Never Stand Still

Ensemble Learning

集成学习

COMP9417 Machine Learning & Data Mining

Term 1, 2020

Adapted from slides by Dr Michael Bain

Aims

This lecture will develop your understanding of ensemble methods in machine learning, based on analyses and algorithms covered previously. Following it you should be able to:

- Describe the framework of the bias-variance decomposition and some of its practical implications
- describe how ensembles might be used to address the bias and variance components of error
- outline the concept of the stability of a learning algorithm
- describe different ensemble methods of bagging, randomization, boosting, etc.

Introduction

In previous lectures, introduced some theoretical ideas about limits on machine learning. But do these have any practical impact ?

The answer is **yes** !

One of these theoretical tools is bias-variance decomposition:

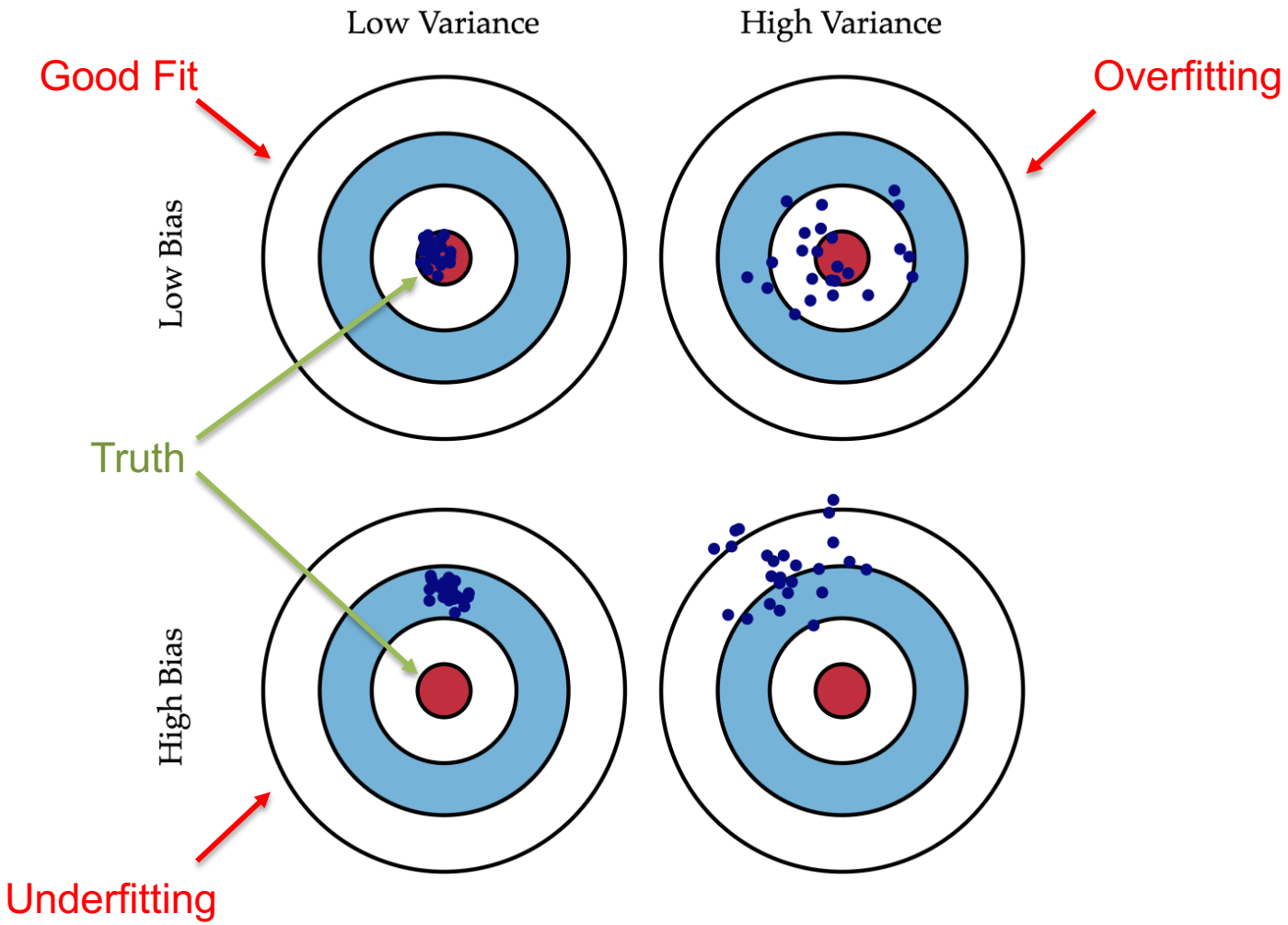
- The bias-variance decomposition of error can be a tool for thinking about how to reduce error in learning
- Take a learning algorithm and ask:
  - how can we reduce its bias ?
  - how can we reduce its variance ?
- Ensemble learning methods can be viewed in this light
- A form of *multi-level* learning: learning a number of base-level models from the data, and learning to combine these models as an ensemble

Review: bias-variance decomposition

- Theoretical tool for analyzing how much specific training set affects performance of classifier
- Assume we have an infinite number of classifiers built from different training sets all of the same size:
  - The *bias* of a learning scheme is the expected error due to the mismatch between the learner's hypothesis space (class of models) and the space of target concepts
  - The *variance* of a learning scheme is the expected error due to differences in the training sets used
    - The *variance* of a learning scheme is how much the predictions for a given point vary between different model.
  - $\text{Total expected error} \approx \text{bias}^2 + \text{variance}$

Bias-Variance

根据相同算法、不同数据集训练出的模型，对同一个样本进行预测；每个模型作出的预测相当于是一次打靶



Source: Scott-Fortmann,Understanding Bias-variance tradeoff

Bias-Variance

- The inability of the learning algorithm to capture the true relationship between the output and the features/attributes is called **bias**.
- The learning algorithm difference in fits between datasets is called **variance**.

