

---

# Einführung in Web- und Data-Science

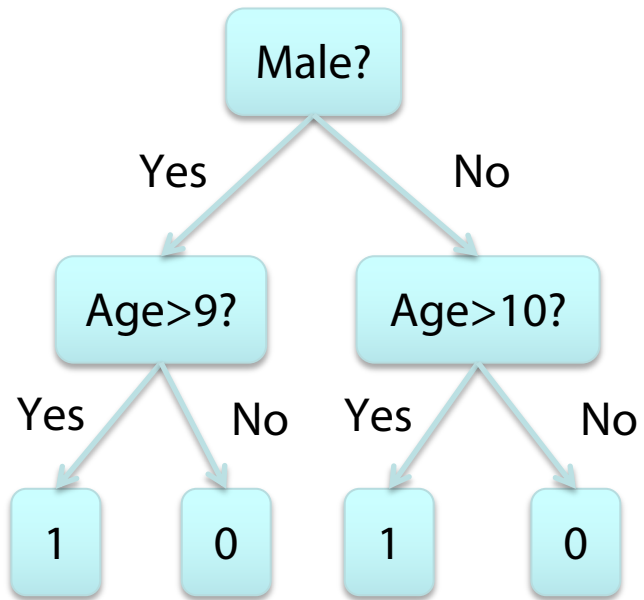
Ensemble Learning

Prof. Dr. Ralf Möller

**Universität zu Lübeck**

**Institut für Informationssysteme**

# Recap: Decision Trees



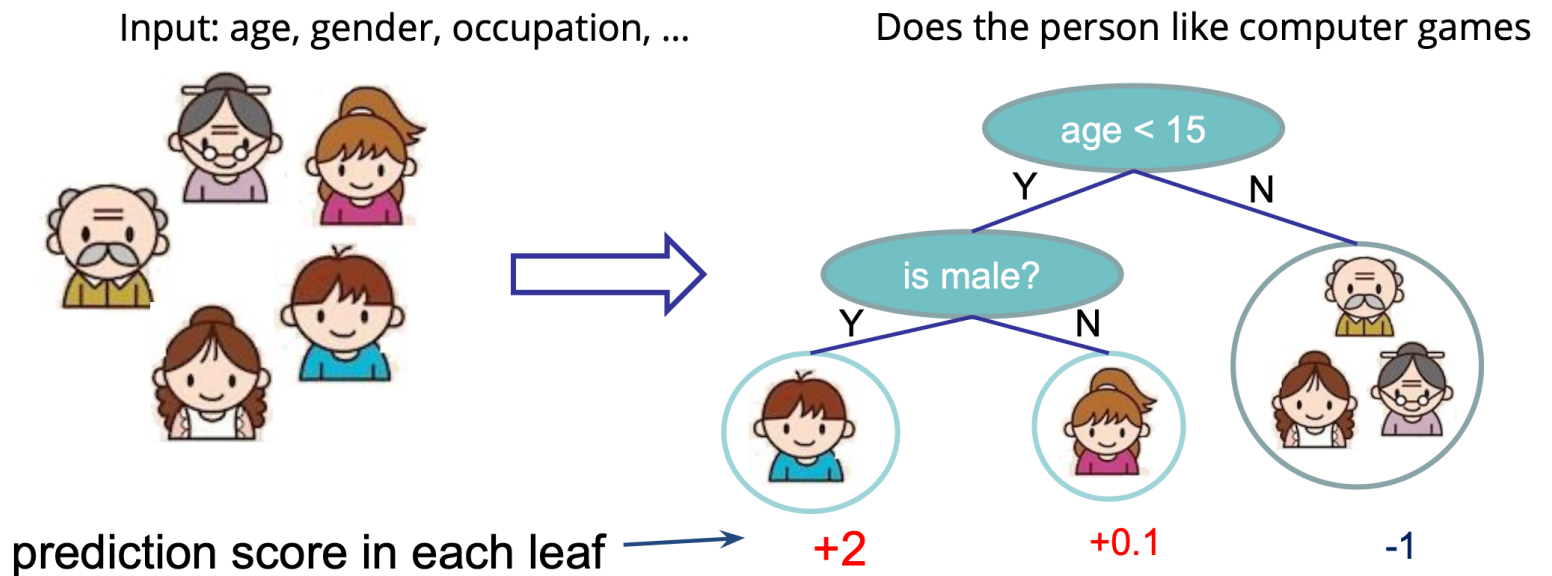
Person	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	8	0	0



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

# Regression Trees

- Regression tree (also known as CART)
- This is what it would look like for a commercial system




















































