Chapter 13

Ensemble Learning

Ensemble methods

Machine learning competition with a $1 million prize

Leaderboard

Display top 20 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
1	The Ensemble	0.8553	10.10	2009-07-26 18:38:22
2	BellKor's Pragmatic Chaos	0.8554	10.09	2009-07-26 18:18:28
	Grand Prize - RMSE <= 0.8563			
3	Grand Prize Team	0.8571	9.91	2009-07-24 13:07:49
4	Opera Solutions and Vandelay United	0.8573	9.89	2009-07-25 20:05:52
5	Vandelay Industries !	0.8579	9.83	2009-07-26 02:49:53
6	PragmaticTheory	0.8582	9.80	2009-07-12 15:09:53
7	BellKor in BigChaos	0.8590	9.71	2009-07-26 12:57:25
8	Dace	0.8603	9.58	2009-07-24 17:18:43
9	Opera Solutions	0.8611	9.49	2009-07-26 18:02:08
10	BellKor	0.8612	9.48	2009-07-26 17:19:11
11	BigChaos	0.8613	9.47	2009-06-23 23:06:52
12	Feeds2	0.8613	9.47	2009-07-24 20:06:46
	Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos			
13	xiangliang	0.8633	9.26	2009-07-21 02:04:40
14	Gravity	0.8634	9.25	2009-07-26 15:58:34
15	Ces	0.8642	9.17	2009-07-25 17:42:38
16	Invisible Ideas	0.8644	9.14	2009-07-20 03:26:12
17	Just a guy in a garage	0.8650	9.08	2009-07-22 14:10:42
18	Craig Carmichael	0.8656	9.02	2009-07-25 16:00:54
19	J Dennis Su	0.8658	9.00	2009-03-11 09:41:54
20	acmehill	0.8659	8.99	2009-04-16 06:29:35
	Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
	Cinematch score on quiz subset - RMSE = 0.9514			

Bias-Variance Decomposition

- Recall, we treat predictions y at a query $\mathbf{x}$ as a random variable (where the randomness comes from the choice of dataset), $y_\star$ is the optimal deterministic prediction, t is a random target sampled from the true conditional $p(t|\mathbf{x})$.

$$\mathbb{E}[(y-t)^2] = \underbrace{(y_\star - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\text{Var}(y)}_{\text{variance}} + \underbrace{\text{Var}(t)}_{\text{Bayes error}}$$

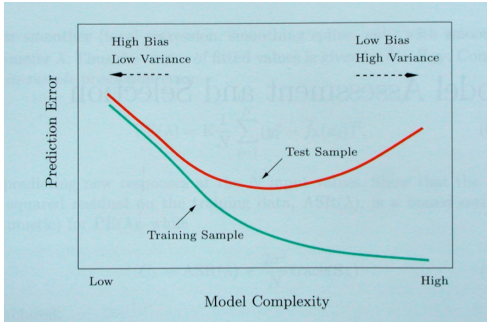
- Bias/variance decomposes the expected loss into three terms:
  - bias: how wrong the expected prediction is (corresponds to underfitting)
  - variance: the amount of variability in the predictions (corresponds to overfitting)
  - Bayes error: the inherent unpredictability of the targets

Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners are good**
  - e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - Low variance, don't usually overfit

- **Simple (a.k.a. weak) learners are bad**
  - High bias, can't solve hard learning problems

- Can we make weak learners always good???
  - **No!!!**
  - **But often yes…**



Ensemble Methods

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**

- **Output class:** (Weighted) vote of each classifier
  - Classifiers that are most "sure" will vote with more conviction
  - Classifiers will be most "sure" about a particular part of the space
  - On average, do better than single classifier!

- **But how do you ???**
  - force classifiers to learn about different parts of the input space?
  - weigh the votes of different classifiers?

only one training set
where do multiple models come from?

Ensemble Method 1

For low bias but high variance models

Bagging!!!

- **Bias: unchanged**, since the averaged prediction has the same expectation

$$\mathbb{E}[y] = \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \mathbb{E}[y_i]$$

- **Variance: reduced**, since we're averaging over independent samples

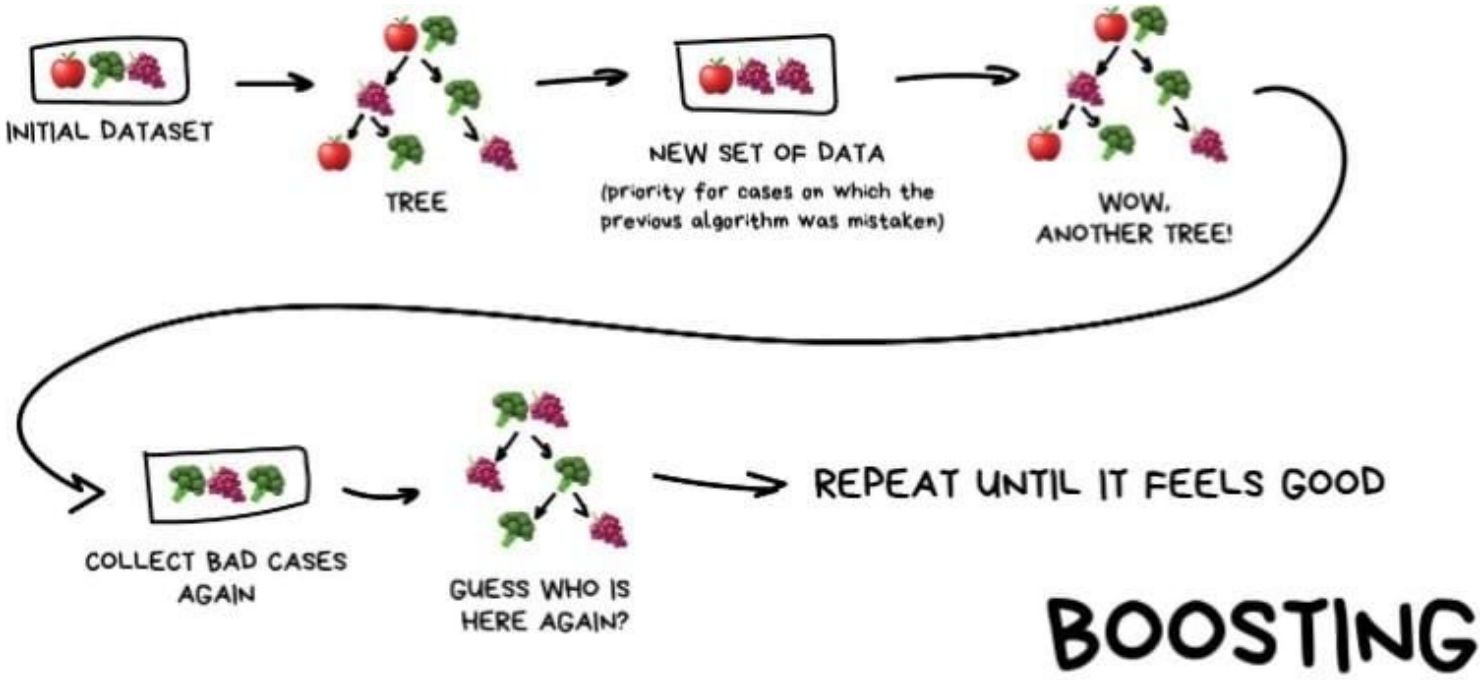
$$\mathrm{Var}[y] = \mathrm{Var}\left[\frac{1}{m}\sum_{i=1}^{m} y_i\right] = \frac{1}{m^2}\sum_{i=1}^{m}\mathrm{Var}[y_i] = \frac{1}{m}\mathrm{Var}[y_i].$$

Ensemble Method 2

For high bias but low variance models

Sequentially generate weak learners to handle wrongly classified samples, hence reduce the bias.

Boosting!!!