Three commonly used methods to find a good bias-variance tradeoff are:

- Regularization
- Bagging
- Boosting

权衡

Bias-variance: a trade-off

Easier to see with regression in the following figure[1](to see the details you will have to zoom in in your viewer):

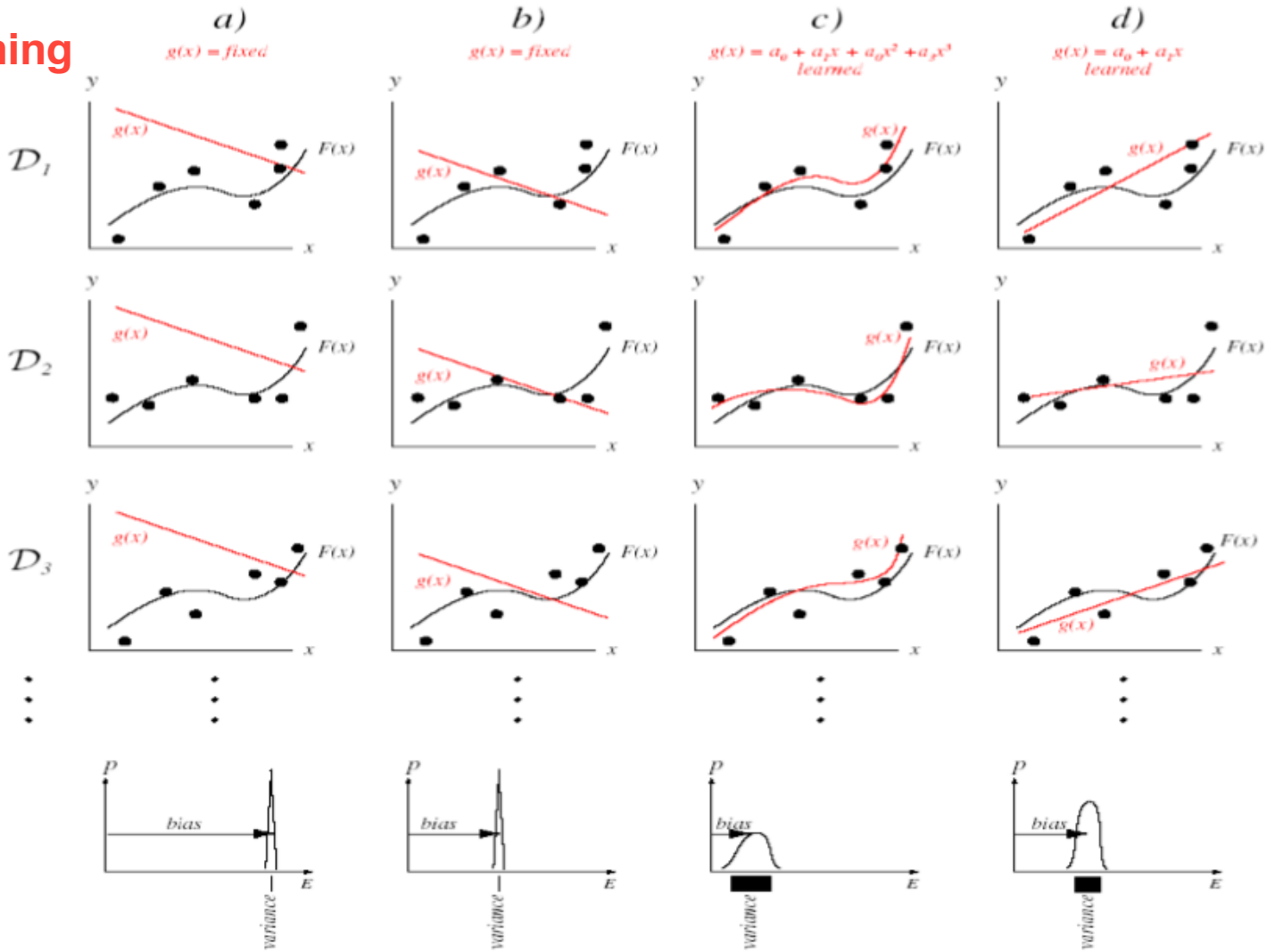
- each column represents a different model class $g(\mathrm{x})$ shown in red
- each row represents a different set of $n = 6$ training points, D_i, randomly sampled from target function $F(\mathrm{x})$ with noise, shown in black
- probability functions of mean squared error E are shown

1- from: “Elements of Statistical Learning” by Hastie, Tibshirani and Friedman (2001

Bias-variance: a trade-off

黑色曲线：
actual fuction

Di：
different training set



红色曲线：
generated model

high bias：红色预测值和黑色实际值差很多
zero variance：对于不同的training set红色预测值都一样

Bias-variance: a trade-off

- "a" is very poor: a linear model with fixed parameters independent of training data; high bias, zero variance
- "b" is better: a linear model with fixed parameters independent of training data; slightly lower bias, zero variance
- "c" is a cubic model with parameters trained by mean-square-error on training data; low bias, moderate variance
- "d" is a linear model with parameters adjusted to fit each training set; intermediate bias and variance
- training with data n $\rightarrow\infty$ would give
  - "c" with bias approaching small value due to noise
  - but not "d"
  - variance for all models would approach zero

Bias-variance in ensemble classification

- Recall that we derived the bias-variance decomposition for regression (squared-error loss function)
- Cannot apply same derivation for classification (zero-one loss can be used)
- Bias-variance decomposition used to analyze how much restriction to a single training set affects performance
- Can decompose expected error of any individual ensemble member as follows:
  - Bias = expected error of the ensemble classifier on new data
  - Variance = component of the expected error due to particular training set being used to build classifier

