ENSEMBLE METHODS

CS5691- MACHINE LEARNING

Bagging: Bishop, 14.2
Boosting: Bishop, 14.3
Boosting Tree, Gradient Boosting: Hastie, 10.9, 10.10.2, 10.10.3

Random Forest: Hastie, 15.1, 15.2

INTRODUCTION

- Improved performance can be obtained by combining multiple models together in some way, instead of just using a single model in isolation
- For instance, we might train L different models and then make predictions using the average of the predictions made by each model. Such combinations of models are sometimes called **committees**.
- An important variant of the committee method, known as **boosting**, involves training multiple models in sequence in which the error function used to train a particular model depends on the performance of the previous models

BAGGING

- Find a way to introduce variability between the different models within the committee
- One approach is to use **bootstrap** datasets where multiple data sets are created
- Suppose our original data set consists of N data points $X = \{x_1, \ldots, x_N\}$. Create a new data set X_B by drawing M points at random from X, with replacement, so that some points in X may be replicated in X_B, whereas other points in X may be absent from X_B.
- This process can be repeated L times to generate L data sets each of size M

BAGGING

- Consider a regression problem in which we are trying to predict the value of a single continuous variable, and suppose we generate M bootstrap data sets and then use each to train a separate copy $y_m(\mathbf{x})$ of a predictive model where $m = 1, \ldots, M$. The committee prediction is given by

$$y_{\mathrm{COM}}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} y_m(\mathbf{x})$$

- The procedure is known as **bootstrap aggregation** or **bagging**

BAGGING

- Suppose the true regression function that we are trying to predict is given by $h(\mathbf{x})$, so that the output of each of the models can be written as the true value plus an error in the form

$$y_m(\mathbf{x}) = h(\mathbf{x}) + \epsilon_m(\mathbf{x})$$

- Average sum-of-squares error then takes the form

$$\mathbb{E}_{\mathbf{x}}\left[\{y_m(\mathbf{x}) - h(\mathbf{x})\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

where, $\mathbb{E}_{\mathbf{x}}[\cdot]$ denotes a frequentist expectation with respect to the distribution of the input vector $\mathbf{x}$. The average error made by the models acting individually is therefore

$$E_{\mathrm{AV}} = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

BAGGING

- Similarly, the expected error from the committee is given by

$$E_{\mathrm{COM}} = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right]$$

$$= \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\right\}^2\right]$$

- If we assume that the errors have zero mean and are uncorrelated, so that

$$\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\right] = 0$$

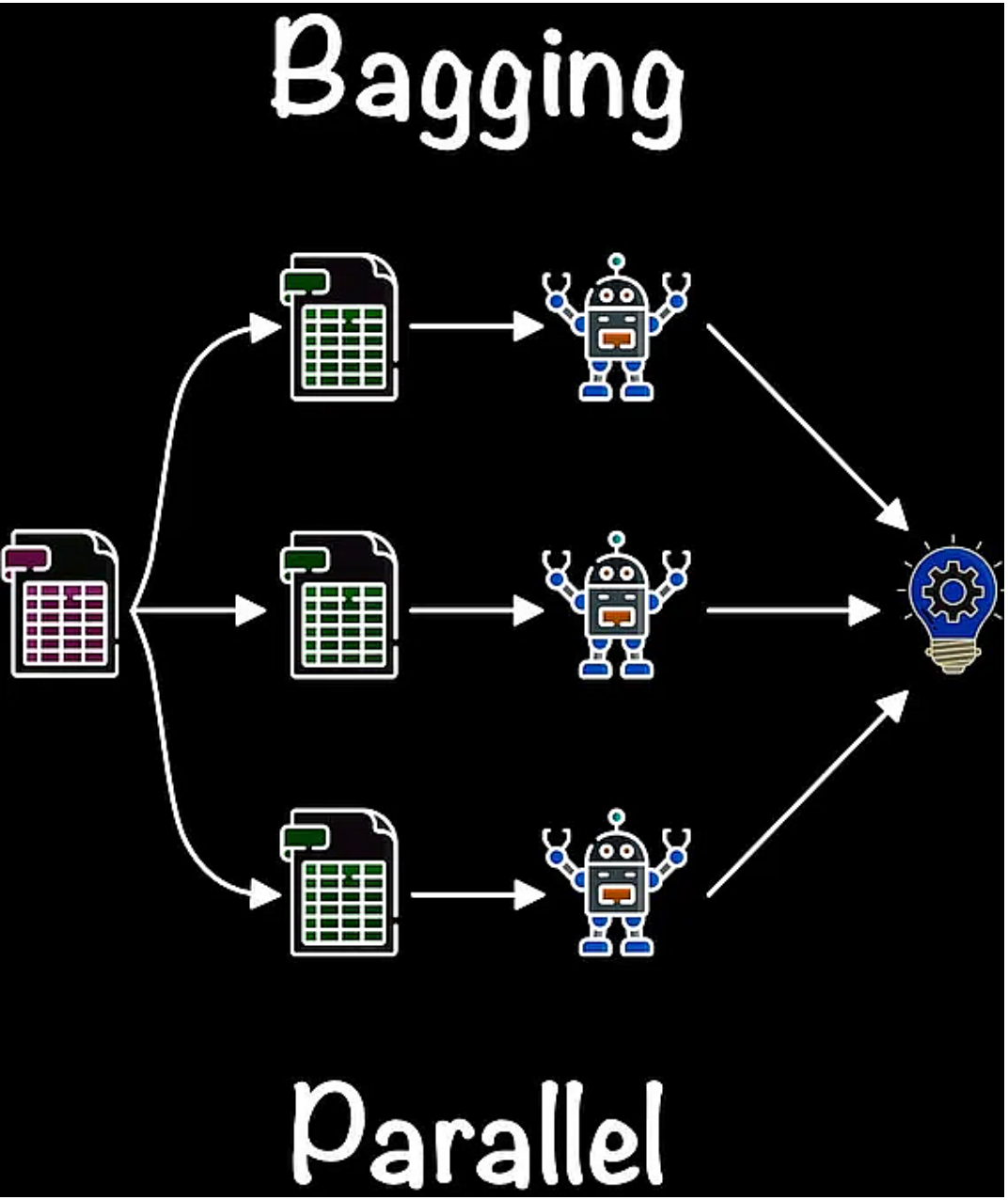
$$\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_l(\mathbf{x})\right] = 0, \qquad m \neq l$$

