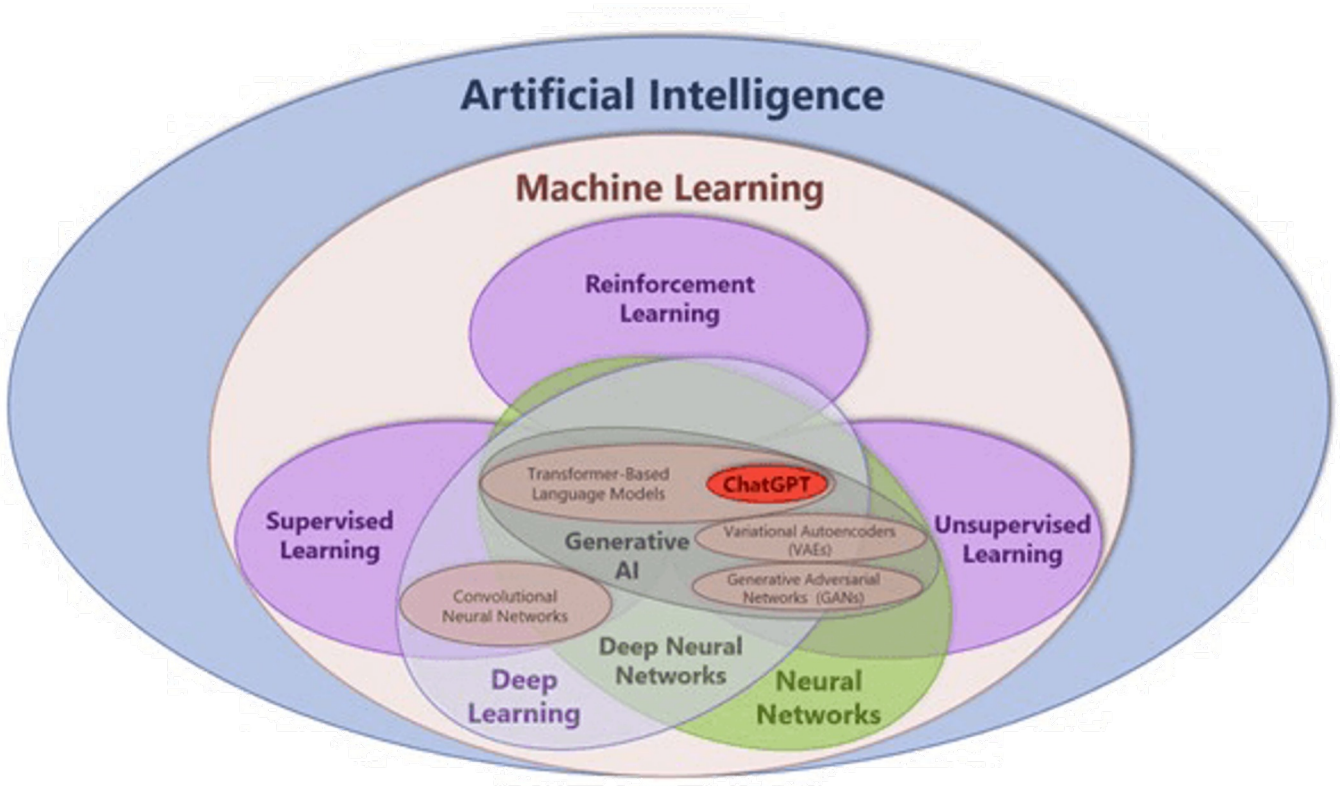
机器学习基础

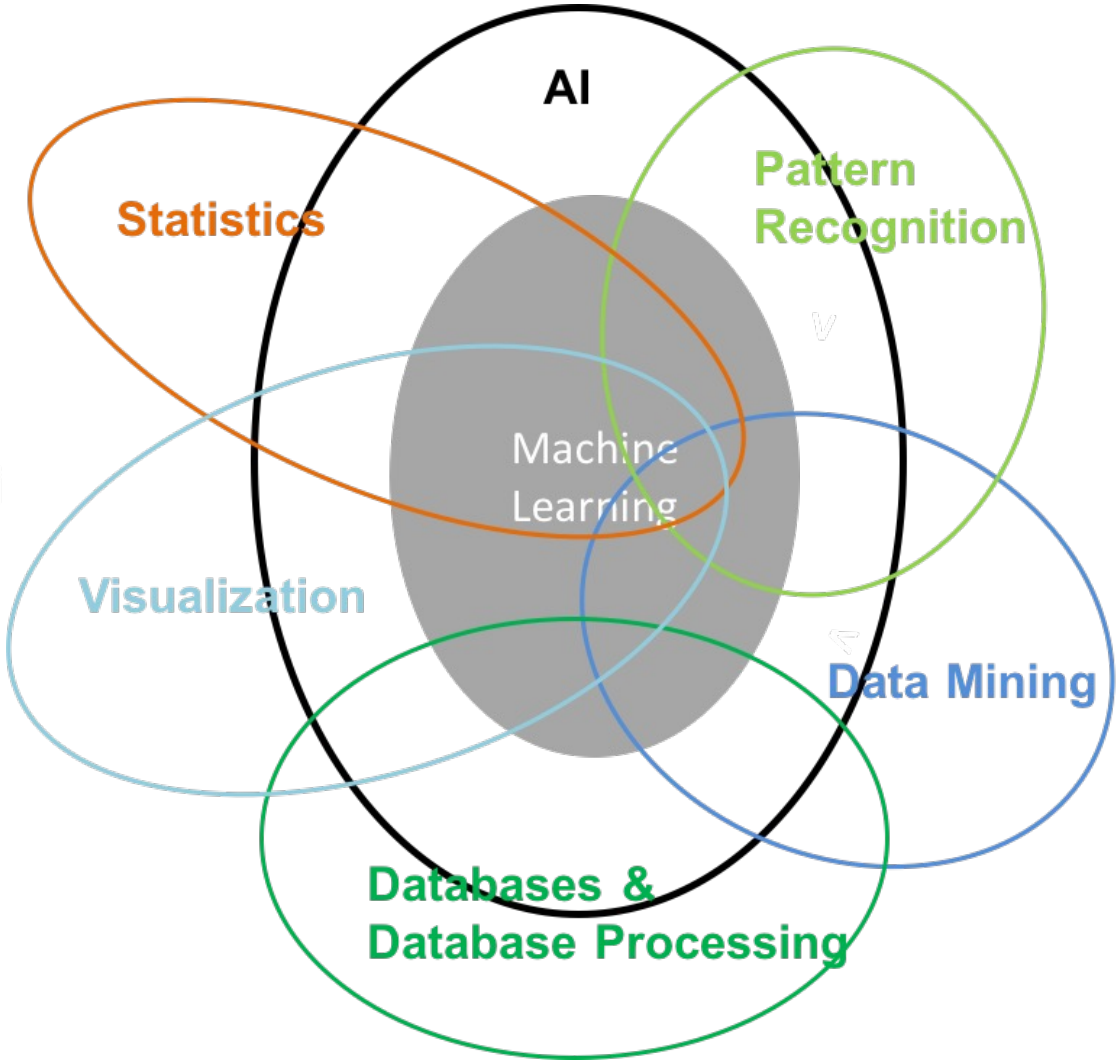
吴永成
南京师范大学
2025新物理冬季学校
2025.01.03-2025.01.13

机器学习简介

机器学习 vs. 人工智能



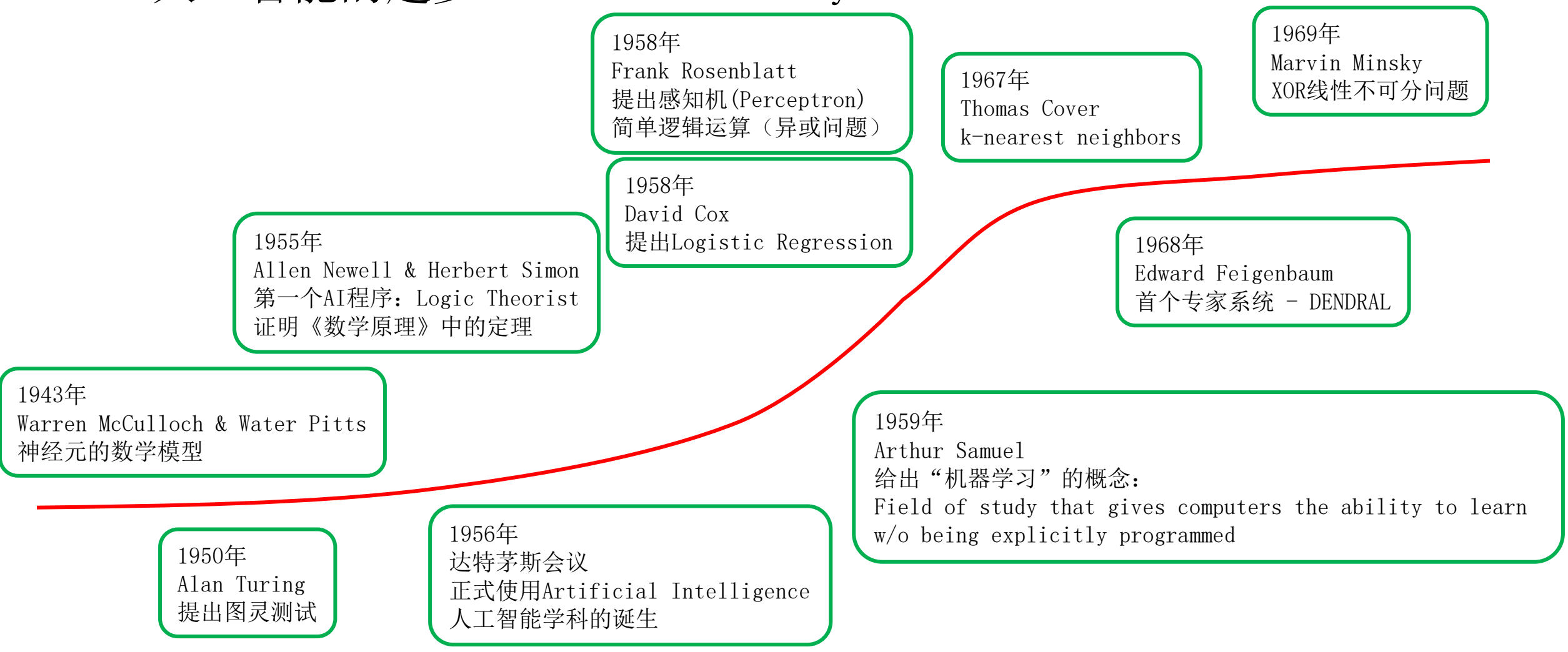
Credit: https://medium.com/@talukder9712/what-is-generative-ai-how-does-it-relate-to-ai-ml-and-dl-52be0b58fda3



Credit: https://dun.163.com/news/p/682a20476d5d4aacb707e09610b1874a

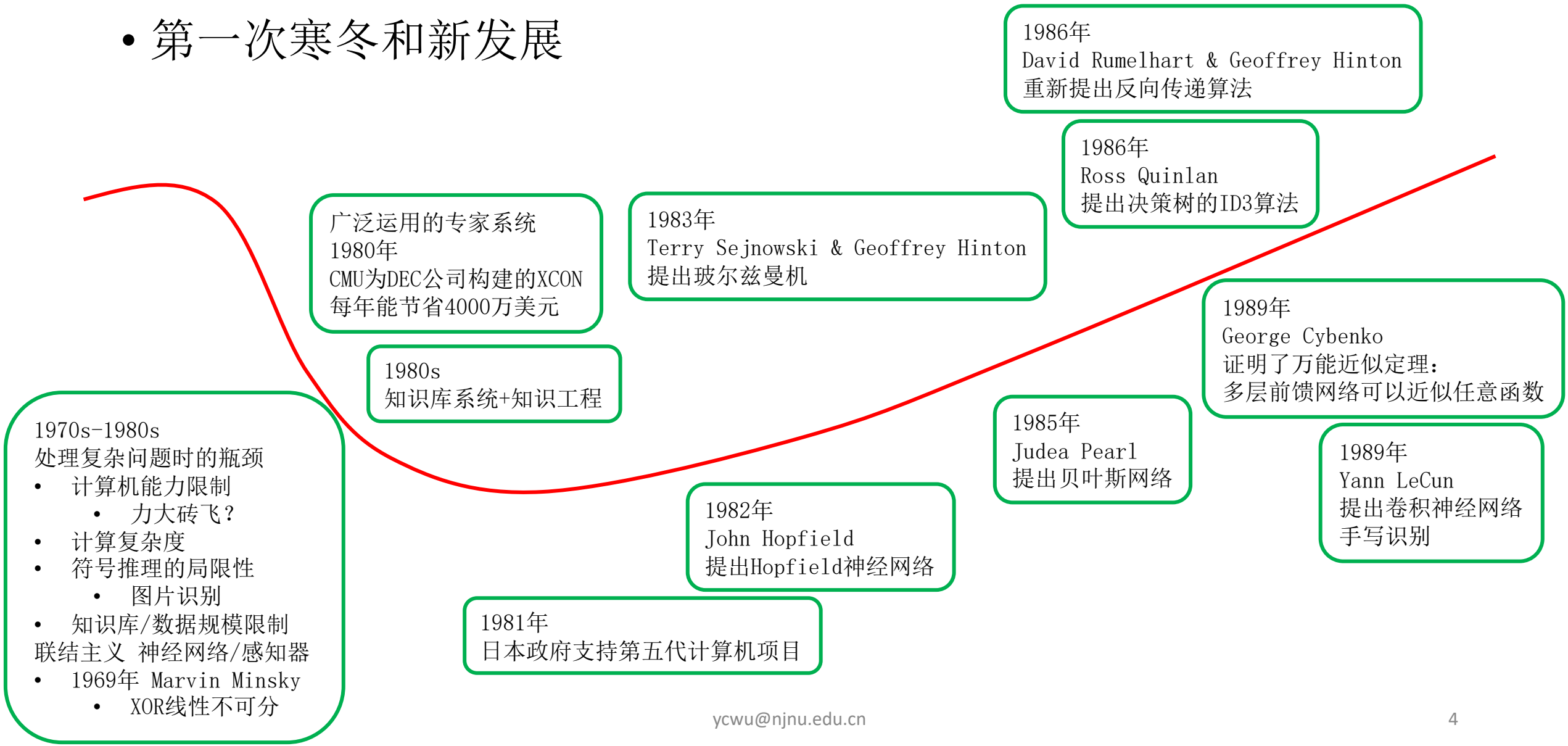
简史

- 人工智能的起步：1940s - early 1970s



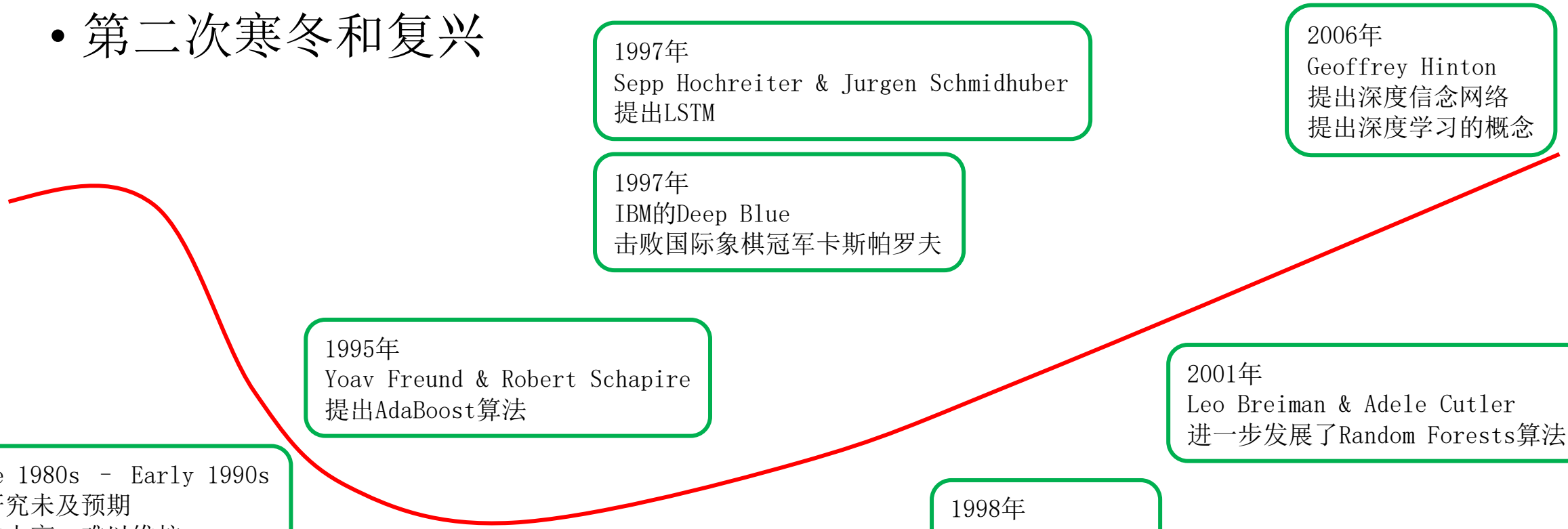
简史

- 第一次寒冬和新发展



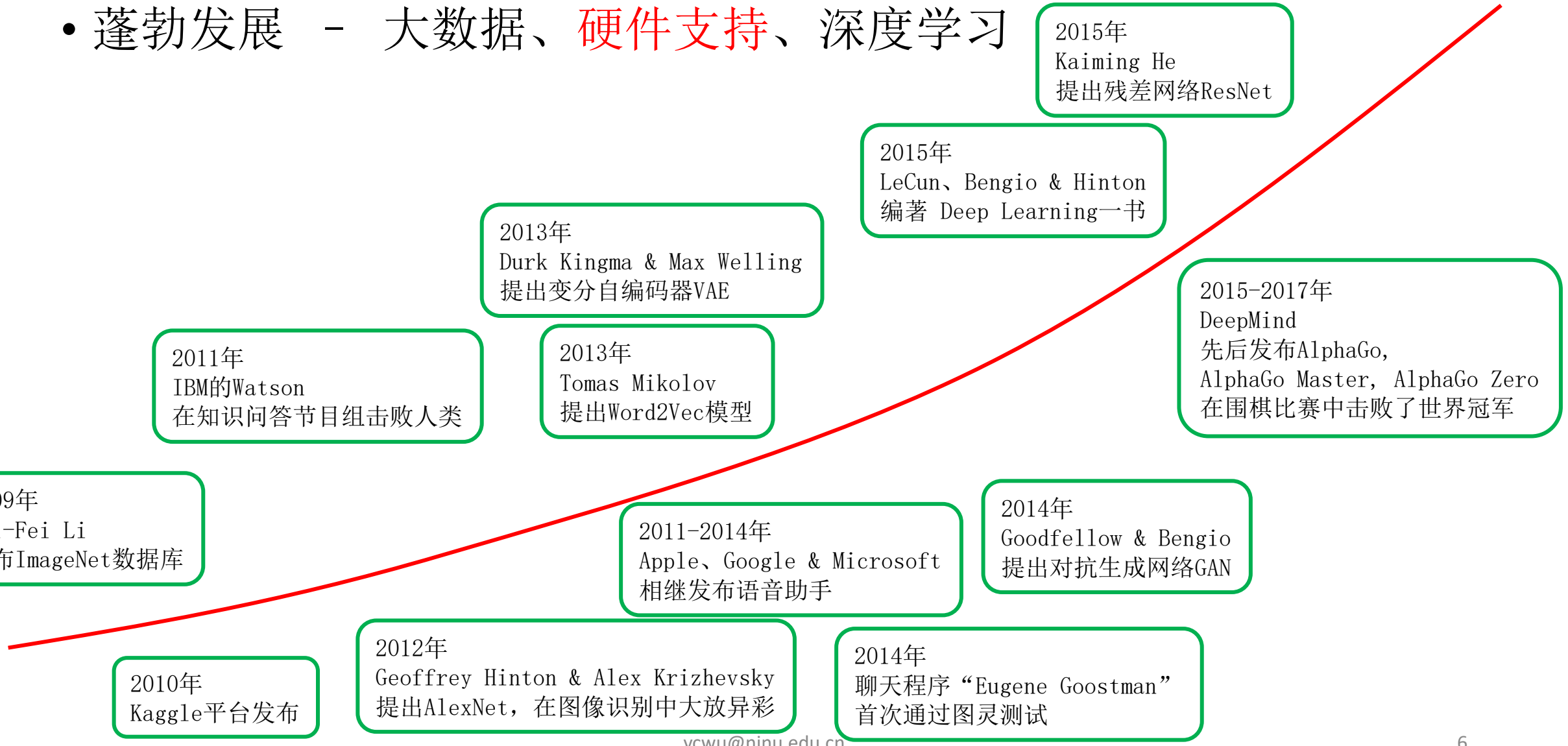
简史

- 第二次寒冬和复兴


1997年
Sepp Hochreiter & Jurgen Schmidhuber
提出LSTM
2006年
Geoffrey Hinton
提出深度信念网络
提出深度学习的概念
1997年
IBM的Deep Blue
击败国际象棋冠军卡斯帕罗夫
1995年
Yoav Freund & Robert Schapire
提出AdaBoost算法
2001年
Leo Breiman & Adele Cutler
进一步发展了Random Forests算法
Late 1980s - Early 1990s
AI研究未及预期
成本太高，难以维护
1998年
Yann LeCun
发布MNIST数据库
1995年
Tin Kam Ho
提出Random Decision Forests
1995年
Corinna Cortes & Vladimir Vapnik
发表关于支持向量机的文章

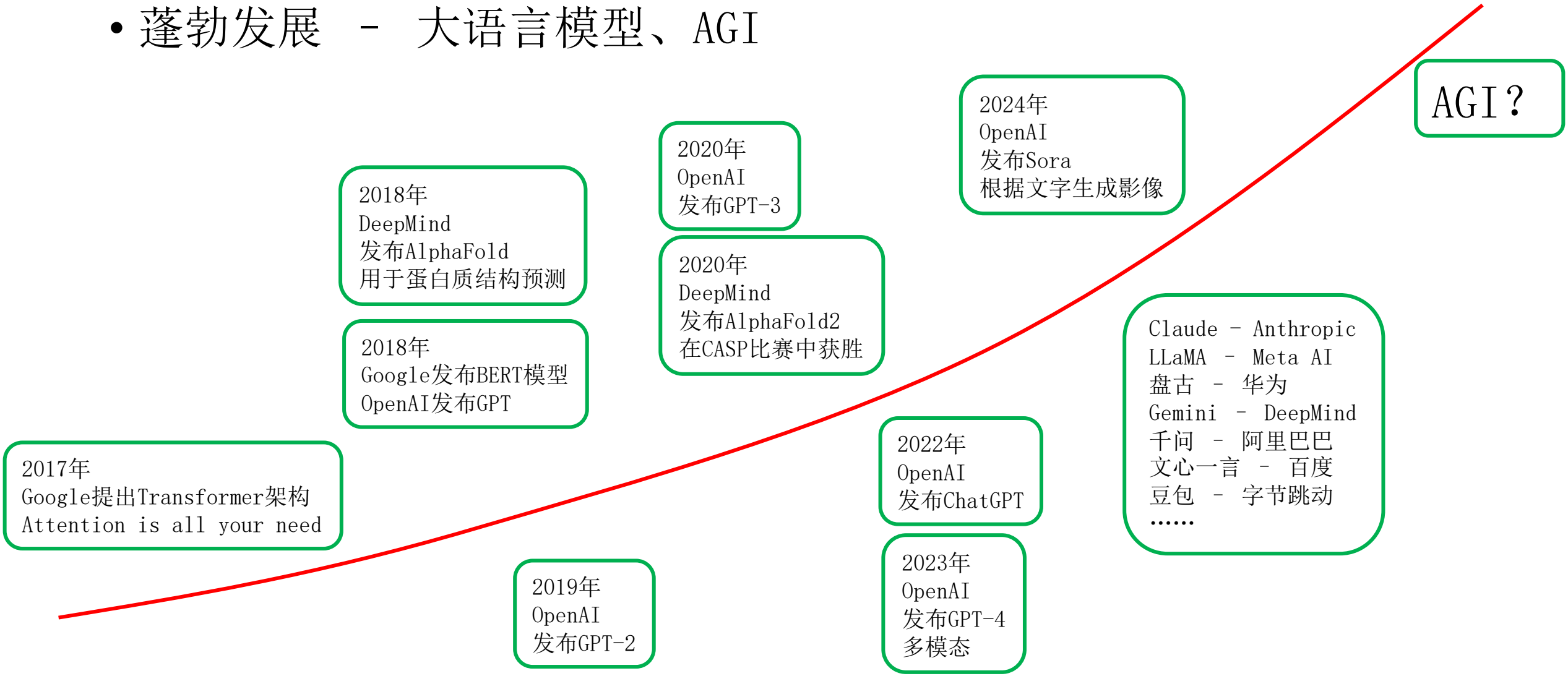
简史

- 蓬勃发展 – 大数据、硬件支持、深度学习



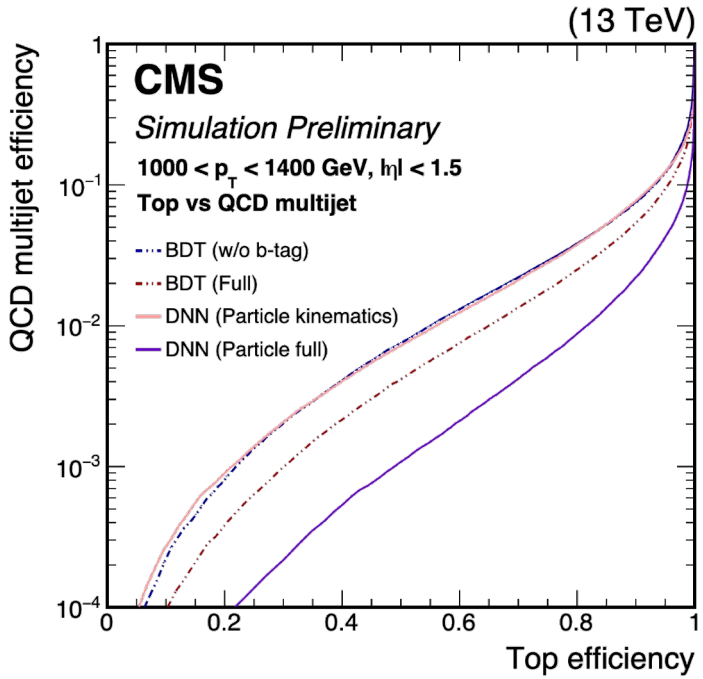
简史

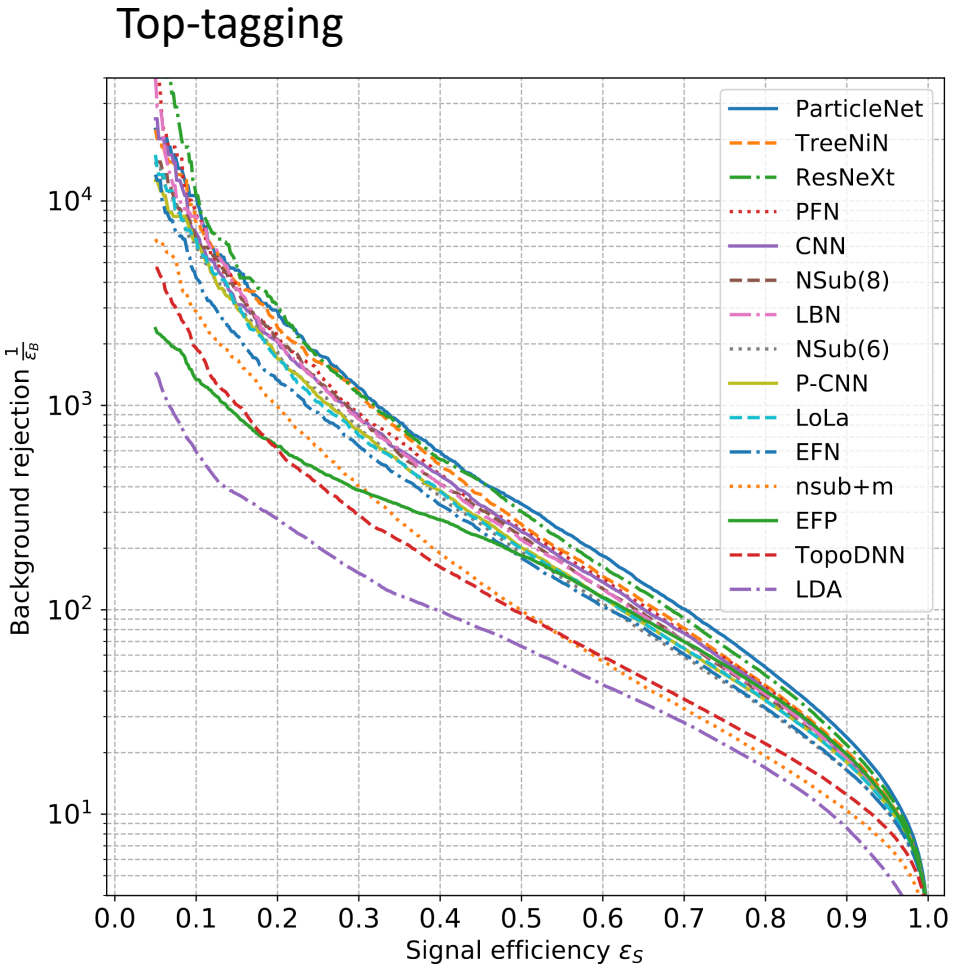
- 蓬勃发展 - 大语言模型、AGI



粒子物理中的机器学习应用

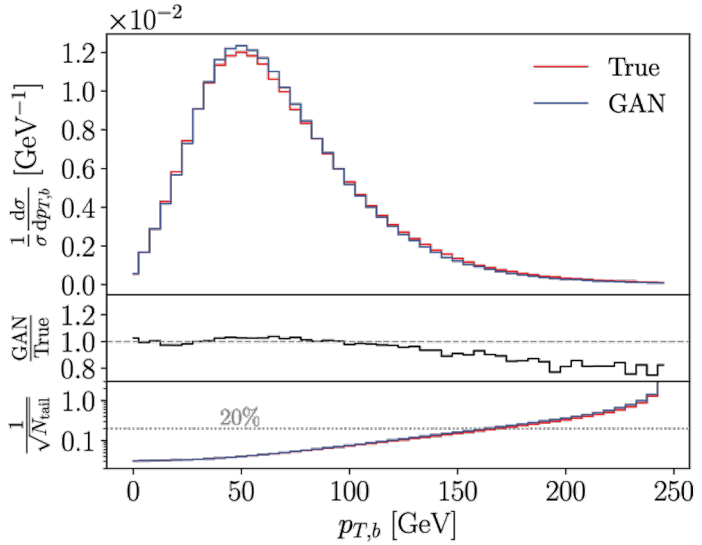
- Jet-tagging
- Signal vs. Background
  - BDT
- Anomaly Searches
  - Unsupervised Learning

