Einführung in Web- und Data-Science

Ensemble Learning

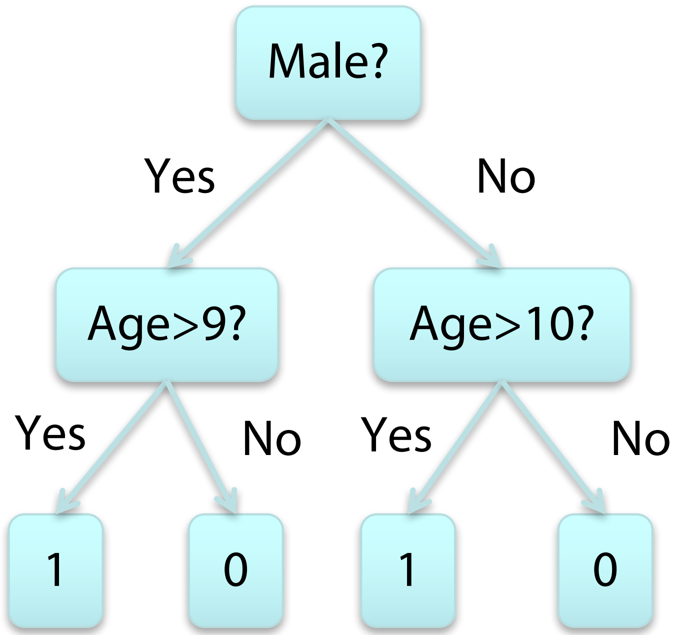
Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Tanya Braun (Übungen)

Decision Trees



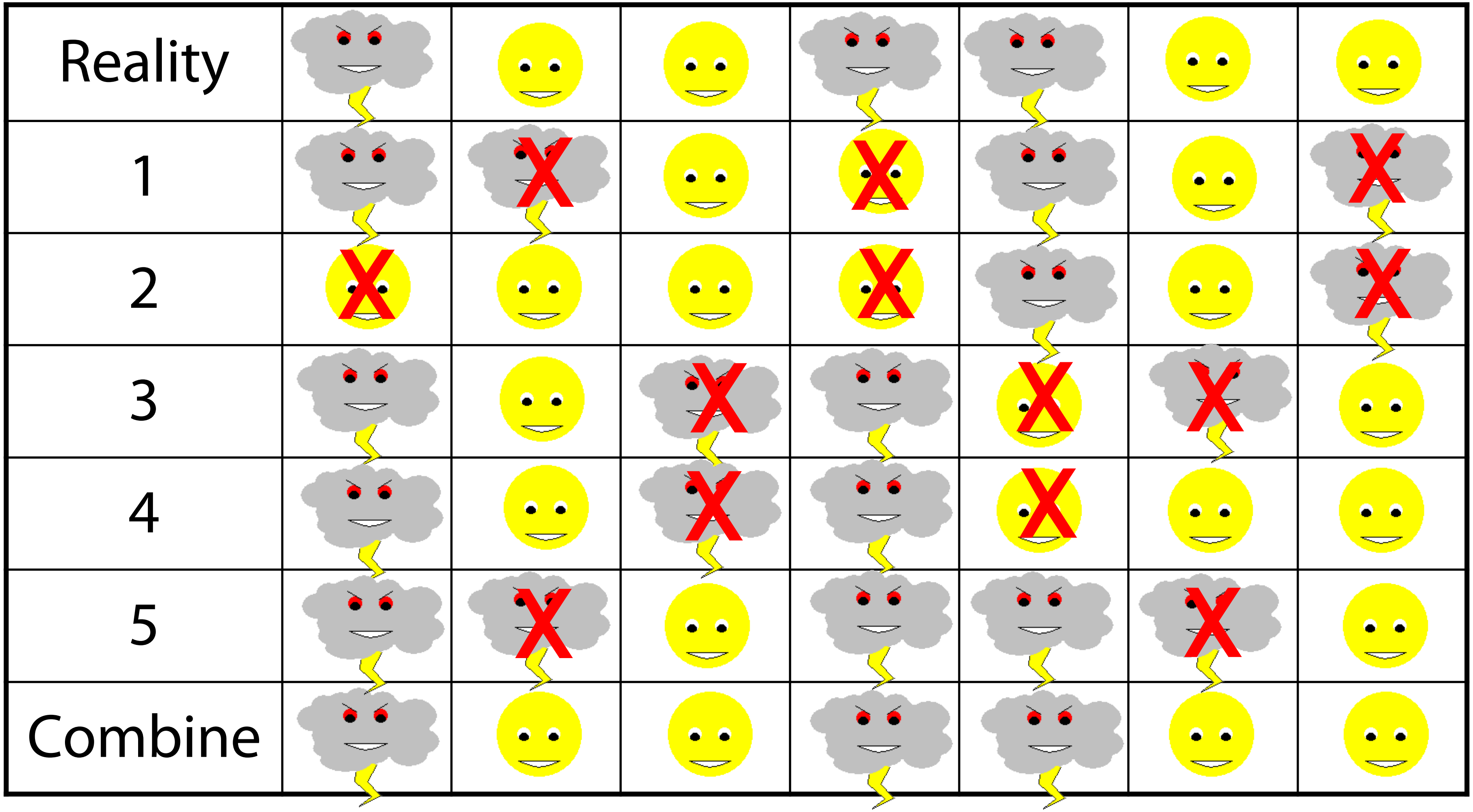
Person	Age	Male?	Height > 55″
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	8	0	0

$$x = \begin{bmatrix} age \\ 1_{[gender=male]} \end{bmatrix} \qquad y = \begin{cases} 1 & height > 55'' \\ 0 & height \leq 55'' \end{cases}$$

Ensembles of Classifiers

- None of the classifiers is perfect
- Idea
  - Combine the classifiers to improve performance
- Ensembles of classifiers
  - Combine the classification results from different classifiers to produce the final output
    - Unweighted voting
    - Weighted voting

Example: Weather Forecast



Voting

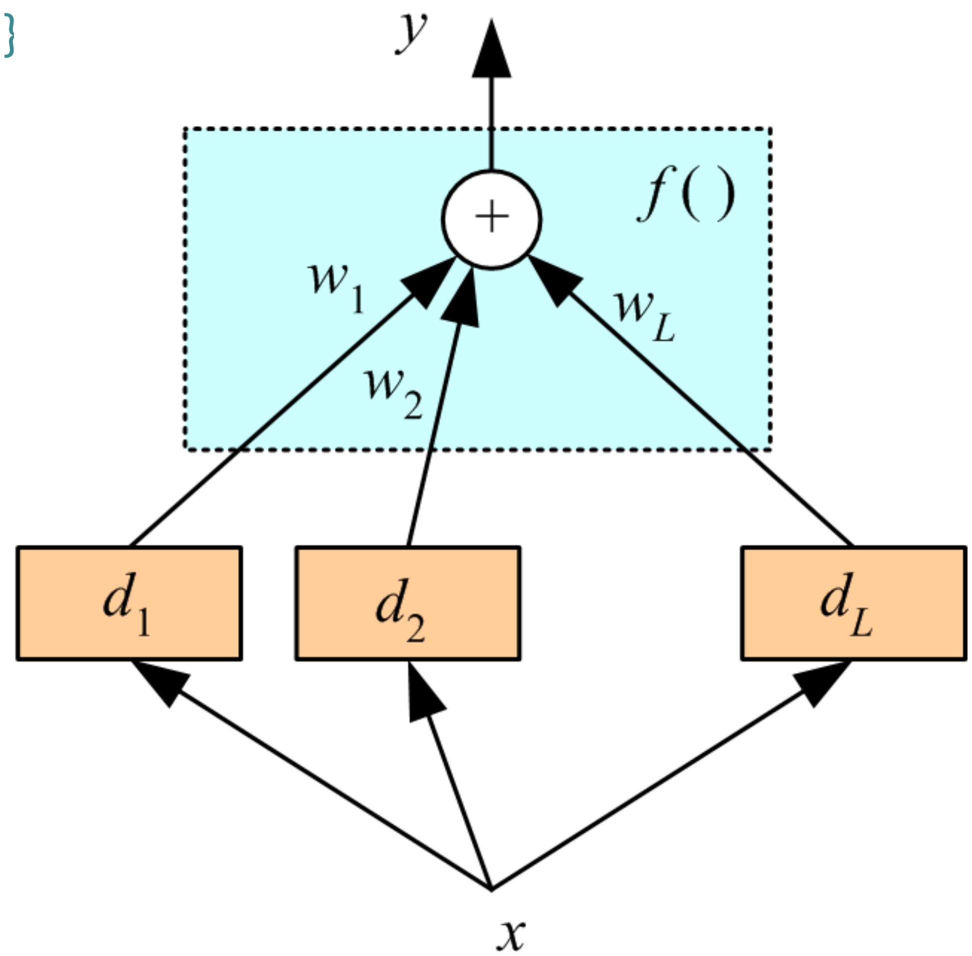
- Linear combination of $d_j \in \{-1, 1\}$

$$y = \sum_{j=1}^{L} w_j d_j$$

$$w_j \geq 0 \text{ and } \sum_{j=1}^{L} w_j = 1$$

- Unweighted voting: $w_j = 1/L$

- Also possible $d_j \in \mathbb{Z}$
- High values for $|y|$ means high "confidence"
- Possible use $\text{sign}(y) \in \{-1, 1\}$



Why does it work?

- Suppose there are 25 independent base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Majority vote with wrong decision: i >12
  - Probability that the ensemble classifier makes a wrong prediction (choose i from 25 (combination w/o repetition):

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

- But: How to ensure that the classifiers are independent?

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \qquad (k \leq n)$$

Outline

- **Bias/Variance Tradeoff**

- Ensemble methods that minimize variance
  - Bagging [Breiman 94]
  - Random Forests [Breiman 97]

- Ensemble methods that minimize bias
  - Functional Gradient Descent
  - Boosting [Freund&Schapire 95, Friedman 98]
  - Ensemble Selection

Subsequent slides are based on a presentation by Yisong Yue
An Introduction to Ensemble Methods
Bagging, Boosting, Random Forests, and More

Generalization Error

- **"True" distribution:** P(x,y)
  - Unknown to us
- **Train:** $h(x) = y$
  - Using training data $S = \{(x_1,y_1),\ldots,(x_n,y_n)\}$
  - Sampled from P(x,y)
- **Generalization Error:**
  - $\mathcal{L}(h) = E_{(x,y)\sim P(x,y)}[\ f(h(x),y)\ ]$
  - E.g., $f(a,b) = (a-b)^2$

Person	Age	Male?	Height > 55"
James	11	1	1
Jessica	14	0	1
Alice	14	0	1
Amy	12	0	1
Bob	10	1	1
Xavier	9	1	0
Cathy	9	0	1
Carol	13	0	1
Eugene	13	1	0
Rafael	12	1	1
Dave	8	1	0
Peter	9	1	0
Henry	13	1	0
Erin	11	0	0
Rose	7	0	0
Iain	8	1	1
Paulo	12	1	0
Margaret	10	0	1
Frank	9	1	1
Jill	13	0	0
Leon	10	1	0
Sarah	12	0	0
Gena	8	0	0
Patrick	5	⋮	1

Person	Age	Male?	Height > 55"	h(x)
Alice	14	0	1	✓
Bob	10	1	1	✓
Carol	13	0	1	✓
Dave	8	1	0	✓
Erin	11	0	0	✗
Frank	9	1	1	✗
Gena	8	0	0	✓

y h(x)

Generalization Error:

$$\mathcal{L}(h) = E_{(x,y)\sim P(x,y)}[\ f(h(x),y)\ ]$$

Bias/Variance Tradeoff

- Treat h(x|S) has a random function
  - Depends on training data S

- $\mathcal{L} = E_S[\ E_{(x,y)\sim P(x,y)}[\ f(h(x|S),y)\ ]\ ]$
  - Expected generalization error
  - Over the randomness of S