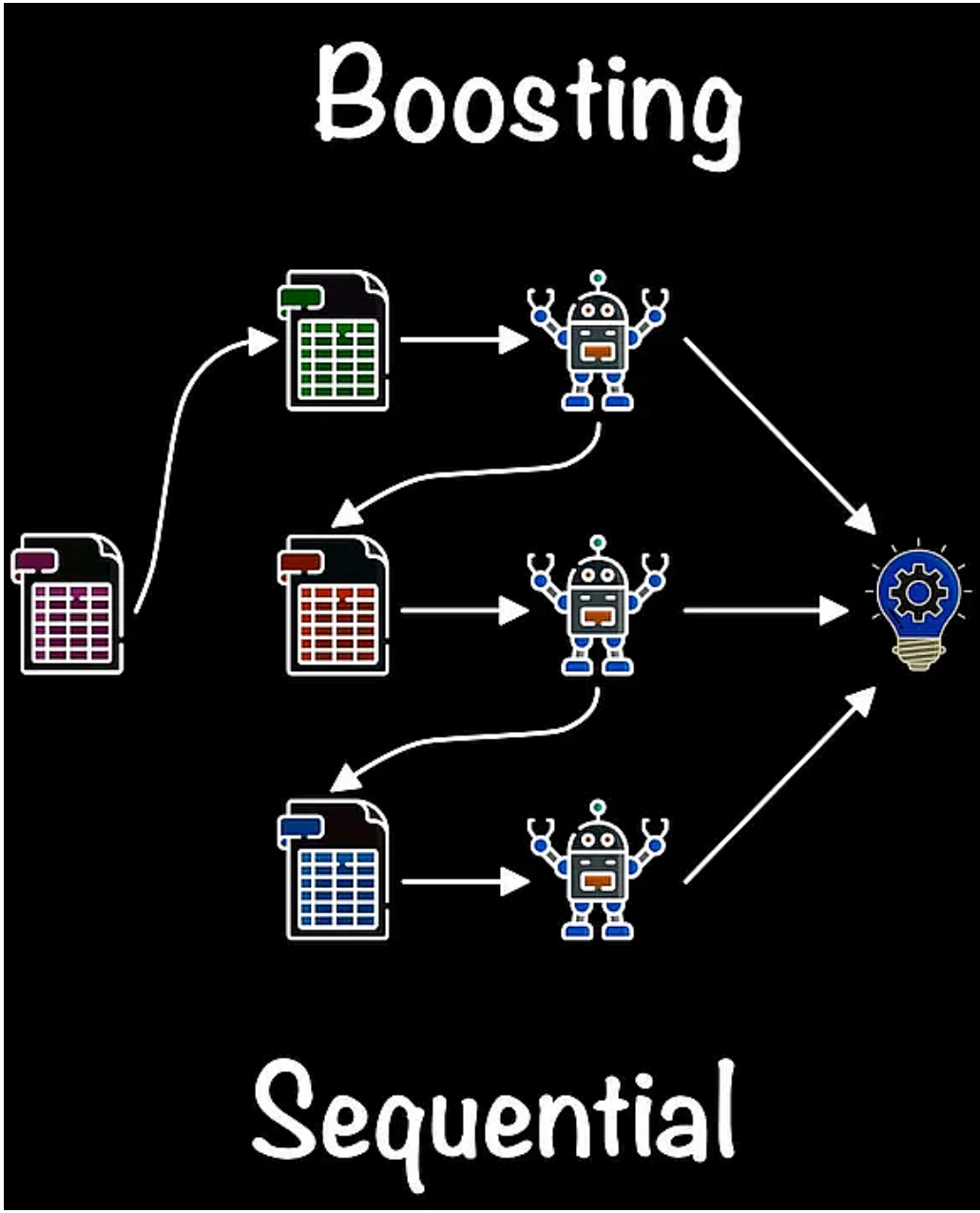
BAGGING

- We obtain

$$E_{\mathrm{COM}} = \frac{1}{M} E_{\mathrm{AV}}$$

- The average error of a model can be reduced by a factor of M simply by averaging M versions of the model
- Unfortunately, it depends on the key assumption that the errors due to the individual models are uncorrelated
- In practice, the errors are typically highly correlated, and the reduction in overall error is generally small
- However, it can be shown that the expected committee error will not exceed the expected error of the constituent models i.e. $E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}$

