Ensemble Learning

Zhiyao Duan
Associate Professor of ECE and CS
University of Rochester

Some figures are copied from the following book

- **LWLS** - Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, Thomas B. Schön, *Machine Learning: A First Course for Engineers and Scientists*, Cambridge University Press, 2022.

Boosting part is adapted from Robert Schapire's tutorial in 2005.

What is Ensemble Learning?

- Key idea
  - Building a highly accurate model (e.g., classification, regression) is difficult
  - Building many not-so-accurate models is easy
  - Can we generate a single, highly accurate model from these not-so-accurate models?

- Answer: Yes
  - “Two heads are better than one”
  - “三个臭皮匠，赛过诸葛亮” (Three Stooges, the top of *ZHUGE Liang, the mastermind*.)

General Steps

- 1. Train a number of weak models from training data
- 2. Each model predicts on test data
- 3. Combine these predictions as the final prediction on test data

- Questions:
  - How to train these models?
  - How to combine their predictions?
  - Why is the combined prediction more accurate?

Bagging

- Bagging = **B**ootstrap **agg**regat**ing**
- [Breiman, 1996]
  - Leo Breiman, “Bagging predictors,” *Machine Learning*, 1996.

- Bootstrapping: “pull oneself over a fence by one's bootstraps”

- In statistics, boostrapping means “estimating properties of an estimand (e.g., variance) by measuring those properties when sampling from an approximation distribution.” ---- Wikipedia

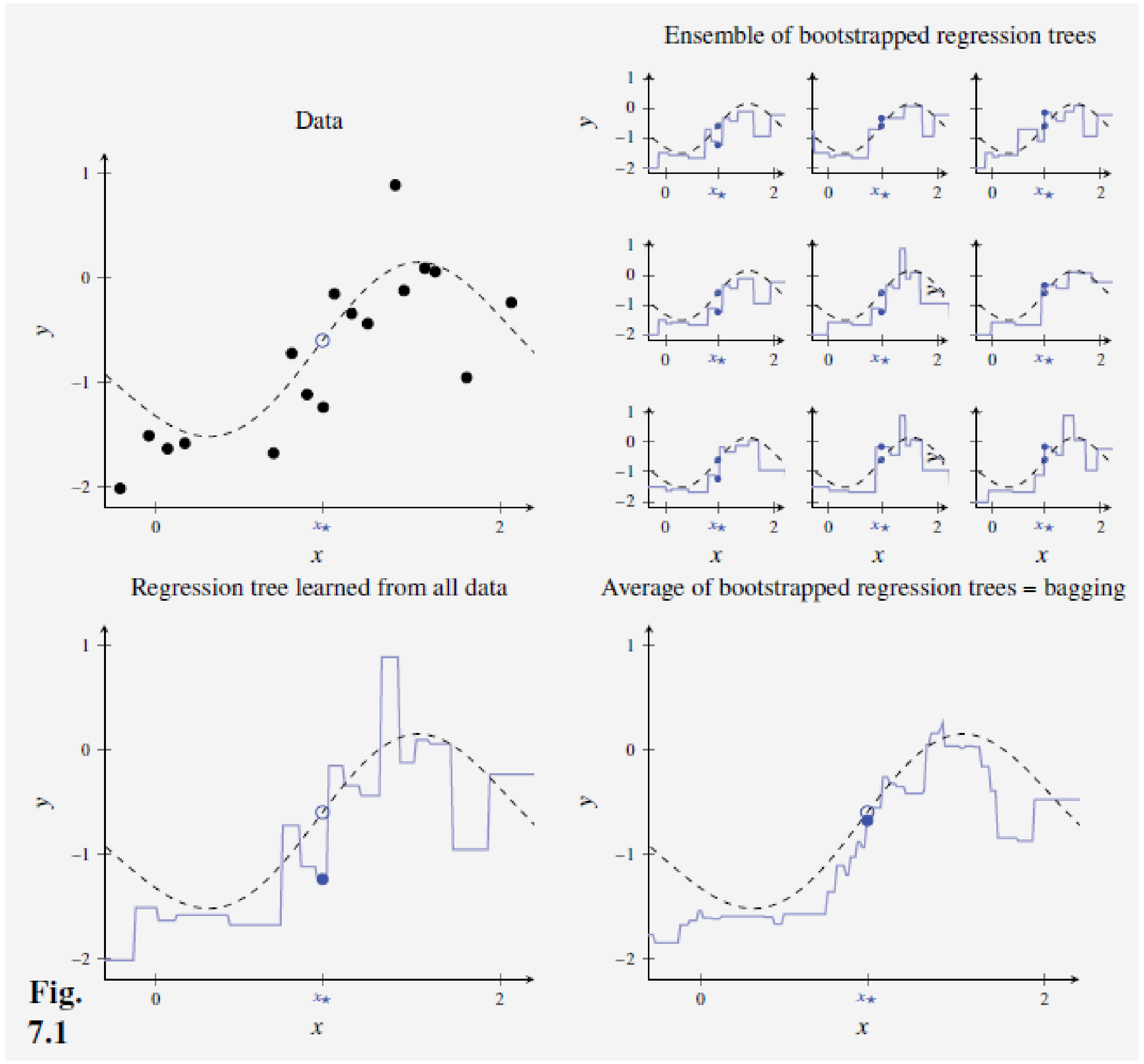
Bagging

- Given $\mathcal{L} = \{N$ training instances$\}$
- For $i = 1$ to T
  - Sample N instances with replacement from $\mathcal{L}$ to form a new training set $\mathcal{L}_i$
  - Train a model using $\mathcal{L}_i$
- For a test instance, combine predictions of the T models as the final prediction, e.g.,
  - Majority vote for classification
  - Average for regression

- Note: the T models are independently constructed
- Question: What if we sample without replacement?