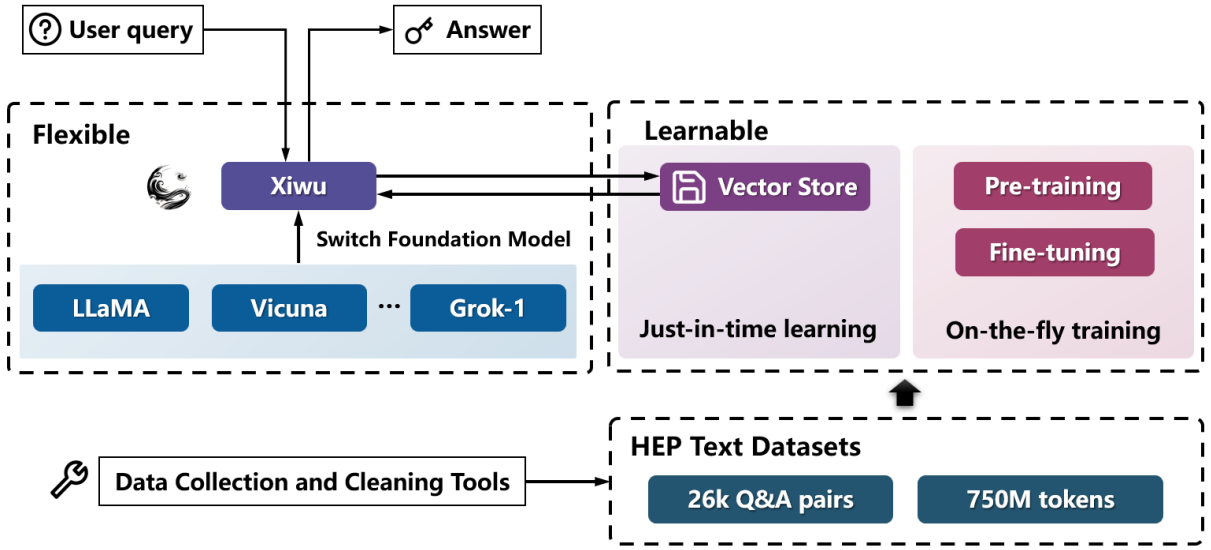


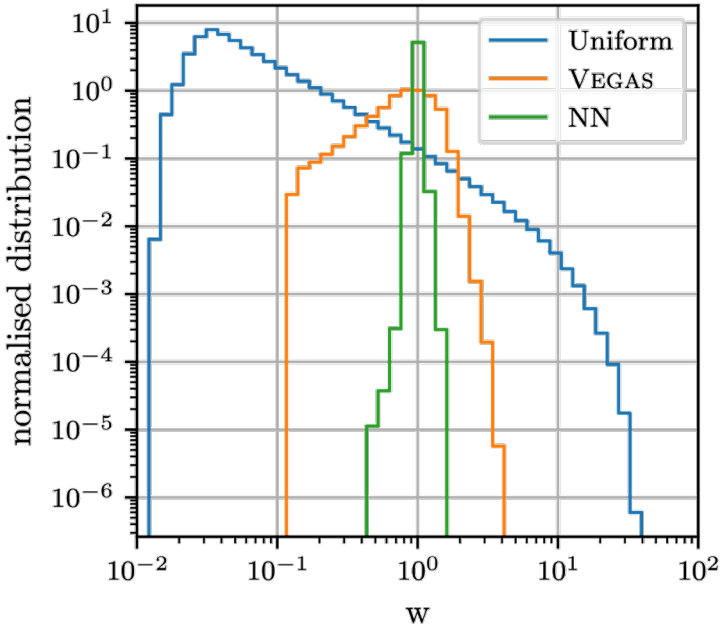


粒子物理中的机器学习应用

- Event generations
  - GAN
  - Diffusion
- Phase space generation
- LLM for high energy physics







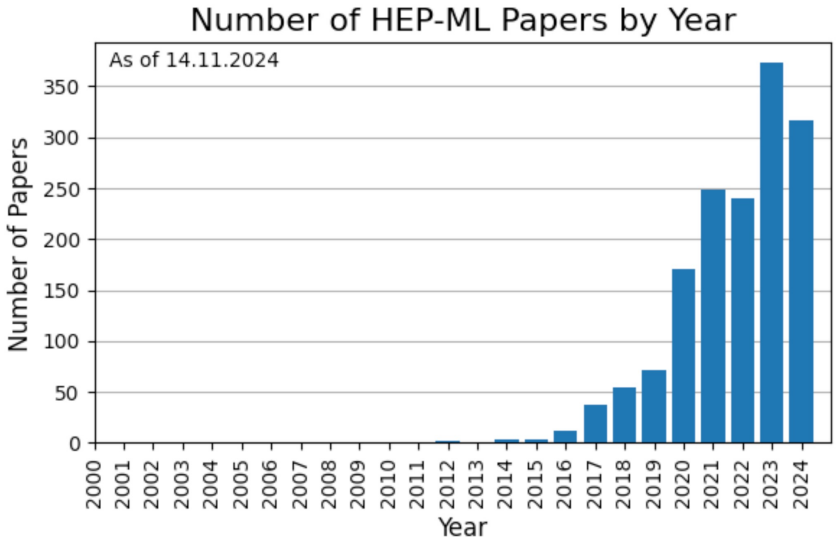
粒子物理中的机器学习应用

- Living Review: https://iml-wg.github.io/HEPML-LivingReview/

A Living Review of Machine Learning for Particle Physics

Modern machine learning techniques, including deep learning, is rapidly being applied, adapted, and developed for high energy physics. The goal of this document is to provide a nearly comprehensive list of citations for those developing and applying these approaches to experimental, phenomenological, or theoretical analyses. As a living document, it will be updated as often as possible to incorporate the latest developments. A list of proper (unchanging) reviews can be found within. Papers are grouped into a small set of topics to be as useful as possible. Suggestions are most welcome.

download review GitHub



Reviews ¶

Modern reviews

- Snowmass 2021 Computational Frontier CompF03 Topical Group Report: Machine Learning (2022)
- Artificial Intelligence and Machine Learning in Nuclear Physics [DOI] (2021)
- Machine Learning in the Search for New Fundamental Physics (2021)
- Modern Machine Learning and Particle Physics [DOI] (2021)
- Machine and Deep Learning Applications in Particle Physics [DOI] (2019)
- Machine learning and the physical sciences [DOI] (2019)
- Machine learning at the energy and intensity frontiers of particle physics (2018)
- Machine Learning in High Energy Physics Community White Paper [DOI] (2018)
- Deep Learning and its Application to LHC Physics [DOI] (2018)
- Jet Substructure at the Large Hadron Collider: A Review of Recent Advances in Theory and Machine Learning [DOI] (2017)

Classification

- Parameterized classifiers
- Representations
- Targets
- Learning strategies
- Fast inference / deployment

Generative models / density estimation

- GANs
- (Variational) Autoencoders
- (Continuous) Normalizing flows
- Diffusion Models
- Transformer Models
- Physics-inspired
- Mixture Models
- Phase space generation
- Gaussian processes
- Other/hybrid

机器学习

- 机器学习的定义

> A computer program is said to learn from experience $\mathbb{E}$
> with respect to some class of tasks $\mathbb{T}$ and performance measure $\mathbb{P}$,
> if its performance at tasks $\mathbb{T}$, as measured by $\mathbb{P}$, improves with experience $\mathbb{E}$.
>
> -- Tom M. Mitchell, *Machine Learning*, 1997

  - 三个主要组成部分：
    - 任务(Tasks $\mathbb{T}$)：目标是什么？
      - 分类，回归，……
    - 经验(Experience $\mathbb{E}$)：
      - 训练集
    - 性能度量(Performance measure $\mathbb{P}$)：如何评估模型的学习效果？
      - ROC，AUC，confusion matrix，准确率，均方差，……

机器学习 - 任务

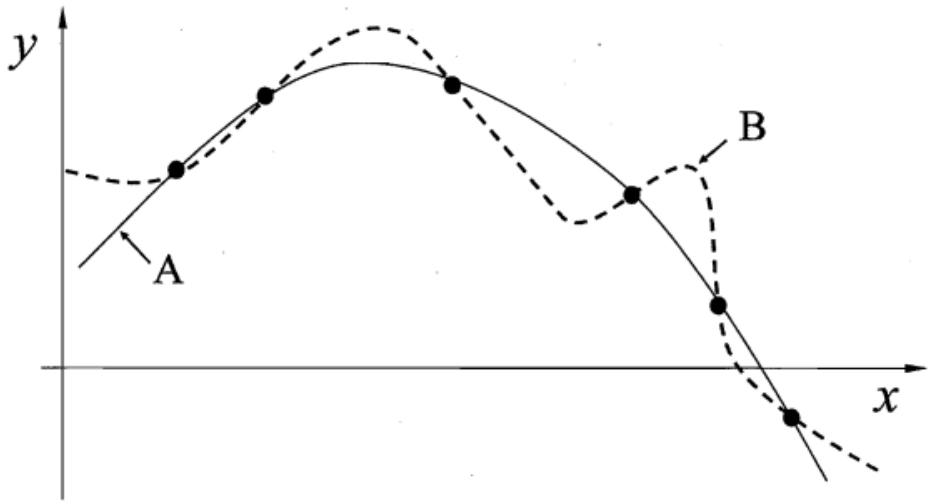
- 任务 - Task：
  - 分类、回归、翻译、问题、游戏……
- 监督学习 - Supervised Learning
  - 使用有标记的数据集来训练算法，以找到数据到标记的映射的数学模型
    - 信号和背景分类， Jet-tagging
    - Invariant mass improvement
- 无监督学习 - Unsupervised Learning
  - 使用无标记的数据集来训练算法，试图找到数据中的隐藏特征
    - 聚类、异常检测
- 强化学习
  - 与环境交互，获得反馈，自我提升
    - 数据质量检测（DQM)、多粒子纠缠态制备

机器学习 - 经验

- 经验 - Experience:
  - 数据集
  - 数据集中的每一条数据：样本
- 样本：
  - 属性/特征：
    - Jet-tagging为例：Jet的组分，Jet组分的四动量，Jet的子结构等等
  - 标签：（无监督学习，没有标签）
    - Jet-tagging为例：某个Jet的母粒子是Higgs粒子，W/Z玻色子，top夸克
  - $(\vec{x}_i, y_i)$：第i个样本，其具有$\vec{x}_i = \left(x_i^{(1)}, \cdots, x_i^{(n)}\right)$特征，以及标签$y_i$
- 机器学习是希望学到的模型能很好的适用于“新样本”
  - 训练集、验证集、测试集
  - 每个样本都是独立地从某个分布上采样获得 - 独立同分布

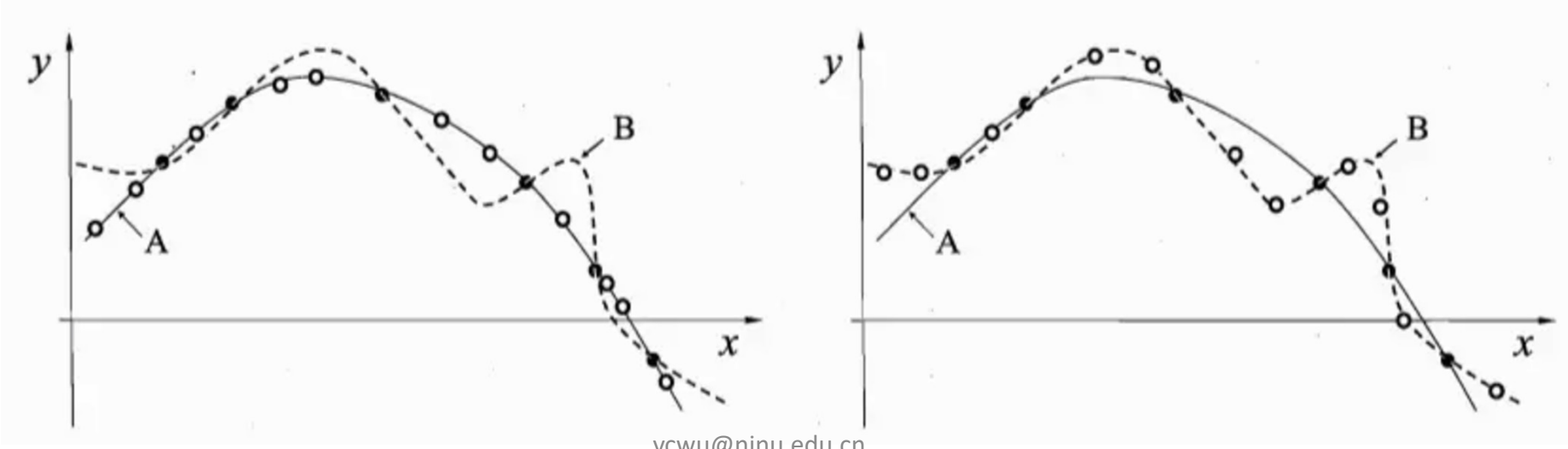
机器学习 - 性能度量

- 归纳偏好 - 训练前对模型好坏的偏好
  - 模型A和模型B在有限样本上一致
    - 谁更好？
  - 学习算法的倾向性（价值观）
- 奥卡姆剃刀
  - 若有多个假设与观察一致，选最简单的那个
    - 朴素的想法：A曲线更简单，更好
      - 更一般的情况中，并没有一个简单的判断手段



机器学习 - 性能度量

- 归纳偏好 - 训练前对模型好坏的偏好
  - 学习算法的倾向性
- No Free Lunch Theorem (NFL) 没有免费的午餐
  - 对于一个学习算法L_a（比如会倾向模型A）若在某些问题上比另一个学习算法L_b（比如会倾向模型B）更好（比如左图）
  - 那必定存在另一些问题，学习算法L_b比L_a更好（比如右图）
  - 极端情况（L_a是某些精巧的模型，L_b是随机猜测模型）也成立



机器学习 - 性能度量

- No Free Lunch Theorem（NFL）没有免费的午餐
  - 离散问题上的简单讨论
    - 样本空间$\mathcal{X}$和假设空间$\mathcal{H}$都是离散的（比如一个简单的二分类问题）
      - 样本空间中可以有一部分作为训练集X，那相应的测试集为$\mathcal{X}-X$
      - 假设空间中的每一条假设：一个从样本空间到最终预测结果的映射$\mathcal{X}\mapsto\{0,1\}$
        - 对二分类而言，最大的假设空间大小为：$2^{|\mathcal{X}|}$
        - 设真实的映射是$f:\mathcal{X}\mapsto\{0,1\}$
  - 假设$P(h|X,L_a)$为学习算法L_a在训练集X上，得到假设h的概率
  - 学习算法L_a在测试集上的误差的期望

$$\epsilon(L_a|X,f)=\sum_h\sum_{x\in\mathcal{X}-X}P(x)\delta\big(h(x)-f(x)\big)P(h|X,L_a)$$

样本的概率分布

$$\delta(x)=\begin{cases}0, & x\neq 0\\ 1, & x=0\end{cases}$$

机器学习 - 性能度量

- No Free Lunch Theorem（NFL）没有免费的午餐
  - 学习算法L_a在测试集上的误差的期望
    - 针对某一特殊任务 - 即为了得到$f: \mathcal{X} \mapsto \{0,1\}$

$$\epsilon(L_a|X,f) = \sum_h \sum_{x \in \mathcal{X}-X} P(x)\delta\big(h(x) - f(x)\big)P(h|X,L_a)$$

  - 对所有任务求和（对所有f）求和 - 假设均匀分布

$$\epsilon(L_a|X) = \sum_f \epsilon(L_a|X,f) = \sum_f \sum_h \sum_{x \in \mathcal{X}-X} P(x)\delta\big(h(x) - f(x)\big)P(h|X,L_a)$$

$$= \sum_x P(x) \sum_h P(h|X,L_a) \sum_f \delta\big(h(x) - f(x)\big)$$

$$\sum_f \delta\big(h(x) - f(x)\big) = \frac{1}{2}\, 2^{|\mathcal{X}|} = 2^{|\mathcal{X}|-1}$$

f均匀分布，将有一半的f和假设h对这个样本x的预测不一致

机器学习 - 性能度量

- No Free Lunch Theorem（NFL）没有免费的午餐
  - 学习算法L_a在测试集上的误差的期望，对所有任务求和（对所有f）求和

$$\epsilon(L_a|X) = \sum_f \epsilon(L_a|X,f) = \sum_f \sum_h \sum_{x\in\mathcal{X}-X} P(x)\delta\big(h(x)-f(x)\big)P(h|X,L_a)$$

$$= 2^{|\mathcal{X}|-1} \sum_x P(x) \sum_h P(h|X,L_a)$$

$$= 2^{|\mathcal{X}|-1} \sum_x P(x)\cdot 1$$

  - 总误差$\epsilon(L_a|X)$与算法无关
    - 如果算法L_a在某个任务上更好，那肯定存在某些任务，算法L_a的效果很差

机器学习 - 性能度量

- No Free Lunch Theorem (NFL) 没有免费的午餐
  - 总误差$\epsilon(L_a|X)$与算法无关
    - 如果算法L_a在某个任务上更好，那肯定存在某些任务,算法L_a的效果很差
  - 在这种情况下，任何算法都和随机猜测差不多
    - 那我们还学什么？
- 两个问题
  - 对所有任务的总误差期望 & 所有任务均匀分布
- 对算法的偏好需要和具体问题结合
  - 没有一个算法能适用所有问题，如果有，那它和随机猜测没什么区别
- 模型选择：MLP，BDT, CNN，RNN，GNN，Transformer

机器学习 - 性能度量

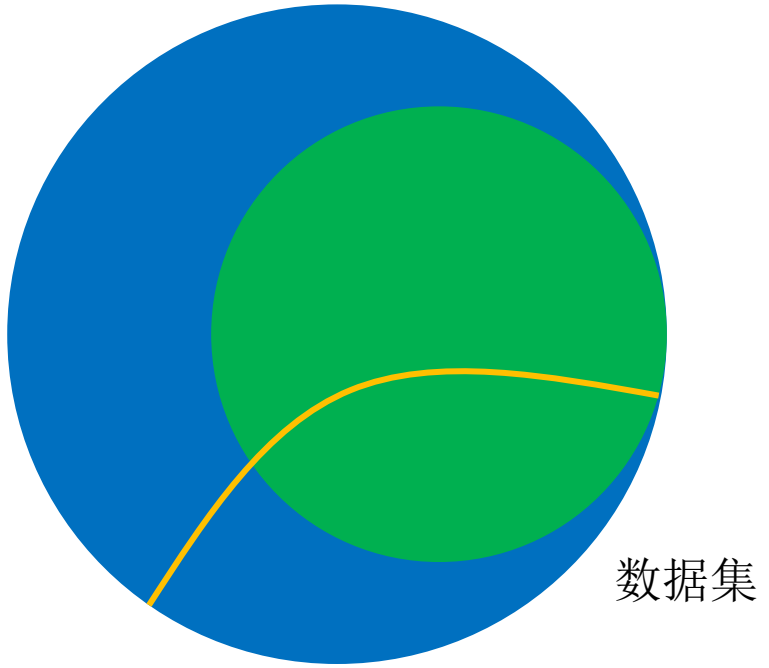
- 模型评估 - 训练后对模型好坏的评判，从而能够做模型选择
- 误差：
  - 模型的预测输出与样本真实值之间的差别
  - 训练误差 - 在训练集上
  - 泛化误差 - 在新样本上
    - 真正关心的是泛化误差
- 过拟合 vs. 欠拟合
  - 过拟合：在训练集上学得过于好了，以至于把一些训练样本本身的特征当做一般性质，导致泛化性能下降
  - 欠拟合：对训练样本的一些一般性质也没有学好
    - 增加模型复杂度、增加训练轮次

机器学习 - 性能度量

- 模型评估 - 训练后对模型好坏的评判，从而能够做模型选择
- 评估方法 - 估计泛化误差
  - 验证集 - 计算误差
    - 用验证集上误差来作为泛化误差的估计
- 从 数据集 到 训练集 + 验证集
  - 假设有一个庞大的包含m个样本的数据集$D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}$
  - 如何划分？
    - 留出法
    - 交叉验证法
    - 自助法

机器学习 - 性能度量

- 数据集划分 - 留出法
  - 直接将数据集D划分为两个互斥的集合
    - $D = S \cup T,\ S \cap T = \emptyset$
  - 保持数据分布一致性
    - 正例 vs. 反例
    - 分层采样(stratified sampling)



数据集

训练集

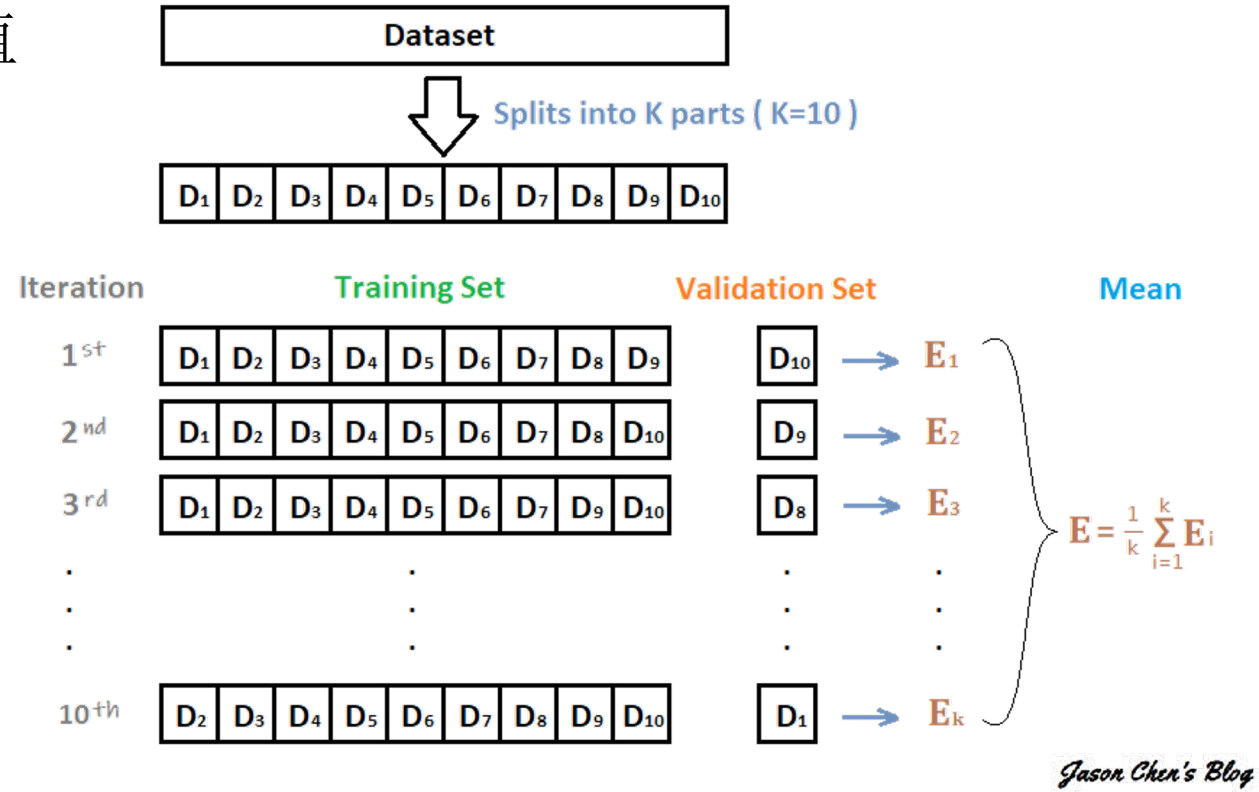
验证集

机器学习 - 性能度量

- 数据集划分 - 留出法
  - 直接将数据集D划分为两个互斥的集合
    - $D = S \cup T,\ S \cap T = \emptyset$
  - 保持数据分布一致性
    - 正例 vs. 反例
    - 分层采样(stratified sampling)
- 多次留出法 - 单次留出法结果不够稳定可靠
  - 采用多次随机划分，分别对模型进行评估 - 平均值作为最终评估结果
- 可能的问题
  - 训练集和验证集的大小
  - 训练集大，验证集小，评估结果的方差大
  - 训练集小，验证集大，评估结果的偏差大

机器学习 － 性能度量

- 数据集划分 － 交叉验证法
  - 先将数据集D划分为K个大小相似的互斥子集$\{D_1, \cdots, D_K\}$
    - 划分方法不一，可以采用不同划分方法重复评估
  - 用其中$K-1$个作为训练集，剩下的作为对模型效果的验证
  - 最终评估效果为每次评估的平均值
    - K-Fold cross validation
- 特例 － 留一法
  - $K = |D|$ － K为数据集中样本个数
  - 训练集更完整 － 偏差小
  - 数据集比较大时，计算量过大

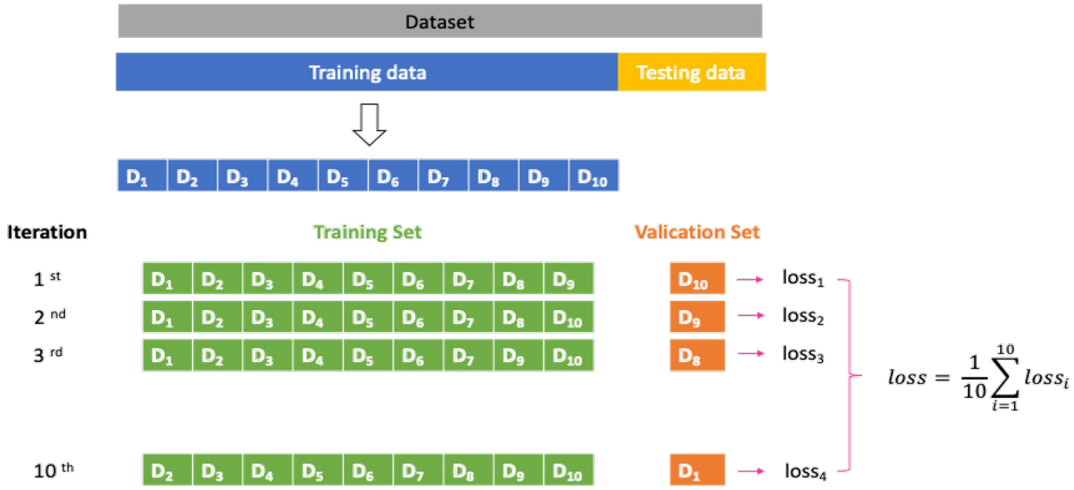


机器学习 - 性能度量

- 数据集划分 - Bootstrapping
  - 给定某个数据集D，$|D| = m$
  - 对数据集D进行m次有放回的随机采样 - 产生数据集D'，$|D'| = m$
    - 某个样本不出现在D'中的概率：$\left(1 - \frac{1}{m}\right)^m \to 36.8\%$
  - 用D'训练，用$D - D'$做模型评估
- 优缺点：
  - 数据量较少时，比较有效
  - 有利于集成学习
  - D'与D的分布可能会有区别，会引入偏差
    - 数据量足够的时候，可以不用Bootstrapping

机器学习 - 性能度量

- 模型评估与选择 - 调参
  - 超参数(hyper-parameters)
    - 模型超参数：网络结构参数
    - 算法超参数：Learning rate, batch size
- 测试集
  - 验证集：模型提供者用来评估模型好坏
  - 测试集：模型使用者实际会用到的数据
  - 测试集和训练集互斥
- 信号背景鉴别问题
  - 训练集、验证集
    - 通常是模拟产生（也可以直接拿真实数据训练）
  - 测试集
    - 实验学家拿模型应用在真实数据上（唯象研究中，用模拟数据代替真实数据）



机器学习 – 性能度量

- 模型评估与选择 – 性能度量
  - 在验证集上表现到底如何
    - 与任务有关，不同任务可能会选择不同的性能度量方法
- 考虑预测问题
  - 给定验证集$D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}$
  - 给出某个训练好的模型$f: X \mapsto Y$
  - 比较$f(\vec{x}_i)$和y_i
  - 均方差：

$$E(f; D) = \frac{1}{m}\sum_{i=1}^{m}(f(\vec{x}_i) - y_i)^2$$

机器学习 – 性能度量

- 考虑分类问题 – 错误率与精度
  - 错误率：分类错误的样本数占总样本数的比例
  - 精度：分类正确的样本数占样本总数的比例
  - $D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}$

$$E(f; D) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(f(\vec{x}_i) \neq y_i)$$

$$acc(f; D) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(f(\vec{x}_i) = y_i) = 1 - E(f; D)$$