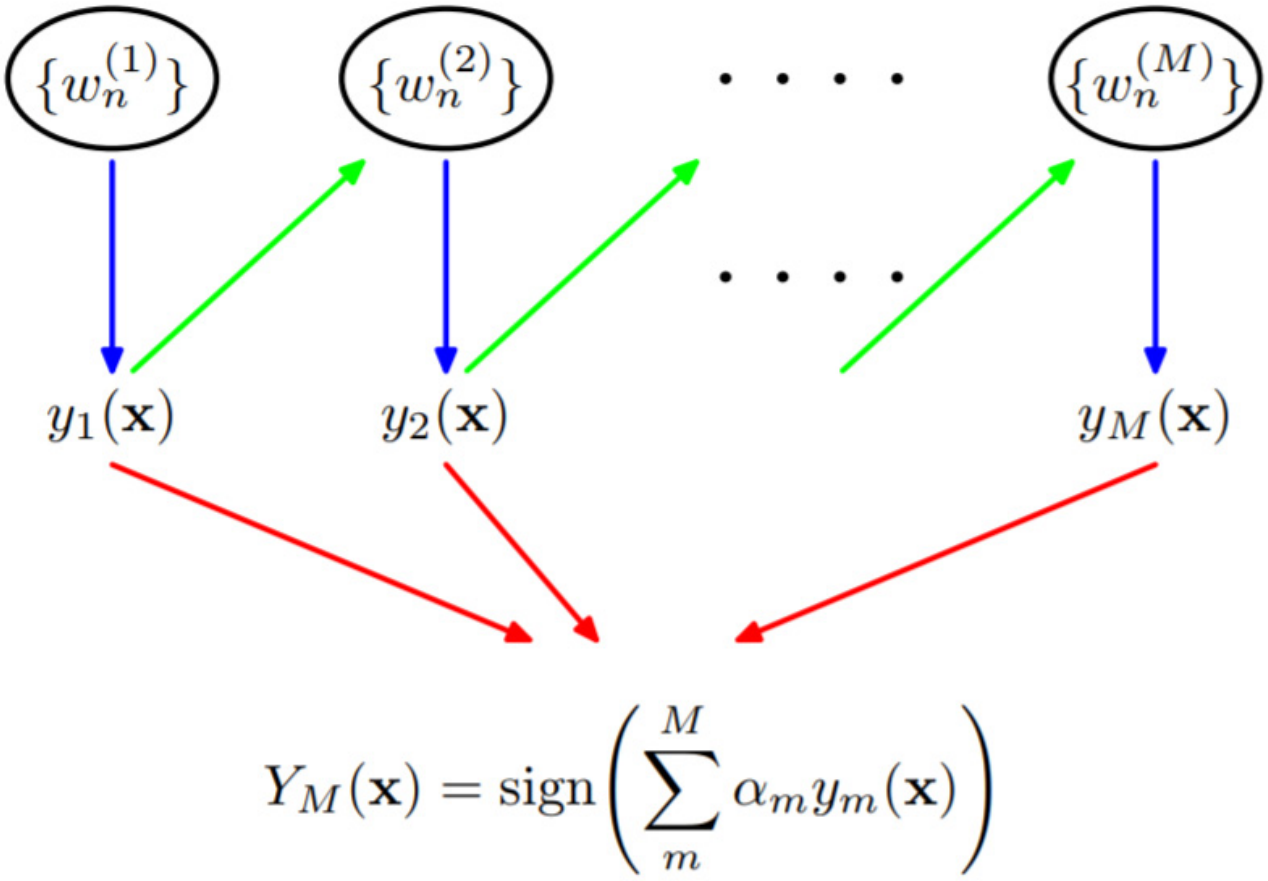

Bagging
Parallel


Boosting
Sequential

BOOSTING

- The principal difference from bagging, is that the base classifiers are here are trained in sequence
- Each base classifier is trained using a weighted form of the data set in which the weighting coefficient associated with each data point depends on the performance of the previous classifiers
- *Points that are misclassified by one of the base classifiers are given greater weight when used to train the next classifier in the sequence*
- Once all the classifiers have been trained, their predictions are then combined through a weighted majority voting scheme
- Here we describe the most widely used form of boosting algorithm called **AdaBoost** for a 2-class classification problem

BOOSTING



Schematic illustration of the boosting framework. Each base classifier $y_m(\mathbf{x})$ is trained on a weighted form of the training set (**blue arrows**) in which the weights $w_n^{(m)}$ depend on the performance of the previous base classifier $y_{m-1}(\mathbf{x})$ (**green arrows**). Once all base classifiers have been trained, they are combined to give the final classifier $Y_M(\mathbf{x})$ (red arrows)

AdaBoost Algorithm

The training data comprises input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ along with corresponding binary target variables $\mathrm{t}_1, \ldots, t_N$ where $t_n \in \{-1, 1\}$

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $\{w_n^{(1)}\} = 1/N$ for $\mathrm{n} = 1, \ldots, \mathrm{N}$
2. For $\mathrm{m} = 1, \ldots, \mathrm{M}$:
   a) Fit a classifier $y_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)$$

where, $I(y_m(\mathbf{x}_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(\mathbf{x}_n) \neq t_n$; and 0 otherwise

AdaBoost Algorithm

b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

and then use these to evaluate

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}$$

c) Update the data weighting coefficients:

$$w_n^{(m+1)} = w_n^{(m)} \exp \{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\}$$

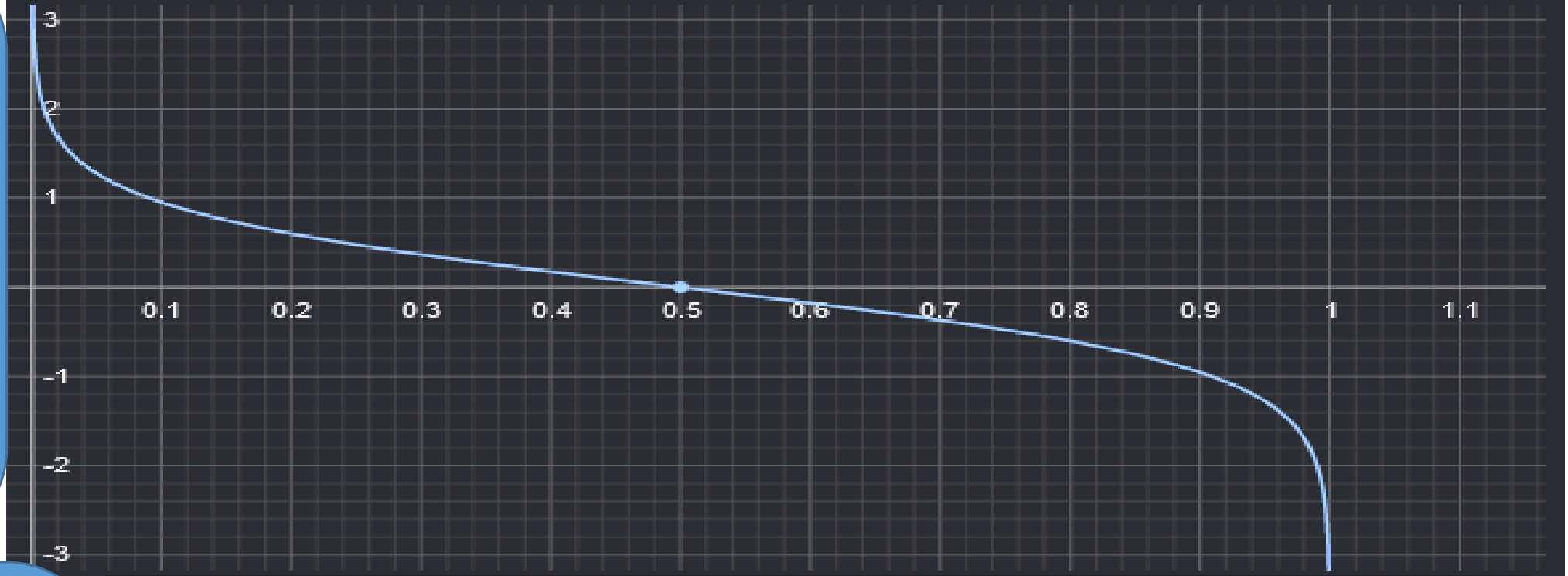
3. Make predictions using the final model, which is given by

