# 机器学习与数据挖掘

## Machine Learning & Data Mining

权小军 教授

中山大学数据科学与计算机学院

quanxj3@mail.sysu.edu.cn

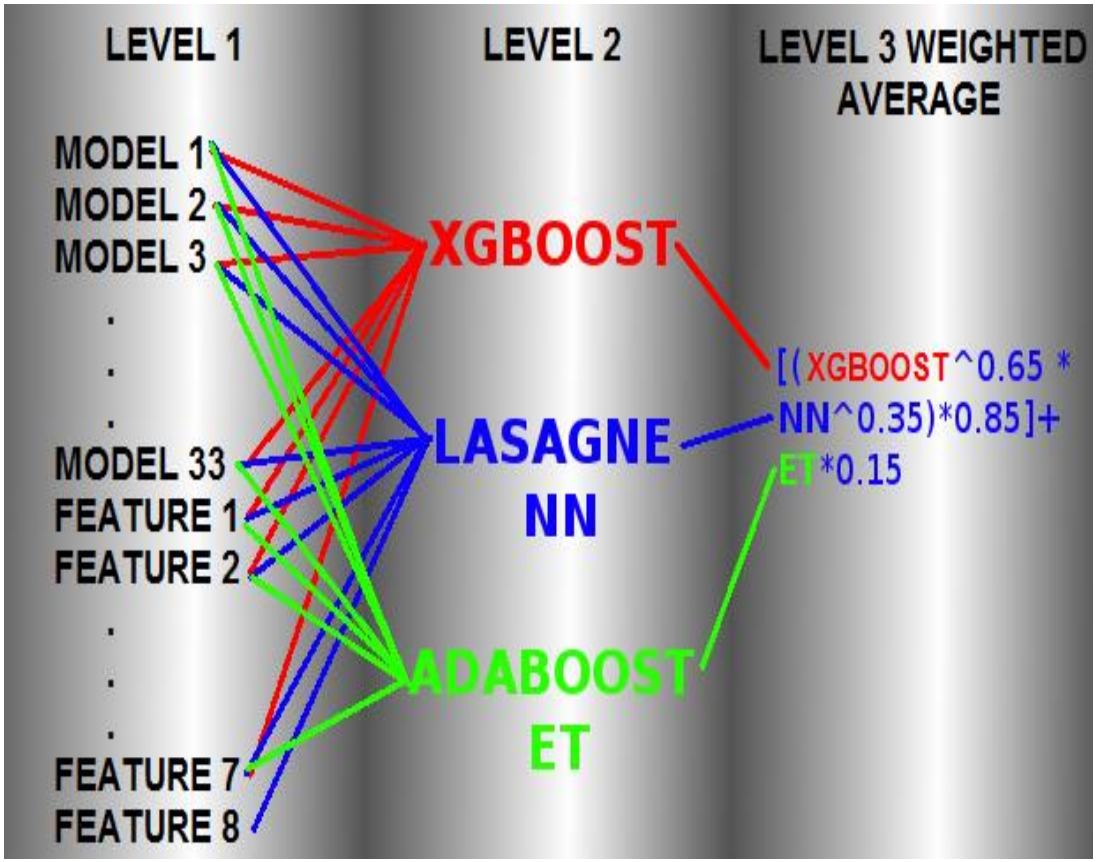
# Preface

# Preface: Kaggle's Winning Solution

- The Otto Group is one of the world's biggest e-commerce companies.
- Due to our diverse global infrastructure, many identical products get classified differently. Therefore, the quality of our product analysis depends heavily on the ability to accurately cluster similar products.
- The better the classification, the more insights we can generate about our product range.