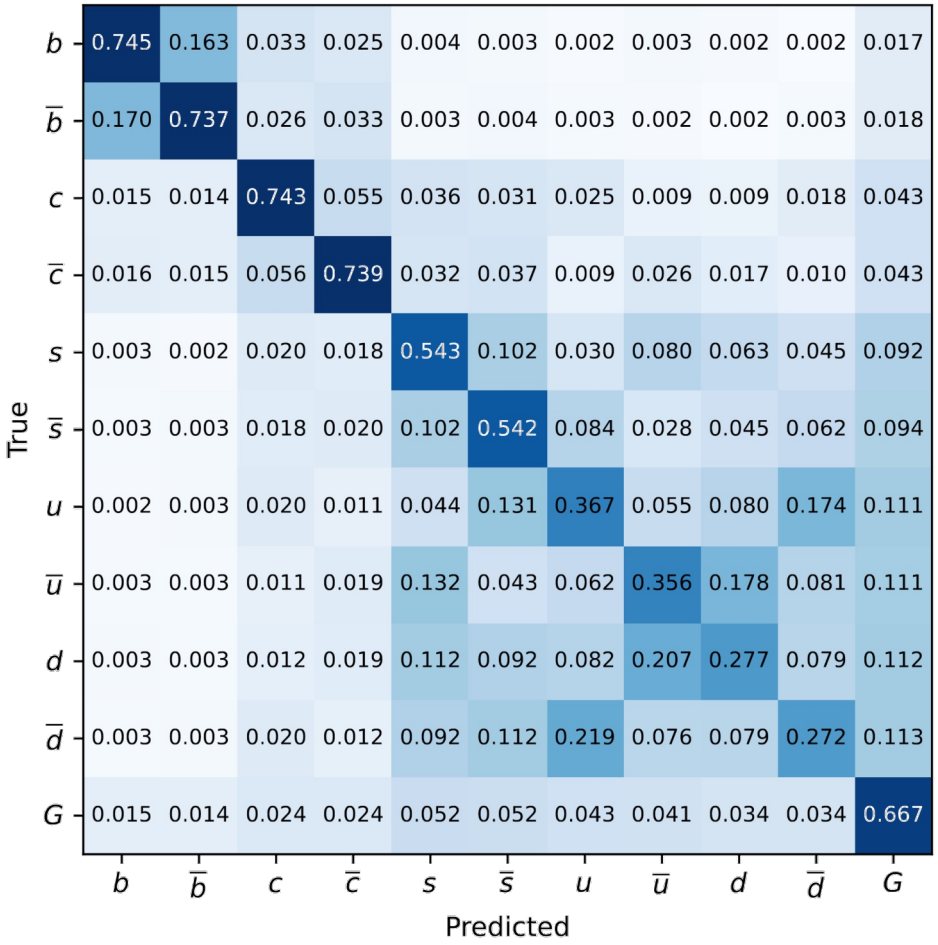
机器学习 - 性能度量

- 分类问题 - 查准率(Precision)与查全率(Recall)
  - 以二分类为例
    - 混淆矩阵 - confusion matrix

真实 \ 预测	正例	反例
正例	True Positive (TP) 真正例	False Negative (FN) 假反例
反例	False Positive (FP) 假正例	True Negative (TN) 真反例

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}$$

Jet-tagging 11分类的混淆矩阵



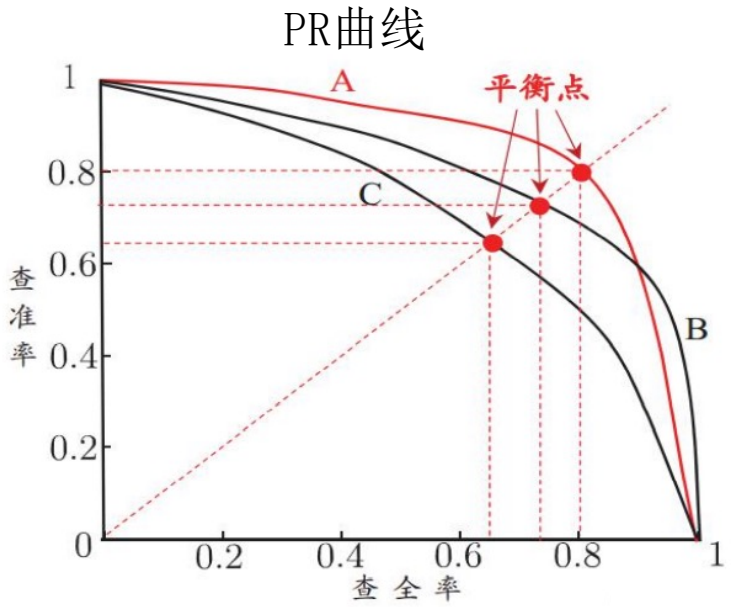
Phys.Rev.Lett. 132 (2024) 221802/arXiv:2310.03440

机器学习 - 性能度量

- 以二分类为例
  - 查准(P)和查全(R)是一对矛盾的变量
    - 为了查准 - 会提高“标准”，任何可疑样本会被预测为反例，降低查全
    - 为了查全 - 会降低“标准”，任何相似样本会被预测为正例，降低查准

真实 \ 预测	正例	反例
正例	True Positive (TP) 真正例	False Negative (FN) 假反例
反例	False Positive (FP) 假正例	True Negative (TN) 真反例

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}$$



性能上：$A > C$，$B > C$
如何比较A和B?

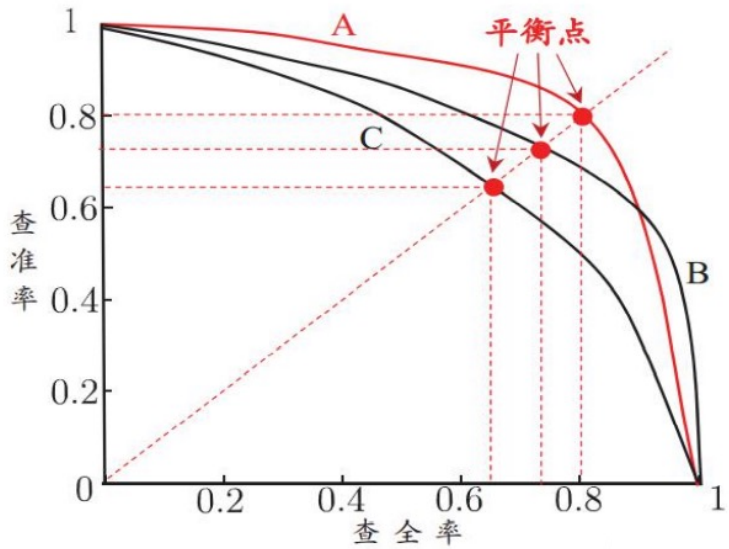
机器学习 - 性能度量

- F度量
  - 加权调和平均

$$F_\beta = \frac{(1+\beta^2)\times P\times R}{\beta^2\times P + R}, \qquad \frac{1}{F_\beta} = \frac{1}{1+\beta^2}\left(\frac{1}{P} + \frac{\beta^2}{R}\right)$$

  - $\beta > 1$时：查全率R更重要
  - $\beta < 1$时：查准率P更重要

PR曲线



性能上：$A > C$，$B > C$
如何比较A和B?

- 例：
  - A模型某个情况下：$P = 0.9, R = 0.4$
  - B模型某个情况下：$P = 0.4, R = 0.9$

$\beta = 2$

$$F_2^A = \frac{5\times 0.9\times 0.4}{4\times 0.9 + 0.4} = 0.45$$

$$F_2^B = \frac{5\times 0.4\times 0.9}{4\times 0.4 + 0.9} = 0.72$$

$\beta = 0.5$

$$F_{0.5}^A = \frac{1.25\times 0.9\times 0.4}{0.25\times 0.9 + 0.4} = 0.72$$

$$F_{0.5}^B = \frac{1.25\times 0.9\times 0.4}{0.25\times 0.4 + 0.9} = 0.45$$

机器学习 - 性能度量

- ROC和AUC
  - ROC曲线
    - 真正例率(True Positive Rate - TPR) vs. 假正例率(False Positive Rate - FPR)

$$TPR = \frac{TP}{TP + FN}$$

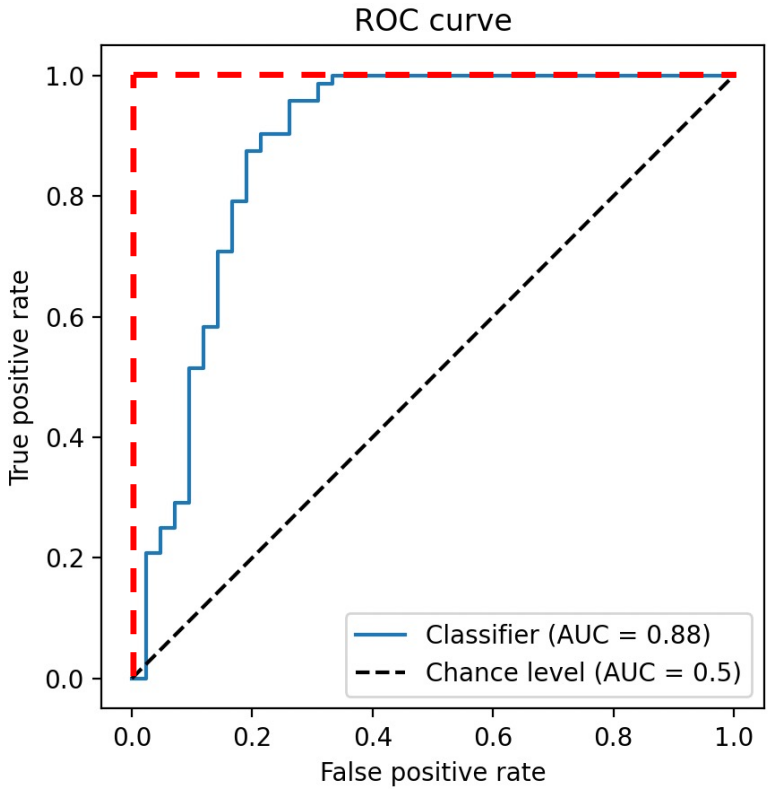
$$FPR = \frac{FP}{FP + TN}$$

- 信号背景分析
  - TPR:
    - 真实信号进入信号区的比例
  - FPR:
    - 真实背景进入信号区的比例
- 理想的模型：TPR=1，FPR=0

真实 \ 预测	正例	反例
正例	True Positive (TP) 真正例	False Negative (FN) 假反例
反例	False Positive (FP) 假正例	True Negative (TN) 真反例

机器学习 - 性能度量

- ROC曲线
  - 对样本分类的预测给出一个[0,1]的数，代表样本为正例的概率
    - 所有样本（正例和反例）可以按照这个概率排序
    - 给定一个概率的阈值，大于阈值预测为正例，其余为反例
      - 每个阈值对应一个混淆矩阵
- AUC - Area under the ROC curve
  - 理想模型的AUC=1
  - 随机猜测模型的AUC=0.5
- ROC曲线 vs. PR曲线
  - ROC更偏向整体性能
  - PR曲线更关注对正例的预测



机器学习 - 性能度量

- 损失函数(loss function)
  - 学习过程中被优化的对象
- 常见的损失函数：
  - 0-1损失函数

$$L(f;X,Y)=\begin{cases}0, & f(X)=Y\\ 1, & f(X)\neq Y\end{cases}$$

  - 平方损失函数

$$L(f;X,Y)=\big(Y-f(X)\big)^2$$

  - 交叉熵损失函数
    - 二分类

$$L(f;X,Y)=Y\ln f(X)+(1-Y)\ln\big(1-f(X)\big)$$

线性模型/神经网络

线性模型

- 基本形式：
  - 样本由d个特征描述：$\vec{x} = \{x_1, x_2, \cdots, x_d\}$
  - 线性模型 - 特征的线性方程

$$f(\vec{x}) = w_1 x_1 + \cdots + w_d x_d + b$$

    - w_i：权重（weight），$\vec{w} = (w_1, \cdots, w_d)$
      - 可解释性：特征的重要性
    - b：偏置（bias）

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b$$

- 线性模型：
  - 线性回归
  - 对数几率回归
  - 多分类学习

线性回归 - Linear Regression

- 线性回归：
  - 试图学习一个线性模型来尽可能的准确预测实数范围内的标签
- 假设样本只具有单个特征：
  - 数据：$D = \{(x_1, y_1), \cdots, (x_m, y_m)\}$
    - 假设特征$x_i \in \mathbb{R}$

$$f(x_i) = wx_i + b \quad \text{vs.} \quad y_i$$

  - 损失函数 - 平方损失函数 - 对所有样本 - 均方差

$$L(f; D) = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 = \frac{1}{m}\sum_{i}^{m}(wx_i + b - y_i)^2 \equiv E(w, b; D)$$

    - 最小二乘法 - 线性拟合

$$(w^*, b^*) = \arg\min_{(w,b)} L(f; D) = \arg\min_{(w,b)} E(w, b; D)$$

线性回归 - Linear Regression

- 模型参数估计 “手动”学习

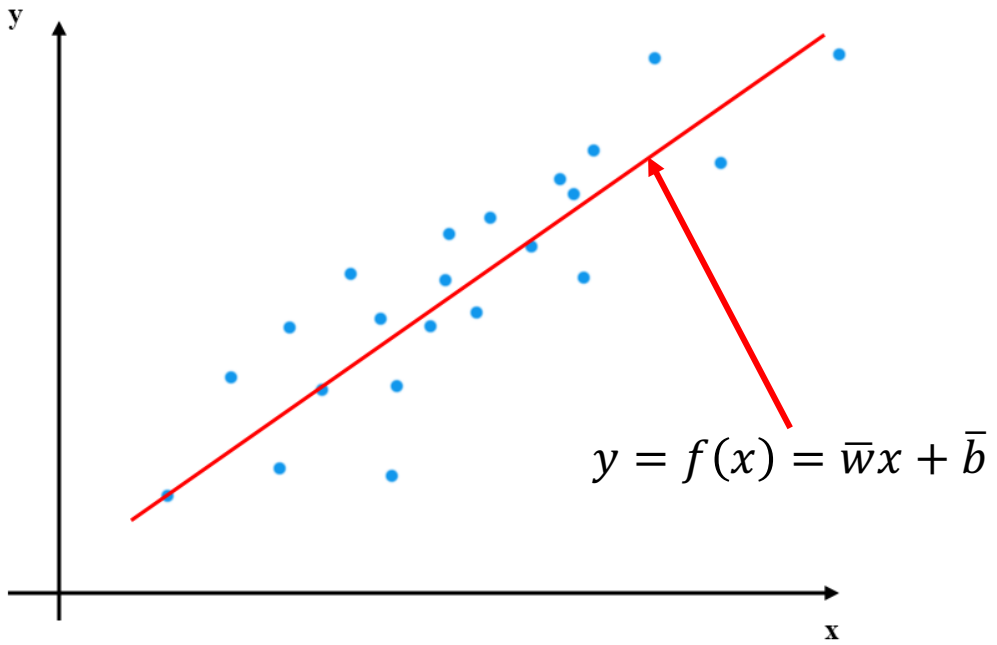
$$(w^*, b^*) = \arg\min_{(w,b)} L(f; D) = \arg\min_{(w,b)} E(w, b; D)$$

$$E(w, b; D) = \frac{1}{m}\sum_{i}^{m}(wx_i + b - y_i)^2$$

$$0 = \left.\frac{\partial E}{\partial w}\right|_{(w^*,b^*)} = \frac{2}{m}\left(w^*\sum_{i=1}^{m}x_i^2 - \sum_{i=1}^{m}(y_i - b^*)x_i\right)$$

$$0 = \left.\frac{\partial E}{\partial b}\right|_{(w^*,b^*)} = \frac{2}{m}\left(mb^* - \sum_{i}^{m}(y_i - w^*x_i)\right)$$

$$w^* = \frac{\sum y_i(x_i - \bar{x})}{\sum x_i^2 - \frac{1}{m}(\sum x_i)^2}, \qquad b^* = \frac{1}{m}\sum(y_i - w^*x_i)$$



线性回归 - Linear Regression

- 一般情况 - 样本由d维特征描述$\vec{x} = \{x_1, x_2, \cdots, x_d\}$
  - 多变量线性回归

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b \quad \Rightarrow \quad \vec{W} \cdot \tilde{\vec{x}} = W_\ell \tilde{x}^\ell, \qquad \ell = 1, \cdots, d, d+1$$

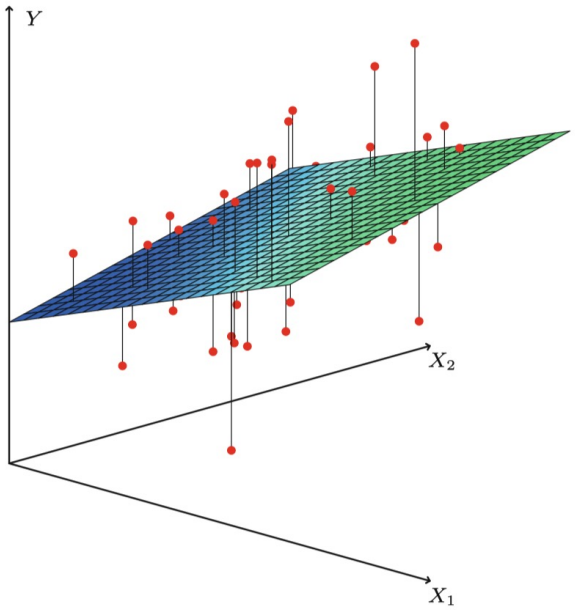
$\vec{W} = (\vec{w}, b)$
$\tilde{\vec{x}} = (\vec{x}, 1)$

  - 均方差损失函数

$$L(f; D) = \frac{1}{m}\left(y_i - W_\ell x_i^\ell\right)\left(y_i - W_k x_i^k\right) \equiv E(\{W_\alpha\}; D)$$

$$0 = \frac{\partial E}{\partial W_\alpha} = \frac{2}{m}\left(W_k x_i^k x_i^\alpha - y_i x_i^\alpha\right) \quad \Rightarrow \quad W_k^*$$

    - $d+1$维空间中的d维超平面



- 广义线性模型

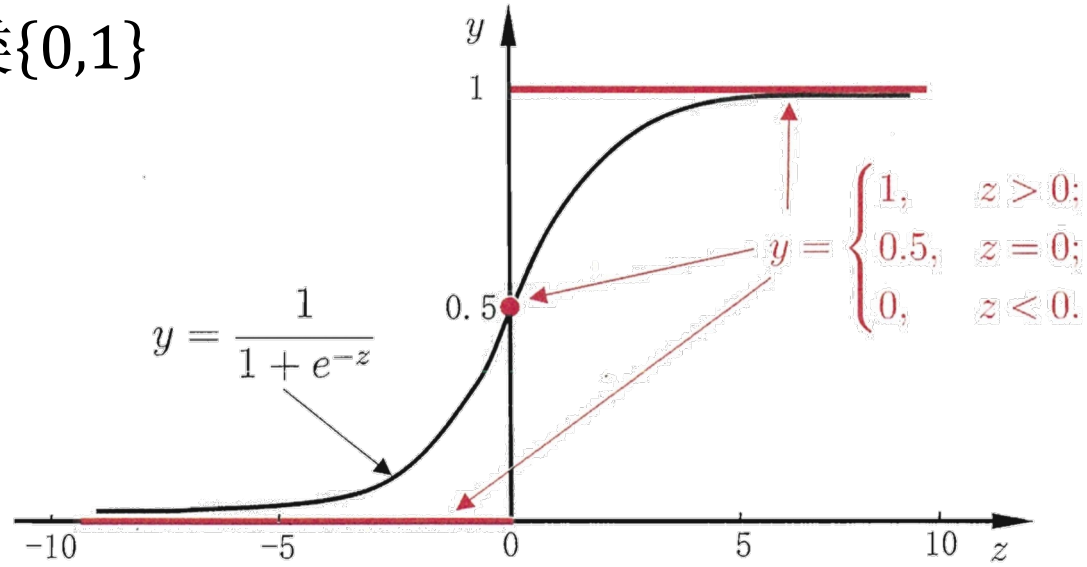
$$f(x) = g(\vec{w} \cdot \vec{x} + b)$$

  - $g(\cdot)$：单调可微函数
    - 虽然显式上模型已经是非线性的，但是X和$g^{-1}(Y)$之间模型关系还是线性的

对数几率回归 - Logistic Regression

- 分类问题
  - 线性模型 - 输出实数：$z = \vec{w} \cdot \vec{x} + b$
  - 分类问题 - 输出离散值；二分类{0,1}

实数值 → 离散值



阶跃(Heaviside)函数

$$g_H(z) = \begin{cases} 0, & z < 0 \\ 0.5, & z = 0 \\ 1, & z > 0 \end{cases}$$

Sigmoid函数

$$g_S(z) = \frac{1}{1+e^{-z}}$$

广义线性变换

$$f(x) = g_S(\vec{w} \cdot \vec{x} + b) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

对数几率回归 - Logistic Regression

- 分类问题
  - 线性模型 - 输出实数：$z = \vec{w} \cdot \vec{x} + b$
  - 分类问题 - 输出离散值；二分类$\{0,1\}$

Sigmoid函数

$$g_S(z) = \frac{1}{1+e^{-z}}$$

$$f(x) = g_S(z) = \frac{1}{1+e^{-(\vec{w}\cdot\vec{x}+b)}} \longrightarrow \ln\frac{f(x)}{1-f(x)} = \vec{w}\cdot\vec{x}+b$$

预测为正例的概率：$p(y=1|\vec{x};\vec{w},b)$

$$p(y=1|\vec{x};\vec{w},b) = \frac{e^z}{1+e^z}$$

$$p(y=0|\vec{x};\vec{w},b) = \frac{1}{1+e^z}$$

$$\ln\frac{p(y=1|\vec{x})}{p(y=0|\vec{x})} = \vec{w}\cdot\vec{x}+b$$

几率(Odds)

对数几率回归

对数几率回归 - Logistic Regression

- 对数几率回归
  - 模型参数估计 - 最大似然法
    - 每个样本属于其真实标签的概率越大越好

$$p_1(\vec{x};\vec{w},b) \equiv p(y=1|\vec{x};\vec{w},b) = \frac{e^z}{1+e^z},$$

$$p_0(\vec{x};\vec{w},b) \equiv p(y=0|\vec{x};\vec{w},b) = \frac{1}{1+e^z}$$

$$L\left(\vec{w},b|(\vec{x}_i,y_i)\right) = \begin{cases} -\ln p_1(\vec{x}_i;\vec{w},b), & y_i = 1 \\ -\ln p_0(\vec{x}_i;\vec{w},b), & y_i = 0 \end{cases}$$

$$= -y_i \ln p_1(\vec{x}_i;\vec{w},b) - (1-y_i)\ln p_0(\vec{x}_i;\vec{w},b)$$

交叉熵损失函数

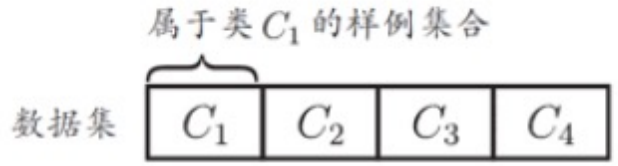
$$L(\vec{w},b|D) = -\frac{1}{m}\sum_i^m (y_i \ln p_1(\vec{x}_i;\vec{w},b) + (1-y_i)\ln p_0(\vec{x}_i;\vec{w},b))$$

线性多分类问题

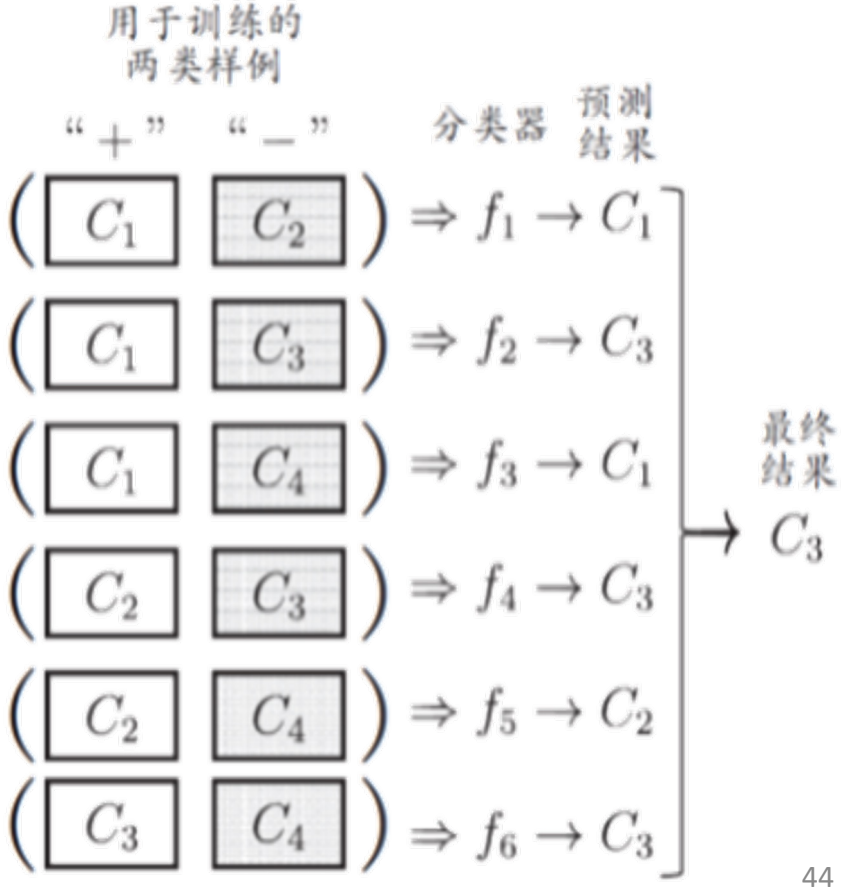
- 线性模型 – 超平面，只能分成两类
  - 多分类问题 – 多个超平面组合
    - 每个超平面单独产生然后组合 – 后组合
    - “直接得到”多个超平面组合 – 多输出
- 多分类 – 多个二分类
  - 考虑数据集中样本属于N个不同的类别$\{C_1, C_2, \cdots, C_N\}$
    - $D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}, y_i \in \{C_1, C_2, \cdots, C_N\}$
  - 多分类到多个二分类的拆分：
    - 一对一(One vs. One; OvO)
    - 一对其余(One vs. Rest; OvR)；有时也称一对所有(One vs. All; OvA)
    - 多对多(Many vs. Many; MvM)

线性多分类问题

- 多分类 - 多个二分类
  - 考虑数据集中样本属于N个不同的类别$\{C_1, C_2, \cdots, C_N\}$
    - $D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}, y_i \in \{C_1, C_2, \cdots, C_N\}$

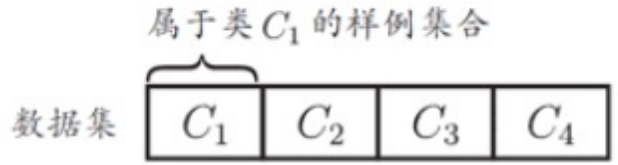


- 一对一 0v0
  - 将数据集D中的N个类别两两配对，
    - 产生$N(N-1)/2$个二分类数据集
      - $D_{ij} = \left\{(\vec{x}_1^{ij}, y_1^{ij}), \cdots, (\vec{x}_\ell^{ij}, y_\ell^{ij})\right\},\ y_\ell^{ij} \in \{C_i, C_j\}$
    - 训练出$N(N-1)/2$个二分类器
  - 最终预测结果：全民投票
  - 类别多时，测试耗时
    - 训练更省时

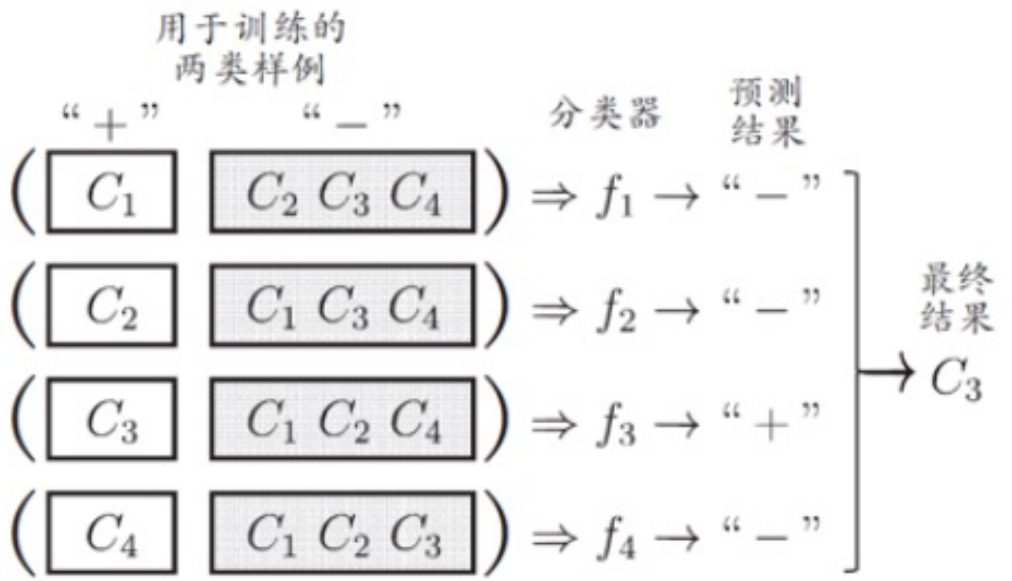


线性多分类问题

- 多分类 - 多个二分类
  - 考虑数据集中样本属于N个不同的类别$\{C_1, C_2, \cdots, C_N\}$
    - $D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}, y_i \in \{C_1, C_2, \cdots, C_N\}$