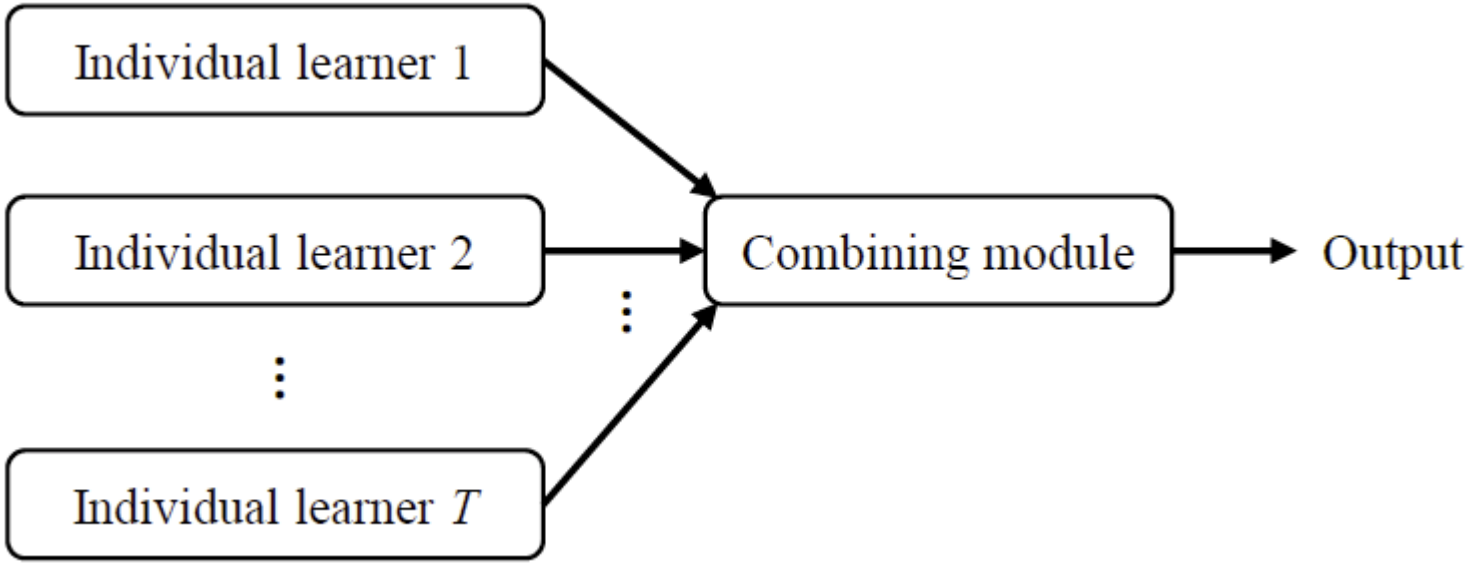


Ensemble Learning

- Individual and Ensemble
- Bagging and Random Forest
- Boosting
  - Adaboost
- Combination Strategies
  - Averaging
  - Voting
  - Combining by Learning
- Diversity
  - Error-Ambiguity Decomposition
  - Diversity Measures
  - Diversity Generation

Individual and Ensemble

- Ensemble learning trains and combines multiple learners to solve a learning problem.



Individual and Ensemble

- Taking binary classification as an example, suppose three classifiers are applied to three testing samples, where √ indicate the correct classifications, × indicate the incorrect classifications. The classification of ensemble learning is made by voting.

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
Ensemble	√	√	√

(a) Ensemble helps.

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
Ensemble	√	√	×

(b) Ensemble doesn't help.

	Testing sample 1	Testing sample 2	Testing sample 3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
Ensemble	×	×	×

(c) Ensemble hurts.

- Individual learners should be "accurate and diverse".

Individual and Ensemble

- Consider binary classification, the error rate of each base learner is

$$P\left(h_i\left(\boldsymbol{x}\right) \neq f\left(\boldsymbol{x}\right)\right) = \epsilon$$

- Suppose ensemble learning combines the T base learners by voting, then the ensemble will make an correct classification if more than half of the base learners are correct:

$$F(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{T} h_i(\boldsymbol{x})\right)$$

Individual and Ensemble

- Assuming the error rates of base learners are independent, then, from *Hoeffding's inequality*, the error rate of the ensemble is

$$P\left(F(\boldsymbol{x}) \neq f(\boldsymbol{x})\right) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k}$$

$$\leq \exp\left(-\frac{1}{2} T \left(1-2\epsilon\right)^2\right)$$

- The above equation shows that as the number of base learners in the ensemble increases, the error rate decreases exponentially and eventually approaches zero.

Individual and Ensemble

- The above analysis made a critical assumption that the error rates of base learners are independent.
- This assumption is invalid in practice since the learners are trained to solve the same problem and thus cannot be independent.
- In fact, accuracy and diversity are two conflicted properties of individual learners.
- The generation and combination of "accurate and diverse" individual learners are the fundamental issues in ensemble learning.

Ensemble Learning

- Individual and Ensemble
- Bagging and Random Forest
- Boosting
  - Adaboost
- Combination Strategies
  - Averaging
  - Voting
  - Combining by Learning
- Diversity
  - Error-Ambiguity Decomposition
  - Diversity Measures
  - Diversity Generation

Bagging and Random Forest

Machine Learning, 24, 123–140 (1996)
© 1996 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Bagging Predictors

LEO BREIMAN leo@stat.berkeley.edu
Statistics Department, University of California, Berkeley, CA 94720

Editor: Ross Quinlan

Abstract. Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor. The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote when predicting a class. The multiple versions are formed by making bootstrap replicates of the learning set and using these as new learning sets. Tests on real and simulated data sets using classification and regression trees and subset selection in linear regression show that bagging can give substantial gains in accuracy. The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.

Keywords: Aggregation, Bootstrap, Averaging, Combining

Bagging

- Bagging = Bootstrap AGGregatING

Algorithm 8.2 Bagging.

Input: Training set: $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of training rounds T.

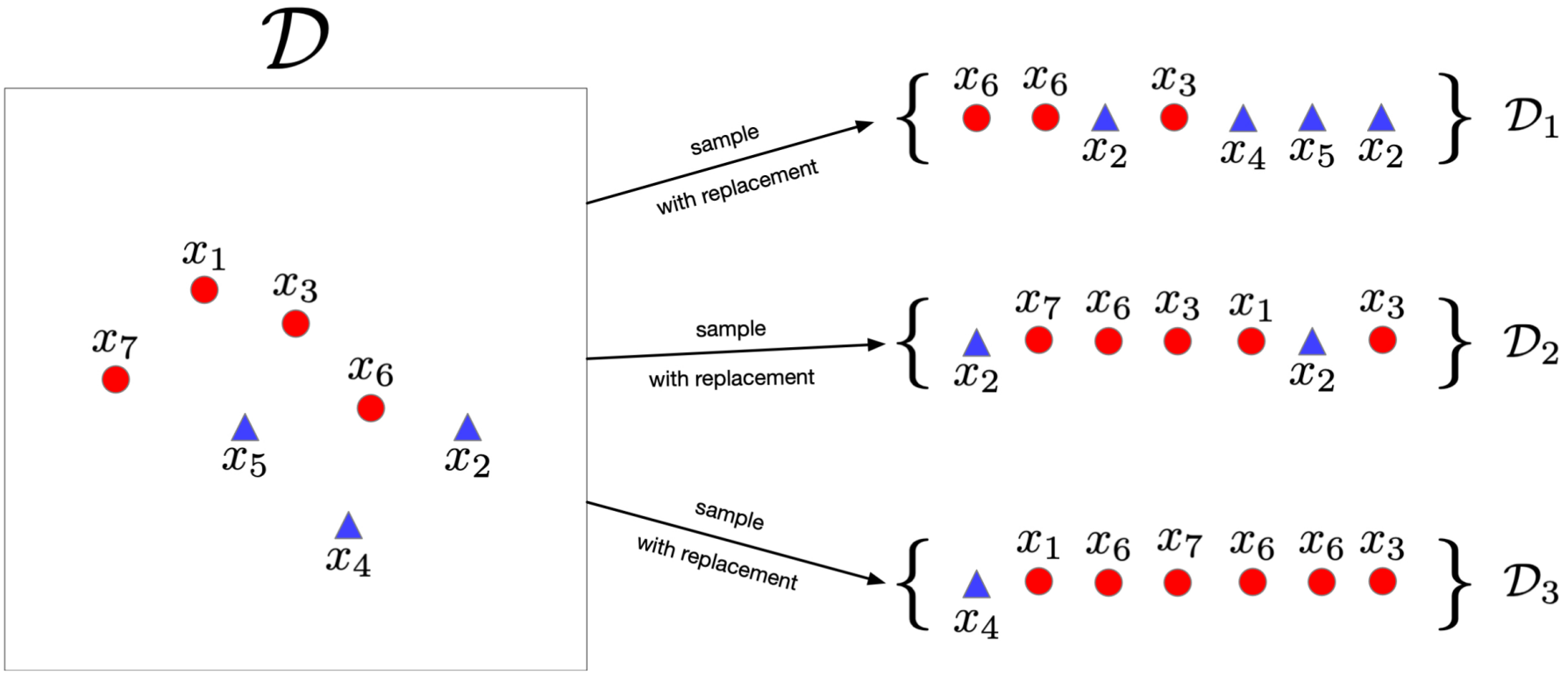
Process:

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$.
3: **end for**

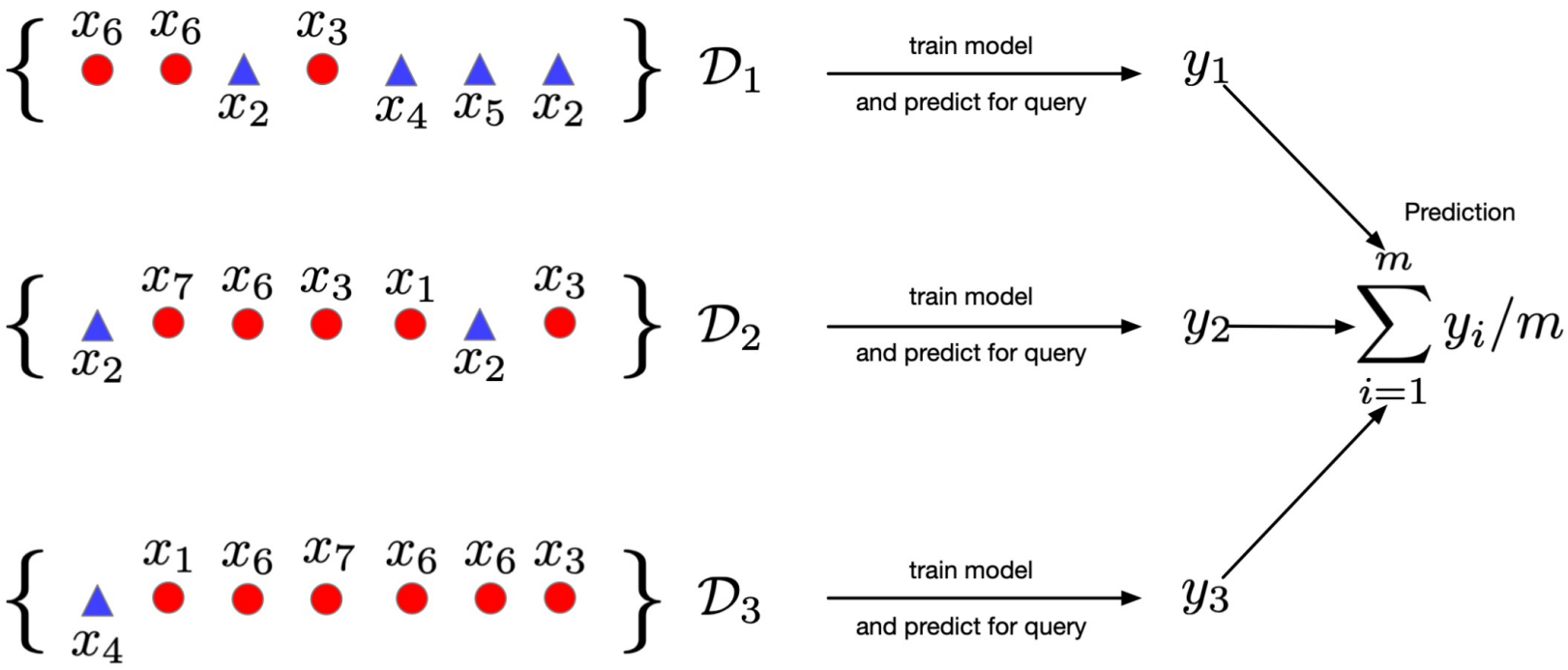
Output: $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$.

The bootstrap is one of the most important ideas in all of statistics!

Bagging


$\mathcal{D}$
x_1
x_3
x_7
x_6
x_5
x_2
x_4
sample
with replacement
x_6 x_6 x_3 x_2 x_4 x_5 x_2
$\mathcal{D}_1$
sample
with replacement
x_7 x_6 x_3 x_1 x_3 x_2 x_2
$\mathcal{D}_2$
sample
with replacement
x_1 x_6 x_7 x_6 x_6 x_3 x_4
$\mathcal{D}_3$

Bagging



predicting on a query point x

Random Forests

- Random Forests = bagged decision trees, with one extra trick to decorrelate the predictions
  - When choosing each node of the decision tree, choose a random set of input features, and only consider splits on those features
- Random forests are probably the best black-box machine learning algorithm — they often work well with no tuning whatsoever.
  - one of the most widely used algorithms in Kaggle competitions

Bagging Summary

- Bagging reduces overfitting by averaging predictions.
- Used in most competition winners
  - Even if a single model is great, a small ensemble usually helps.
- Limitations:
  - Does not reduce bias in case of squared error.
  - There is still correlation between classifiers.
    - Random forest solution: Add more randomness.
  - Naive mixture (all members weighted equally).
- Boosting, up next, can be viewed as an approach to weighted ensembling that strongly decorrelates ensemble members.

Ensemble Learning

- Individual and Ensemble
- Bagging and Random Forest
- Boosting
  - Adaboost
- Combination Strategies
  - Averaging
  - Voting
  - Combining by Learning
- Diversity
  - Error-Ambiguity Decomposition
  - Diversity Measures
  - Diversity Generation

Boosting

Idea: given a weak learner, run it multiple times on weighted training set, then let the learned classifiers vote

- The learners are generated sequentially
- Adjust the distribution of the training samples

$$\left\{ x \to \text{sign}(H(x)) \mid H(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \text{ for some } \alpha_1, \ldots, \alpha_T \geq 0 \text{ and } h_1, \ldots, h_T \in \mathcal{H}, T \geq 1 \right\}$$

where $\mathcal{H}$ is the weak predictor class (e.g., decision stumps).

Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September, 1999.
(In Japanese, translation by Naoki Abe.)

A Short Introduction to Boosting

Yoav Freund Robert E. Schapire
AT&T Labs – Research
Shannon Laboratory
180 Park Avenue
Florham Park, NJ 07932 USA
www.research.att.com/~{yoav, schapire}
{yoav, schapire}@research.att.com

Abstract

Boosting is a general method for improving the accuracy of any given learning algorithm. This short overview paper introduces the boosting algorithm AdaBoost, and explains the underlying theory of boosting, including an explanation of why boosting often does not suffer from overfitting as well as boosting's relationship to support-vector machines. Some examples of recent applications of boosting are also described.

Boosting - AdaBoost

- Weight each training example by how incorrectly it was classified
- A strength for each hypothesis

Input: Training set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learner $\mathfrak{L}$;
Number of training rounds T.

Process:

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$; → initially the uniform distribution
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$;
4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;
5: **if** $\epsilon_t > 0.5$ **then break**
6: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;
7:
$$\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}); \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}); \end{cases}$$
$$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t};$$
→ normalization factor
8: **end for**

Output: $F(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$.

算法 8.1（AdaBoost）

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathcal{X}\subseteq\mathbf{R}^n$，$y_i\in\mathcal{Y}=\{-1,+1\}$；弱学习算法；

输出：最终分类器 $G(x)$.

（1）初始化训练数据的权值分布

$$D_1=(w_{11},\cdots,w_{1i},\cdots,w_{1N}),\quad w_{1i}=\frac{1}{N},\quad i=1,2,\cdots,N$$

（2）对 $m=1,2,\cdots,M$

（a）使用具有权值分布 D_m 的训练数据集学习，得到基本分类器

$$G_m(x):\mathcal{X}\to\{-1,+1\}$$

（b）计算 $G_m(x)$ 在训练数据集上的分类误差率

$$e_m=P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i) \tag{8.1}$$

（c）计算 $G_m(x)$ 的系数

$$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m} \tag{8.2}$$

这里的对数是自然对数.

（d）更新训练数据集的权值分布

$$D_{m+1} = (w_{m+1,1},\cdots,w_{m+1,i},\cdots,w_{m+1,N}) \tag{8.3}$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)),\quad i=1,2,\cdots,N \tag{8.4}$$

这里，Z_m 是规范化因子

$$Z_m = \sum_{i=1}^{N} w_{mi}\exp(-\alpha_m y_i G_m(x_i)) \tag{8.5}$$

它使 D_{m+1} 成为一个概率分布.