Bagging Example

- Regression trees in 1D
- Each tree makes a prediction on the test example
- The final prediction of bagging is the average



(Fig. 7.1 in LWLS)

Averaging Reduces Variance

- Let $\{z_t\}_{t=1,\cdots,T}$ be a collection of identically distributed (possibly dependent) random variables, with $\mathbb{E}(z_t) = \mu$ and $Var(z_t) = \mathbb{E}[(z_t - \mu)^2] = \sigma^2$
- Assume the average correlation between any pair is ρ, i.e., $\mathbb{E}[(z_s - \mu)(z_t - \mu)] = \rho\sigma^2$ if $s \neq t$
- Then we have

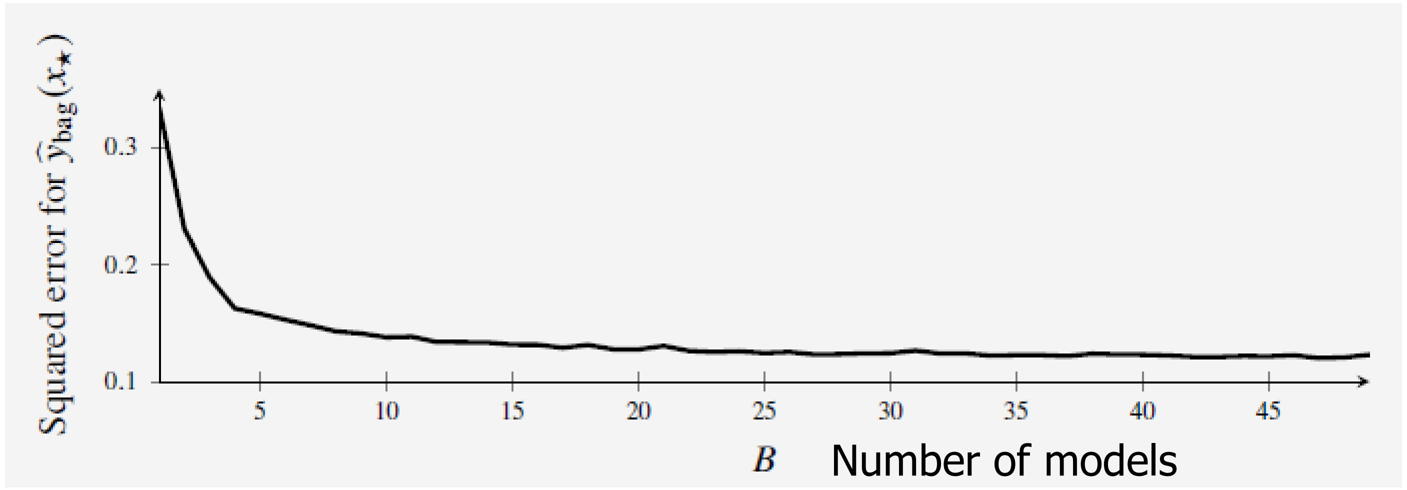
$$\mathbb{E}\left(\frac{1}{T}\sum_{t=1}^{T} z_t\right) = \mu$$

$$Var\left(\frac{1}{T}\sum_{t=1}^{T} z_t\right) = \mathbb{E}\left[\left(\frac{1}{T}\sum_{t=1}^{T} z_t - \mu\right)^2\right] = \frac{1}{T^2}\mathbb{E}\left[\left(\sum_{t=1}^{T}(z_t - \mu)\right)^2\right] = \frac{1-\rho}{T}\sigma^2 + \rho\sigma^2$$

- If $\rho < 1$, then increasing T decreases variance!

Bagging Reduces Model Variance

- Bagging for regression is averaging predictions made by different models
- These models are trained on different samples of the training set
  - Model predictions on the same test data instance can be viewed as random variables z_t, and they are identically distributed
  - Averaging these predictions reduces the variance
  - More models → lower variance but higher computational cost



(Fig. 7.4 in LWLS)

Out-of-Bag Error Estimation

- This is a method to estimate E_{new} of bagging without cross validation

- Due to sampling with replacement, each model is only trained on a subset (on average 63%) of the original training set
- Equivalently, each training example is not seen by about 1/3 of all models
  - If we form an ensemble using these models, then this training example can be used as an unseen test example for this ensemble, i.e., it is "out of bag"
  - We can compute its error $E_{OOB}^{(i)}$
- We can average this error over all training examples $E_{OOB} = \frac{1}{N}\Sigma_{i=1}^{N} E_{OOB}^{(i)}$
- E_{OOB} is a good estimate of E_{new} for an ensemble with only $\frac{T}{3}$ models
  - If T is large enough, this is also a good estimate of the original ensemble with T models

Random Subspace

- [Ho, 1998]
  - Tin Kam Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.
- Create the training set in each round by randomly choosing a subset of all attributes, i.e., using a random subspace of the original feature space
- Train a decision tree using this training set
- Combine predictions of these trees on test data

Random Forests

- [Breiman ‘01]
  - Leo Breiman, “Random forests,” *Machine Learning*, 2001.

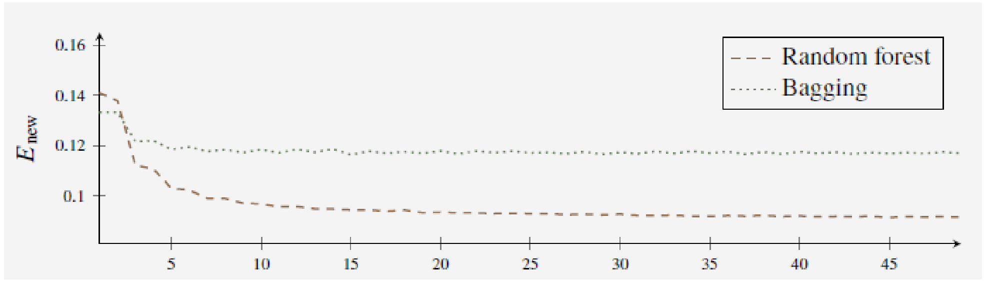
- Use decision tree as the weak model
- Generate a number of trees
  - For each tree, randomly sample a subset of training examples with replacement
  - For each tree, randomly select a subset of features to determine splitting at each node
- Combine predictions of all the trees

- This combines the bagging idea (randomly sampling data) and the random subspace idea (randomly sampling features)
  - Further reduces correlation between models

