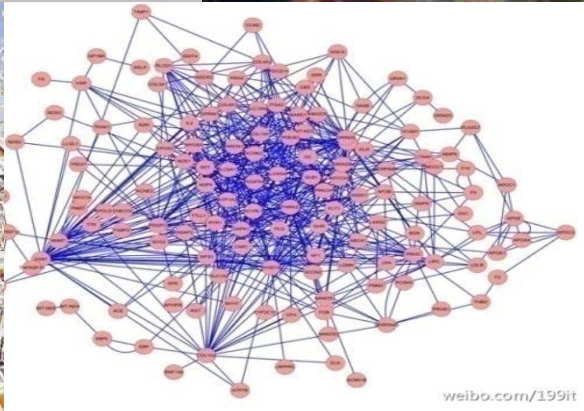


Chapter 4 Ensemble Model

2025 Autumn

Lei Sun

Framework of Ensemble

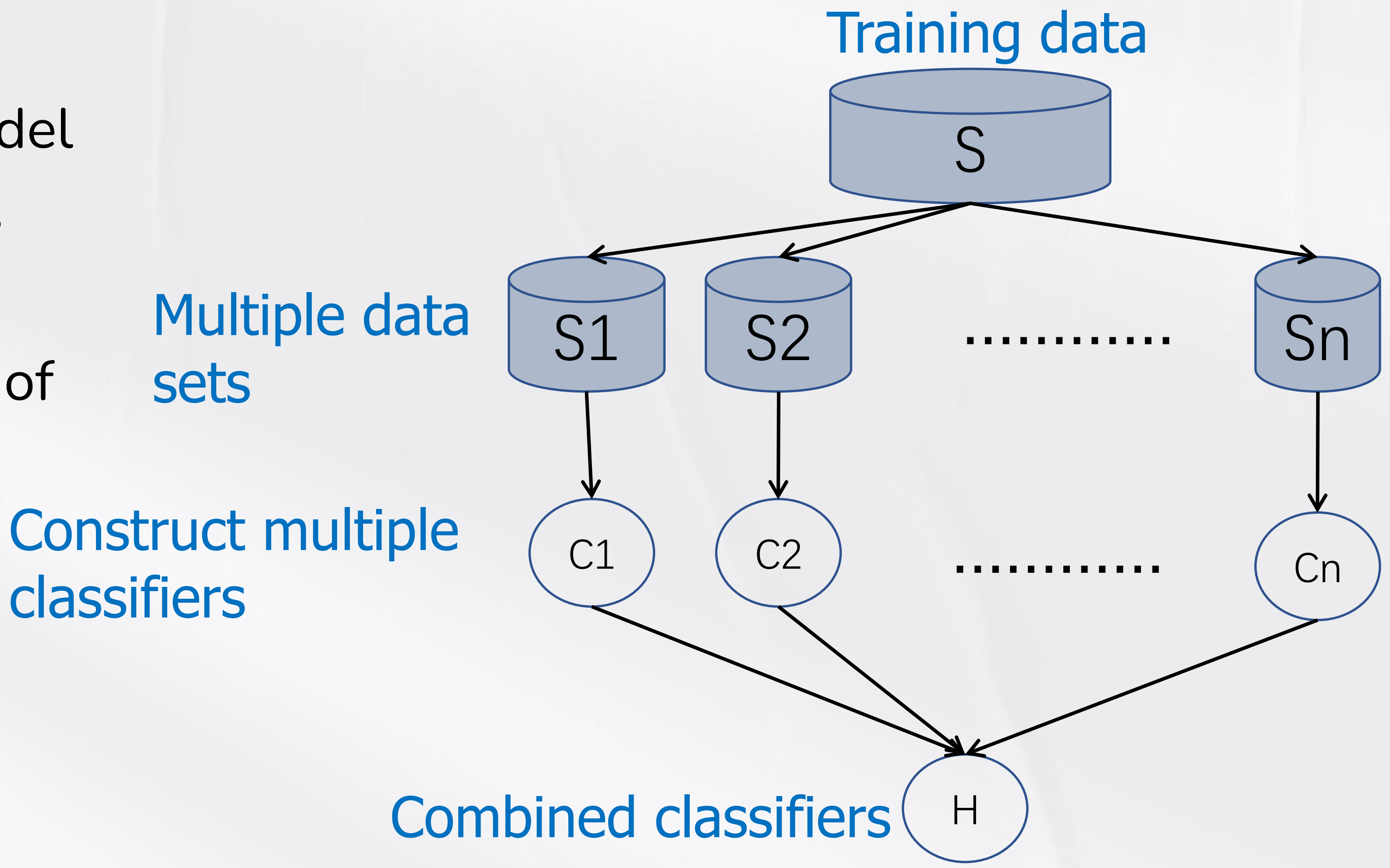
✓ Minimize the probability of model prediction errors on future data