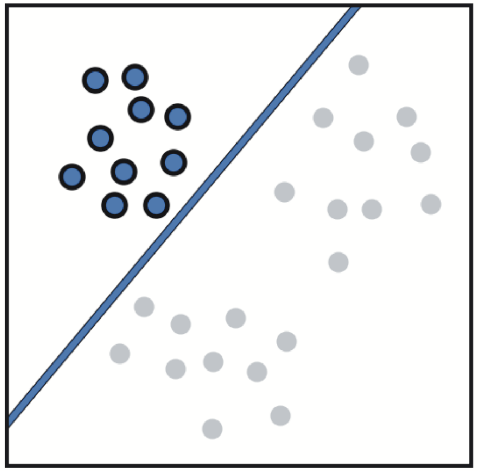


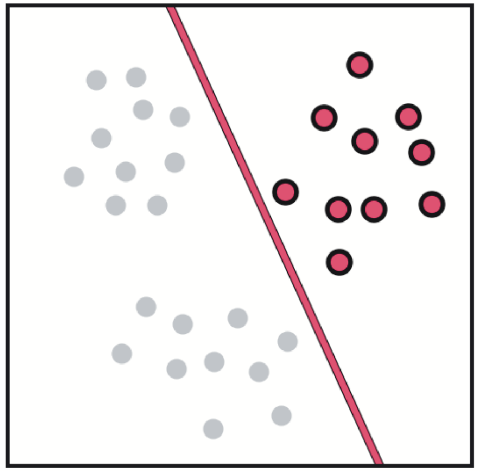
- 一对其余 OvR
  - 将数据集中某个类别C_i作为正例，其余为反例进行二分类学习
    - $D_i = \{(\vec{x}_1, y_1^i), \cdots, (\vec{x}_m, y_m^i)\},\ y_\ell^i = \begin{cases} 1, & y_\ell = C_i \\ 0, & y_\ell \neq C_i \end{cases}$
  - 训练得到N个二分类器

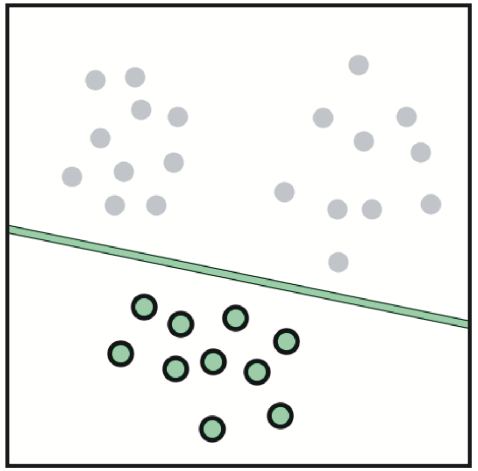


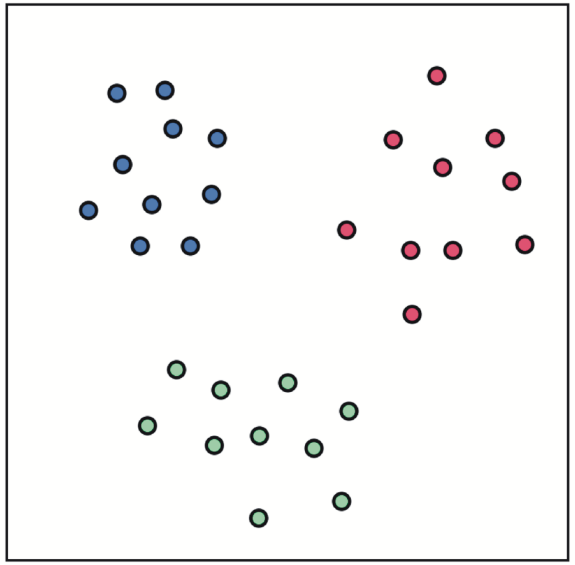
线性多分类问题

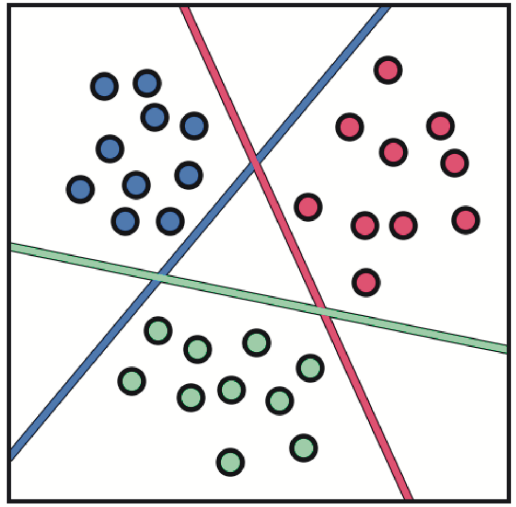
- 一对其余 OvR





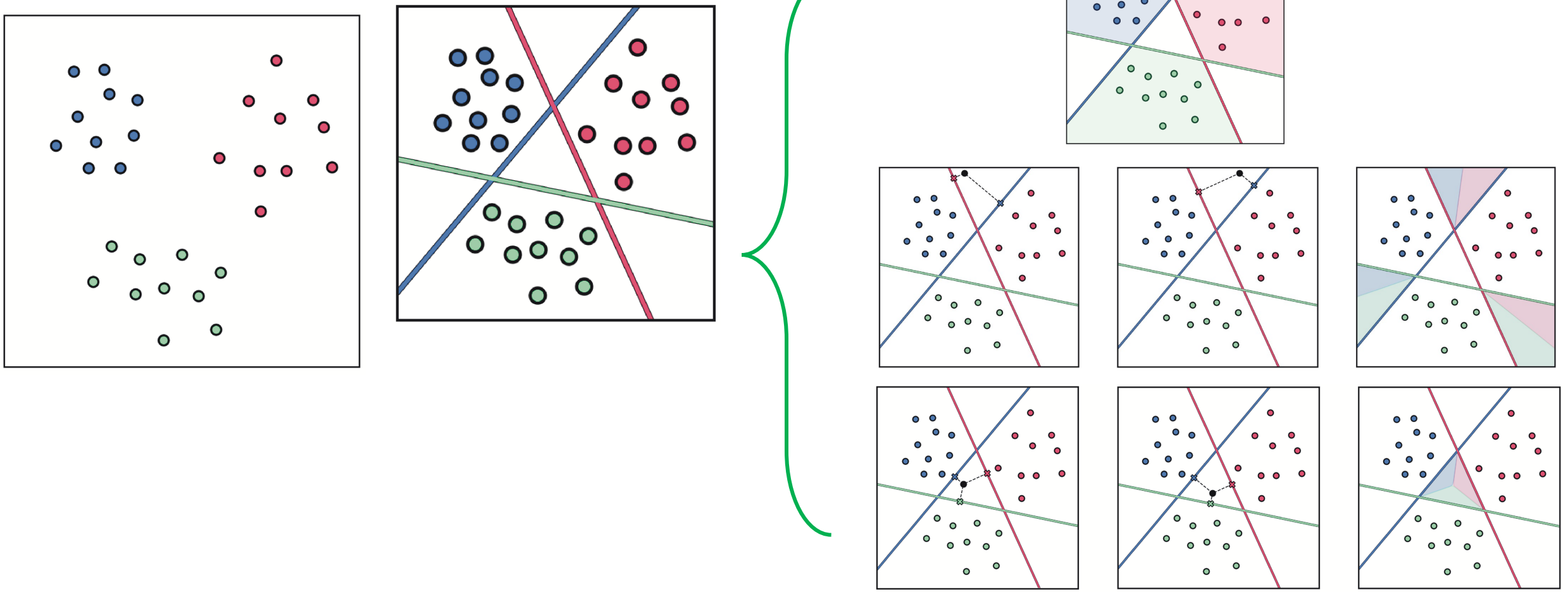






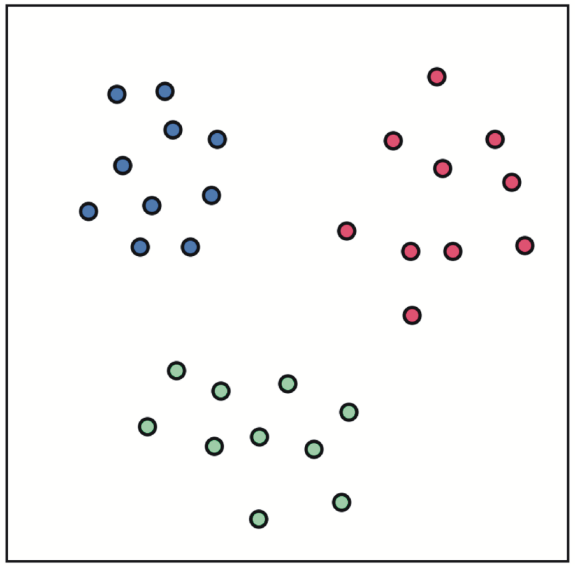
线性多分类问题

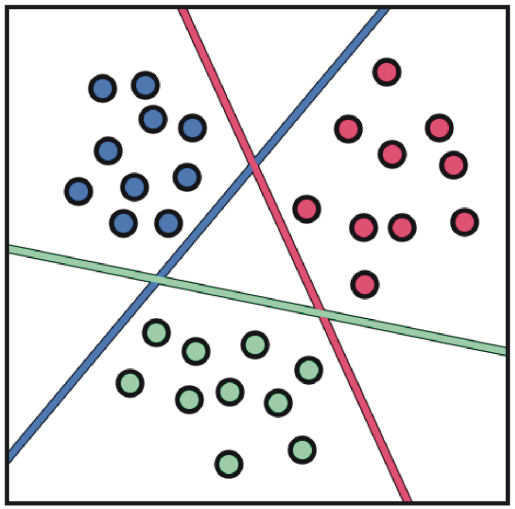
- 一对其余 OvR

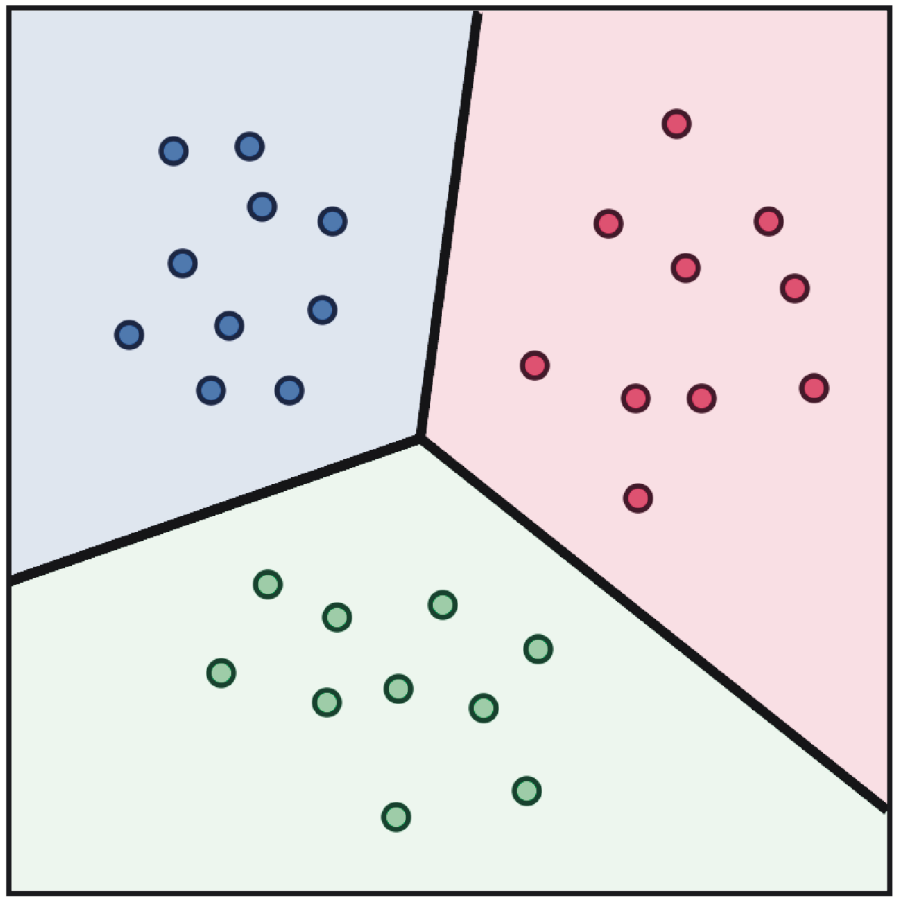


线性多分类问题

- 一对其余 OvR







线性多分类问题

- 多分类 - 多个二分类
  - 考虑数据集中样本属于N个不同的类别$\{C_1, C_2, \cdots, C_N\}$
    - $D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}, y_i \in \{C_1, C_2, \cdots, C_N\}$
- 多对多 MvM
  - 一些类别为正例，其他类别为反例
    - 正例、反例构造 - 纠错输出码
  - 编码 + 解码

编码长度 = 分类器数目

	f_1	f_2	f_3	f_4	f_5	距离
C_1	−1	+1	−1	+1	+1	$2\sqrt{3}$
C_2	+1	−1	−1	+1	−1	4
C_3	−1	+1	+1	−1	+1	2
C_4	−1	−1	+1	+1	−1	$2\sqrt{2}$
测试样本	−1	−1	+1	−1	+1	C_3

每个类别的编码

测试样本通过分类器之后得到的编码

测试样本编码与每个类别编码之间的距离

解码

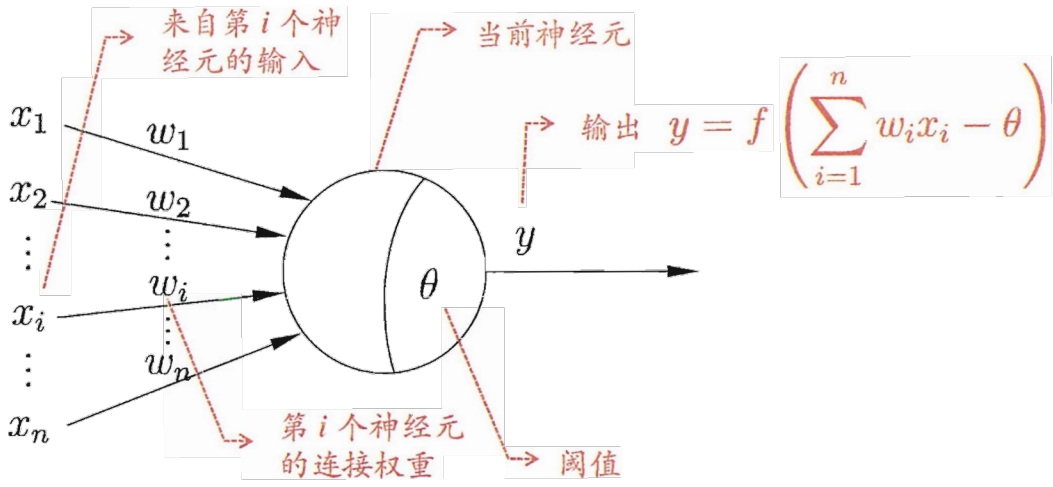
类别不平衡

- 对撞机事例分析：信号 vs 背景
  - 新物理过程截面 ≪ 标准模型过程截面

    10 vs. 990
    - 全预测为标准模型背景，准确率99.9%
- 处理方法：
  - 根据正例、反例样本数调整预测值
    - 训练时调整 - Rescaling
    - 预测时调整 - 阈值调节
  - 欠采样/过采样
  - 加权分类
  - 代价损失函数
  - …

神经网络

- 神经网络：
  - 由具有适应性的简单单元组成的并行互连的网络
    - 模拟生物神经系统对真实世界带来的刺激的交互反应
  - 简单单元 – 神经元
  - 适应性 – 权重和阈值的调整
- 神经元模型
  - 输入
    - 接受一定数目的输入值：$x_1, \cdots, x_n$
  - 处理
    - 连接权重，阈值
    - 激活
  - 输出
  - 广义线性模型 $y = f(\vec{w} \cdot \vec{x} - \theta)$

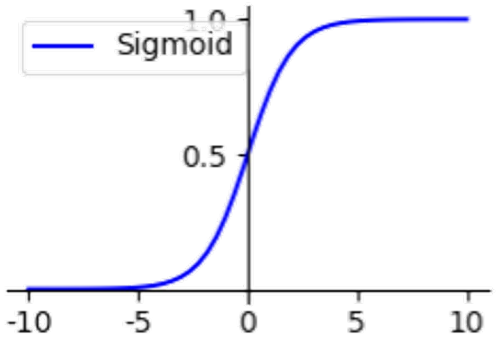


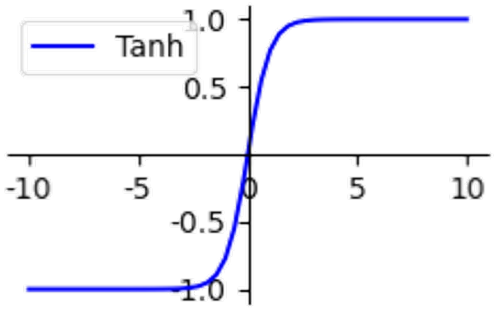
神经元模型

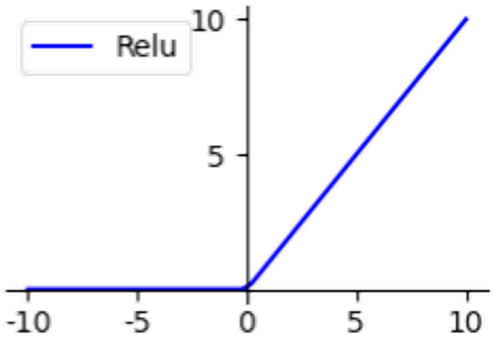
- 神经元模型
  - 输入：$x_1, \cdots, x_n$
  - 处理
  - 输出

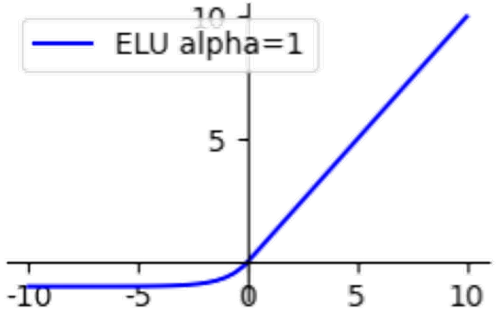
$$y = f(\vec{w} \cdot \vec{x} - \theta)$$

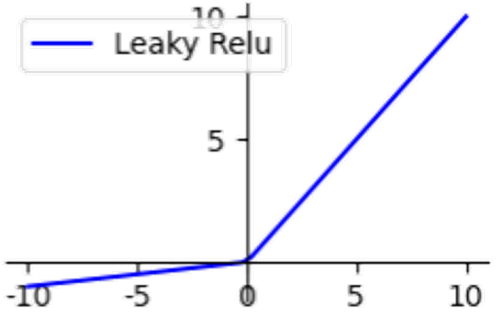
- 激活函数
  - 阶跃函数
  - Sigmoid
  - ReLU
  - Leaky ReLU
  - …





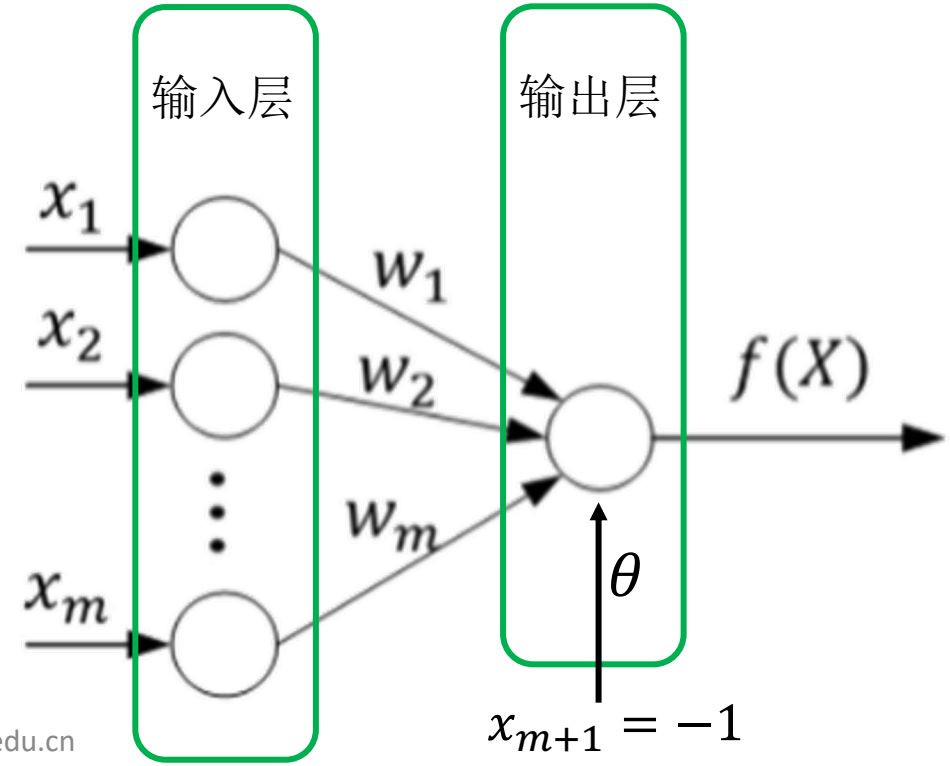






神经网络 - 感知机

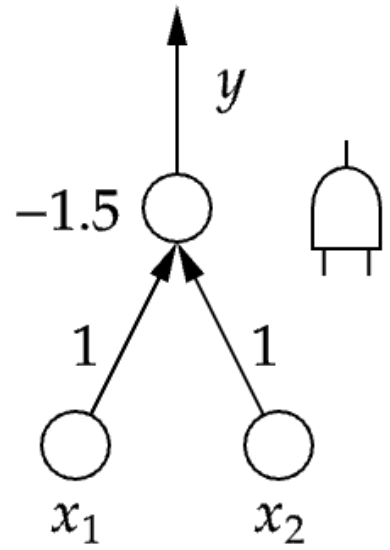
- 神经网络：
  - 由具有适应性的简单单元组成的并行互连的网络
    - 模拟生物神经系统对真实世界带来的刺激的交互反应
  - 简单单元 - 神经元
  - 适应性 - 权重和阈值的调整
- 感知机 - Perceptron
  - 最简单的神经网络
  - 两层神经元
    - 输入层：
      - 接收外界输入直接传递
    - 输出层：
      - 接收输入 - 权重
      - 往外输出 - 阈值、激活



感知机 - 逻辑运算

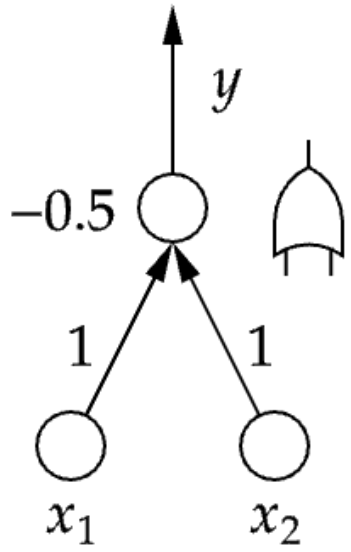
- 感知机
  - 参数合适可组成简单的逻辑运算 - 与、或、非、与非

$y = x_1 \wedge x_2$



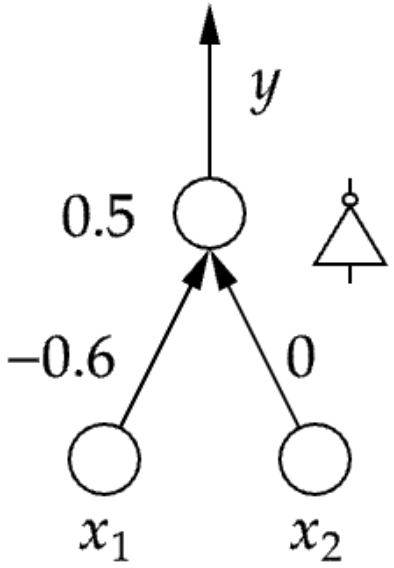
$y = \Theta(x_1 + x_2 - 1.5)$

$y = x_1 \vee x_2$



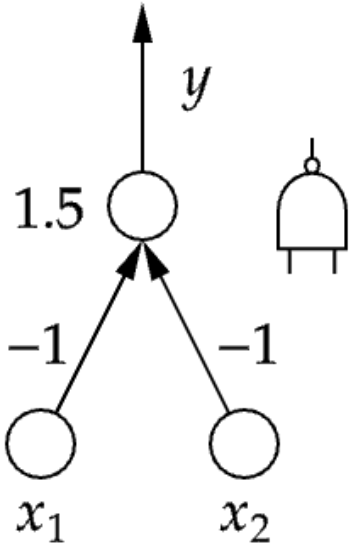
$y = \Theta(x_1 + x_2 - 0.5)$

$y = \neg x_1$



$y = \Theta(-0.6x_1 + 0.5)$

$y = \neg(x_1 \wedge x_2)$



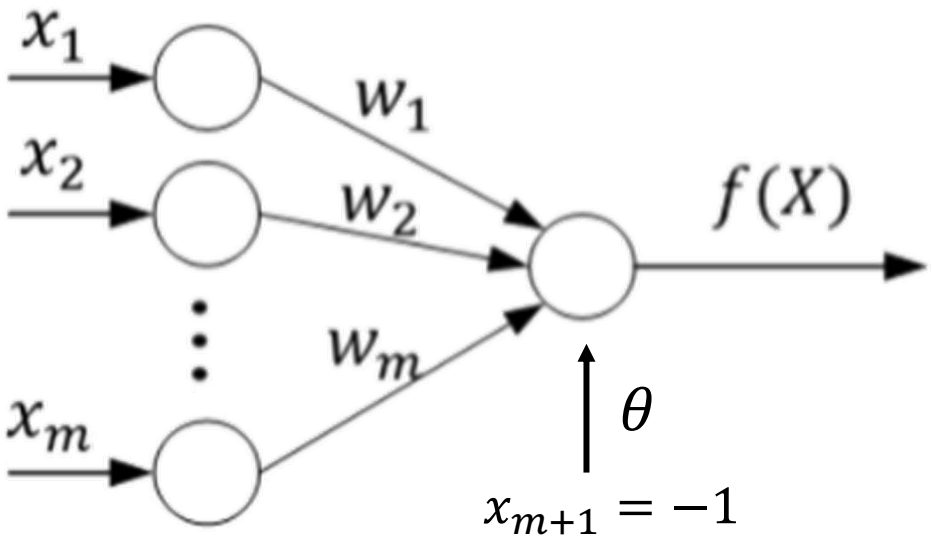
$y = \Theta(-x_1 - x_2 + 1.5)$

感知机

- 感知机
  - 参数合适可组成简单的逻辑运算
  - 参数的学习：以阶跃激活函数为例
    - 对样本$(\vec{x}_i, y_i)$
    - 感知机输出$\hat{y}_i = f(\vec{x}_i, \vec{w})$