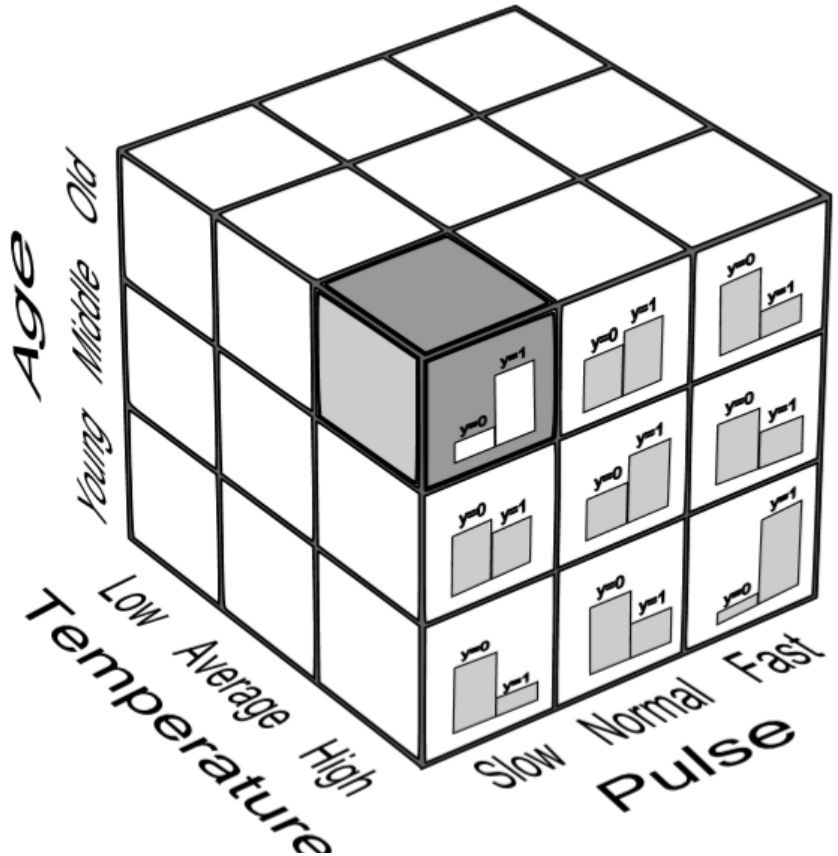
Bias-variance with "Big Data"

- high bias algorithms often used for efficiency
  - why ?
- big data can reduce variance
  - why ?

This slide and the following 3 slides are due to Prof. G. Webb, Monash U.

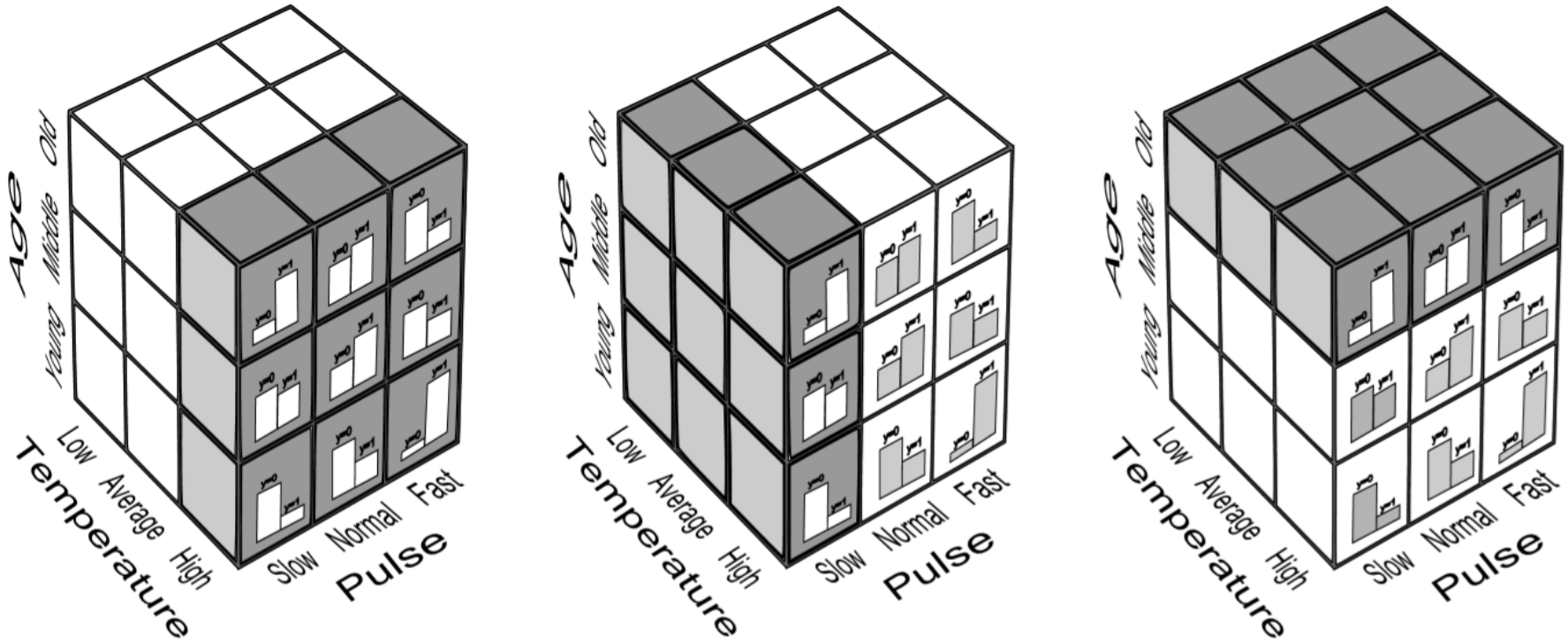
Bias-variance with “Big Data”

Suppose we have a low bias representation (e.g., all conjunctive concepts), but such concepts may not always occur frequently in small datasets:



Bias-variance with “Big Data”

So we can increase bias – e.g., by Naive Bayes-type conditional independence assumptions – but this forces averaging of class distributions over all “small concepts”:



Bias-variance with “Big Data”

困境

“Big Data” may help to resolve the bias-variance dilemma:

- high bias algorithms are often used for efficiency
  - usually simpler to compute
- big data can reduce variance
  - “small” concepts will occur more frequently
  - low bias algorithms can be applied
  - but: how to compute efficiently ?