（3）构建基本分类器的线性组合

$$f(x) = \sum_{m=1}^{M}\alpha_m G_m(x) \tag{8.6}$$

得到最终分类器

$$G(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right) \tag{8.7}$$

■

对 AdaBoost 算法作如下说明：

步骤(1) 假设训练数据集具有均匀的权值分布，即每个训练样本在基本分类器的学习中作用相同，这一假设保证第 1 步能够在原始数据上学习基本分类器 $G_1(x)$.

步骤(2) AdaBoost 反复学习基本分类器，在每一轮 $m=1,2,\cdots,M$ 顺次地执行下列操作：

(a) 使用当前分布 D_m 加权的训练数据集，学习基本分类器 $G_m(x)$.

(b) 计算基本分类器 $G_m(x)$ 在加权训练数据集上的分类误差率：

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{G_m(x_i)\neq y_i} w_{mi} \tag{8.8}$$

这里，w_{mi} 表示第 m 轮中第 i 个实例的权值，$\sum_{i=1}^{N} w_{mi} = 1$. 这表明，$G_m(x)$ 在加权的训练数据集上的分类误差率是被 $G_m(x)$ 误分类样本的权值之和，由此可以看出数据权值分布 D_m 与基本分类器 $G_m(x)$ 的分类误差率的关系.

(c) 计算基本分类器 $G_m(x)$ 的系数 α_m. α_m 表示 $G_m(x)$ 在最终分类器中的重要性. 由式 (8.2) 可知，当 $e_m \leqslant \frac{1}{2}$ 时，$\alpha_m \geqslant 0$，并且 α_m 随着 e_m 的减小而增大，所以分类误差率越小的基本分类器在最终分类器中的作用越大.

（d）更新训练数据的权值分布为下一轮作准备．式 (8.4) 可以写成：

$$w_{m+1,i}=\begin{cases}\dfrac{w_{mi}}{Z_m}\mathrm{e}^{-\alpha_m}, & G_m(x_i)=y_i\\ \dfrac{w_{mi}}{Z_m}\mathrm{e}^{\alpha_m}, & G_m(x_i)\neq y_i\end{cases}$$

由此可知，被基本分类器 $G_m(x)$ 误分类样本的权值得以扩大，而被正确分类样本的权值却得以缩小．两相比较，误分类样本的权值被放大 $\mathrm{e}^{2\alpha_m}=\dfrac{e_m}{1-e_m}$ 倍．因此，误分类样本在下一轮学习中起更大的作用．不改变所给的训练数据，而不断改变训练数据权值的分布，使得训练数据在基本分类器的学习中起不同的作用，这是 AdaBoost 的一个特点．

步骤（3） 线性组合 $f(x)$ 实现 M 个基本分类器的加权表决．系数 α_m 表示了基本分类器 $G_m(x)$ 的重要性，这里，所有 α_m 之和并不为 1．$f(x)$ 的符号决定实例 x 的类，$f(x)$ 的绝对值表示分类的确信度．利用基本分类器的线性组合构建最终分类器是 AdaBoost 的另一特点．

Training error goes to zero

Theorem 0.1. *Suppose the* weak learning assumption *holds for all t: each h_t is better than random guessing: for some $\gamma > 0$,*

$$\epsilon_t \leq 1/2 - \gamma$$

Then the training error

$$\hat{\mathcal{R}}_{01}(\hat{f}) \leq \exp\left(-2\gamma^2 T\right).$$

Boosting - AdaBoost

- Linear combination of base learners

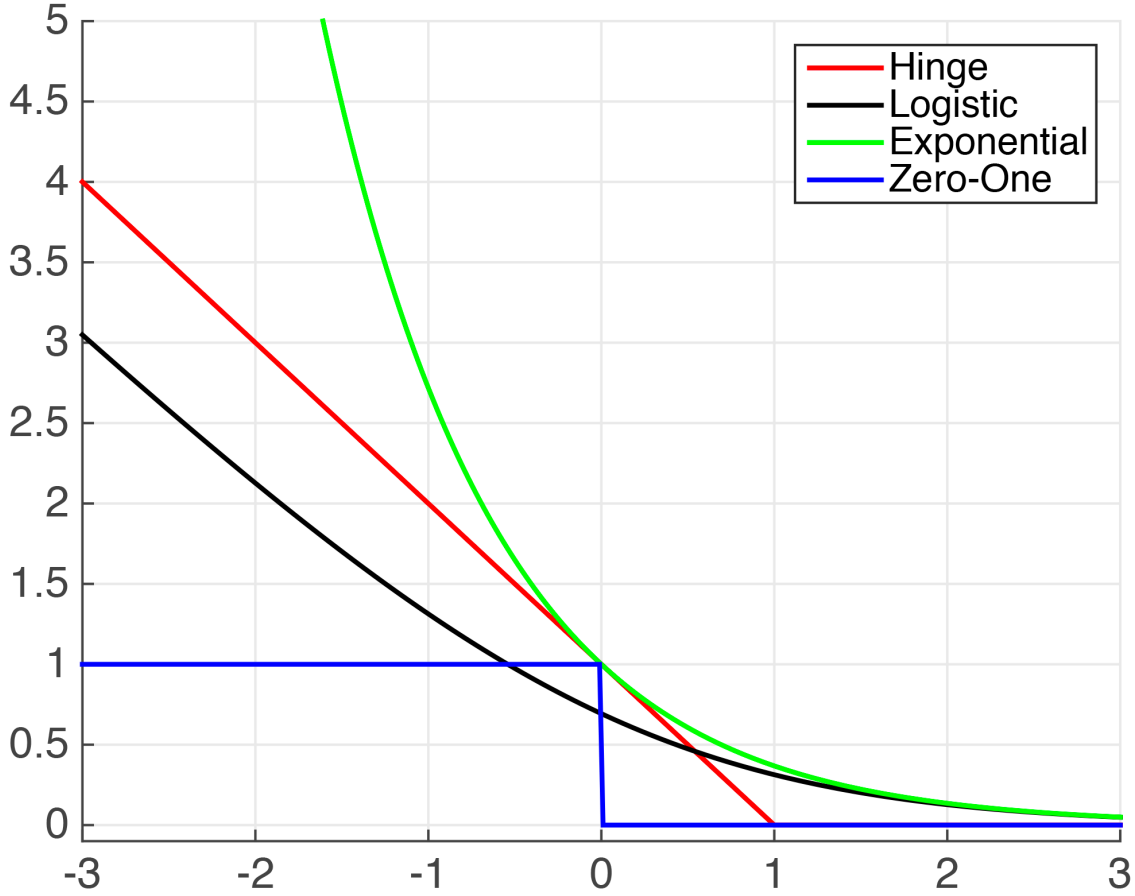
$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

- AdaBoost can then be viewed as optimizing the **exponential loss**:

$$\ell_{\exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}]$$

- WHY? Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(\operatorname{sign}(H(x_i)) \neq y_i) \leq \frac{1}{m}\sum_{i=1}^{m} \exp(-y_i H(x_i))$$



$$L_{exp}(\mathbf{x}, y) \geq L_{0-1}(\mathbf{x}, y)$$

Boosting - AdaBoost

$$\ell_{\exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}]$$

- If $H(x)$ minimizes the exponential loss, then the partial derivative with respect to $H(x)$ is zero:

$$\frac{\partial \ell_{\exp}(H \mid \mathcal{D})}{\partial H(\boldsymbol{x})} = -e^{-H(\boldsymbol{x})}P(f(\boldsymbol{x}) = 1 \mid \boldsymbol{x}) + e^{H(\boldsymbol{x})}P(f(\boldsymbol{x}) = -1 \mid \boldsymbol{x}) = 0$$

$$H(\boldsymbol{x}) = \frac{1}{2}\ln\frac{P(f(\boldsymbol{x}) = 1 \mid \boldsymbol{x})}{P(f(\boldsymbol{x}) = -1 \mid \boldsymbol{x})}$$

$$\begin{aligned}\operatorname{sign}(H(\boldsymbol{x})) &= \operatorname{sign}\left(\frac{1}{2}\ln\frac{P(f(\boldsymbol{x}) = 1 \mid \boldsymbol{x})}{P(f(\boldsymbol{x}) = -1 \mid \boldsymbol{x})}\right) \\ &= \begin{cases} 1, & P(f(\boldsymbol{x}) = 1 \mid \boldsymbol{x}) > P(f(\boldsymbol{x}) = -1 \mid \boldsymbol{x}) \\ -1, & P(f(\boldsymbol{x}) = 1 \mid \boldsymbol{x}) < P(f(\boldsymbol{x}) = -1 \mid \boldsymbol{x}) \end{cases} \\ &= \underset{y\in\{-1,1\}}{\arg\max} P(f(\boldsymbol{x}) = y \mid \boldsymbol{x})\end{aligned}$$

$sign(H(x))$ achieves the Bayes optimal error rate. Hence the exponential loss function is a *consistent* surrogate function of the original 0/1 loss function.