$$Y_M(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}) \right)$$

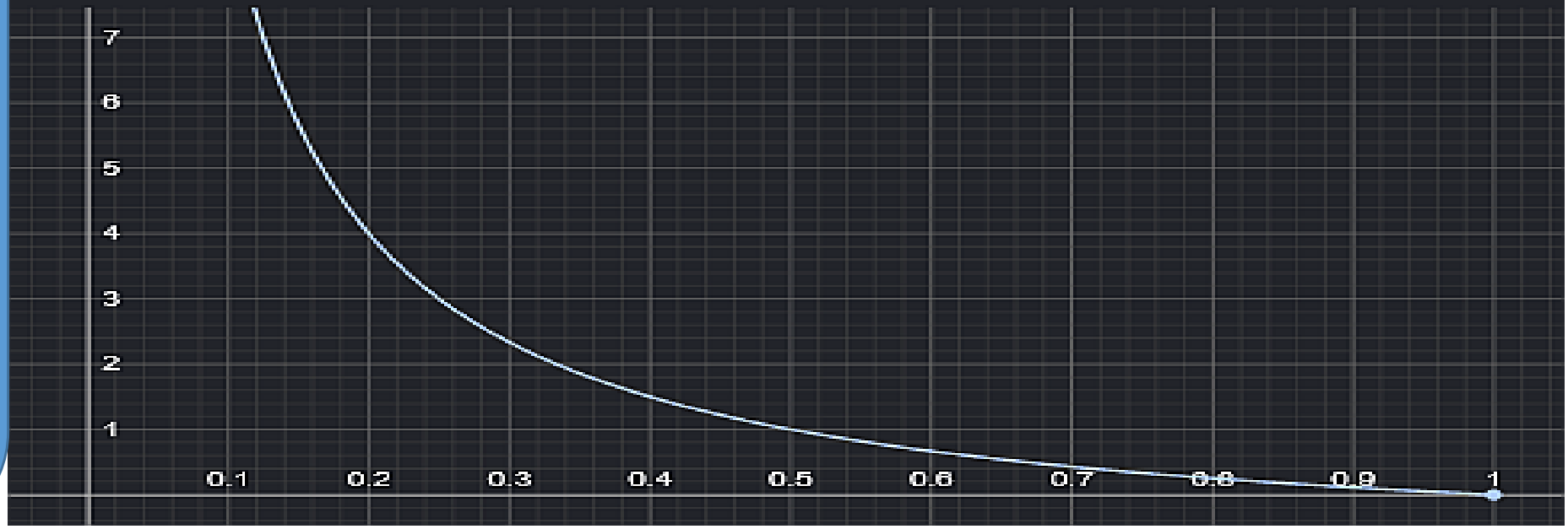
h for $(1 - x)/x$

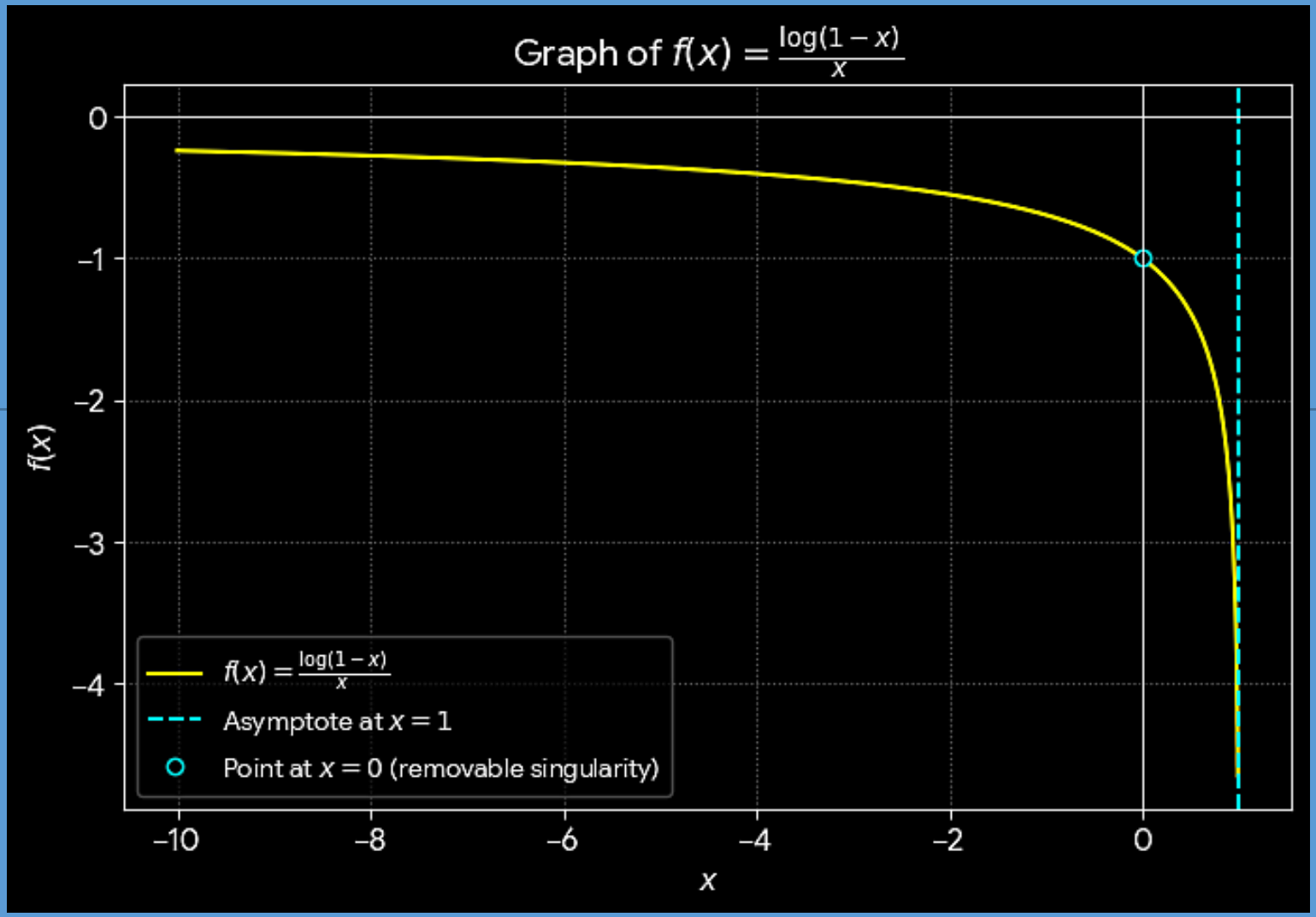
$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$$

for log((1-x)/x)



h for exp(ln((1 − x)/x))

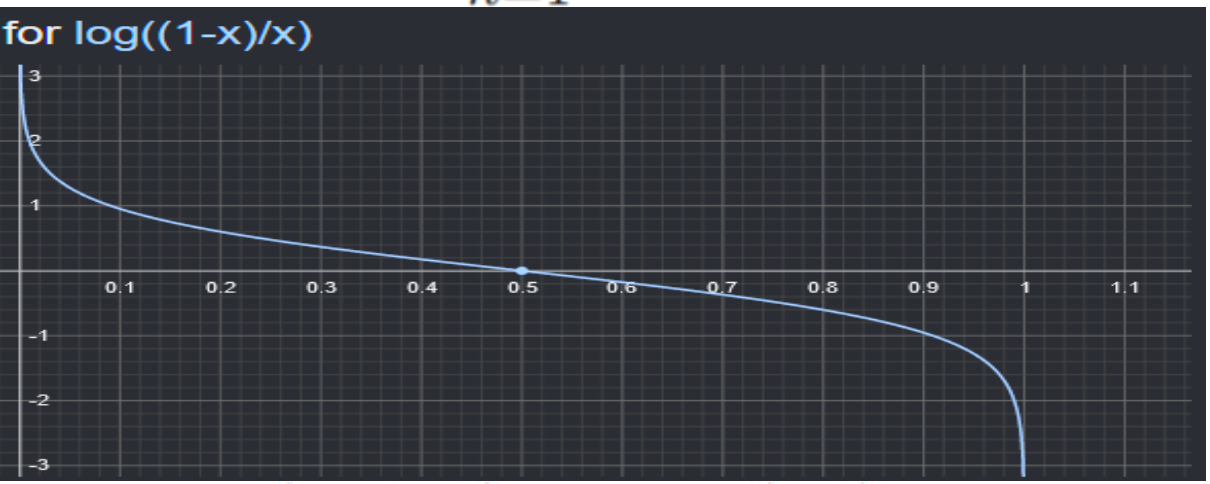



Graph of $f(x) = \frac{\log(1-x)}{x}$
0
−1
−2
−3
−4
f(x)
−10
−8
−6
−4
−2
0
x
$f(x) = \frac{\log(1-x)}{x}$
Asymptote at $x = 1$
Point at $x = 0$ (removable singularity)