# Ensembles of Classifiers

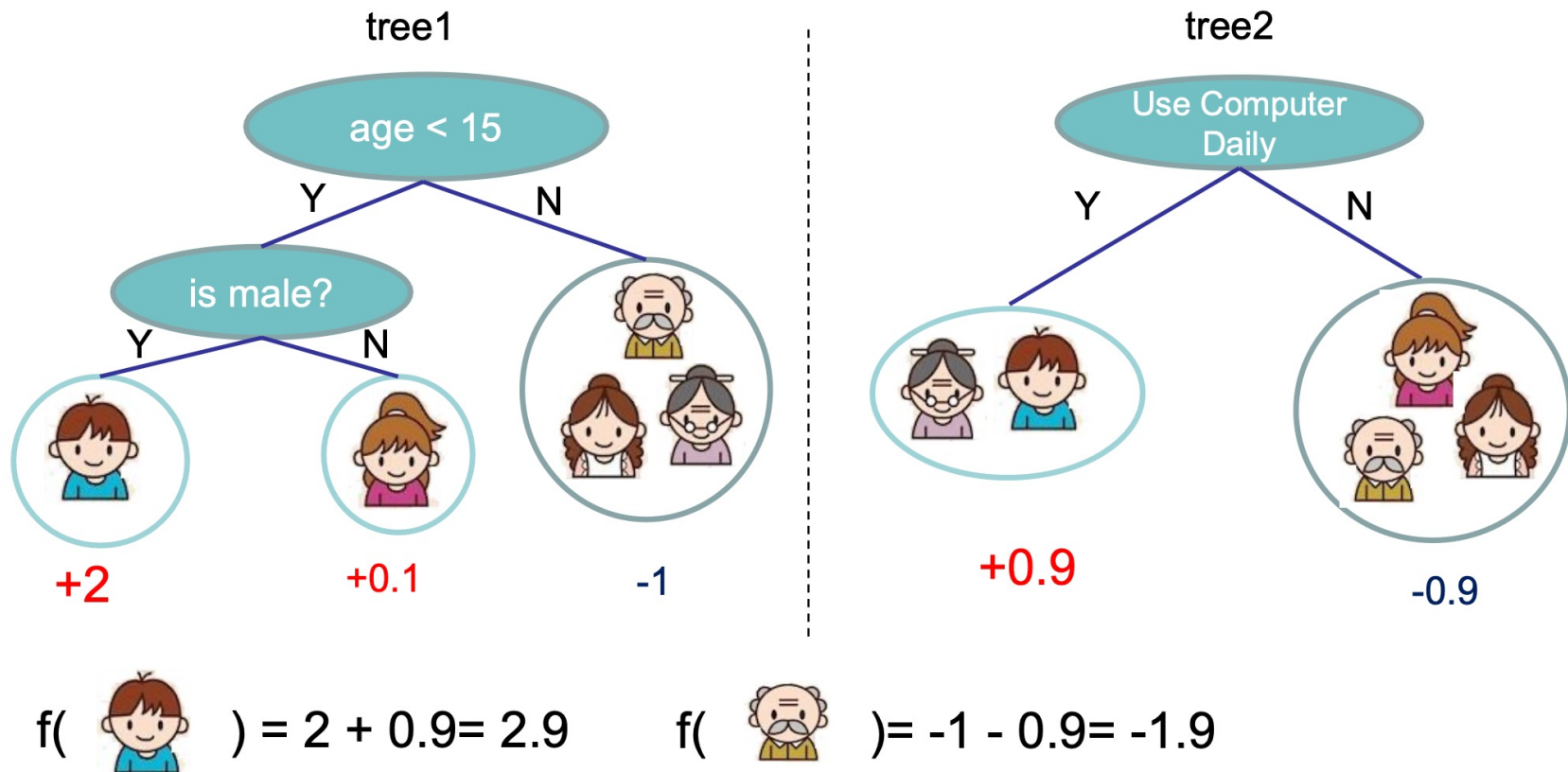
---

- 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

Reality							
1							
2							
3							
4							
5							
Combine							

# Regression Tree Ensembles



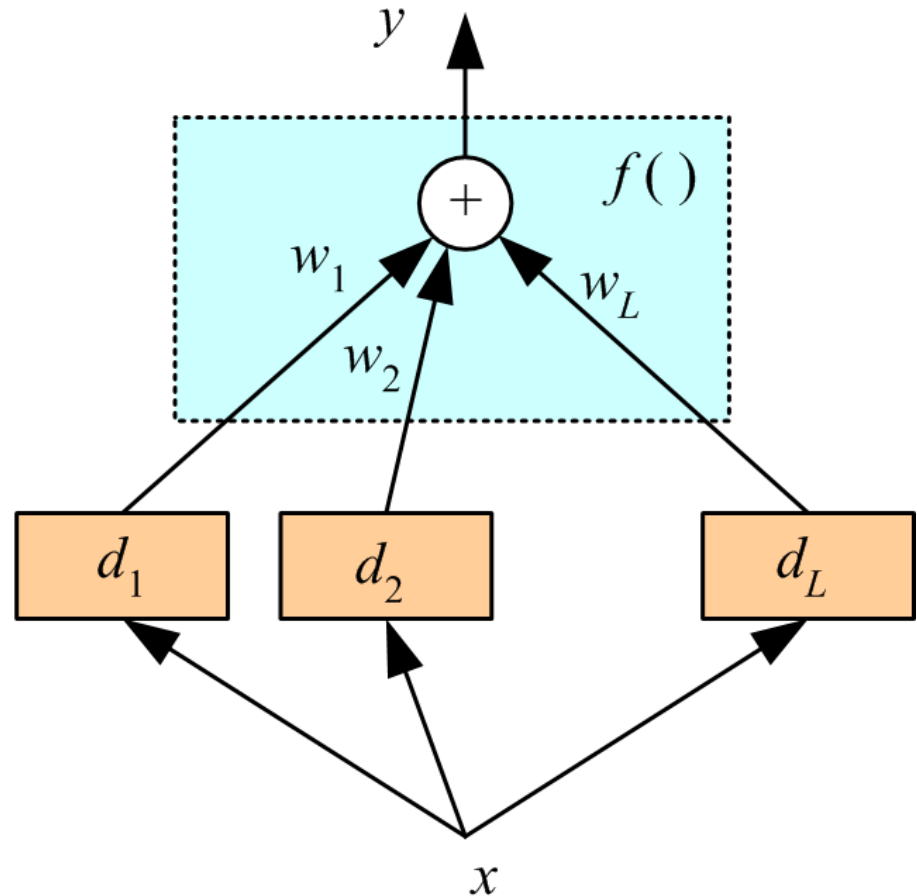
# Voting

- 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"
- Possibly use  $\text{sign}(y) \in \{-1, 1\}$



# Outline

---

- **Bias/Variance Tradeoff**
- Ensemble methods that **minimize variance**
  - Bagging [Breiman 94]
  - Random Forests [Breiman 97]
- Ensemble methods that **minimize bias**
  - Boosting [Freund&Schapire 95, Friedman 98]
  - Ensemble Selection

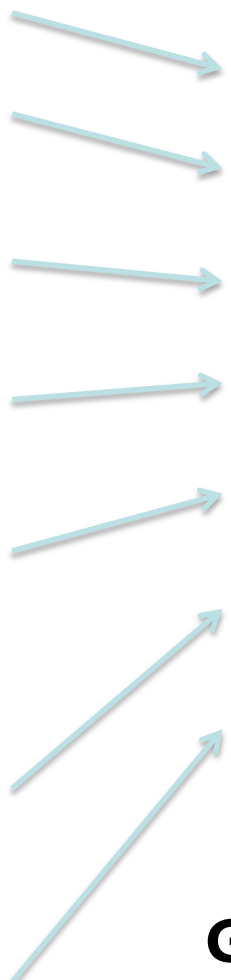


# Generalization Error

---

- **“True” distribution:**  $P(x,y)$ 
  - Unknown to us
- **Train:**  $h(x) = y$ 
  - Using training data  $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$
  - 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