$$w_k \leftarrow w_k + \Delta w_k$$

$$\Delta w_k = \eta(y_i - \hat{y}_i)x_i^k$$

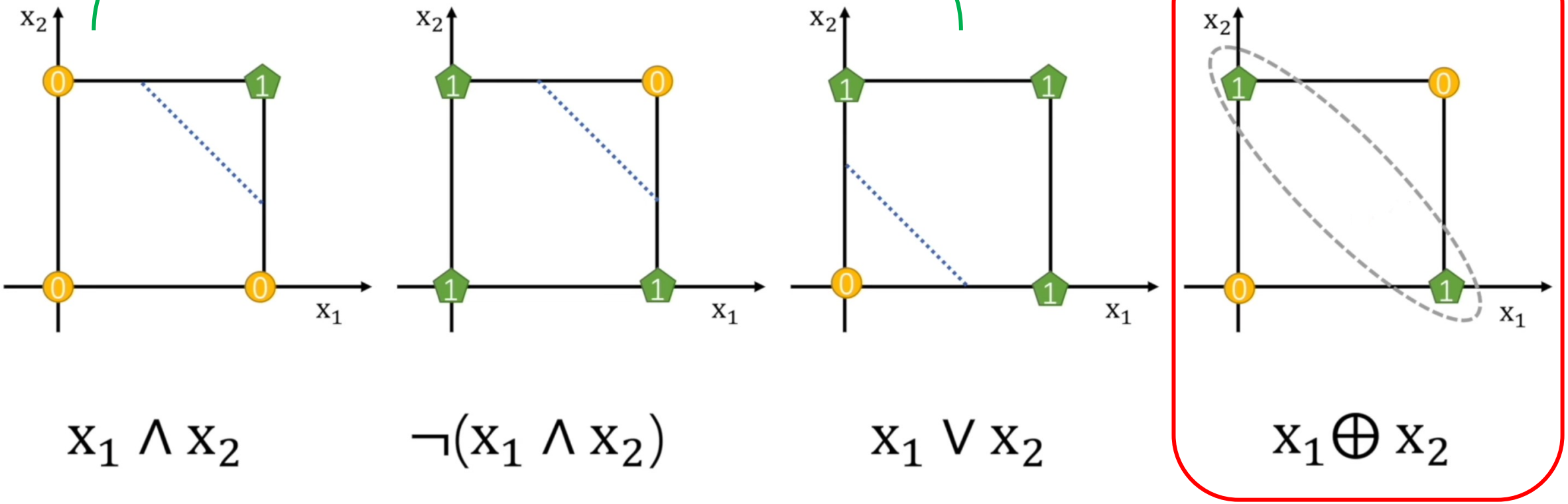


感知机 - 异或问题

- 感知机的局限性

线性可分

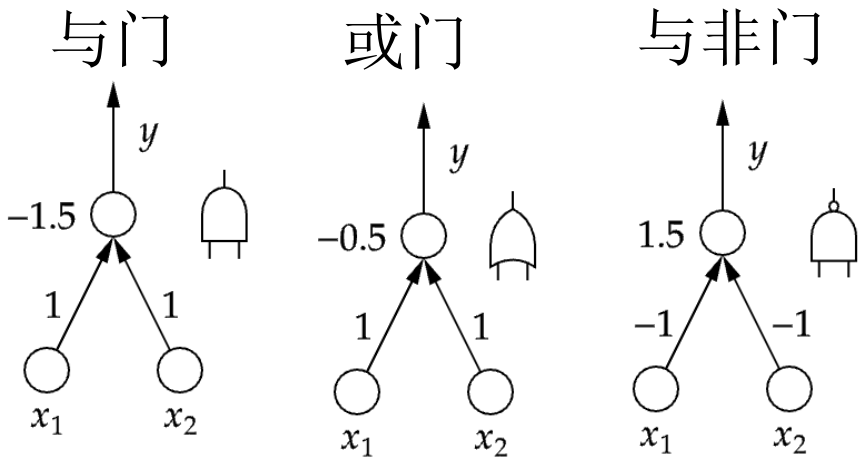
非线性可分
异或问题

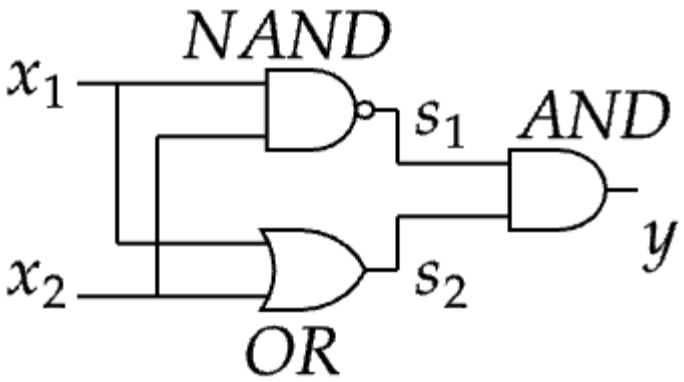


$x_1 \wedge x_2$ $\neg(x_1 \wedge x_2)$ $x_1 \vee x_2$ $x_1 \oplus x_2$

异或问题

- 非线性可分 – 异或问题
  - 异或门
    - 用与门、或门、与非门组合

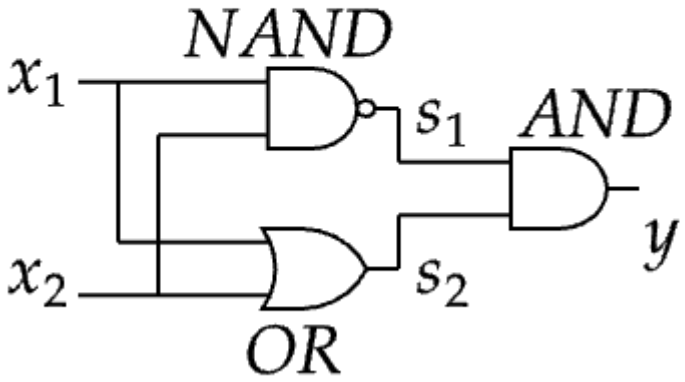




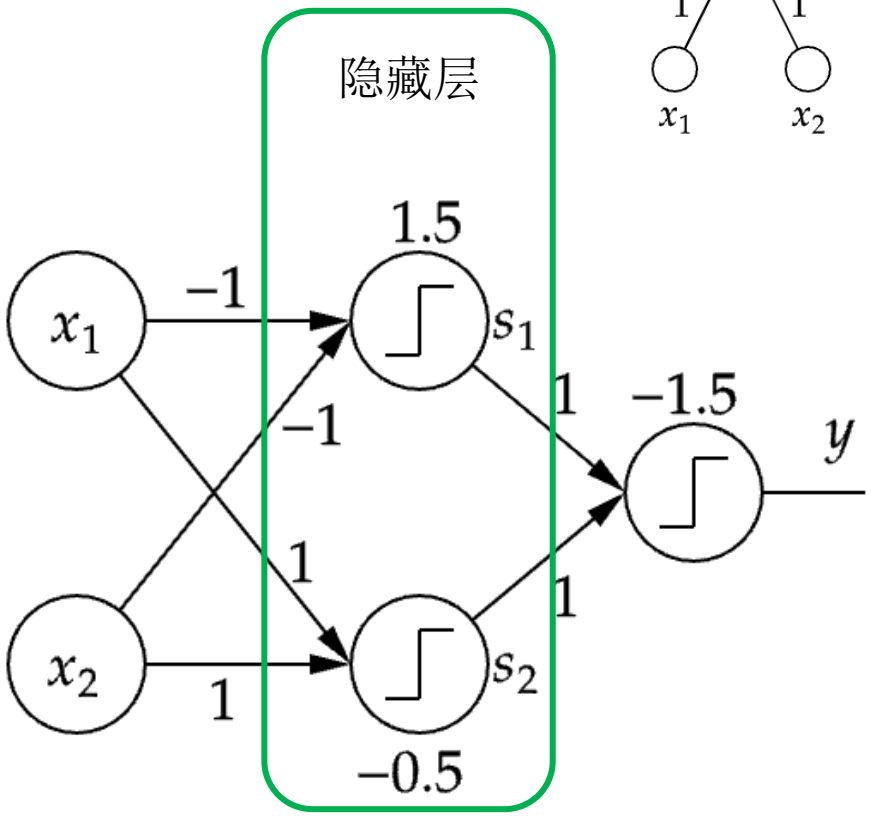
x_1	x_2	s_1	s_2	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

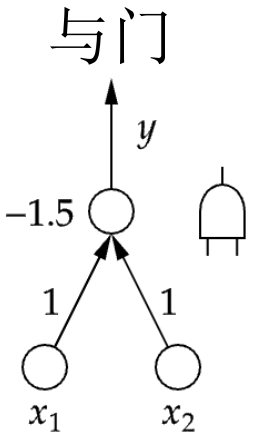
多层神经网络

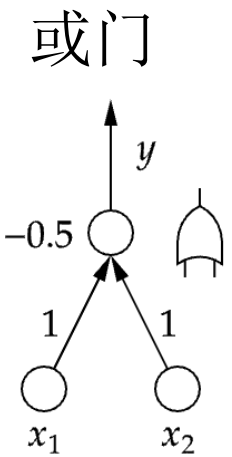
- 非线性可分 – 异或问题
  - 异或门
    - 用与门、或门、与非门组合

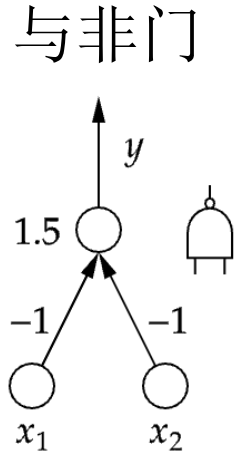


多层神经网络



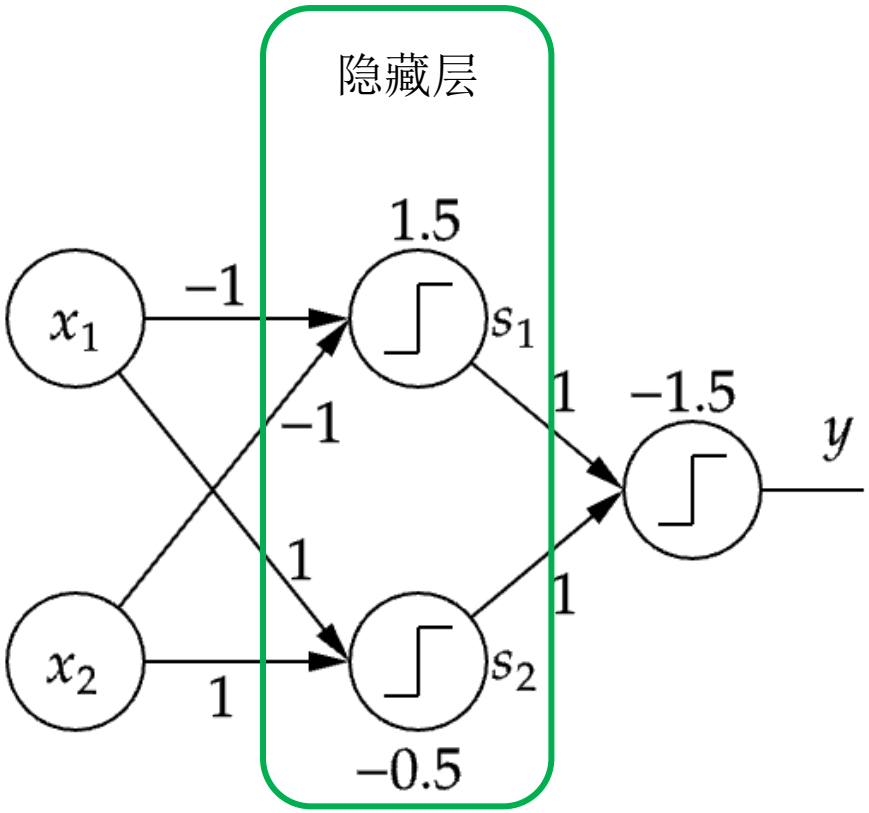


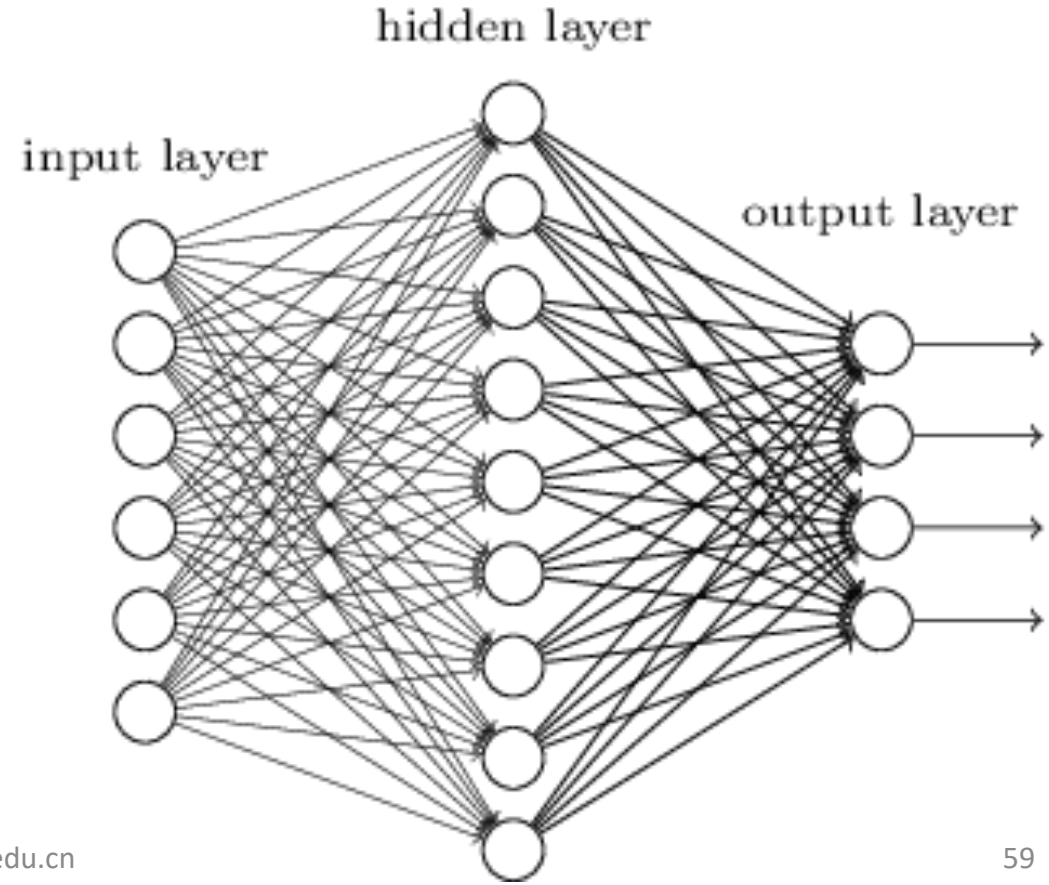




多层前馈神经网络

- 多层前馈神经网络(Multi-layer feedforward neural network)
  - 学习过程：
    - 根据样本来调整神经元之间的连接权重以及每个神经元的阈值(偏置)



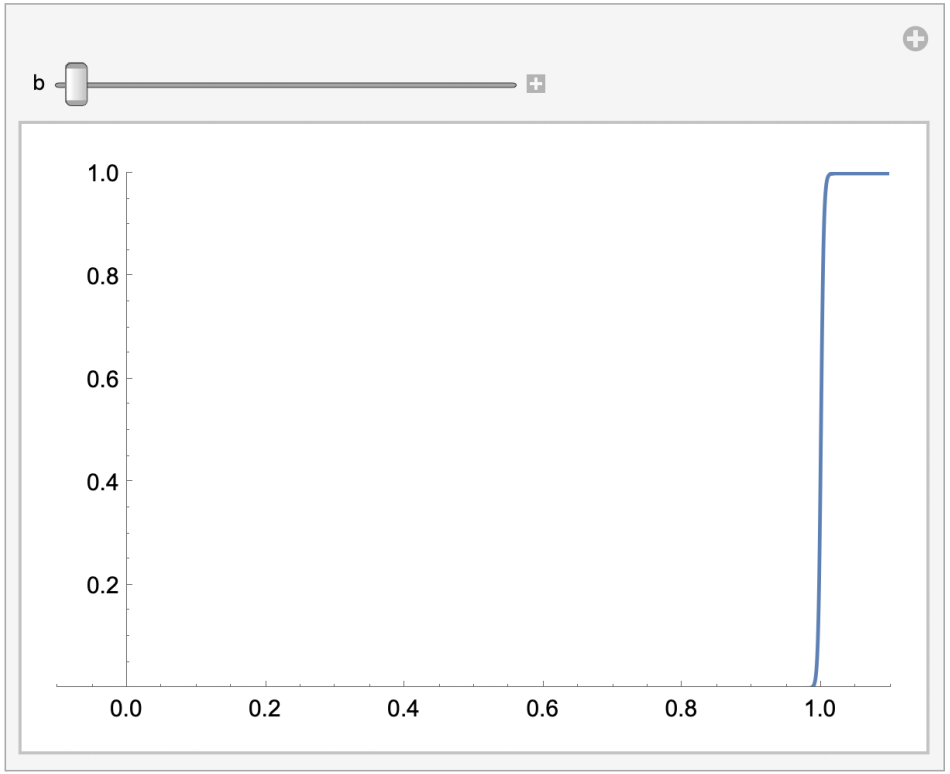


通用近似定理

- 多层神经网络 - 能力巨大
  - 仅用与非门就可以构建整个计算机系统
  - 靠组合最简单的感知机构成更复杂的网络系统
    - 可以产生任何复杂的模型 - 原则上
- 通用/万能近似定理(Universal Approximation Theorem)
  - 总能构造出相应的神经网络，在给定精度下（在相应的函数空间内）逼近任何欧式空间到欧式空间的连续函数
    - 类比：实数中，有理数的稠密性 - 能逼近任何实数
  - 原则上一层隐藏层+足够多隐藏层节点可以做到
    - 但可能没法有效训练 - 没法有效找到对应参数

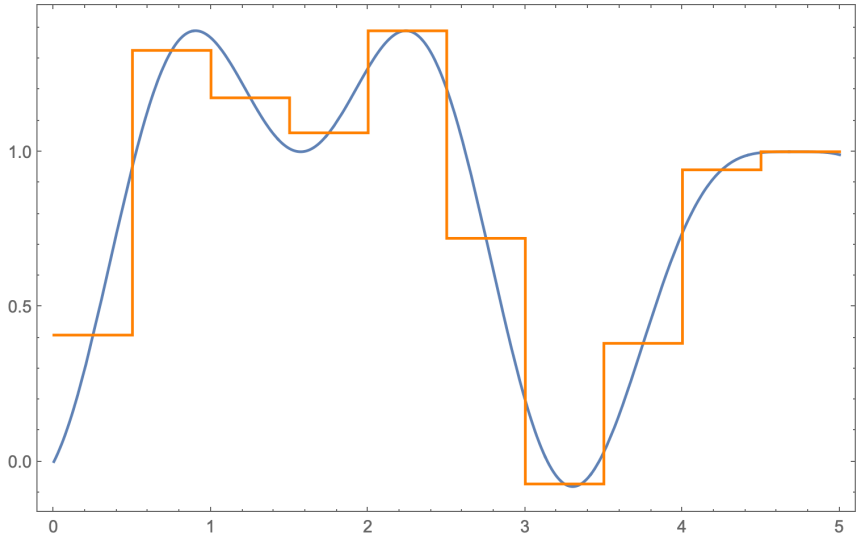
通用近似定理

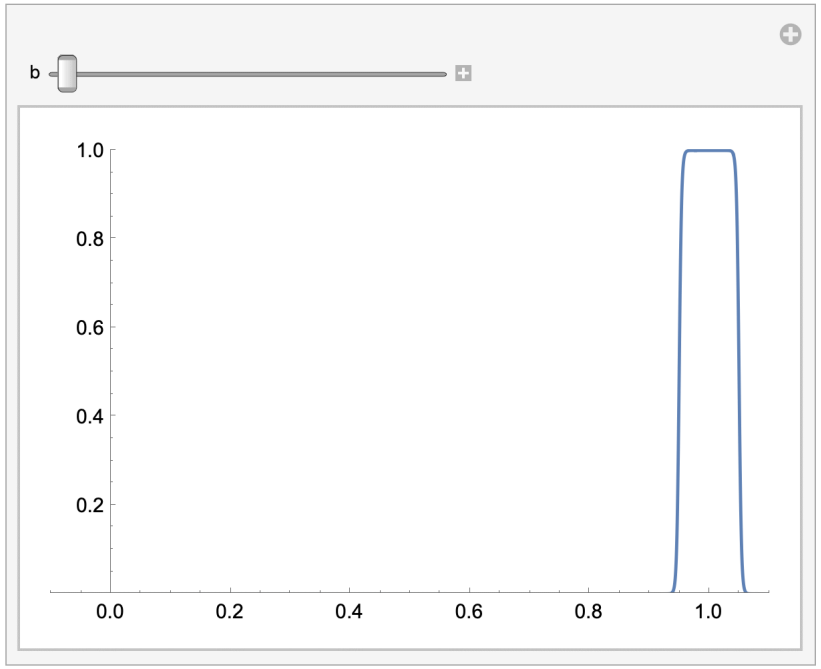
- 通用近似定理 – 一维的直观说明 http://neuralnetworksanddeeplearning.com/chap4.html
  - Sigmoid激活的感知机
    - $y = \frac{1}{1+e^{-z}}$，$z = wx + b$，通过改变w, b可以近似在任意位置的阶跃函数



通用近似定理

- 通用近似定理 - 一维的直观说明
  - Sigmoid激活的感知机
    - $y = \frac{1}{1+e^{-z}}$，$z = wx + b$，通过改变w, b可以近似在任意位置的阶跃函数
  - 两个Sigmoid激活的感知机
    - 通过改变相应的权重/偏置，可以近似在任意位置附近，任意宽度的方形函数
  - 将任意这样的方形函数叠加
    - 近似任意函数



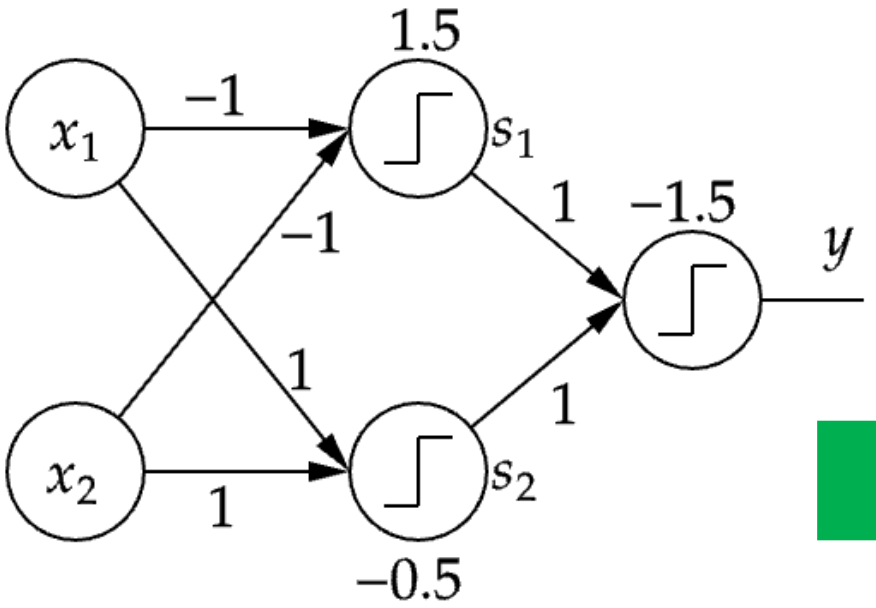


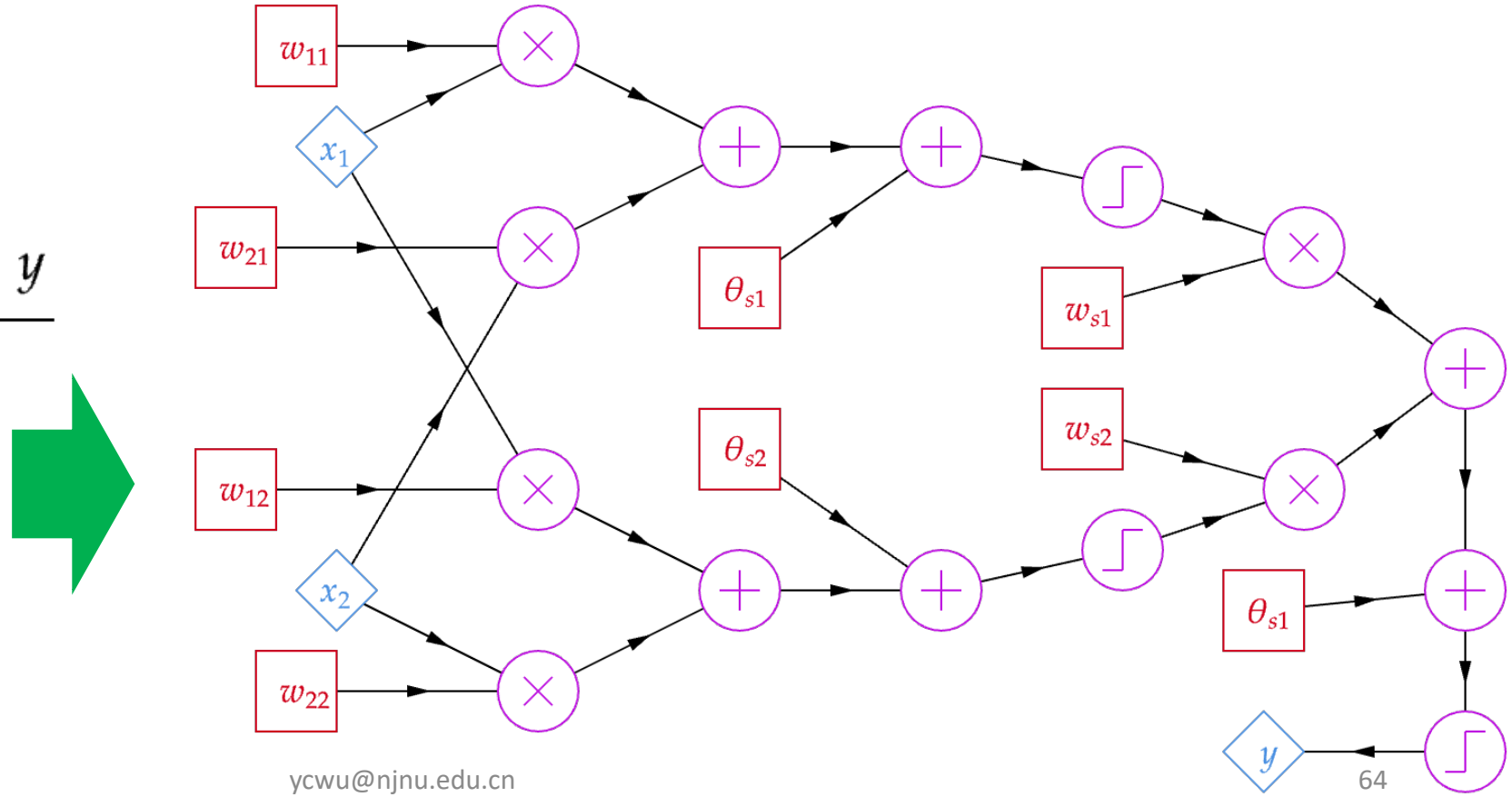
通用近似定理

- 通用/万能近似定理(Universal Approximation Theorem)
  - 总能构造出相应的神经网络，在给定精度下（在相应的函数空间内）逼近任何欧式空间到欧式空间的连续函数
    - 类比：实数中，有理数的稠密性 - 能逼近任何实数
  - 原则上一层隐藏层+足够多隐藏层节点可以做到
    - 但可能没法有效训练 - 没法有效找到对应参数
  - 网络宽度 vs. 网络深度
    - 模型复杂度：
      - 深度的线性增加 vs. 宽度的指数增加
    - 特征学习 - 逐层处理/特征变换、组合
  - 深层网络 - Deep Learning
    - 不同层处理不同的任务

反向传播 – 学习/更新参数

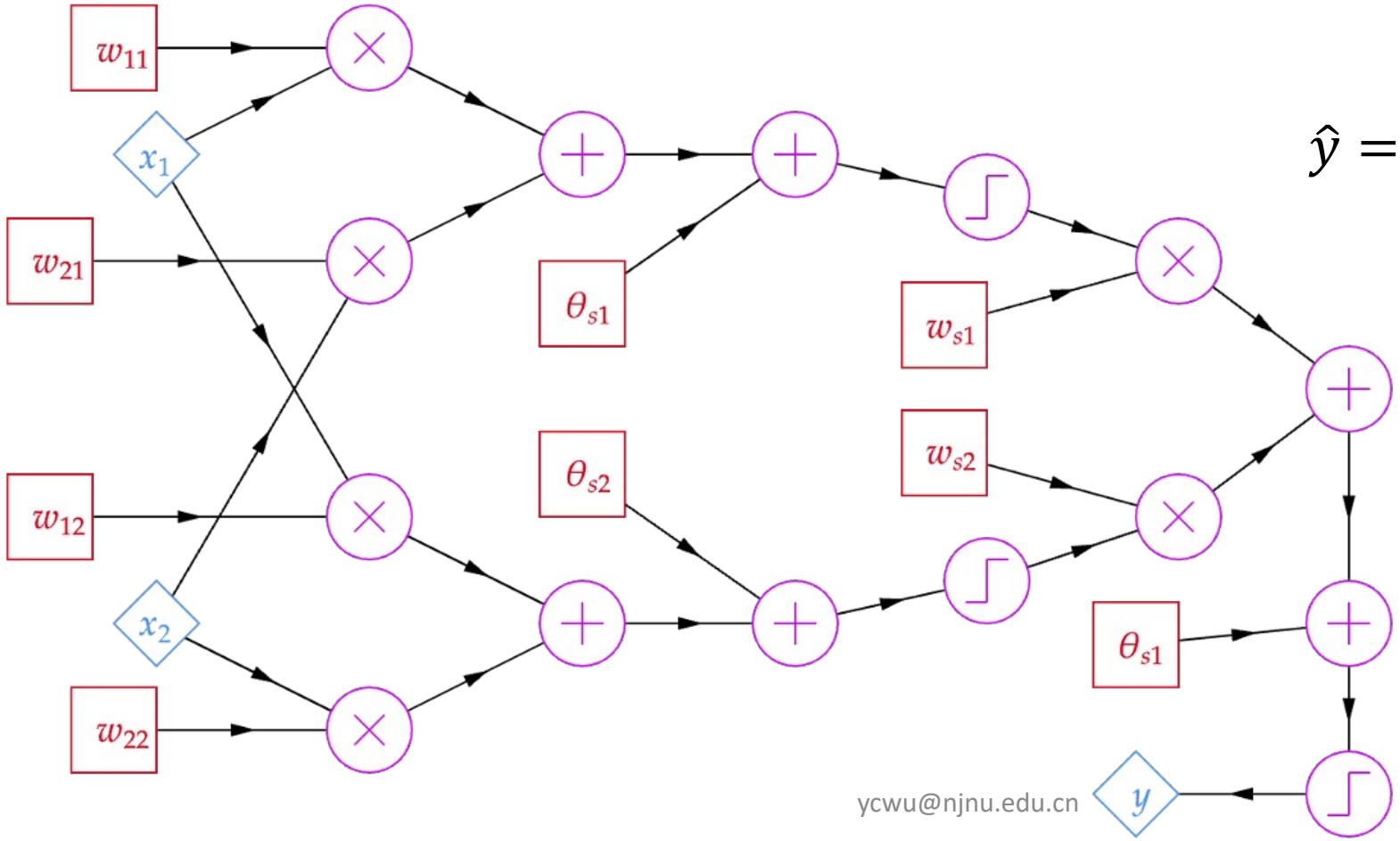
- 反向传播算法 – 神经网络学习算法
  - 如何从数据中学得神经网络的参数 – 权重和偏置
- 神经网络的前向传播 – 计算图