https://www.kaggle.com/c/otto-group-product-classification-challenge

# Preface: Kaggle's Winning Solution




https://www.kaggle.com/c/otto-group-product-classification-challenge

# Preface: Kaggle's Winning Solution

## Level 1: 33 models

-Model 1: RandomForest(R). Dataset: X
-Model 2: Logistic Regression(scikit). Dataset: Log(X+1)
-Model 3: Extra Trees Classifier(scikit). Dataset: Log(X+1) (but could be raw)
-Model 4: KNeighborsClassifier(scikit). Dataset: Scale( Log(X+1) )
-Model 5: libfm. Dataset: Sparse(X). Each feature value is a unique level.
-Model 6: H2O NN. Bag of 10 runs. Dataset: sqrt( X + 3/8)
-Model 7: Multinomial Naive Bayes(scikit). Dataset: Log(X+1)
-Model 8: Lasagne NN(CPU). Bag of 2 NN runs. First with Dataset Scale( Log(X+1) ) and second with Dataset Scale( X )
-Model 9: Lasagne NN(CPU). Bag of 6 runs. Dataset: Scale( Log(X+1) )
-Model 10: T-sne. Dimension reduction to 3 dimensions. Also stacked 2 kmeans features using the T-sne 3 dimensions. Dataset: Log(X+1)
-Model 11: Sofia(R). Dataset: one against all with learner_type="logreg-pegasos" and loop_type="balanced-stochastic". Dataset: Scale(X)
-Model 12: Sofia(R). Trainned one against all with learner_type="logreg-pegasos" and loop_type="balanced-stochastic". Dataset: Scale(X, T-sne Dimension, some 3 level interactions between 13 most important features based in randomForest importance )
-Model 13: Sofia(R). Trainned one against all with learner_type="logreg-pegasos" and loop_type="combined-roc". Dataset: Log(1+X, T-sne Dimension, some 3 level interactions between 13 most important features based in randomForest importance )
-Model 14: Xgboost(R). Trainned one against all. Dataset: (X, feature sum(zeros) by row ). Replaced zeros with NA.
-Model 15: Xgboost(R). Trainned Multiclass Soft-Prob. Dataset: (X, 7 Kmeans features with different number of clusters, rowSums(X==0), rowSums(Scale(X)>0.5), rowSums(Scale(X)< -0.5) )
-Model 16: Xgboost(R). Trainned Multiclass Soft-Prob. Dataset: (X, T-sne features, Some Kmeans clusters of X)
-Model 17: Xgboost(R): Trainned Multiclass Soft-Prob. Dataset: (X, T-sne features, Some Kmeans clusters of log(1+X) )
-Model 18: Xgboost(R): Trainned Multiclass Soft-Prob. Dataset: (X, T-sne features, Some Kmeans clusters of Scale(X) )
-Model 19: Lasagne NN(GPU). 2-Layer. Bag of 120 NN runs with different number of epochs.
-Model 20: Lasagne NN(GPU). 3-Layer. Bag of 120 NN runs with different number of epochs.
-Model 21: XGboost. Trained on raw features. Extremely bagged (30 times averaged).
-Model 22: KNN on features X + int(X == 0)
-Model 23: KNN on features X + int(X == 0) + log(X + 1)
-Model 24: KNN on raw with 2 neighbours
-Model 25: KNN on raw with 4 neighbours
-Model 26: KNN on raw with 8 neighbours
-Model 27: KNN on raw with 16 neighbours
-Model 28: KNN on raw with 32 neighbours
-Model 29: KNN on raw with 64 neighbours
-Model 30: KNN on raw with 128 neighbours
-Model 31: KNN on raw with 256 neighbours
-Model 32: KNN on raw with 512 neighbours
-Model 33: KNN on raw with 1024 neighbours

**https://www.kaggle.com/c/otto-group-product-classification-challenge**

# Preface: Kaggle's Winning Solution

## Level 1: 8 features

-Feature 1: Distances to nearest neighbours of each classes

-Feature 2: Sum of distances of 2 nearest neighbours of each classes

-Feature 3: Sum of distances of 4 nearest neighbours of each classes

-Feature 4: Distances to nearest neighbours of each classes in TFIDF space

-Feature 5: Distances to nearest neighbours of each classed in T-SNE space (3 dimensions)

-Feature 6: Clustering features of original dataset

-Feature 7: Number of non-zeros elements in each row

-Feature 8: X (That feature was used only in **NN** 2nd level training)

**https://www.kaggle.com/c/otto-group-product-classification-challenge**

# Preface: Kaggle's Winning Solution

## Level 2: 3 models

- XGBOOST
  - XGBoost： eXtreme Gradient Boosting
  - 是由Tianqi Chen（http://homes.cs.washington.edu/~tqchen）最初开发的实现可扩展，便携，分布式 gradient boosting算法的一个库
- Neural Network
- ADABOOST
  - AdaBoost： Adaptive Boosting
  - AdaBoost方法是一种迭代算法，在每一轮中加入一个新的弱分类器，直到达到某个预定的足够小的错误率

**https://www.kaggle.com/c/otto-group-product-classification-challenge**

# Preface: Kaggle's Winning Solution

Level 3: average of 3 models

0.85 * [XGBOOST^0.65 * NN^0.35] + 0.15 * [ET]

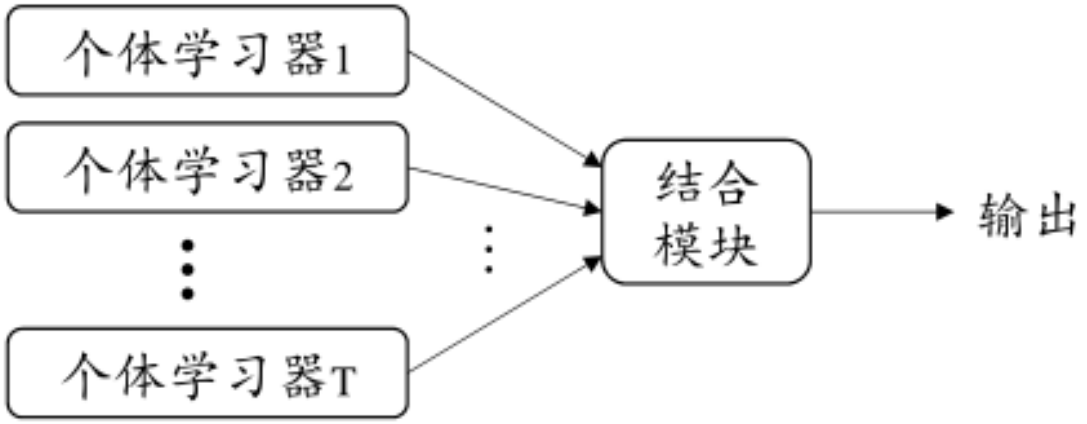
https://www.kaggle.com/c/otto-group-product-classification-challenge