Perso n	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



## Generalization Error:

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

# Bias/Variance Tradeoff

---

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

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

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

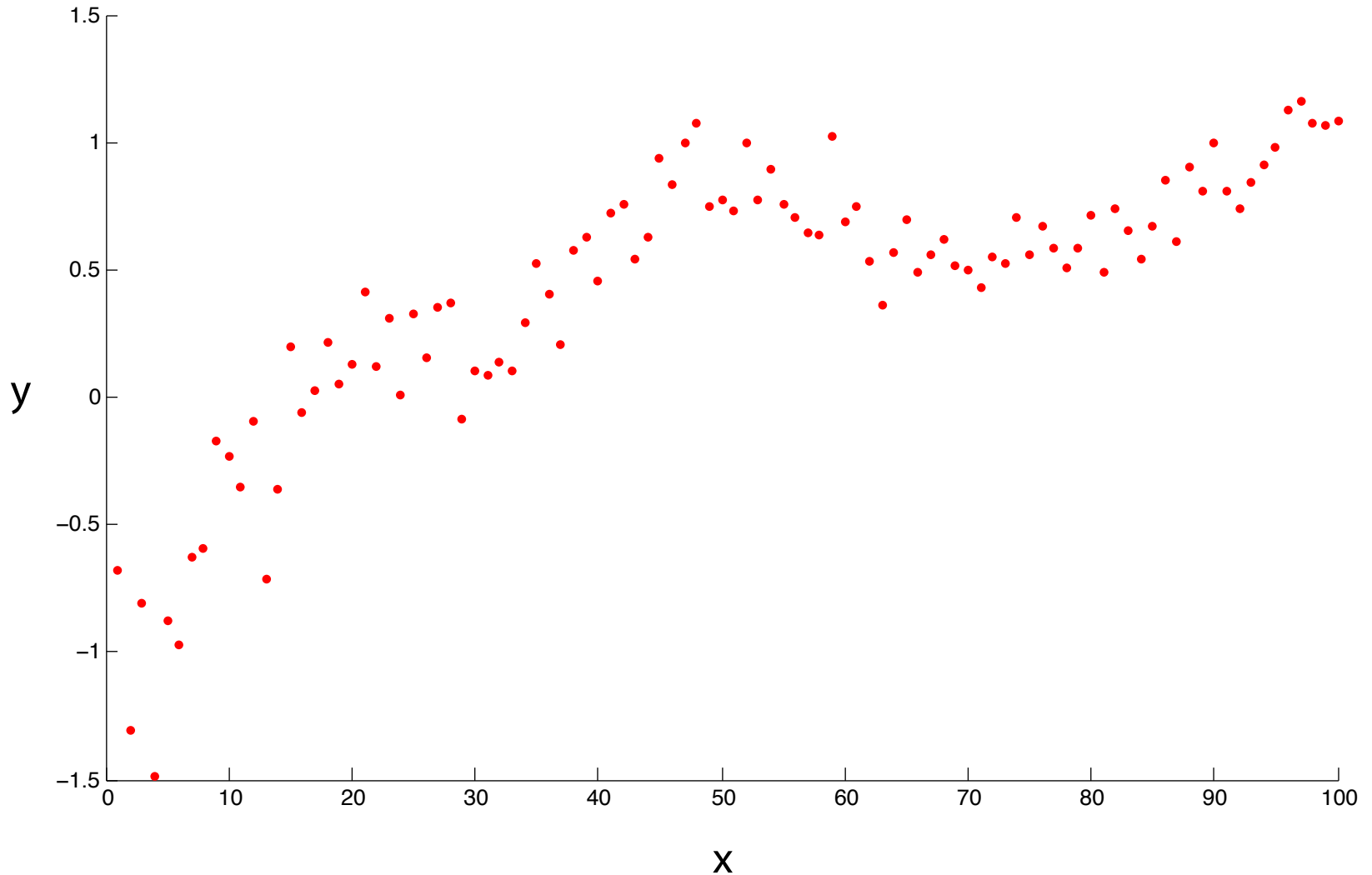
**Expected Error**

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

**Variance**

**Bias**

# Example



$$h(x|S)$$

---



$$h(x|S)$$

---

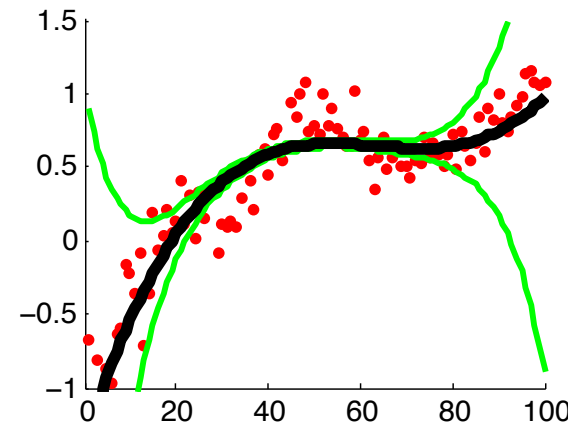
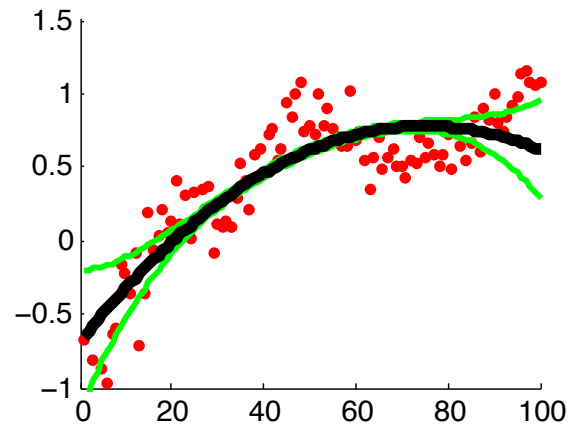
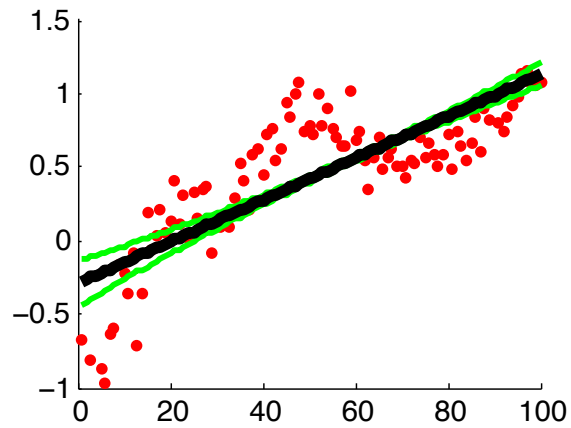


$$h(x|S)$$

---







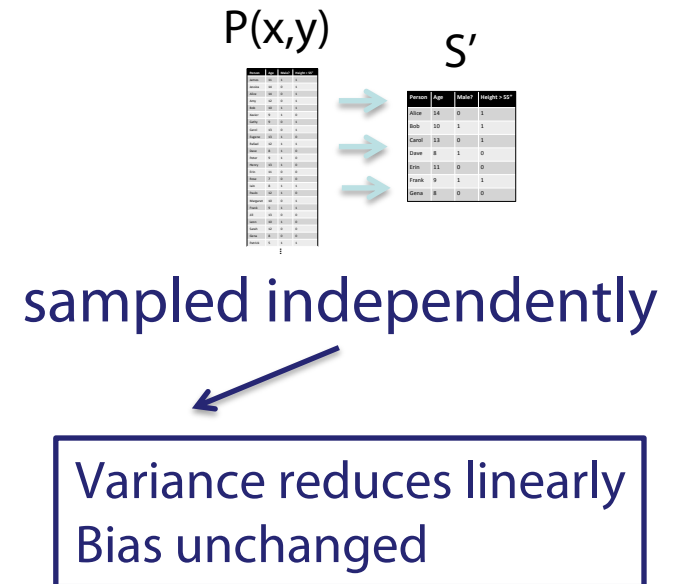
# Outline

---

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

# Bagging

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



$$E_S[(h(x|S) - y)^2] = E_S[(Z - \check{z})^2] + \check{z}^2$$

Expected Error      ↑ Variance      ↑ Bias

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

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

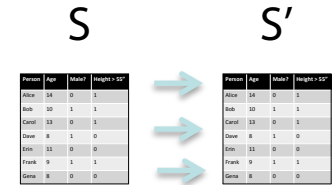
“Bagging Predictors” [Leo Breiman, 1994]

Bagging = Bootstrap Aggregation

# Bagging

Sampling schemes may be *without replacement* ('WOR'—no element can be selected more than once in the same sample) or *with replacement* ('WR'—an element may appear multiple times in the one sample). [Wikipedia]

- **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

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

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

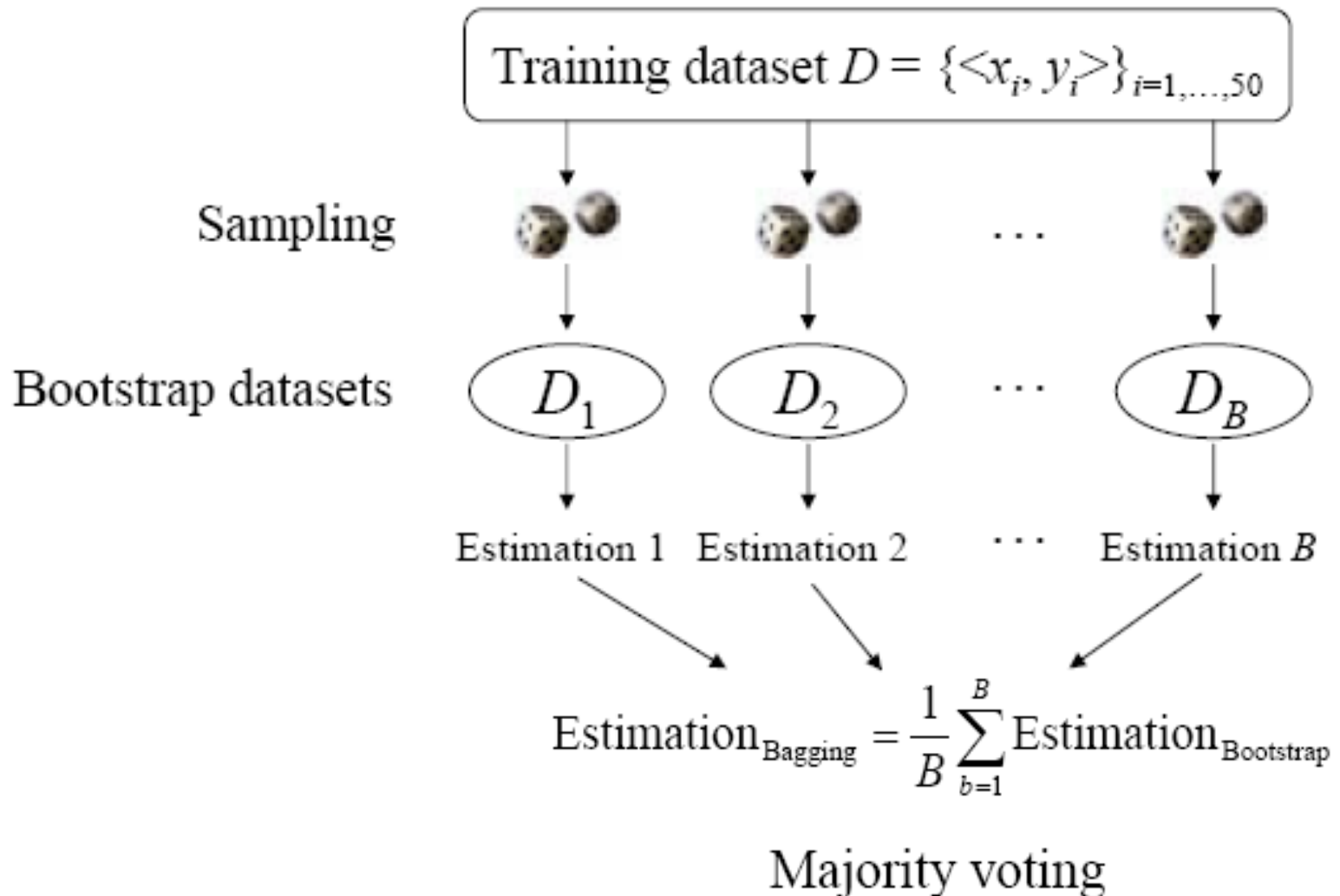
$$E_S[(h(x|S) - y)^2] = E_S[(Z - \check{z})^2] + \check{z}^2$$