This is still largely an open problem!

Bias-variance in “Real-world AI”

Imagine the following situation:

- Applications increasingly require machine-learning systems to perform at “human-level” (e.g., personal assistants, self-driving vehicles, etc.)
- Suppose you are developing an application and you know what “human-level” error would typically be on this task.
- You have sufficient data for training and validation datasets, and you are not restricted in terms of the models that you could learn (e.g., from linear regression or classification up to kernel methods, ensembles, deep networks, etc.)
- How can an understanding of the bias-variance decomposition help ?

Bias-variance in "Real-world AI"

The following scenarios can happen:

1. Training-set error is observed to be high compared to human-level – why ?
   - Bias is too high – solution: move to a more expressive (lower bias) model

2. Training-set error is observed to be similar to human-level, but validation set error is high compared to human-level – why ?
   - Variance is too high – solution: get more data (!), try regularization, ensembles, move to a different model architecture

These scenarios are often found in applications of deep learning[2]

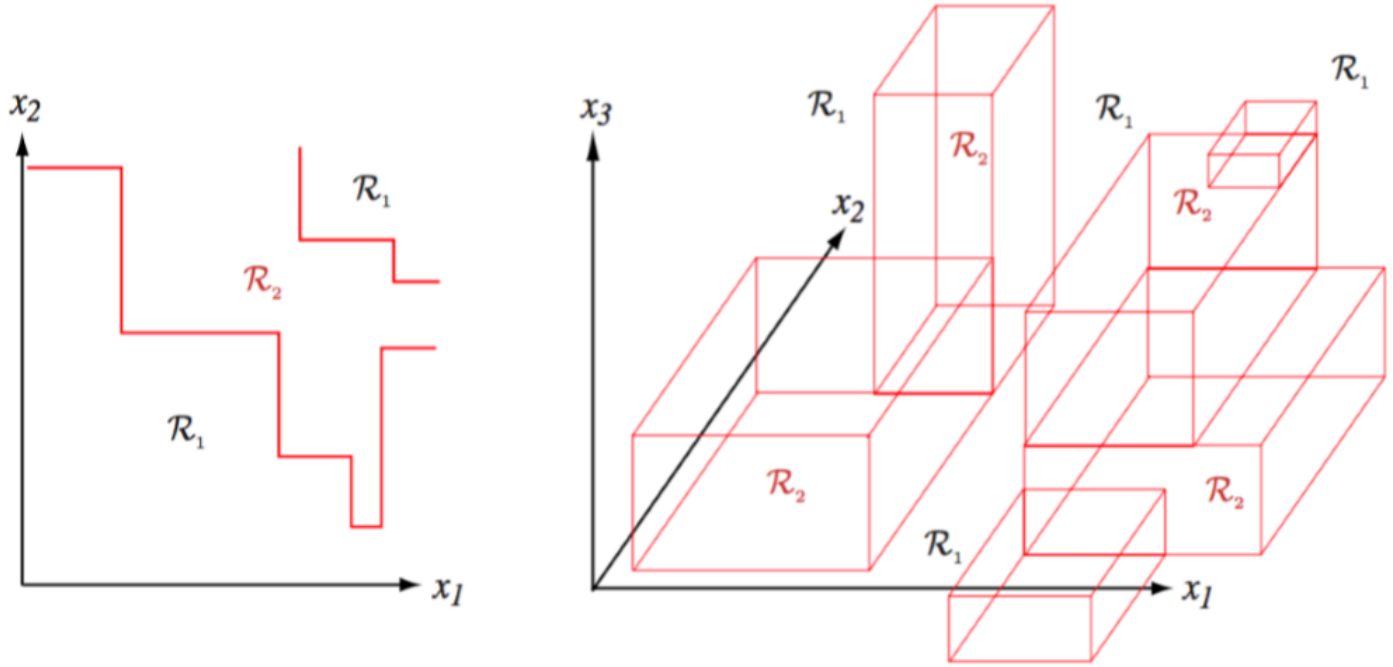
2-"Nuts and Bolts of Applying Deep Learning" by Andrew Ng
http://www.youtube.com/watch?v=F1ka6a13S9I

Stability

- for a given data distribution $\mathcal{D}$
- train algorithm L on training sets S_1, S_2 sampled from $\mathcal{D}$
- expect that the model from L should be the same (or very similar) on both S_1 and S_2
- if so, we say that L is a stable learning algorithm
- otherwise it is unstable
- typical stable algorithm: kNN (for some k)
- typical unstable algorithm: decision-tree learning

Turney, P. "Bias and the Quantification of Stability"

Decision boundaries in tree learning



Decision boundaries for monothetic two-class trees in two and three dimensions; arbitrarily fine decision regions for classes $\mathcal{R}_1$, $\mathcal{R}_1$ can be learned by recursively partitioning the instance space.