# Lecture 5: Ensembles I

# 5.1 The Wisdom of Ensembling

# 个体与集成

- 集成学习(ensemble learning)通过构建并结合多个学习器来提升性能
- 集成方法的思想简单直接，用一句谚语就可以概括：“三个臭皮匠，顶个诸葛亮”
- 集成多个模型的能力，达到比单一模型更佳的效果。这些算法可以是不同的算法，也可以是相同的算法



# 个体与集成

- 考虑一个简单的例子，在二分类问题中，假定3个分类器在三个样本中的表现如下图所示，其中√ 表示分类正确，X 号表示分类错误，集成的结果通过投票产生。

| | 测试例1 | 测试例2 | 测试例3 |
|---|---|---|---|
| $h_1$ | √ | √ | × |
| $h_2$ | × | √ | √ |
| $h_3$ | √ | × | √ |
| 集群 | √ | √ | √ |

(a) 集群提升性能

| | 测试例1 | 测试例2 | 测试例3 |
|---|---|---|---|
| $h_1$ | √ | √ | × |
| $h_2$ | √ | √ | × |
| $h_3$ | √ | √ | × |
| 集群 | √ | √ | × |

(b) 集群不起作用

| | 测试例1 | 测试例2 | 测试例3 |
|---|---|---|---|
| $h_1$ | √ | × | × |
| $h_2$ | × | √ | × |
| $h_3$ | × | × | √ |
| 集群 | × | × | × |

(c) 集群起负作用

- 结论：？？？？

# 个体与集成

- 考虑一个简单的例子，在二分类问题中，假定3个分类器在三个样本中的表现如下图所示，其中√ 表示分类正确，X 号表示分类错误，集成的结果通过投票产生。

| | 测试例1 | 测试例2 | 测试例3 |
|---|---|---|---|
| $h_1$ | √ | √ | × |
| $h_2$ | × | √ | √ |
| $h_3$ | √ | × | √ |
| 集群 | √ | √ | √ |

(a) 集群提升性能

| | 测试例1 | 测试例2 | 测试例3 |
|---|---|---|---|
| $h_1$ | √ | √ | × |
| $h_2$ | √ | √ | × |
| $h_3$ | √ | √ | × |
| 集群 | √ | √ | × |

(b) 集群不起作用

| | 测试例1 | 测试例2 | 测试例3 |
|---|---|---|---|
| $h_1$ | √ | × | × |
| $h_2$ | × | √ | × |
| $h_3$ | × | × | √ |
| 集群 | × | × | × |

(c) 集群起负作用

- 结论：集成个体应好而不同

# 个体与集成 – 简单分析

- 考虑二分类问题，假设基分类器的错误率为：

$$P\left(h_i\left(\boldsymbol{x}\right) \neq f\left(\boldsymbol{x}\right)\right) = \epsilon$$

- 假设集成通过简单投票法结合$T$个分类器，若有超过半数的基分类器正确则分类就正确

$$H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{T} h_i\left(\boldsymbol{x}\right)\right)$$

# 个体与集成 – 简单分析

- 假设基分类器的错误率相互独立，则由Hoeffding不等式可得集成的错误率为：

$$P\left(H\left(\boldsymbol{x}\right)\neq f(\boldsymbol{x})\right)=\sum_{k=0}^{\lfloor T/2\rfloor}\binom{T}{k}(1-\epsilon)^{k}\epsilon^{T-k}$$

$$\leq\exp\left(-\frac{1}{2}T\left(1-2\epsilon\right)^{2}\right)$$

- 上式显示，在一定条件下，随着集成分类器数目的增加，集成的错误率将指数级下降，最终趋向于0

# 个体与集成 – 简单分析

- 上面的分析有一个关键假设：基学习器的误差相互独立
- 现实任务中，个体学习器是为解决同一个问题训练出来的，显然不可能互相独立
- 个体学习器的“准确性”和“多样性”本身就存在冲突
- 如何产生“好而不同”的个体学习器是集成学习研究的核心
- 集成学习大致可分为三大类：**Boosting, Bagging, Stacking**