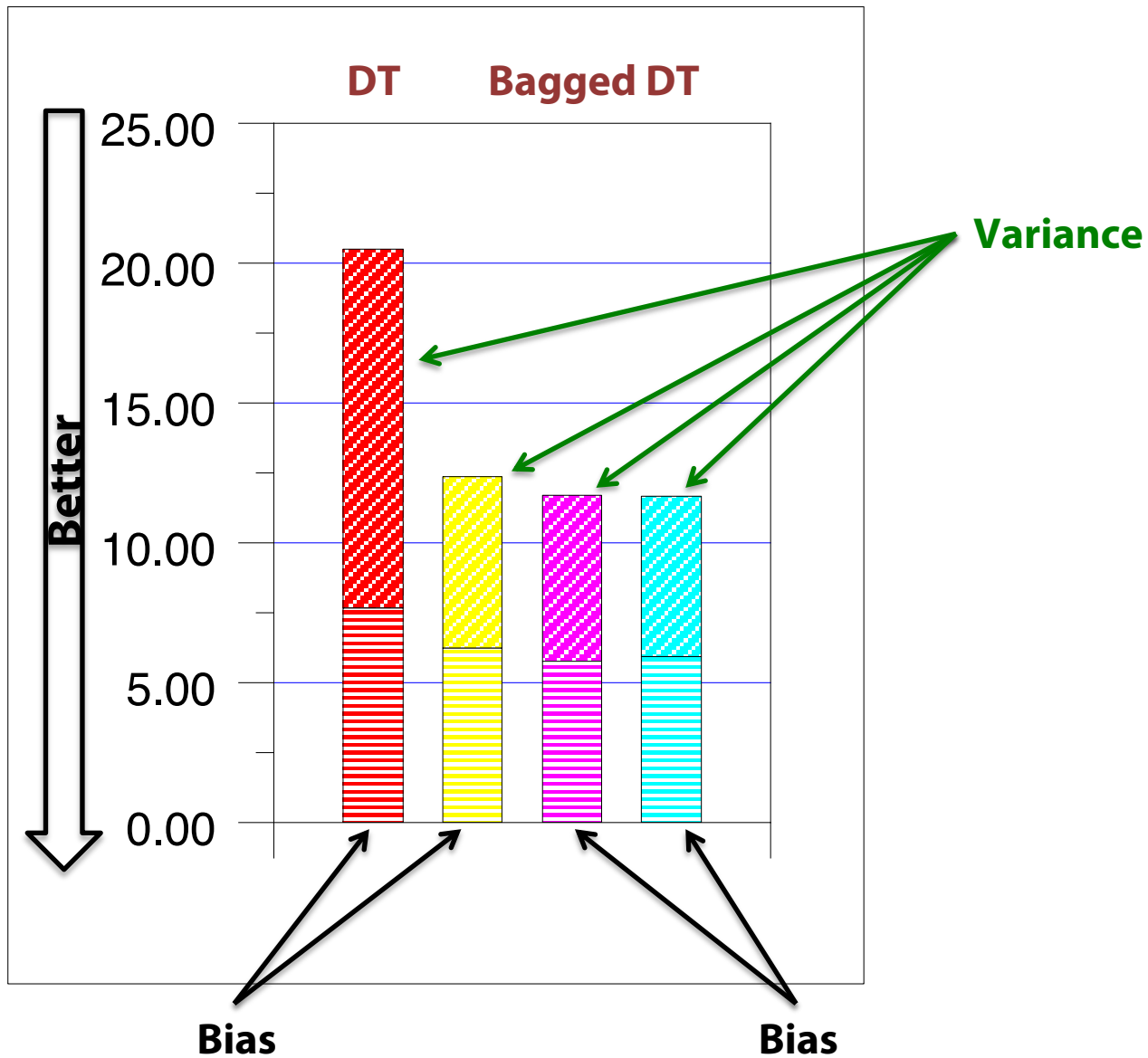
Expected Error      ↑ Variance      ↑ Bias

“Bagging Predictors” [Leo Breiman, 1994]

Bagging = Bootstrap Aggregation

# Bagging





# 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



# 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**

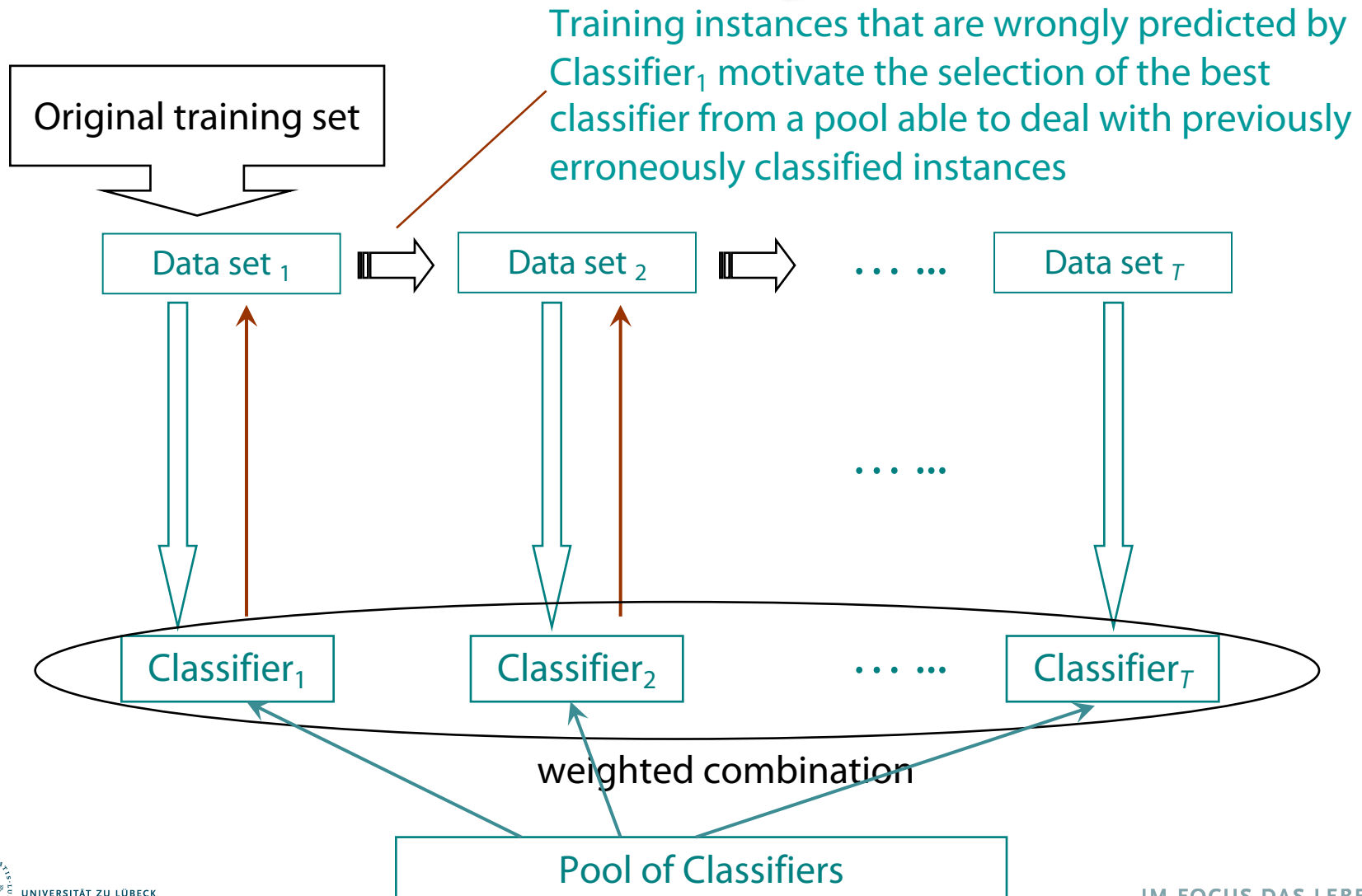
- **Boosting**
- **Ensemble Selection**

[Yoav Freund](#) and [Robert Schapire](#) who won the [Gödel Prize](#) in 2003

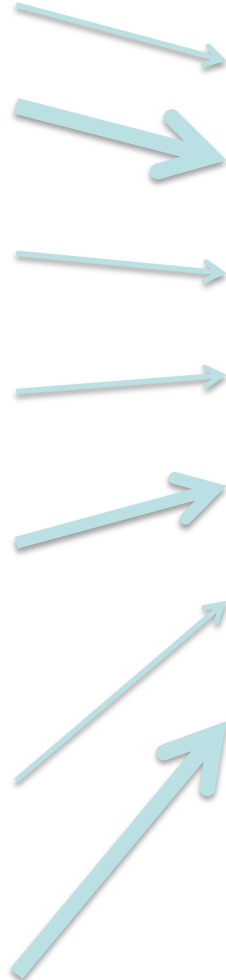
Y. Freund, and R. Shapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Proceedings of the Second European Conference on Computational Learning Theory, 1995, pp. 23–37.