Bias/Variance Tradeoff

- Squared loss: $f(a,b) = (a-b)^2$
- Consider one data point (x,y)
- Notation:
  - $Z = h(x|S) - y$
  - $\check{z} = E_S[Z]$
  - $Z-\check{z} = h(x|S) - E_S[h(x|S)]$

$$
\boxed{\begin{aligned} E_S[(Z-\check{z})^2] &= E_S[Z^2 - 2Z\check{z} + \check{z}^2] \\ &= E_S[Z^2] - 2E_S[Z]\check{z} + \check{z}^2 \\ &= E_S[Z^2] - \check{z}^2 \end{aligned}}
$$

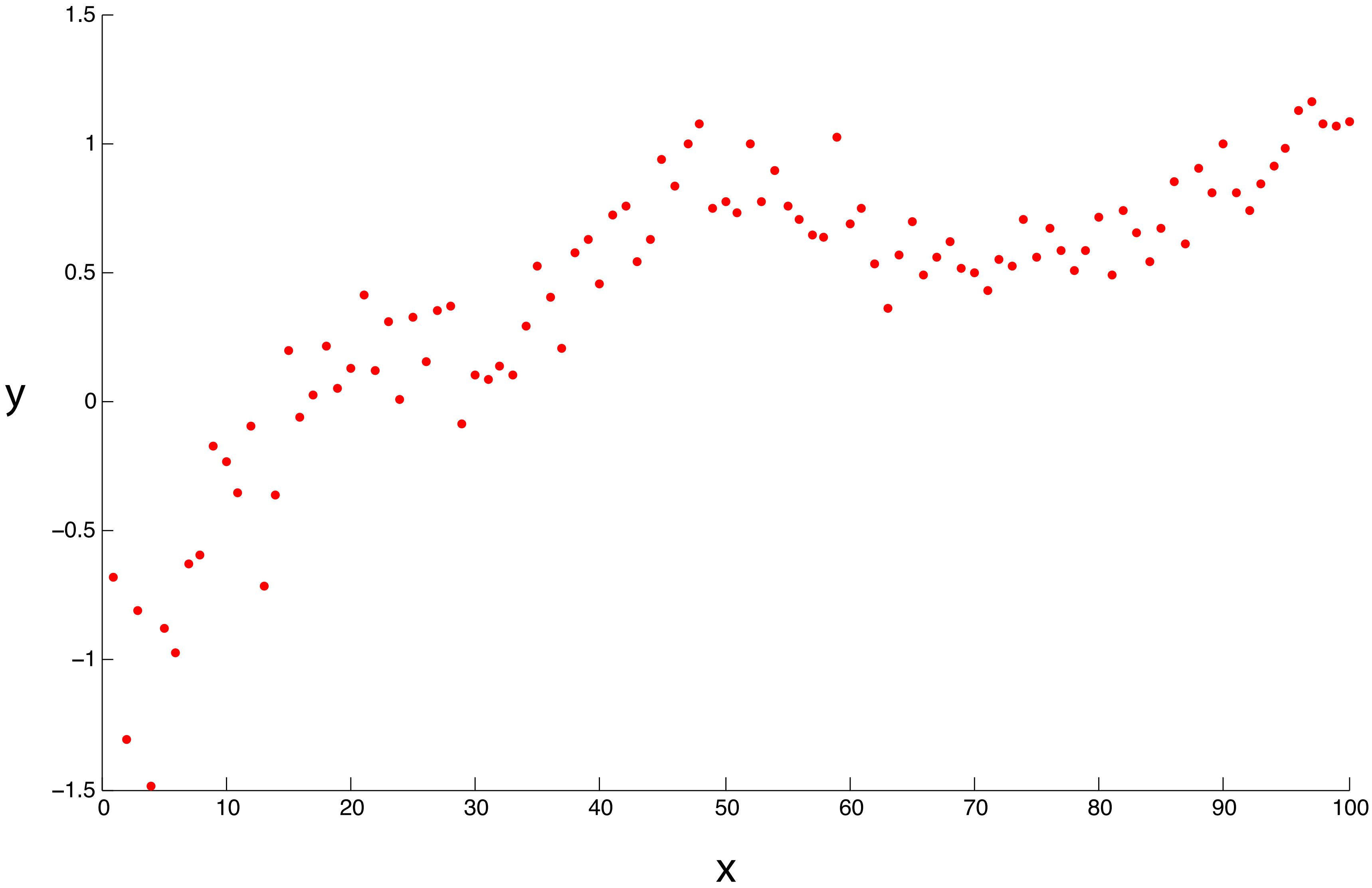
Expected Error

$$
\boxed{\begin{aligned} E_S[f(h(x|S),y)] &= E_S[Z^2] \\ &= E_S[(Z-\check{z})^2] + \check{z}^2 \end{aligned}}
$$

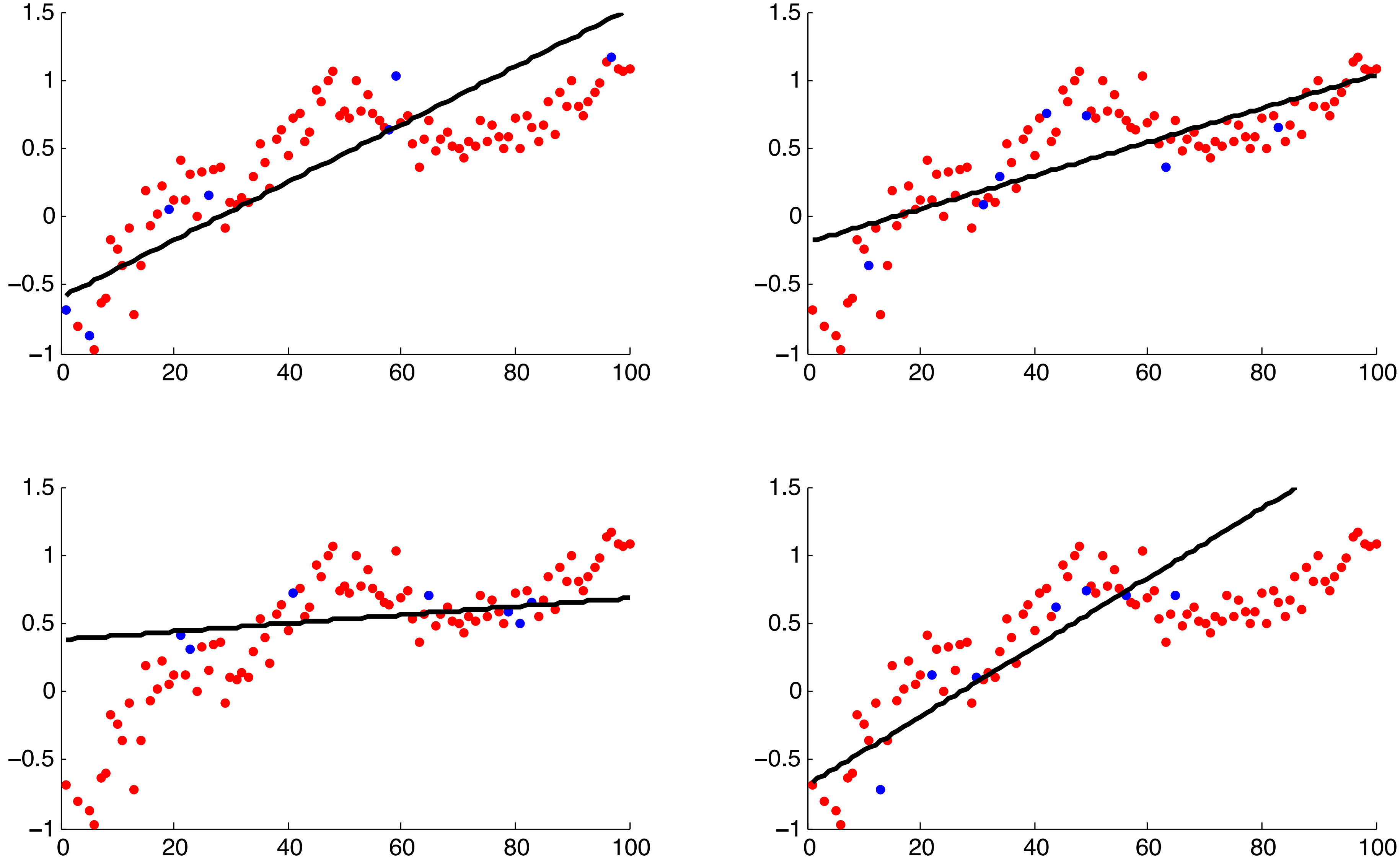
Variance **Bias**

Bias = systematic error resulting from the effect that the expected value of estimation results differs from the true underlying quantitative parameter being estimated.