# 个体与集成 – 简单分析

- **Bagging**：对样本或特征随机取样，学习产生多个独立的模型，然后平均所有模型的预测值。典型代表是随机森林;
- **Boosting**：串行训练多个模型，后面的模型是基于前面模型的训练结果(误差)。它的代表是AdaBoost;
- **Stacking**的具体过程如下：

1. 划分训练数据集为两（或多）个不相交的集合。
2. 在第一个集合上训练多个学习器。
3. 在第二个集合上测试这几个学习器
4. 把第三步得到的预测结果作为输入，把正确的回应作为输出，训练一个高层学习器

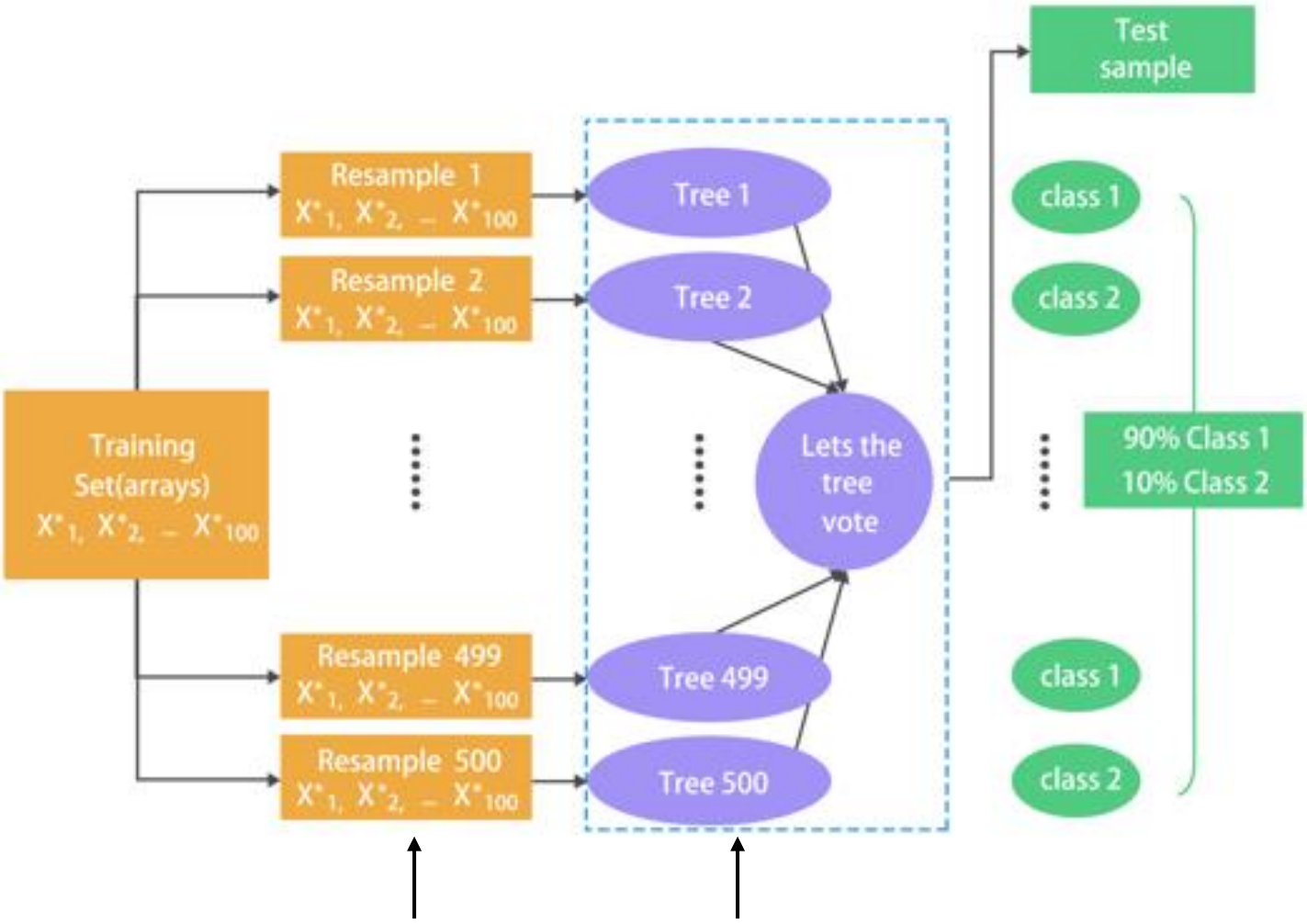
集成模型三大利器：Bagging，Boosting，Stacking

# 2. Bagging and Random Forest

# Bagging

- Bagging是”Bootstrap aggregating”的缩写
- Bagging用来生成多个分类器并且集成这些分类器形成一个整体的模型
- 不稳定性是单一预测算法的明显的不足，Bagging可以通过所有的分类器投票的方式，提升算法的稳定性

