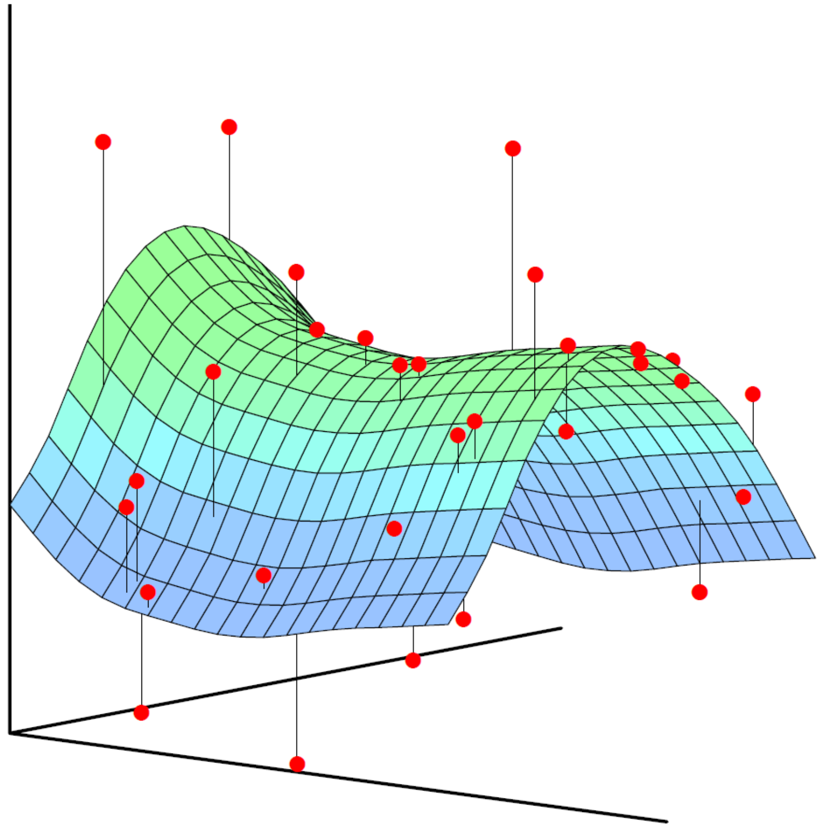
Machine Learning

第9讲 集成学习
Ensemble Learning



刘 峤

电子科技大学计算机科学与工程学院

9.1 Introduction to Ensemble learning

Ensemble learning

- In statistics and machine learning,

  ※ ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

- The term **ensemble** is usually reserved for methods that generate multiple hypotheses using the same base learner.

  ※ The broader term of multiple classifier systems also covers hybridization of hypotheses that are not induced by the same base learner.

Ensemble theory

- An ensemble is itself a supervised learning algorithm
- The trained ensemble represents a single hypothesis
  ※ It is not necessarily contained within the hypothesis space of the models from which it is built.
- Empirically, ensembles tend to yield better results when there is a significant diversity among the models.
  ※ Many ensemble methods, therefore, seek to promote diversity among the models they combine.
  ※ Although perhaps non-intuitive, more random algorithms (like random decision trees) can be used to produce a stronger ensemble than very deliberate algorithms (like entropy-reducing decision trees)

Multiple classifier systems

- Multiple classifier systems: approaches to combine several machine learning techniques into one predictive model in order to
  - ※ decrease the variance (bagging)
  - ※ decrease the bias (boosting) or
  - ※ improving the predictive force (stacking)
- Why multiple classifier systems ?
  - ※ The main causes of error in learning are due to noise, bias and variance. meta-algorithms helps to minimize these factors--improve the stability and the accuracy of Machine Learning algorithms.

Bias/Variance Decomposition

- Squared loss of model on test case i:

$$[\text{Learner}(x_i, \mathcal{D}) - \text{Truth}(x_i)]^2$$

- Squared loss of model on test case i:

$$E\left\{[\text{Learner}(x_i, \mathcal{D}) - \text{Truth}(x_i)]^2\right\}$$
$$= \text{Noise}^2 + \text{Bias}^2 + \text{Variance}$$

- $\text{Noise}^2 =$ lower bound on performance
- $\text{Bias}^2 =$ (expected error due to model mismatch)²
- $\text{Variance} =$ variation due to train sample and randomization

Sources of “variance” in Supervised Learning

- noise in targets or input attributes
- bias (model mismatch)
- training sample
- randomness in learning algorithm
  - ※ Eg. neural net weight initialization
- randomized subsetting of train set
  - ※ Eg. cross validation, train and early stopping set

9.2 Blending

Aggregation Models

- **Aggregation**: combine hypotheses for better performance
  - ※ 例如：若我们有H个学习器可用于股票价格涨跌的预测
  - ※ 策略1：选择性能表现最好的学习器
  - ※ 策略2：让H个学习器进行无差别投票
  - ※ 策略3：投票时给不同的学习器不同的权重
  - ※ 策略4：有条件地combine各学习器的预测结果
    - ➢ 若 H_i 满足某些特定条件，则赋予其较多的投票权
- 问：这几种实际工作中常用的策略有何关联？对其进行形式化

Aggregation with Math Notations

- T个学习器：$g_1(x), \cdots, g_T(x)$

※ 策略1：选择性能表现最好的学习器

$$G(\mathbf{x}) = g_{t^*}(\mathbf{x}) \text{ with } t^* = \underset{t \in \{1,2,\ldots,T\}}{argmin} \; E_{val}(g_t(\mathbf{x}'))$$

※ 策略2：让T个学习器进行无差别投票

$$G(\mathbf{x}) = sign\left(\sum_{t=1}^{T} 1 \cdot g_t(\mathbf{x})\right)$$

※ 策略3：投票时给不同的学习器不同的权重

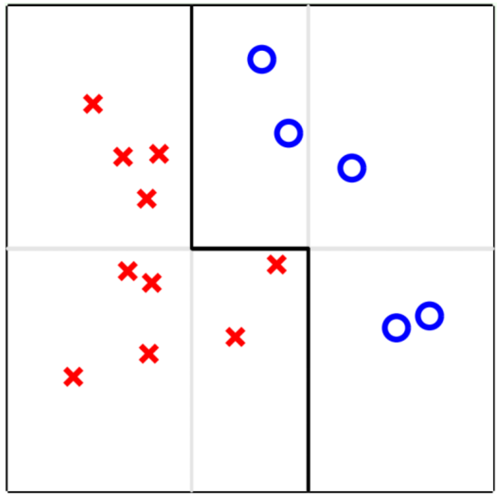
$$G(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t \cdot g_t(\mathbf{x})\right) \text{ with } \alpha_t \geq 0$$

※ 策略4：有条件地combine各学习器的预测结果

$$G(\mathbf{x}) = sign\left(\sum_{t=1}^{N} q_t(\mathbf{x}) \cdot g_t(\mathbf{x})\right) \text{ with } q_t(\mathbf{x}) \geq 0$$

Why Might Aggregation Work?

- **aggregation**: can we do better with many (possibly weaker) hypotheses?

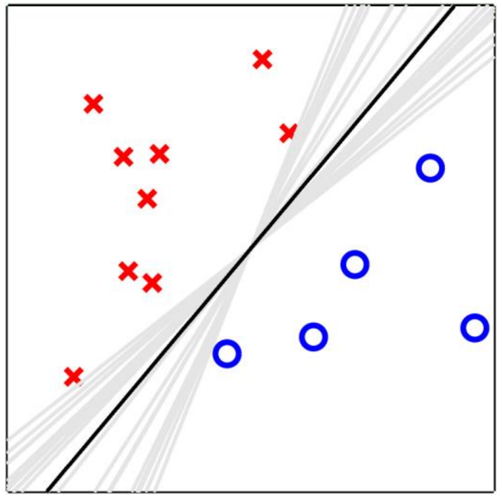


- mix different weak hypotheses uniformly

  ※ G(x) will be **stronger**

  ※ feature transform?

proper aggregation → better performance



- mix different random-PLA hypotheses uniformly

  ※ G(x) will be **moderate**

  ※ Regularization?

Quize

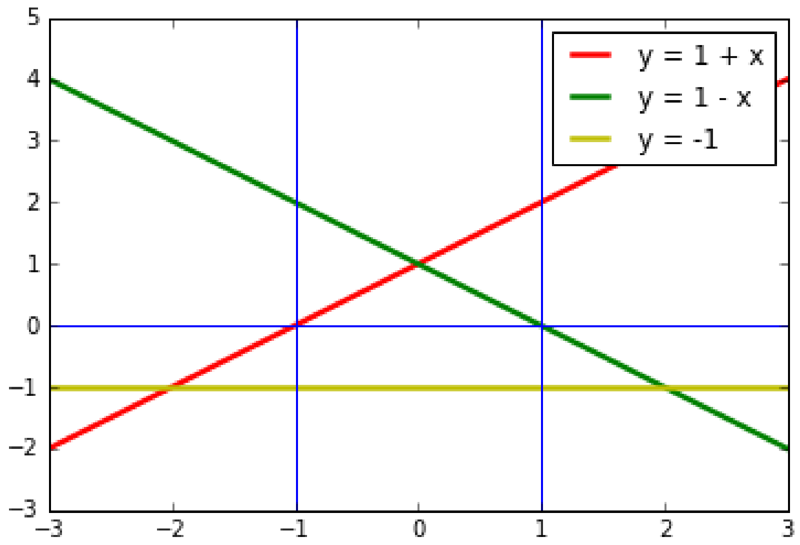
- Consider three decision stump hypotheses from R to {-1,+1}:

$$g_1(x) = \text{sign}(1 - x),\ \ g_2(x) = \text{sign}(1 + x),\ \ g_3(X) = -1.$$

- When mixing the three hypotheses uniformly, what is the resulting G(x)?

(A) $2[\![|x| \le 1]\!] - 1$ (B) $2[\![|x| \ge 1]\!] - 1$

(C) $2[\![x \le -1]\!] - 1$ (D) $2[\![x \ge +1]\!] - 1$



The region that gets two positive votes from g1 and g2 is |x| <= 1, and thus G(x) is positive within the region only. We see that the three decision stumps g_t can be aggregated to form a more sophisticated hypothesis G .

Reference Answer: 1

9.2.1 Uniform Blending

Uniform Blending for Classification/Regression

- Uniform Blending: known $g_t(x)$, each with 1 ballot

$$G(\mathbf{x}) = sign\left(\sum_{t=1}^{T} 1 \cdot g_t(\mathbf{x})\right)$$

- Uniform Blending for Regression

$$G(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} g_t(\mathbf{x})$$

- Diverse hypotheses:

  ※ Empirically, ensembles tend to yield better results when there is a significant diversity among the models

  ※ Uniform blending can be better than any single hypothesis

Theoretical Analysis of Uniform Blending

- Uniform Blending for Regression $G(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} g_t(\mathbf{x})$

$$\begin{aligned}
\text{avg}\{(g_t(\mathbf{x}) - f(\mathbf{x}))^2\} &= \text{avg}\left(g_t^2 - 2g_t f + f^2\right) \\
&= \text{avg}\left(g_t^2\right) - 2Gf + f^2 \\
&= \text{avg}(g_t^2) - G^2 + (G - f)^2 \\
&= \text{avg}(g_t^2) - 2G^2 + G^2 + (G - f)^2 \\
&= \text{avg}(g_t^2 - 2g_t G + G^2) + (G - f)^2 \\
&= \text{avg}\{(g_t - G)^2\} + (G - f)^2
\end{aligned}$$

$$\text{avg}\{E_{out}(g_t)\} = \text{avg}\{\mathbf{E}(g_t - G)^2\} + E_{out}(G) \geq E_{out}(G)$$

Some Special g_t

- consider a virtual iterative process that for $t = 1, 2, \ldots, T$
  - ※ request size-N data $\mathcal{D}_t$ from P^N (i.i.d.)
  - ※ obtain g_t by $\mathcal{A}(\mathcal{D}_t)$

$$\bar{g} = \lim_{T\to\infty} G = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} g_t = \mathop{\mathbf{E}}_{\mathcal{D}}\{\mathcal{A}(\mathcal{D})\}$$

$$\text{avg}\{E_{out}(g_t)\} = \text{avg}\{\mathbf{E}(g_t - \bar{g})^2\} + E_{out}(\bar{g})$$

- expected performance of A = expected deviation to consensus (variance)

  + performance of consensus (bias)

uniform blending: reduces variance for more stable performance

Quize

- Consider applying uniform blending : $G(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} g_t(\mathbf{x})$

※ on linear regression hypotheses : $g_t(\mathbf{x}) = \mathbf{w}_t \cdot \mathbf{x}$

※ Which of the following property best describes the resulting G(x)?

① a constant function of x

② a linear function of x

③ a quadratic function of x

④ none of the other choices

$$G(\mathbf{x}) = \left(\frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t\right) \cdot \mathbf{x}$$

Reference Answer: 2

9.2.2 Linear Blending

Linear Blending

- Linear Blending: known $g_t(x)$, each to be given α_t ballot

$$G(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t \cdot g_t(\mathbf{x})\right) \quad \text{with } \alpha_t \geq 0$$

- Computing good α_t: $\min_{\alpha_t \geq 0} E_{in}(\alpha)$

- Linear blending for regression: $\min_{\alpha_t \geq 0} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_i)\right)^2$

- Linear Regression + transformation: $\min_{\mathbf{w}_i} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{k} \mathbf{w}_j \Phi_j(\mathbf{x}_i)\right)^2$

- Linear blending = LinModel + hypotheses as transform + constraints

Constraint on α_t

- linear blending = LinModel + hypotheses as transform + constraints

$$\min_{\alpha_t \geq 0} \frac{1}{N} \sum_{i=1}^{N} err\left(y_i, \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_i)\right)$$

- Linear blending for binary classification

$$\text{if } \alpha_t < 0 \;\Rightarrow\; \alpha_t g_t(\mathbf{x}) = |\alpha_t|(-g_t(\mathbf{x}))$$

※ negative α_t for g_t $\equiv$ positive $|\alpha_t|$ for $-g_t$

- in practice, often the constraints are ignorable

※ Linear blending = LinModel + hypotheses as transform

Linear blending in action

- blending practically done with

$$E_{val} \text{ (instead of } E_{in}) + g_t \text{ from minimum } E_{train}$$

- Given : $g_1, g_2, \ldots, g_T$ from $\mathcal{D}_{train}$

※ Transform (x_i, y_i) in D_{val} to $(\mathbf{z}_i = \Phi(\mathbf{x}_i), y_i)$

※ where : $\Phi(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_T(\mathbf{x}))$

- Any Blending (Stacking) : g could be any model

※ powerful, achieves conditional blending

※ but danger of overfitting, as always :-(

Brief Summary

- blending: aggregate after getting g_t
- learning g_t for uniform aggregation: **diversity** important
- diversity by different models: $g_1 \in \mathcal{H}_1, g_2 \in \mathcal{H}_2, \ldots, g_T \in \mathcal{H}_T$
- diversity by different parameters:

  ※ Eg. gradient descent with $\eta = 0.001, 0.01, \ldots, 10$
- diversity by algorithmic randomness:

  ※ Eg. random PLA with different random seeds
- diversity by data randomness:

  ※ within-cross-validation hypotheses g_v^-

9.3 Resampling and Bagging

Cross-validation and the Bootstrap

- In the section we discuss two ***resampling*** methods

  ※ cross-validation and the bootstrap.

- These methods refit a model of interest to samples formed from the *training set*, in order to obtain *additional information* about the fitted model.