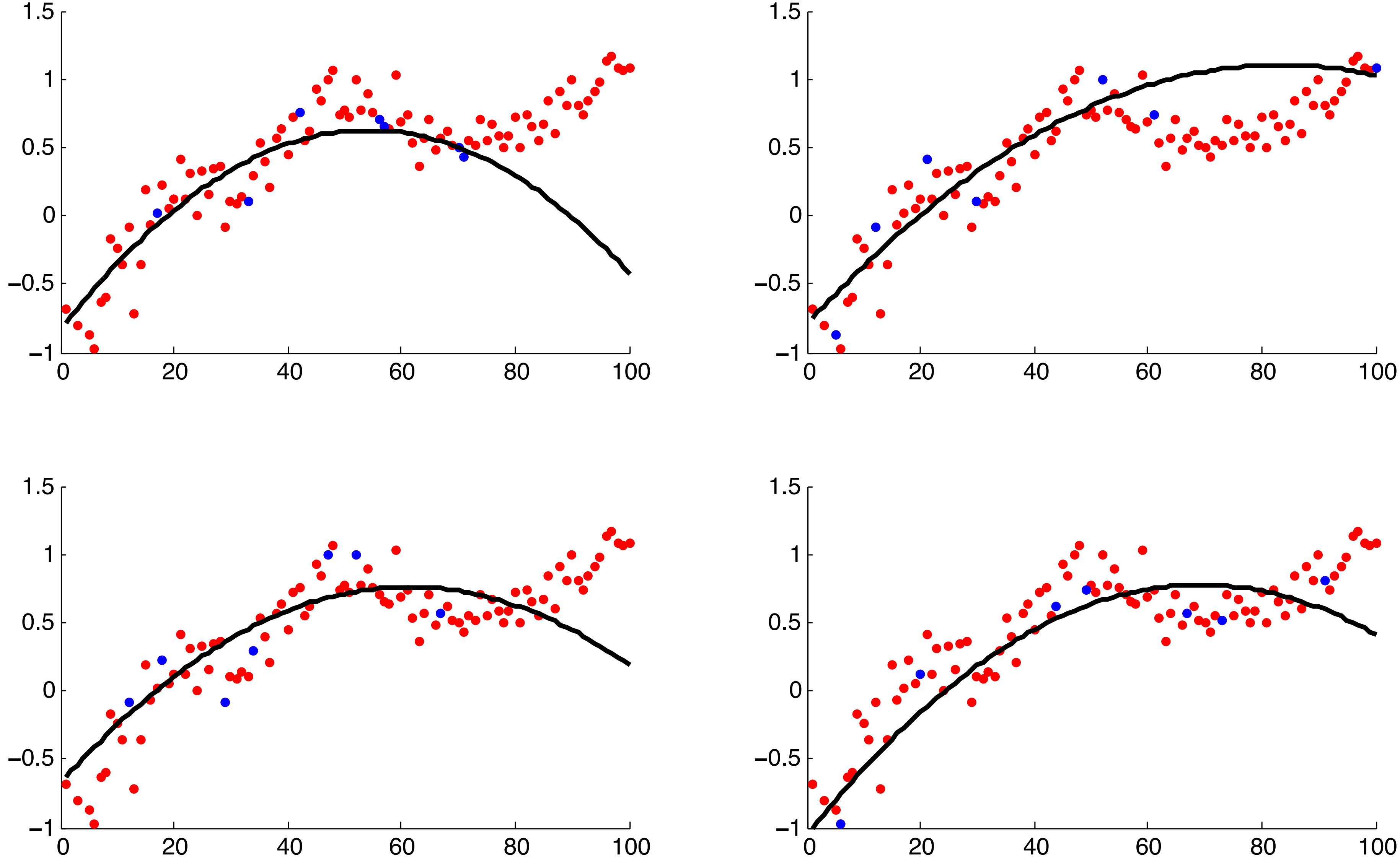
Example



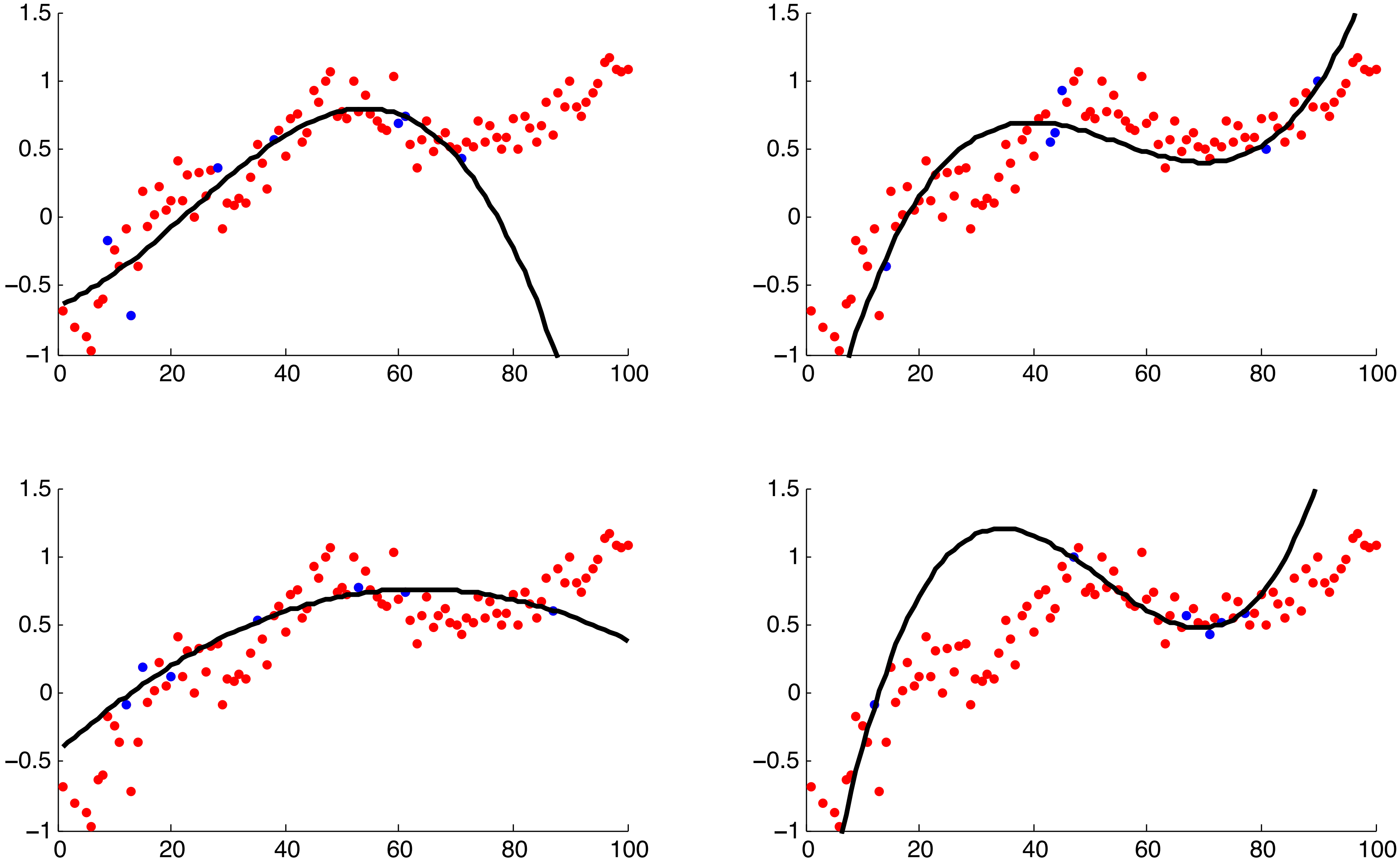
h(x|S)

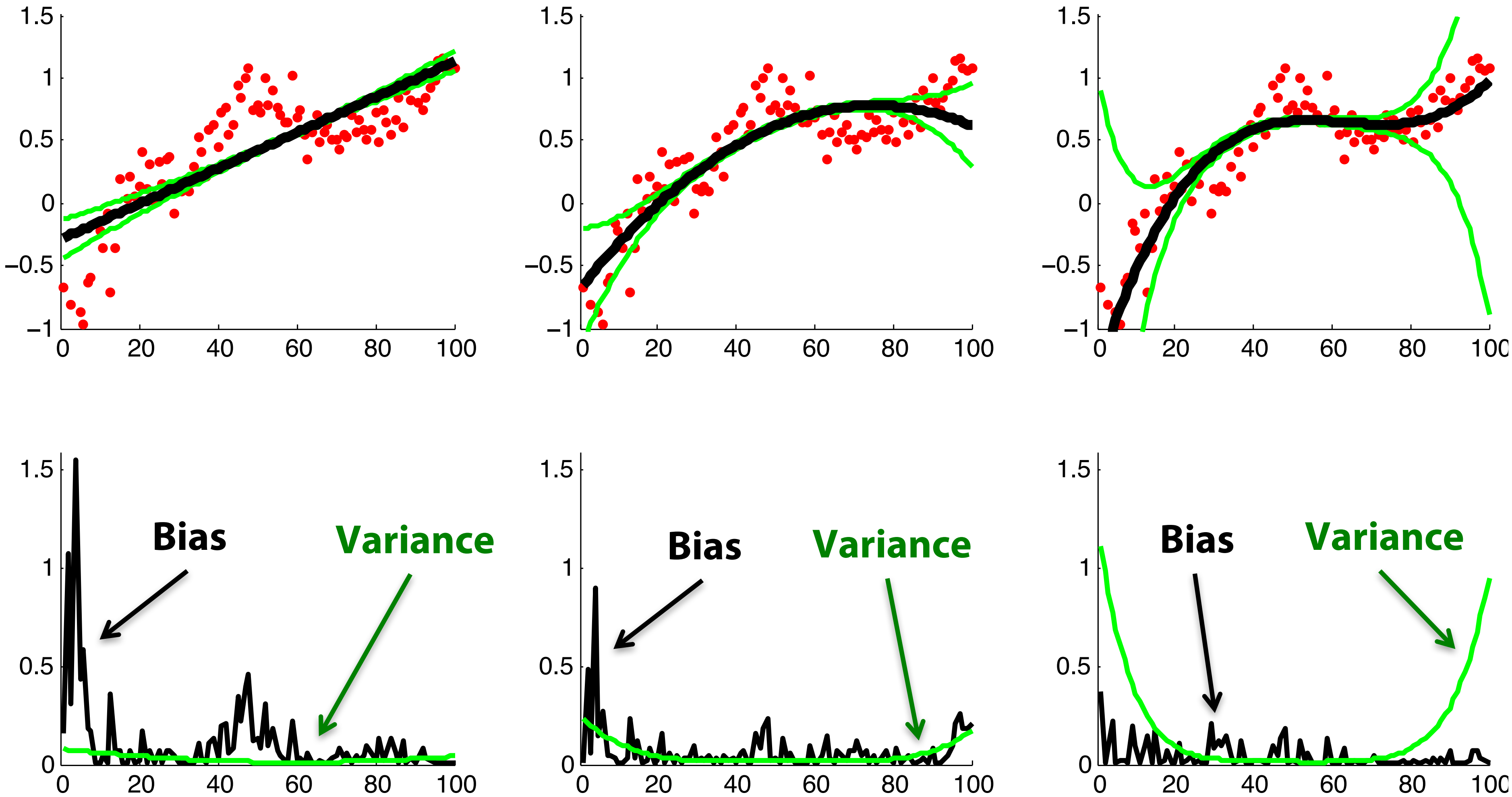


h(x|S)



h(x|S)





$$\boxed{\underbrace{E_S[(h(x|S) - y)^2]}_{\text{Expected Error}} = \underbrace{E_S[(Z-\check{z})^2]}_{\text{Variance}} + \underbrace{\check{z}^2}_{\text{Bias}}}$$

$$Z = h(x|S) - y$$

$$\check{z} = E_S[Z]$$

Outline

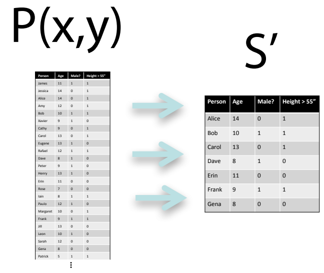
- Bias/Variance Tradeoff

- **Ensemble methods that minimize variance**
  - **Bagging**
  - **Random Forests**

- Ensemble methods that minimize bias
  - Functional Gradient Descent
  - Boosting
  - Ensemble Selection

Subsequent slides by Yisong Yue
An Introduction to Ensemble MethodsBoosting,
Bagging, Random Forests and More

Bagging



- **Goal:** reduce variance
- **Ideal setting:** many training sets S′
  - Train model using each S′
  - Average predictions

sampled independently

Variance reduces linearly
Bias unchanged

$$\boxed{\underbrace{E_S[(h(x|S) - y)^2]}_{\text{Expected Error}} = \underbrace{E_S[(Z-\check{z})^2]}_{\text{Variance}} + \underbrace{\check{z}^2}_{\text{Bias}}}$$

$$Z = h(x|S) - y$$
$$\check{z} = E_S[Z]$$

"Bagging Predictors" [Leo Breiman, 1994]

Bagging = Bootstrap Aggregation

http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

Bagging

- **Goal:** reduce variance

- **In practice:** resample S′ with replacement
  - Train model using each S′
  - Average predictions

from S

Variance reduces sub-linearly
(Because S′ are correlated)
Bias often increases slightly

$$\boxed{\underbrace{E_S[(h(x|S) - y)^2]}_{\text{Expected Error}} = \underbrace{E_S[(Z-\check{z})^2]}_{\text{Variance}} + \underbrace{\check{z}^2}_{\text{Bias}}}$$