Boosting - AdaBoost

- Adjust the sample distribution based on H_{t-1} such that the base learner h_t in the next round can correct some mistakes made by H_{t-1}.

$$\begin{aligned}\ell_{\exp}(H_{t-1}+h_t \mid \mathcal{D}) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})(H_{t-1}(\boldsymbol{x})+h_t(\boldsymbol{x}))}] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}e^{-f(\boldsymbol{x})h_t(\boldsymbol{x})}]\end{aligned}$$

- Use Taylor expansion to approximate:

$$\begin{aligned}\ell_{\exp}(H_{t-1}+h_t \mid \mathcal{D}) &\simeq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h_t(\boldsymbol{x})+\frac{f^2(\boldsymbol{x})h_t^2(\boldsymbol{x})}{2}\right)\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h_t(\boldsymbol{x})+\frac{1}{2}\right)\right]\end{aligned}$$

Boosting - AdaBoost

- Hence, the ideal classifier is

$$
\begin{aligned}
h_t(\boldsymbol{x}) &= \arg\min_h \ell_{\exp}(H_{t-1} + h \mid \mathcal{D}) \\
&= \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1 - f(\boldsymbol{x})h(\boldsymbol{x}) + \frac{1}{2}\right)\right] \\
&= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})} f(\boldsymbol{x})h(\boldsymbol{x})\right] \\
&= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]} f(\boldsymbol{x})h(\boldsymbol{x})\right],
\end{aligned}
$$

- where $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]$ is a constant. Let D_t denote a distribution

$$
\mathcal{D}_t(\boldsymbol{x}) = \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]}
$$

Boosting - AdaBoost

- According to the definition of mathematical expectation, the ideal classifier is equivalent to

$$h_t(\boldsymbol{x}) = \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]} f(\boldsymbol{x})h(\boldsymbol{x})\right]$$

$$= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[f(\boldsymbol{x})h(\boldsymbol{x})\right] \ .$$

- Since $f(x), h(x) \in \{-1, +1\}$, we have

$$f(\boldsymbol{x})h(\boldsymbol{x}) = 1 - 2\,\mathbb{I}\big(f(\boldsymbol{x}) \neq h(\boldsymbol{x})\big)$$

Boosting - AdaBoost

- The ideal classifier is

$$h_t(\boldsymbol{x}) = \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\big(f(\boldsymbol{x}) \neq h(\boldsymbol{x})\big)\right]$$

- The update rule of the sample distribution is

$$\begin{aligned}
\mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}(\boldsymbol{x})\, e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \\
&= \frac{\mathcal{D}(\boldsymbol{x})\, e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})} e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \\
&= \mathcal{D}_t(\boldsymbol{x}) \cdot e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})} \frac{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]}
\end{aligned}$$

Boosting - Notice

- In each round, check whether the current base learner is better than random guessing.

- To learn from specified sample distributions
  - re-weighting
  - re-sampling

Boosting - AdaBoost

- The base classifier h_t is generated from the distribution D_t. Its weight α_t is estimated by letting $\alpha_t h_t$ minimize the exponential loss function:

$$\begin{aligned}
\ell_{\exp}\left(\alpha_t h_t \mid \mathcal{D}_t\right) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-\alpha_t}\mathbb{I}\left(f\left(\boldsymbol{x}\right)=h_t\left(\boldsymbol{x}\right)\right)+e^{\alpha_t}\mathbb{I}\left(f\left(\boldsymbol{x}\right)\neq h_t\left(\boldsymbol{x}\right)\right)\right] \\
&= e^{-\alpha_t}P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(f\left(\boldsymbol{x}\right)=h_t\left(\boldsymbol{x}\right)\right)+e^{\alpha_t}P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(f\left(\boldsymbol{x}\right)\neq h_t\left(\boldsymbol{x}\right)\right) \\
&= e^{-\alpha_t}\left(1-\epsilon_t\right)+e^{\alpha_t}\epsilon_t
\end{aligned}$$

where $\epsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(h_t(\boldsymbol{x})\neq f(\boldsymbol{x})\right)$

- Set the derivative of the exponential loss function to 0

$$\frac{\partial \ell_{\exp}\left(\alpha_t h_t \mid \mathcal{D}_t\right)}{\partial \alpha_t} = -e^{-\alpha_t}\left(1-\epsilon_t\right)+e^{\alpha_t}\epsilon_t = 0$$

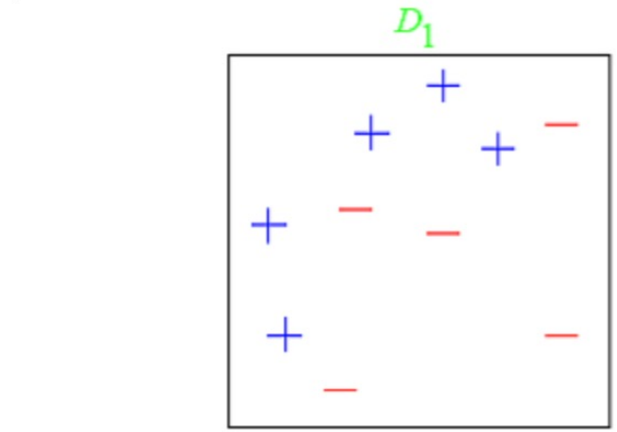
$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

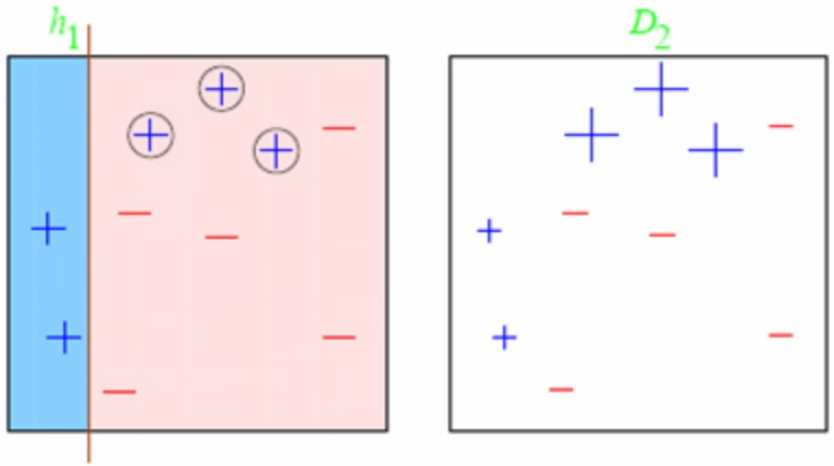
Boosting – AdaBoost Experiment

weak classifiers are decision stumps

(decision tree with a single split)