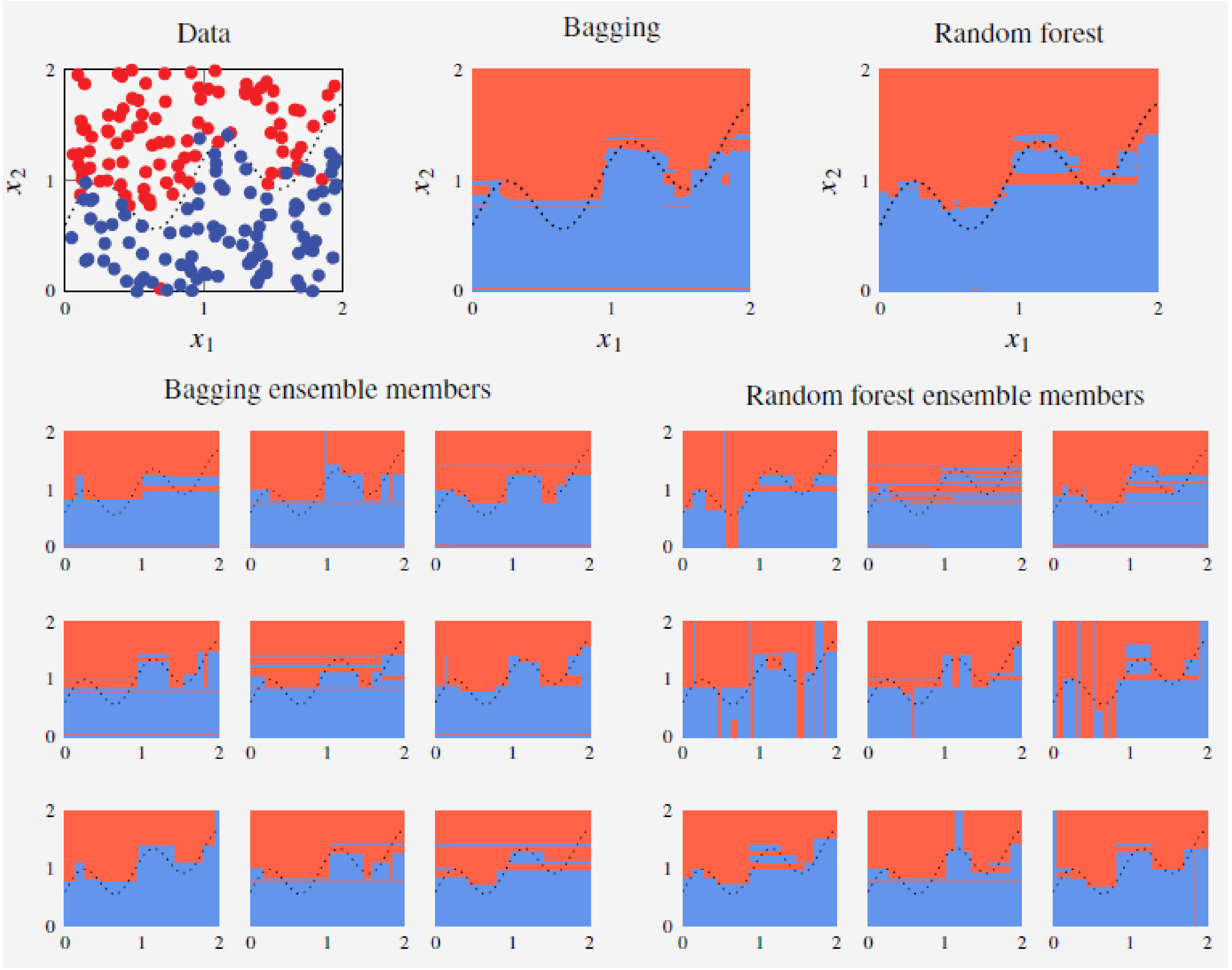
Random Forests Example

- Binary classification in 2D
  - 1 feature is randomly selected at each node of each tree

- Random forest ensemble members show larger variation



(Fig. 7.5 in LWLS)



Boosting

- Construct a classifier using a weak learning algorithm based on previous classifiers
  - Create a training set which weights more on the "hardest" examples (those most often misclassified by previous classifiers)
  - Combine classifiers by weighted majority vote, putting more weights on accurate classifiers

- Assumptions:
  - The weak learning algorithm can consistently find classifier with error $\leq 1/2-\gamma$

- Conclusion:
  - A boosting algorithm can provably construct a single classifier with arbitrarily small error

A Formal View of Boosting

- Given training set $L=\{(x_1,y_1),\ldots,(x_m,y_m)\}$
- $y_i \in \{-1,+1\}$ correct label of instance x_i

- For $t = 1,\ldots,T$:
  - construct a distribution D_t on $\{1,\ldots,m\}$
  - Find a weak classifier $f_t : X \rightarrow \{-1,+1\}$
    with small error ε_t on D_t: $$\varepsilon_t = \Pr_{D_t}[f_t(x_i) \neq y_i]$$

- Output a final classifier f_{final} that combines the weak classifiers in a good way

AdaBoost [Freund & Schapire '95]

- Constructing D_t:
  - $D_1(i) = 1/m$ ← Size of the training set
  - given D_t and f_t:

Correct label → y_i ; Predicted label → $f_t(x_i)$

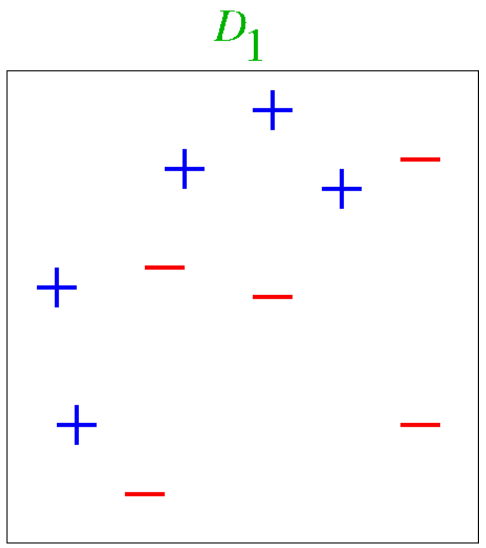
$$D_{t+1}(i) = \underbrace{\frac{D_t(i)}{Z_t}}_{\text{Normalization factor}} \cdot \begin{cases} e^{-\alpha_t} & \text{if } y_i = f_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq f_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \cdot \exp(-\alpha_t \cdot y_i \cdot f_t(x_i))$$

where $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$

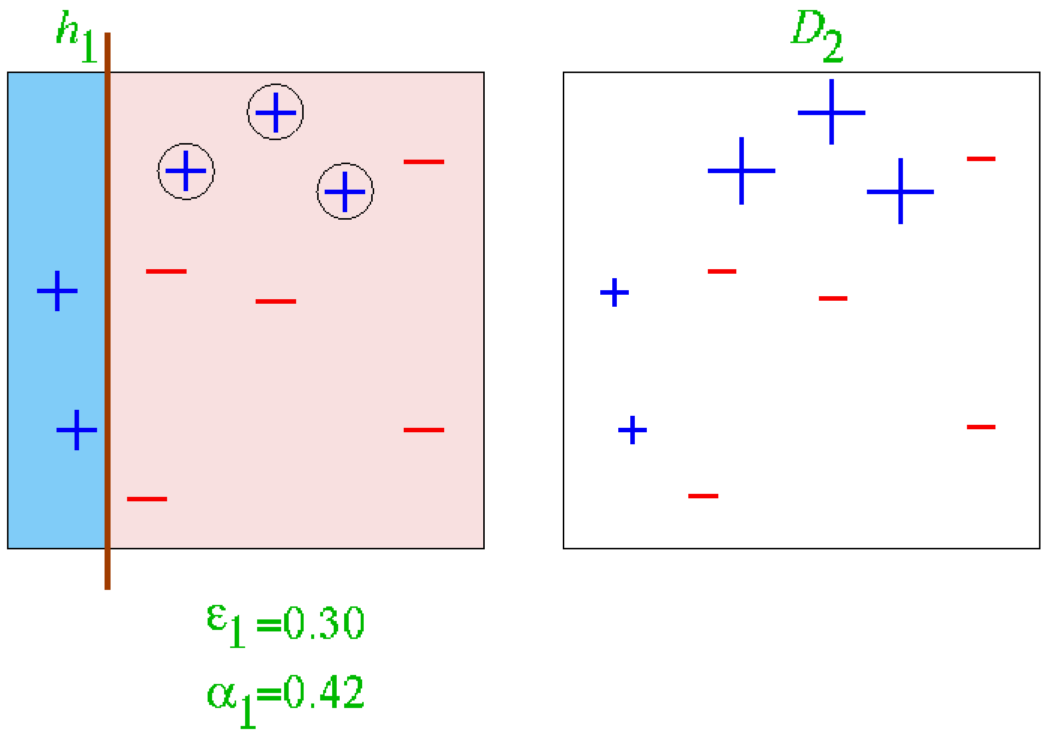
- final classifier: $f_{\text{final}}(x) = \operatorname{sgn}\left(\sum_t \alpha_t f_t(x)\right)$