Expected Error **Variance** **Bias**

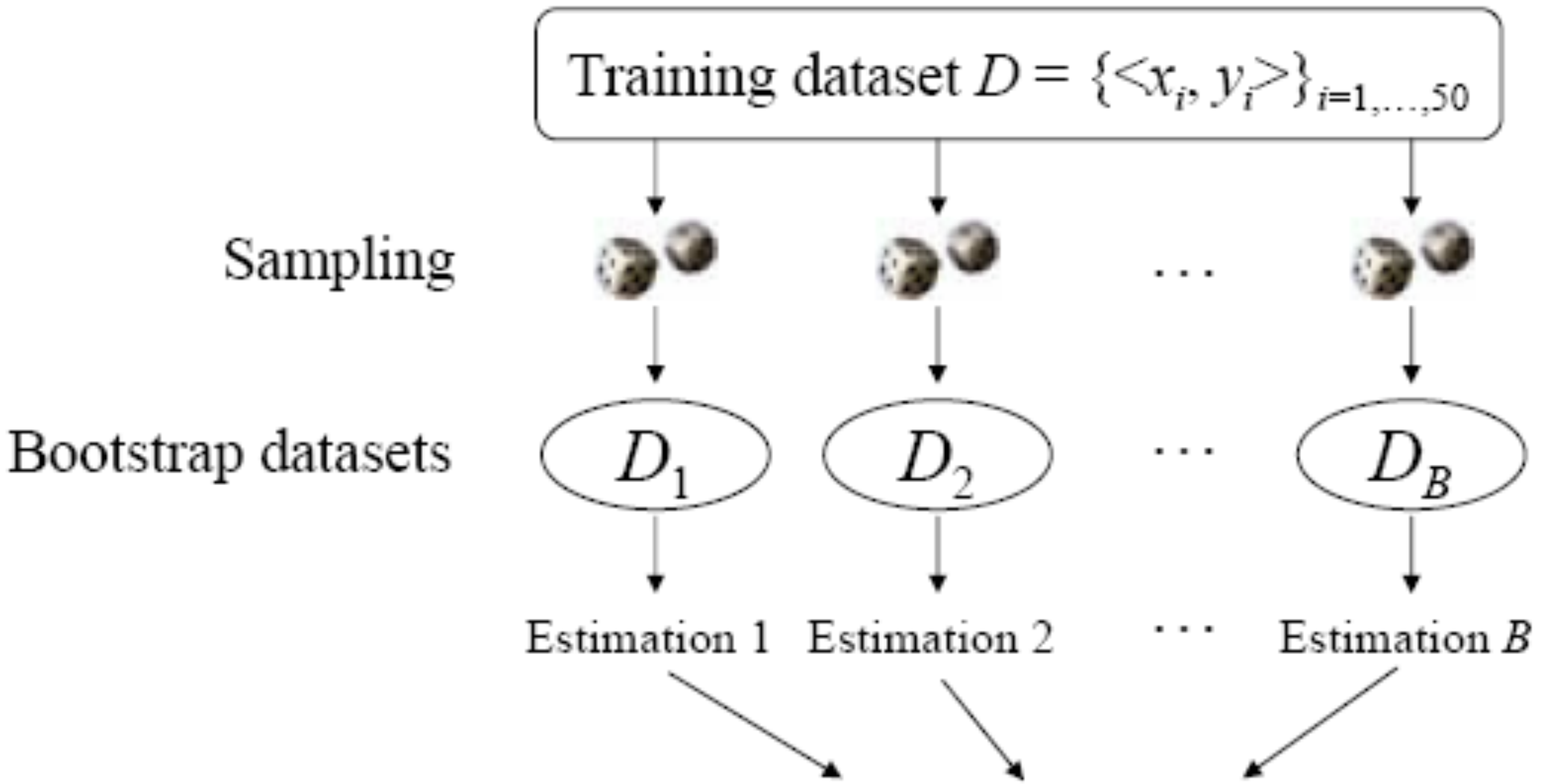
$$Z = h(x|S) - y$$
$$\check{z} = E_S[Z]$$

"Bagging Predictors" [Leo Breiman, 1994]

Bagging = Bootstrap Aggregation

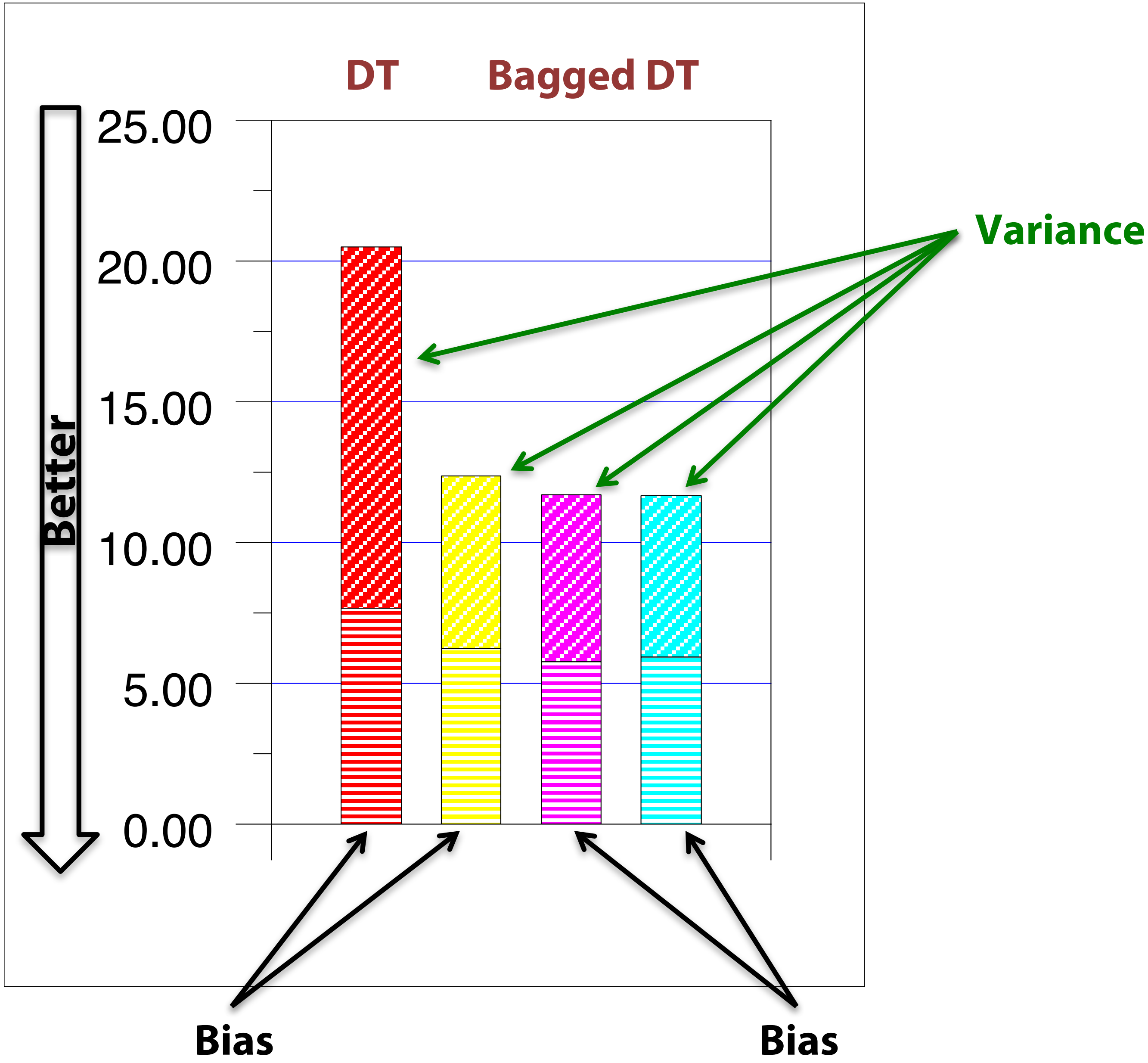
http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf

Bagging



$$\text{Estimation}_{\text{Bagging}} = \frac{1}{B}\sum_{b=1}^{B} \text{Estimation}_{\text{Bootstrap}}$$

Majority voting



"An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants"
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139 (1999)

Random Forests

- **Goal:** reduce variance
  - Bagging can only do so much
  - Resampling training data converges asymptotically to minimum reachable error

- **Random Forests:** sample data & features!
  - Sample S'
  - Train DT
    - At each node, sample feature subset
  - Average predictions

Further de-correlates trees

"Random Forests – Random Features" [Leo Breiman, 1997]
http://oz.berkeley.edu/~breiman/random-forests.pdf

The Random Forest Algorithm

Given a training set S

For i := 1 **to** k **do**:

Build subset S_i by sampling with replacement from S

Learn tree T_i from S_i

At each node:

Choose best split from **random subset of F features**

Each tree grows to the largest extent, and no pruning

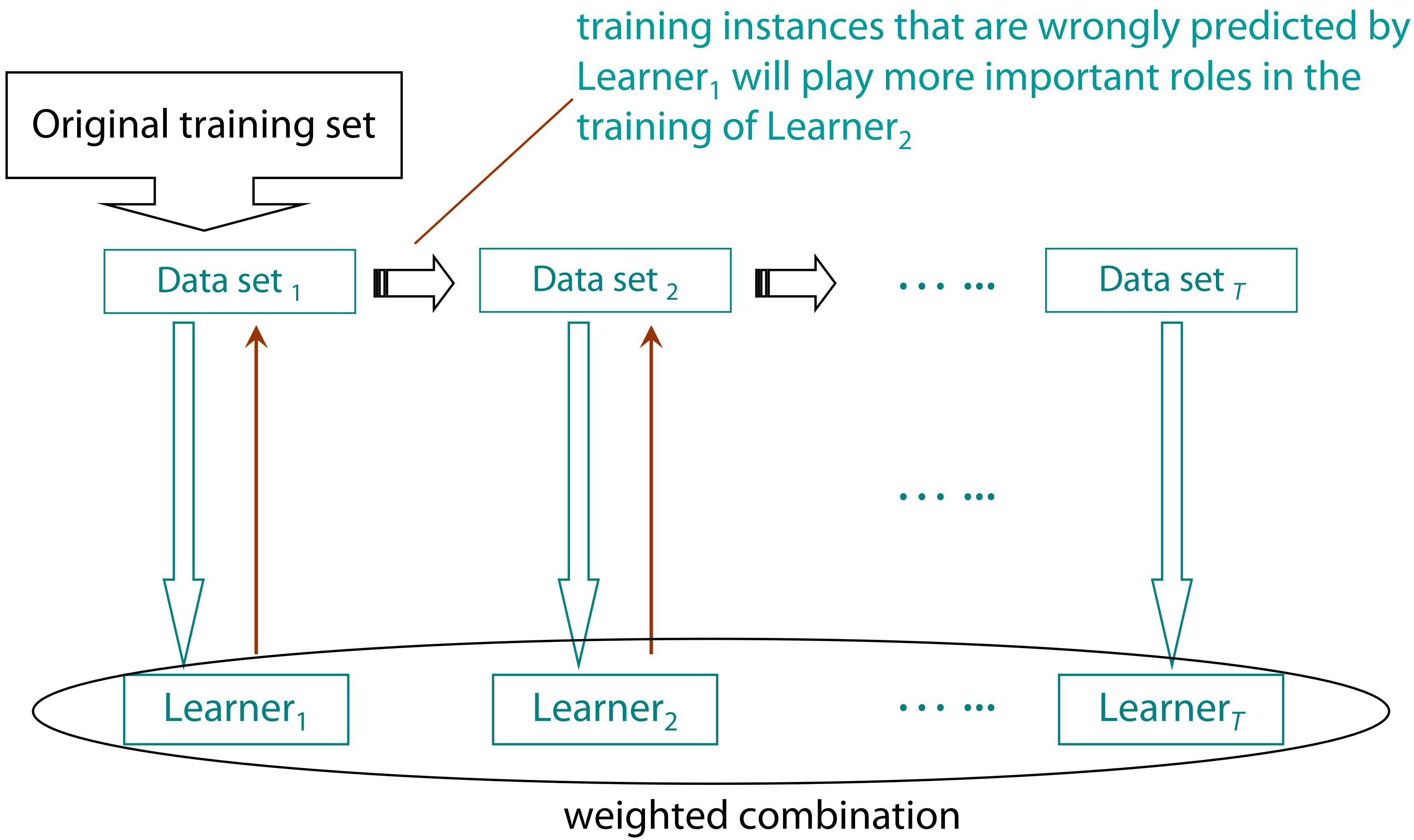
Make predictions according to majority vote of the set of k trees.