Two competing methodologies

- Build one really good model
  - Traditional approach
- Build many models and average the results
  - Ensemble learning (more recent)

Framework of Ensemble

- ✓ Do not learn a single model but learn a set of models
- ✓ Combine the predictions of multiple models



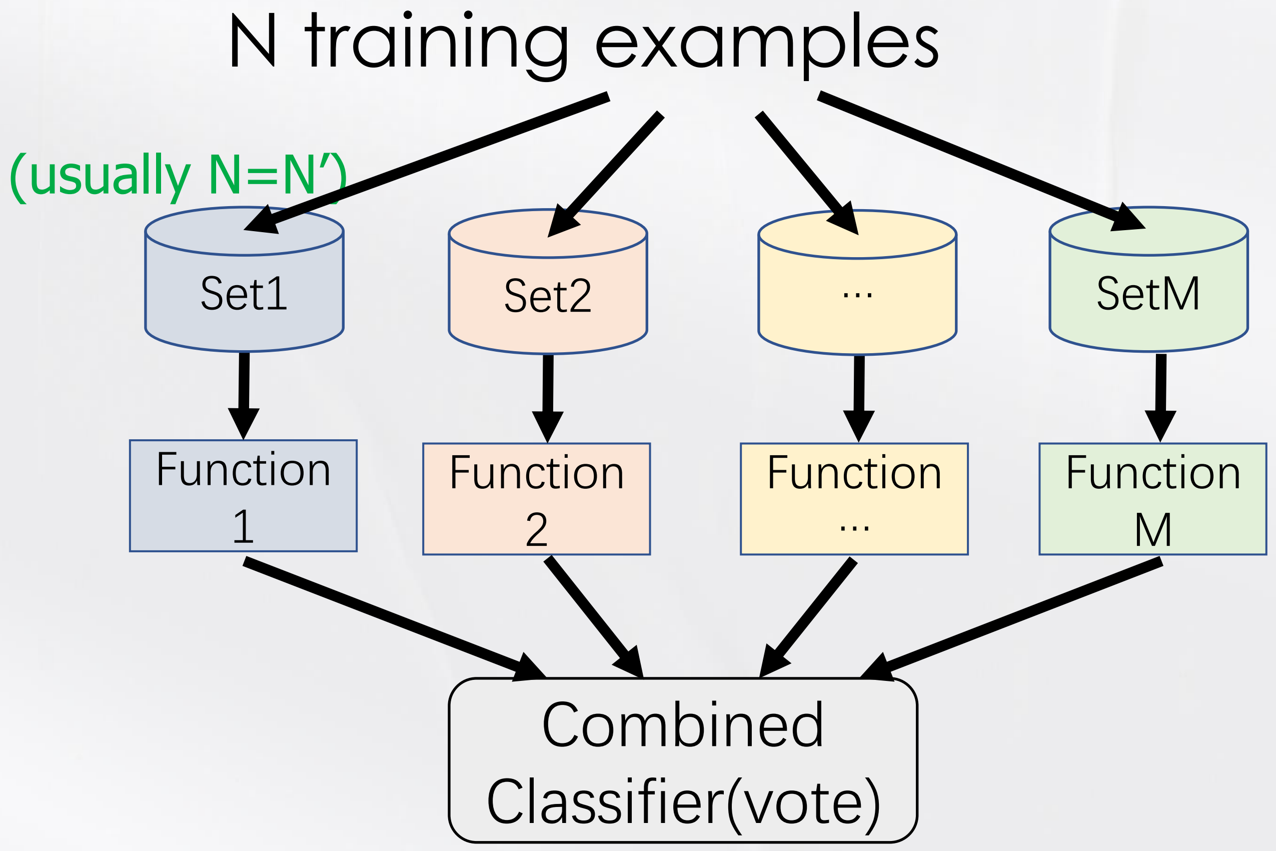
02

Bagging (Bootstrap aggregating)

①**Bootstrap Sampling:**

random ‘M’ subsets of original training data are sampled with replacement to ensure that the base models are trained on diverse subsets, as some samples may appear multiple times in the new subset, while others may be omitted.

It reduces the risks of overfitting and improves the accuracy of the model.



To make the model computationally efficient and less time-consuming, the base models can be trained **in parallel.**

Bagging (Bootstrap aggregating)

② **Aggregation:** Once all the base models are trained, it is used to make predictions on the unseen data. The predicted class label for the given instance is chosen based on the majority voting. The class which has the majority voting is the prediction of the model.