Instability of tree learning

An example shows the effect of a small change in the training data on the structure of an unpruned binary tree learned by CART. The training set has 8 instances for each class:

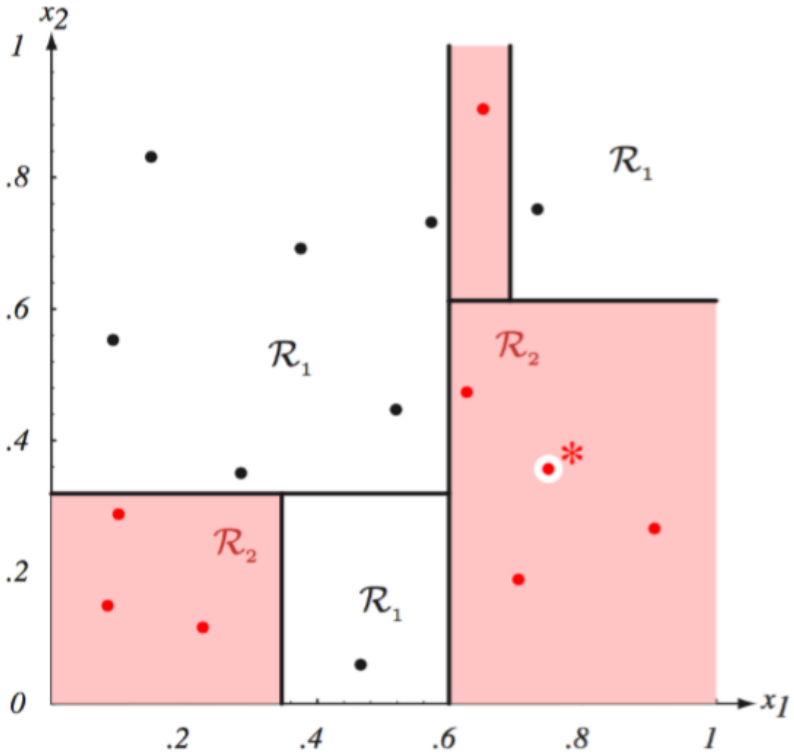
ω_1 (black)	
x_1	x_2
.15	.83
.09	.55
.29	.35
.38	.70
.52	.48
.57	.73
.73	.75
.47	.06

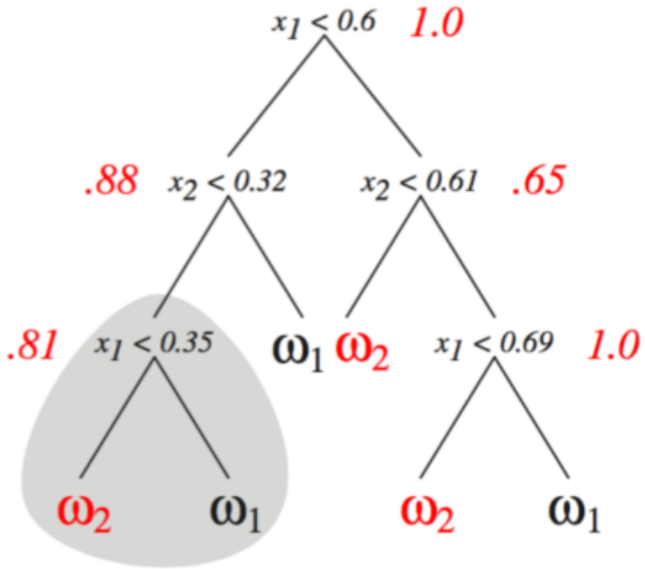
ω_2 (red)	
x_1	x_2
.10	.29
.08	.15
.23	.16
.70	.19
.62	.47
.91	.27
.65	.90
.75	.36* (.32†)

Note: for class ω_2 (red) the last instance has two values for feature x_2 . On the next slide is a tree learned from the data where this instance has value $x_2 = .36$ (marked *), and on the following slide we see the tree obtained when this value is changed to $x_2 = .32$ (marked †).

From: “Pattern Classification”. R. Duda, P. Hart, and D. Stork (2001) Wiley.

Instability of tree learning

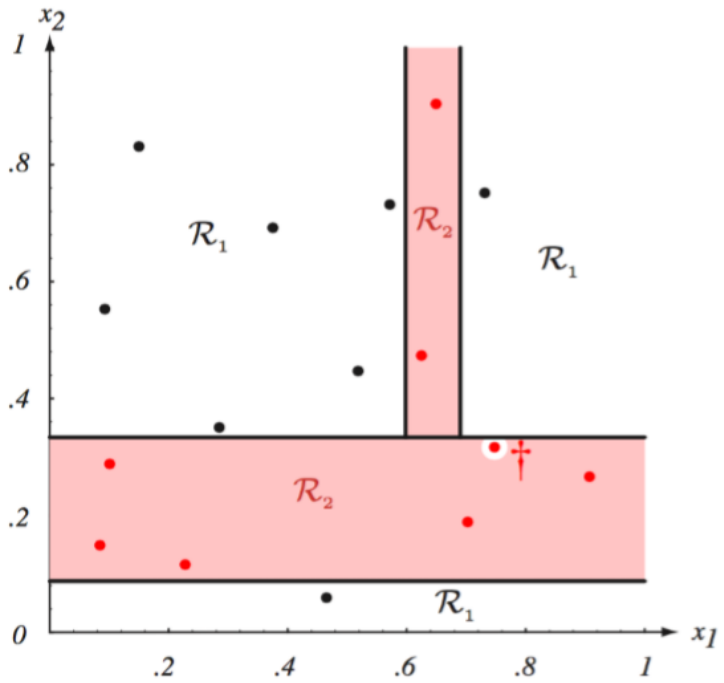


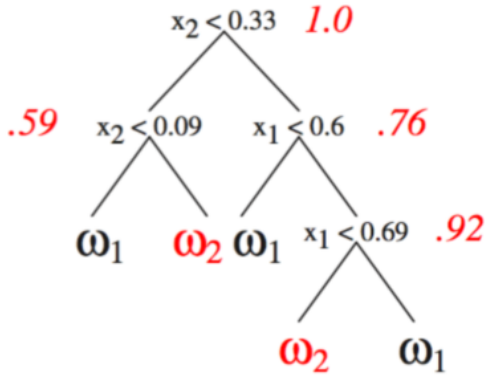


The partitioned instance space (left) contains the instance marked * and corresponds to the decision tree (right).