计算图

- 计算图
  - 前向传播 - 计算过程沿着箭头方向流到最终结果
    - 每个节点只负责局域的计算：接收输入，计算输出



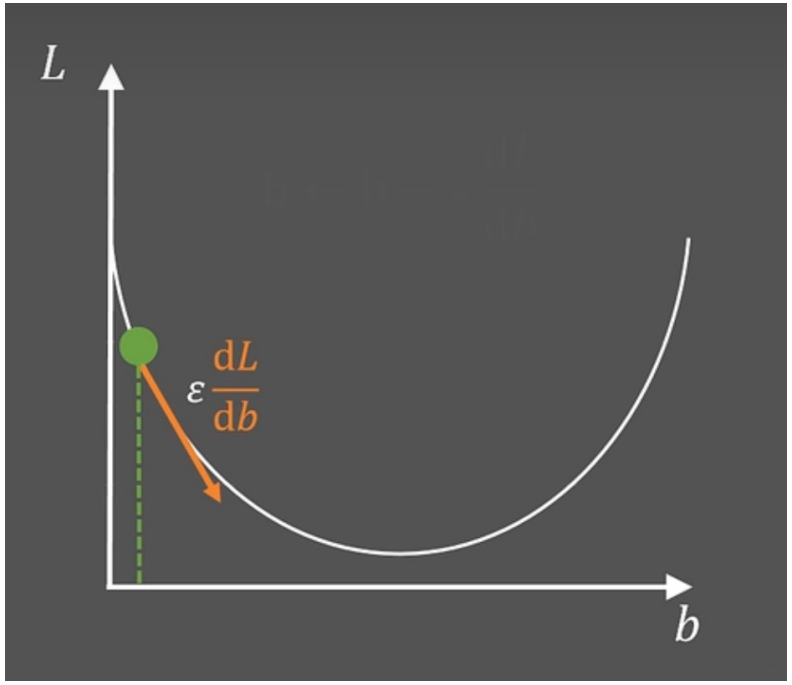
$$\hat{y} = f(\{w, \theta\}; x_1, x_2)$$

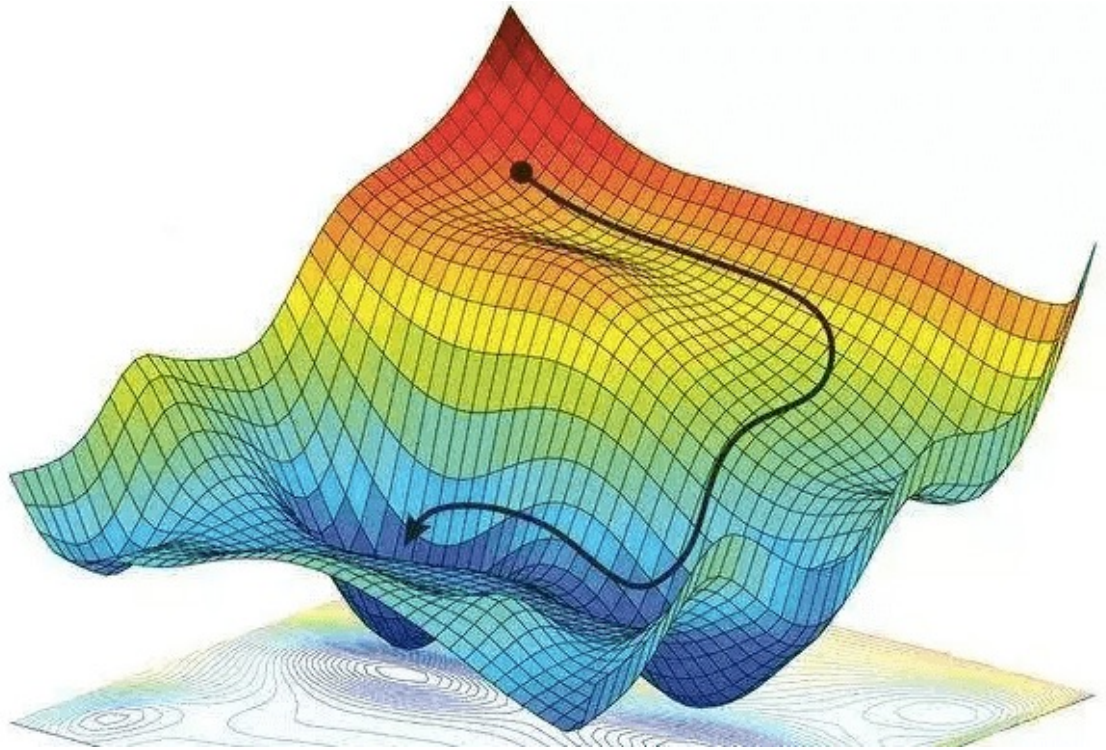
梯度下降

- 神经网络的学习 – 有监督
  - 最小化损失函数 – 通过调整权重和偏置

$$E = \frac{1}{m}\sum_{i}^{m}(y_i - \hat{y}_i)^2$$

  - 梯度下降法





梯度下降

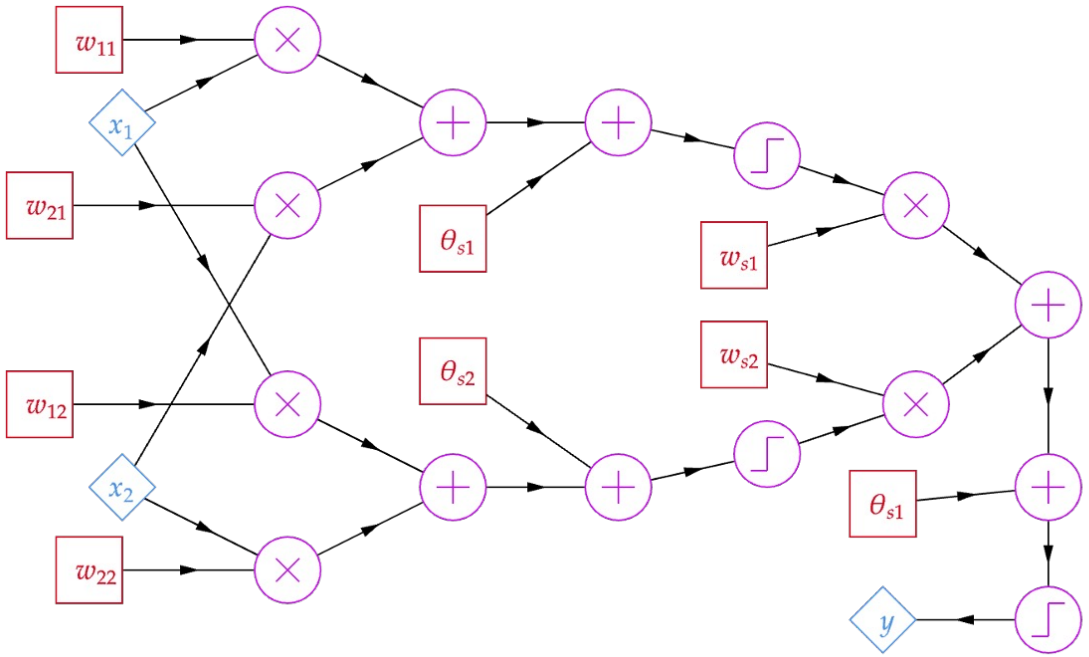
- 梯度下降 – 优化损失函数

$$E = \frac{1}{m}\sum_{i}^{m}(y_i - \hat{y}_i)^2$$

  - 如何求梯度？ $\frac{\partial E}{\partial w_k}$
    - 对于简单的感知机
      - 解析求导

$$\hat{y}_i = f(\vec{x}_i, \{w_k, \theta_k\}) = w_1 x_i^1 + w_2 x_i^2 - \theta_1$$

$$\frac{\partial E}{\partial w_1} = \frac{1}{m}\sum_{i}^{m} -2x_i^1(y_i - \hat{y}_i)$$



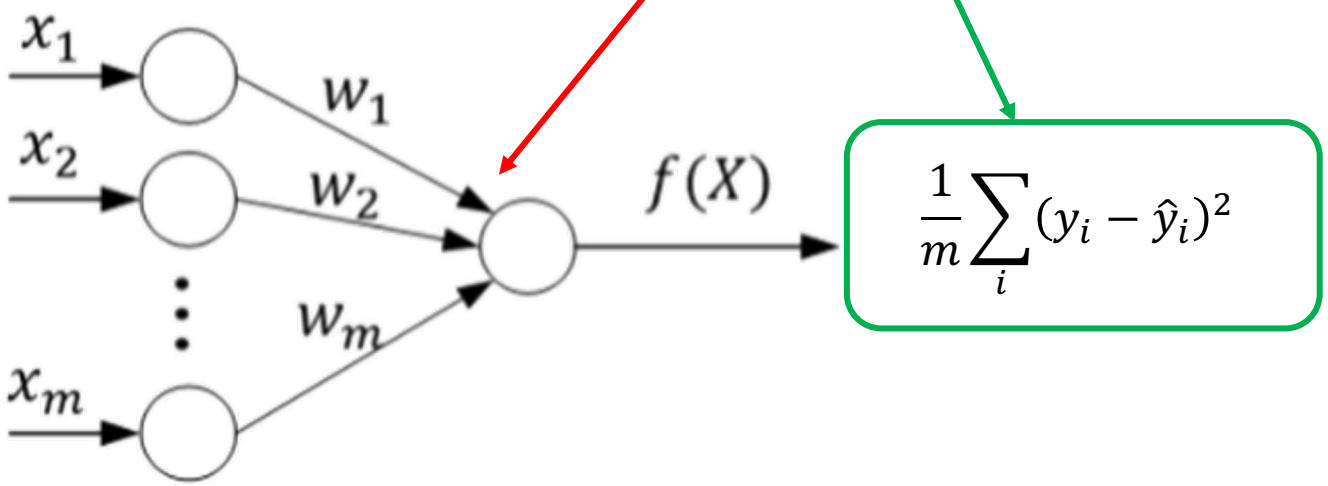
如何有效得到相应的梯度？

链式法则

- 链式法则

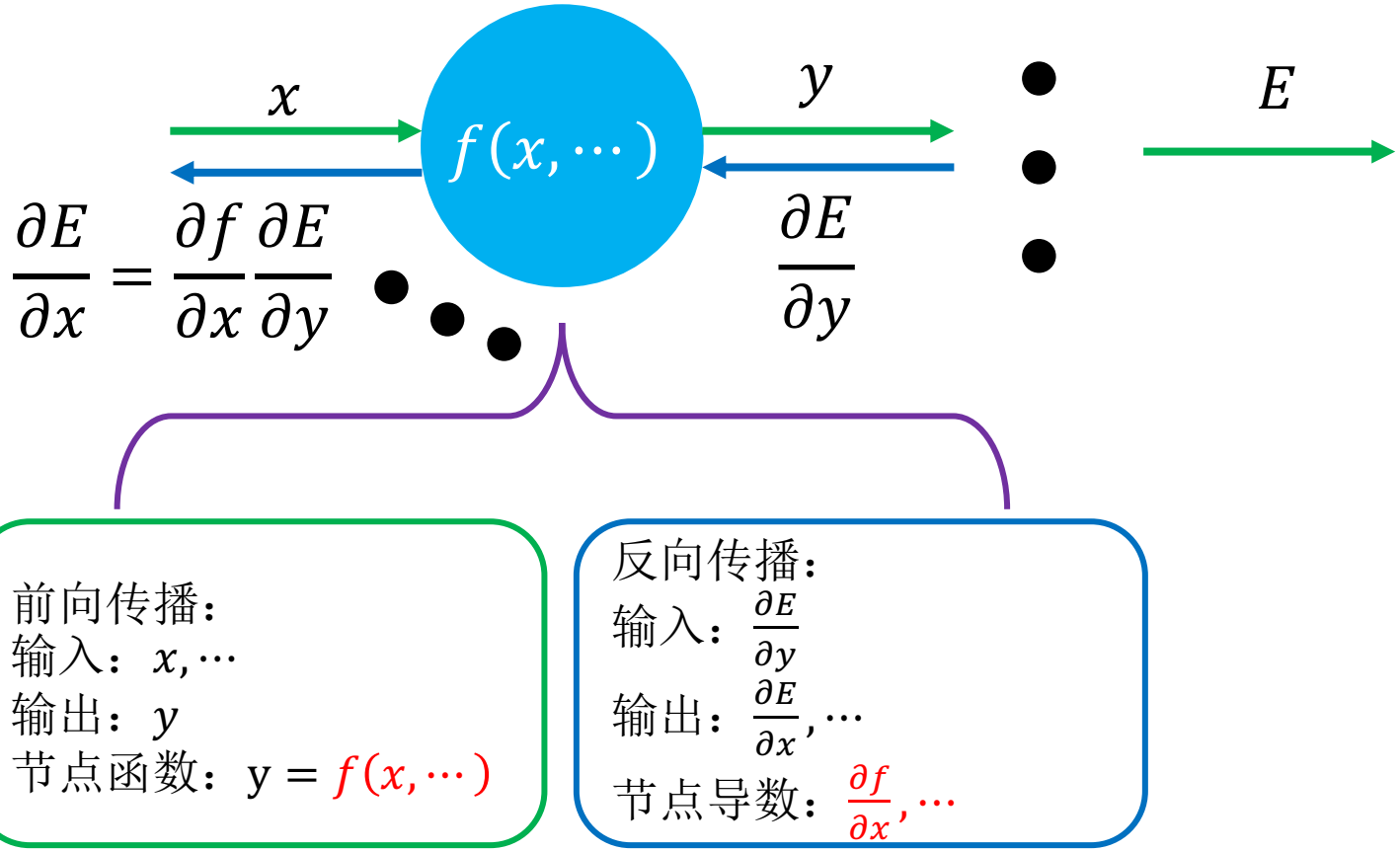
$$E = \frac{1}{m}\sum_{i}^{m}(y_i - \hat{y}_i)^2 \qquad \hat{y}_i = f(\vec{x}_i, \{w_k, \theta_k\}) = w_1 x_i^1 + w_2 x_i^2 - \theta_1$$

$$\frac{\partial E}{\partial w_1} = \sum_k \frac{\partial \hat{y}_k}{\partial w_1}\frac{\partial E}{\partial \hat{y}_k}$$



反向传播

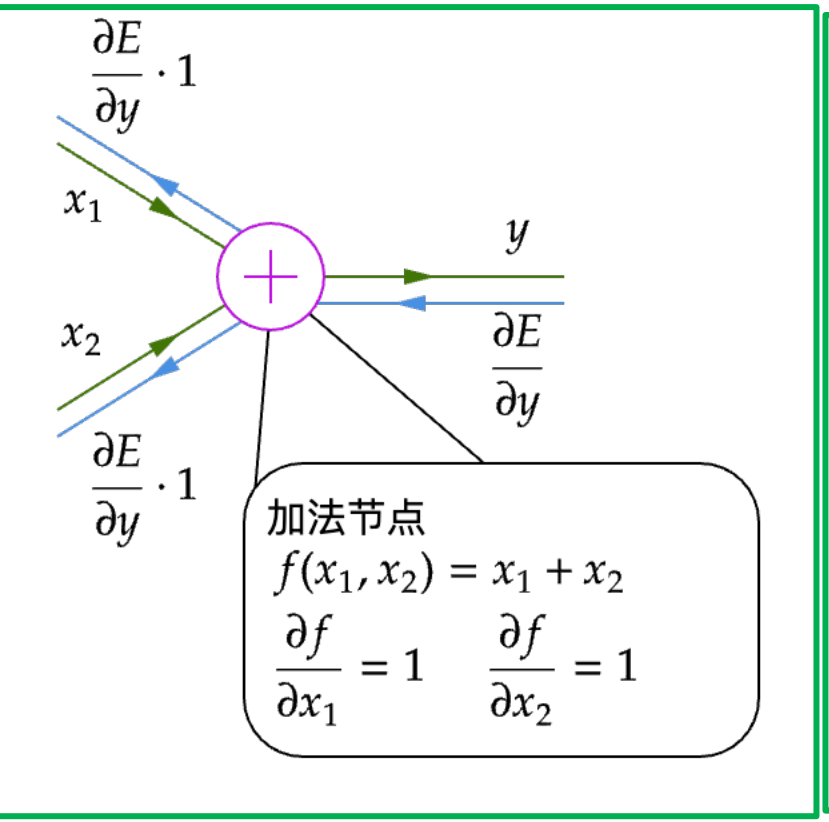
- 反向传播
  - 前向传播 – 局部计算
  - 梯度也可以局部计算，反向传递



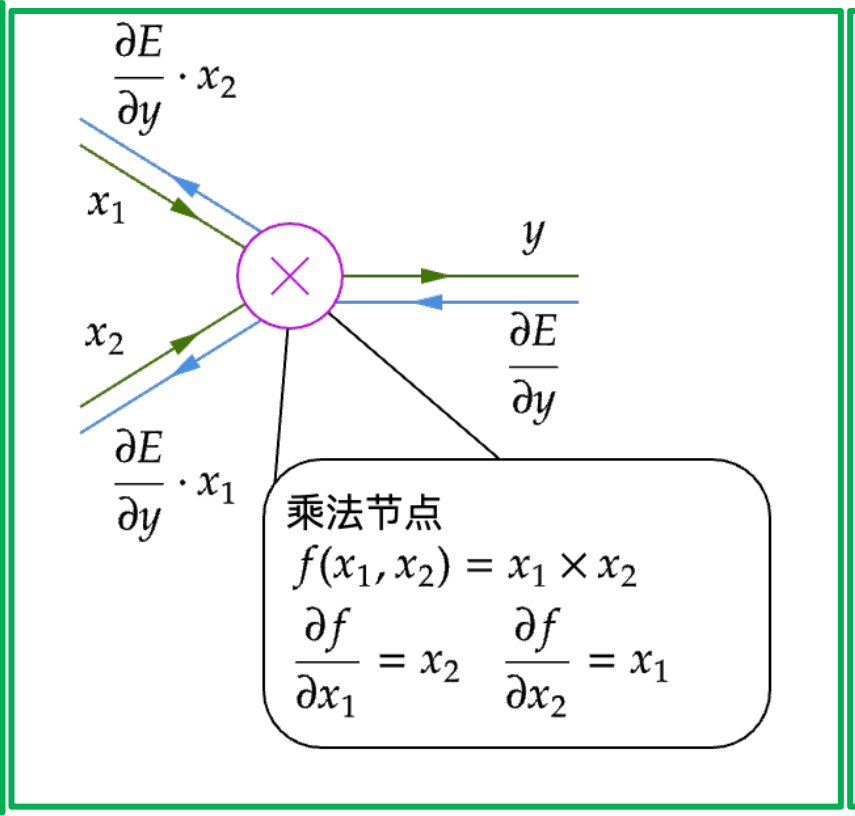
反向传播 - 计算节点

- 反向传播

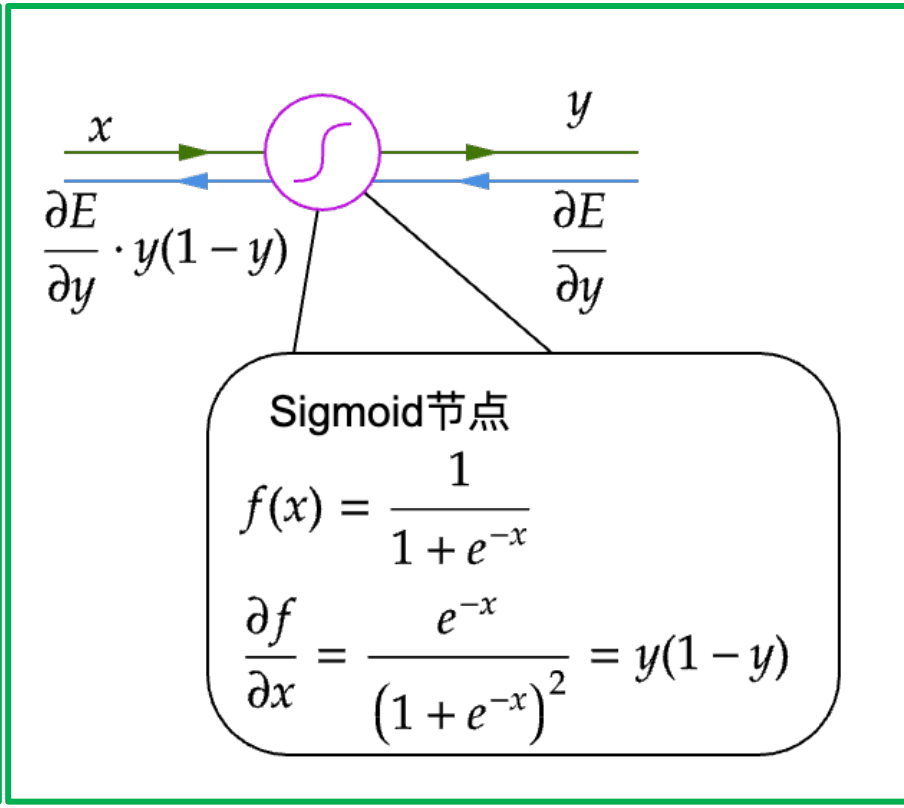
加法计算节点



乘法计算节点

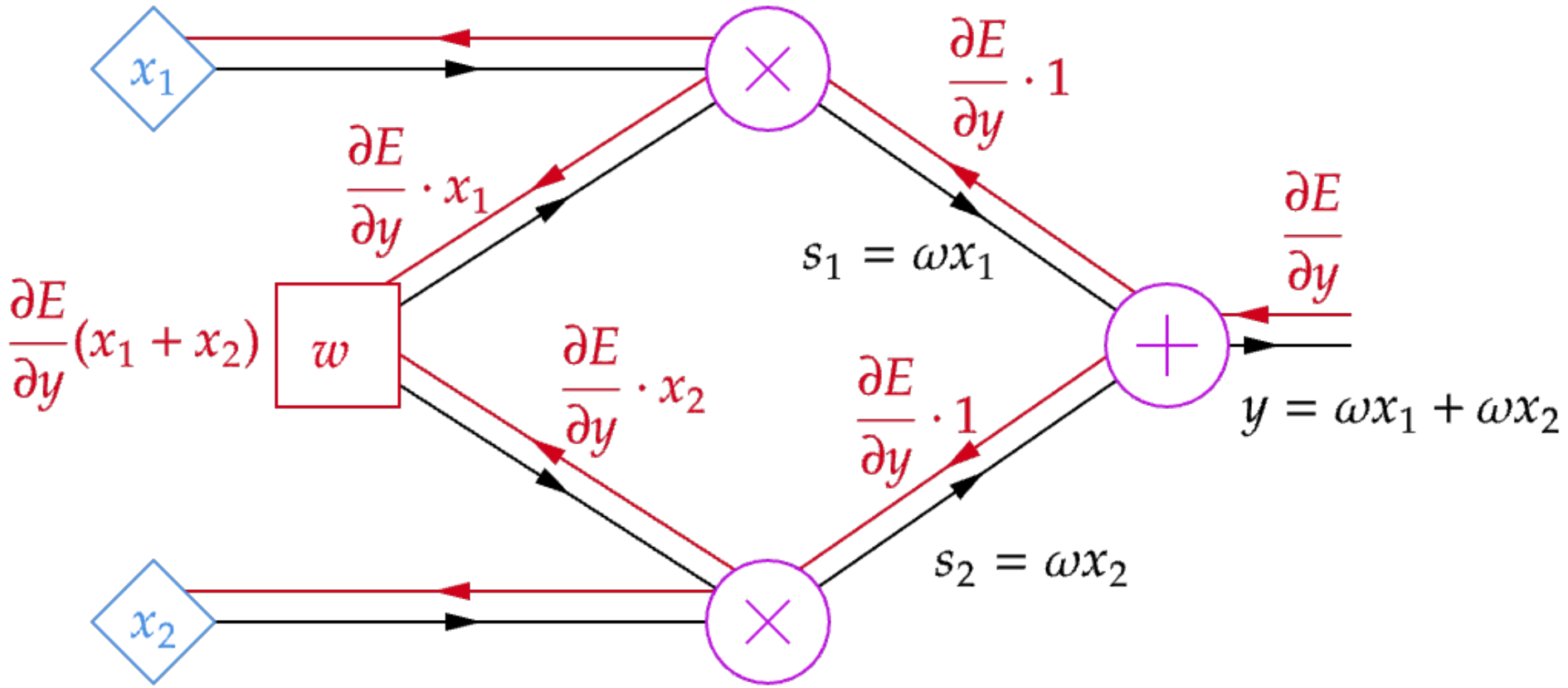


Sigmoid节点



反向传播

- 多重依赖

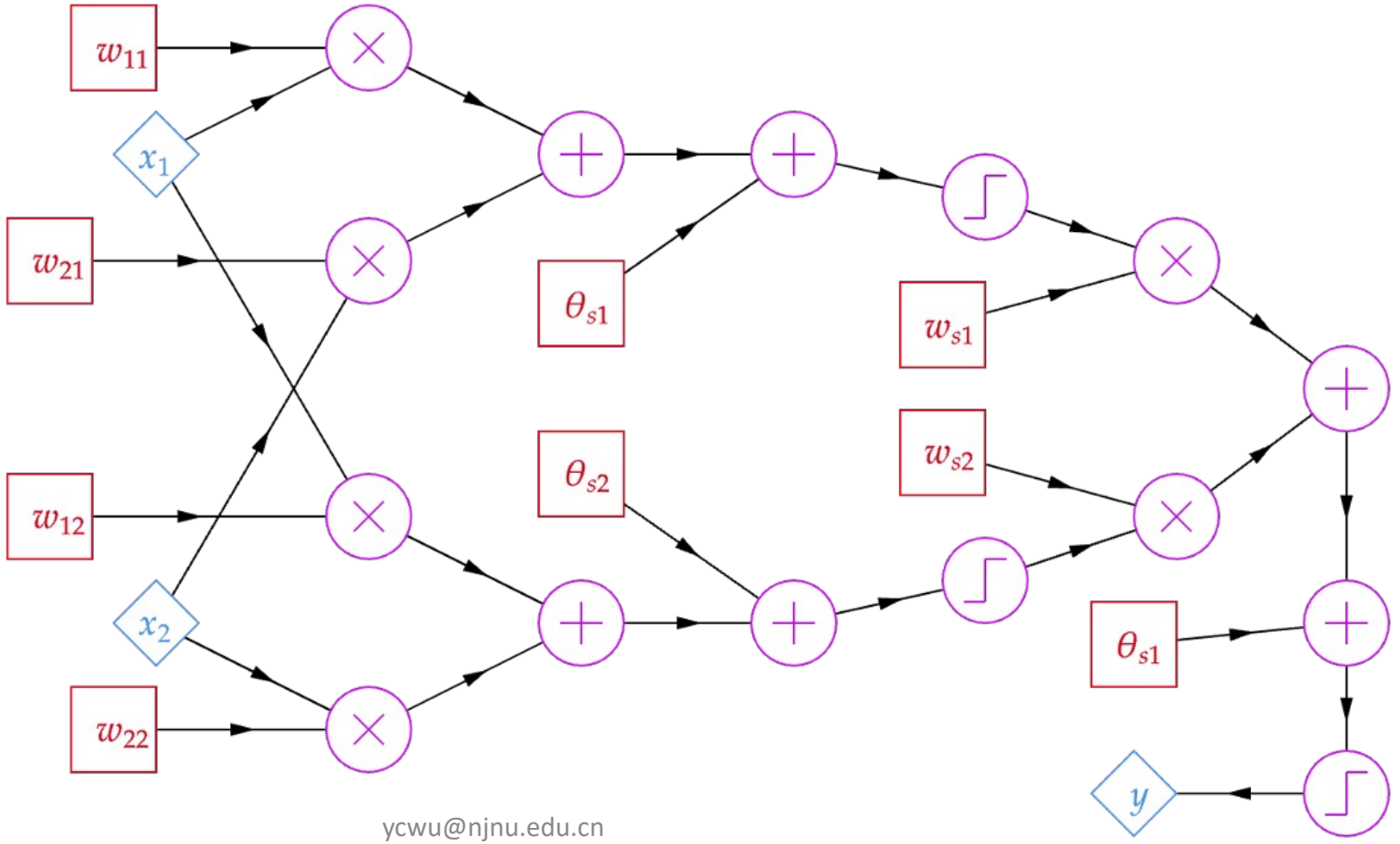

x_1
x_2
w
$\frac{\partial E}{\partial y}(x_1+x_2)$
$\frac{\partial E}{\partial y}\cdot x_1$
$\frac{\partial E}{\partial y}\cdot x_2$
$\frac{\partial E}{\partial y}\cdot 1$
$\frac{\partial E}{\partial y}\cdot 1$
$s_1=\omega x_1$
$s_2=\omega x_2$
$\frac{\partial E}{\partial y}$
$y=\omega x_1+\omega x_2$

神经网络

- 前向传播 – 计算模型预测
- 反向传播 – 根据预测与真实标签的差别，更新权重/偏置

$$\omega_i \leftarrow \omega_i + \Delta\omega_i$$

$$\Delta\omega_i = -\eta \frac{\partial E}{\partial \omega_i}$$



决策树

决策树

- 决策树
  - 结构化的分类过程
- 以quark/gluon-jet tagging为例为了简便离散化处理
  - 相关变量
    - n_{PF}：Jet内组分数目 - 多/略少/少
    - w_{PF}：Jet内组分p_T加权角分布 - 分散/稍集中/集中
    - p_T^D：Jet内组分p_T分布 - 宽/稍宽/窄
    - C_α：Jet内两点能量关联 - 强/次强/弱
      - $\alpha \sim 0.2$有最优的灵敏度
    - x_{hard}：Jet内能量最大的组分占Jet能量的比例 - 高/中/低
  - 加一个无关变量
    - η_j：Jet的快度 - 前向区/中央区