# Bagging


Test sample
Resample 1
X*1, X*2, – X*100
Resample 2
X*1, X*2, – X*100
Resample 499
X*1, X*2, – X*100
Resample 500
X*1, X*2, – X*100
Training Set(arrays)
X*1, X*2, – X*100
Tree 1
Tree 2
Tree 499
Tree 500
Lets the tree vote
class 1
class 2
class 1
class 2
90% Class 1
10% Class 2
样本采样
投票预测

# Bagging

- 给定$n$个样本的数据集$D$，对于迭代$(t = 1,2,\ldots,T)$，训练集$D_t$采用有放回抽样，从原始样本集$D$抽取。用来创建分类器$M_t$。基于基分类器的投票返回类预测
- 有放回抽样，$D$的某些样本可能不在$D_t$中出现，样本可能出现多次。
- 平均而言，对每个$D_t$有63.2%的样本至少出现一次
- 数学证明

？？？？

# Bagging

- 数学证明

一个样本有$1-\frac{1}{n}$的概率不会被选到，则一个样本不会被抽到的概率是

$$\lim_{n\to\infty}(1-\frac{1}{n})^n = 0.368$$

意味着训练集包含63.2%的不同的样本

# Bagging

- 每个$D_t$学习得到一个分类器$M_t$。
- 对未知样本$X$分类，每个分类器返回类预测，将得票最高的类赋予$X$。
- 对于回归问题，????????

# Bagging

- 每个$D_t$学习得到一个分类器$M_t$。
- 对未知样本$X$分类，每个分类器返回类预测，将得票最高的类赋予$X$。
- 对于回归问题，则取每个分类器的预测值的平均值

# Bagging算法

- 训练算法：

  1）for $t = 1\ to\ T$ do

  2）　通过对$D$有放回抽样，得到训练样本$D_t$

  3）　使用$D_t$训练得到模型$M_t$

  4）结束

- 预测：使用组合分类器对样本$X$分$T$个模型预测并返回多数表决

# Bagging算法特点

- 时间复杂度低
  - 假定基学习器的计算复杂度为O(m)，采样与投票/平均过程的复杂度为O(s)，则bagging的复杂度大致为T(O(m)+O(s))
  - 由于O(s)很小且T是一个不大的常数
  - 因此训练一个bagging集成与直接使用基学习器的复杂度同阶

# Bagging算法特点

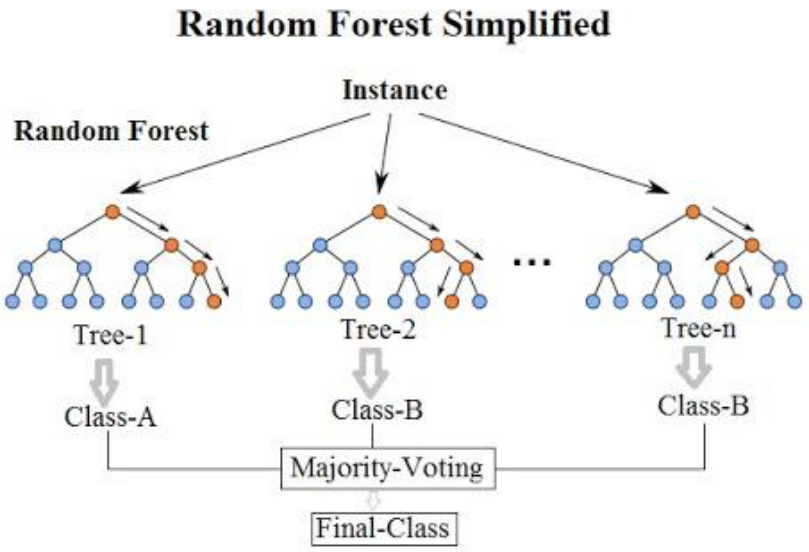
- 个体学习器不存在强依赖关系
- 并行化生成
- 自助采样法

# 何时使用**Bagging**?

- 单个模型不稳定：轻微的训练集的改变能够造成分类器明显的变化
- 使用Bagging可以综合投票结果，从而提升稳定性以及准确率。
- 不稳定的模型：决策树，神经网络等。