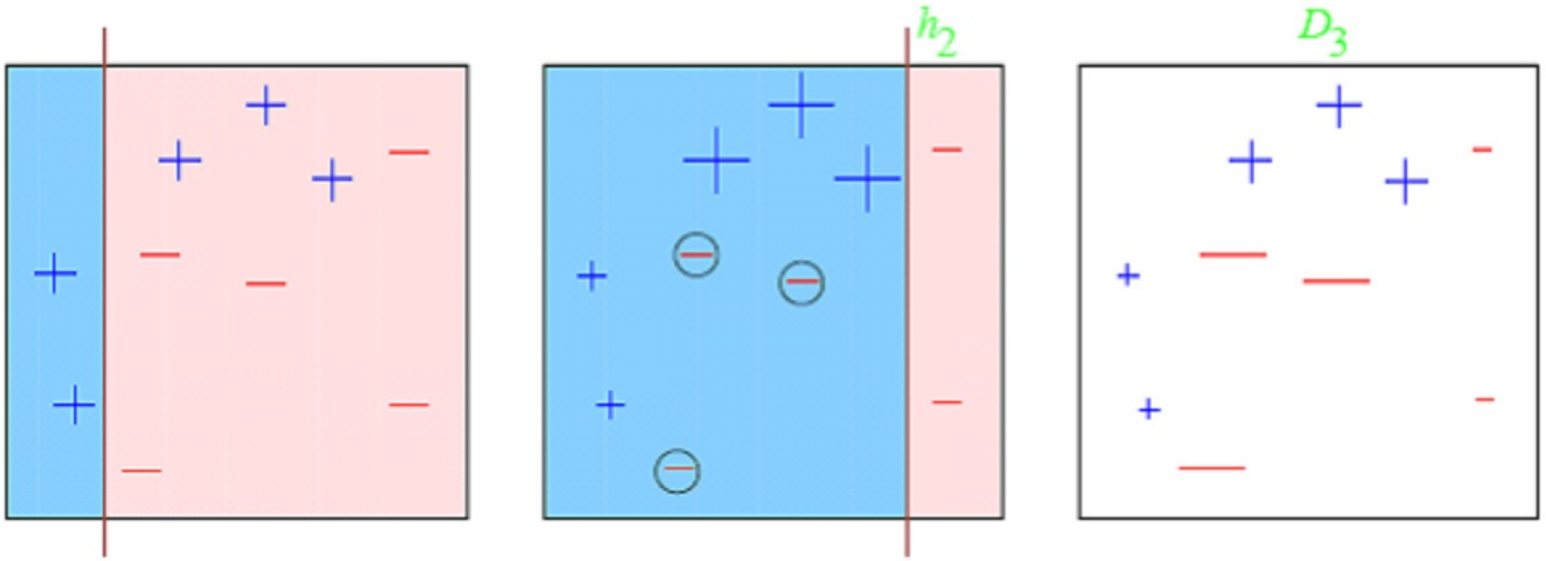
The first round:





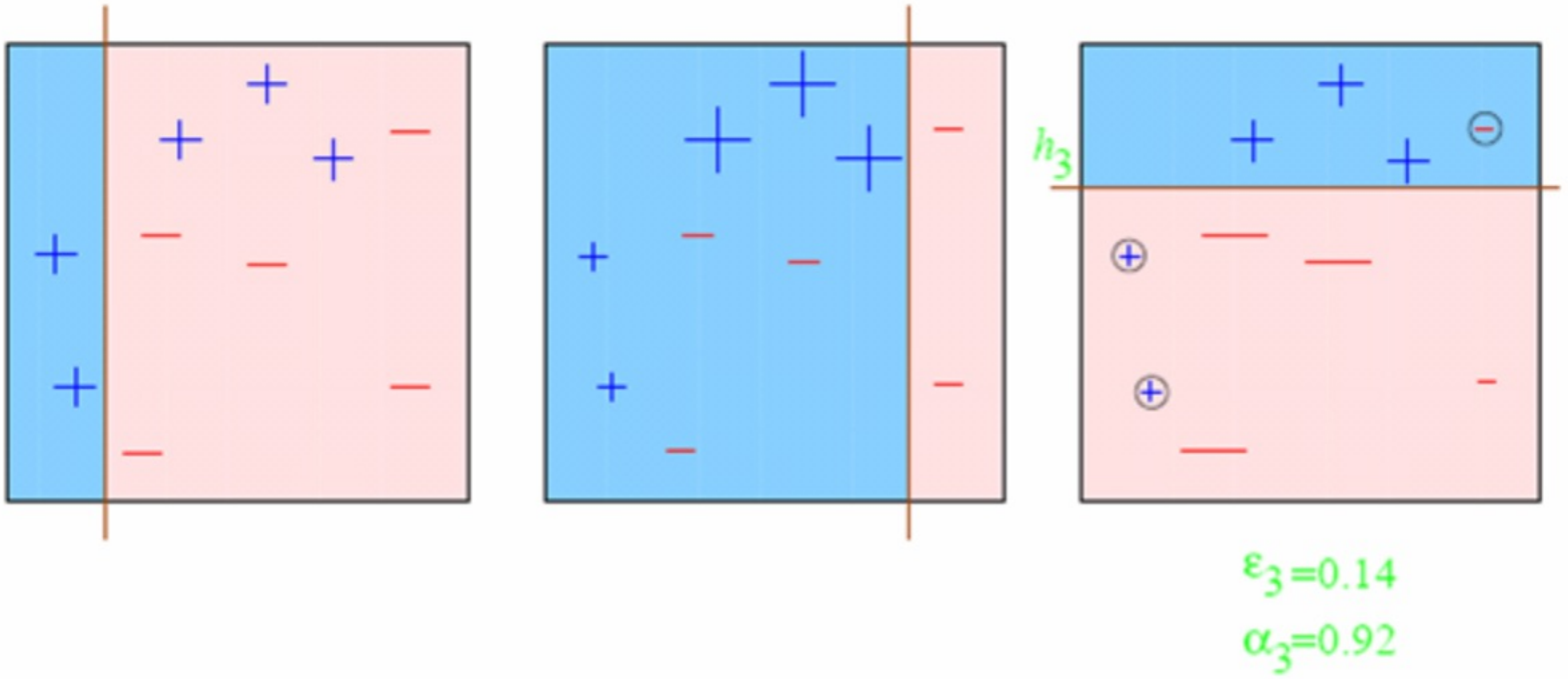
Boosting – AdaBoost Experiment

The second round:



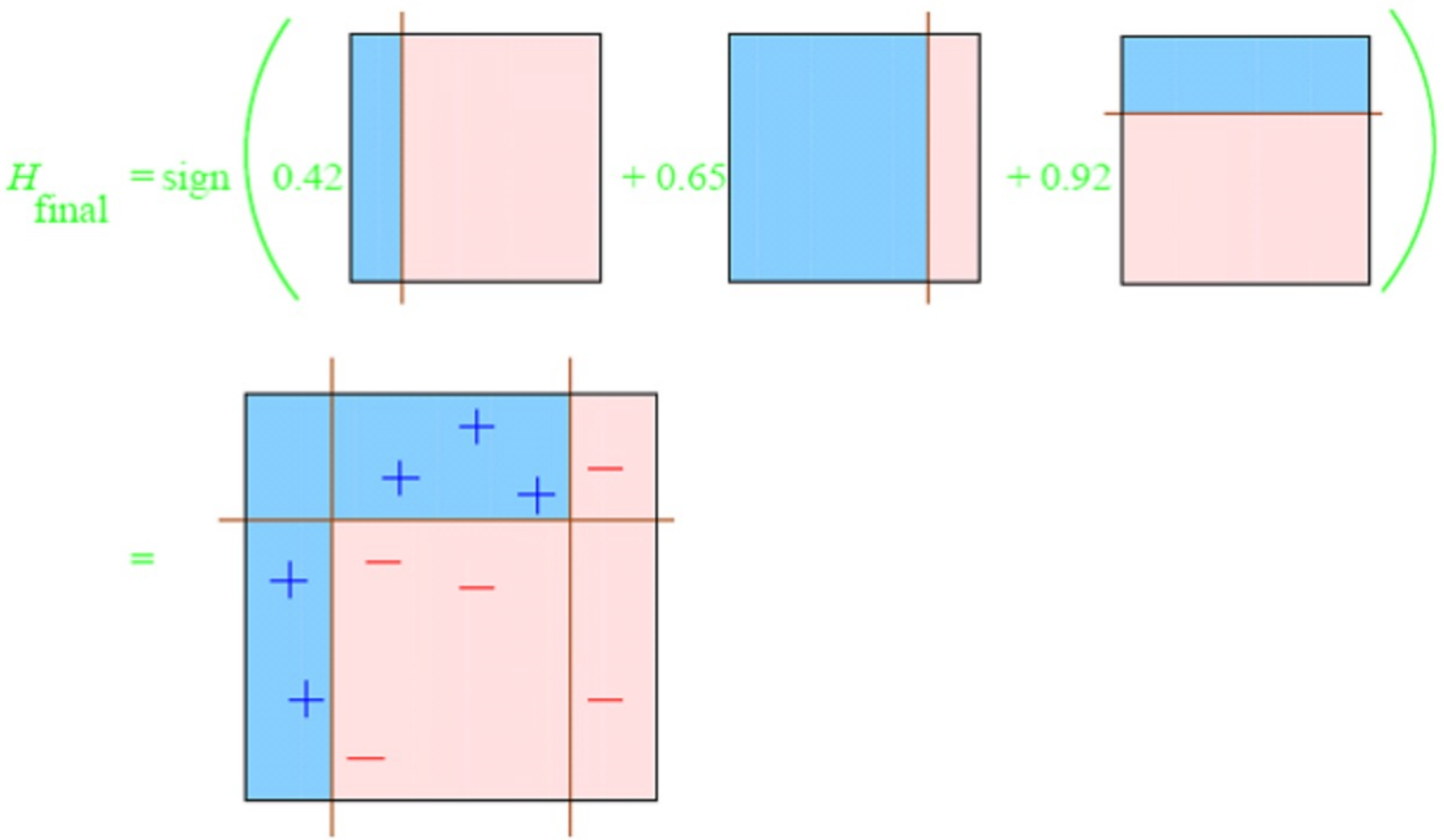
Boosting – AdaBoost Experiment

The third round:



Boosting – AdaBoost Experiment

The final round:



Logistic regression and Boosting

- Logistic regression equivalent to minimizing log loss

$$\frac{1}{m}\sum_{i=1}^{m}\ln(1+\exp(-y_i f(x_i)))$$

- Boosting minimizes similar loss function!!

$$\ell_{\exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}]$$

Both smooth approximations of 0/1 loss!