③ **Out-of-Bag (OOB) Evaluation:** Some samples are excluded from the training subset of particular base models during the bootstrapping method. These "out-of-bag" samples can be used to estimate the model's performance without the need for cross-validation.

④ **Final Prediction:** After aggregating the predictions from all the base models, Bagging produces a final prediction for each instance.

Bagging (Bootstrap aggregating)

Idea： Obtaining Independent Classifier

Boostrap Sampling

- Given Sn with n training examples, create S_n' by sampling with replacement n examples
- Create m bootstrap datasets $S_n^1, S_n^2, \ldots.., S_n^m$

Bagging

- Training distinct classifier on each S_n^i
- Classify new examples by majority vote

Bagging (Bootstrap aggregating)

Why reduce variance?

n classifiers(trees) , the output of each tree is X_i, the variance of all the outputs is $Var(X_i)$,

Suppose after bagging, the output is an average $\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$,

variance $Var(\bar{X}) = Var\left(\frac{\sum_{i=1}^{n} X_i}{n}\right)$

When each tree is independent

$$Var(\bar{X}) = Var\left(\frac{\sum_{i=1}^{n} X_i}{n}\right) = \frac{1}{n^2} Var\left(\sum_{i=1}^{n} X_i\right)$$

$$= \frac{1}{n^2}\left(Var(x_1) + Var(x_2) + \cdots + Var(x_n)\right) = \frac{1}{n^2} n\sigma^2 = \frac{\sigma^2}{n} = \frac{Var(X_i)}{n} < Var(X_i)$$

Bagging (Bootstrap aggregating)

When does Bagging work?

- ✓ Bagging tends to reduce variance of the classifier
  - -- By voting, the ensemble classifier is more robust to noisy examples
- ✓ Bagging is most useful for classifiers that are
  - -- Unstable
    - Small changes in training set produce very different models
  - -- Prone to overfitting

RandomForest(随机森林)

Idea： Obtaining Independent Classifier

- ✓ Resampling training data is not sufficient
- ✓ The generalization error depends on the strength of the individual trees and the correlation between them
- ✓ Each tree depends on the values of a random vector and sampled independently

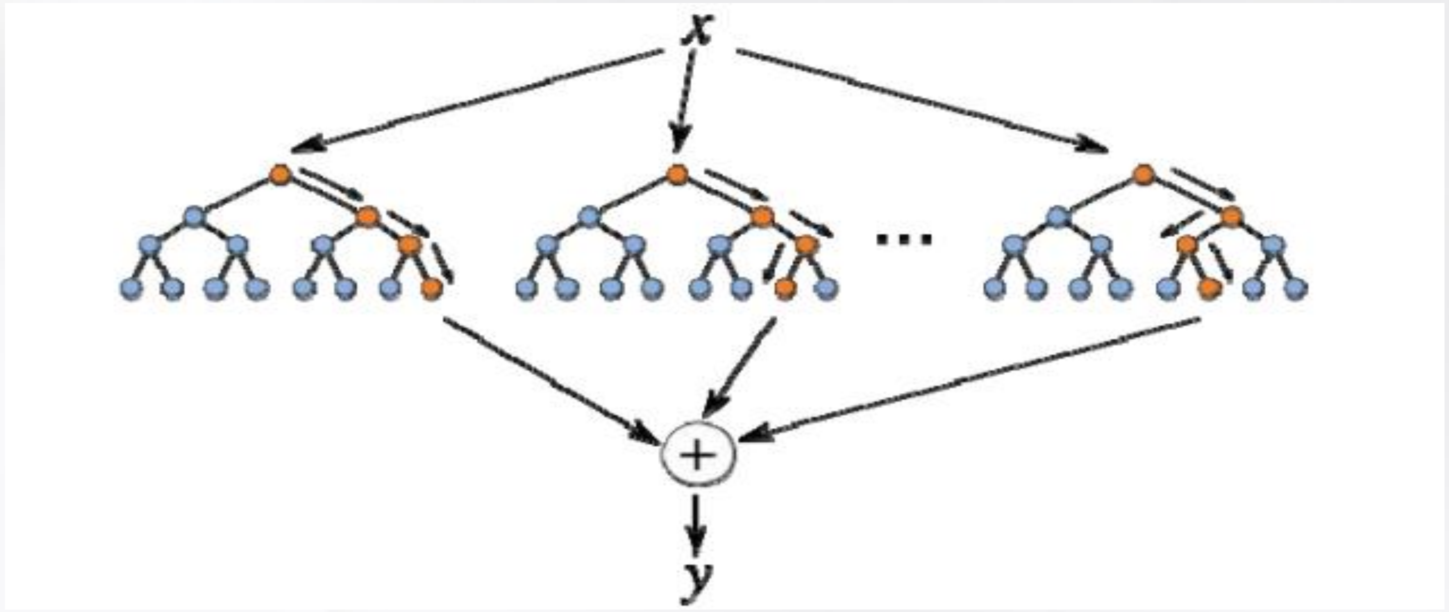
Randomly restrict the features used each split

