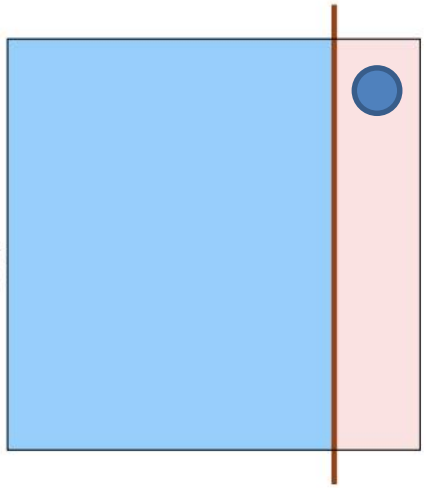
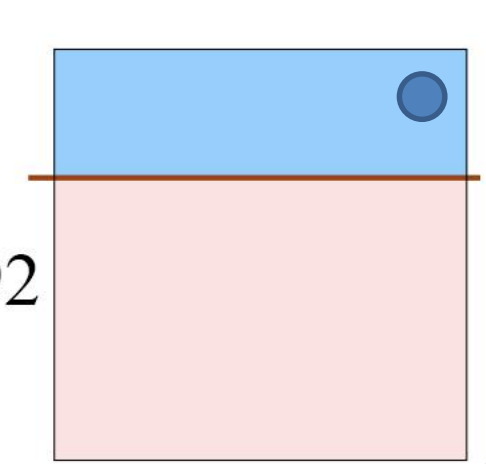
$\alpha_2 = 0.65$

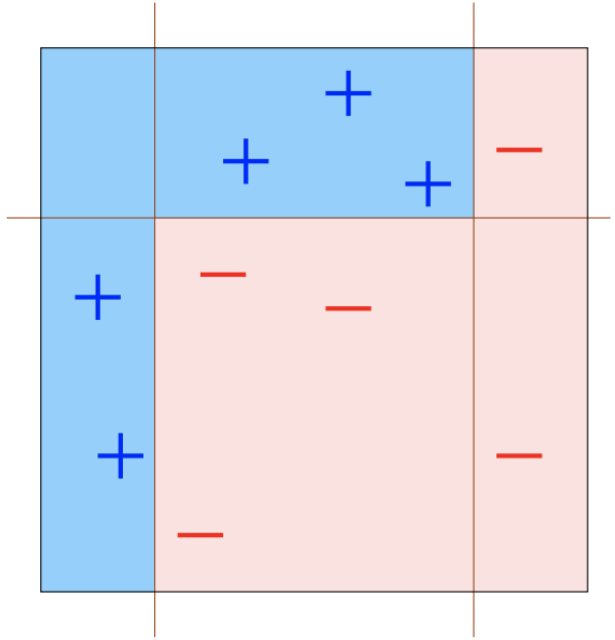


$\epsilon_3 = 0.14$

$\alpha_3 = 0.92$

Boosting - Toy Example

H_{final} =sign(0.42  +0.65  +0.92 )

= 

H = sign[0.42*(−1) + 0.65*(−1) + 0.92*(+1)]
= sign[−0.15]
= −1

算法原理

- 那么问题来了！
  - 1. 对每一个 “new model” 有什么要求？
  - 2. 为什么子模型的叠加可以提高总模型的效果？
  - 3. 样本权值$D_t(i)$是如何调整的？
  - 4. α_t的表达式是如何确定的？

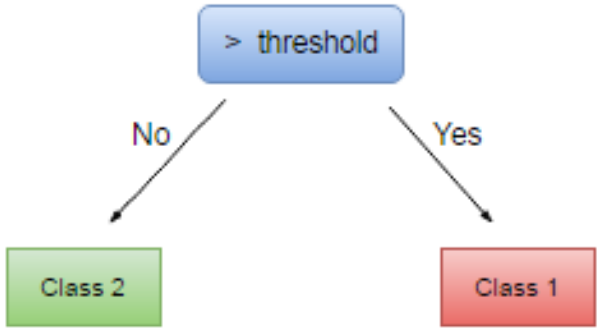
$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(i)\delta(h(\mathbf{x}_i) \neq y_i)$$