What you need to know about Boosting

- Combine weak classifiers to obtain very strong classifier
  - Weak classifier – slightly better than random on training data
  - Resulting very strong classifier – can eventually provide zero training error
- AdaBoost algorithm
- Boosting v. Logistic Regression
  - Similar loss functions
  - Single optimization (LR) v. Incrementally improving classification (B)
- Most popular application of Boosting:
  - Boosted decision stumps!
  - Very simple to implement, very effective classifier

Ensemble Learning

- Individual and Ensemble
- Bagging and Random Forest
- Boosting
  - Adaboost
- Combination Strategies
  - Averaging
  - Voting
  - Combining by Learning
- Diversity
  - Error-Ambiguity Decomposition
  - Diversity Measures
  - Diversity Generation

Averaging

Numerical output

- *Simple averaging*

$$H(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\boldsymbol{x}).$$

- *Weighted averaging*

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x})\,, \quad w_i \geq 0 \ \text{ and } \ \sum_{i=1}^{T} w_i = 1\,.$$

Averaging

- Simple averaging is a special case of weighted averaging.
- Weighted averaging has been widely used since the 1950s.
- Other combination methods can all be viewed as its special cases or variants of weighted averaging.
- Weighted averaging can be regarded as a fundamental motivation of ensemble learning studies.
- Weighted averaging is not necessarily better than simple averaging

Voting

For classification

- Majority voting

$$H(\boldsymbol{x}) = \begin{cases} c_j, & \text{if } \sum_{i=1}^{T} h_i^j(\boldsymbol{x}) > 0.5 \sum_{k=1}^{N} \sum_{i=1}^{T} h_i^k(\boldsymbol{x}); \\ \text{reject}, & \text{otherwise.} \end{cases}$$

- Plurality voting

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} h_i^j(\boldsymbol{x})}$$

- Weighted voting

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} w_i h_i^j(\boldsymbol{x})}$$

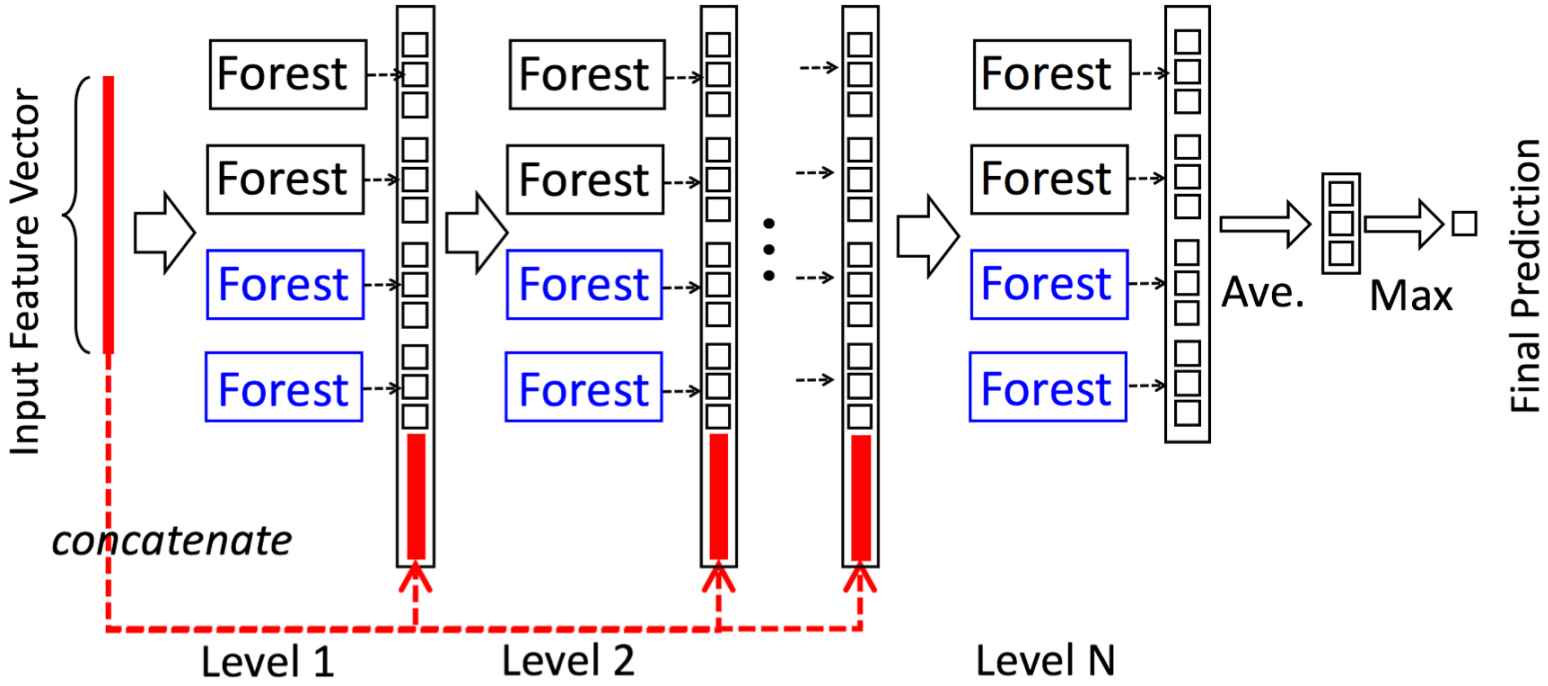
Voting

- Two common value types for the output of h_i
- 1) Class label $h_i^j(\boldsymbol{x}) \in \{0, 1\}$ hard voting
- 2) Class probability $h_i^j(\boldsymbol{x}) \in [0, 1]$ soft voting

calibration

- Confidence values ⟶ probabilities
- If different types of base learners are used, the class probabilities can be converted into class labels before voting.

Stacking



Zhi-Hua Zhou. Deep Forest: Towards an Alternative to Deep Neural Networks. IJCAI'17

Ensemble Learning

- Individual and Ensemble
- Bagging and Random Forest
- Boosting
  - Adaboost
- Combination Strategies
  - Averaging
  - Voting
  - Combining by Learning
- Diversity
  - Error-Ambiguity Decomposition
  - Diversity Measures
  - Diversity Generation

Error-Ambiguity Decomposition

- The *ambiguity* of the learner h_i is defined as

$$A(h_i \mid \boldsymbol{x}) = \big(h_i(\boldsymbol{x}) - H(\boldsymbol{x})\big)^2$$

- The *ambiguity* of the ensemble is defined as

$$\begin{aligned}\overline{A}(h \mid \boldsymbol{x}) &= \sum_{i=1}^{T} w_i A(h_i \mid \boldsymbol{x}) \\ &= \sum_{i=1}^{T} w_i \big(h_i(\boldsymbol{x}) - H(\boldsymbol{x})\big)^2\end{aligned}$$