- For example, they provide estimates of test-set prediction error, and the standard deviation and bias of our parameter estimates

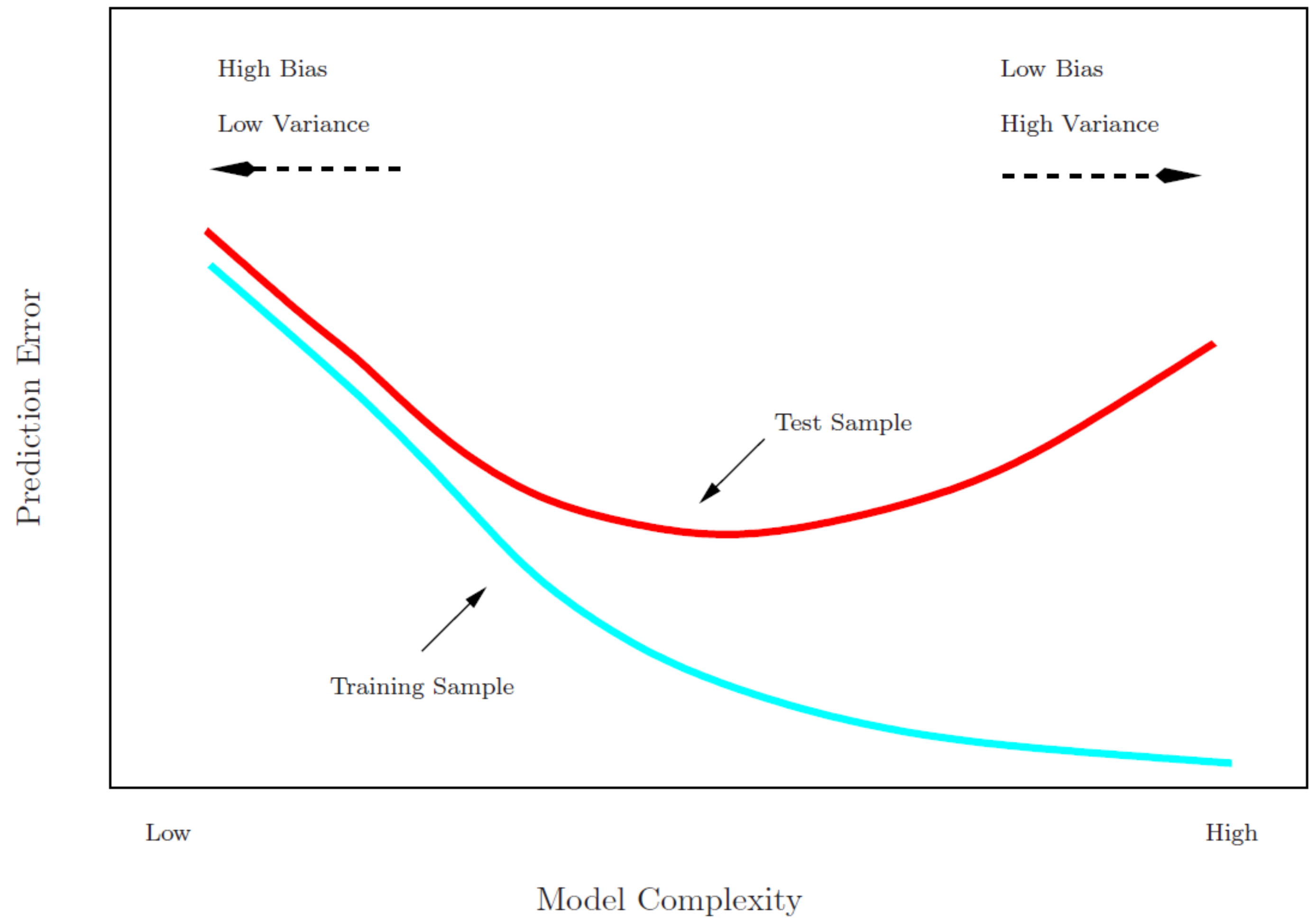
9.3.1 Cross-Validation

Training Error versus Test error

- Recall the distinction between the *test error* and the *training error*:
- The *test error* is the average error that results from using a statistical learning method to predict the response on a new observation, one that was not used in training the method.
- In contrast, the *training error* can be easily calculated by applying the statistical learning method to the observations used in its training.
- But the training error rate often is quite different from the test error rate, and in particular the former can dramatically *underestimate* the latter.

Training- versus Test-Set Performance

- $\text{Bias}^2 + \text{Variance}$ is what counts for prediction

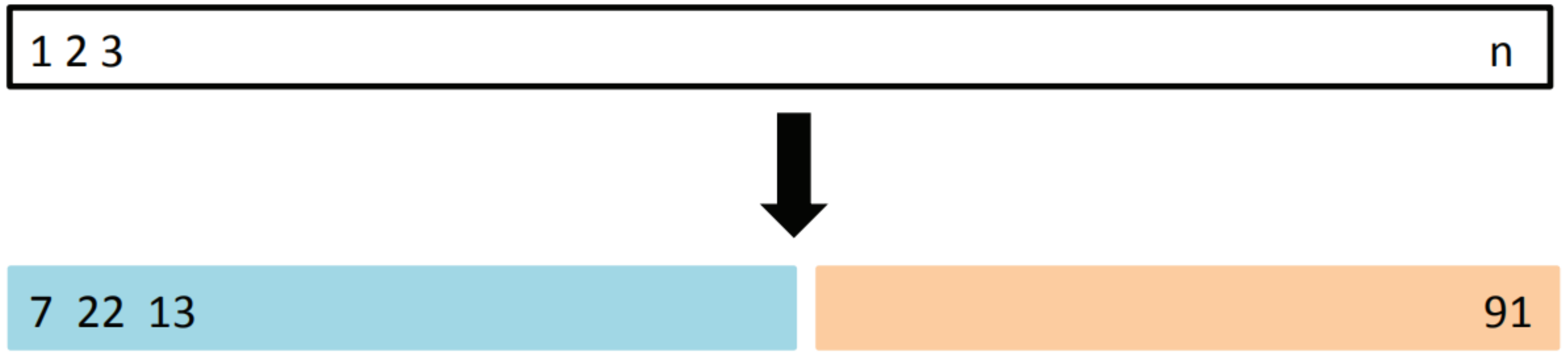


Source: Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Validation-set approach

- Here we *randomly divide* the available set of samples into two parts: a *training set* and a *validation* or *hold-out set*.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.
- The resulting validation-set error provides an estimate of the test error.
- This is typically assessed using *MSE* in the case of a quantitative response and *misclassification rate* in the case of a qualitative (discrete) response.

The Validation process

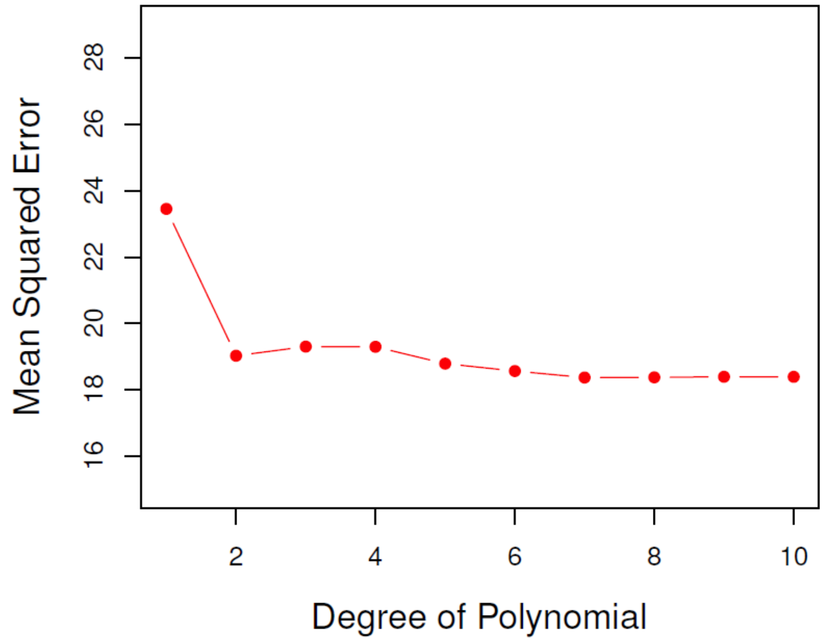


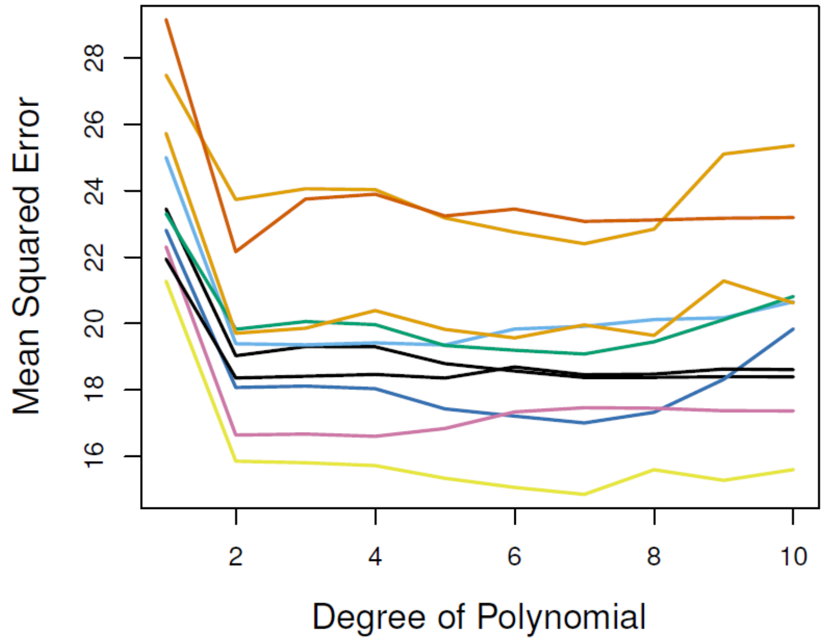
A random splitting into two halves:

left part is training set, right part is validation set

Example: automobile data

- Want to compare linear vs higher-order polynomial terms in a linear regression
- We randomly split the 392 observations into two sets, a training set containing 196 data points, a validation set containing the remaining 196 observations.





Left panel shows single split; right panel shows multiple splits

Drawbacks of validation set approach

- the validation estimate of the test error can be **highly variable**

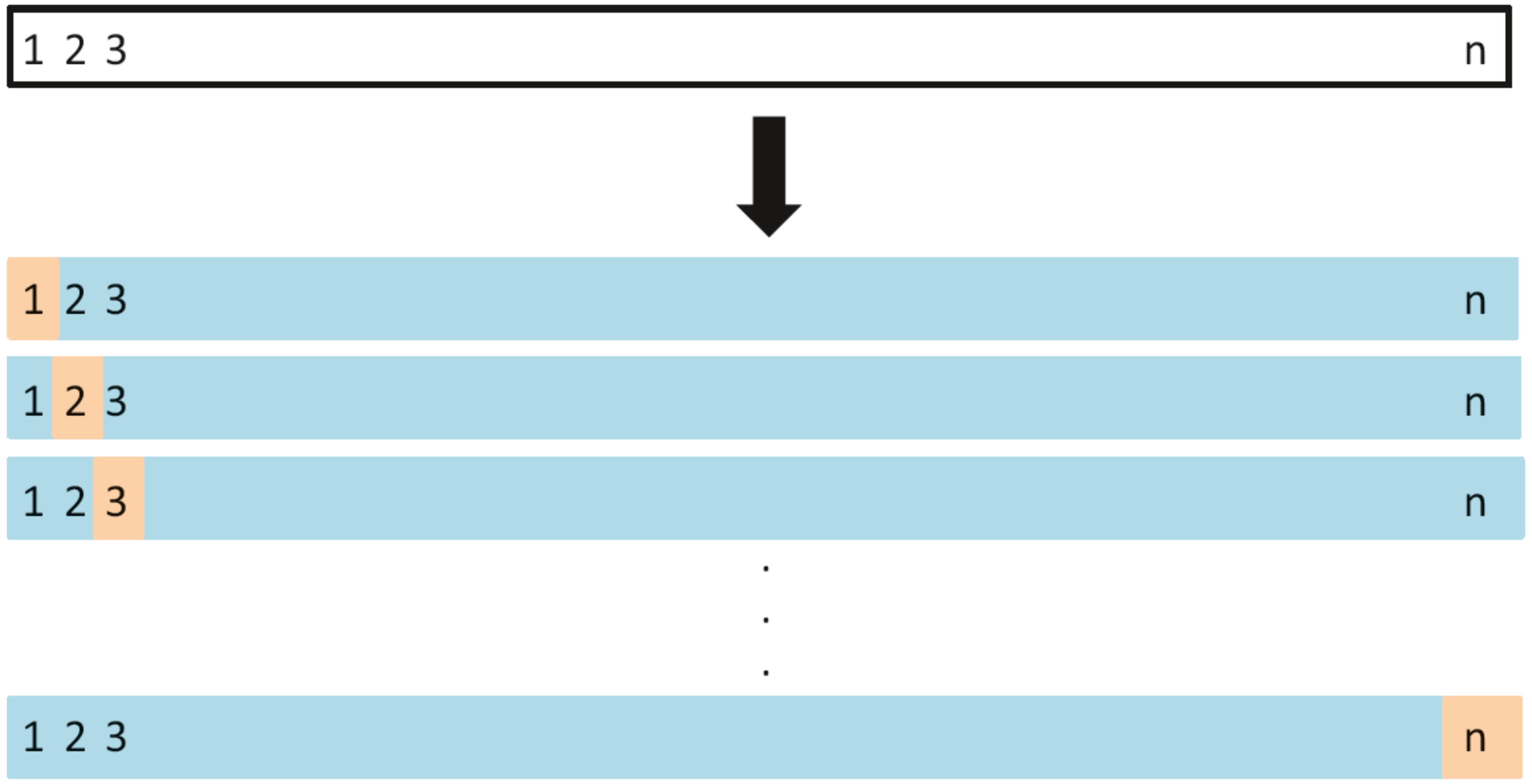
  ※ depending on precisely which observations are included in the training set and which observations are included in the validation set.

- In this approach, only a subset of the observations are used to fit the model.

  ※ those that are included in the training set rather than in the validation set

- This suggests that the validation set error may tend to ***overestimate*** the test error for the model fit on the entire data set. ***-- Why?***

Leave-one-out cross-validation (LOOCV)



A schematic display of LOOCV.

The first training set contains all but observation 1,

the second training set contains all but observation 2, and so forth.

Leave-one-out cross-validation (LOOCV)

- LOOCV is closely related to the validation

  ※ but it attempts to address that method's drawbacks

- A single observation is used for the validation set, and the remaining observations make up the training set.

  ※ So that the resulting MSE provides an approximately *unbiased estimate* for the test error. However it is a *poor* estimate because it is highly variable, since it is based upon a single observation

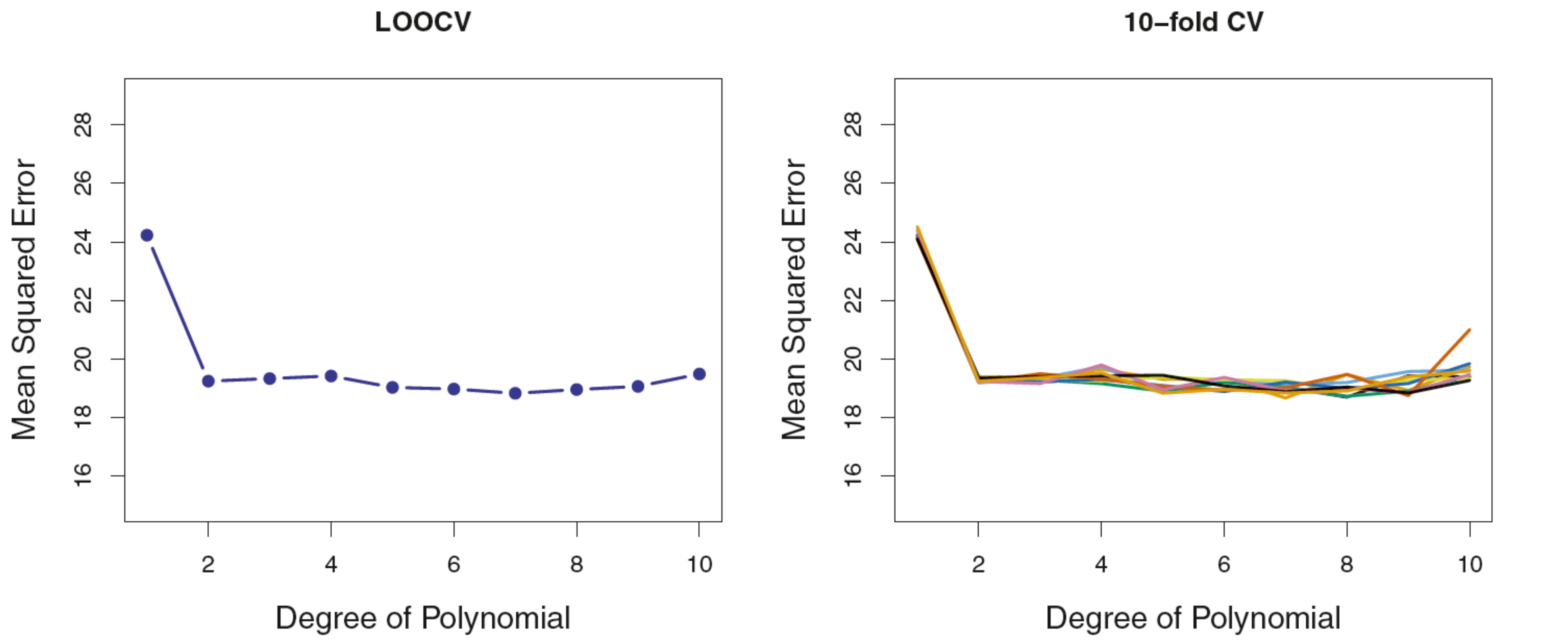
  ※ Repeating this approach *n times* can make a rescue.

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

Advantages

LOOCV has some major advantages over the validation set approach.

- LOOCV has far less bias, it tends not to overestimate the test error rate as much as the validation set approach does.
- Performing LOOCV multiple times will always yield the same results.



Leave-one-out cross-validation (LOOCV)

- Computing CV(n) can be computationally expensive

  ※ since it involves fitting the model n times

- For linear regression, there is a shortcut:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_{ii}}\right)^2$$

where h_{ii} is the leverage statistic -- diagonal values of the "hat" matrix.

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$$

- Proof: https://robjhyndman.com/hyndsight/loocv-linear-models/

K-fold Cross-validation

- Widely used approach for estimating test error.

  ※ Estimates can be used to select best model, and

  ※ to give an idea of the test error of the final chosen model.

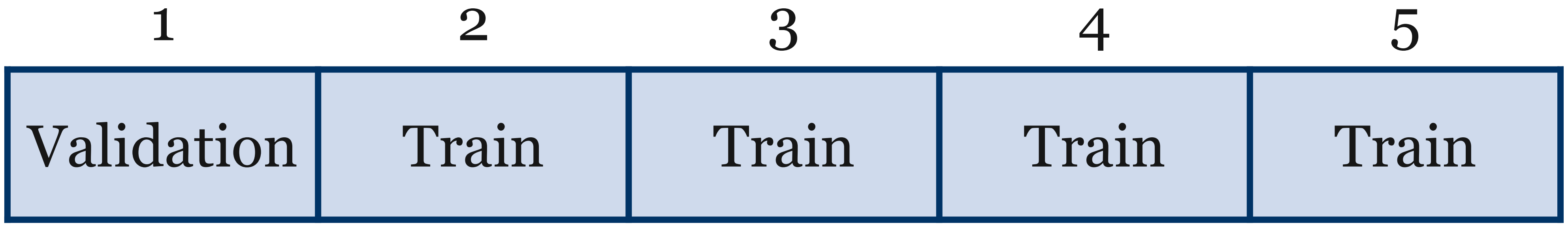
- Idea is to randomly divide the data into K equal-sized parts.