Toy Example

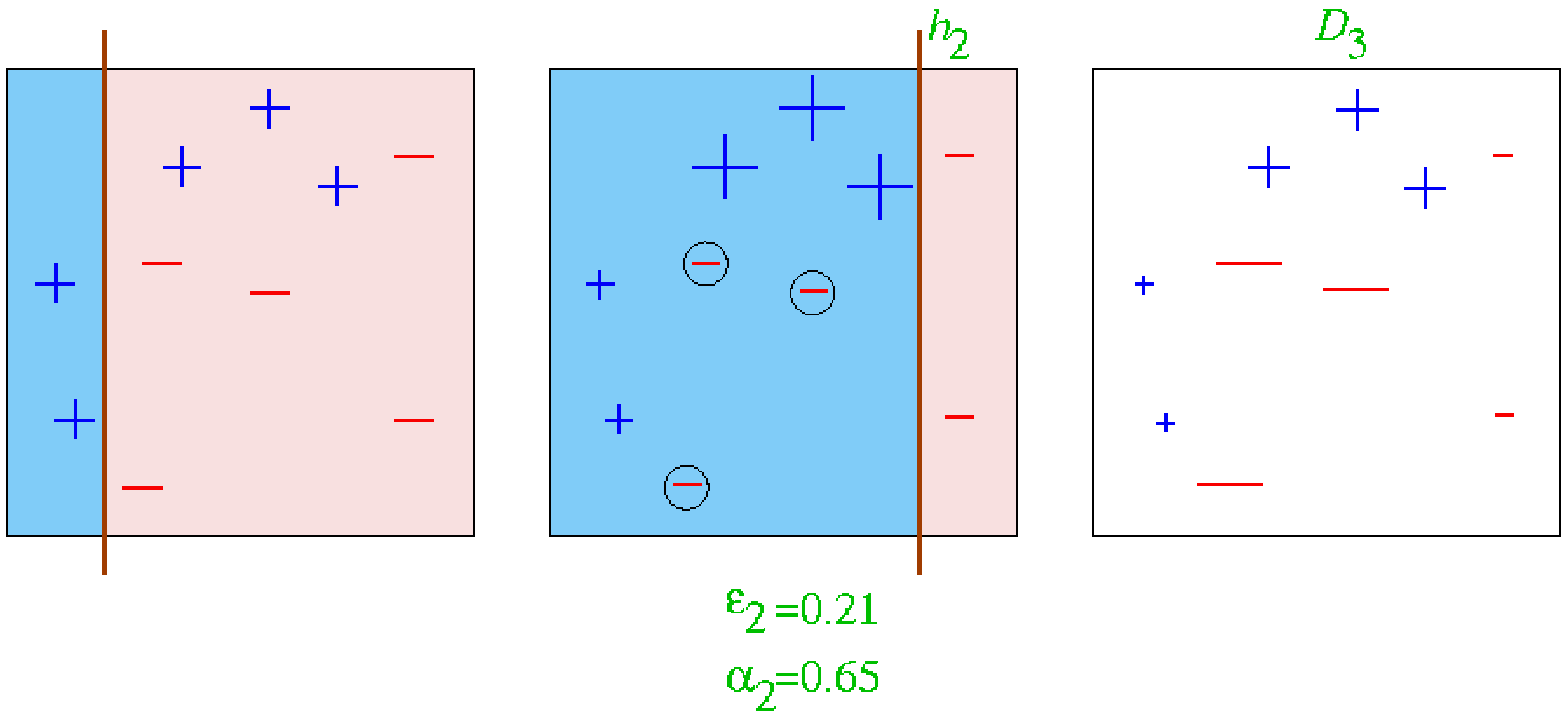


Weak classifiers: vertical or horizontal half planes

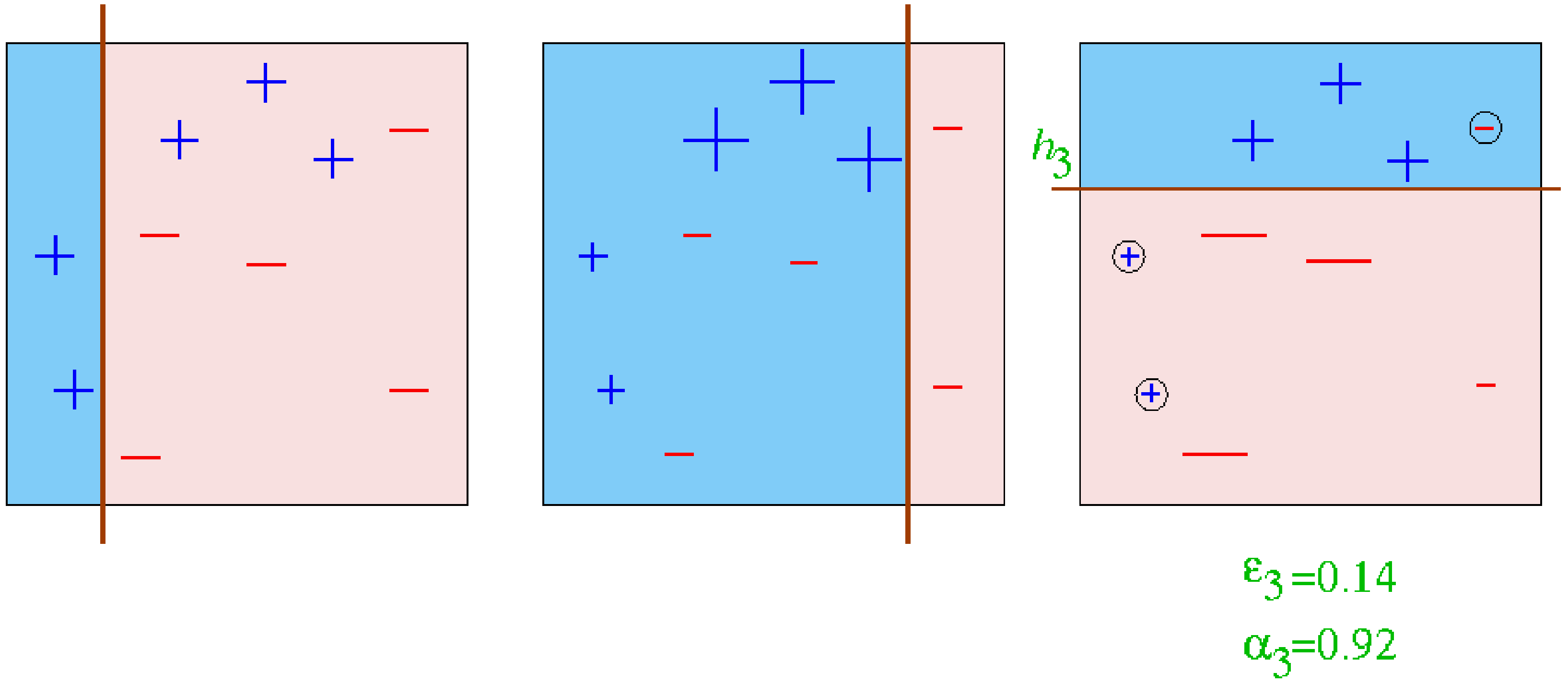
Round 1



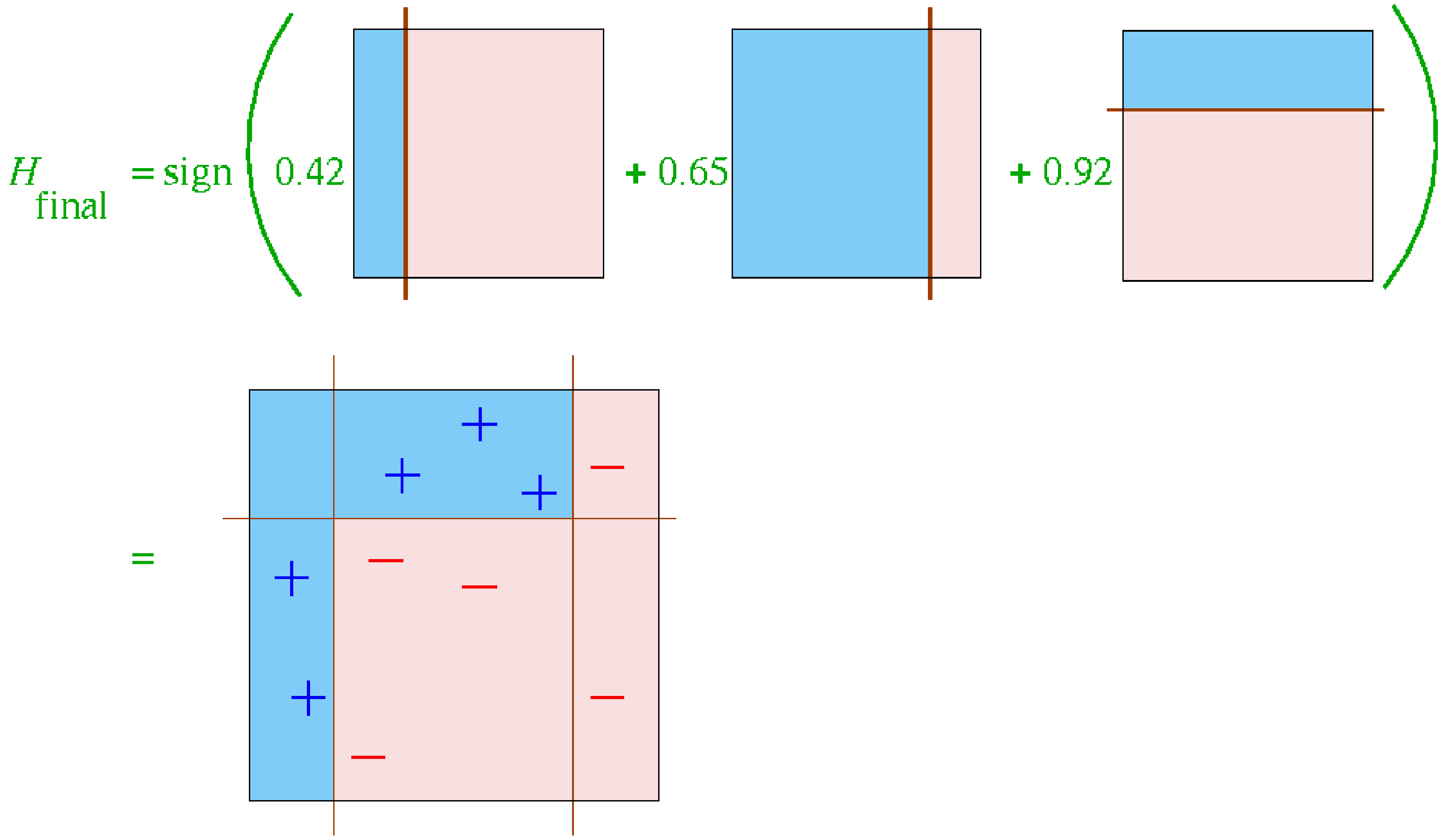
Round 2



Round 3



Final Classifier


$H_{\text{final}} = \text{sign}\left(0.42 \quad + 0.65 \quad + 0.92 \right)$
=
+
−

Analyzing the Training Error

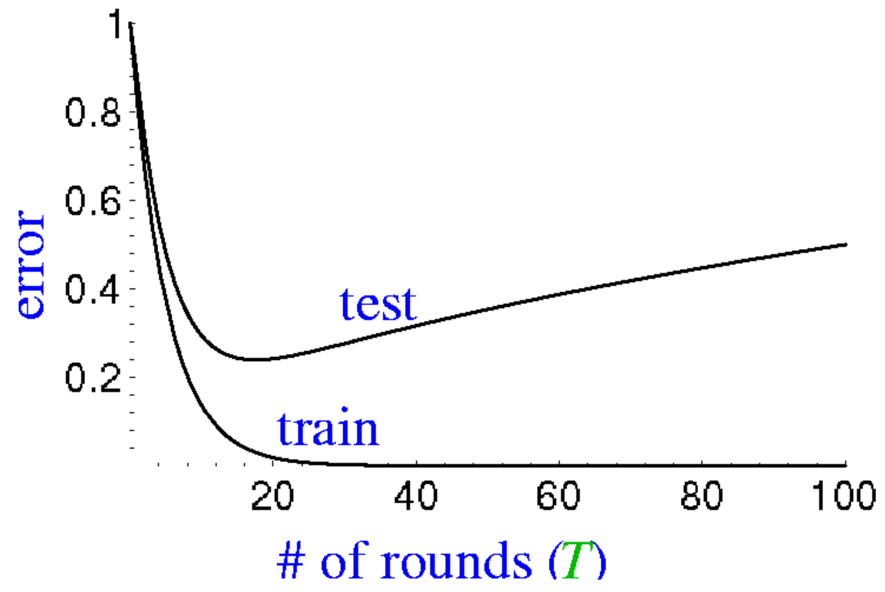
- Theorem [Freund&Schapire '97]:
  - write ε_t as ½-γ_t
  - the $\text{training error}(f_{\text{final}}) \leq \exp\left(-2\sum_t \gamma_t^2\right)$

- so if $\forall t$: $\gamma_t \geq \gamma > 0$ then

$$\text{training error}(f_{\text{final}}) \leq \exp\left(-2\gamma^2 T\right)$$

- AdaBoost is adaptive:
  - does not need to know γ or T a priori
  - can exploit $\gamma_t >> \gamma$

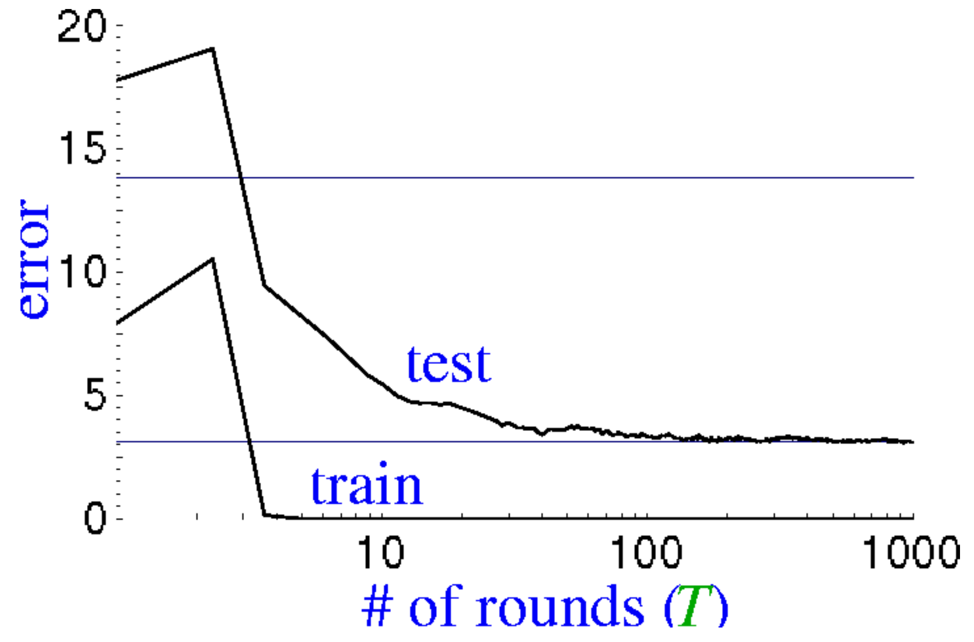
Guess the Test Error



We expect:

- training error to continue to drop (or reach zero)
- test error to <u>increase</u> when f_{final} becomes “too complex” (Occam’s razor)

A Typical Run



(boosting on C4.5 on "letter" dataset)

- Test error does not increase even after 1,000 rounds (~2,000,000 nodes)
- Test error continues to drop after training error is zero!
- Occam's razor wrongly predicts "simpler" rule is better.

A Better Story: Margins

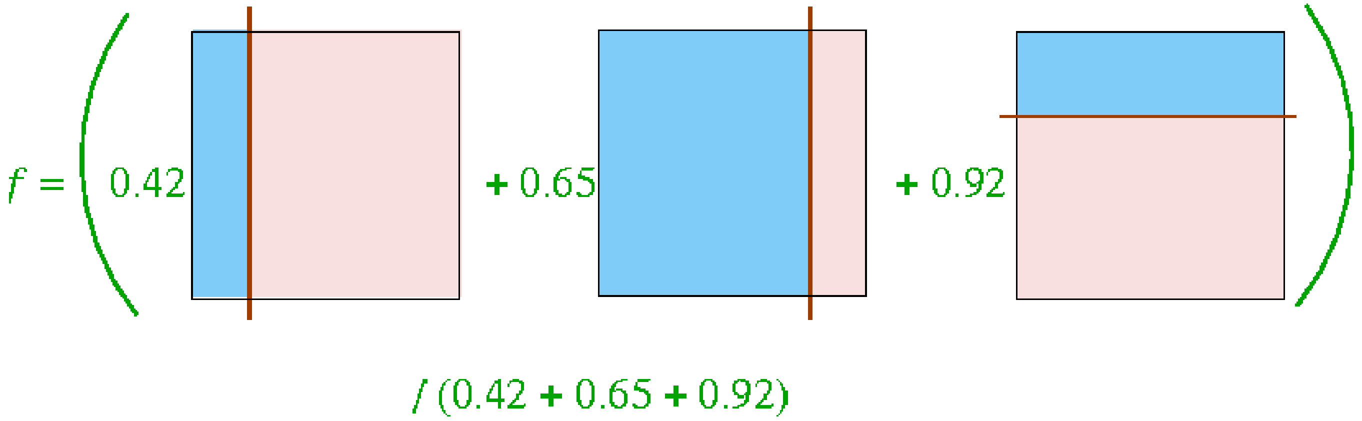
- Key idea:
  - training error only measures whether classifications are right or wrong
  - should also consider confidence of classifications

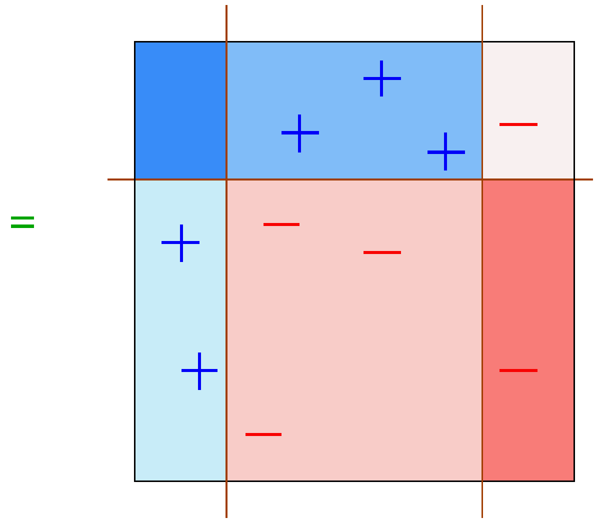
- Consider <u>confidence</u> (margin):

$$f_{\text{final}}(x) = \text{sgn}(f(x)) \qquad f(x) = \frac{\sum_t \alpha_t f_t(x)}{\sum_t \alpha_t} \in [-1,1]$$

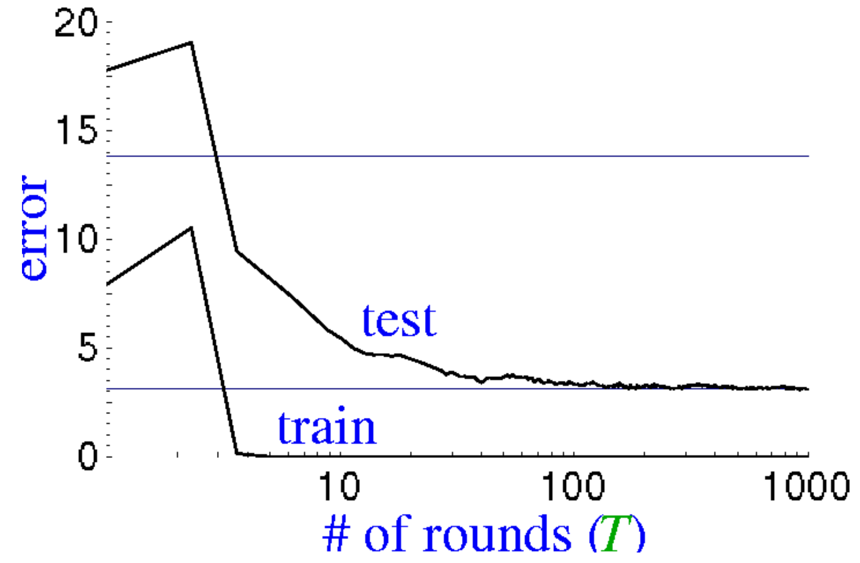
- Define: <u>margin</u> of $(x, y) = y \cdot f(x) \in [-1,1]$