From: “Pattern Classification”. R. Duda, P. Hart, and D. Stork (2001) Wiley.

Instability of tree learning





The partitioned instance space (left) contains the instance marked † and corresponds to the decision tree (right). Note that both the decision boundaries and the tree topology are considerably changed, for example, testing x_2 rather than x_1 at the tree root, although the change in data was very small.

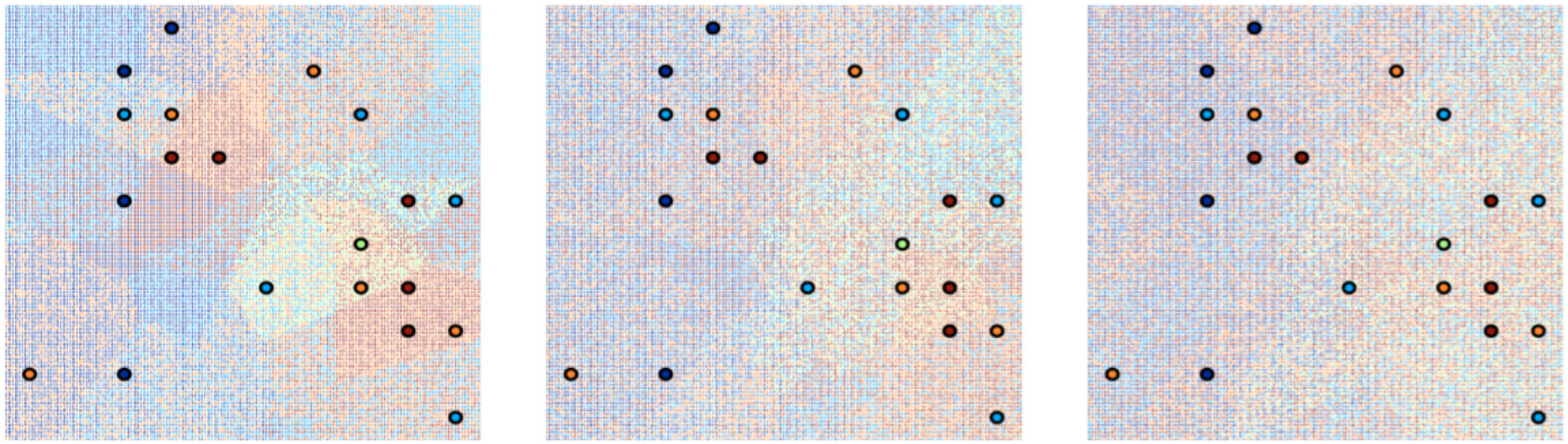
From: "Pattern Classification". R. Duda, P. Hart, and D. Stork (2001) Wiley.

Stability and Bias-Variance

- stable algorithms typically have high bias
- unstable algorithms typically have high variance
- BUT: take care to consider effect of parameters on stability, e.g., in kNN:
  - 1NN perfectly separates training data, so low bias but high variance
  - By increasing the number of neighbors k we increase bias and decrease variance (what happens when $k = n$?)
  - Every test instance will have the same number of neighbors, and the class probability vectors will all be the same !

Three-, five- and seven-nearest neighbour

Decision regions of kNN classifiers; the shading represents the predicted probability distribution over the five classes.



Illustrates the effect of varying k on stability (i.e., bias and variance).

Ensemble Methods

Supervised learning

We have looked into different supervised methods:

- Basic linear classifier
- kNN
- Naïve Bayes
- Logistic regression
- Decision trees
- Perceptron
- Support vector machine
- …

For a new problem, which method to pick?

Supervised learning

- We can test all different methods on a validation set and pick the one which gives the best performance
- Each learning method gives a different hypothesis,… but no hypothesis is perfect
- Maybe we can combine different hypotheses to build a better hypothesis (model) !?