  ※ We leave out part k (k=1,2, … K), fit the model to the other K-1 parts (combined), and then obtain predictions for the left-out k-th part.

  ※ This is done in turn for each part , and then the results are combined.

K-fold Cross-validation in detail

Divide data into K *roughly* equal-sized parts (K=5 here)



The most obvious advantage is computational (*w.r.t.* LOOCV).

The details

- Let the *K* parts be $C_1, C_2, \ldots, C_K$,

  ※ where C_k denotes the indices of the observations in part *k*.

- There are n_k observations in part *k*: $n_k = n/K$
- Compute

$$\text{CV}_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} \text{MSE}_k$$

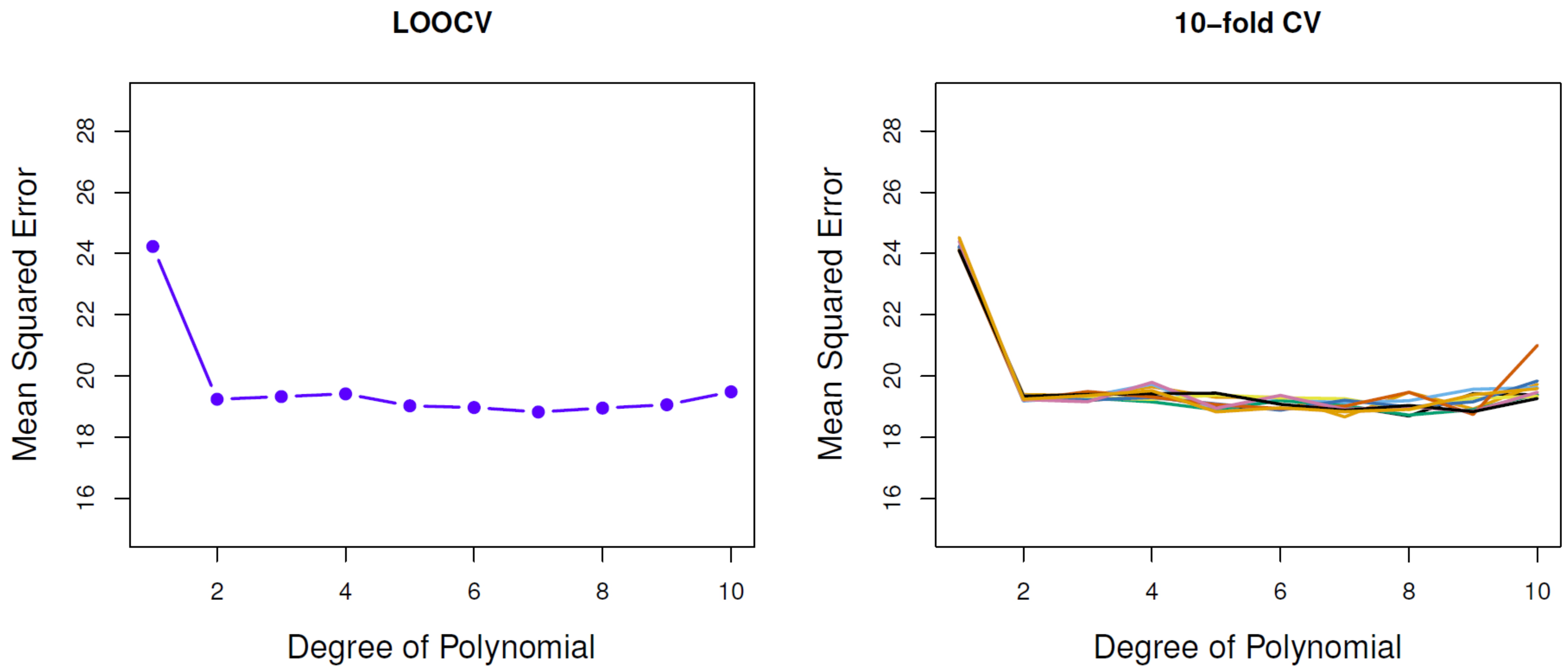
where

$$\text{MSE}_k = \frac{1}{n_k} \sum_{i \in C_k} (y_i - \hat{y}_i)^2$$

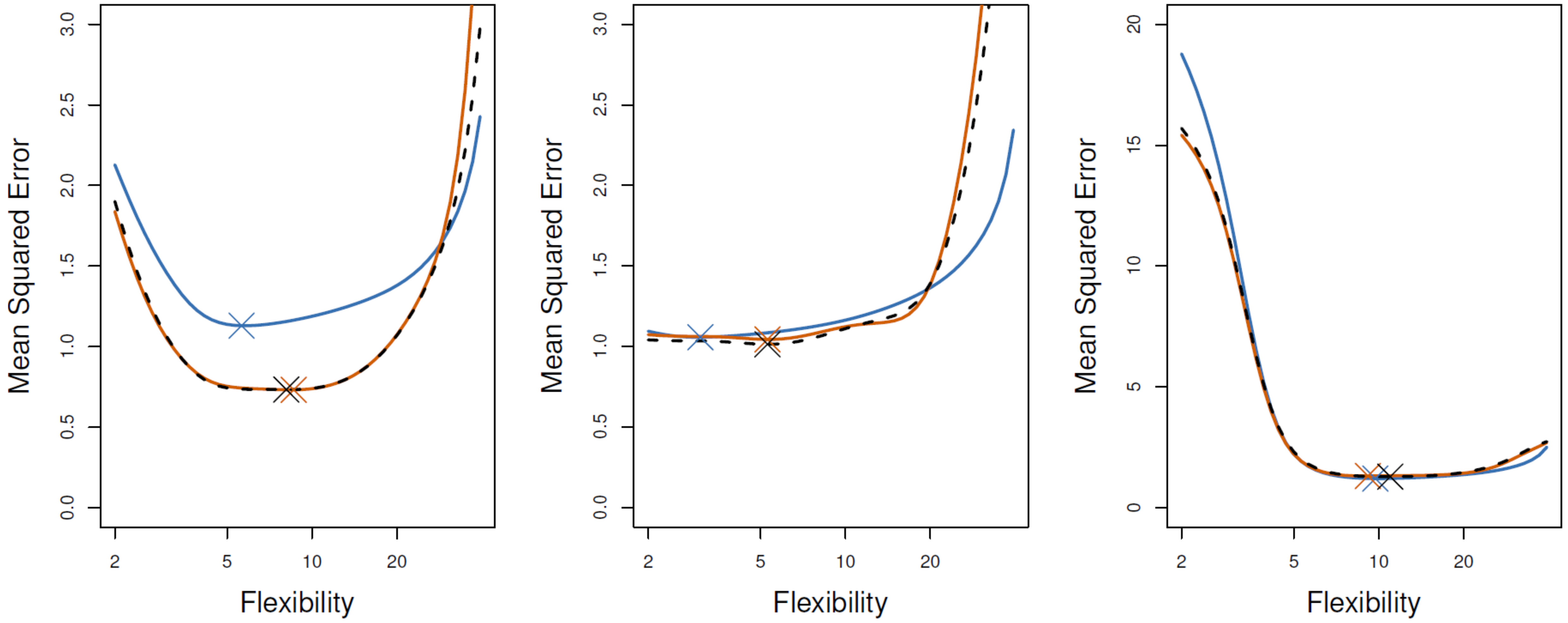
and $\hat{y}_i$ is the fit for observation *i*, obtained from the data with part *k* removed.

- Setting *K=n* yields *n-fold* or *leave-one out cross-validation* (*LOOCV*).

Auto data revisited



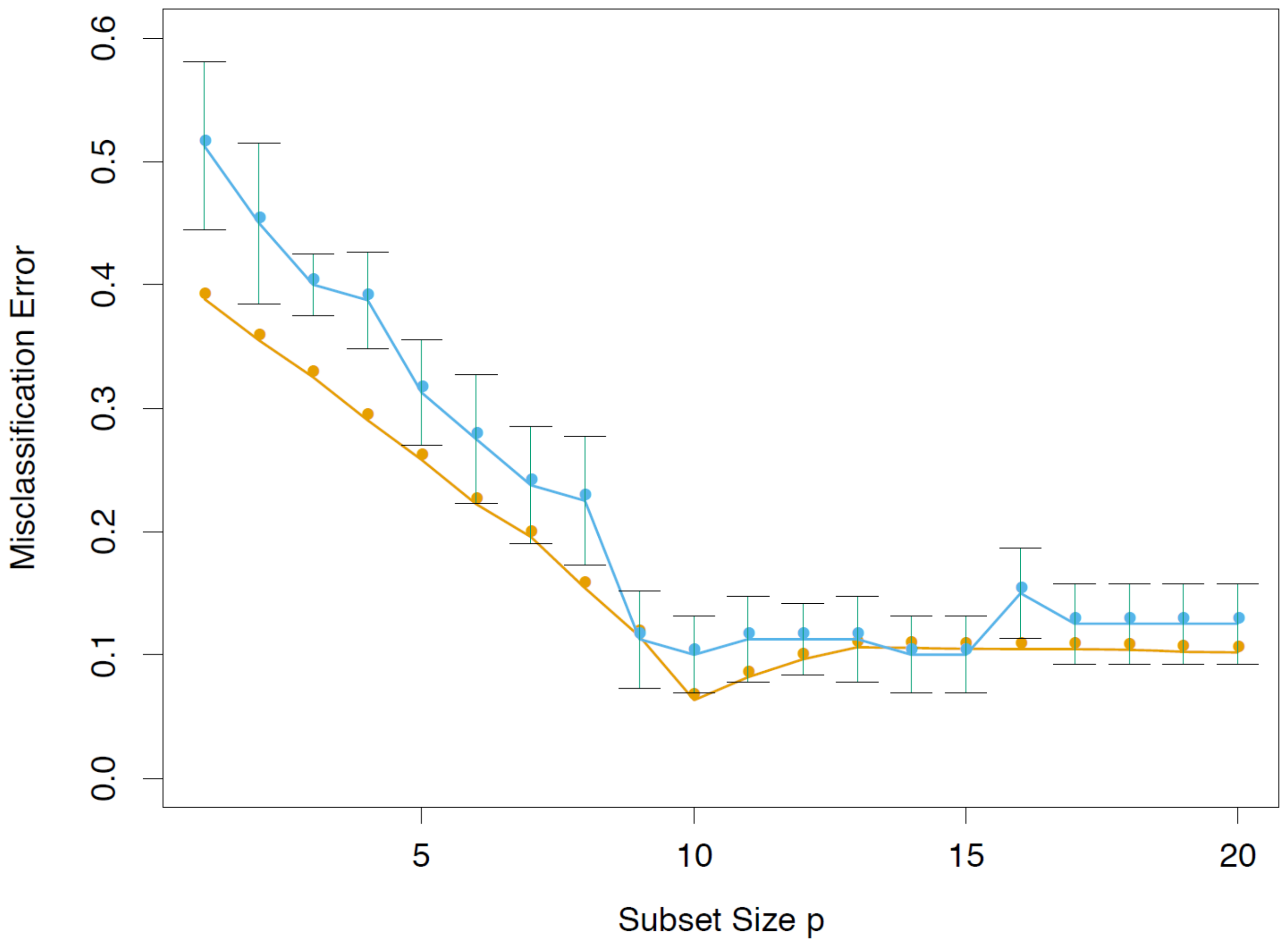
True and estimated test MSE for the simulated data



横轴是模型的复杂度，纵轴是模型的预测误差，其中蓝色线表示模拟数据集上的true test MSE，黑色虚线表示采用LOOCV对test MSE的估计，橙色线表示采用10-fold CV对test MSE的估计

The one standard error rule

Forward stepwise selection



Blue:10-fold CV; Yellow: True test error

- Curves minimized at p = 10.
- Models with $9 \leq p \leq 15$ have very similar CV error.
- The vertical bars represent 1 standard error in the test error from the 10 folds.
- **Rule of thumb:** Choose the simplest model whose CV error is no more than one standard error above the model with the lowest CV error.

Other issues with Cross-validation

Bias-Variance Trade-Off for k-Fold Cross-Validation

- Since each training set is only $(K-1)/K$ as big as the original training set, the estimates of prediction error will typically be biased upward. **Why?**

- This bias is minimized when $K = n$ (LOOCV), but this estimate has high variance, as noted earlier.

- **K = 5 or 10** provides a good compromise for this bias-variance trade off.

9.3.2 The Wrong and Right Way to Do Cross-validation

Despite the best efforts of statistical methodologists, users frequently invalidate their results by inadvertently peeking at the test data.

-- Page 708, Artificial Intelligence: A Modern Approach (3rd Edition), 2009.

Cross-validation: right and wrong

- Consider a simple classifier applied to some two-class data:
  1. Starting with **5000** predictors and **50** samples, find the **100** predictors having the largest correlation with the class labels.
  2. We then apply a classifier such as logistic regression, using only these **100** predictors.
- How do we estimate the test set performance of this classifier?
- Can we apply cross-validation in step 2, forgetting about step 1?

The wrong way to do cross validation

To see how that works, let's use the following simulated data:

- Each gene expression is standard *normal* and *independent* of all others.
- The *response* (cancer or not) is sampled from a *coin flip*

  ※ which means no correlation to any of the "genes".

- What should the misclassification rate be for any classification method using these predictors?

  ※ Roughly ***50%***.

The wrong way to do cross validation

- A typical strategy for analysis might be as follows:

1. Screen the predictors: find a subset of “good” predictors that show fairly strong (univariate) correlation with the class labels.
    - Using all the data, select the ***100*** most significant genes using ***z-tests***
2. Using just this subset of predictors, build a multivariate classifier.
3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.

- In our simulation, this produces an error estimate of close to 3%.

The wrong way to do cross validation

- We run this simulation, and obtain a CV error rate of ***3%***!
- Why is this?

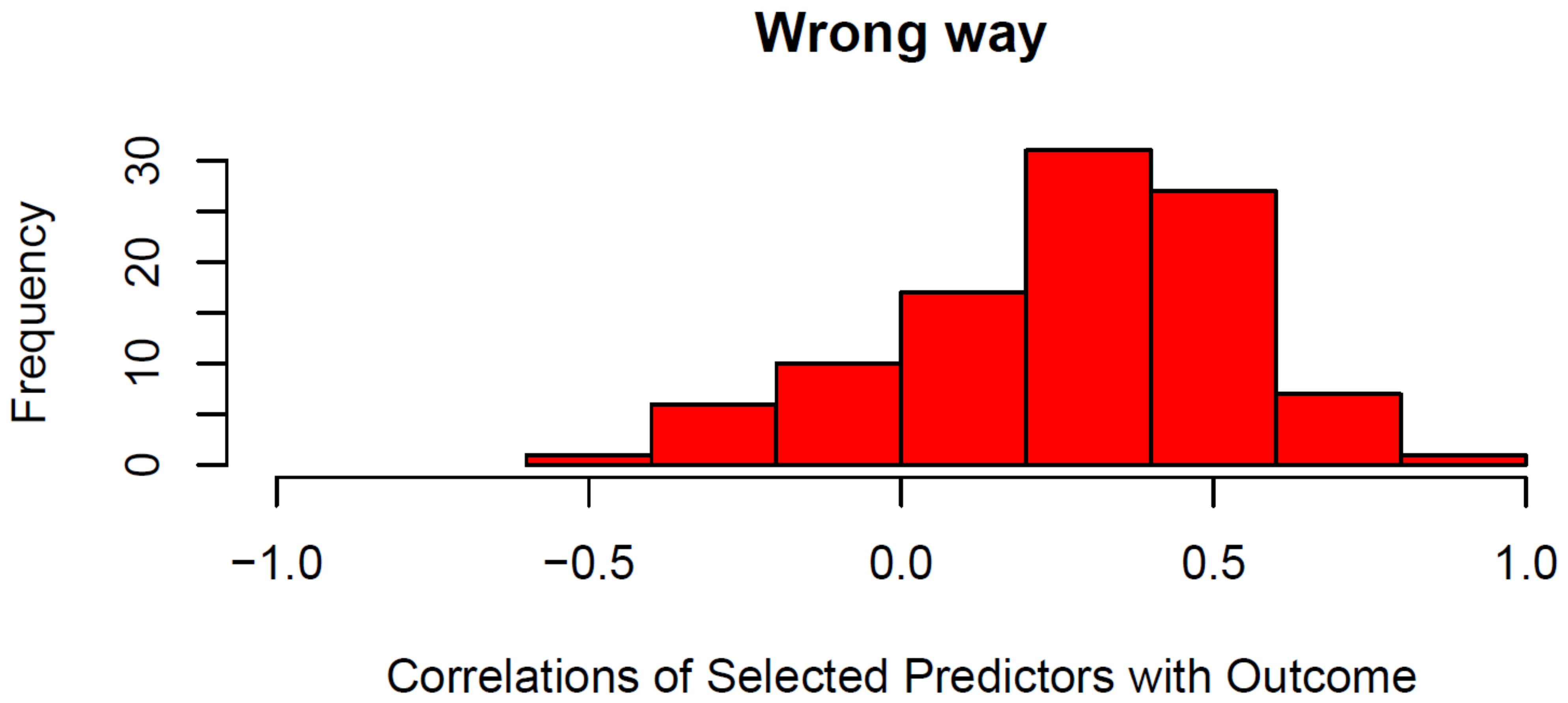
※ Since we only have ***50*** individuals in total, among ***100*** variables, at least some will be correlated with the response.