# Selection of a Series of Classifiers

Next set of training instance is determined by weighted sampling



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

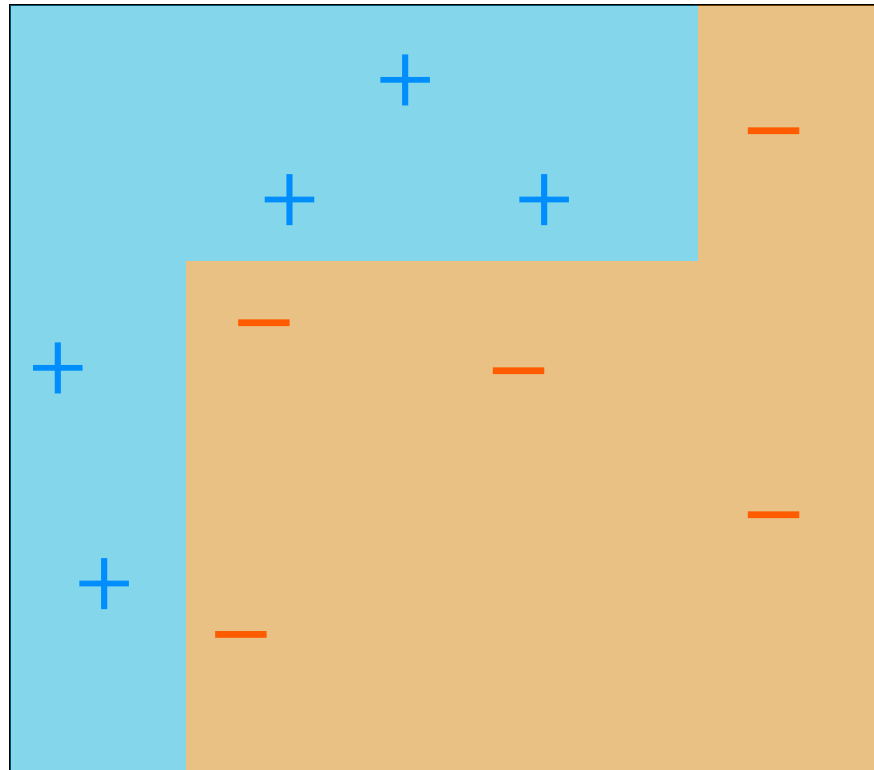


Perso n	Age	Male?	Height > 55"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Erin	11	0	0
Frank	9	1	1
Gena	8	0	0

How to implement weighted sampling?

# 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:  $\text{sign}(C(x))$  where 
$$C(x_i) = \alpha_1 k_1(x_i) + \alpha_2 k_2(x_i) + \dots + \alpha_{11} k_{11}(x_i)$$
- $k_1, k_2, \dots, k_{11}$  denote the eleven experts
- $\alpha_1, \alpha_2, \dots, \alpha_{11}$  are the weights we assign to the opinion of each expert
- Problem: How to derive  $\alpha_j$  (and  $k_j$ )?

# AdaBoost: Constructing the Ensemble

- Derive expert ensemble iteratively
- Let us assume we have already  $m-1$  experts
  - $C_{m-1}(x_i) = \alpha_1 k_1(x_i) + \alpha_2 k_2(x_i) + \dots + \alpha_{m-1} k_{m-1}(x_i)$
- 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  $\alpha_m$  and  $k_m$  are to be determined in an optimal way

# AdaBoost (cntd.)

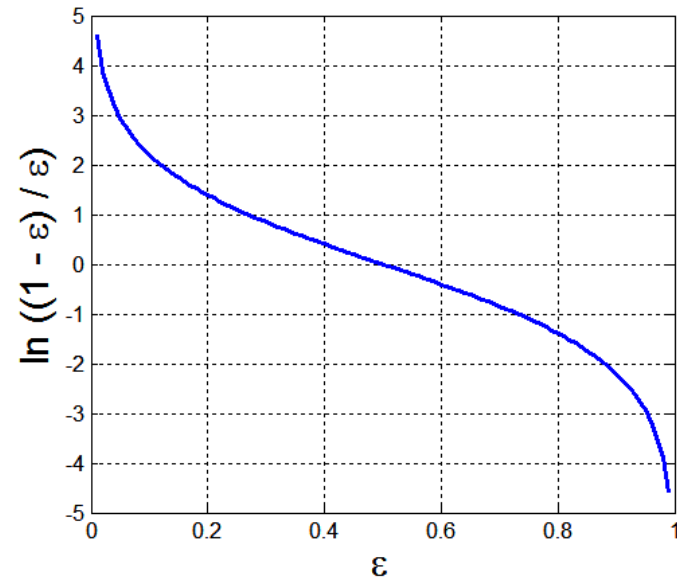
- $E(x) = \sum_{i=1}^N w_i^{(m)} \cdot e^{-y_i \alpha_m k_m(x_i)}$   
with  $w_i^{(m)} = e^{-y_i (C_{m-1}(x_i))}$  for  $i \in \{1..N\}$  and  $w_i^{(1)} = 1$
- $E(x) = \sum_{y_i=k_m(x_i)} w_i^{(m)} e^{-\alpha_m} + \sum_{y_i \neq k_m(x_i)} w_i^{(m)} e^{\alpha_m}$
- $E(x) = W_c e^{-\alpha_m} + W_e e^{\alpha_m}$
- $e^{\alpha_m} E(x) = W_c + W_e e^{2\alpha_m} \quad e^{2\alpha_m} > 1$
- $e^{\alpha_m} E(x) = (W_c + W_e) + W_e (e^{2\alpha_m} - 1)$

constant in each iteration, call it  $W$

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

# AdaBoost (cntd.)

- $\delta E / \delta \alpha_m = -W_c e^{-\alpha_m} + W_e e^{\alpha_m}$
- Find minimum
- $-W_c e^{-\alpha_m} + W_e e^{\alpha_m} = 0$
- $-W_c + W_e e^{2\alpha_m} = 0$
- $\alpha_m = \frac{1}{2} \ln (W_c / W_e)$
- $\alpha_m = \frac{1}{2} \ln ((W - W_e) / W_e)$
- $\alpha_m = \frac{1}{2} \ln ((1 - \varepsilon_m) / \varepsilon_m)$   
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  $\alpha_m$  of the classifier to

$$\alpha_m = \frac{1}{2} \ln \left( \frac{1 - \varepsilon_m}{\varepsilon_m} \right)$$

where  $\varepsilon_m = W_e / W$

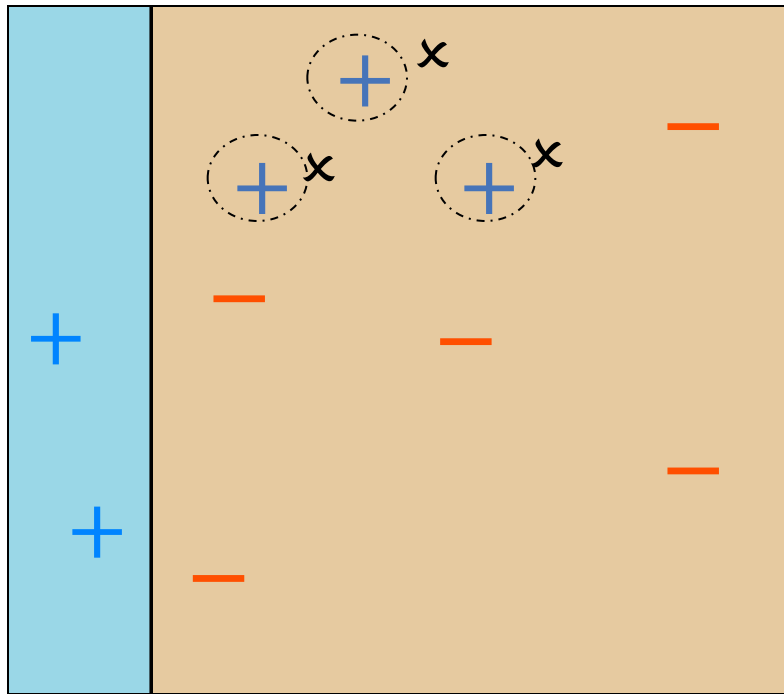
3. Update the weights of the data points for the next iteration. If  $k_m(x_i)$  is a miss, set