Relook at Eqns:

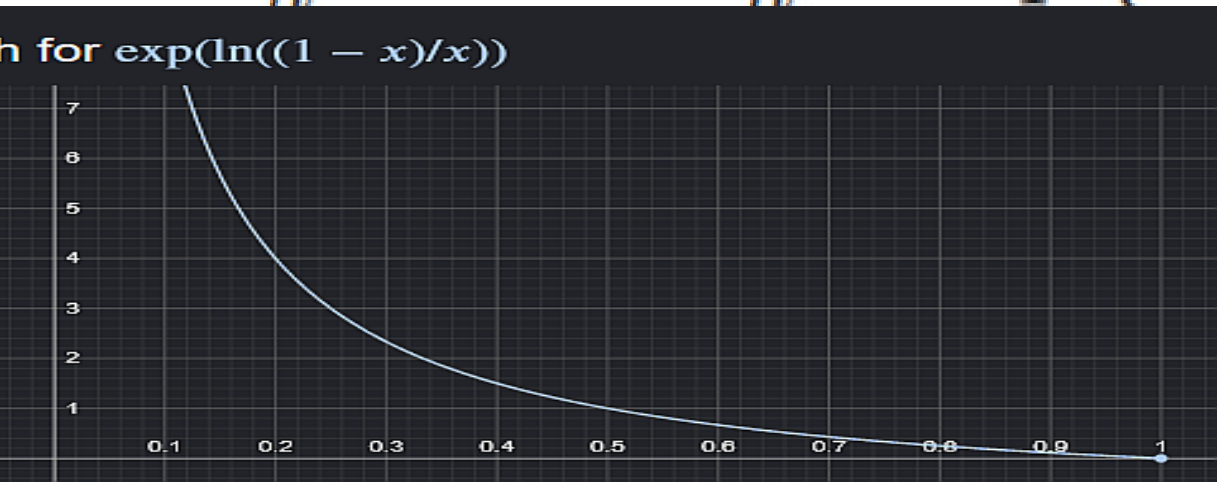
$$\epsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

$$\alpha_m = \ln\left\{\frac{1-\epsilon_m}{\epsilon_m}\right\}$$



So, what is the weight of the <u>Weakest</u> Classifier ?

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\right\}$$



Observe two effects:
Classifier-wise m
Sample-wise n

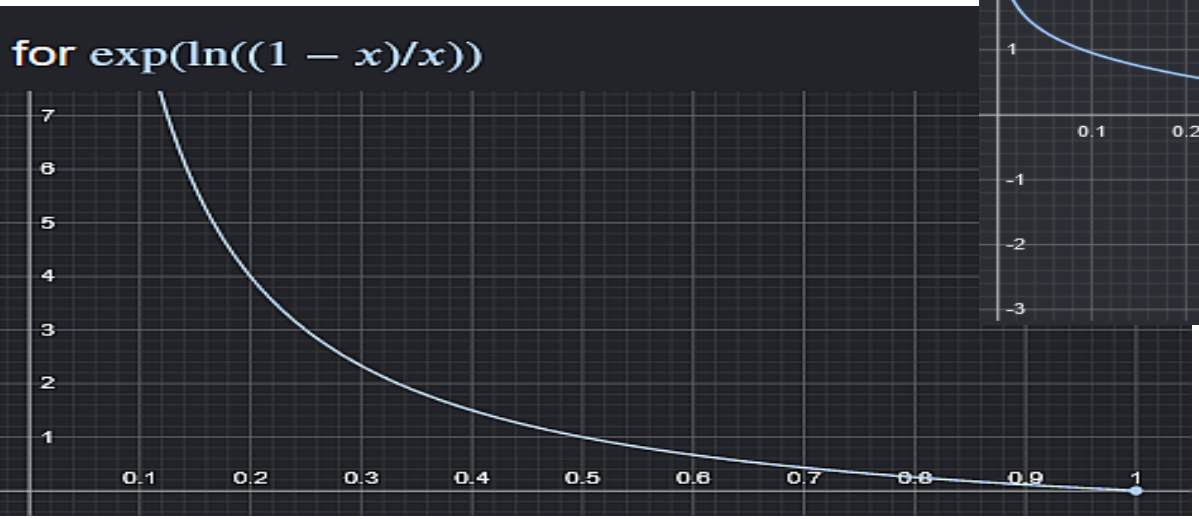
The selected weight for each new weak classifier is always positive.

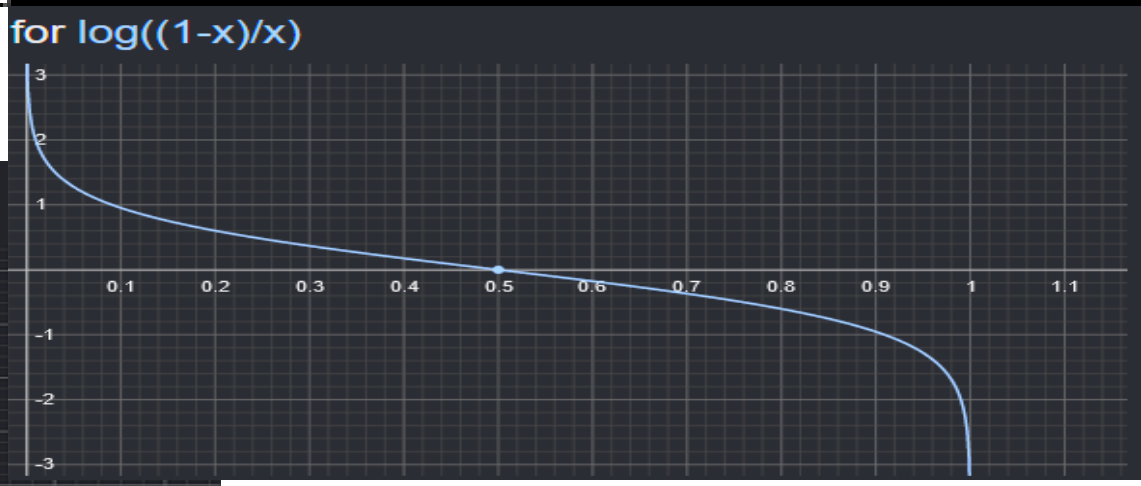
$$\epsilon_t(h_t) < \frac{1}{2} \Rightarrow \alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t(h_t)}{\epsilon_t(h_t)} > 0$$

The weights of the data points are multiplied by $\exp\left[-y_i \alpha_t h_t(\mathbf{x}_i)\right]$.