# 随机森林(Random Forest)

- 随机森林：最典型的Bagging算法
- 主要特点：
  - 对样本进行有放回抽样
  - 对特征也进行随机抽样
  - 基本分类器最常见的为决策树



# 决策树的局限性

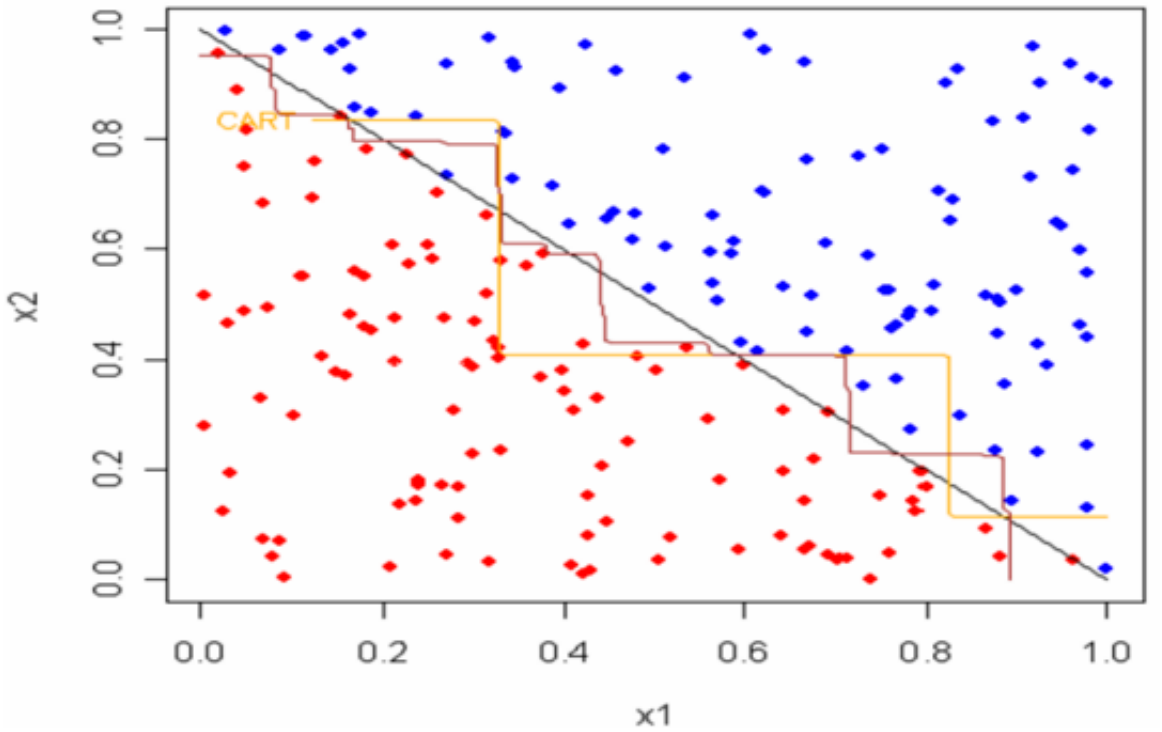
- **局部最优问题**

决策树由于使用贪婪算法的思想，即在每次分割时选择当前情况下带来信息增益最高的属性进行分割，容易陷入局部最优

- **分类边界问题**

单棵的决策树在确定分类边界时，由于决策特征只涉及单个特征的逻辑判断，导致决策边界是平行于坐标轴的直线，这就限制了决策树对分类边界的表达能力，导致模型的准确性较差。

# 决策树的局限性




橙色和棕色的线代表单个决策树的分类边界，黑色表示真实的分类边界。

# 随机森林原理

- 随机森林使用并汇总多棵决策树进行预测，所以即使每棵树的决策能力很弱，由它们组合起来形成的随机森林的决策能力也会较强。假设使用三棵决策树组合成随机森林，每棵树各不相同且预测结果相互独立，每棵树的预测错误率为40%，那么两棵树以及两棵树以上预测错误的概率下降为：

$$\text{三棵全部错误} + \text{两棵树错误一个棵树正确} = 0.4^3 + 3 * 0.4^2 * (1 - 0.4) = 0.352$$

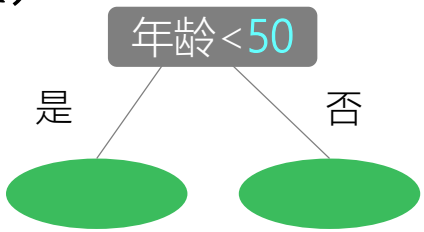
40% → 35.2%

# 随机森林算法

- 在上述过程中，需要注意一点是错误率降低的前提假设：每棵树各不相同且预测结果相互独立
- 随机森林在构建每棵树的时候，为了保证各棵树之间的独立性，通常会采用两到三层的随机性：

1. 随机有放回的抽取数据（使用Bagging），放入决策树模型
2. 随机选取$m$个特征 从m个中选局部最优
3. 随机选择特征取值进行分割（不遍历特征所有取值）

必备



# 随机森林算法

## 随机森林算法

1. 用$n$来表示训练用例（样本）的个数，$d$表示特征数目；
2. 输入特征数目$m$，用于确定决策树上一个节点的决策结果；其中$m$应远小于$d$；
3. 从$n$个训练用例（样本）中以有放回抽样的方式，取样$n$次，形成一个训练集（即bootstrap取样）；
4. 对于每一个节点，随机选择$m$个特征，决策树上每个节点的决定都是基于这些特征确定的。根据这m个特征，计算其最佳的分裂方式；