RandomForest(随机森林)

Introduces two sources of randomness

- bagging
- random feature subset

  at each node, best split is chosen from random subset of k < d features



RandomForest(随机森林)

RF procedure

```
for  $b=1,\ldots,m$
        draw bootstrap samples $S_n^b$ of size $n$ from $S_n$
        grow random forest DT$^{(b)}$
output ensemble
```

[subprocedrure for growing DT $^{(b)}$]

Until stopping criteria are met, recursively repeat

① select *m* features at random from *F* features

② pick best feature to split on using info gain

③ split node into children

Make predictions according to majority vote of the set of *b* trees

RandomForest(随机森林)

Error rate depends on：

- ✓ Correlation between any two trees in the forest
  Higher correlation increases the forest error rate
- ✓ Error rate of each individual tree in the forest
  A tree with a low error rate decreases the forest error rate

Typical value of m:

Increasing *m* (suppose *F* is not changed)：

- decrease bias(increase accuracy) of each tree
- Increase correlation
- Increase variance

RandomForest(随机森林)

Variable importance

- ✓ Importance of a variable v
  - –For each tree, run the oob cases through the tree and find the number of correct classifications (nCorrectA)
  - –For a variable v, randomly permute (改变顺序) its values amongst the oob cases and run the oob cases through the tree; find the number of correct classifications (nCorrectP)
  - –Calculate (nCorrectA–nCorrectP) for the tree
  - –Average over all trees gives importance score of variable v
- ✓ Features which produces large values of this score are ranked as more important than features which produce small values.

RandomForest(随机森林)

Key Features of Random Forest

① **High Predictive Accuracy:** the averaging of uncorrelated trees lowers the overall variance and prediction error.

② **Reduce variance**

③ **Resistance to Overfitting with enough trees:** This approach helps prevent getting too caught up with the training data which makes the model less prone to overfitting.

④ **Variable Importance Assessment:** identifies the average decrease in accuracy by randomly permutating the feature values in oob samples.

⑤ **Parallelization for Speed**

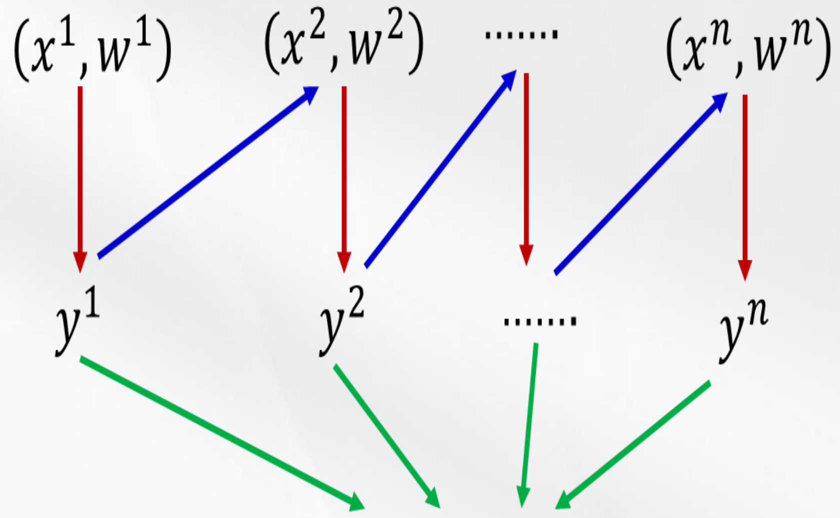
AdaBoosting(提升)

✓ Framework of boosting

- Obtain the first classifier $f_1(x)$
- Find another function $f_2(x)$ to help $f_1(x)$
  - However, if $f_2(x)$ is similar to $f(x_1)$, it will not help a lot
  - We want $f_2(x)$ to be complementary with $f_1(x)$ (HOW?)
- Obtain the second classifier
- … Finally, combine all the classifiers

By focusing on the errors and constantly learning from them, it creates a strong model from a series of weak classifiers. The classifiers are learned sequentially

AdaBoosting(提升)