※ We do variable selection using all the data, so *the variables we select have some correlation with the response* in every *subset* or *fold* in the cross validation.

NO!

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and must be included in the validation process
- It is easy to simulate realistic data with the class labels independent of the outcome, so that true test error = 50%, but the CV error estimate that ignores Step 1 is zero! *-- Try to do this yourself*
- We have seen this error made in many high profile genomics papers.

The wrong way to do cross validation

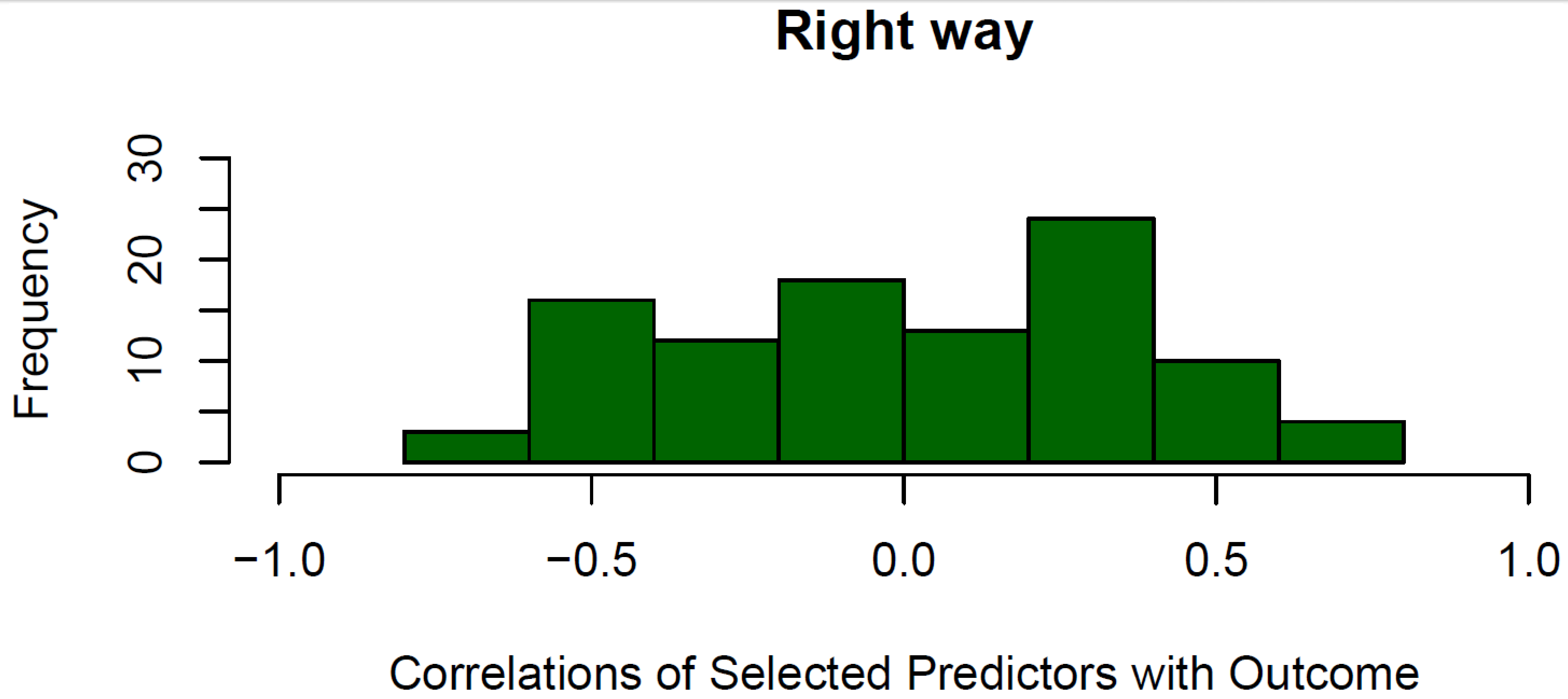


Cross-validation the wrong way:
histograms shows the correlation of class labels, in 10 randomly chosen samples, with the 100 predictors chosen using the incorrect versions of cross-validation.

The right way to do cross validation

- Divide the data into 10 folds at random.
- For ***k = 1, …, 10***:
  1. Find a subset of "good" predictors that show fairly strong correlation with the class labels, using all of the samples except those in fold k.
  2. Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.
  3. Use the classifier to predict the class labels for the samples in fold k.
- In our simulation, this produces an error estimate of close to 50%.
- **Moral of the story**: Every aspect of the learning method that involves using the data (eg. variable selection) must be cross-validated.

The right way to do cross validation



Cross-validation the right way:
histograms shows the correlation of class labels, in 10 randomly chosen samples, with the 100 predictors chosen using the and correct versions of cross-validation.

The Wrong and Right Way

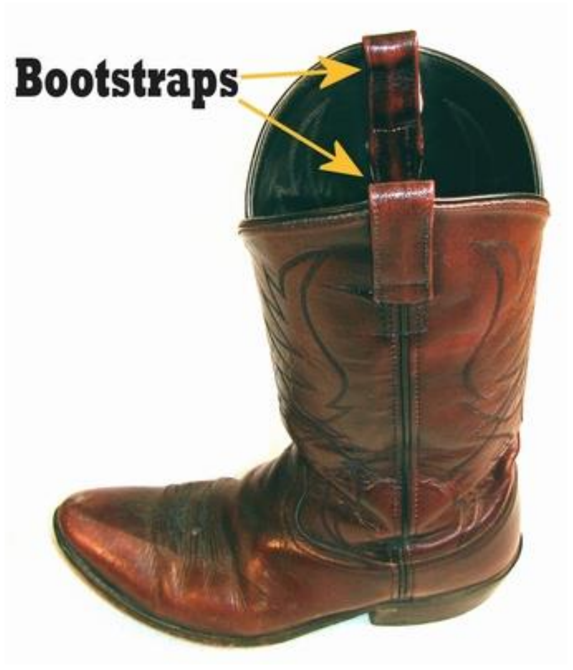
- *Wrong*: Apply cross-validation in step 2.
- *Right*: Apply cross-validation to steps 1 and 2.

9.3.3 The Bootstrap

The Bootstrap

- The ***bootstrap*** is a widely applicable and extremely powerful statistical tool that can be used to *quantify the uncertainty* associated with a given estimator or statistical learning method.

- For example, it can provide an estimate of the *standard error* of a coefficient, or a *confidence interval* for that coefficient.



Where does the name came from?

- The use of the term bootstrap derives from the phrase :

to pull oneself up by one's bootstraps

- widely thought to be based on one of the eighteenth century "*The Surprising Adventures of Baron Munchausen*" by Rudolph Erich Raspe:

The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.

- It is not the same as the term "bootstrap" used in computer science meaning to "boot" a computer from a set of core instructions.

Bootstrap procedure

- The bootstrap method can be used to estimate a quantity of a population. This is done by repeatedly taking small samples, calculating the statistic, and taking the average of the calculated statistics.
- We can summarize this procedure as follows:

1. Choose a number of bootstrap samples to perform
2. Choose a sample size
3. For each bootstrap sample
   a) Draw a sample with replacement with the chosen size
   b) Calculate the statistic on the sample
4. Calculate the mean of the calculated sample statistics.

A simple example

- Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities.
  - ※ We will invest a fraction α of our money in X
  - ※ and will invest the remaining $1-\alpha$ in Y.
- We wish to choose α to minimize the total risk (or variance) of our investment.
- In other words, we want to minimize $\mathrm{Var}(\alpha X + (1-\alpha)Y)$.
- One can show that the value that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

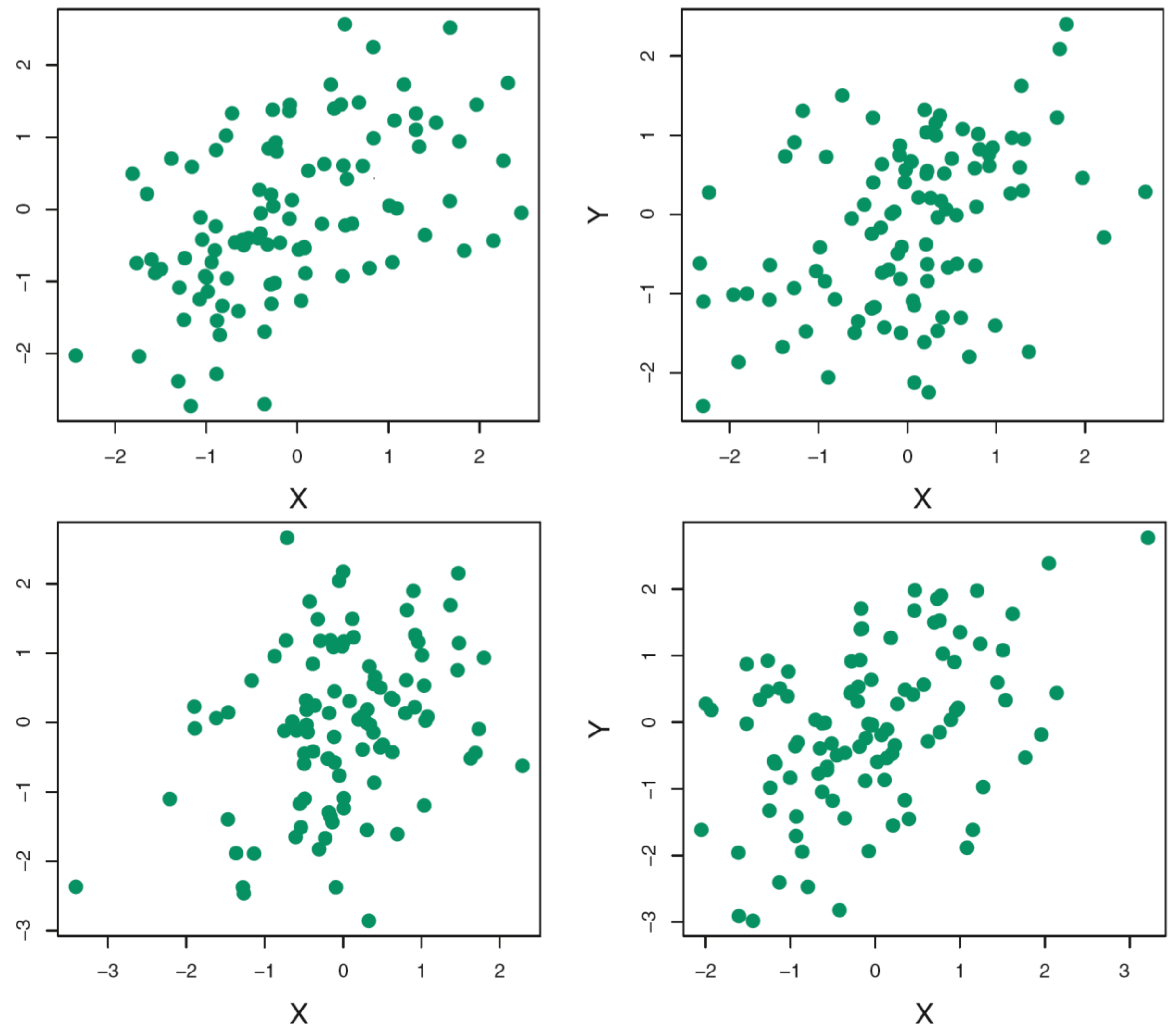
where $\sigma_X^2 = Var(X)$, $\sigma_Y^2 = Var(Y)$, and $\sigma_{XY} = Cov(X,Y)$.

Example continued

- But the values of σ_X^2, σ_Y^2, and σ_{XY} are unknown.
- We can compute estimates for these quantities, $\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$, and $\hat{\sigma}_{XY}$, using a data set that contains measurements for X and Y.
- We can then estimate the value of α that minimizes the variance of our investment using

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

Example continued



Each panel displays 100 simulated returns for investments X and Y. From left to right and top to bottom, the resulting estimates for α are 0.576, 0.532, 0.657, and 0.651.

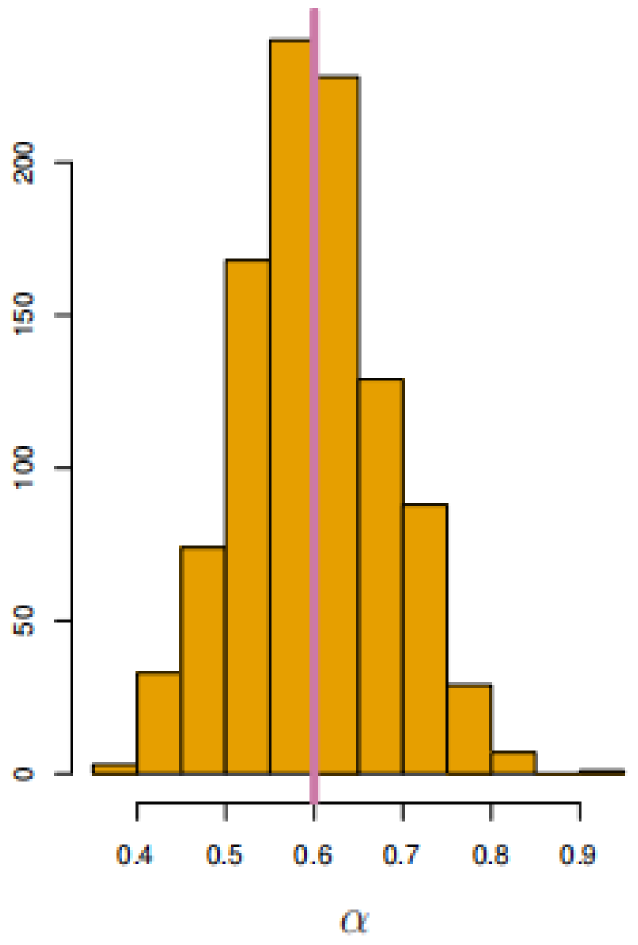
Example continued

- To estimate the standard deviation of $\hat{\alpha}$, we repeated the process of simulating 100 paired observations of X and Y, and estimating α 1,000 times.

- We thereby obtained 1,000 estimates for α, which we can call $\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{1000}$

- The Figure displays a histogram of the resulting estimates.

- For these simulations the parameters were set to $\sigma_X^2 = 1$, $\sigma_Y^2 = 1.25$, and $\sigma_{XY} = 0.5$, and so we know that the true value of α is 0.6 (indicated by the red line)



Example continued

- The mean over all 1,000 estimates for α is

$$\bar{\alpha} = \frac{1}{1000} \sum_{r=1}^{1000} \hat{\alpha}_r = 0.5996$$

very close to $\alpha = 0.6$, and the standard deviation of the estimates is

$$\sqrt{\frac{1}{1000-1} \sum_{r=1}^{1000} (\hat{\alpha}_r - \bar{\alpha})^2} = 0.083$$

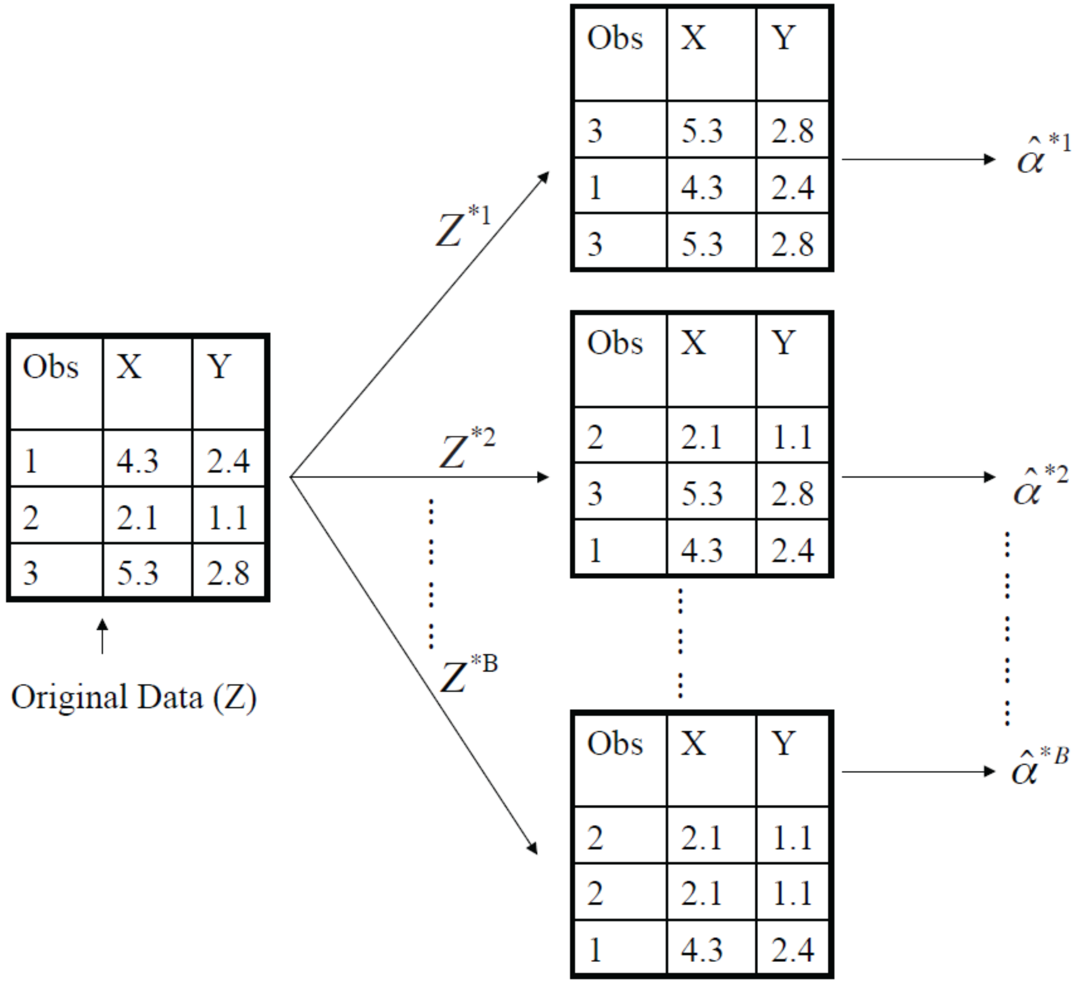
- This gives us a very good idea of the accuracy of $\hat{\alpha}$: $\mathrm{SE}(\hat{\alpha}) \approx 0.083$

- So roughly speaking, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

Now back to the real world

- The procedure outlined above cannot be applied, because for real data we cannot generate new samples from the original population.
- However, the bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set with replacement.
- Each of these “*bootstrap data sets*” is created by *sampling with replacement*, and is the *same size* as our original dataset.
- As a result some observations may appear more than once in a given bootstrap data set and some not at all.