**随机森林是代表集成学习技术水平的方法**

# 随机森林(Random Forest)

## 每次采样多少特征?

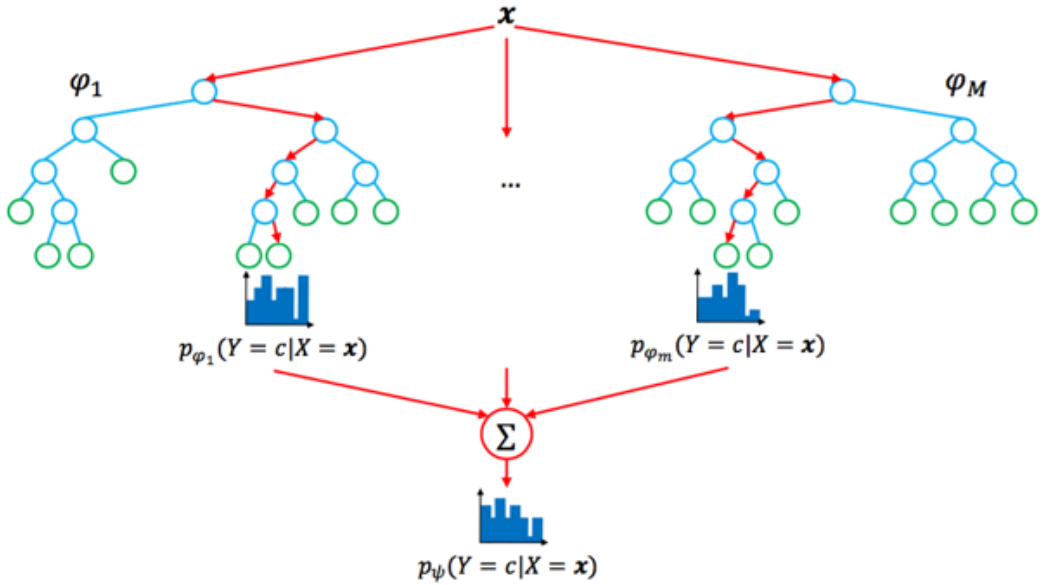
- 每一棵树的建造方式与标准的决策树不同的是，在每一个节点，随机选取$m$个属性。另外，由于建立决策树时的样本选择和特征选择所提供的随机性，不需要剪枝。
- 当所有的树之间不相关时（或很大程度上相互独立），即两棵树之间的分裂属性的重合度较小时，随机森林取得最佳的性能。通过随机选取63.2% $n$个样本以及为每个分裂节点选择$m$个属性。大的$m$值可以使得决策树的准确率提高，而每棵树的独立性基于较小的$m$值。因此，最佳的$m$值需要通过实验的来确定。通常选取$\log_2 d + 1$个属性。 其中$d$为原始数据维度。

所有特征个数

# 随机森林(Random Forest)

## 随机森林的预测

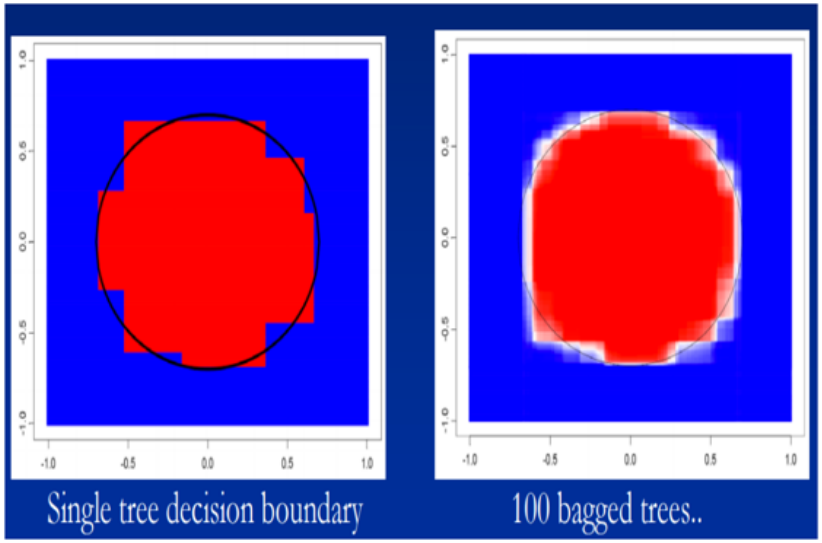
- 每一棵树进行预测
- 多数投票进行综合预测



# 随机森林(Random Forest)

## 随机森林性能优势

- 通过平均不同的决策树所得到的结果，拟合出复杂的决策边界，使得模型具有较高的准确性。如图所示，通过平均100棵树的结果，我们可以得到一个平滑的决策边界，对样本数据有更好的拟合效果



# 随机森林(Random Forest)

## 随机森林的其他功能

- 显示样本点的相似度
  - 在建立随机森林的时候，记录样本两两之间出现在同一叶子节点的次数，生成相似性矩阵（Proximity matrix）。
  - 如果越多的出现在同一叶节点，说明这两个样本越相似；如果越少的出现在同一叶节点，说明两个样本差异越大。