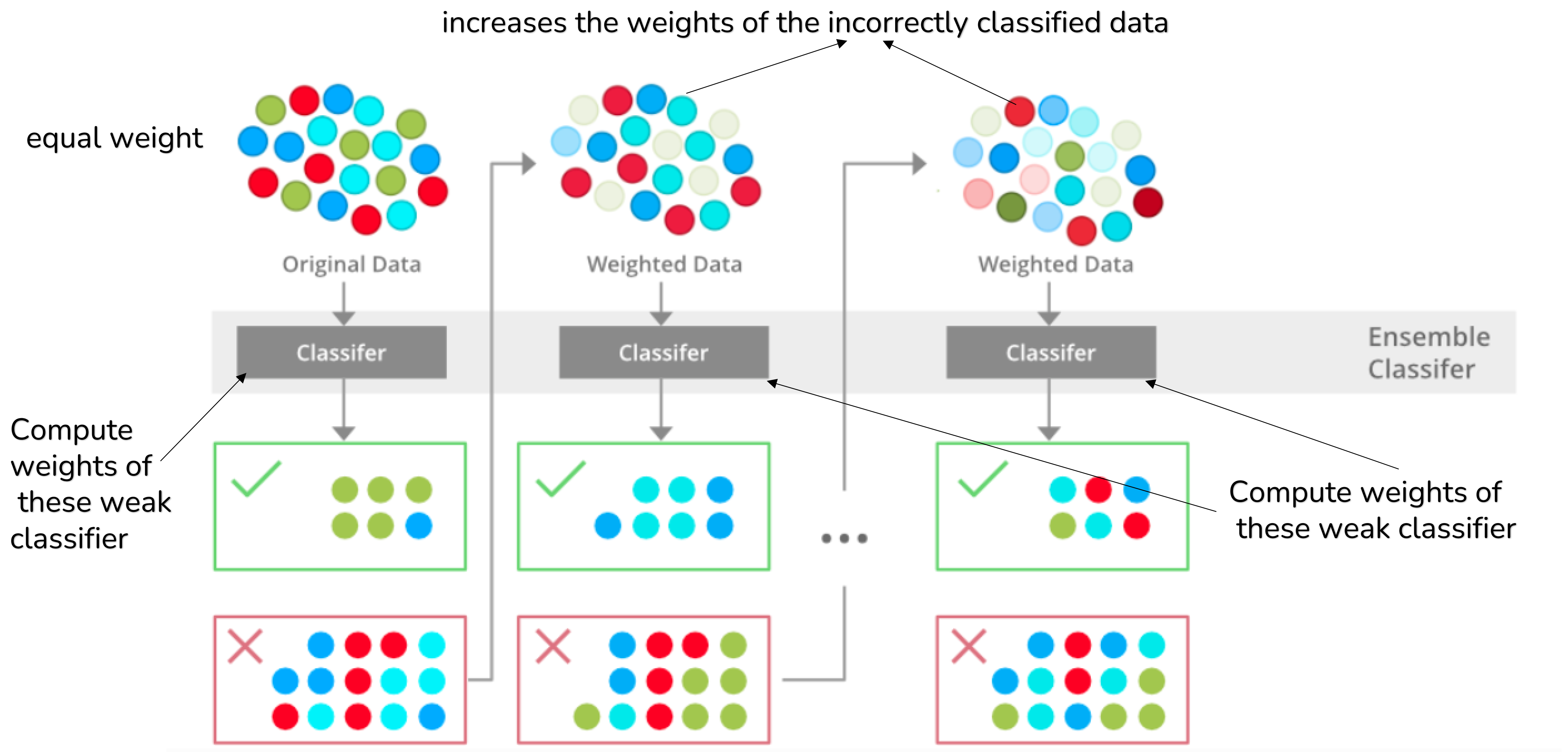


$$H(x) = sign\left[\sum_{i=1}^{n} \alpha_i y^i(x)\right]$$

linear combination

It works in the following steps:

① all data points are given an equal weight. selects a training subset randomly
② iteratively trains the AdaBoost model by selecting the training set based on the accurate prediction of the last training
③ assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification
④ assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight
⑤ This process iterates until reached to the specified maximum number of estimators
⑥ In the end, combines all the weak learners into a single model. It gives more important to the weak learners that had a better performance with a lower error rate.


increases the weights of the incorrectly classified data
equal weight
Original Data
Weighted Data
Weighted Data
Classifer
Classifer
Classifer
Ensemble Classifer
Compute weights of these weak classifier
Compute weights of these weak classifier

AdaBoosting(提升)

How to get different classifiers?

✓ Training on different training data sets

✓ How to have different training datasets?

- Re-sampling your training data to form a new set
- Re-weighting your training data to form a new set

✓ How to obtain different classifiers based on weighted datasets?