Outline

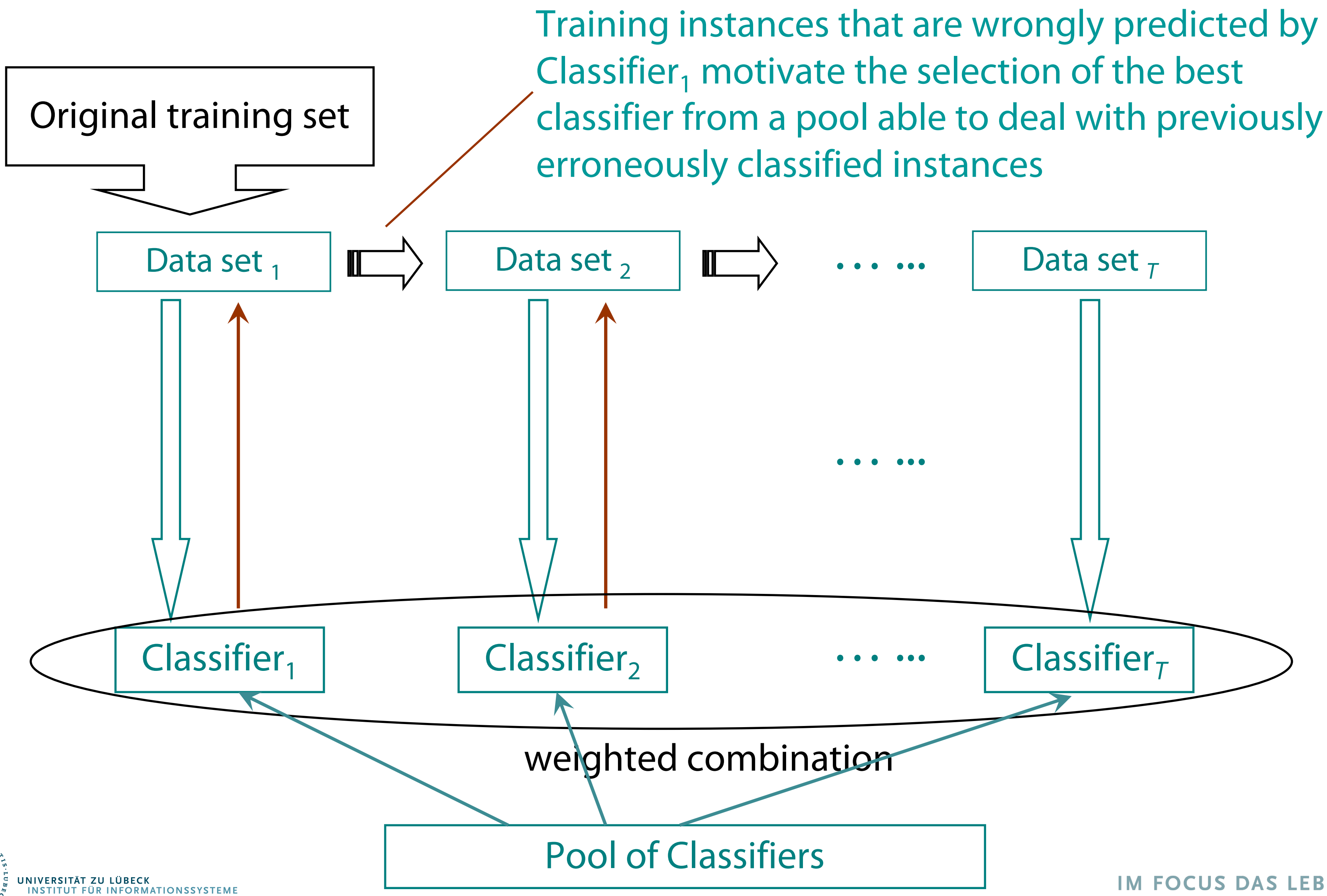
- Bias/Variance Tradeoff
- Ensemble methods that minimize variance
  - Bagging
  - Random Forests
- **Ensemble methods that minimize bias**
  - **Boosting**
  - **Ensemble Selection**

Yoav Freund and Robert Schapire who won the Gödel Prize in 2003

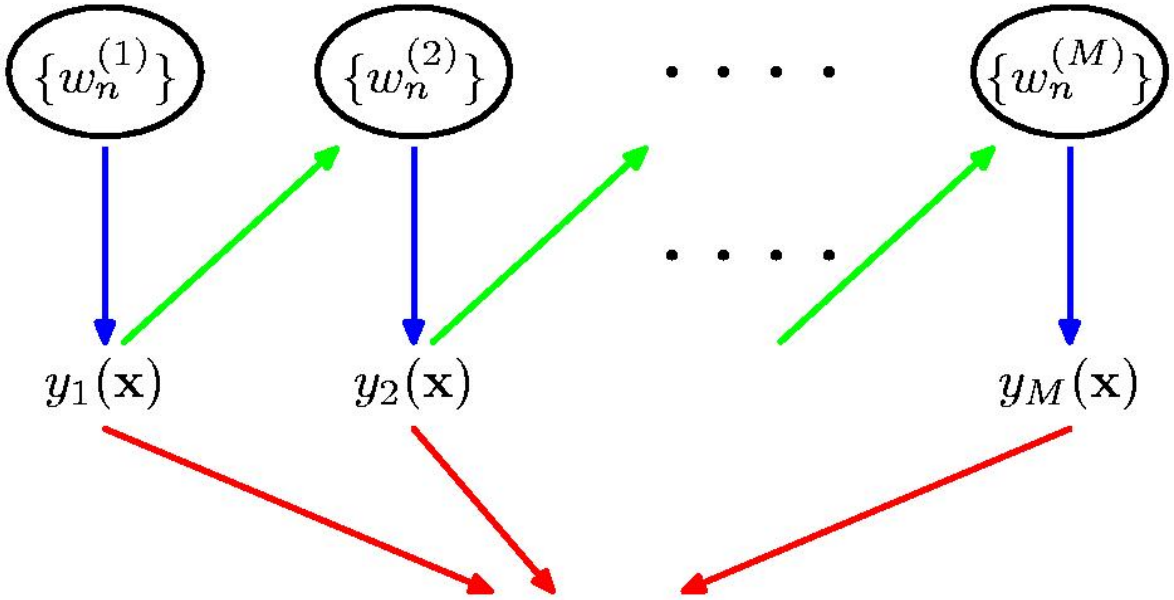
Generation of a Series of Learners



Selection of a Series of Classifiers



Adaptive Boosting (Adaboost)

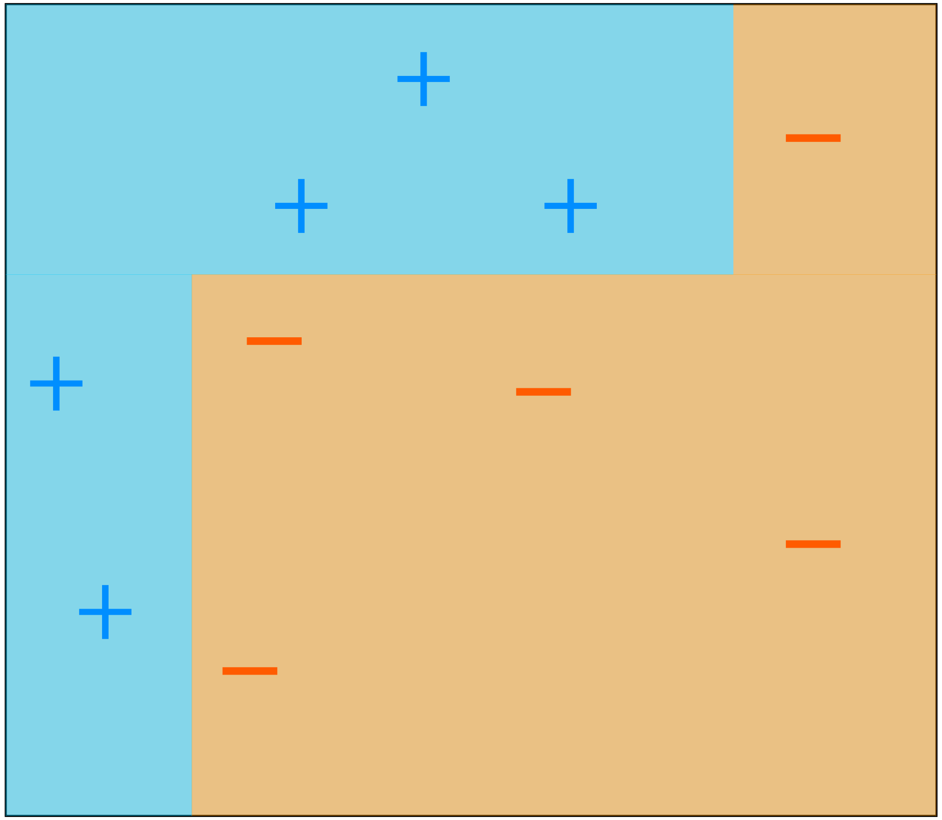


$$Y_M(\mathbf{x}) = \operatorname{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

Target values: 1, -1

Y. Freund, and R. Shapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Proceedings of the Second European Conference on Computational Learning Theory, **1995**, pp. 23–37.

Example of a Good Classifier: Bias minimal



How can we automatically construct such a classifier?

Adaboost (Adaptive Boosting)

- Wanted: Two-class classifier for pattern recognition problem
- Given: Pool of 11 classifiers (experts)
- For a given pattern x_i each expert k_j can emit an opinion $k_j(x_i) \in \{-1, 1\}$
- Final decision: $\mathrm{sign}(C(x))$ where $C(x_i) = \alpha_1 k_1(x_i) + \alpha_2 k_2(x_i) + \cdots + \alpha_{11} k_{11}(x_i)$
- $k_1, k_2, \ldots, k_{11}$ denote the eleven experts
- $\alpha_1, \alpha_2, \ldots, \alpha_{11}$ are the weights we assign to the opinion of each expert
- Problem: How to derive α_j (and k_j)?

Rojas, R. (2009). AdaBoost and the super bowl of classifiers a tutorial introduction to adaptive boosting. Freie University, Berlin, Tech. Rep.

Adaboost: Constructing the Ensemble

- Derive expert ensemble iteratively
- Let us assume we have already m-1 experts
  - $C_{m-1}(x_i) = \alpha_1 k_1(x_i) + \alpha_2 k_2(x_i) + \cdots + \alpha_{m-1} k_{m-1}(x_i)$
- For the next one, classifier m, it holds that
  - $C_m(x_i) = C_{m-1}(x_i) + \alpha_m k_m(x_i)$ with $C_{m-1} = 0$ for $m = 1$
- Let us define an error function for the ensemble
  - If y_i and $C_m(x_i)$ coincide, the error for x_i should be small (in particular when $C_m(x_i)$ is large), if not error should be large
  - $E(x) = \sum_{i=1}^{N} e^{-y_i(C_{m-1}(x_i)+\alpha_m k_m(x_i))}$ where α_m and k_m are to be determined in an optimal way

Adaboost (cntd.)

- $E(x) = \sum_{i=1}^{N} w_i^{(m)} \cdot e^{-y_i \alpha_m k_m(x_i)}$
  with $w_i^{(m)} = e^{-y_i(C_{m-1}(x_i)}$ for $i \in \{1..N\}$ and $w_i^{(1)} = 1$

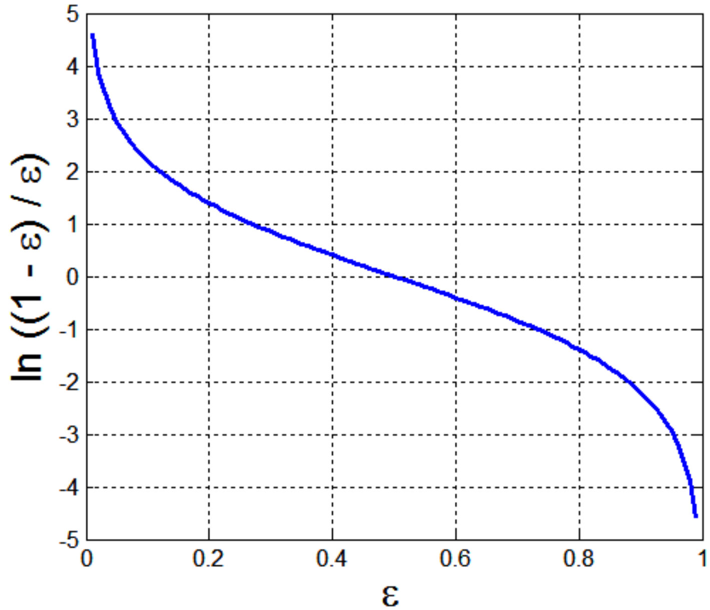
- $E(x) = \sum_{y_i = k_m(x_i)} w_i^{(m)} e^{-\alpha_m} + \sum_{y_i \neq k_m(x_i)} w_i^{(m)} e^{\alpha_m}$
- $E(x) = W_c e^{-\alpha_m} + W_e e^{\alpha_m}$
- $e^{\alpha_m} E(x) = W_c + W_e e^{2\alpha_m}$ $\quad e^{2\alpha_m} > 1$
- $e^{\alpha_m} E(x) = (W_c + W_e) + W_e (e^{2\alpha_m} - 1)$

  constant in each iteration, call it W

- Pick classifier k_m with lowest lowest weighted error to minimize right-hand side of equation
- Select k_m's weight α_m : Solve $\text{argmin}_{\alpha_m} E(x)$

Adaboost (cntd.)