# 随机森林(Random Forest)

## 评价与性能

构建完成随机森林之后，主要有以下两种指标评价随机森林的表现:

- 分类间隔（Margins）：随机森林的分类间隔是指森林中正确分类样本的决策树的比例减去错误分类样本决策树的比例。假设对样本 A 有75%的树分类正确， 那么分类间隔就是75% - 25% = 50%。通过平均随机森林在各个样本上的分类间隔得到随机森林的分类间隔。实际中，我们希望分类间隔越大越好，因为大的分类间隔表示我们的分类效果比较稳定，泛化效果更好;

# 随机森林(Random Forest)

## 评价与性能

- 袋外错误率（Out-Of-Bag Error): 对每一棵树来说，都有部分样本没有被抽样到进入训练的样本，它们叫做袋外样本。随机森林对袋外样本的预测错误比率被称为袋外错误率。计算方式如下:

  1. 对每个样本，计算把该样本作为袋外样本的树对该样本的分类情况
  2. 以简单多数投票作为该样本的分类结果
  3. 用误分样本个数占样本总数的比率作为随机森林的袋外错误率

# 随机森林(Random Forest)

## 随机森林的优缺点

**优点：**

- 它能够处理很高维度的数据，并且不用做特征选择
- 在训练完后，它能够给出哪些特征比较重要
  - https://blog.datadive.net/selecting-good-features-part-iii-random-forests/
- 容易做成并行化方法

**缺点：**

- 处理噪音较大的小样本和低维数据集的问题上会过拟合
- 相对于决策树，执行速度较慢
- 相对于决策树，模型可解释性较差

# Variable importance 特征重要性

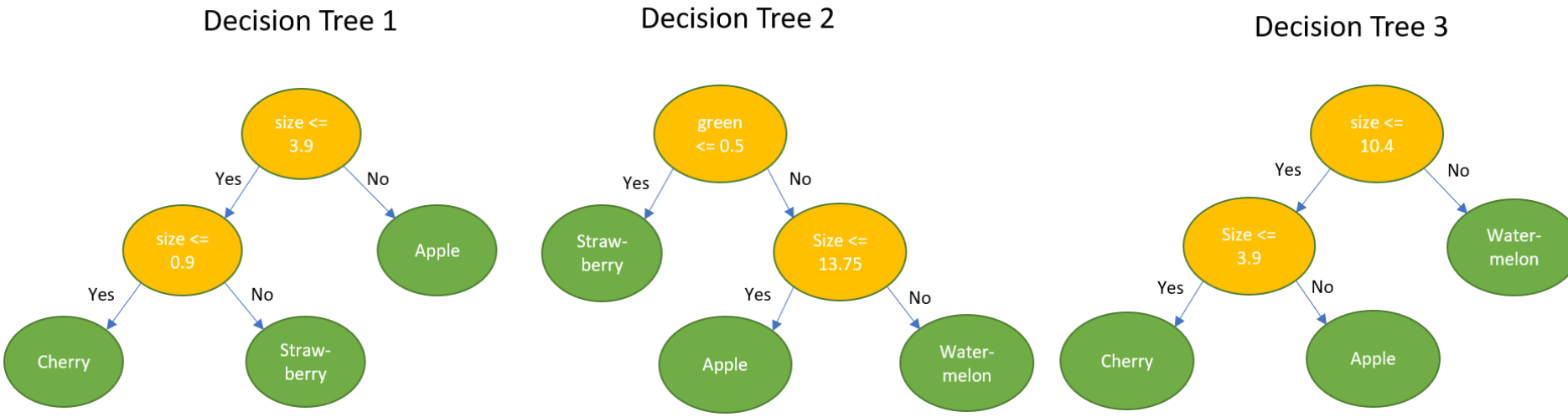
- The first step in measuring the variable importance in a data set $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$ is to fit a random forest to the data. During the fitting process the out-of-bag error for each data point is recorded and averaged over the forest;
- To measure the importance of the *j*-th feature after training, the values of the *j*-th feature are permuted among the training data and the out-of-bag error is again computed on this perturbed data set.

原本属性取值打乱 一如男→女
女→男

# Unsupervised learning with random forests

- As part of their construction, random forest predictors naturally lead to a dissimilarity measure among the observations. One can also define a random forest dissimilarity measure between unlabeled data: the idea is to construct a random forest predictor that distinguishes the “observed” data from suitably generated synthetic data.

# Random forests: example



# 思考

1. 随机森林的随机性体现在哪里？

样本随机性，特征随机性。

2. 随机森林为什么不容易过拟合？

因为有很强的随机性，随机选择样本来构建没个树，特征随机性等。

3. 随机森林为什么不能用全样本去训练m棵决策树？

全样本忽视了局部样本的规律，对模型的泛化能力是有害的

单棵树的拟合能力是比较弱的，通过成百上千的模型集合起来能力就会很强

# Thank you!

**权小军** 中山大学数据科学与计算机学院