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

(- High Alpha)

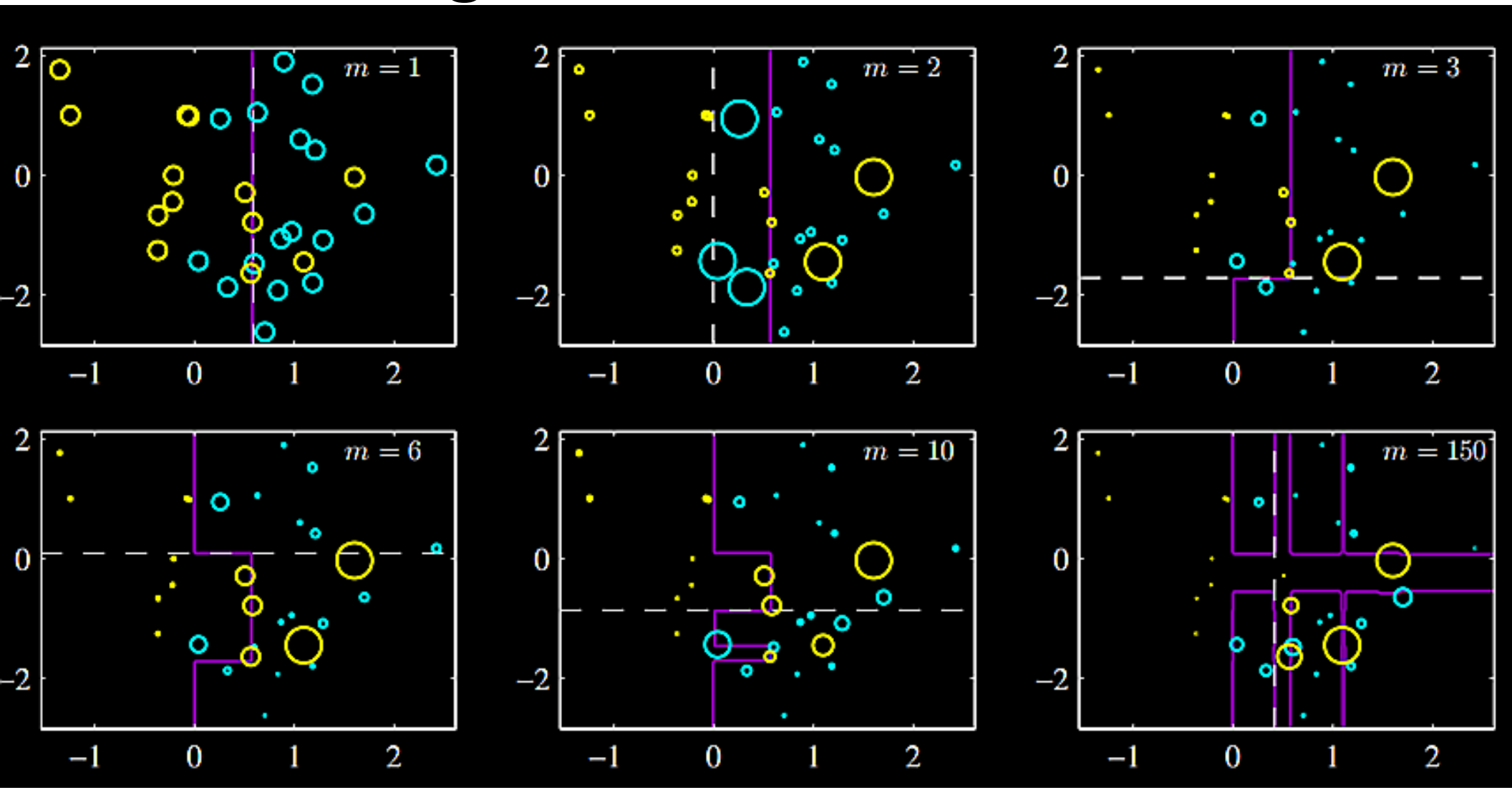




BOOSTING

- In subsequent iterations the weighting coefficients $w_n^{(m)}$ are *increased for data points that are misclassified and decreased for data points that are correctly classified*
- Successive classifiers are thereby forced to place greater emphasis on points that have been misclassified by previous classifiers, and data points that continue to be misclassified by successive classifiers receive ever greater weightage
- The quantities ϵ_m represent weighted measures of the error rates of each of the base classifiers on the data set
- The *weighting coefficients α_m give greater weight to the more accurate classifiers* when computing the overall output

AdaBoost Algorithm



The base learners consist of simple thresholds applied to one or other of the axes. Each figure shows the decision boundary of the most recent base learner (**dashed WHITE line**) and the combined decision boundary of the ensemble (***solid purple/magenta line***). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner.

Minimizing exponential error

- An interpretation of boosting in terms of the sequential minimization of an exponential error function
- Consider the exponential error function defined by

$$E = \sum_{n=1}^{N} \exp\{-t_n f_m(\mathbf{x}_n)\}$$

where $f_m(\mathbf{x})$ is a classifier defined in terms of a linear combination of base classifiers $y_l(\mathbf{x})$ of the form

$$f_m(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(\mathbf{x})$$

- $t_n \in \{-1, 1\}$ are the training set target values
- Goal is to minimize E w.r.t. the weighting coefficients α_l and the parameters of the base classifiers $y_l(\mathbf{x})$

Minimizing exponential error

- Suppose that the base classifiers $y_1(\mathbf{x}), \ldots, y_{m-1}(\mathbf{x})$ and their coefficients $\alpha_1, \ldots, \alpha_{m-1}$, are fixed.
- We are minimizing only with respect to α_m and $y_m(\mathbf{x})$
- The error function can be written as

$$E = \sum_{n=1}^{N} \exp\left\{-t_n f_{m-1}(\mathbf{x}_n) - \frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}$$

$$= \sum_{n=1}^{N} w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}$$

where the coefficients $w_n^{(m)} = \exp\{-t_n f_{m-1}(\mathbf{x}_n)\}$ can be viewed as constants because we are optimizing only α_m and $y_m(\mathbf{x})$

Minimizing exponential error

- Let us denote by $\mathcal{T}_m$ the set of data points that are correctly classified by $y_m(\mathbf{x})$, and the remaining misclassified points by $\mathcal{M}_m$
- The error function can be rewritten as (adding-SUB terms)

$$
\begin{aligned}
E &= e^{-\alpha_m/2} \sum_{n \in \mathcal{T}_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in \mathcal{M}_m} w_n^{(m)} \\
&= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{n=1}^{N} w_n^{(m)} I(y_m(\mathbf{x}_n) \neq t_n) \\
&\quad + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}
\end{aligned}
$$

- When we minimize this with respect to $y_m(\mathbf{x})$, we see that the second term is constant which is equivalent to minimizing the expression given in the algorithm. Similar, is the case where we minimize with respect to α_m

Minimizing exponential error

- Having found α_m and $y_m(\mathbf{x})$, the weights on the data points are updated using

$$w_n^{(m+1)} = w_n^{(m)} \exp\left\{-\frac{1}{2} t_n \alpha_m y_m(\mathbf{x}_n)\right\}$$

- Making use of the fact that

$$t_n y_m(\mathbf{x}_n) = 1 - 2I(y_m(\mathbf{x}_n) \neq t_n)$$

we see that the weights $w_n^{(m)}$ are updated at the next iteration using

$$w_n^{(m+1)} = w_n^{(m)} \exp(-\alpha_m/2) \exp\{\alpha_m I(y_m(\mathbf{x}_n) \neq t_n)\}$$

- Because the term $\exp(-\alpha_m/2)$ is independent of n, we see that it weights all data points by the same factor and can be discarded (See step C of Algo.)