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}\right)$$

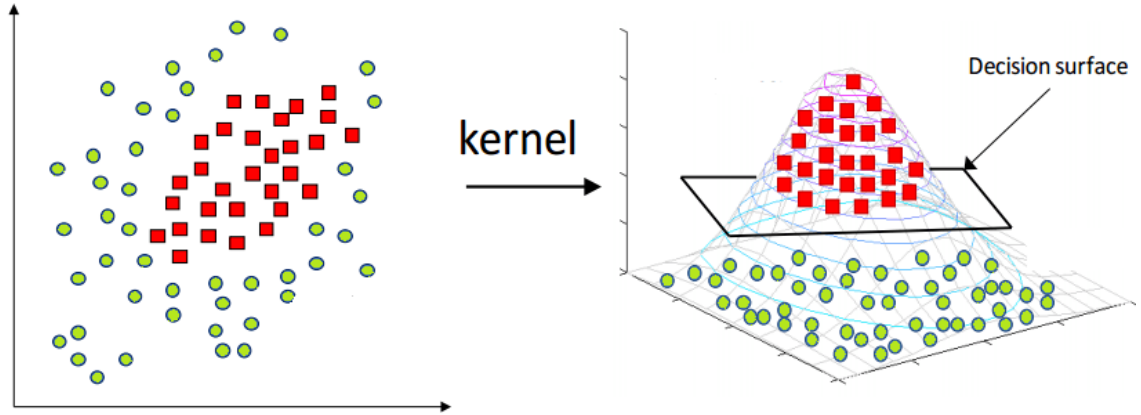
算法原理

- “New model” - Weak learner
  - A ‘weak’ learner is just one which performs just slightly better than random guessing. —— Schapire, 1990

  - Decision stump



  - SVM



算法原理

- The Strength of Weak Learnability -- Schapire, 1990
  - Boosting is based on the question posed by Kearns and Valiant (1988, 1989)

"Can a set of weak learners create a single strong learner?"

Machine Learning, 5, 197-227 (1990)

The Strength of Weak Learnability

ROBERT E. SCHAPIRE (rs@theory.lcs.mit.edu)
MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139

Abstract. This paper addresses the problem of improving the accuracy of an hypothesis output by a learning algorithm in the distribution-free (*PAC*) learning model. A concept class is *learnable* (or *strongly learnable*) if, given access to a source of examples of the unknown concept, the learner with high probability is able to output an hypothesis that is correct on all but an arbitrarily small fraction of the instances. The concept class is *weakly learnable* if the learner can produce an hypothesis that performs only slightly better than random guessing. In this paper, it is shown that these two notions of learnability are equivalent.

A method is described for converting a weak learning algorithm into one that achieves arbitrarily high accuracy. This construction may have practical applications as a tool for efficiently converting a mediocre learning algorithm into one that performs extremely well. In addition, the construction has some interesting theoretical consequences, including a set of general upper bounds on the complexity of any strong learning algorithm as a function of the allowed error ϵ.

Keywords. Machine learning, learning from examples, learnability theory, PAC learning, polynomial-time identification.

1. Introduction

Since Valiant's pioneering paper (1984), interest has flourished in the so-called *distribution-free* or *probably approximately correct* (PAC) model of learning. In this model, the learner tries to identify an unknown concept based on randomly chosen examples of the concept. Examples are chosen according to a fixed but unknown and arbitrary distribution on the space of instances. The learner's task is to find an hypothesis or prediction rule of his own that correctly classifies new instances as positive or negative examples of the concept. With high probability, the hypothesis must be correct for all but an arbitrarily small fraction of the instances.

Often, the inference task includes a requirement that the output hypothesis be of a specified form. In this paper, however, we will instead be concerned with a representation-independent model of learning in which the learner may output any hypothesis that can be used to classify instances in polynomial time.

A class of concepts is *learnable* (or *strongly learnable*) if there exists a polynomial-time algorithm that achieves low error with high confidence for all concepts in the class. A weaker model of learnability, called *weak learnability*, drops the requirement that the learner be able to achieve arbitrarily high accuracy; a weak learning algorithm need only output an hypothesis that performs slightly better (by an inverse polynomial) than random guessing. The notion of weak learnability was introduced by Kearns and Valiant (1988; 1989) who left open the question of whether the notions of strong and weak learnability are equivalent. This question was termed the *hypothesis boosting problem* since showing the notions are equivalent requires a method for boosting the low accuracy of a weak learning algorithm's hypotheses.