- $dE/d\alpha_m = - W_c\, e^{-\alpha_m} \;+\; W_e\, e^{\alpha_m}$
- Find minimum
- $- W_c\, e^{-\alpha_m} \;+\; W_e\, e^{\alpha_m} = 0$
- $- W_c \;+\; W_e\, e^{2\alpha_m} = 0$
- $\alpha_m = ½ \ln (W_c / W_e)$
- $\alpha_m = ½ \ln ((W - W_e) / W_e)$
- $\alpha_m = ½ \ln ((1 - \varepsilon_m) / \varepsilon_m)$
  with $\varepsilon_m = W_e / W$ being the percentage rate of error given the weights of the data points



AdaBoost

For $m = 1$ to M

1. Select and extract from the pool of classifiers the classifier k_m which minimizes

$$W_e = \sum_{y_i \neq k_m(x_i)} w_i^{(m)}$$

2. Set the weight α_m of the classifier to

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_\mathrm{m}}{\varepsilon_\mathrm{m}} \right)$$

   where $\varepsilon_\mathrm{m} = W_e / W$

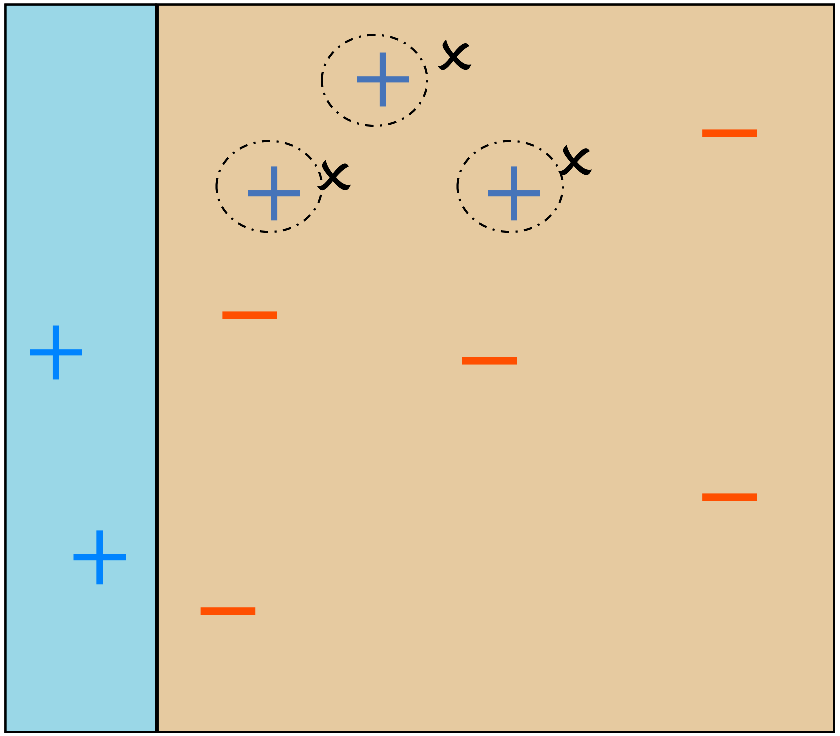
3. Update the weights of the data points for the next iteration. If $k_m(x_i)$ is a miss, set

$$w_i^{(m+1)} = w_i^{(m)} \mathrm{e}^{\alpha_m} = w_i^{(m)} \sqrt{\frac{1 - e_m}{e_m}}$$

   otherwise

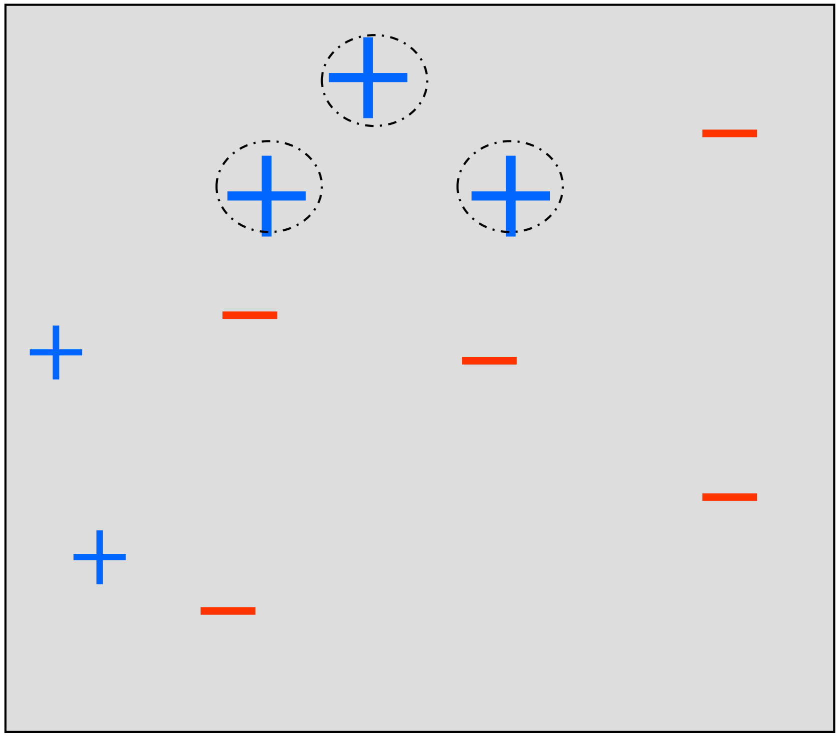
$$w_i^{(m+1)} = w_i^{(m)} \mathrm{e}^{-\alpha_m} = w_i^{(m)} \sqrt{\frac{e_m}{1 - e_m}}$$

Round 1 of 3



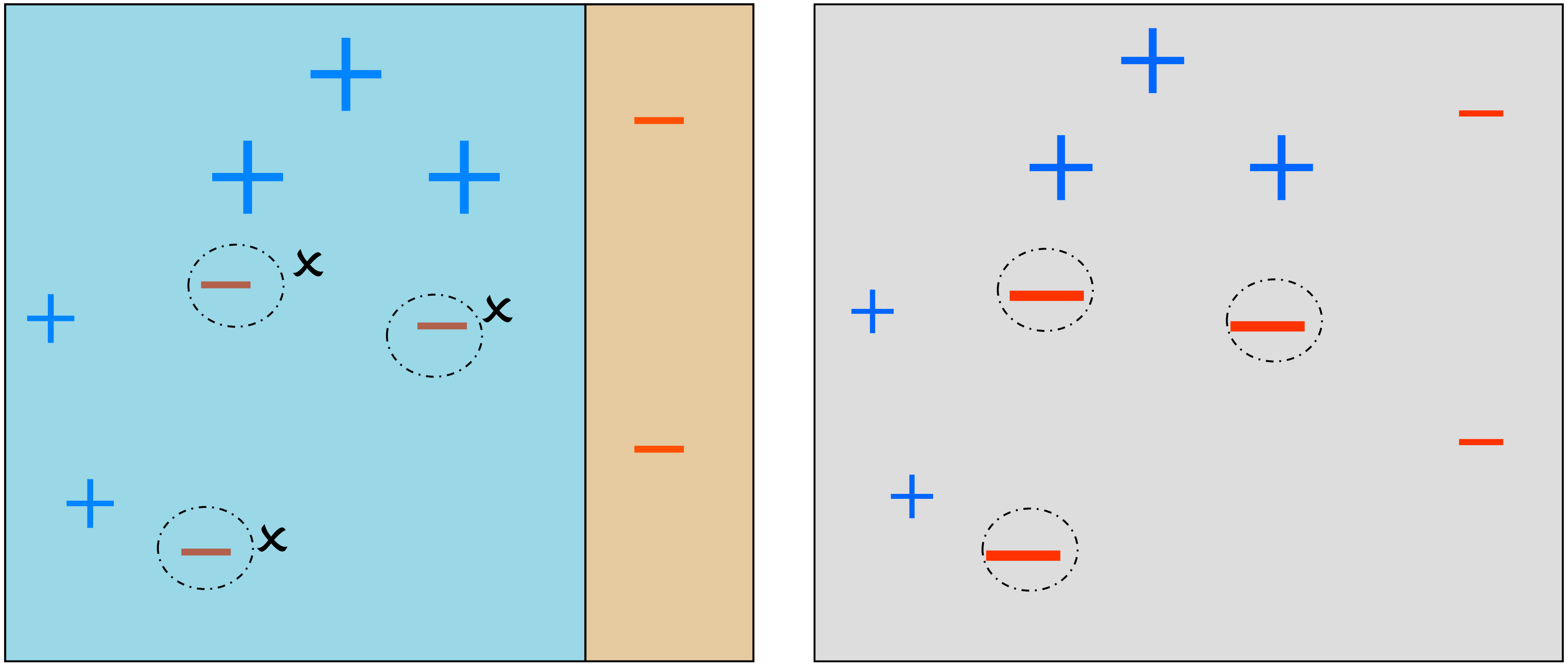
h_1 $\varepsilon_1 = 0.300$

$\alpha_1 = 0.424$



D_2

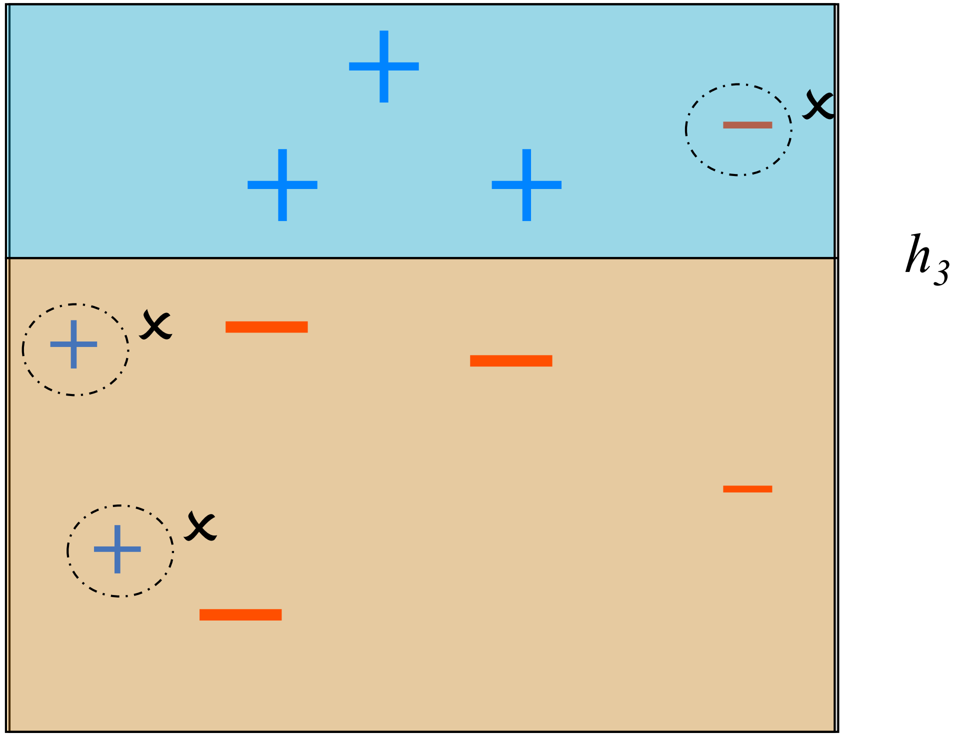
Round 2 of 3



$\varepsilon_2 = 0.196$ h_2 D_2

$\alpha_2 = 0.704$

Round 3 of 3

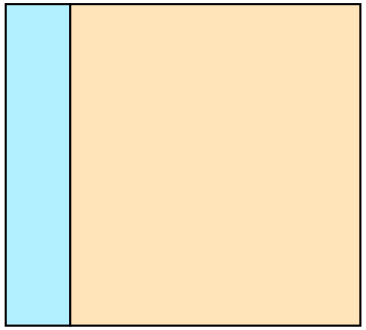
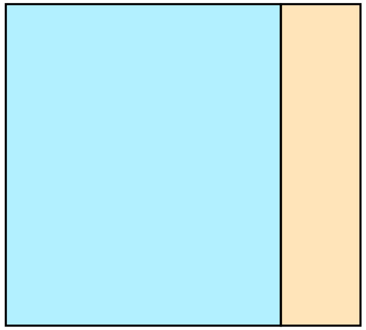