$(x^1, y^1, w^1)\ w^1 = \cancel{\frac{1}{n}}$ 0. 4

$(x^2, y^2, w^2)\ w^2 = \cancel{\frac{1}{n}}$ 0. 6

$(x^n, y^n, w^n)\ w^n = \cancel{\frac{1}{n}}$ 0. 8

$$H(x) = sign\left[\sum_{i=1}^{n} \alpha_i y^i(x)\right]$$

- α_i is the weight applied to classifier $y^i(x)$

AdaBoosting(提升)

Idea

training $f_2(x)$ on the training set that fails $f_1(x)$

How to find a training set that fails $f_1(x)$?

the error rate of $f_1(x)$ on its training data

$$\varepsilon_1 = \frac{\sum_n w_1^n \delta(f_1(x^n) \neq y^n)}{\sum_n w_1^n} \qquad \varepsilon_1 < 0.5$$

Changing the example weights from w_1^n to w_2^n such that

$\frac{\sum_n w_2^n \delta(f_1(x^n) \neq y^n)}{\sum_n w_2^n} = 0.5$ Training $f_2(x)$ based on the new weights w_2^n

AdaBoosting(提升)

Re-weighting Training Data

- training $f_2(x)$ on the training set that fails $f_1(x)$
- How to find the training set that fails

$$\begin{cases} \text{if } x_n \text{ misclassified by } f_1(f_1(x_n) \neq y^1) \\ w_2^n \leftarrow w_1^n \text{ times } d_1 \quad \uparrow \\ \text{if } x_n \text{ corerectly classified by } f_1(f_1(x_n = y^1) \\ w_2^n \leftarrow w_1^n \text{ divided by } d_1 \quad \downarrow \end{cases}$$

$f_2(x)$ will be learned based on samples weights w_2^n

What is value d_1?/

04

AdaBoosting(提升)

Re-weighting Training Data

$$f_1(x_n) \neq y^1$$

$$w_2^n \leftarrow \ w_1^n \ \textit{times} \ d_1$$

$$f_1(x_n) = y^1$$

$$w_2^n \leftarrow \ w_1^n \ \textit{divided by} \ d_1$$

$$\varepsilon_1 = \frac{\sum_n w_1^n \delta(f_1(x_n) \neq y^n)}{\sum_n w_1^n}$$

$$\frac{\sum_n w_2^n \delta(f_2(x_n) \neq y^n)}{\sum_n w_2^n} = 0.5$$

$$= \sum_{f_1(x_n)\neq y^1} w_1^n d_1$$

$$= \sum_{f_1(x_n)\neq y^1} w_1^n d_1 + \sum_{f_1(x_n)= y^1} w_1^n / d_1$$

$$\frac{\sum_{f_1(x_n)\neq y^1} w_1^n d_1}{\sum_{f_1(x_n)\neq y^1} w_1^n d_1 + \sum_{f_1(x_n)= y^1} w_1^n / d_1} = 0.5$$

04

AdaBoosting(提升)

$$\frac{\sum_{f_1(x_n)\neq y^1} w_1^n d_1 + \sum_{f_1(x_n)=y^1} w_1^n / d_1}{\sum_{f_1(x_n)\neq y^1} w_1^n\, d_1} = 2$$

$$\frac{\sum_{f_1(x_n)=y^1} w_1^n / d_1}{\sum_{f_1(x_n)\neq y^1} w_1^n\, d_1} = 1$$

$$\sum_{f_1(x_n)=y^1} w_1^n / d_1 = \sum_{f_1(x_n)\neq y^1} w_1^n\, d_1$$

$$\frac{1}{d_1} \sum_{f_1(x_n)=y^1} w_1^n = d_1 \sum_{f_1(x_n)\neq y^1} w_1^n$$

$$\frac{1}{d_1}(1-\varepsilon_1) \textstyle\sum_n w_1^n = d_1 \boxed{\varepsilon_1 \textstyle\sum_n w_1^n}$$

see $\varepsilon_1 = \dfrac{\sum_n w_1^n \delta(f_1(x_n) \neq y^n)}{\sum_n w_1^n}$

$$\frac{1}{d_1}(1-\varepsilon_1) = d_1\ \varepsilon_1$$

$$d_1 = \sqrt{(1-\varepsilon_1)/\varepsilon_1} \quad >1$$

Algorithm

- Giving training data $\{(x_1, y^1, w_1^1), \dots, (x_n, y^n w_1^n), \dots, (x_n, y^n w_1^n)\}$

  $y = \pm 1$(Binary classification), $w_1^n = \frac{1}{N}$(equal weight)