Example with just 3 observations



Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

A graphical illustration of the bootstrap approach on a small sample containing n = 3 observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α .

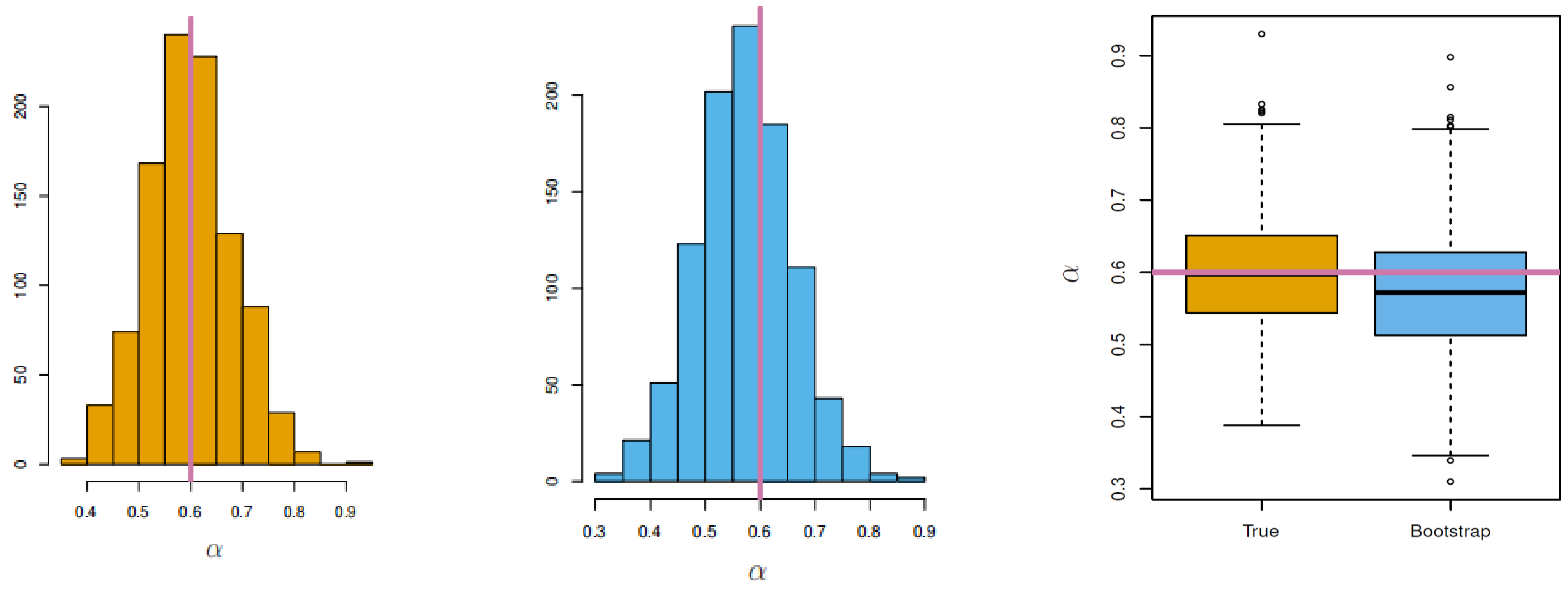
Example with just 3 observations

- Denoting the first bootstrap data set by Z^{*1}, we use Z^{*1} to produce a new bootstrap estimate for α , which we call $\hat{\alpha}^{*1}$
- This procedure is repeated B times for some large value of B (say 100 or 1000), in order to produce B different bootstrap data sets, $Z^{*1}, Z^{*2}, \ldots, Z^{*B}$, and B corresponding α estimates : $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \ldots, \hat{\alpha}^{*B}$.
- We estimate the standard error of these bootstrap estimates using the formula

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}(\hat{\alpha}^{*r} - \bar{\hat{\alpha}}^{*})^2}$$

- This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set. See center and right panels of Figure on slide 55. Bootstrap results are in blue. For this example $SE_{B(\hat{\alpha})} = 0.087$.

Results



Left: A histogram of the estimates of α obtained by generating 1,000 simulated data sets from the true population. Center: A histogram of the estimates of α obtained from 1,000 bootstrap samples from a single data set. Right: The estimates of α displayed in the left and center panels are shown as boxplots.

In each panel, the pink line indicates the true value of α.

9.3.4 Bagging

Reduce Variance Without Increasing Bias

- Averaging reduces variance: $$Var(\bar{\mathbf{x}}) = \frac{Var(\mathbf{x})}{N}$$
- Average models to reduce model variance
  - ※ One problem: only one train set
  - ※ where do multiple models come from?
- Bagging: Bootstrap Aggregation (1994)
  - ※ **Leo Breiman** (1928 - 2005)
  - ※ Bootstrap Sample:
    - ➢ draw sample of size |D| with replacement from D

Revisit of Bias-Variance

$$\text{avg}\{E_{out}(g_t)\} = \text{avg}\{\mathbf{E}(g_t - \bar{g})^2\} + E_{out}(\bar{g})$$

- expected performance of A = variance + bias

  ※ variance : expected deviation to consensus

  ※ bias : performance of consensus

- consensus more stable than direct $A(D)$

  ※ but comes from many more D_t than the D on hand

- want: approximate g by

  ※ finite (large) T

  ※ approximate $g_t = \mathcal{A}(\mathcal{D}_t)$ from $\mathcal{D}_t \sim P^N$ using only D

Bootstrapping:

re-samples from D to simulate D_t

Bootstrap Aggregation

- bootstrap sample D_t
  ※ re-sample N examples from D uniformly with replacement
  ※ can also use arbitrary N' instead of original N
- **B**ootstrap **agg**regat**ing** (**Bagging**)
  ※ consider a iterative process that for $t = 1, 2, \ldots, T$
  ※ request size-N' data D_t from bootstrapping
  ※ obtain g_t by $\mathcal{A}(\mathcal{D}_t)$: $\mathrm{G} = \mathrm{Uniform}(\{\ \mathrm{g}_t\})$
- Bagging : a simple meta algorithm on top of base algorithm $\boldsymbol{A}$

Pocket Algorithm

- initialize pocket weights $\hat{\boldsymbol{w}}$
- For $t = 0, 1, \ldots$

  ① find a (random) mistake of $\boldsymbol{w}_t$ called $(\boldsymbol{x}_{n(t)}, y_{n(t)})$

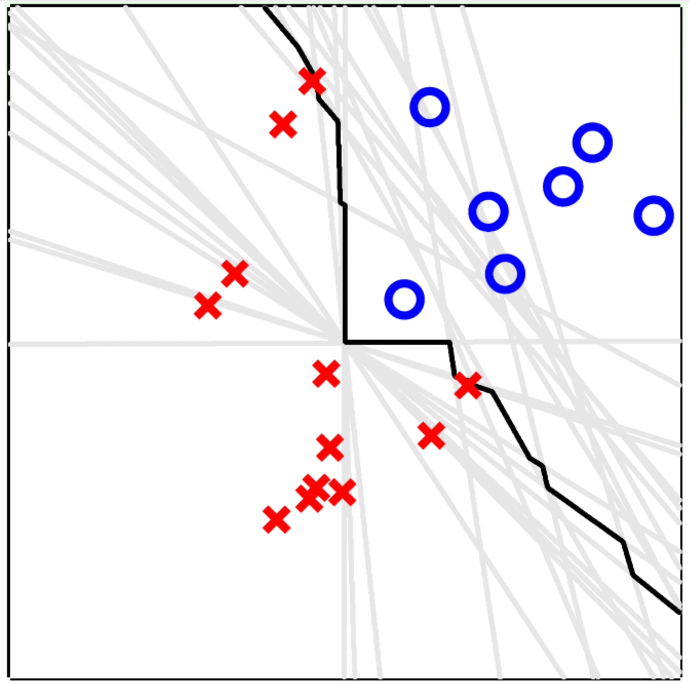
  ② (try to) correct the mistake by

  $$\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + y_{n(t)}\boldsymbol{x}_{n(t)}$$

  ③ If $\boldsymbol{w}_{t+1}$ makes fewer mistakes than $\hat{\boldsymbol{w}}$, replace $\hat{\boldsymbol{w}}$ by $\boldsymbol{w}_{t+1}$

- …… until enough iterations
- return $\hat{\boldsymbol{w}}$ (called $\boldsymbol{w}_{\text{POCKET}}$) as g

Bagging Pocket in Action



$T_{Pocket} = 1000$

$T_{Bagging} = 25$

- very diverse g_t from bagging
- proper non-linear boundary after aggregating binary classifiers
- bagging works reasonably well

  ※ if base algorithm sensitive to data randomness

Quize

When using bootstrapping to re-sample N examples $\tilde{\mathcal{D}}_t$ from a data set D with N examples, what is the probability of getting $\tilde{\mathcal{D}}_t$ exactly the same as D?

(1) $0/N^N = 0$ (2) $1/N^N$ (3) $N!/N^N$ (4) $N^N/N^N = 1$

- Consider re-sampling in an ordered manner for N steps
  - ※ Then there are (N^N) possible outcomes $\tilde{\mathcal{D}}_t$
    - ➢ each with equal probability
  - ※ (N!) of the outcomes are permutations of the original D

Reference Answer: 3

Bagging: Bootstrap Aggregation

- Best case:

$$\mathrm{Var}(\mathrm{Bagging}(L(\mathbf{x},,\mathcal{D}))) = \frac{\mathrm{Var}(L(\mathbf{x},,\mathcal{D})}{N}$$

- In practice:
  - ※ models are correlated, so reduction is smaller than 1/N
  - ※ variance of models trained on fewer training cases usually larger
  - ※ stable learning methods have low variance to begin with,
    - ➢ so bagging may not help much

Can Bagging Hurt?

- Each base classifier is trained on less data
  - ※ Only about **63.2%** of the data points are in any bootstrap sample
- However the final model has seen all the data
  - ※ On average a point will be in **>50%** of the bootstrap samples

Javed A. Aslam, et al. *On Estimating the Size and Confidence of a Statistical Audit.* Proceedings of the Electronic Voting Technology Workshop. Boston, MA, August 6, 2007.

Reduce Bias^2 and Decrease Variance?

- Bagging reduces variance by averaging

- Bagging has little effect on bias

- Can we ***average*** and ***reduce bias***?

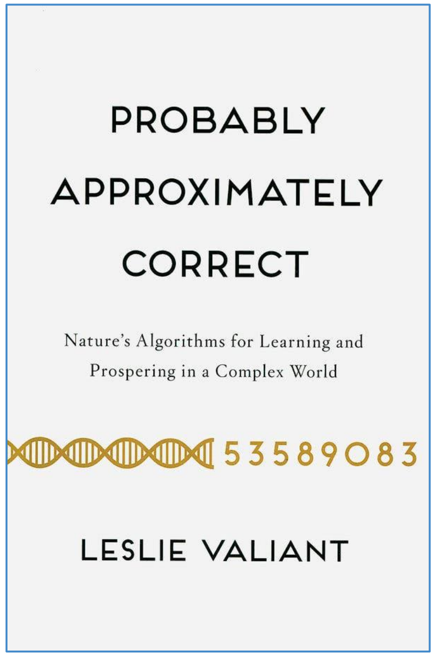
Yes : Boosting

9.4 Boosting Method

9.4.1 Boosting

Boosting

- Yoav Freund & Robert Schapire (2003 Gödel Prize)
  - ※ Boosting： Foundations and Algorithms （2012）
- Weak Learner
  - ※ performance on any train set is *slightly* better than *chance* prediction
  - ※ PAC : theory for weak learners in late 80’s （Valiant）
  - ※ intended to answer a theoretical question
    - ➢ not as a practical way to improve learning
  - ※ Tested in mid 90's using not-so-weak learners
    - ➢ works anyway!



PROBABLY APPROXIMATELY CORRECT

Nature's Algorithms for Learning and Prospering in a Complex World

53589083

LESLIE VALIANT

Boosting

1. Weight all training samples equally
2. Train model on train set
3. Compute error of model on train set
4. Increase weights on train cases model gets wrong
5. Train new model on re-weighted train set
6. Re-compute errors on weighted train set
7. Increase weights again on cases model gets wrong
8. Repeat until tired (100+ iterations)
9. Final model: weighted prediction of each model

Boosting vs. Bagging

- Bagging doesn't work so well with stable models.

  ※ Boosting might still help.
- Boosting might hurt performance on noisy datasets.

  ※ Bagging doesn't have this problem
- In practice bagging almost always helps.
- On average, boosting helps more than bagging

  ※ but it is also more common for boosting to hurt performance.

  ※ The weights grow exponentially.
- Bagging is easier to parallelize.

Bagging and Boosting

- Probably Approximately Correct (PAC, Kearns & Valiant)
- Ensemble : learners are trained using same learning techniques.

  ※ Bagging : bootstrap aggregating (random forest)

  ➢ bootstrap : pull up by your own bootstraps

  ※ Boosting (adaboost)

  ➢ Can a set of weak learners create a single strong learner?
- Hybrid: learners are trained using different learning techniques.

  ※ Stacking : combining multiple models (meta learners)

9.4.2 Adaptive Boosting

Bootstrapping as Re-weighting Process

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3), (\mathbf{x}_4, y_4)\} \overset{bootstrap}{\Longrightarrow} \mathcal{D}_t = \{(\mathbf{x}_1, y_1), (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_4, y_4)\}$$

- E$_{in}$ on $\mathcal{D}_t$: $E_{in}(h) = \frac{1}{4} \sum_{(\mathbf{x},y)\in\mathcal{D}_t} [\![y \neq h(\mathbf{x})]\!]$

- weighted E$_{in}$ on $\mathcal{D}$: $E_{in}(\alpha, h) = \frac{1}{4} \sum_{i=1}^{4} \alpha_i \cdot [\![y_i \neq h(\mathbf{x}_i)]\!]$

$$(\mathbf{x}_1, y_1) : \alpha_1 = 2 \quad (\mathbf{x}_2, y_2) : \alpha_2 = 1 \quad (\mathbf{x}_3, y_3) : \alpha_3 = 0 \quad (\mathbf{x}_4, y_4) : \alpha_4 = 1$$

- each diverse g_t in bagging:

  ※ by minimizing bootstrap-weighted error

Re-weighting for More Diverse Hypothesis

- improving bagging for binary classification:

※ how to re-weight for more diverse hypotheses?

$$g_t \leftarrow \underset{h\in\mathcal{H}}{argmin}\left(\sum_{i=1}^{N}\alpha_i^t \cdot [\![y_i \neq h(\mathbf{x}_i)]\!]\right) \qquad g_{t+1} \leftarrow \underset{h\in\mathcal{H}}{argmin}\left(\sum_{i=1}^{N}\alpha_i^{t+1} \cdot [\![y_i \neq h(\mathbf{x}_i)]\!]\right)$$

※ if g_t not good for $\boldsymbol{\alpha}^{t+1}$, then:

➢ g_t-like hypotheses not returned as g_{t+1}

➢ Want: g_{t+1} diverse from g_t

※ idea: construct $\boldsymbol{\alpha}^{t+1}$ to make g_t random-like

$$\sum_{i=1}^{N}\alpha_i^{t+1} \cdot [\![y_i \neq g_t(\mathbf{x}_i)]\!] = \frac{1}{2}\sum_{i=1}^{N}\alpha_i^{t+1}$$

Optimal Re-weighting

- Let : $e^{t+1} = \sum_{i=1}^{N} \alpha_i^{t+1} \cdot [\![y_i \neq g_t(\mathbf{x}_i)]\!]$ denotes total α_i^{t+1} of incorrect
- Let : $r^{t+1} = \sum_{i=1}^{N} \alpha_i^{t+1} \cdot [\![y_i = g_t(\mathbf{x}_i)]\!]$ denotes total α_i^{t+1} of correct
- Want : $\sum_{i=1}^{N} \alpha_i^{t+1} \cdot [\![y_i \neq g_t(\mathbf{x}_i)]\!] = \frac{1}{2} \sum_{i=1}^{N} \alpha_i^{t+1}$
- one possibility by re-scaling (multiplying) weights, if