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

otherwise

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

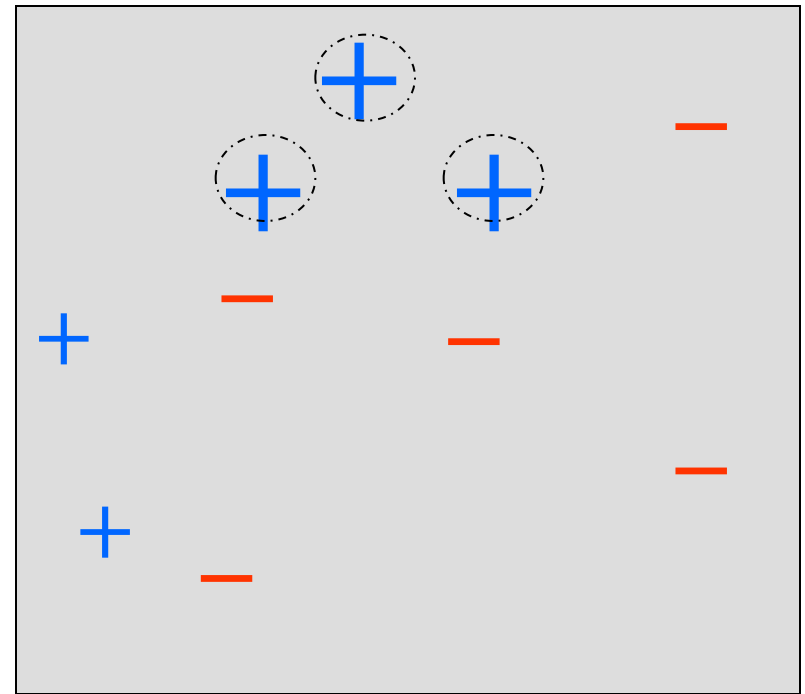
# Round 1 of 3



$h_1$

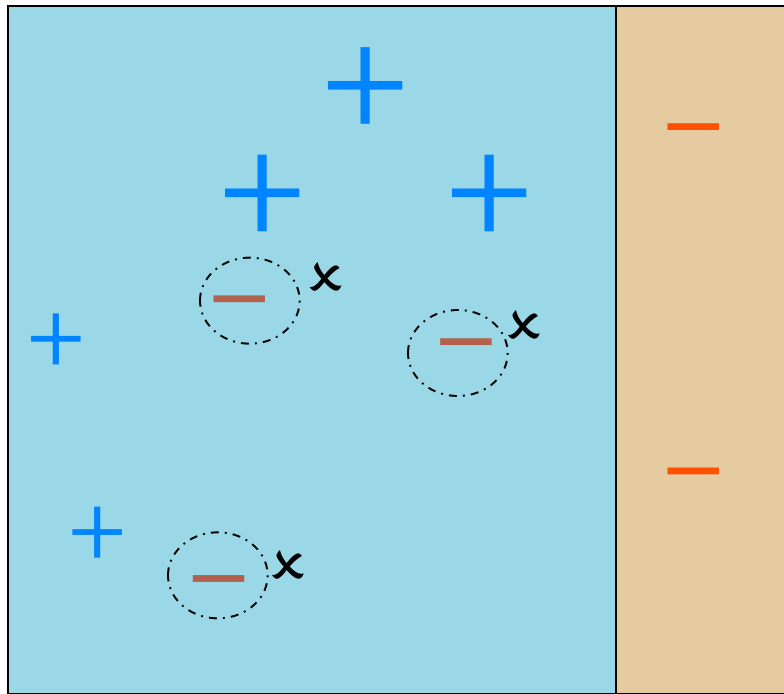
$$\varepsilon_1 = 0.300$$

$$\alpha_1 = 0.424$$



$D_2$

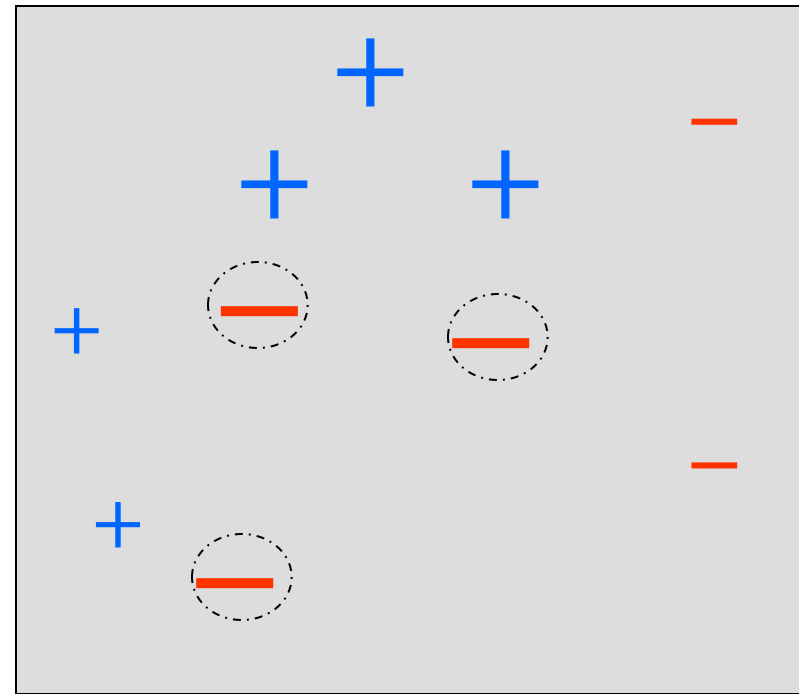
# Round 2 of 3



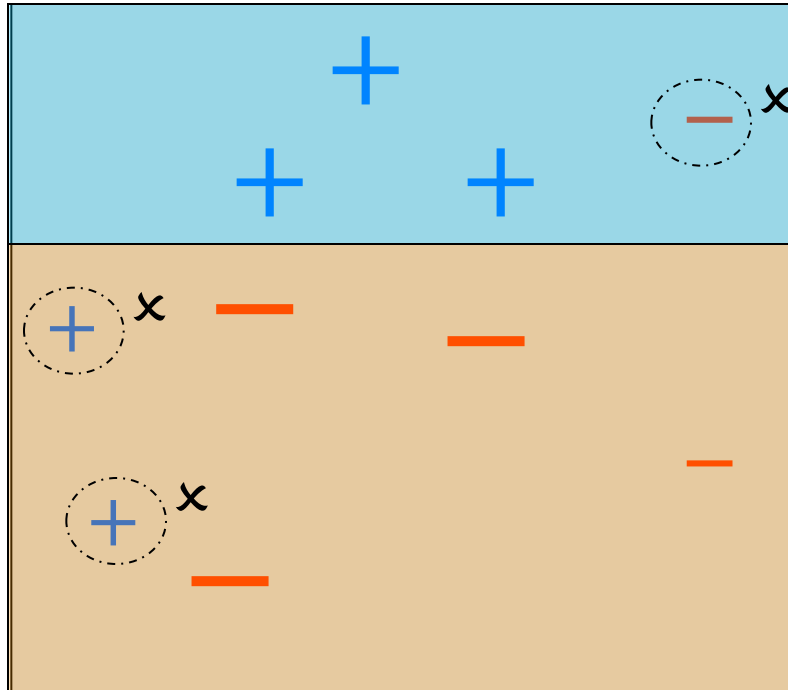
$$\varepsilon_2 = 0.196$$

$$h_2$$

$$\alpha_2 = 0.704$$


$$D_2$$

# Round 3 of 3



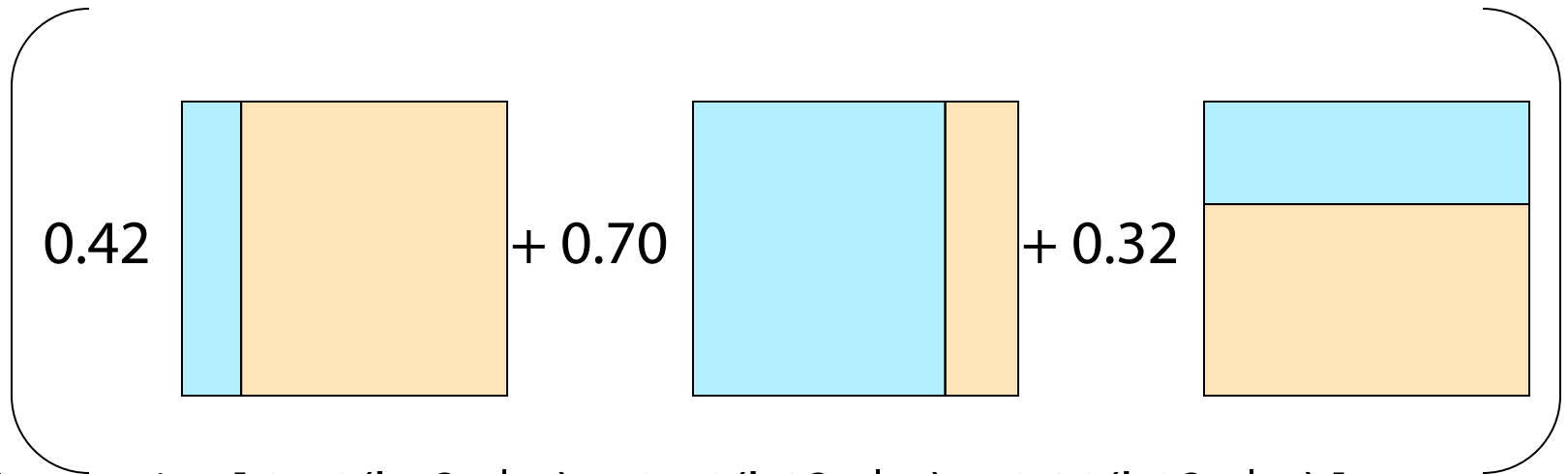
$h_3$

STOP

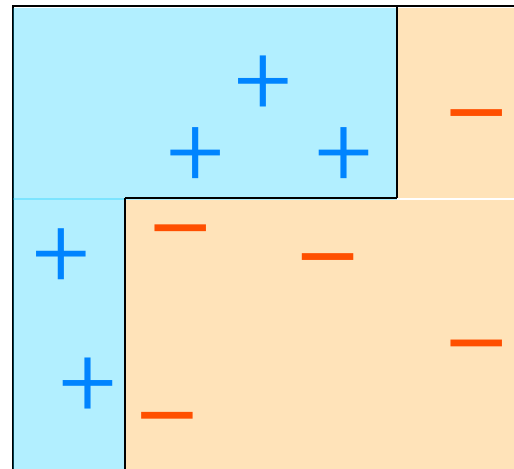
$$\varepsilon_3 = 0.344$$

$$\alpha_2 = 0.323$$

# Final Hypothesis



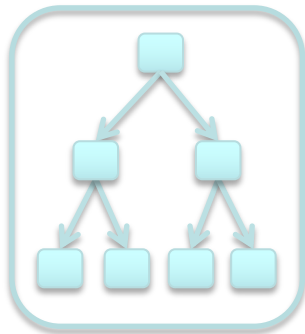
$$H_{\text{final}} = \text{sign}[ 0.42(h_1? 1|-1) + 0.70(h_2? 1|-1) + 0.32(h_3? 1|-1) ]$$



# AdaBoost with Decision Trees

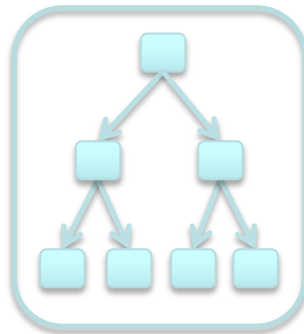
$$h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_n h_n(x)$$

$$S' = \{(x, y, w_1)\}$$



$h_1(x)$

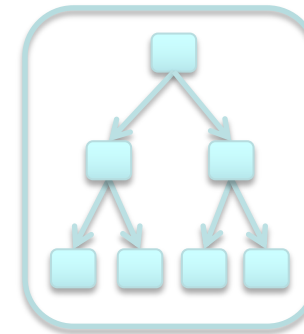
$$S' = \{(x, y, w_2)\}$$



$h_2(x)$

...

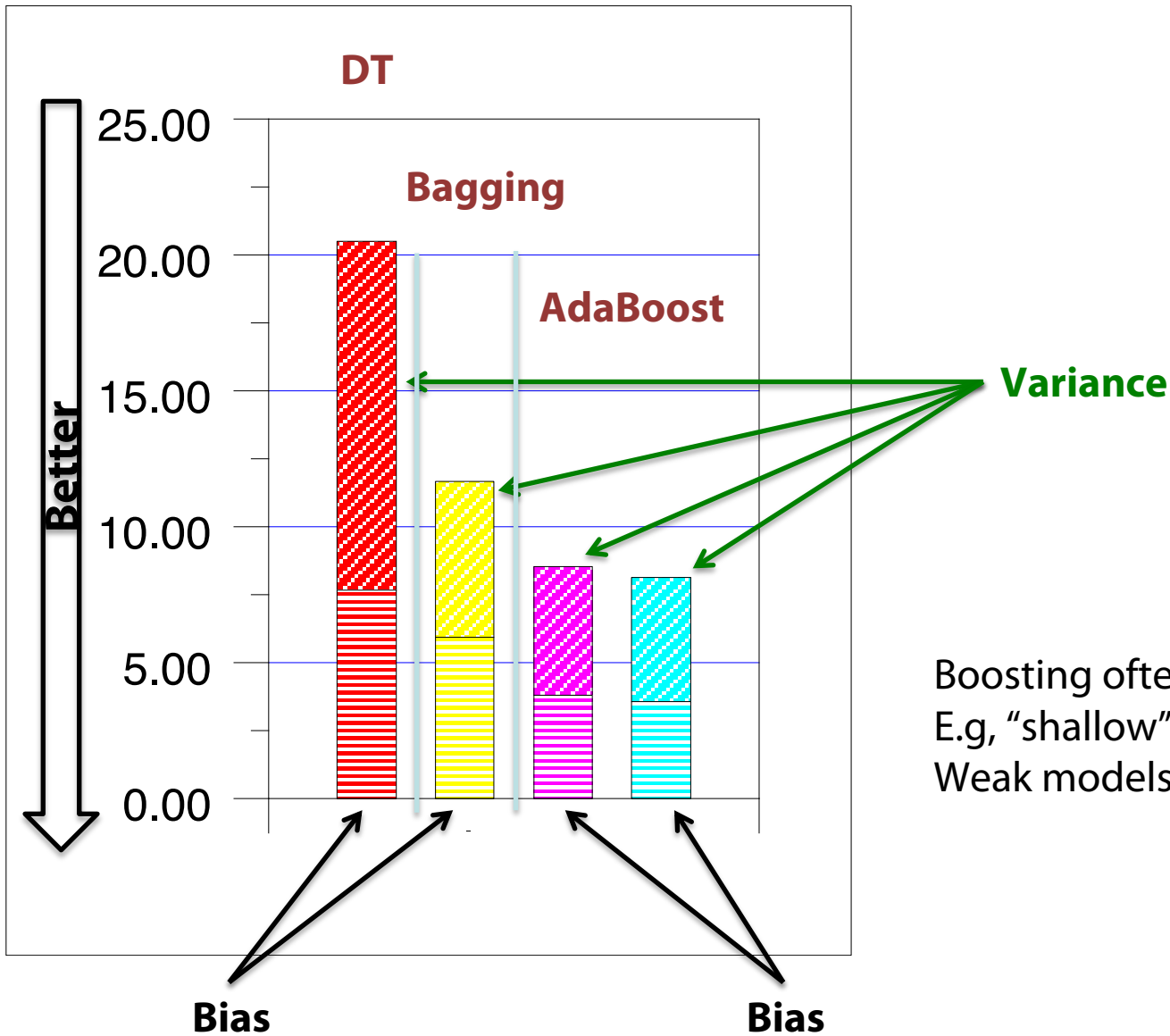
$$S' = \{(x, y, w_n)\}$$



$h_n(x)$

$w$  – weighting on data points  
 $\alpha$  – 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

# Gradient Boosting (XGBoost) Ensemble Learning

---

- The term "gradient" in gradient boosting refers to the fact that the method is based on minimizing the gradient of the loss function
  - Fit each new model to the gradient of the loss function with respect to the ensemble's predictions
  - By iteratively minimizing the loss function's gradient, the ensemble of models is improved, leading to the term "gradient boosting"
- XGBoost: popular implementation of gradient boosting
  - Parallelization
  - Regularization (penalty on the complexity of the loss function to prevent overfitting)
  - Captures non-linear relationships between features and the target variable
- Gradient boosting has been widely used in machine learning and has been a key component in many winning models of Kaggle competitions



# Bagging vs Boosting

---

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

# Ensemble Selection

Person	Age	Male?	Height > 5'5"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Elin	11	0	0
Frank	9	1	1
Genia	8	0	0

S

Person	Age	Male?	Height > 5'5"
Alice	14	0	1
Bob	10	1	1
Carol	13	0	1
Dave	8	1	0
Elin	11	0	0
Frank	9	1	1
Genia	8	0	0

Training  $S'$

Validation  $V'$

$H = \{2000 \text{ models trained using } S'\}$

Maintain ensemble model as combination of  $H$ :

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

Denote as  $h_{n+1}$

Add model from  $H$  that maximizes performance on  $V'$

Repeat

Models are trained on  $S'$   
Ensemble built to optimize  $V'$

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.