- **For** t=1,…, T:

  Training week classifier $f_t(x)$ with weights $\{w_t^1, \dots, w_t^N\}$

  ε_t is the error rate of $f_t(x)$ with weights $\{w_t^1, \dots, w_t^N\}$

  If $\varepsilon_t > 0.5$, choose next $f_t(x)$ else

  - For n=1,…, N:

    If x_n is misclassified by $f_t(x)$ $\quad y^n \neq f_t(x_n)$

    $w_{t+1}^n = w_t^n \times d_t = w_t^n \exp(\alpha_t)$ $\quad d_t = \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$

    else:

    $w_{t+1}^n = w_t^n / d_t = w_t^n \exp(-\alpha_t)$ $\quad \alpha_t = \ln \sqrt{(1 - \varepsilon_t)/\varepsilon_t}$

$$w_{t+1}^n = w_t^n \times \exp(-y^n f_t(x^n)\alpha_t)$$

Algorithm

- We obtain a set of functions: $f_1(x), \dots, f_t(x), \dots, f_T(x)$
- How to aggregate them?
  - Non-uniform weight:
  - $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

$$\alpha_t = \ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$w_{t+1}^n = w_t^n \times \exp(-y^n f_t(x^n)\alpha_t) \searrow$$

Renormalize weights

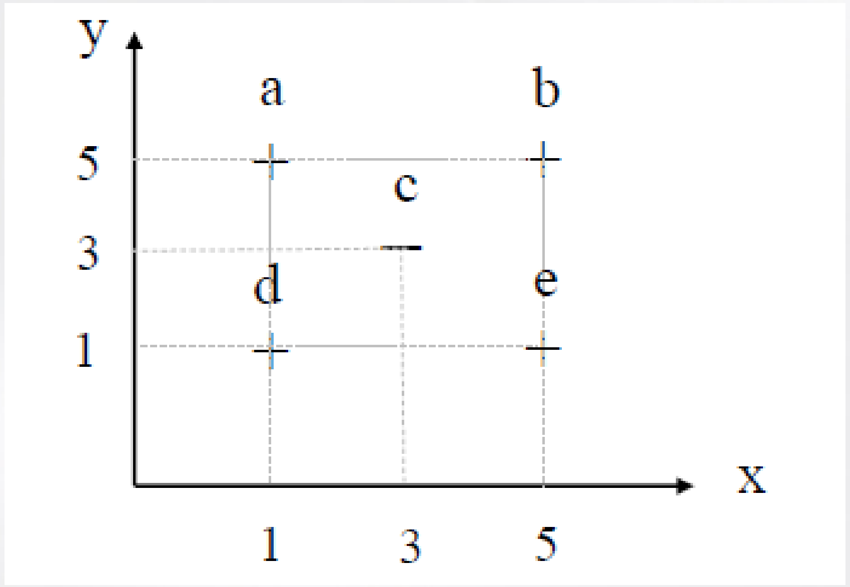
$$e.g.\quad \varepsilon_t = 0.1 \qquad \alpha_t = 1.10$$

$$\varepsilon_t = 0.4 \qquad \alpha_t = 0.20$$

Smaller error, larger weight for final voting.

As we have more and more f(x), H(x) achieves smaller and smaller error rate on training data

AdaBoosting(提升)



If $x<2$ then+

If $x>2$ then +

If $x<4$ then +

If $x>4$ then +

If $x>6$ then +

If $x<6$ then+

- ✓ For the purpose of illustration, we only consider the tree stumps on variable x. But for the complete approach, you need to consider all possible decision tree stumps on both x and y. Suppose $x< a$ indicates the tree stump "if $x< a$ then + otherwise –" or $x>a$ indicates the tree stump "if $x> a$ then + otherwise –

Weights	Round1	Round 2	Round 3
w_a	1/5	0.125	1/12
w_b	1/5	0.125	3/12
w_c	1/5	0.5	4/12
w_d	1/5	0.125	1/12
w_e	1/5	0.125	3/12