※ total α_i^t of incorrect = 1234 ; total α_i^t of correct = 4321

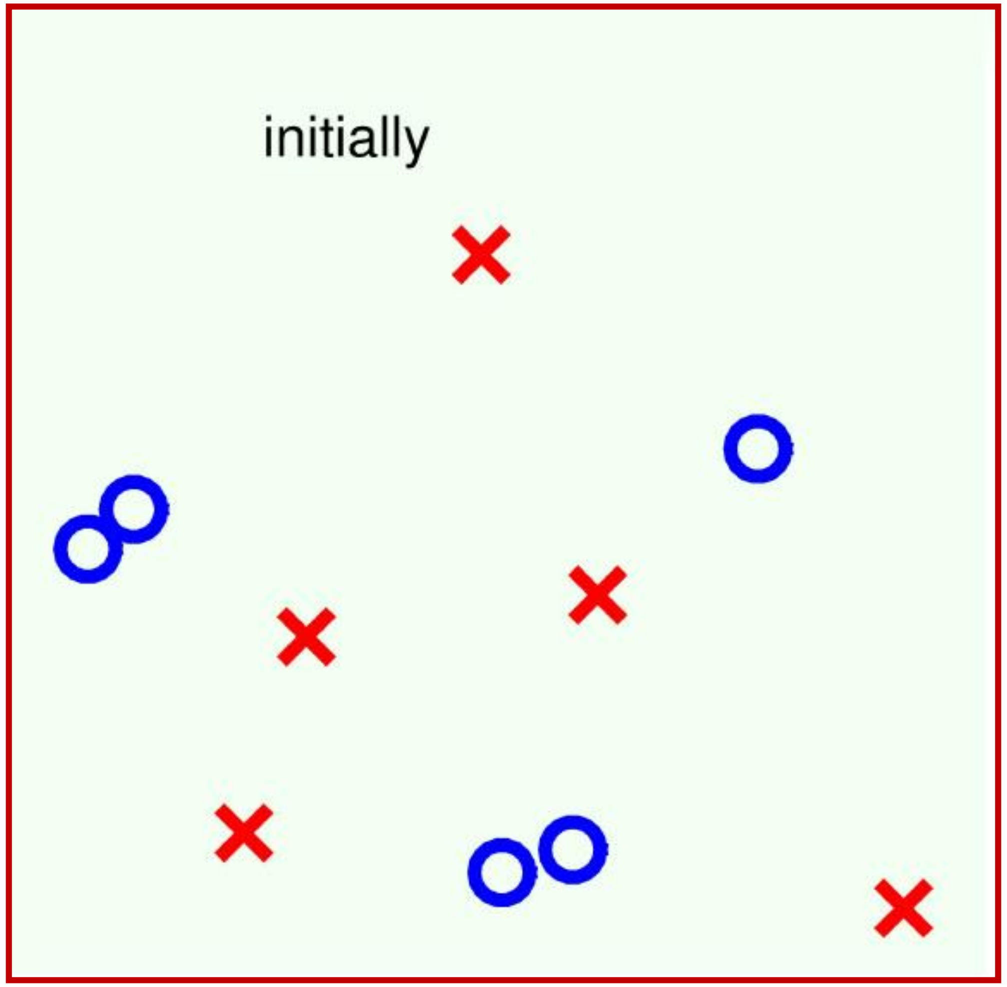
※ weighted incorrect rate = 1234 / 5555

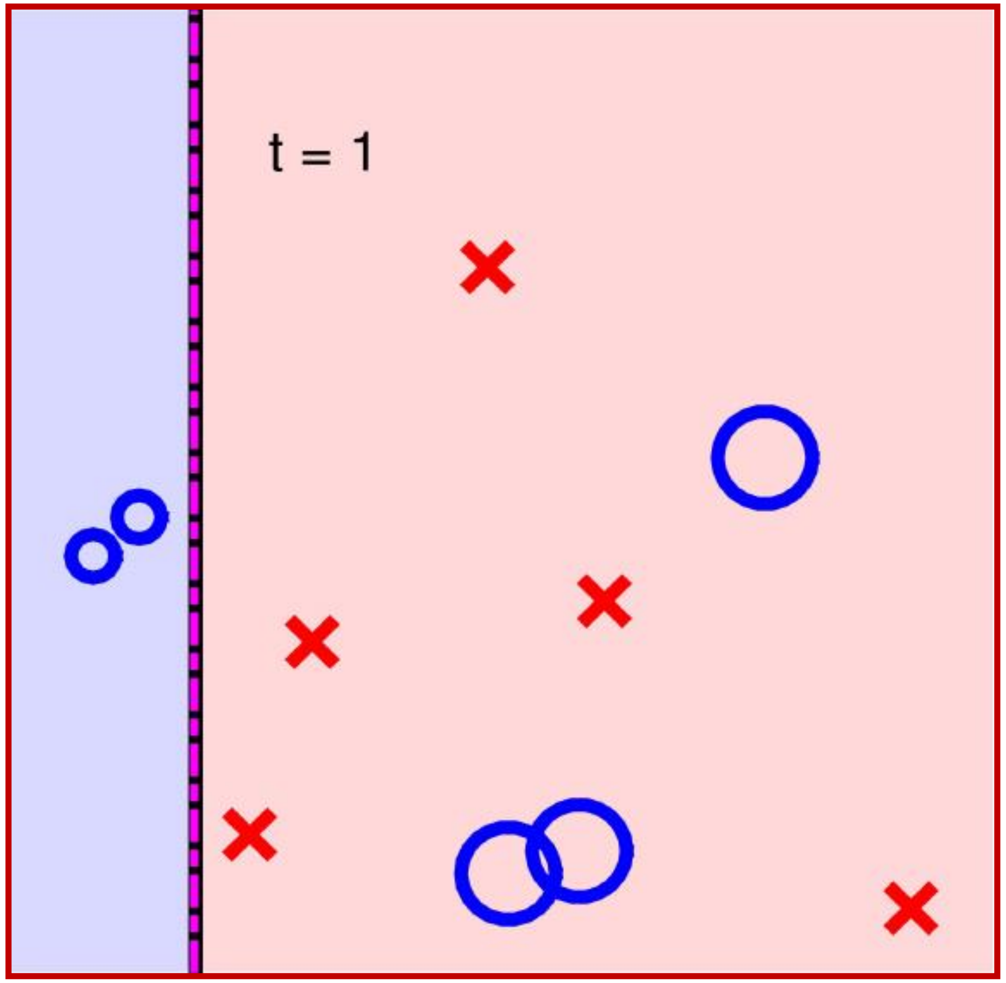
※ incorrect : $\alpha_i^{t+1} \leftarrow \alpha_i^t \cdot 4321$ correct : $\alpha_i^{t+1} \leftarrow \alpha_i^t \cdot 1234$

Decision Stump

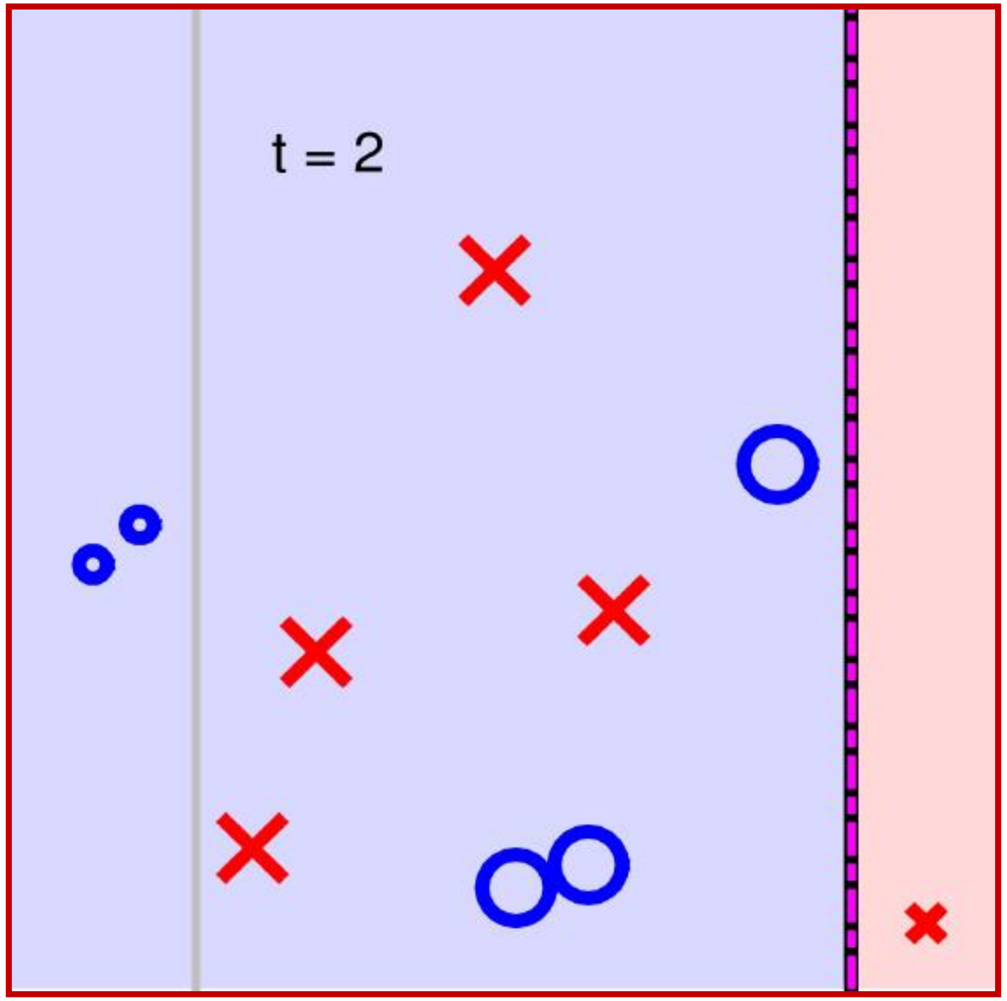
- want: a 'weak' base learning algorithm A
  ※ that minimizes $E_{in}^{\boldsymbol{u}}(h) = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{u}_n \cdot [\![y_n \neq h(\mathbf{x}_n)]\!]$ a little bit
- a popular choice: decision stump $h_{s,i,\theta}(\mathbf{x}) = s \cdot \text{sign}(x_i - \theta)$
  ※ positive and negative rays on some feature
    - ➢ three parameters: feature i, threshold θ, direction s
  ※ physical meaning: vertical/horizontal lines in 2D
  ※ efficient to optimize: $O(d \cdot N \log N)$ time
- decision stump model: allows efficient minimization of $E_{in}^{\boldsymbol{u}}(h)$
  ※ but perhaps too weak to work by itself