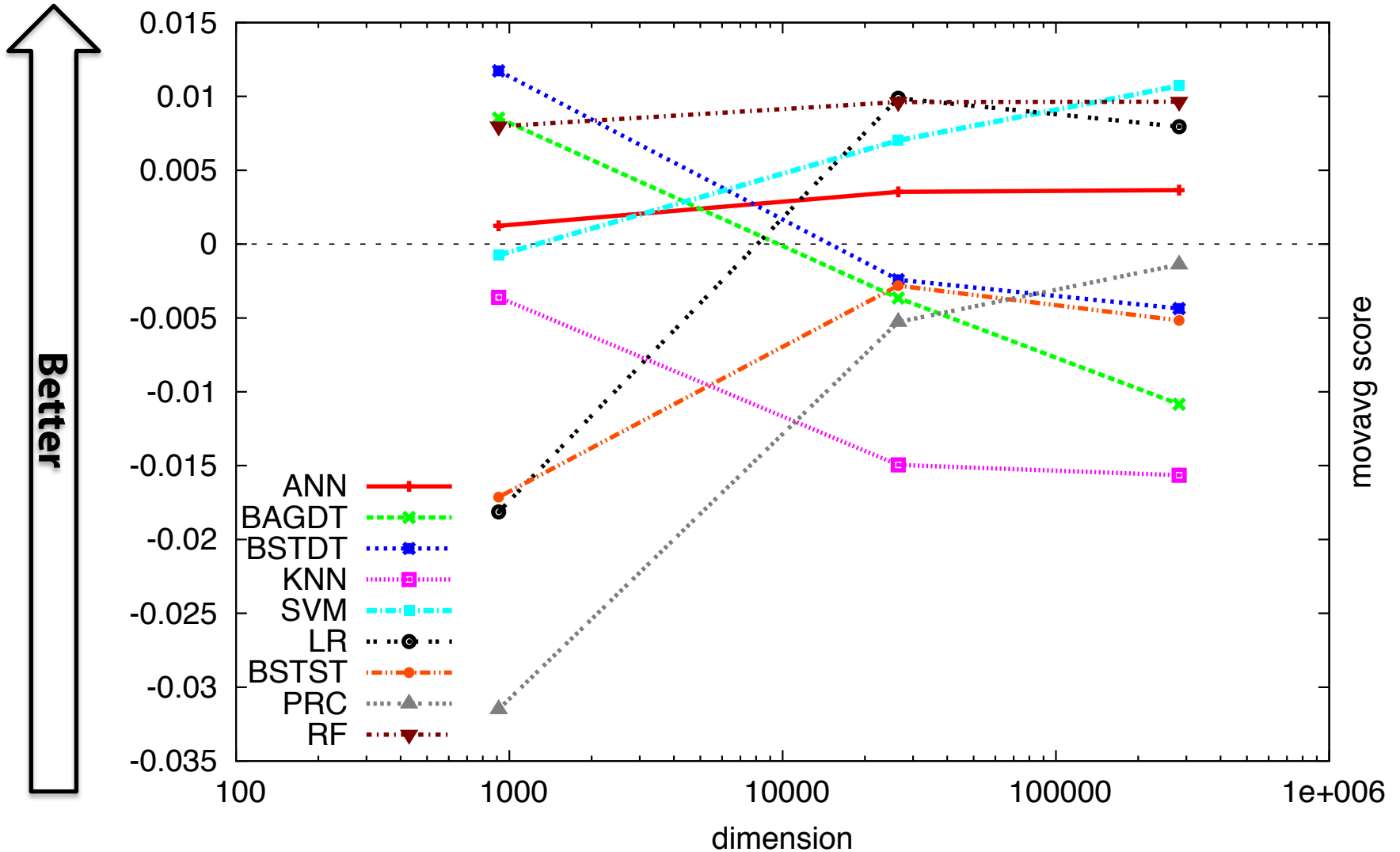
...and many other ensemble methods as well.

- **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

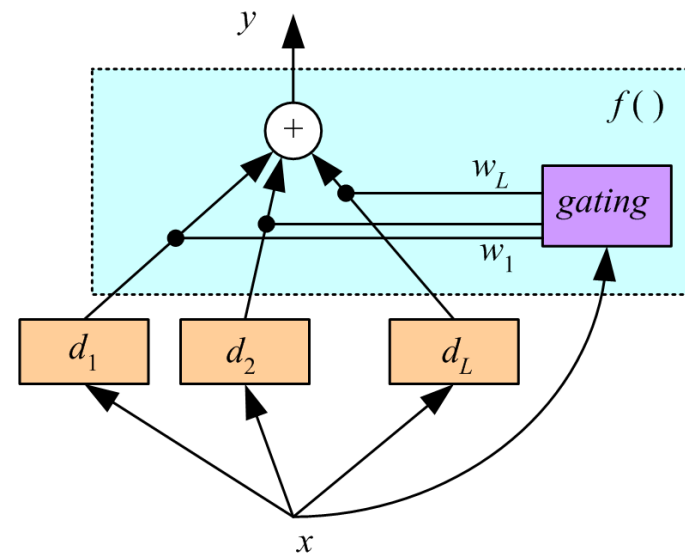
# Mixture of Experts: Gating

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

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

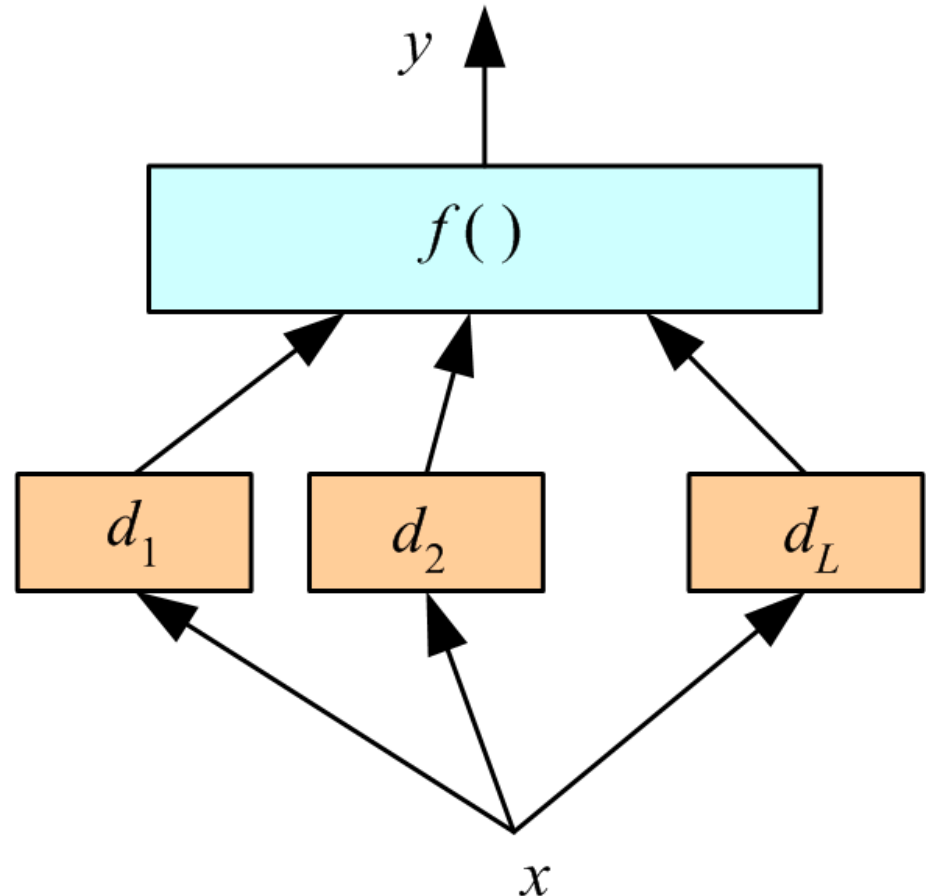
- Gating decides which expert to use
- Need to learn the individual experts as well as the gating functions  $w_i(x)$ :

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



# Mixture of Experts: Stacking

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



# Mixture of Experts: Cascading

Use  $d_j$  only if preceding ones are not confident

Cascade learners in order of complexity