Ensemble methods

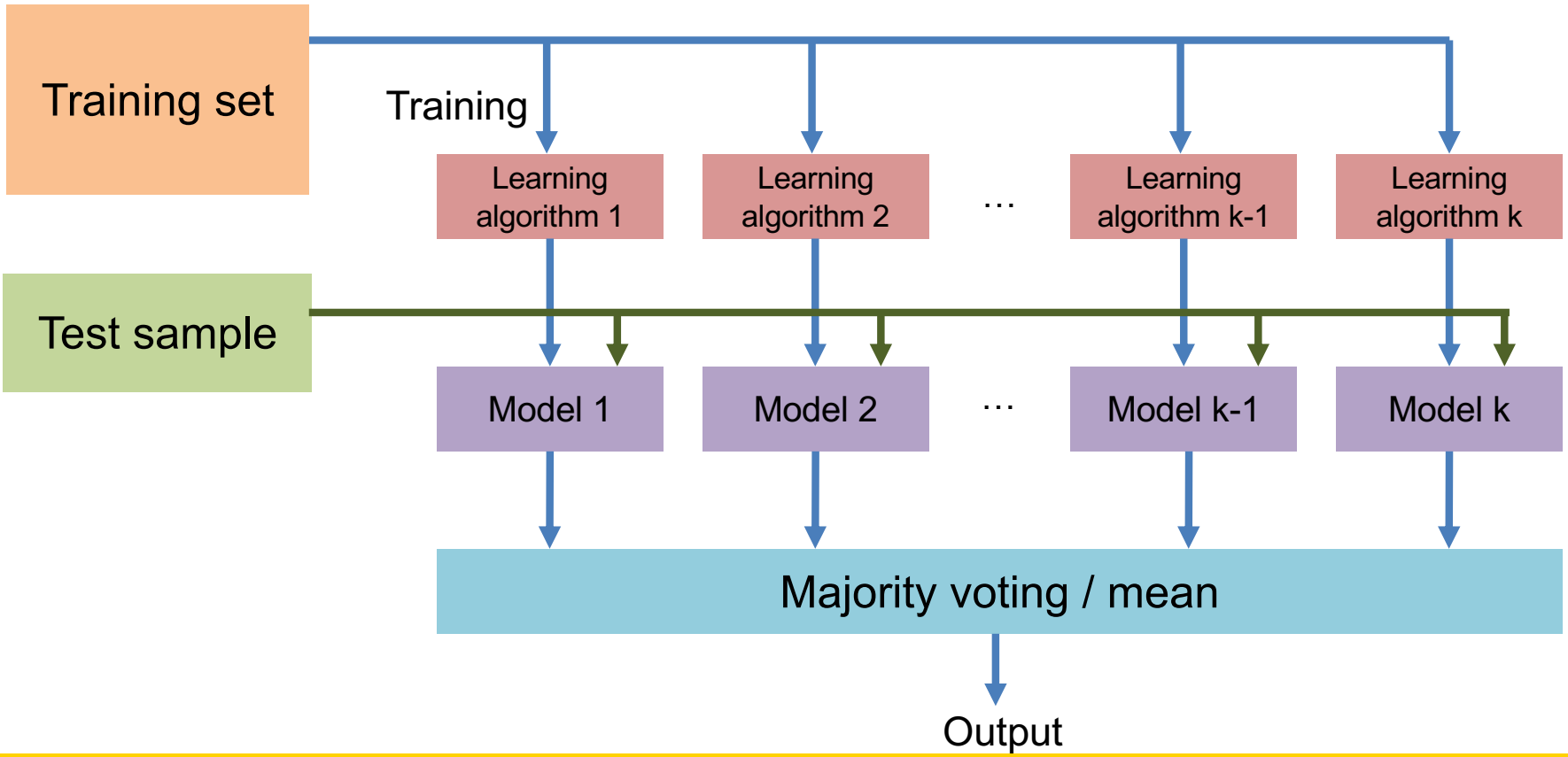
- Ensemble methods are meta-algorithms that combine different models into one model and they can:
  - Decrease variance
  - Decrease bias
  - Improve performance

- The idea relates to the “wisdom of crowd” phenomenon:
  - Aggregation of information in groups (by James Surowiecki, 1907)
  - Aggregation of independent estimate can be really effective for prediction
    - » Unweighted average (e.g. majority vote)
    - » Weighted average (e.g. committee of experts)

Simple ensembles

There are different ways of combining predictors:

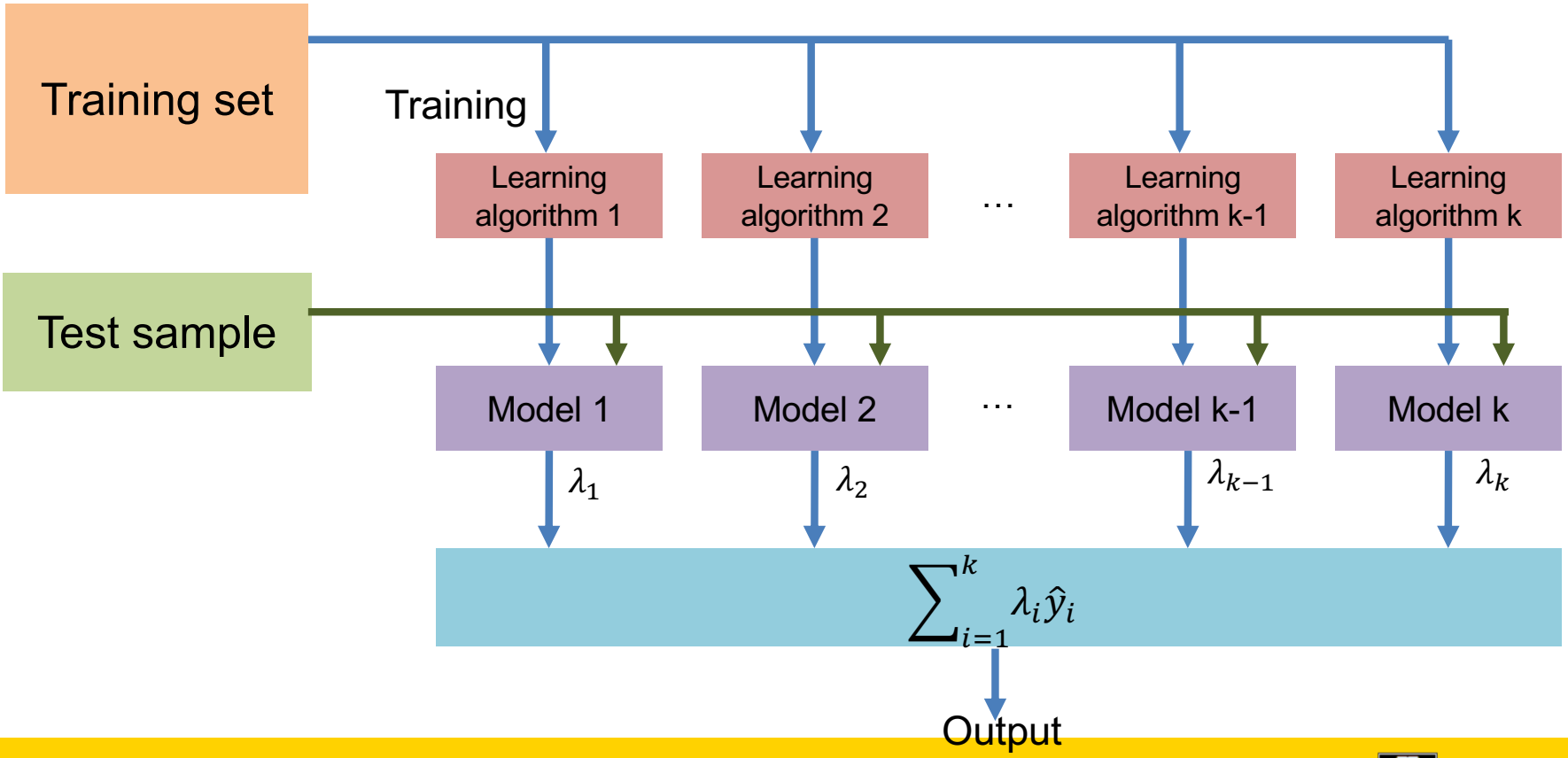
1. Simple ensembles like majority vote or unweighted average