Error-Ambiguity Decomposition

- The ambiguity term represents the degree of disagreement among individual learners on the sample $\boldsymbol{x}$, which reflects the level of diversity in some sense.
- The squared errors of the individual learner h_i and the ensemble H are, respectively,

$$E(h_i \mid \boldsymbol{x}) = \big(f(\boldsymbol{x}) - h_i(\boldsymbol{x})\big)^2$$

$$E(H \mid \boldsymbol{x}) = \big(f(\boldsymbol{x}) - H(\boldsymbol{x})\big)^2$$

Error-Ambiguity Decomposition

- Let $\overline{E}(h \mid \boldsymbol{x}) = \sum_{i=1}^{T} w_i \cdot E(h_i \mid \boldsymbol{x})$ denote the weighted average error of individual learners, then, we have

$$\begin{aligned}\overline{A}(h \mid \boldsymbol{x}) &= \sum_{i=1}^{T} w_i E(h_i \mid \boldsymbol{x}) - E(H \mid \boldsymbol{x}) \\ &= \overline{E}(h \mid \boldsymbol{x}) - E(H \mid \boldsymbol{x}) \ .\end{aligned}$$

- Let $p(\boldsymbol{x})$ denote the probability density of the sample $\boldsymbol{x}$, for all samples we have

$$\sum_{i=1}^{T} w_i \int A(h_i \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} = \sum_{i=1}^{T} w_i \int E(h_i \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} - \int E(H \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

Error-Ambiguity Decomposition

- The generalization error and the ambiguity term of the learner h_i on all samples are, respectively,

$$E_i = \int E(h_i \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

$$A_i = \int A(h_i \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

- The generalization error of the ensemble is

$$E = \int E(H \mid \boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

- Let $\overline{E} = \sum_{i=1}^{T} w_i E_i$ denote the weighted average error of individual learners, and $\overline{A} = \sum_{i=1}^{T} w_i A_i$ denote the weighted average ambiguity of individual learners. Then

$$E = \overline{E} - \overline{A}$$

Error-Ambiguity Decomposition

- This elegant equation clearly shows that the generalization ability of an ensemble depends on the accuracy and diversity of individual learners. The above analysis is known as the *error-ambiguity decomposition.*

- Why can't we optimize $\bar{E} - \bar{A}$ directly?
  - Direct optimization of $\bar{E} - \bar{A}$ is hard in practice:
    - Both terms are defined in the entire sample space;
    - $\bar{A}$ is not a diversity measure that is directly operable;
    - The above derivation process is only applicable to regression and is difficult to extend to classification.

Diversity Measures

- Diversity measures are for measuring the diversity of individual learners in an ensemble.

- The *contingency table* of the classifiers h_i and h_j for binary classification is

	$h_i = +1$	$h_i = -1$
$h_j = +1$	a	c
$h_j = -1$	b	d

$$a + b + c + d = m$$

Diversity Measures

- Some representative diversity measures:
  - Disagreement Measure

$$dis_{ij} = \frac{b+c}{m}$$

  - Correlation Coefficient

$$\rho_{ij} = \frac{ad-bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}$$

Diversity Measures

- Some representative diversity measures:
  - Q-Statistic

$$Q_{ij} = \frac{ad - bc}{ad + bc} \qquad |Q_{ij}| \geq |\rho_{ij}|$$

  - κ-Statistic

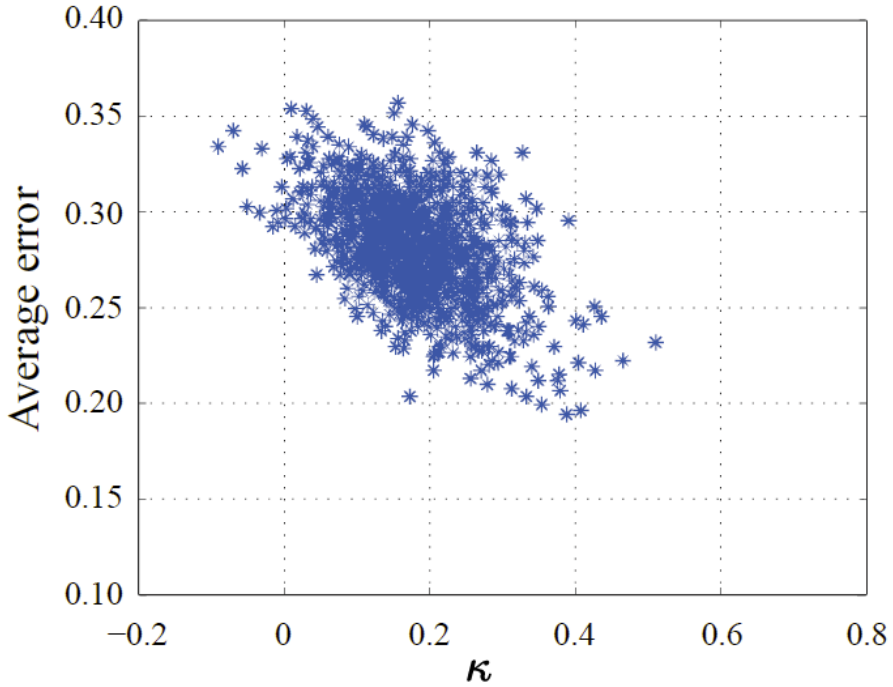
$$\kappa = \frac{p_1 - p_2}{1 - p_2}$$

$$p_1 = \frac{a + d}{m},$$

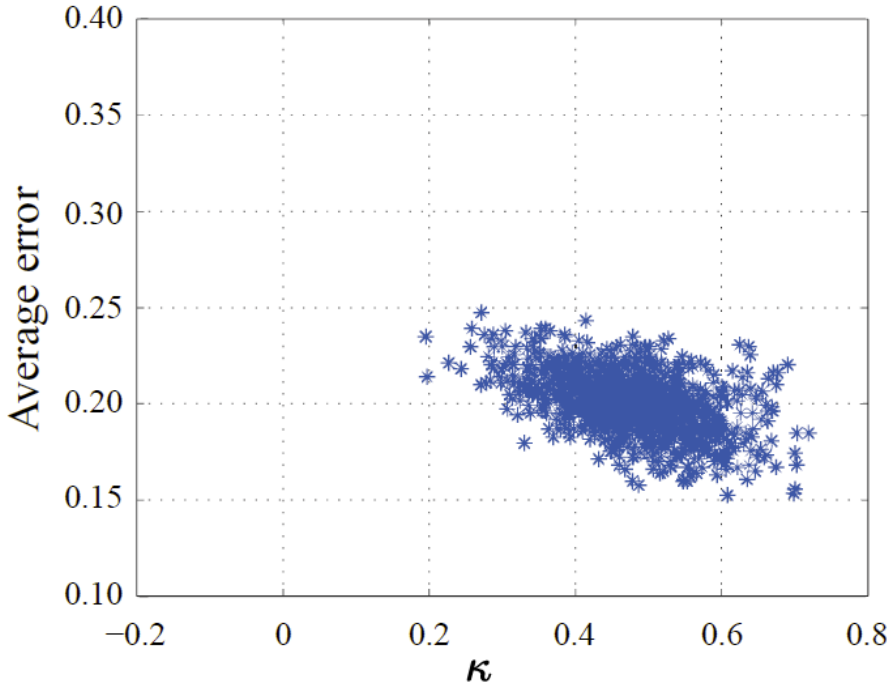
$$p_2 = \frac{(a+b)(a+c) + (c+d)(b+d)}{m^2}$$

Diversity Measures

- κ-error diagrams



(a) AdaBoost ensemble.



(b) Bagging ensemble.

Diversity Generation

- How can we enhance diversity?
  - Data sample manipulation
  - Input feature manipulation
  - Output representation manipulation
  - Algorithm Parameter Manipulation
  - Different diversity generation mechanisms can be used together

Data sample manipulation

- Data sample manipulation is often based on sampling methods
  - Bootstrap sampling used by Bagging
  - Sequential sampling used by AdaBoost

- Base learners that are are sensitive to data sample manipulation (*unstable base learners*)
  - Such as decision trees and neural networks

Data sampling manipulation is particularly effective for unstable base learners

- Base learners that are are insensitive to data sample manipulation (*stable base learners*)
  - Such as linear learners, SVM, naïve Bayes, and k-nearest neighbors

Input feature manipulation

- Random subspace

Algorithm 8.4 Random Subspace.

Input: Training set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of base learners T;
Number of features in subspace d'.

Process:

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\mathcal{F}_t = \mathrm{RS}(D, d')$;
3: $D_t = \mathrm{Map}_{\mathcal{F}_t}(D)$;
4: $h_t = \mathfrak{L}(D_t)$.
5: **end for**

d' randomly selected features

Keeps only the selected features

Output: $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}\left(h_t\left(\mathrm{Map}_{\mathcal{F}_t}(\boldsymbol{x})\right) = y\right)$.