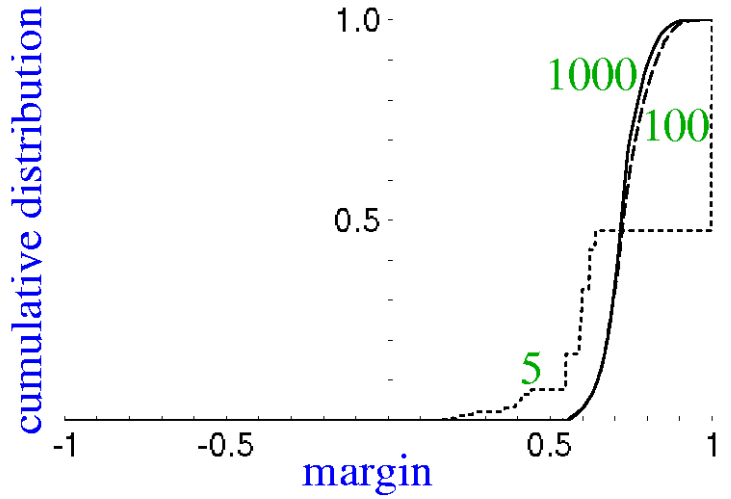
Margins for Toy Example





The Margin Distribution





rounds	5	100	1000
training error	0.0	0.0	0.0
test error	8.4	3.3	3.1
%margins≤0.5	7.7	0.0	0.0
Minimum margin	0.14	0.52	0.55

Analyzing Boosting Using Margins

- Theorem: boosting tends to increase margins of training examples

- Theorem: large margins => better bounds on generalization error (independent of number of rounds)
  - Proof idea: if all margins are large, then the final classifier can be approximated by a much simpler classifier

- Consequence: although the final classifier gets larger, margins are likely to increase too, making the final classifier actually become closer to a simpler classifier, driving down the test error

Practical Advantages of AdaBoost

- Simple + easy to program
- Flexible: can be combined with any classifier (neural nets, C4.5, …)
- Only a single parameter to tune (T)
- No prior knowledge
- Provably effective (assuming weak learner)

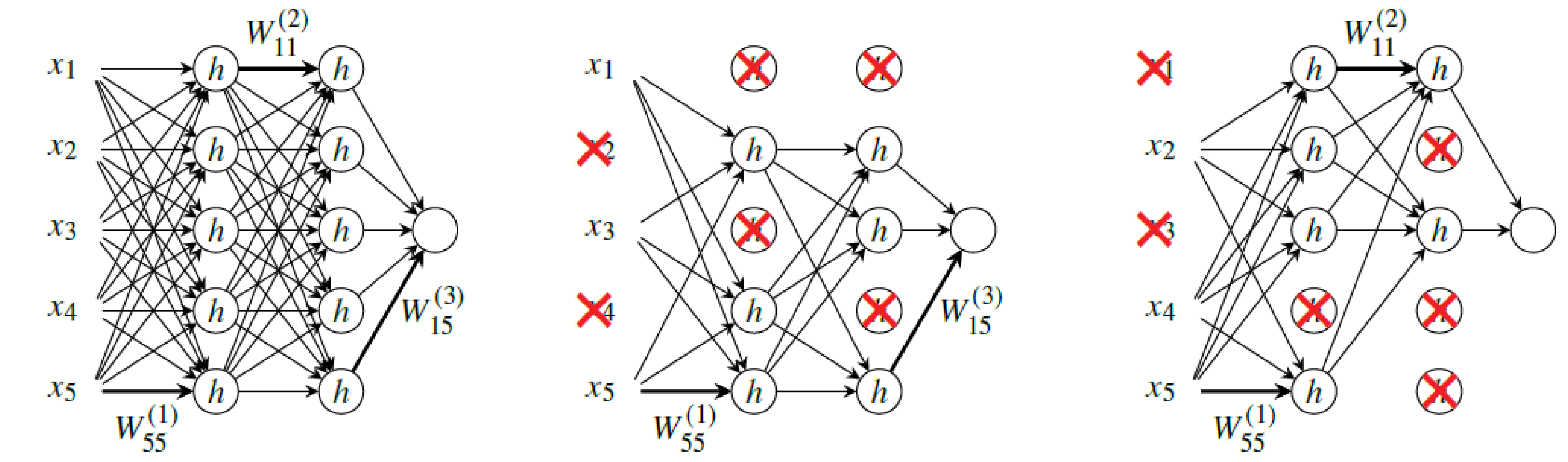
Cons

- AdaBoost can fail if
  - Weak classifier is too complex (overfitting)
  - Weak classifier is too weak ($\gamma_t \rightarrow 0$ too quickly),

- Empirically, AdaBoost seems especially susceptible to noise

Dropout

- An important technique to alleviate overfitting
- Randomly (with probability 1-r) dropout some neurons/filters in each iteration of training
  - They do not participate in either forward computation or backpropagation
- During inference (i.e., predicting on unseen data), multiple network weights with r
- Conceptually, the learned model is like an ensemble of networks that share some weights
- Practically very effective; theoretically unclear why



(a) A standard neural network (Fig. 6.18 in LWLS) **(b)** Two sub-networks

Summary

- Ensemble learning: combine weak models to obtain a strong model
- Bagging: sample training data with replacement into multiple training sets to train multiple models; combine models through majority vote or averaging
  - Reduces model variance
  - Random subspace: sample features
  - Random forests: sample training data and features, using decision trees as base models
- Boosting: construct models sequentially, weighting difficult data more; combine models with weighted average according to their performance
  - AdaBoost
    - Error on the training set can be arbitrarily small (given enough data and enough rounds)
    - Often resistant to overfitting
    - Margins are increased with more rounds
    - Suspicious to noise
- Dropout for neural networks: ensemble of networks that share some weights

Summary

- Clustering (e.g., assigning data points to different clusters) is an unsupervised learning problem

- Centroid-based
  - K-means
    - Optimizes the intra-cluster squared distance objective
    - Converges to local minimum
    - Initialization
    - How to choose K
- Density-based
  - DBSCAN
- Hierarchical Agglomerative Clustering