STOP

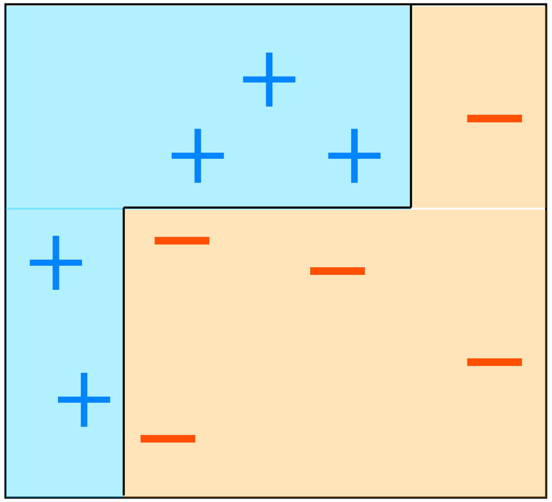
$\varepsilon_3 = 0.344$

$\alpha_2 = 0.323$

Final Hypothesis

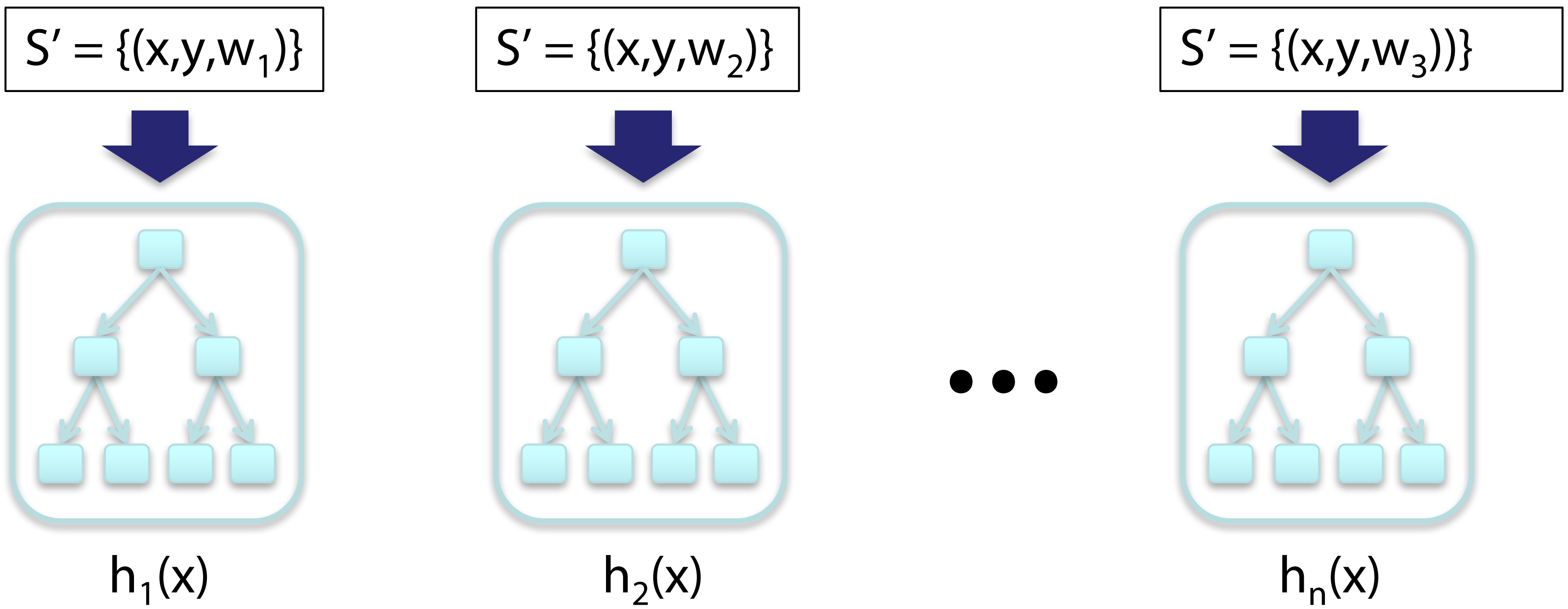
0.42  + 0.70  + 0.32 

$$H_{final} = \mathrm{sign}[\ 0.42(h1?\ 1|\text{-}1) + 0.70(h2?\ 1|\text{-}1) + 0.32(h3?\ 1|\text{-}1)\ ]$$



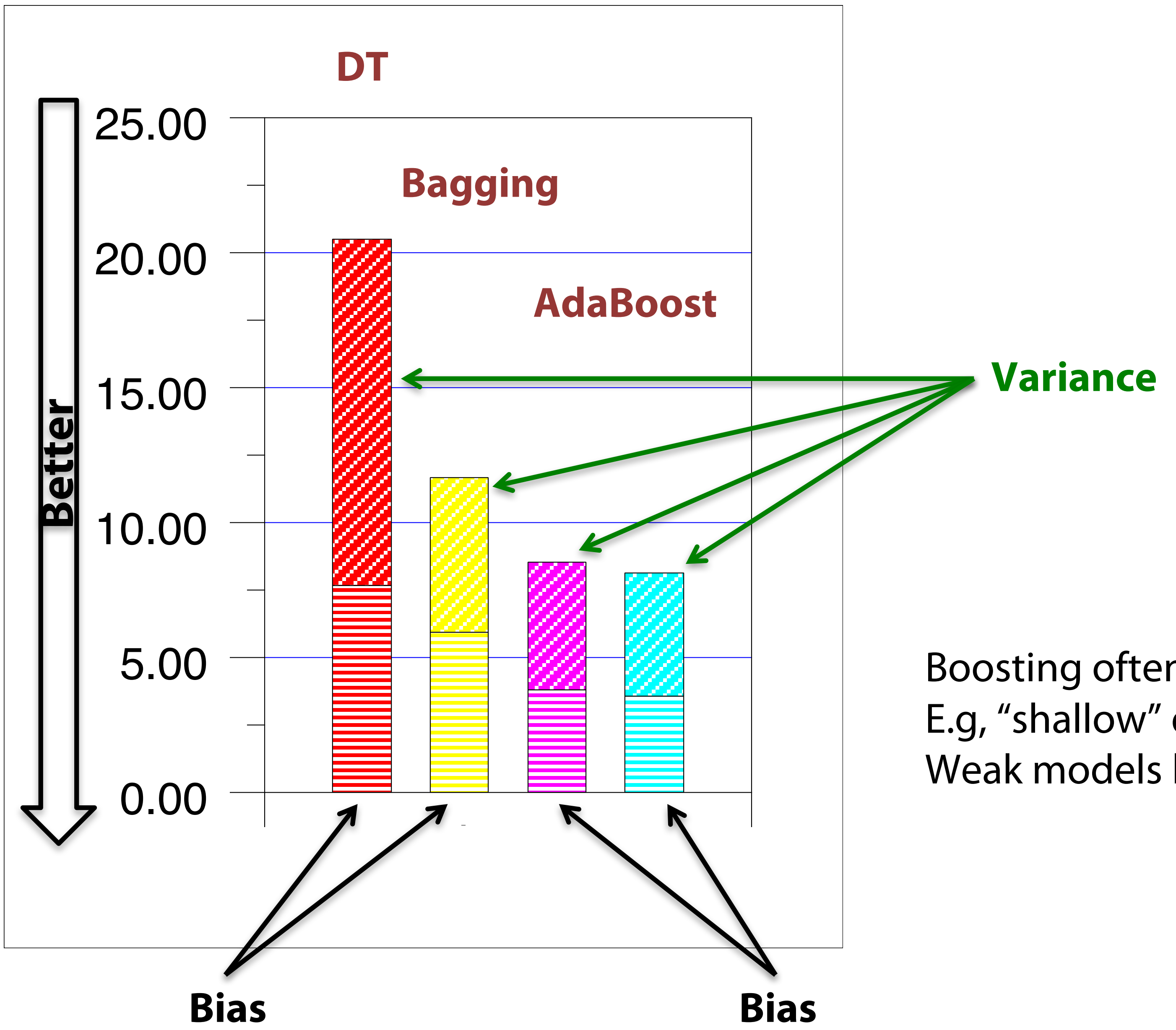
AdaBoost with Decision Trees

$$h(x) = a_1 h_1(x) + a_2 h_2(x) + \ldots + a_n h_n(x)$$



w – weighting on data points
a – weight of linear combination

Stop when validation performance plateaus



Boosting often uses weak models
E.g, “shallow” decision trees
Weak models have lower variance

“An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”
Eric Bauer & Ron Kohavi, Machine Learning 36, 105–139, **1999**

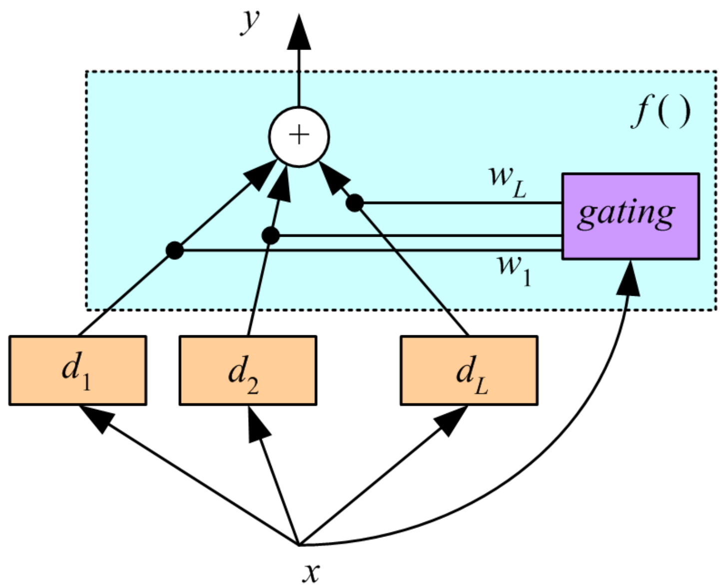
Bagging *vs* Boosting

- Bagging: the construction of complementary base-learners is left to chance and to the unstability of the learning methods.
- Boosting: actively seek to generate complementary base-learner--- training the next base-learner based on the mistakes of the previous learners.