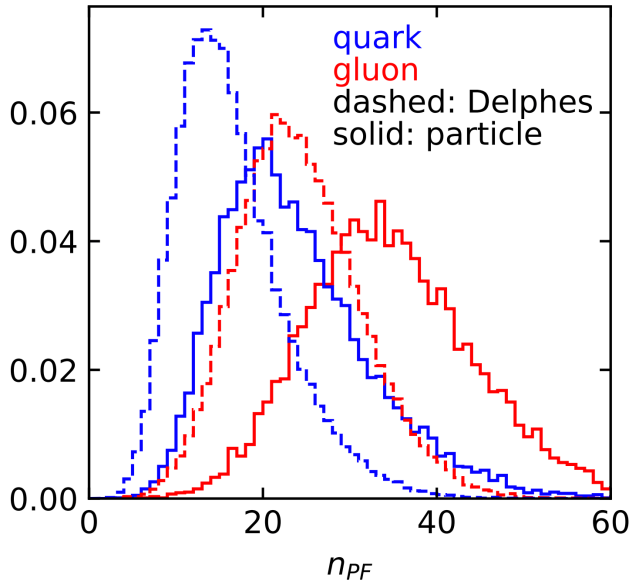
$$w_{PF} = \frac{\sum_i p_{T,i} \Delta R_{i,jet}}{\sum_i p_{T,i}}$$

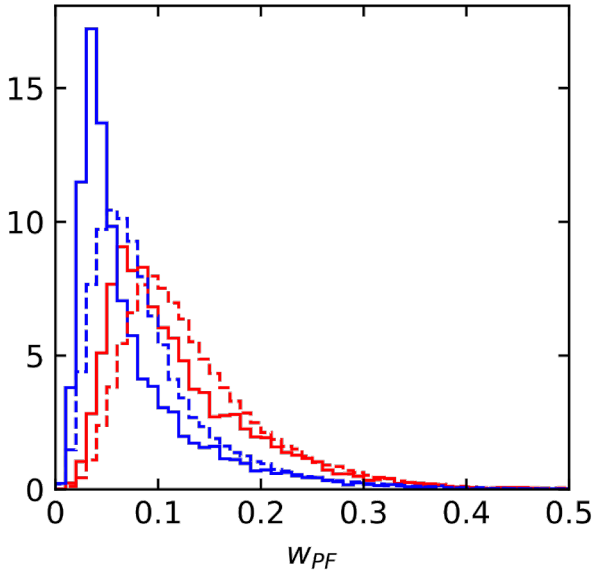
$$p_T^D = \frac{\sqrt{\sum_i p_{T,i}^2}}{\sum_i p_{T,i}}$$

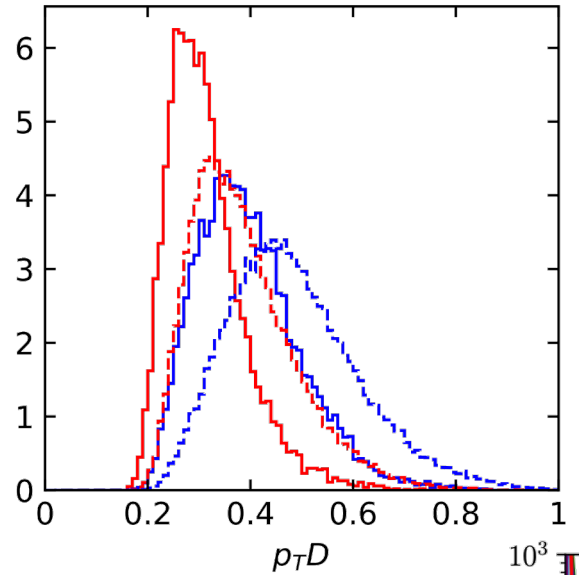
$$C_\alpha = \frac{\sum_{ij} E_{T,i} E_{T,j} (\Delta R_{ij})^\alpha}{\sum_i E_{T,i}^2}$$

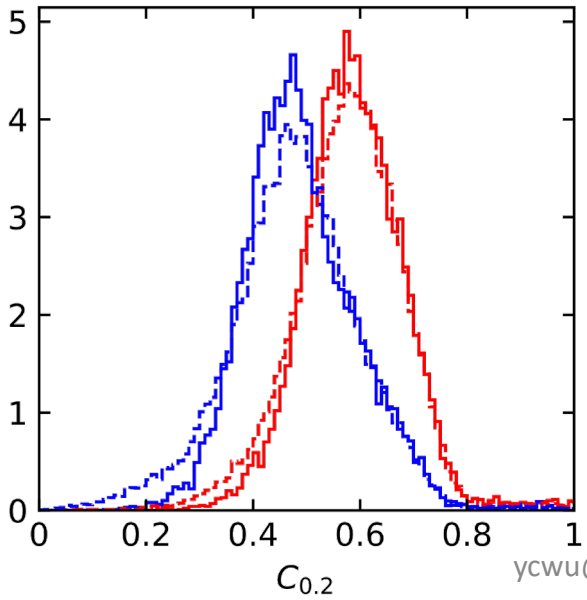
决策树 - quark/gluon tagging

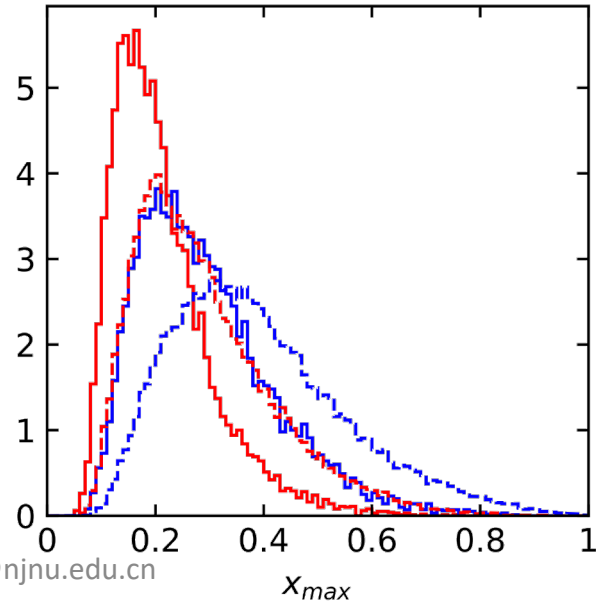
- 相关变量的分布：
  - 区别主要来源
    - 色因子
    - 碎裂函数

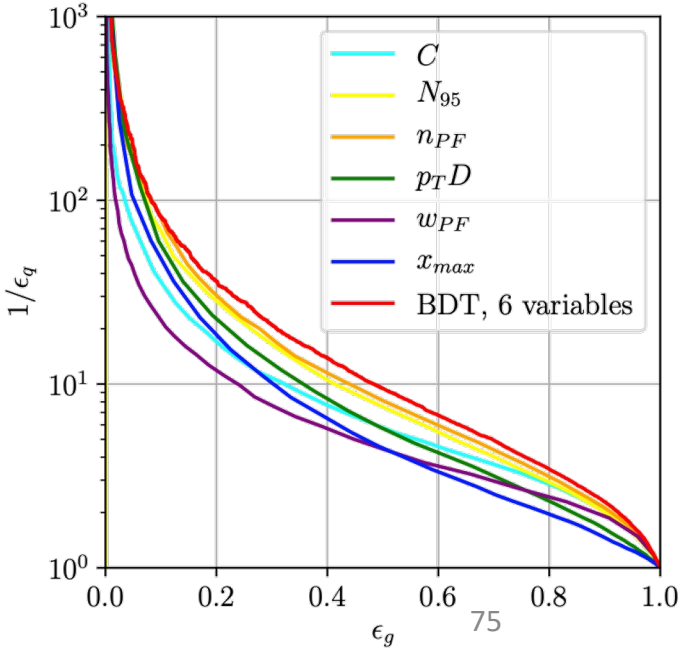










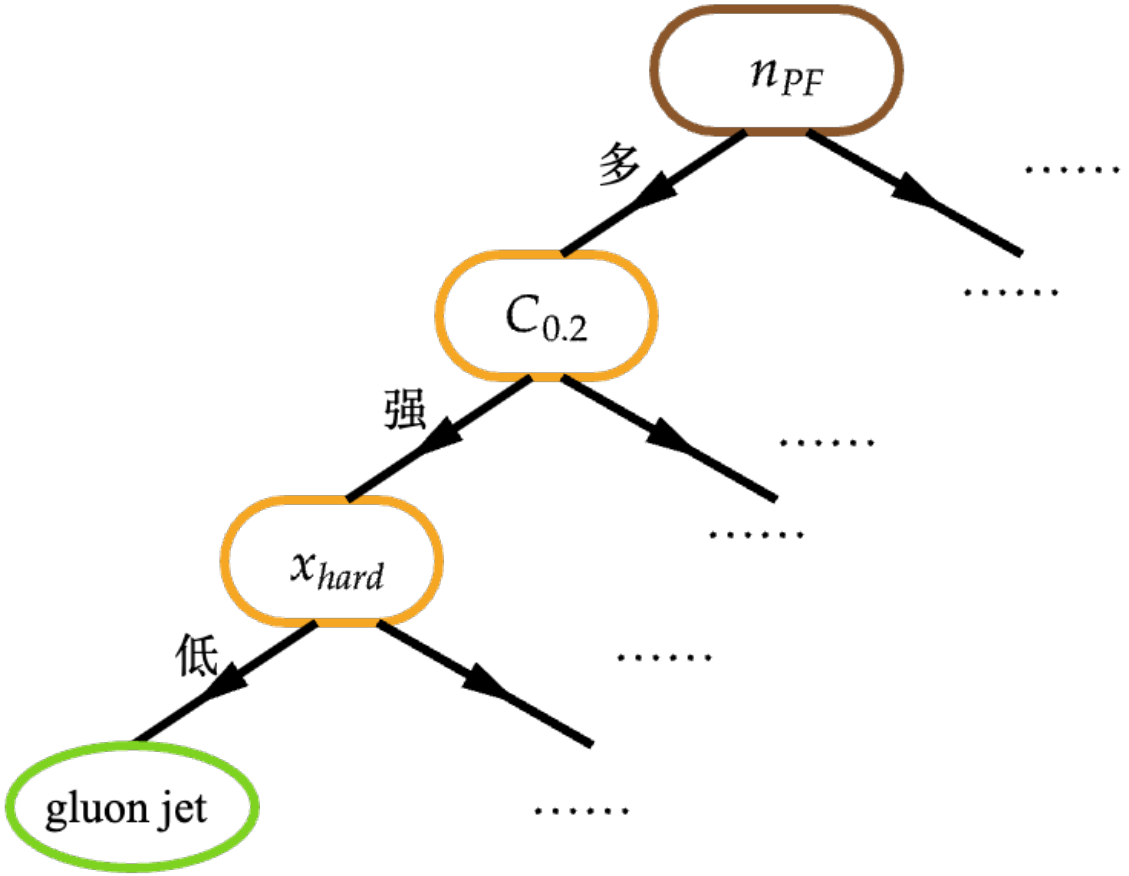


决策树 – 基本流程

- 决策树：
  - 根节点
  - 内部节点
  - 叶节点

  } 特征测试

  → 决策结果
- 决策树的产生
  - 递归产生
  - 如何产生分支？ – 特征测试
    - 选择最优划分特征
  - 如何终结分支？ – 决策
    - 终结分支情形：
      - 一个节点中的样本属于同一类别
      - 一个节点中的样本在所有特征上相同
        - 特例：一个节点中的样本特征已用完
      - 一个节点中无任何训练样本



决策树 - quark/gluon tagging数据集

编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
1	稍集中	适中	低	多	次强	中央	gluon
2	分散	偏小	中	多	强	前向	gluon
3	稍集中	偏小	中	多	强	前向	gluon
4	分散	适中	低	略少	次强	中央	gluon
5	分散	适中	低	多	次强	前向	gluon
6	分散	偏小	低	多	强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
8	稍集中	偏小	低	多	强	前向	gluon
9	集中	偏小	低	少	弱	前向	quark
10	稍集中	偏小	中	略少	次强	前向	quark
11	稍集中	偏大	高	多	弱	中央	quark
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
14	集中	适中	中	略少	强	前向	quark
15	分散	适中	低	多	次强	中央	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

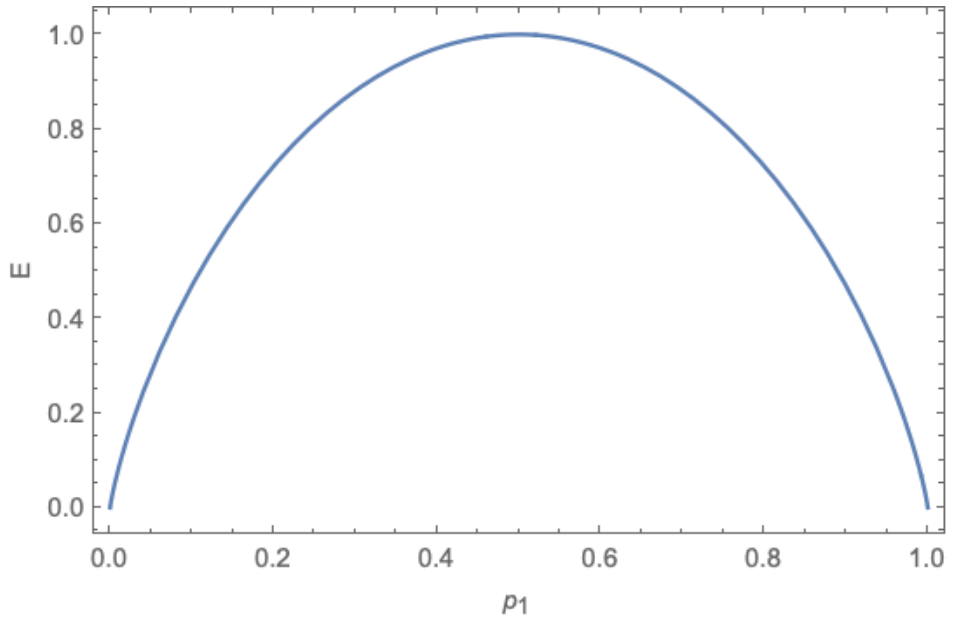
决策树 – 划分选择

- 如何选择特征进行分支？
  - 提升分类效果 – 错误率/查准率……
- 信息熵
  - 度量样本纯度的指标
  - 假设当前样本集合D中，按标签可划分为$|\mathcal{Y}|$类
    - 第k类样本所占的比例为$p_k, k = 1, \cdots, |\mathcal{Y}|$

$$\mathcal{E}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k$$

  - 信息熵越小，样本越纯
    - 假设数据集一共两个标签；$p_1 + p_2 = 1$

$$\begin{aligned}\mathcal{E} &= -p_1 \log_2 p_1 - p_2 \log_2 p_2 \\ &= -p_1 \log_2 p_1 - (1 - p_1) \log_2 (1 - p_1)\end{aligned}$$



决策树 – 信息增益

- 划分选择
  - 信息熵在划分后有下降 – 提纯
    - 信息增益
- 信息增益 – ID3算法中使用的划分方法
  - 假设数据集D的样本含有某个特征a
    - 特征a有V个离散取值$\{a_1, \cdots, a_V\}$
  - 根据特征a对数据集进行划分
    - 分成V个子集 – V个子分支
      - $\{D^1, \cdots, D^V\}$

$$\Delta\mathcal{E}(D,a) = \mathcal{E}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathcal{E}(D^v)$$

决策树 - 信息增益

- 以quark/gluon jet tagging为例
- 原始数据的信息熵：
  - 一共17个样本，8个为gluon jet，9个为quark jet

$$\mathcal{E}(D) = -\frac{8}{17}\log_2\frac{8}{17} - \frac{9}{17}\log_2\frac{9}{17} \approx 0.9975$$

  - 按特征w_{PF}划分 - {集中，稍集中，分散}
    - 集中：{7, 9, 12, 14, 17}

$$\mathcal{E}(D^1) = -\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5} \approx 0.7219$$

    - 稍集中：{1, 3, 8, 10, 11, 16}

$$\mathcal{E}(D^2) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1$$

    - 分散：{2, 4, 5, 6, 13, 15}

$$\mathcal{E}(D^3) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} \approx 0.9183$$

$$\Delta\mathcal{E}(D, w_{PF}) = \mathcal{E}(D) - \sum_{v=1}^{3}\frac{|D^v|}{|D|}\mathcal{E}(D^v)$$

$$\approx 0.9975 - \frac{5}{17}\times 0.7219 - \frac{6}{17}\times 1 - \frac{6}{17}\times 0.9183$$

$$\approx 0.1081$$

决策树 – 信息增益

- 按其他特征划分
  - p_T^D
    - 偏大：{11, 12}
    - 适中：{1, 4, 5, 13, 14, 15, 16}
    - 偏小：{2, 3, 6, 7, 8, 9, 10, 17}
  - x_{hard}
    - 高：{11, 12}
    - 中：{2, 3, 10, 13, 14}
    - 低：{1, 4, 5, 6, 7, 8, 9, 15, 16, 17}
  - n_{PF}
    - 多：{1, 2, 3, 5, 6, 7, 8, 11, 15}
    - 略少：{4, 10, 13, 14, 16}
    - 少：{9, 12, 17}
  - C_α
    - 强：{2, 3, 6, 7, 8, 14, 16}
    - 次强：{1, 4, 5, 10, 13, 15}
    - 弱：{9, 11, 12, 17}
  - η_j
    - 前向：{2, 3, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16}
    - 中央：{1, 4, 11, 15, 17}

$$\Delta\mathcal{E}(D, p_T^D) = 0.9975 - \frac{2}{17}\times 0 - \frac{7}{17}\times 0.9852 - \frac{8}{17}\times 0.9544 \approx 0.1427$$

$$\Delta\mathcal{E}(D, x_{hard}) = 0.9975 - \frac{2}{17}\times 0 - \frac{5}{17}\times 0.9710 - \frac{10}{17}\times 0.9710 \approx 0.1408$$

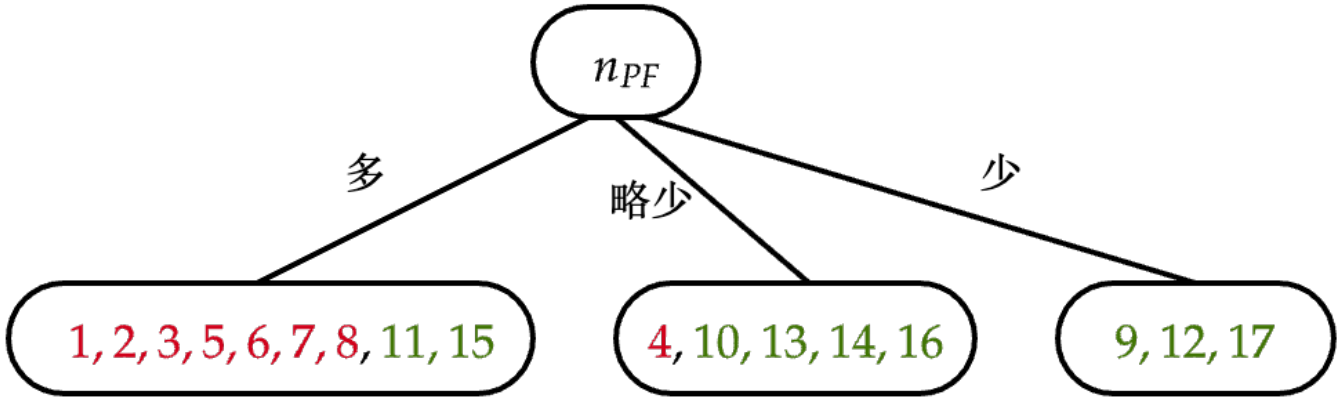
$$\Delta\mathcal{E}(D, n_{PF}) = 0.9975 - \frac{9}{17}\times 0.7642 - \frac{5}{17}\times 0.7219 - \frac{3}{17}\times 0 \approx 0.3806$$

$$\Delta\mathcal{E}(D, C_\alpha) = 0.9975 - \frac{7}{17}\times 0.8631 - \frac{6}{17}\times 1 - \frac{4}{17}\times 0 \approx 0.2892$$

$$\Delta\mathcal{E}(D, \eta_j) = 0.9975 - \frac{12}{17}\times 1 - \frac{5}{17}\times 0.9710 \approx 0.0060$$

决策树 – 信息增益

- 基于n_{PF}的分支

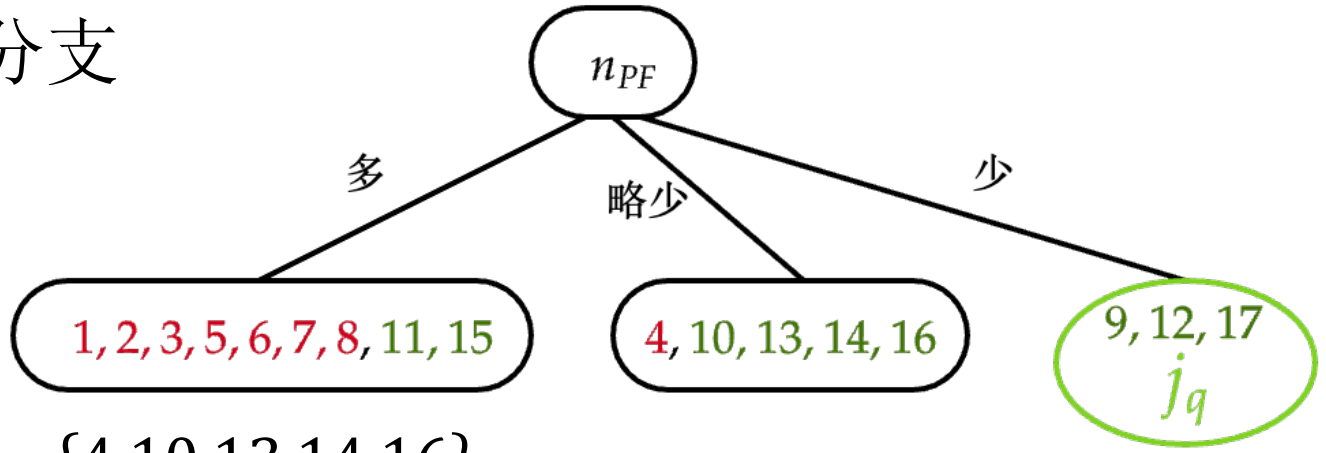


- 每个节点继续做特征划分（不再使用n_{PF}）
  - 递归进行
  - 判断是否终结

节点中样本属于同一类

决策树 – 递归生成

- 递归产生分支



- 对于$D^2 = \{4,10,13,14,16\}$
  - $\mathcal{E}(D^2) = -\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5} \approx 0.7219$

$$\Delta\mathcal{E}(D^2, w_{PF}) = 0.7219 - \frac{2}{5}\times 1 - \frac{2}{5}\times 0 - \frac{1}{5}\times 0 = 0.3219$$

$$\Delta\mathcal{E}(D^2, p_T^D) = 0.7219 - \frac{4}{5}\times 0.8113 - \frac{1}{5}\times 0 = 0.0729$$

$$\Delta\mathcal{E}(D^2, x_{hard}) = 0.7219 - \frac{2}{5}\times 1 - \frac{3}{5}\times 0 = 0.3219$$

$$\Delta\mathcal{E}(D^2, C_\alpha) = 0.7219 - \frac{3}{5}\times 0.9183 - \frac{2}{5}\times 0 = 0.1709$$

$$\Delta\mathcal{E}(D^2, \eta_j) = 0.7219 - \frac{1}{5}\times 0 - \frac{4}{5}\times 0 = 0.7219$$

编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
4	分散	适中	低	略少	次强	中央	gluon
10	稍集中	偏小	中	略少	次强	前向	quark
13	分散	适中	中	略少	次强	前向	quark
14	集中	适中	中	略少	强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark

决策树 – 递归生成

- 递归产生分支 $\mathcal{E}(D^1) = -\frac{7}{9}\log_2\frac{7}{9} - \frac{2}{9}\log_2\frac{2}{9} = 0.7642$

p_T^D, C_α, η_j任挑一个特征产生分支

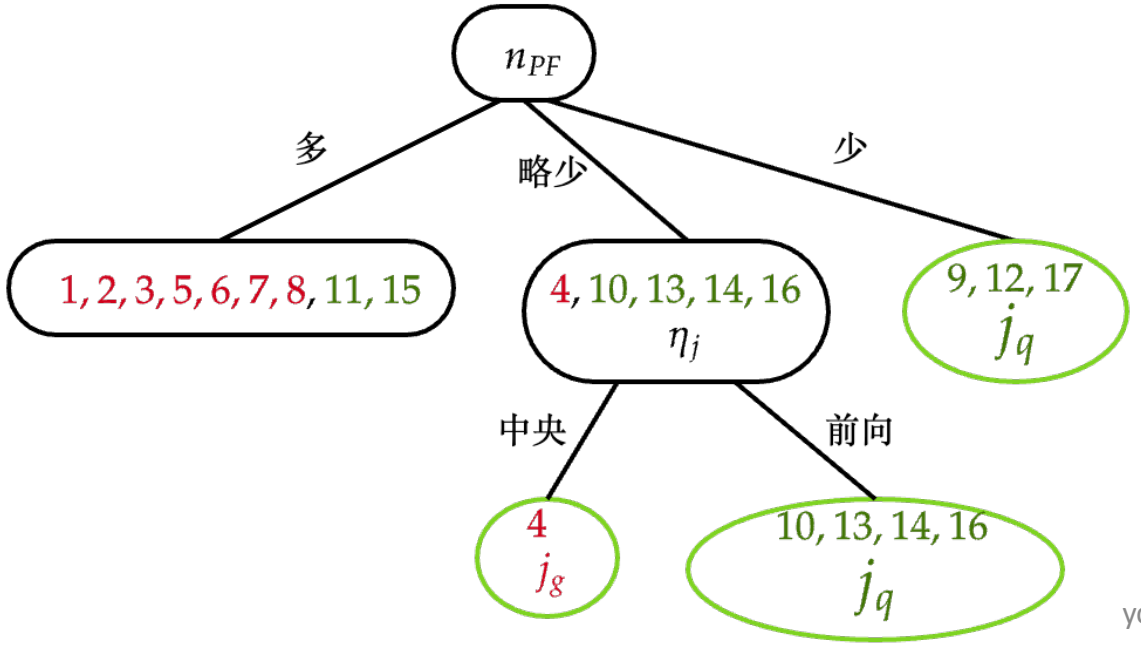
$$\Delta\mathcal{E}(D^1, w_{PF}) = 0.7642 - \frac{4}{9}\times 0.8113 - \frac{4}{9}\times 0.8113 - \frac{1}{9}\times 0 = 0.0430$$

$$\Delta\mathcal{E}(D^1, p_T^D) = 0.7642 - \frac{5}{9}\times 0 - \frac{3}{9}\times 0.9183 - \frac{1}{9}\times 0 = 0.4581$$

$$\Delta\mathcal{E}(D^1, x_{hard}) = 0.7642 - \frac{6}{9}\times 0.6500 - \frac{2}{9}\times 0 - \frac{1}{9}\times 0 = 0.3309$$

$$\Delta\mathcal{E}(D^1, C_\alpha) = 0.7642 - \frac{1}{9}\times 0 - \frac{3}{9}\times 0.9183 - \frac{5}{9}\times 0 = 0.4581$$

$$\Delta\mathcal{E}(D^1, \eta_j) = 0.7642 - \frac{3}{9}\times 0.9183 - \frac{6}{9}\times 0 = 0.4581$$



编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
1	稍集中	适中	低	多	次强	中央	gluon
2	分散	偏小	中	多	强	前向	gluon
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
6	分散	偏小	低	多	强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
8	稍集中	偏小	低	多	强	前向	gluon
11	稍集中	偏大	高	多	弱	中央	quark
15	分散	适中	低	多	次强	中央	quark

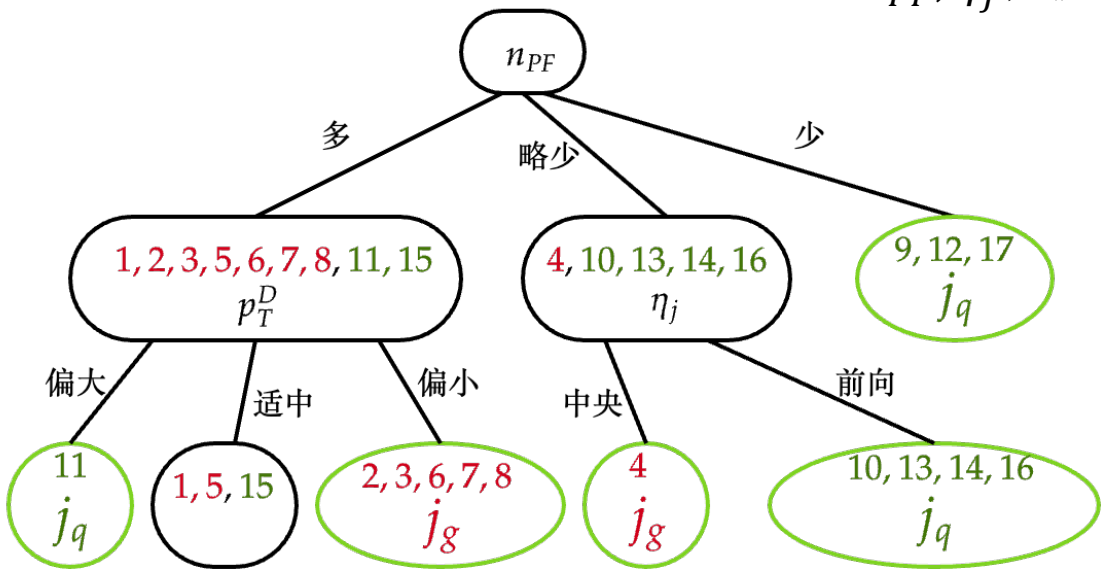
决策树 – 递归生成

- 递归产生分支 $\mathcal{E}(D^{12}) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.9183$

$$\Delta\mathcal{E}(D^{12}, w_{PF}) = 0.9183 - \frac{1}{3}\times 0 - \frac{2}{3}\times 1 - \frac{0}{3}\times 0 = 0.2516$$

$$\Delta\mathcal{E}(D^{12}, \eta_j) = 0.9183 - \frac{2}{3}\times 1 - \frac{1}{3}\times 0 = 0.2516$$

w_{PF}, η_j任挑一个特征产生分支



编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
1	稍集中	适中	低	多	次强	中央	gluon
5	分散	适中	低	多	次强	前向	gluon
15	分散	适中	低	多	次强	中央	quark