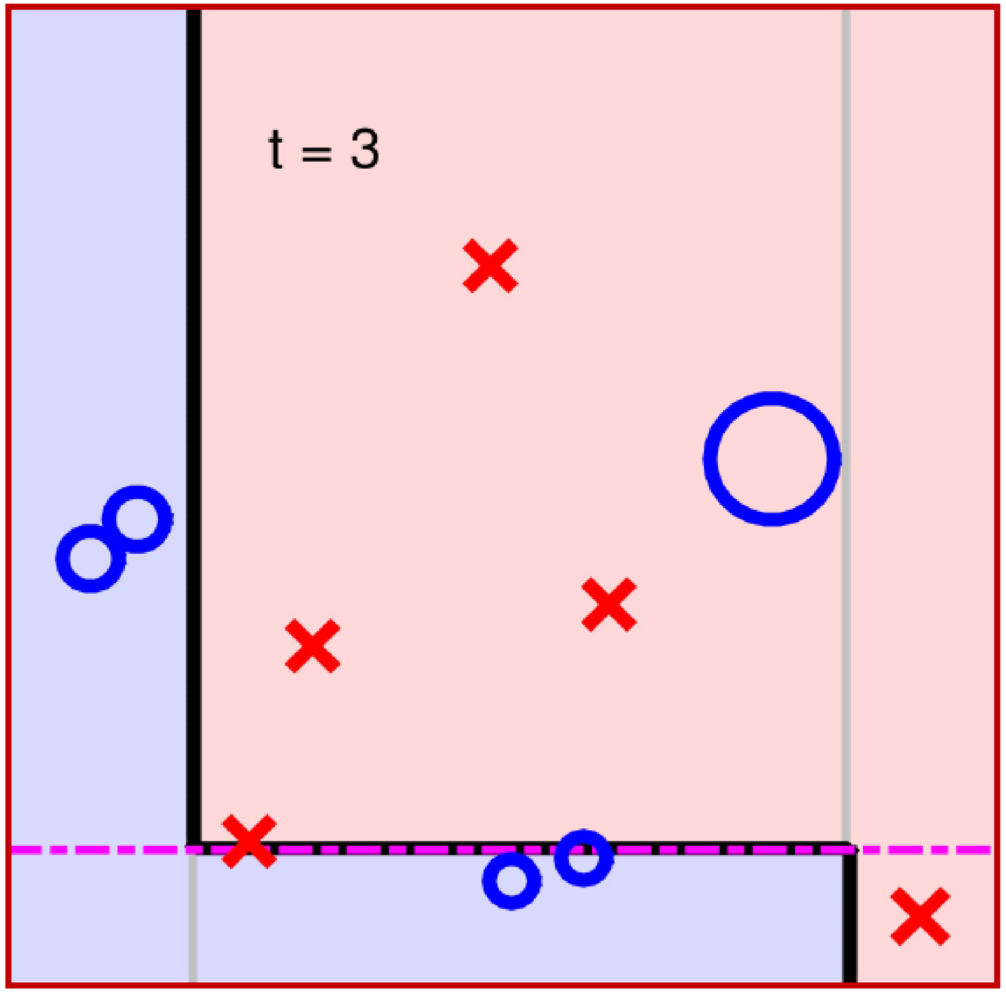
A Simple Data Set



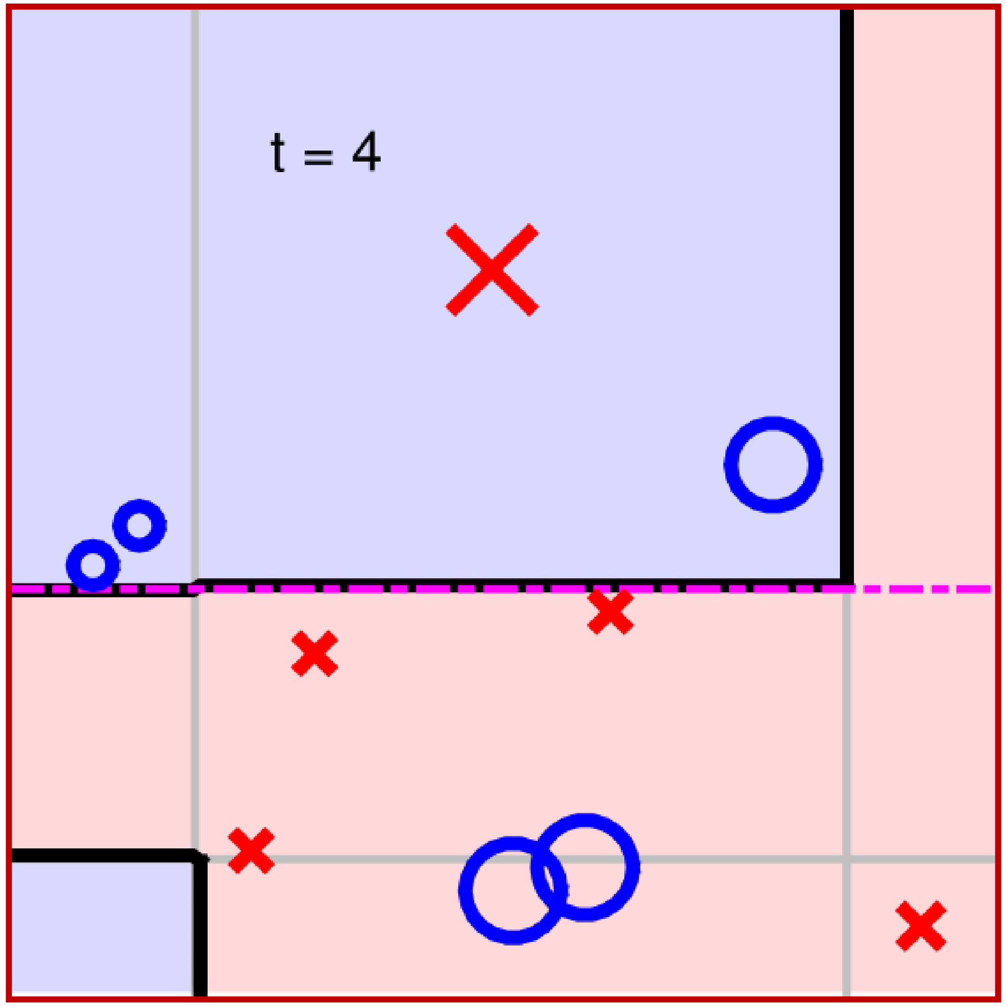


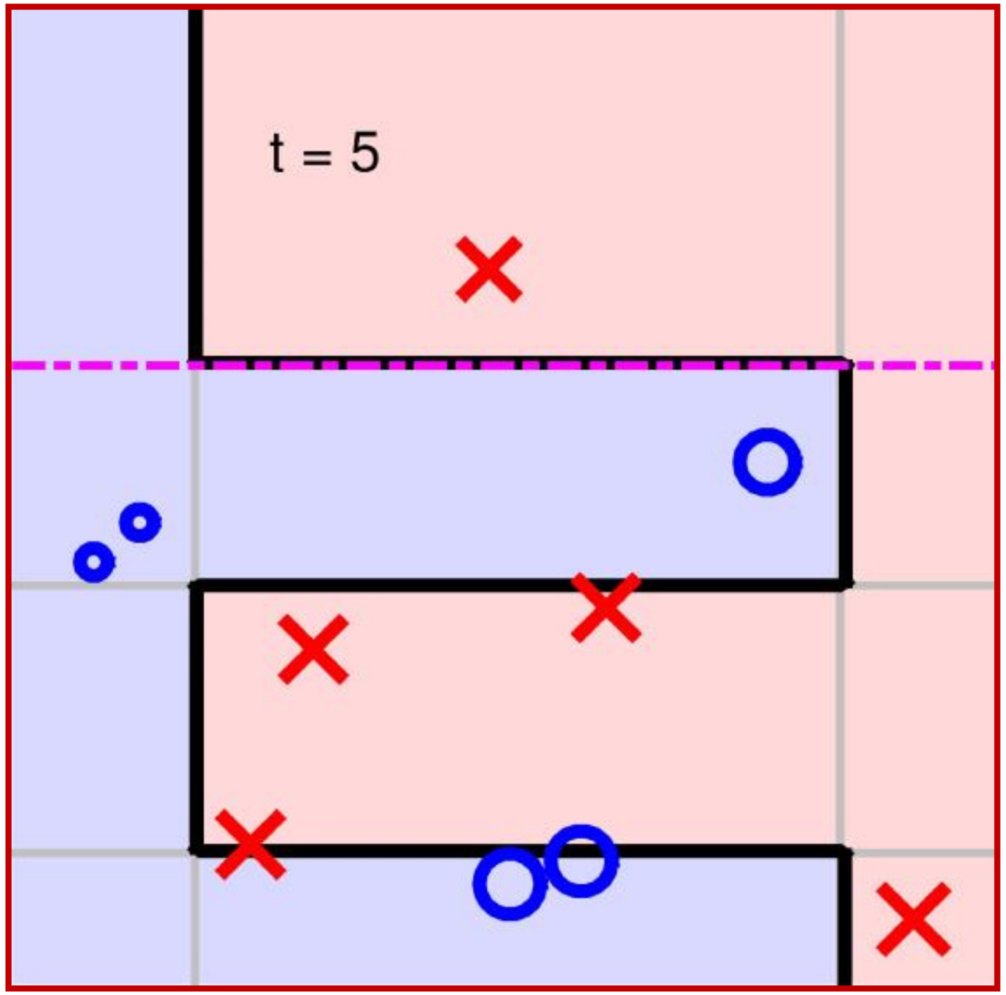
A Simple Data Set



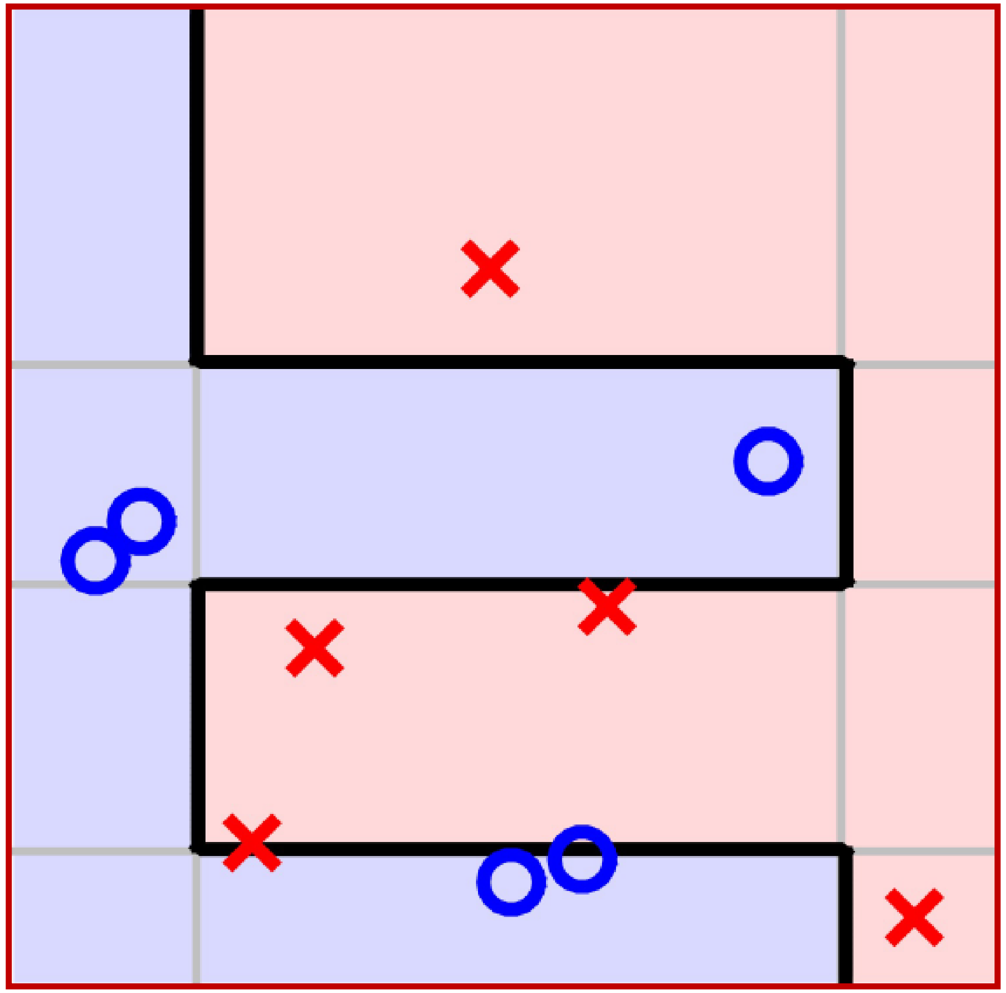


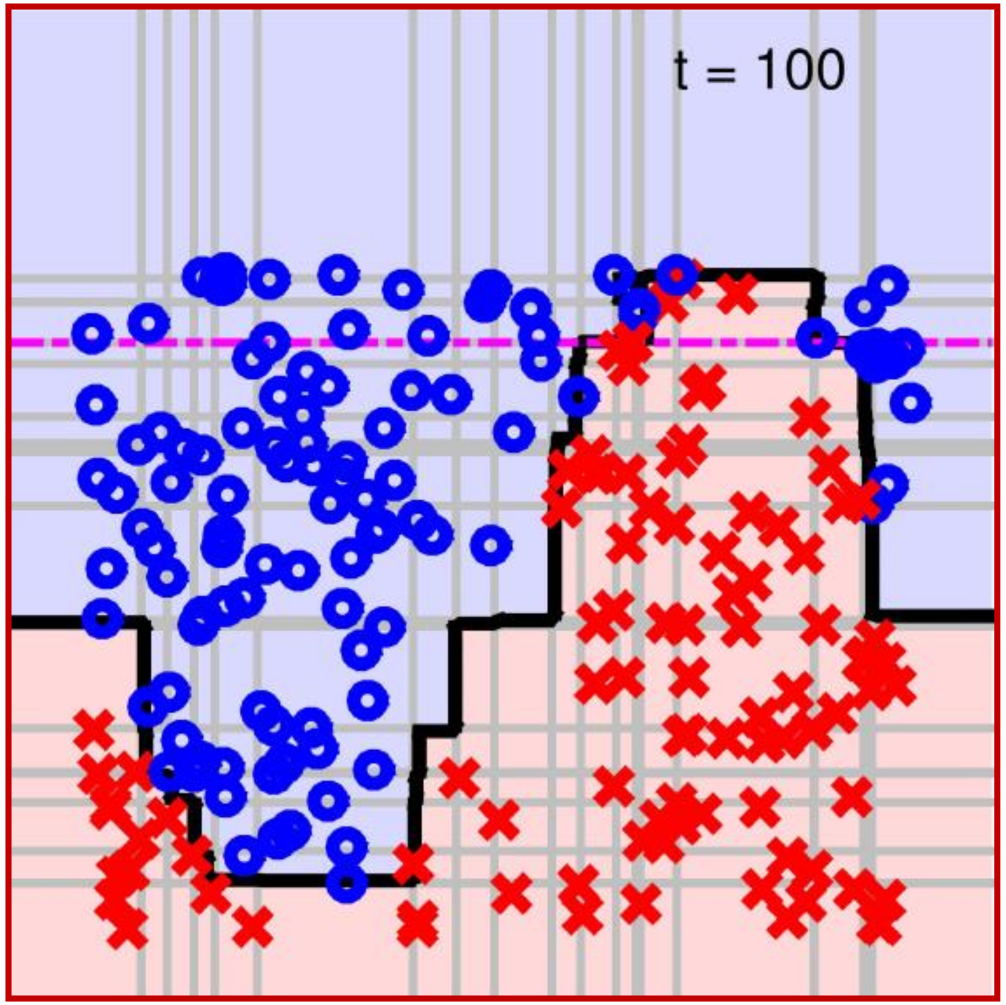
A Simple Data Set





A Simple Data Set





AdaBoost-Stump: non-linear yet efficient

Quize

For four examples with $\alpha_i^{(1)} = 1/4$ for all examples. If g_1 predicts the first example wrongly but all the other three examples correctly. After the optimal re-weighting, what is $\alpha_1^{(2)}/\alpha_2^{(2)}$

(1) 4 **(2) 3** **(3) 1/3** **(4) 1/4**

By optimal re-weighting, α_1 is scaled proportional to 3/4 and every other α_i is scaled proportional to 1/4 . So example 1 is now three times more important than any other example.

Reference Answer: 2

Bagging : Bootstrap Aggregation

- optimal re-weighting: let $\epsilon_t = \sum_{i=1}^{N} \alpha_i^{t+1} \cdot ind\{y_i \neq g_t(\mathbf{x}_i)\} / \sum_{i=1}^{N} \alpha_i^{t+1}$

  ※ multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$

- define scaling factor: $\lambda_t = \sqrt{(1 - \epsilon_t)/\epsilon_t}$

  ※ incorrect $\leftarrow$ incorrect $\cdot \lambda_t$; correct $\leftarrow$ correct $/ \lambda_t$

  ※ equivalent to optimal re-weighting: $\lambda_t \geq 1$ iff $\epsilon_t \leq 1/2$

  ※ physical meaning: scale up incorrect; scale down correct

- scaling-up incorrect examples leads to diverse hypotheses!

A Preliminary Algorithm

- for $t = 1, 2, \cdots, T$
  1. obtain g_t by $\mathcal{A}(D, \alpha^{(t)})$
     where $\mathcal{A}$ tries to minimize $\alpha^{(t)}$-weighted 0/1 error
  2. update $\alpha^{(t)}$ to $\alpha^{(t+1)}$ by $\lambda_t = \sqrt{(1-\epsilon_t)/\epsilon_t}$
     where ϵ_t = weighted error (incorrect) rate of g_t
- return G(x)

- $\alpha^{(1)}$ =? want g_1 "best" for Ein: $\alpha_i^{(1)} = 1/N$
  ※ but g_2 very bad for Ein (why?)
- G(x) =?
  ※ uniform? linear, non-linear? as you wish!

Linear Aggregation on the Fly

- $\alpha^{(1)} = [1/N, 1/N, \ldots, 1/N]$, for $t = 1, 2, \cdots, T$
  1. obtain g_t by $\mathcal{A}(D, \alpha^{(t)})$
  2. update $\alpha^{(t)}$ to $\alpha^{(t+1)}$ by $\lambda_t = \sqrt{(1-\epsilon_t)/\epsilon_t}$
  3. compute $\beta_t = \ln(\lambda_t)$
- return $G(x) = \text{sign}\left(\sum_{t=1}^{T} \beta_t g_t(\mathbf{x})\right)$

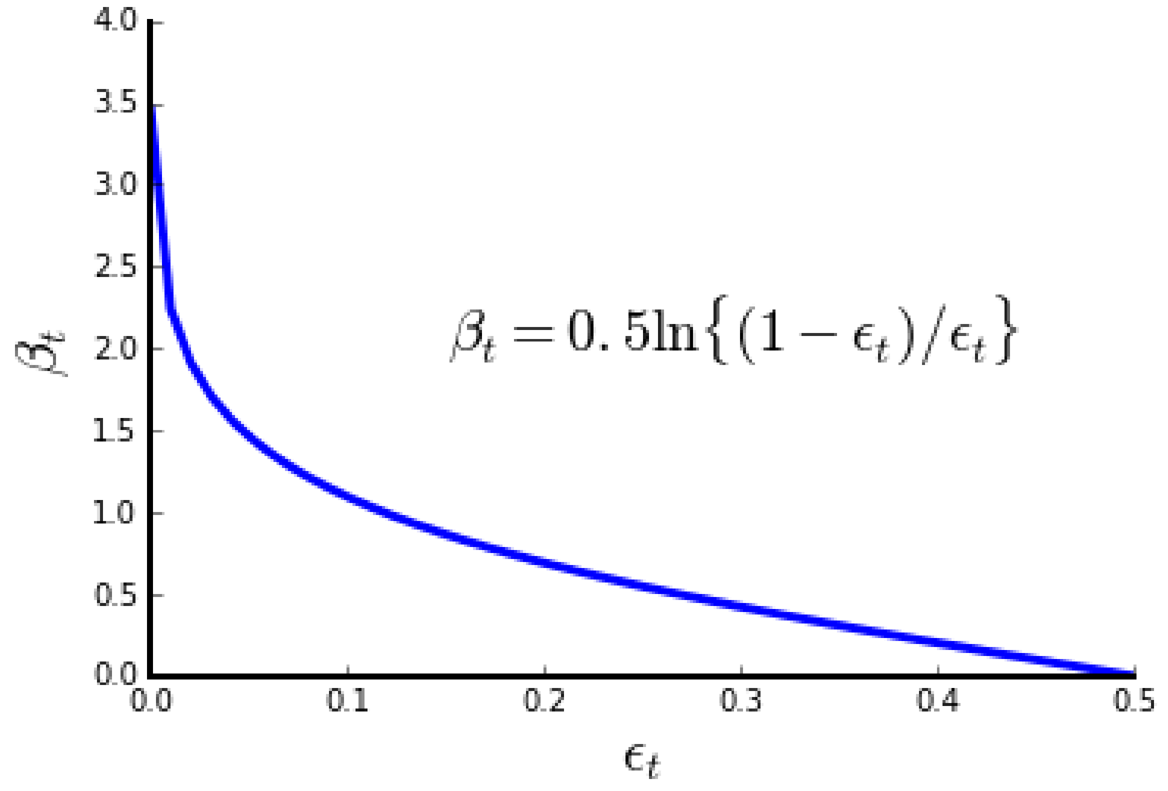
- wish: large β_t for good g_t $\Rightarrow \beta_t = f(\lambda_t)$ monotonic
- will take $\beta_t = \ln(\lambda_t)$

  ※ $\epsilon_t = 0.5 \Rightarrow \lambda_t = 1 \Rightarrow \beta_t = 0$ (bad g_t zero weight)

  ※ $\epsilon_t = 0 \Rightarrow \lambda_t = \infty \Rightarrow \beta_t = \infty$ (super g_t superior weight)

Linear Aggregation on the Fly

- wish: large β_t for good g_t $\Rightarrow \beta_t = f(\lambda_t)$ monotonic



Essentially, the weight of the classifier in the ensemble is proportional to the log-odds of *it being correct* vs *making an error* -- assuming $0 < \epsilon_t < 0.5$

Adaptive Boosting

Adaptive Boosting = weak base learning algorithm $\mathcal{A}$

optimal re-weighting factor λ_t

"magic" linear aggregation β_t

AdaBoost: provable boosting property

Theoretical Guarantee of AdaBoost

- From VC bound:

d_{VC} *of all possible G*

$$E_{out}(G) \le E_{in}(G) + O\left(\sqrt{O(d_{VC}(\mathcal{H}) \cdot T \log T) \cdot \frac{\log N}{N}}\right)$$

- $E_{in}(G)$ can be small:

※ $E_{in}(G) = 0$ after $\mathrm{T} = \mathrm{O}(\log N)$ iterations if: $\epsilon_t \le \epsilon < 0.5$

- second term can be small:

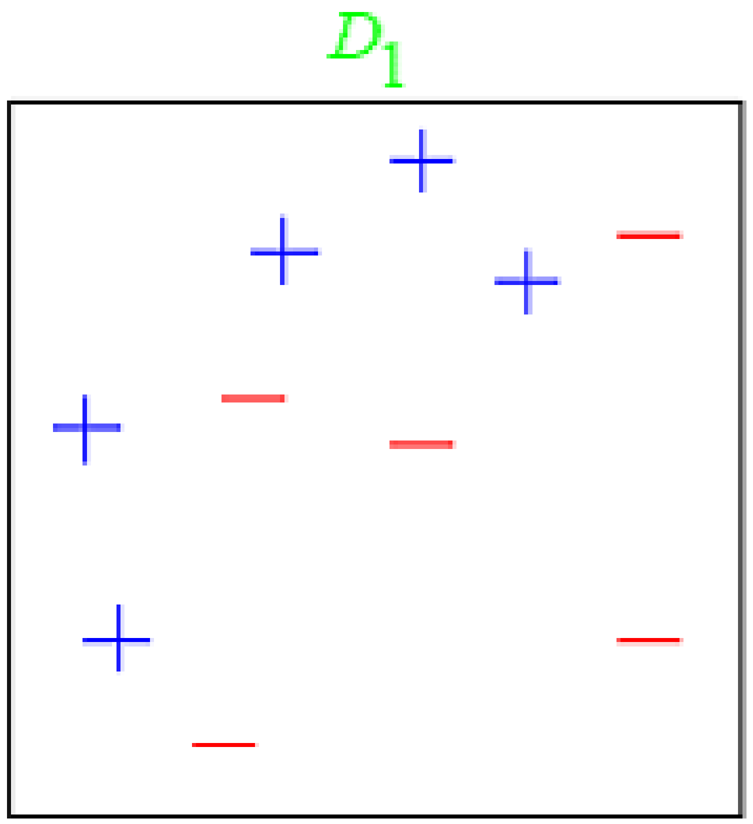
※ overall d_{VC} grows "slowly" with T

- boosting view of AdaBoost:

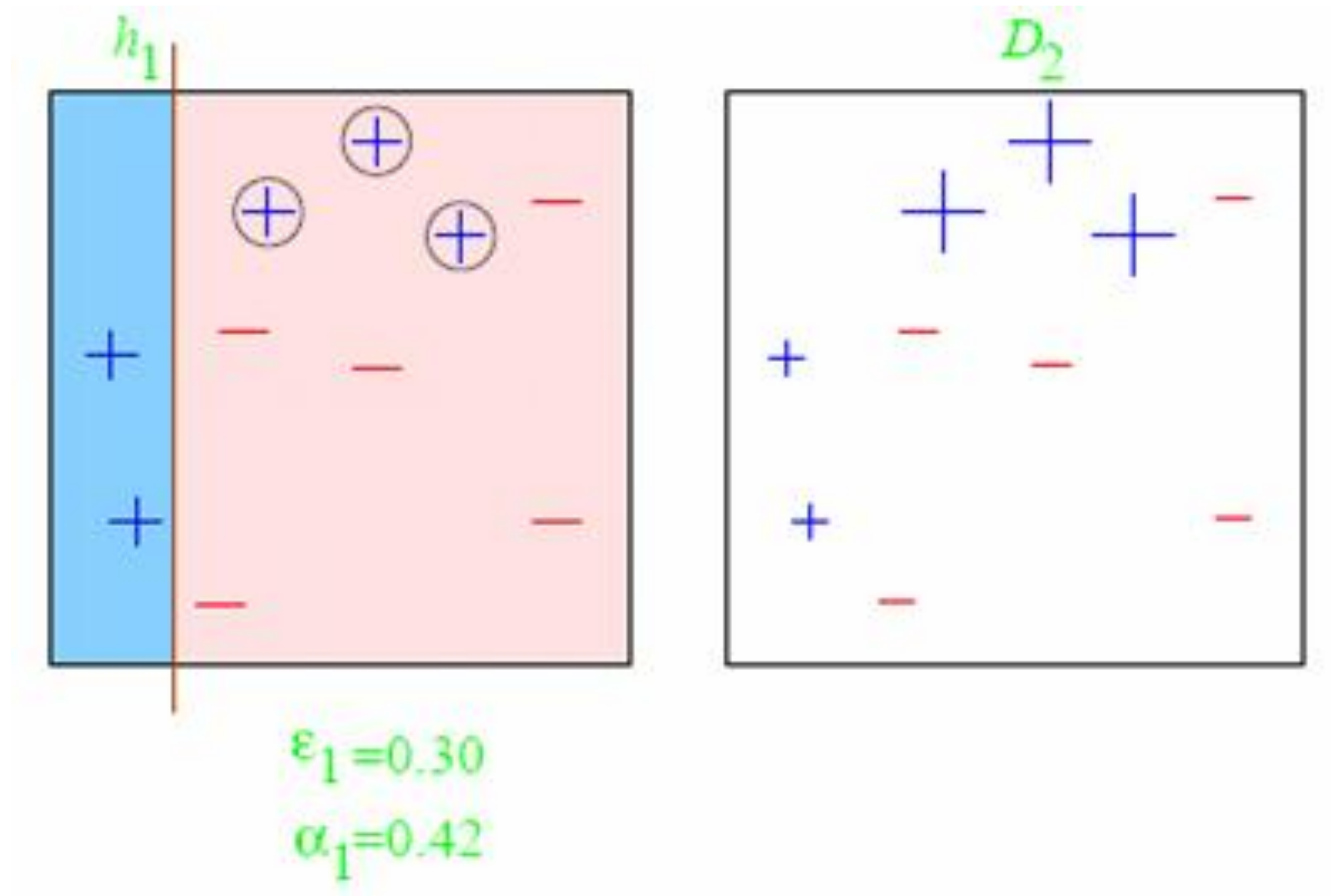
※ if $\mathcal{A}$ is weak but always slightly better than random ($\epsilon_t \le \epsilon < 0.5$),

※ then (AdaBoost+ $\mathcal{A}$) can be strong ($E_{in} = 0$ and E_{out} small)

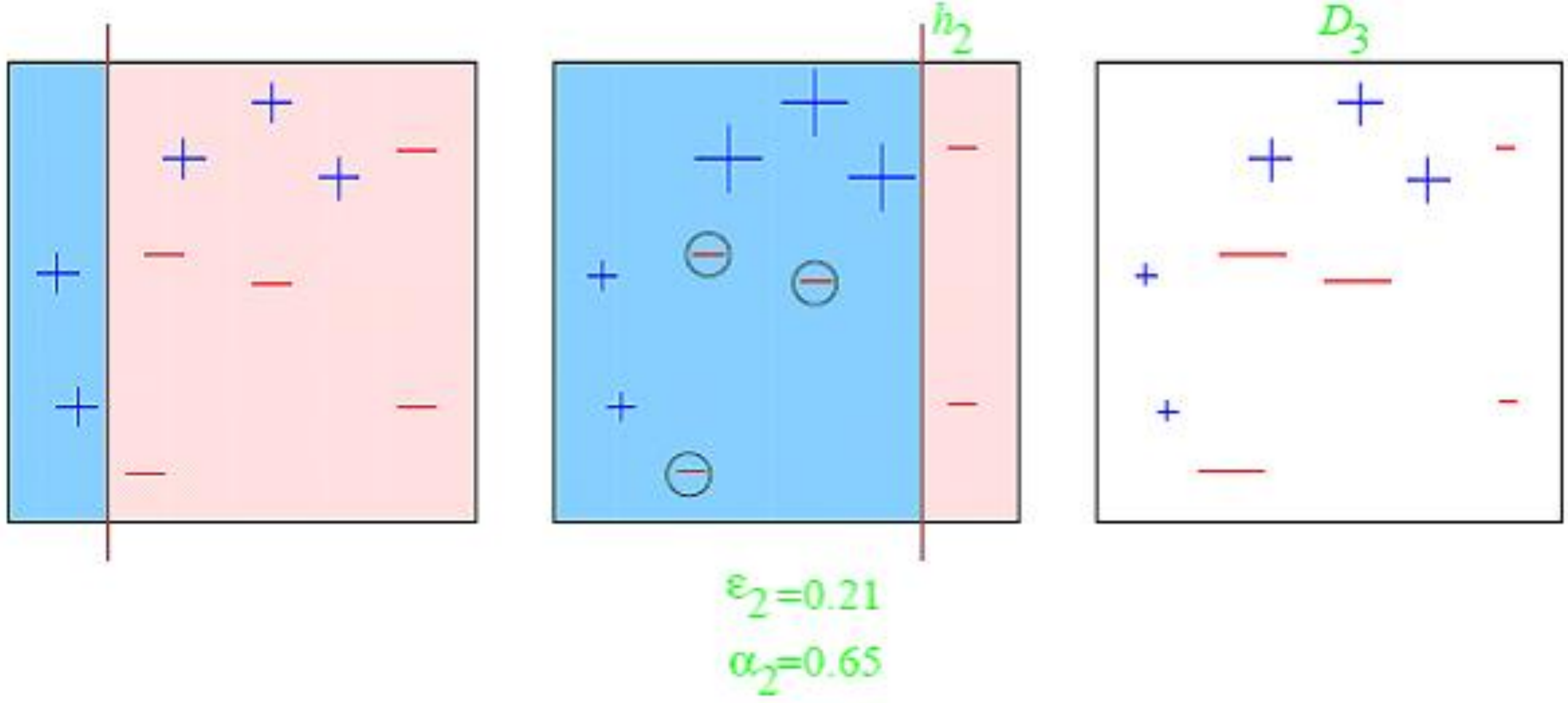
A toy example from Schapire's tutorial



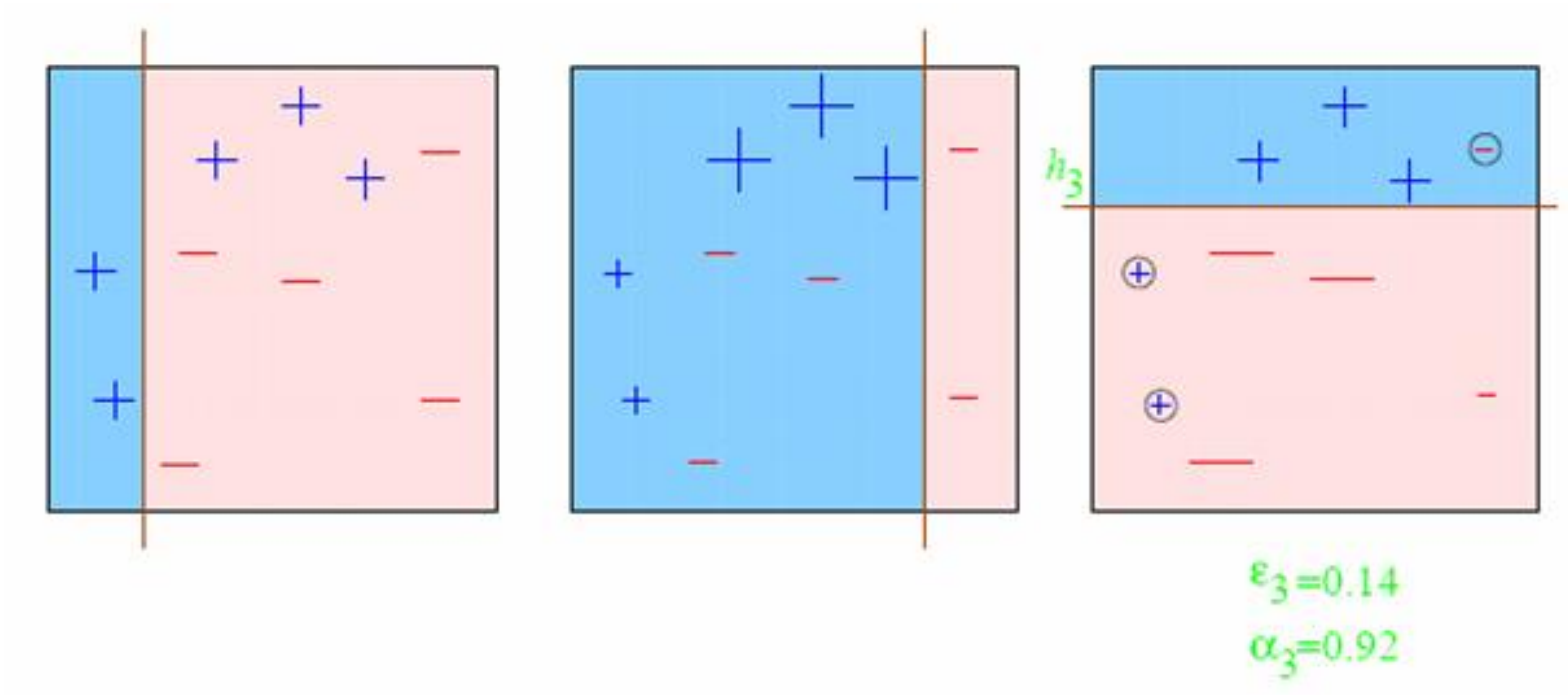
The first round:



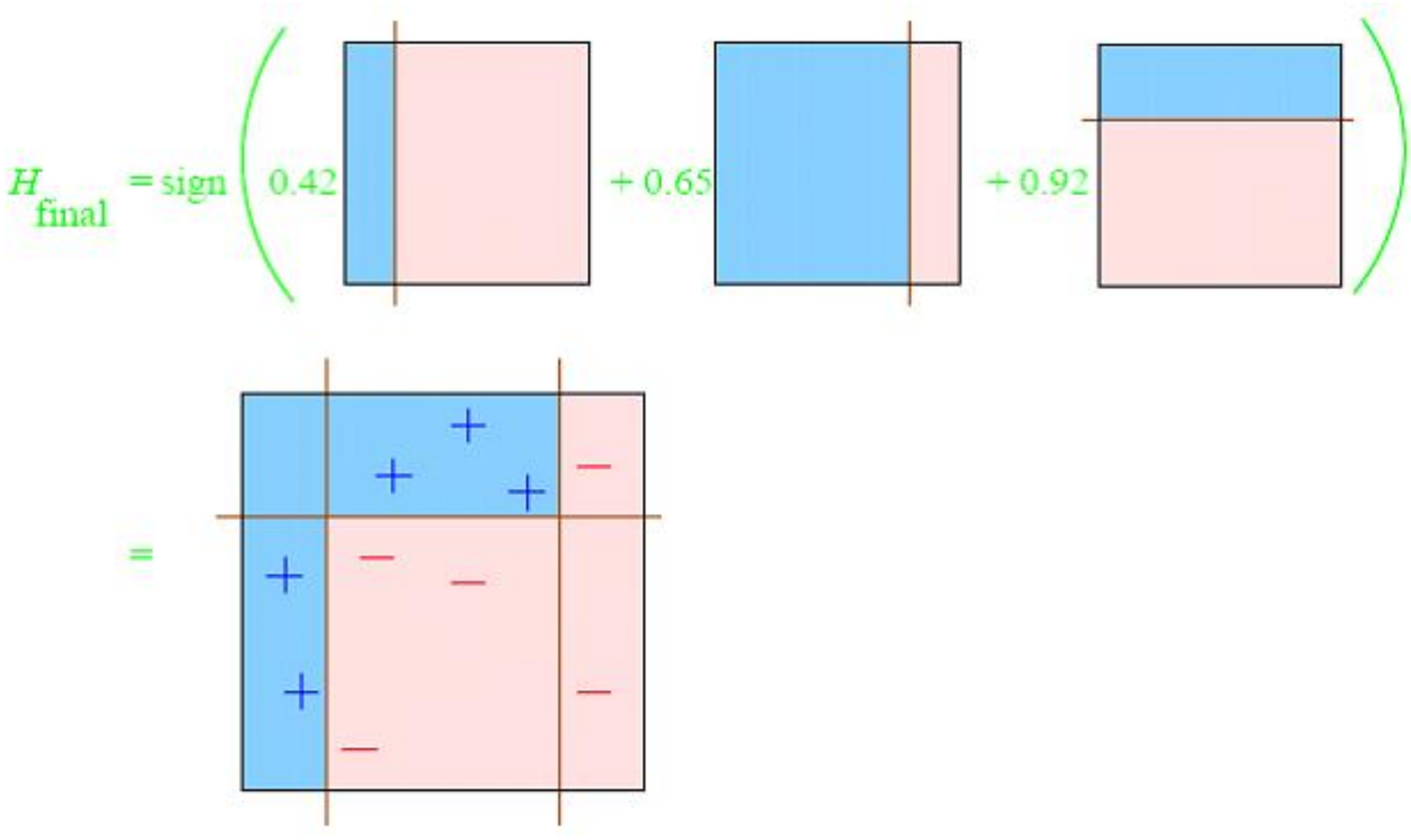
The second round:



The third round:



The final classifier



AdaBoost Algorithm

- $\alpha^{(1)} = [1/N, 1/N, \ldots, 1/N]$ for $t = 1, 2, \cdots, T$
  1. obtain g_t by $\mathcal{A}(D, \alpha^{(t)})$

     where $\mathcal{A}$ tries to minimize $\alpha^{(t)}$-weighted 0/1 error
  2. update $\alpha^{(t)}$ to $\alpha^{(t+1)}$ by $\lambda_t = \sqrt{(1-\epsilon_t)/\epsilon_t}$

     incorrect examples : $\alpha^{(t+1)} = \alpha^{(t)} \cdot \lambda_t$

     incorrect examples : $\alpha^{(t+1)} = \alpha^{(t)} / \lambda_t$

     where: $\epsilon_t = \sum_{i=1}^{N} \alpha_i^t \cdot [\![y_i \neq g_t(\mathbf{x}_i)]\!] / \sum_{i=1}^{N} \alpha_i^t$
  3. compute $\beta_t = \ln(\lambda_t)$
- return $G(x) = \text{sign}\left(\sum_{t=1}^{T} \beta_t g_t(\mathbf{x}) \right)$

Quize

According to $\beta_t = \ln(\lambda_t)$ and $\lambda_t = \sqrt{(1-\epsilon_t)/\epsilon_t}$,

when would $\beta_t > 0$?

(1) $\epsilon_t < 0.5$ (2) $\epsilon_t > 0.5$ (3) $\epsilon_t \neq 1$ (4) $\epsilon_t \neq 0$

The math part should be easy for you, and it is interesting to think about the physical meaning: $\beta_t > 0$ (g_t is useful for G) if and only if the weighted error rate of g_t is better than random!

Reference Answer: 1

Quize

For a data set of size 9527 that contains $\mathbf{x}_n \in \mathbb{R}^{1024}$, after running AdaBoost-Stump for 1000 iterations, what is the number of distinct features within $\mathbf{x}$ that are effectively used by G?

$(\mathbf{A})$ $0 \leq \text{number} \leq 1000$

$(\mathbf{B})$ $1000 < \text{number} \leq 1024$

$(\mathbf{C})$ $1024 < \text{number} \leq 9527$

$(\mathbf{D})$ $9527 < \text{number}$

Each decision stump takes only one feature.

So 1000 decision stumps need at most 1000 distinct features.

Reference Answer: 1

Next chapter: Semi-supervised Learning