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$w_{t+1}^n = w_t^n \times \exp(-y^n f_t(x^n)\alpha_t)$$

$$\alpha_t = \ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

Tree stumps	Misclassified points	Error for round1	Error for round2	Error for round2
x<2	b, e	2/5	0.25	0.5
✖ x<4	b,c,e	3/5	0.75	0.83
x<6	c	1/5	0.5	0.33
✖ x>2	a,c,d	3/5	0.75	0.5
x>4	a,d	2/5	0.25	0.17
✖ x>6	a,b,d,e	4/5	0.5	0.67
	α	0.69	0.55	0.81

$$\varepsilon = \frac{1}{5}$$

$$d = 2$$

$$W_c = \frac{1}{5} \times 2 = \frac{2}{5}$$

$$W_{a,b,d,e} = \frac{1}{5} \div 2 = \frac{1}{10}$$

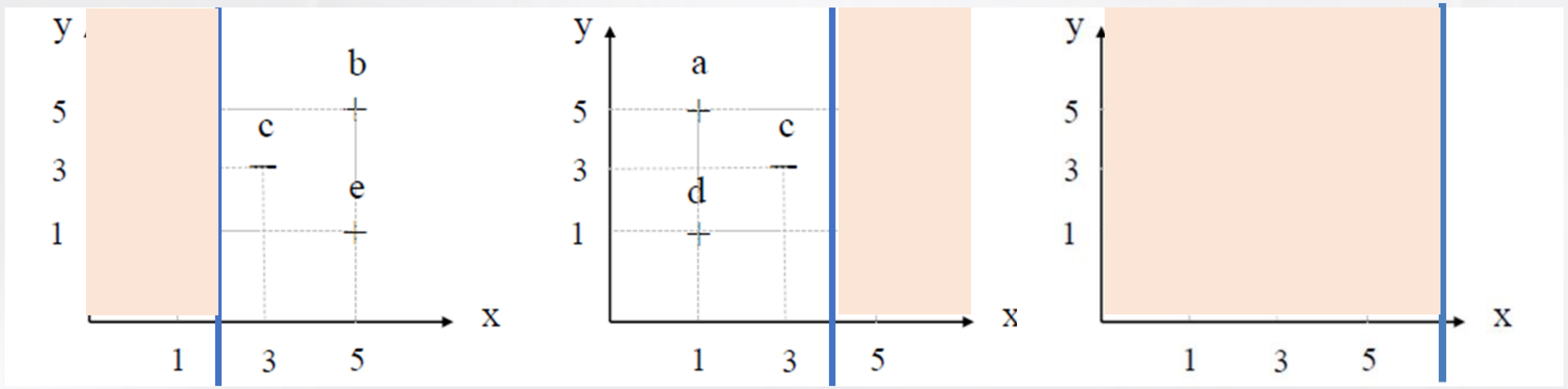
$$W_c = \frac{0.4}{0.4 + 4 \times 0.1} = 0.5$$

$$w_{a,b,d,e} = \frac{0.1}{0.8} = 0.125$$

$$H(X) = sign(0.693 \times (x < 6) + 0.550 \times (x < 2) + 0.805 \times (x > 4))$$

AdaBoosting(提升)

$$H(X) = sign(0.693 \times (x < 6) + 0.550 \times (x < 2) + 0.805 \times (x > 4))$$



0.550(x<2) 0.805(x>4) 0.693(x<6)

- The weak learners in AdaBoost are decision tree with a single split, called decision stumps.
- AdaBoost works by putting more weight on difficult to classify instances and less on those already handled well.

AdaBoosting(提升)

Advantages

- ✓ **Improved Accuracy** (**decrease bias**) **:** Boosting can improve the accuracy of the model by combining several weak models' accuracies and averaging them for regression or voting over them for classification to increase the accuracy of the final model.
- ✓ **Robustness to Overfitting:** Boosting can reduce the risk of overfitting by reweighting the inputs that are classified wrongly.
- ✓ **Better handling of imbalanced data:** Boosting can handle the imbalance data by focusing more on the data points that are misclassified
- ✓ **Better Interpretability:** Boosting can increase the interpretability of the model by breaking the model decision process into multiple processes.

04

AdaBoosting(提升)

Summary

✓ Boosting is a sequential technique.
  - The first algorithm is trained on the dataset (the same size as the original dataset)by replacement. And the subsequent algorithms are built by fitting the residuals of the first model, thus giving higher weight to those observations that are poorly predicted by the previous model.

✓ Relay on a series of weak learners.

✓ There is not much parameters to tune (except the number of runs).

✓ It is flexible (we can combine any learning algorithm).

✓ Boosting Algorithms are vulnerable to the outliers

✓ No prior knowledge is needed about weak learner.

04

AdaBoosting(提升)

Boosting	Bagging
The main aim of boosting is to decrease bias, not variance	The main aim of bagging is to decrease variance not bias
At every successive layer Models are weighted according to their performance	All the models have the same weightage
New Models are influenced by the accuracy of previous Models	All the models are independent of each other

AdaBoosting(提升)

Adaboost算法的主要目的是什么?

A. 降低分类器方差

B. 提高单个弱学习器的性能

C. 通过组合多个弱学习器来构建一个强学习器

D. 优化训练时间