BOOSTING TREES

- Regression trees partition the space of all joint predictor variable values into disjoint regions R_j, *j = 1, 2,... , J*
- A constant γ_j is assigned to each such region and the predictive rule is $x \in R_j \Rightarrow f(x) = \gamma_j$
- Thus a tree can be formally expressed as

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

with parameters $\Theta = \{R_j, \gamma_j\}_1^J$. *J* is usually treated as a meta-parameter

- The parameters are found by minimizing the empirical risk

$$\hat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_j)$$

BOOSTING TREES

- The boosted tree model is a sum of such trees

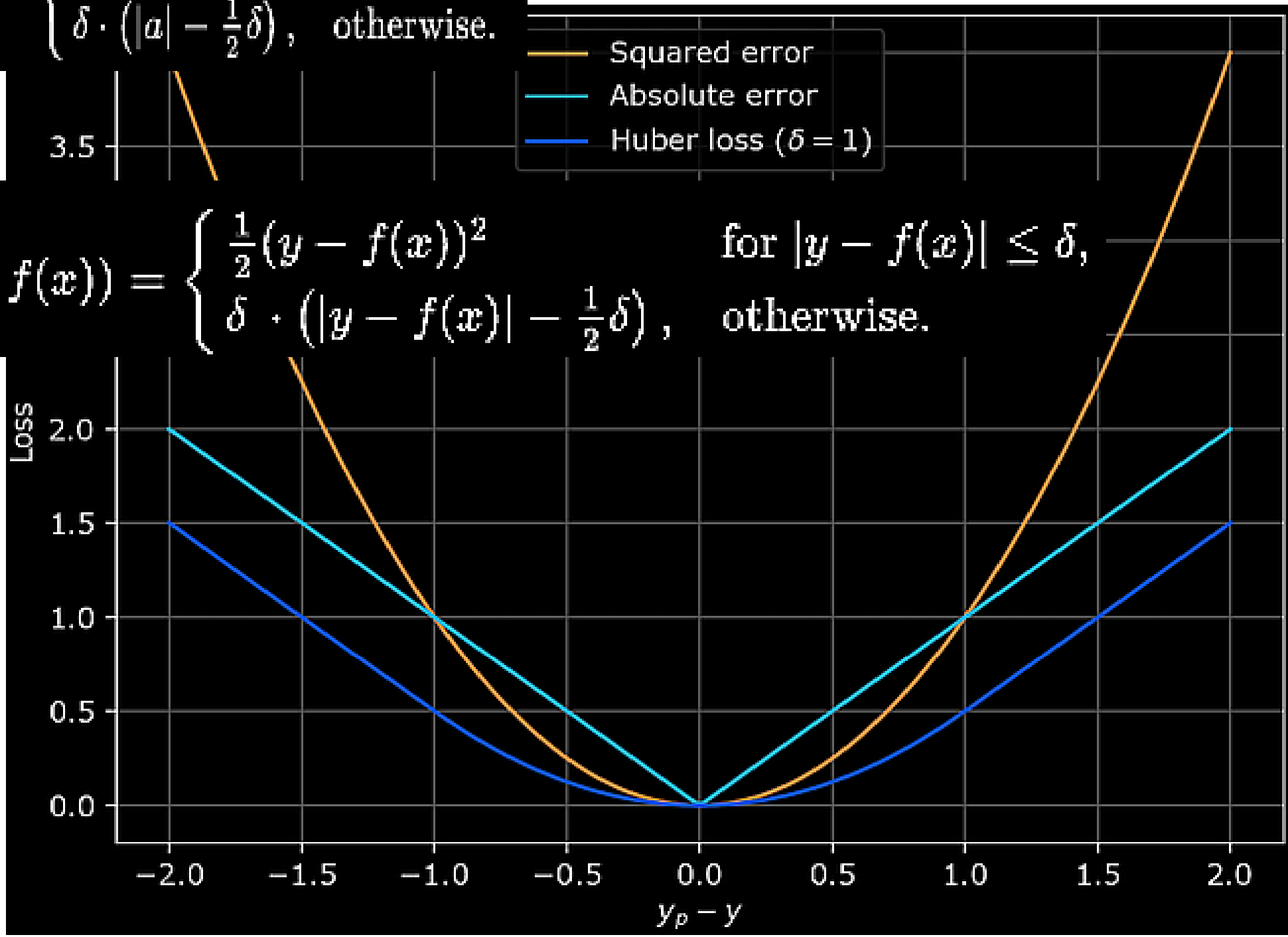
$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

- We look at a generic gradient tree-boosting algorithm for regression. Specific algorithms are obtained by inserting different loss criteria $L(y, f(x))$

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\lvert y_i - f(x_i)\rvert \leq \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $\lvert y_i - f(x_i)\rvert > \delta_m$ where $\delta_m = \alpha\text{th-quantile}\{\lvert y_i - f(x_i)\rvert\}$

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \le \delta, \\ \delta \cdot \left(|y - f(x)| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$



GRADIENT TREE BOOSTING ALGORITHM

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.

2. For $m = 1$ to M:

   (a) For $i = 1, 2, \ldots, N$ compute

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

   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

3. Output $\hat{f}(x) = f_M(x)$.

GRADIENT TREE BOOSTING

- The first line of the algorithm initializes to the optimal constant model, which is just a single terminal node tree
- The components of the negative gradient computed at line 2(a) are referred to as generalized or pseudo residuals, r
- Two basic tuning parameters are the number of iterations M and the sizes of each of the constituent trees J_m, where $m = 1, 2, \ldots, M$

Other Methods of Ensemble

Goals of Combining classifiers (Decision Fusion):

- Improve performance.
- Maximize information use.
- Obtain a reliable system

Challenges:

- Intelligent combination that exploits complementary information.