算法原理

- $D_t(i)$ & α_t

损失函数

$$\epsilon_t(h) = \sum_{i=1}^{m} D_t(i)\delta(h(\mathbf{x}_i) \neq y_i)$$

子模型训练目标

$$h_t = \text{argmin}_h\, \epsilon_t(h)$$

模型权重

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}\right)$$

样本权重

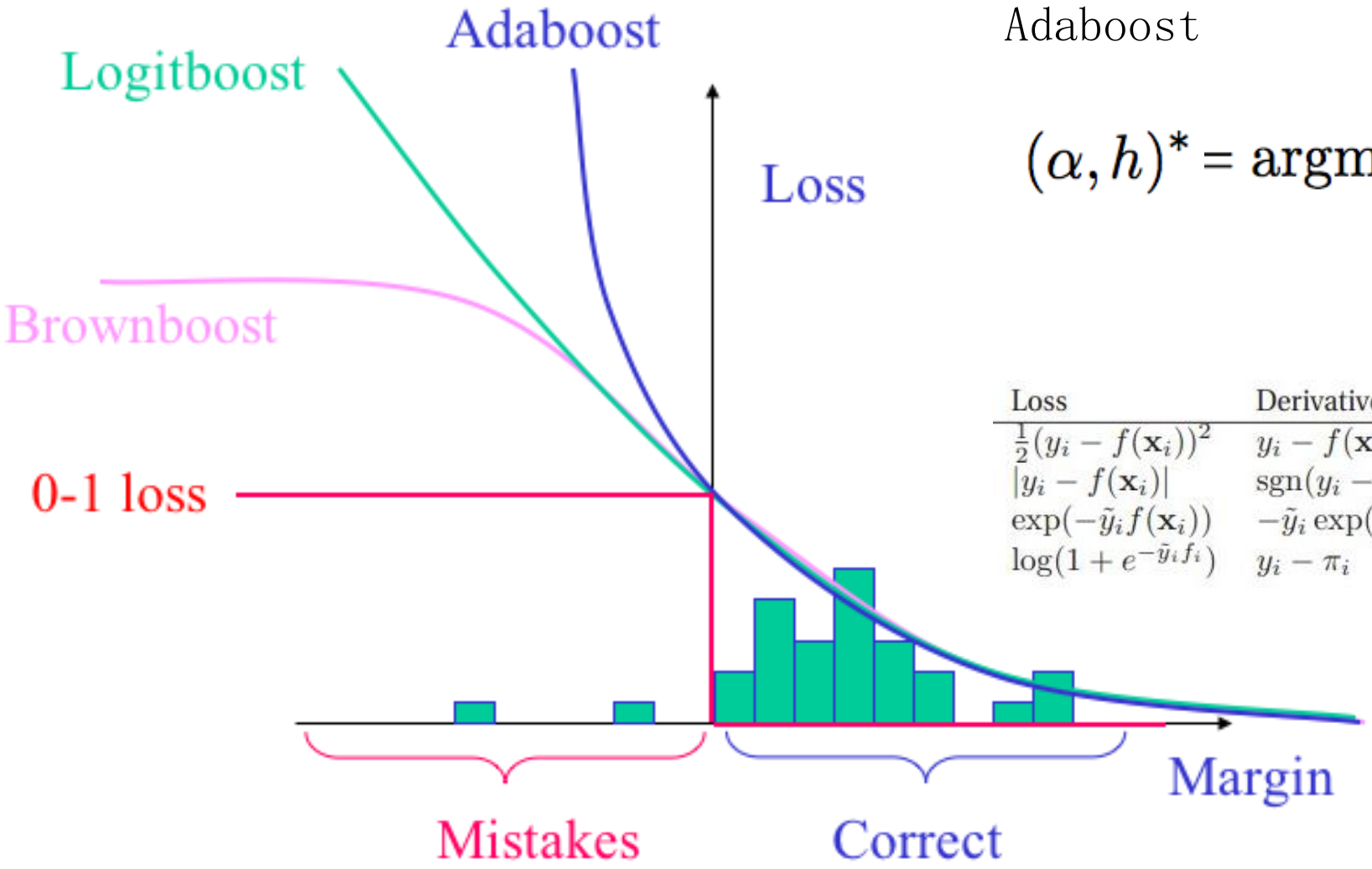
$$D_{t+1}(i) = \frac{1}{Z_t} D_t(i)\exp\left[-\alpha_t y_i h_t(x_i)\right]$$

$$\exp\left[-y_i\alpha_t h_t(\mathbf{x}_i)\right] = \begin{cases} \exp\left[-\alpha_t\right] < 1 & \text{if } h_t(\mathbf{x}_i) = y_i \\ \exp\left[\alpha_t\right] > 1 & \text{if } h_t(\mathbf{x}_i) \neq y_i \end{cases}$$