Mixture of experts

- Voting where weights are input-dependent (gating)
- Different input regions convered by different learners (Jacobs et al., 1991)

$$y = \sum_{j=1}^{L} w_j d_j$$

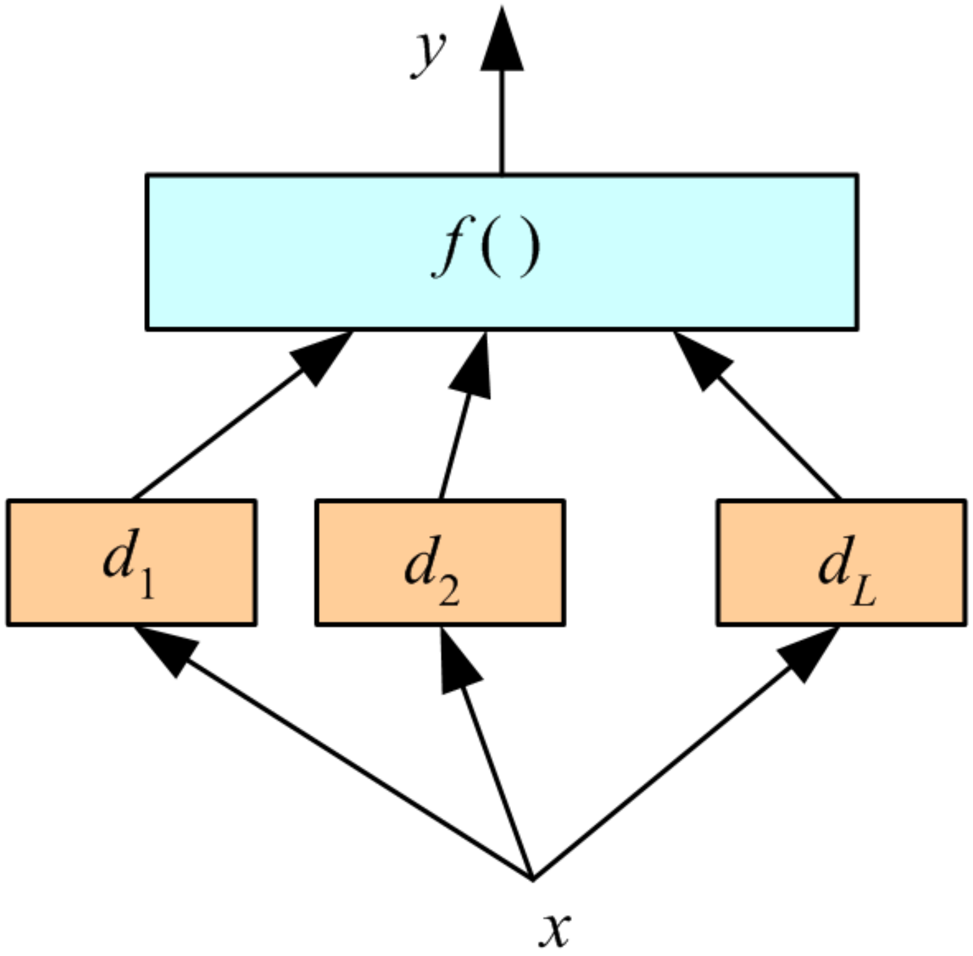


- Gating decides which expert to use
- Need to learn the individual experts as well as the gating functions $w_i(x)$:

$\sum w_j(x) = 1$, for all x

Stacking

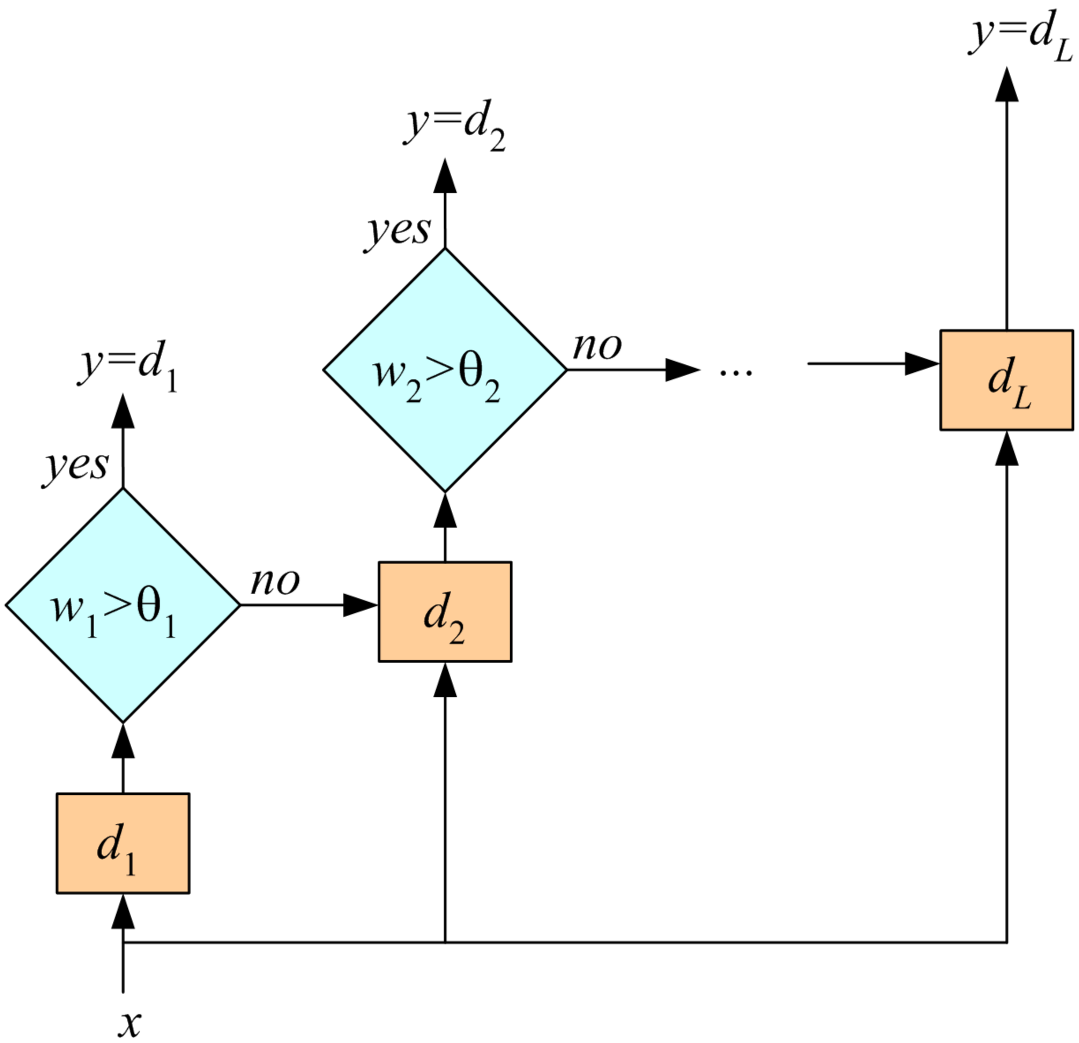
- Combiner *f* () is another learner (Wolpert, 1992)



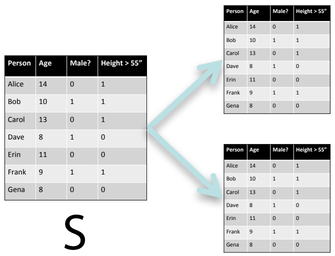
Cascading

Use d_j only if preceding ones are not confident

Cascade learners in order of complexity



Ensemble Selection



S

Training S′

Validation V′

H = {2000 models trained using S′}

Maintain ensemble model as combination of H:

$h(x) = h_1(x) + h_2(x) + \ldots + h_n(x) + h_{n+1}(x)$

Denote as h_{n+1}

Add model from H that maximizes performance on V′

Repeat

Models are trained on S′
Ensemble built to optimize V′

“Ensemble Selection from Libraries of Models”
Caruana, Niculescu-Mizil, Crew & Ksikes, ICML **2004**

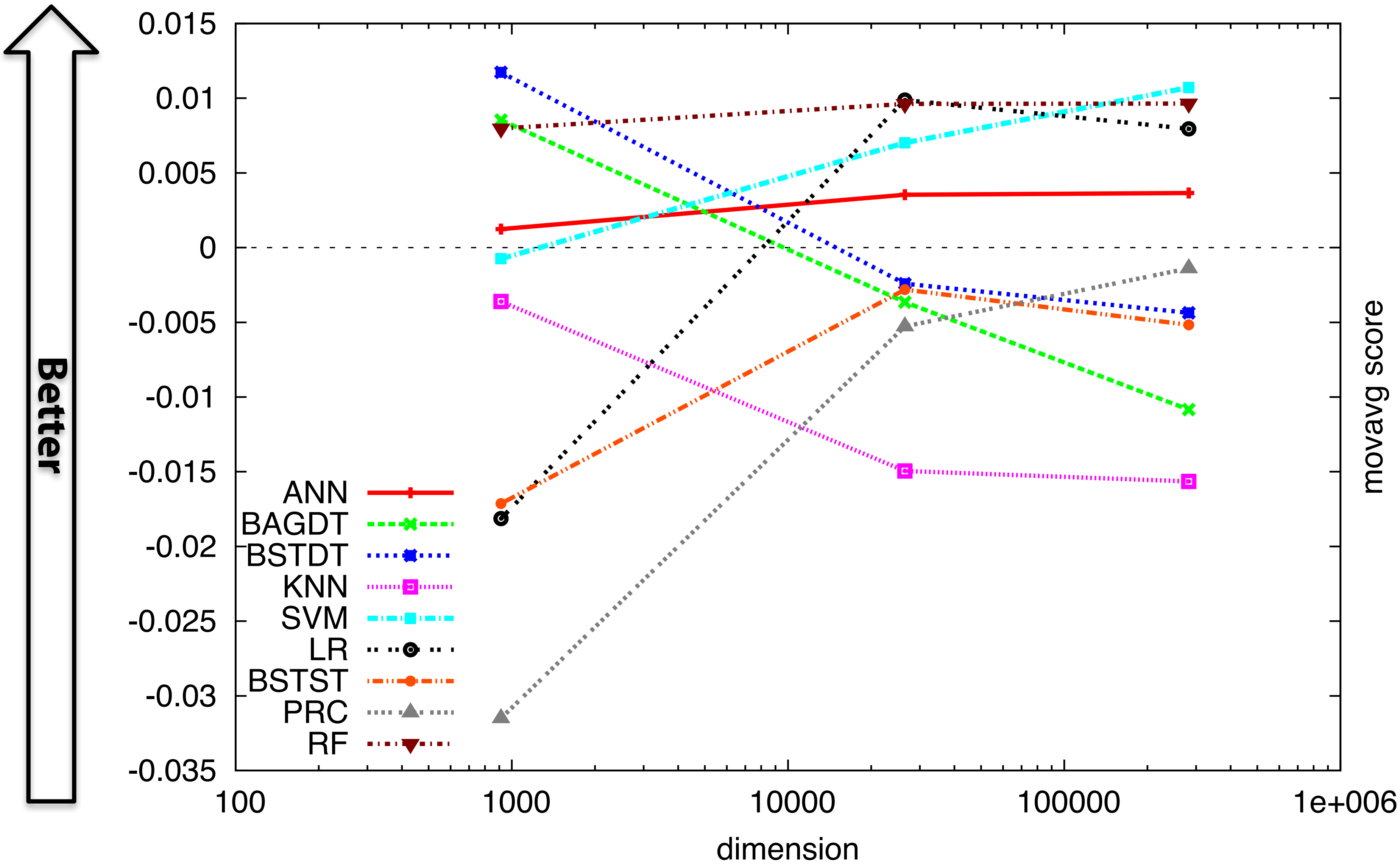
Method	Minimize Bias?	Minimize Variance?	Other Comments
Bagging	Complex model class. (Deep DTs)	Bootstrap aggregation (resampling training data)	Does not work for simple models.
Random Forests	Complex model class. (Deep DTs)	Bootstrap aggregation + bootstrapping features	Only for decision trees.
Gradient Boosting (AdaBoost)	Optimize training performance.	Simple model class. (Shallow DTs)	Determines which model to add at run-time.
Ensemble Selection	Optimize validation performance	Optimize validation performance	Pre-specified dictionary of models learned on training set.

…and many other ensemble methods as well.

- State-of-the-art prediction performance
  - Won Netflix Challenge
  - Won numerous KDD Cups
  - Industry standard

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences. 2009

Although the data sets were constructed to preserve customer privacy, the Prize has been criticized by privacy advocates. In 2007 two researchers from the University of Texas were able to identify individual users by matching the data sets with film ratings on the Internet Movie Database.



Average performance over many datasets
Random Forests perform the best

“An Empirical Evaluation of Supervised Learning in High Dimensions”
Caruana, Karampatziakis & Yessenalina, ICML **2008**