Simple ensembles

2. weighted averages / weighted votes: Every model gets a weight (e.g. depending on its performance)

Type equation here.



Weighted average / weighted majority

In practice:

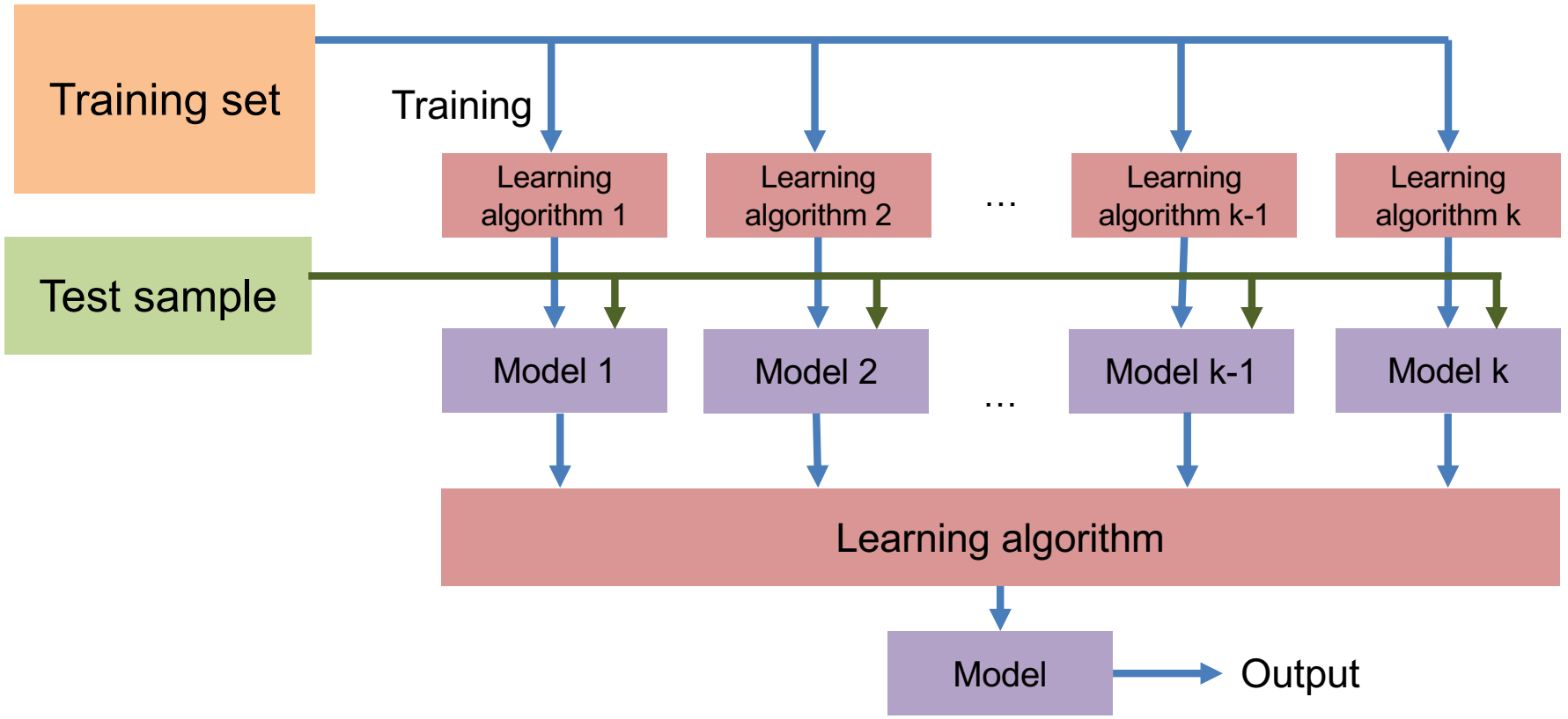
- Learning algorithms may not be independent
- Some better fit the data so make less error

We can define weights in different ways:

- Decrease weight of correlated learners
- Increase weight of good learners

Simple ensembles

3. Treat the output of each model as a feature and train a model on that



Simple ensembles

3. Treat the output of each model as a feature and train a model on that
聚合
   - If the task is a binary classification and we choose the fusion model to be a linear model then this will become a weighted vote
   - We can train the fusion model on a validation set or a portion of training data that we have not used in training the initial models because otherwise it will always be biased towards the models that had a better performance on the training data

Simple ensembles

4. Mixture of experts:
   - Weight $\alpha_i(x)$ indicates "expertise"
   - It divides the feature space into homogeneous regions
   - It may use a weighted average or just pick the model with the largest expertise
   - It is a kind of local learning

同种类的

Ensemble methods

5. “Bagging” method: (“**B**ootstrap **Agg**regation”)
    - Training many classifiers, but each with only a portion of data
    - Then aggregate through model averaging / majority voting

We used data splitting in cross-validation (k-fold CV) to check overfitting and possibility avoid overfitting.

- But, maybe we can combine the models that we train on each split
- This will reduce the overfitting