Adaboost (Adaptive Boosting)

$$(\alpha, h)^* = \text{argmin} \sum_{i=1}^{m} \exp\left[-y_i\langle\alpha, h(\mathbf{x}_i)\rangle\right]$$

算法原理



Adaboost

$$(\alpha, h)^* = \operatorname{argmin} \sum_{i=1}^{m} \exp\left[-y_i \langle \alpha, h(\mathbf{x}_i) \rangle\right]$$

Loss	Derivative	f^*	Algorithm
$\frac{1}{2}(y_i - f(\mathbf{x}_i))^2$	$y_i - f(\mathbf{x}_i)$	$\mathbb{E}[y\|\mathbf{x}_i]$	L2Boosting
$\|y_i - f(\mathbf{x}_i)\|$	$\operatorname{sgn}(y_i - f(\mathbf{x}_i))$	$\operatorname{median}(y\|\mathbf{x}_i)$	Gradient boosting
$\exp(-\tilde{y}_i f(\mathbf{x}_i))$	$-\tilde{y}_i \exp(-\tilde{y}_i f(\mathbf{x}_i))$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	AdaBoost
$\log(1 + e^{-\tilde{y}_i f_i})$	$y_i - \pi_i$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	LogitBoost

算法原理

- Boosting Methods

Discrete AdaBoost [Freund and Schapire (1996b)]

1. Start with weights $w_i = 1/N, i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$.

Real AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier to obtain a class probability estimate $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$, using weights w_i on the training data.
   (b) Set $f_m(x) \leftarrow \frac{1}{2} \log p_m(x)/(1 - p_m(x)) \in R$.
   (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} f_m(x)]$.

Gentle AdaBoost

1. Start with weights $w_i = 1/N, i = 1, 2, \ldots, N, F(x) = 0$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the regression function $f_m(x)$ by weighted least-squares of y_i to x_i with weights w_i.
   (b) Update $F(x) \leftarrow F(x) + f_m(x)$.
   (c) Update $w_i \leftarrow w_i \exp(-y_i f_m(x_i))$ and renormalize.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$.

LogitBoost (two classes)

1. Start with weights $w_i = 1/N$ $i = 1, 2, \ldots, N$, $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Compute the working response and weights

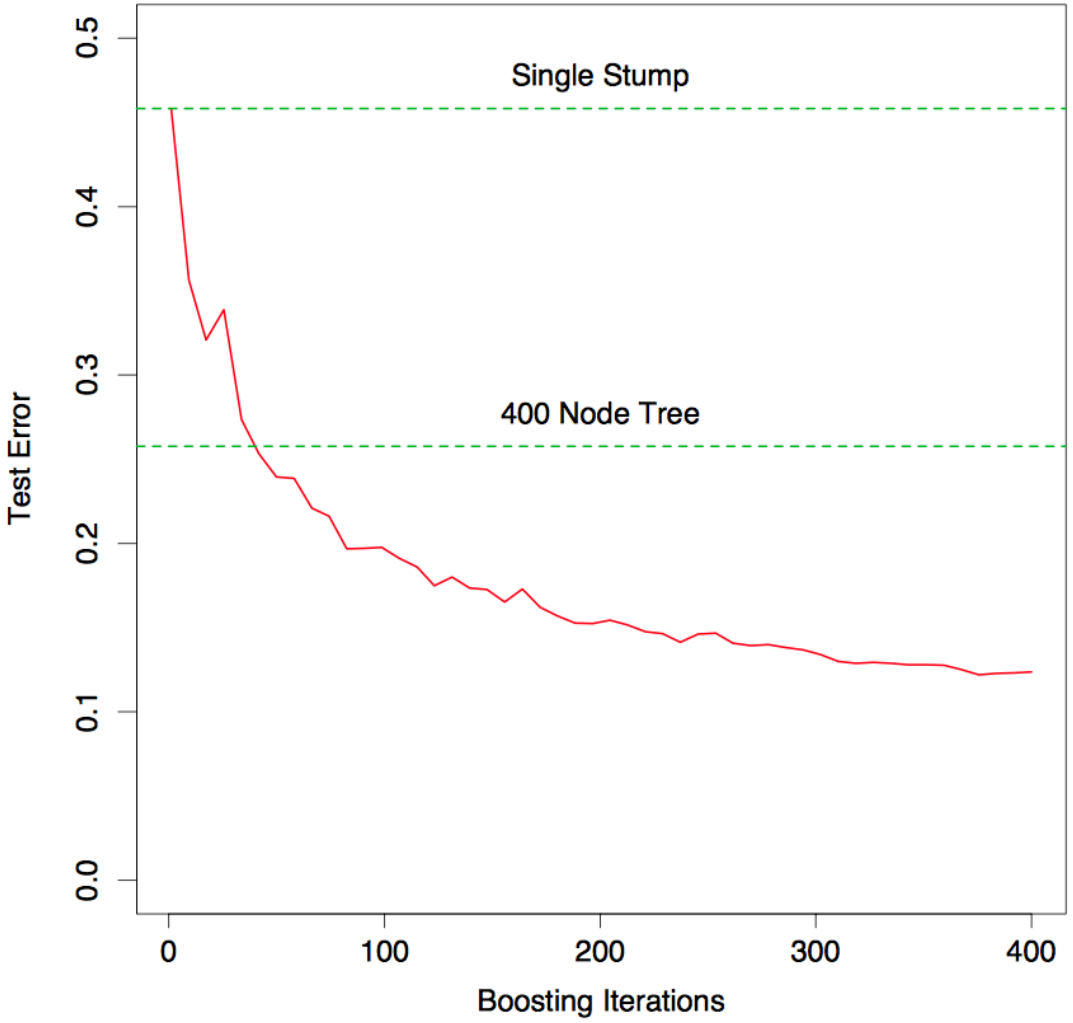
$$z_i = \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))},$$