Multi-Classifier system is obtained using:

- Different classifiers
- Same classifier with different parameters
- Using different feature sets
- Non overlapping training datasets

Decision Fusion method

Decision Profile

- $\mathbf{X} \in R^n$ - feature vector & $\Omega = \{\omega_1, \omega_2, \ldots, \omega_C\}$ - the set of class labels.
- With all classifiers D_i in $D = \{D_1, D_2, \ldots, D_L\}$

one obtains soft class labels as:

$$DP(\mathbf{X}) = \begin{bmatrix} d_{1,1}(\mathbf{X}) \cdots d_{1,j}(\mathbf{X}) \cdots d_{1,C}(\mathbf{X}) \\ d_{i,1}(\mathbf{X}) \cdots d_{i,j}(\mathbf{X}) \cdots d_{i,C}(\mathbf{X}) \\ d_{L,1}(\mathbf{X}) \cdots d_{L,j}(\mathbf{X}) \cdots d_{L,C}(\mathbf{X}) \end{bmatrix}$$

Decision Templates

For j=1, 2,…,C, calculate the mean, $(DT_j)_{L*C}$ of the decision profiles $DP(Z_k)$ of all members of ω_j from the dataset Z.

$$DT_j = \frac{1}{N_j} \sum_{\substack{z_k \in \omega_j \\ z_k \in Z}} DP(Z_k)$$

where N_j is the number of elements of Z from class ω_j

Fusion Rules

- i. MinMax-Avg Rule (MMAV):

$$\Gamma_j(X) = avg\{\min_i\{d_{i,j}(X)\} + \max_i\{d_{i,j}(X)\}\}$$

i=1,2,…,L
j=1,2,…,C

- ii. Mean Deviation about Decision Template (MD-DT):

$$\mu_j = \frac{1}{L}\sum_{i=1}^{L} DT_j(i,j)$$

j=1,2,…,C

$$MD_j = 1 - \min_i\{|\, d_{i,j}(X) - \mu_j \,|\}$$

i=1,2,…,L
j=1,2,…,C

Decision Fusion for 4 classifiers (1000-fold study)

Dataset	GMM	F KNN	Bayes	SVM	Sum Rule	DT (E) / DT (S)	DS-C	Proposed	
								MMAV	MD-DT
Iono	83.54	81.17	80.81	85.38	85.79	85.80	85.78	85.80	85.80
Lymph	54.80	75.34	74.87	80.60	77.15	77.15	77.57	83.15	78.94
spectf	53.22	66.86	74.94	72.06	72.43	72.43	72.55	72.47	72.47
Non-linear	49.88	100	86.61	86.52	94.03	94.03	94.03	100	94.03
tictac	57.45	70.53	57.35	65.24	68.14	68.14	68.78	68.14	68.14

GMM- Gaussian Mixture Model

F KNN- Fuzzy K Nearest Neighbor

SVM- Support Vector Machines

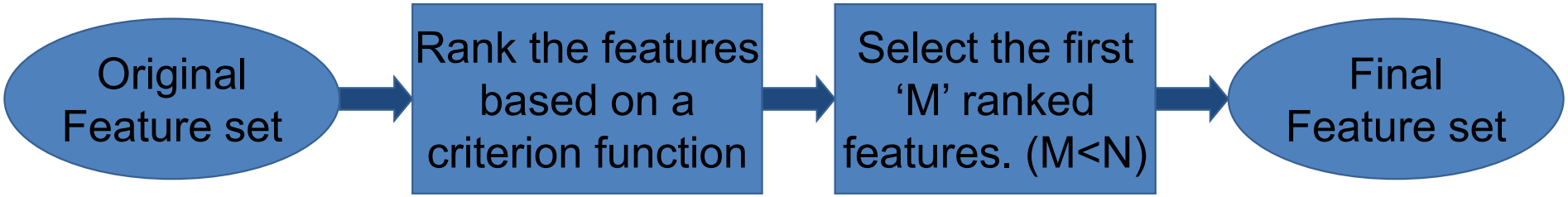
DT (E) / DT (S)- Decision Template with Euclidean Measure / Symmetric difference;
DS - Dempster –Shafer Combination.

MMAV- MinMax-Avg Rule

MD-DT- Mean Deviation about DT

C. L. Blake and C. J. Merz, “UCI repository of machine learning database”, http://archive.ics.uci.edu/ml/.

FEATURE RANKING & SELECTION



The first feature is selected as: $x_1^0 = \max_i I(x_i, y)$

Mutual Information is given by: $I(X,Y) = H(X) - H(X/Y)$

The second feature is selected as:

$$x_2^0 = \max_{i \neq 1}[I(x_i, y) - \sum_{i \neq 1} I(x_i, x_1^0) + \sum_{i \neq 1} I(x_i, x_1^0 \mid y)]$$

The rest of the features are evaluated as:

$$x_n^0 = \max_{i \neq j}[I(x_i, y) - \sum_j \sum_{i \neq j} I(x_i, y)]$$

Decision Fusion (without feature selection)

Dataset	F KNN	SVM-L	SVM-P	Sum Rule	DS	Proposed	
						MMAV	MD-DT
Iono	81.30	84.97	86.72	86.95	86.93	**87.07**	**87.05**
Heart	63.35	82.75	73.32	80.55	**80.57**	**80.55**	**80.55**
Lymph	77.29	81.05	77.76	81.68	81.91	**82.10**	**81.68**

Proposed Framework (Feature Selection+ Decision Fusion)

Feature Selection alone

Dataset	**F KNN**	**SVM-L**	**SVM-P**	**Sum Rule**	**DS**	**Proposed**	
						MMAV	**MD-DT**
Iono	**82.61**	**85.04**	**90.65**	87.80	87.78	**88.04**	**88.04**
Heart	**63.35**	**83**	**75.39**	81.01	81.23	**81.01**	**81.01**
Lymph	**78.94**	82.34	**77.66**	83.49	83.51	**83.95**	**83.49**

Other ML methods:

- **Random Forests**
- **Bayesian Networks, Bayesian Regression**
- **MRF, CRF, Undirected graphical models**
- **PGM, Factor Graphs**
- **Manifold learning**
- **ART, MDS**
-

Kuncheva, L. and Whitaker, C., Measures of diversity in classifier ensembles, *Machine Learning*, 51, pp. 181-207, 2003.

Ensemble Methods in Machine Learning; Thomas G. Dietterich; https://web.engr.oregonstate.edu/~tgd/publications/mcs-ensembles.pdf

Zhou Zhihua (2012). Ensemble Methods: Foundations and Algorithms. Chapman and Hall/CRC. ISBN 978-1-439-83003-1.

Robi Polikar (ed.). "Ensemble learning". Scholarpedia.

L. I. Kuncheva, Combining Pattern Classifiers, Methods and Algorithms. New York, NY: Wiley Interscience, 2005.

J. Kittler, M. Hatef, R. P. W. Duin, and J. Mates, "On combining classifiers," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 20, no. 3, pp. 226-239, 1998.

L. I. Kuncheva, "A theoretical study on six classifier fusion strategies," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 2, pp. 281-286, 2002.