$$w_i = p(x_i)(1 - p(x_i)).$$

   (b) Fit the function $f_m(x)$ by a weighted least-squares regression of z_i to x_i using weights w_i.
   (c) Update $F(x) \leftarrow F(x) + \frac{1}{2} f_m(x)$ and $p(x) \leftarrow (e^{F(x)})/(e^{F(x)} + e^{-F(x)})$.
3. Output the classifier $\text{sign}[F(x)] = \text{sign}[\sum_{m=1}^{M} f_m(x)]$.

优劣分析
应用总结

优劣分析



Some characteristics of different learning methods.
Key: ●= good, ●=fair, and ●=poor.

Characteristic	Neural Nets	SVM	CART	GAM	KNN, Kernel	Gradient Boost
Natural handling of data of "mixed" type	●	●	●	●	●	●
Handling of missing values	●	●	●	●	●	●
Robustness to outliers in input space	●	●	●	●	●	●
Insensitive to monotone transformations of inputs	●	●	●	●	●	●
Computational scalability (large N)	●	●	●	●	●	●
Ability to deal with irrelevant inputs	●	●	●	●	●	●
Ability to extract linear combinations of features	●	●	●	●	●	●
Interpretability	●	●	●	●	●	●
Predictive power	●	●	●	●	●	●

应用总结

- GBDT (Gradient Boosting Decision Tree)
  - Boosting & Gradient boosting
  - Adaboost

$$\{(x_i, y_i)\}_{i=1}^{n}$$

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t(h_t)}{\epsilon_t(h_t)}\right)$$

$$G(x) = \text{sign}\left[\sum_{m=1}^{M}\alpha_m G_m(x)\right]$$

  - Gradient boosting

$$\{(x_i, r_{im})\}_{i=1}^{n}$$

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

$$\gamma_m = \arg\min_{\gamma}\sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right)$$

应用总结

- XGBoost - a system
  - XGBoost provides a parallel tree boosting (GBDT) that solve many data science problems in a fast and accurate way.
  - Technical Highlights
    - Sparse aware tree learning
    - Distributed weighted quantile sketch
    - Cache aware learning algorithm
  - Impact
    - XGBoost is one of the most frequently used package to win machine learning challenges
    - Can solve billion scale problems with few resources

研究现状

- Schapire, 1990
  - The Strength of Weak Learnability
- Schapire, etc. 1993
  - First application of boosting idea to a real world OCR(Optical Character Recognition) task
- CVPR 2017
  - Fast Boosting Based Detection Using Scale Invariant Multimodal Multiresolution Filtered Features
- NIPS 2016
  - SEBOOST – Boosting Stochastic Learning Using Subspace Optimization Techniques
  - Boosting with Abstention
  - Incremental Boosting Convolutional Neural Network for Facial Action Unit Recognition

道德经

知人者智，自知者明。

胜人者有力，自胜者强。

知足者富。

强行者有志。

不失其所者久。

死而不亡者，寿。

谢谢！