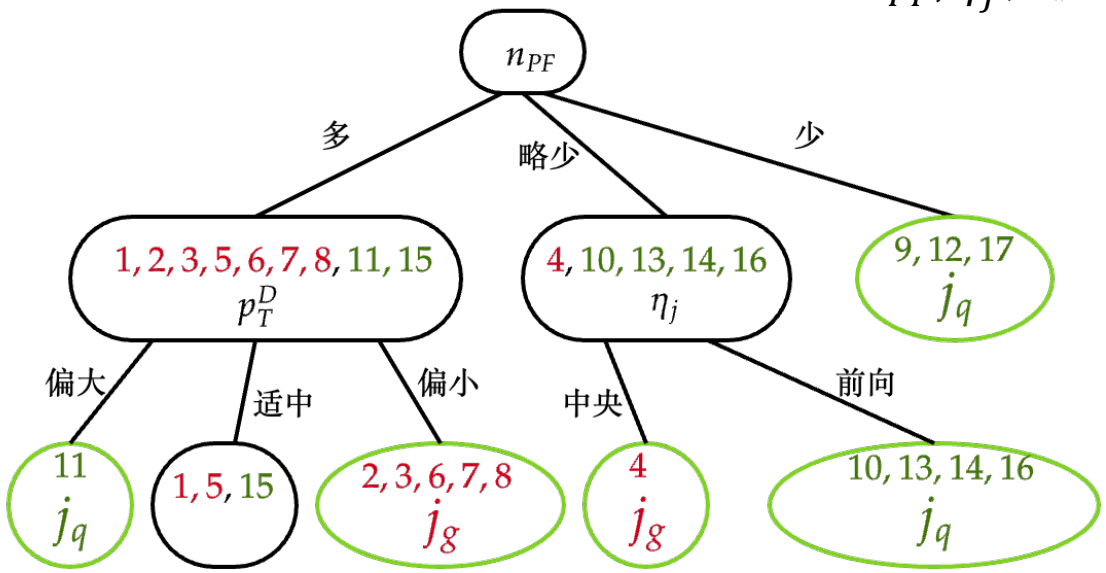
决策树 – 递归生成

- 递归产生分支 $\mathcal{E}(D^{12}) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.9183$

$$\Delta\mathcal{E}(D^{12}, w_{PF}) = 0.9183 - \frac{1}{3}\times 0 - \frac{2}{3}\times 1 - \frac{0}{3}\times 0 = 0.2516$$

$$\Delta\mathcal{E}(D^{12}, \eta_j) = 0.9183 - \frac{2}{3}\times 1 - \frac{1}{3}\times 0 = 0.2516$$

w_{PF}, η_j任挑一个特征产生分支



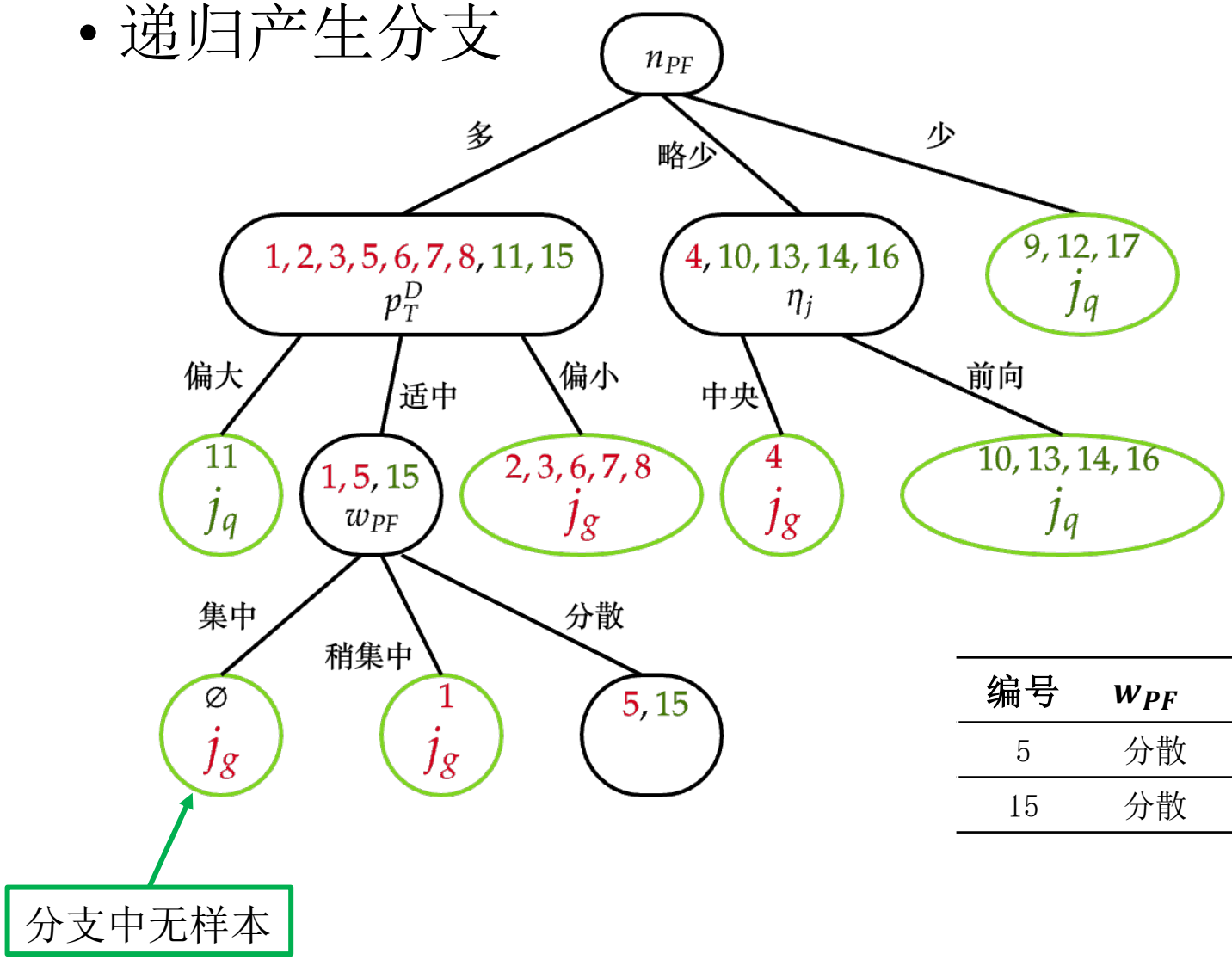
如果不存在w_{PF}, η_j

在所有特征上取值相同
分支终结

编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
1	稍集中	适中	低	多	次强	中央	gluon
5	分散	适中	低	多	次强	前向	gluon
15	分散	适中	低	多	次强	中央	quark

决策树 – 递归生成

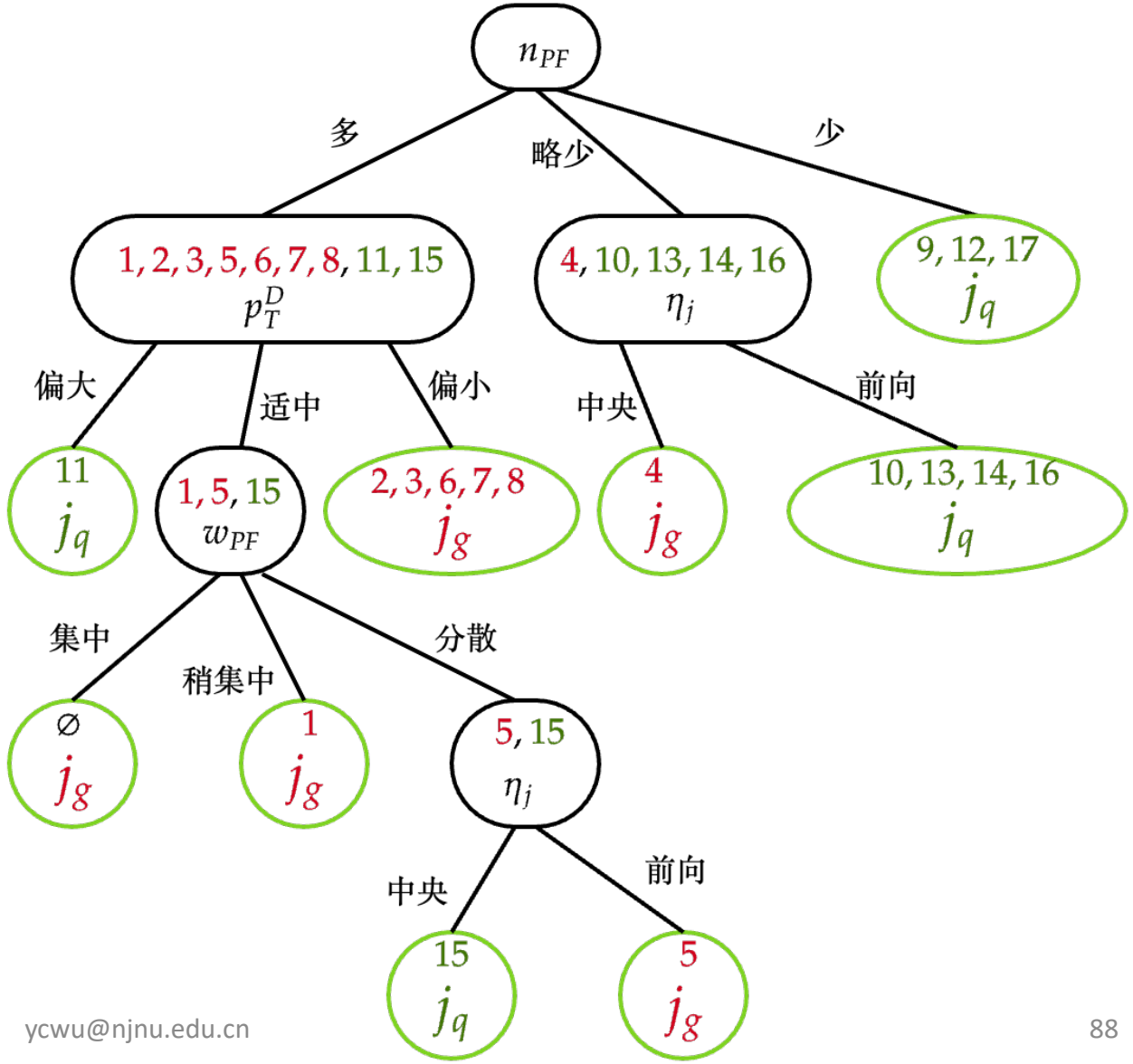
- 递归产生分支



编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
5	分散	适中	低	多	次强	前向	gluon
15	分散	适中	低	多	次强	中央	quark

决策树 – 递归生成

- 完整的决策树
  - 基于信息增益



决策树 - 递归生成

- 其他节点划分方法
  - 增益率：C4.5算法中采用
    - 信息增益更倾向可取值更多的特征
      - 极端情况：每个样本一个取值
    - 用特征的固有信息熵来归一化

$$\mathcal{R}(D,a)=\frac{\Delta\mathcal{E}(D,a)}{N(D,a)},\qquad N(D,a)=-\sum_{v=1}^{V}\frac{|D^v|}{|D|}\log_2\frac{|D^v|}{|D|}$$

    - 增益率倾向可取值较少的特征
    - C4.5算法不直接使用增益率来选择最优特征来进行划分
      - 先找到信息增益高于平均值的特征
      - 然后找到其中增益率最高的

决策树 – 递归生成

- 其他节点划分方法
  - 基尼指数：CART算法中采用
    - 一个数据集的基尼值
      - 随机选两个样本属于不同标签的概率，值越小数据集越纯

$$G(D)=\sum_{k=1}^{|y|}\sum_{k'\neq k}p_k p_{k'}=1-\sum_{k=1}^{|y|}p_k^2$$

    - 基尼指数

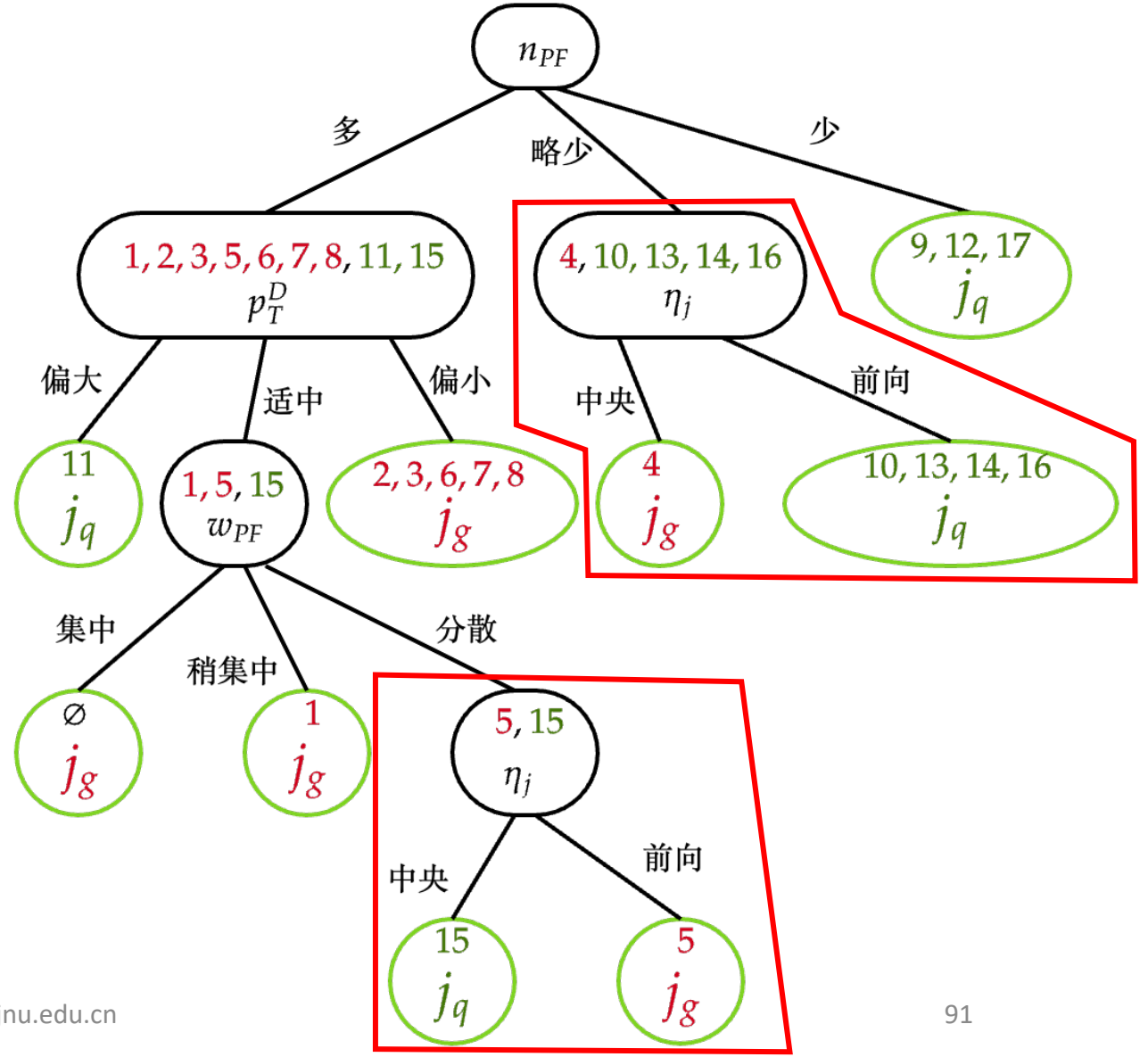
$$I_g(D,a)=\sum_{v=1}^{V}\frac{|D^v|}{|D|}G(D^v)$$

    - 划分选择 – 使得基尼指数最小的特征

$$a^*=\arg\min_a I_g(D,a)$$

决策树 - 剪枝

- 过拟合
  - 剪枝 - 控制节点展开
- 剪枝
  - 预剪枝
    - 在节点展开前
    - 若不能提升泛化性能，停止展开
  - 后剪枝
    - 在决策树完整生成之后
    - 反溯分支节点 - 泛化性能
- 性能评估
  - 训练集 vs. 验证集



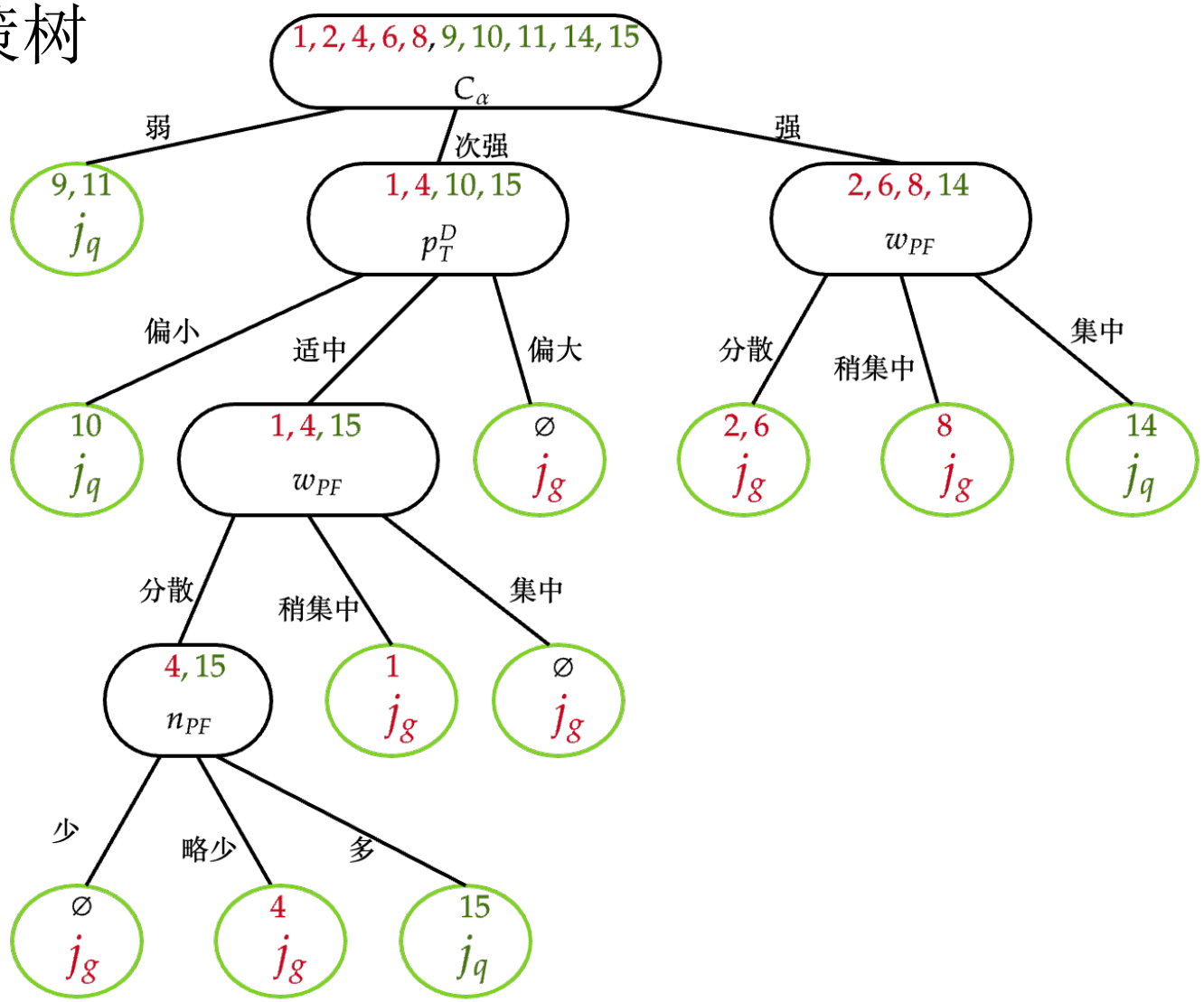
决策树 – 剪枝

- 训练集
- 验证集

编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
1	稍集中	适中	低	多	次强	中央	gluon
2	分散	偏小	中	多	强	前向	gluon
4	分散	适中	低	略少	次强	中央	gluon
6	分散	偏小	低	多	强	前向	gluon
8	稍集中	偏小	低	多	强	前向	gluon
9	集中	偏小	低	少	弱	前向	quark
10	稍集中	偏小	中	略少	次强	前向	quark
11	稍集中	偏大	高	多	弱	中央	quark
14	集中	适中	中	略少	强	前向	quark
15	分散	适中	低	多	次强	中央	quark
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 – 剪枝

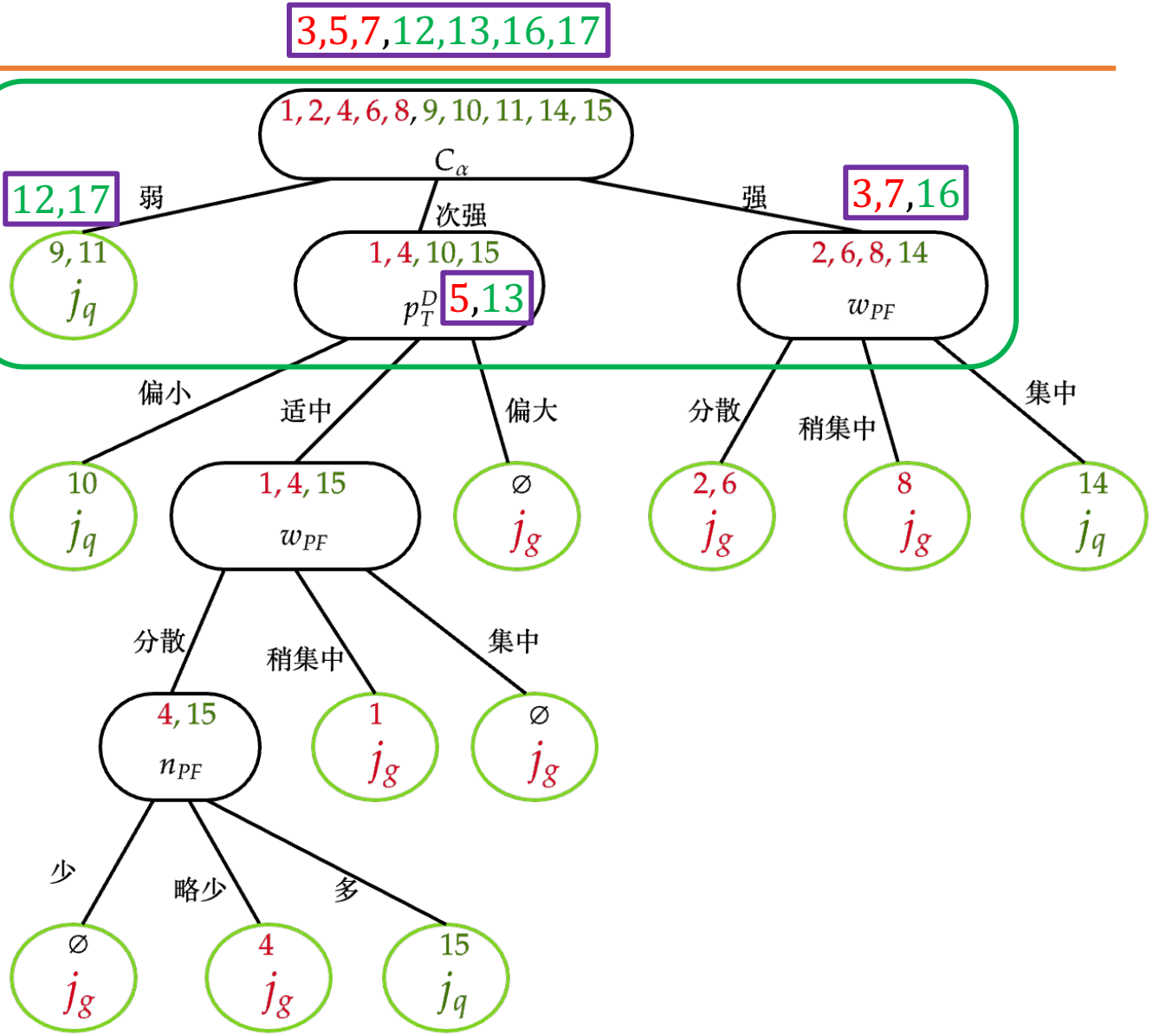
- 训练集生成决策树



决策树 - 预剪枝

- 验证集剪枝
  - 预剪枝

	验证集精度
划分前	4/7=57.1% (3/7=42.9%)
划分后	5/7=71.4%

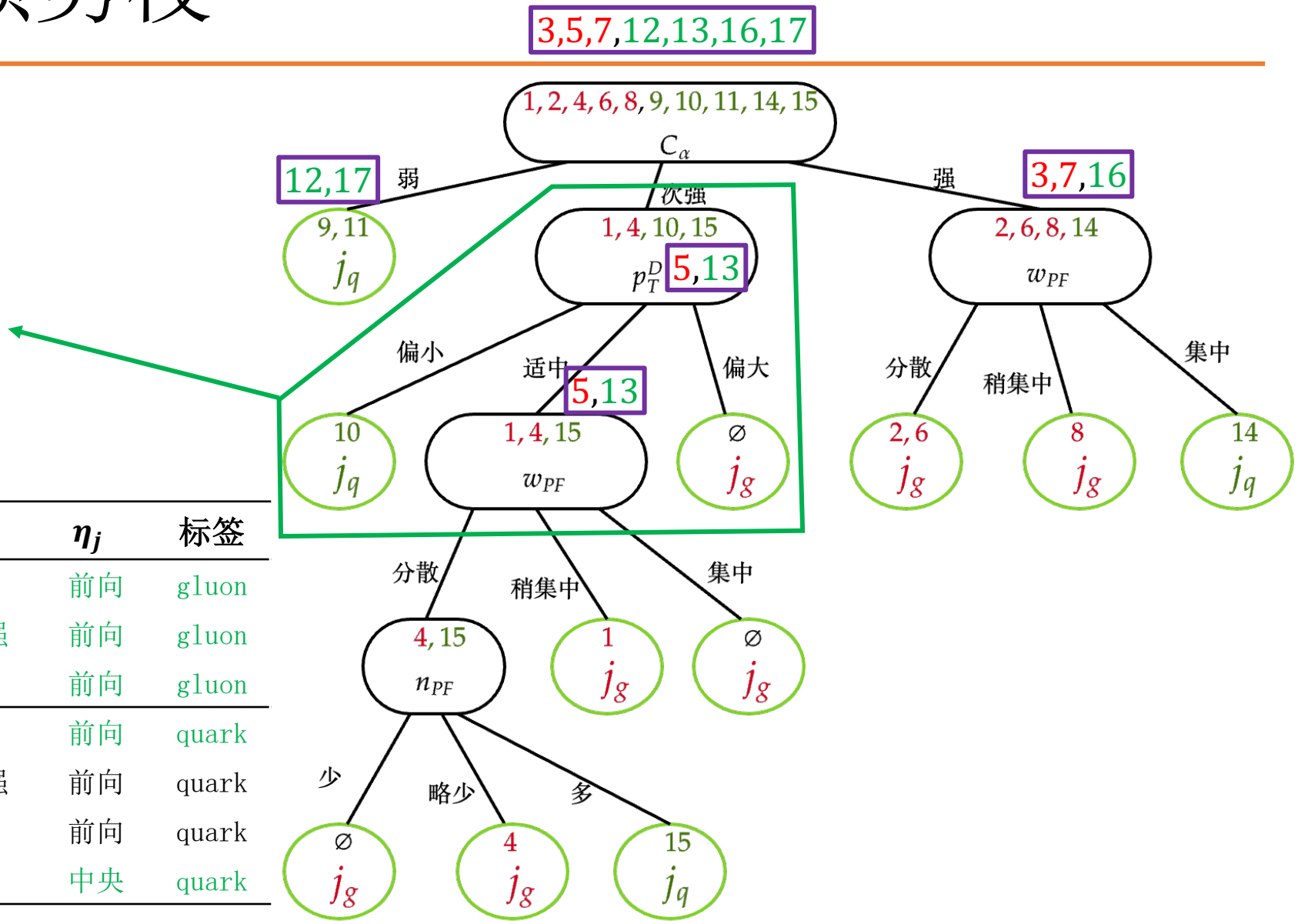


编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 - 预剪枝

- 验证集剪枝
  - 预剪枝

	验证集精度
划分前	5/7=71.4%
划分后	5/7=71.4%

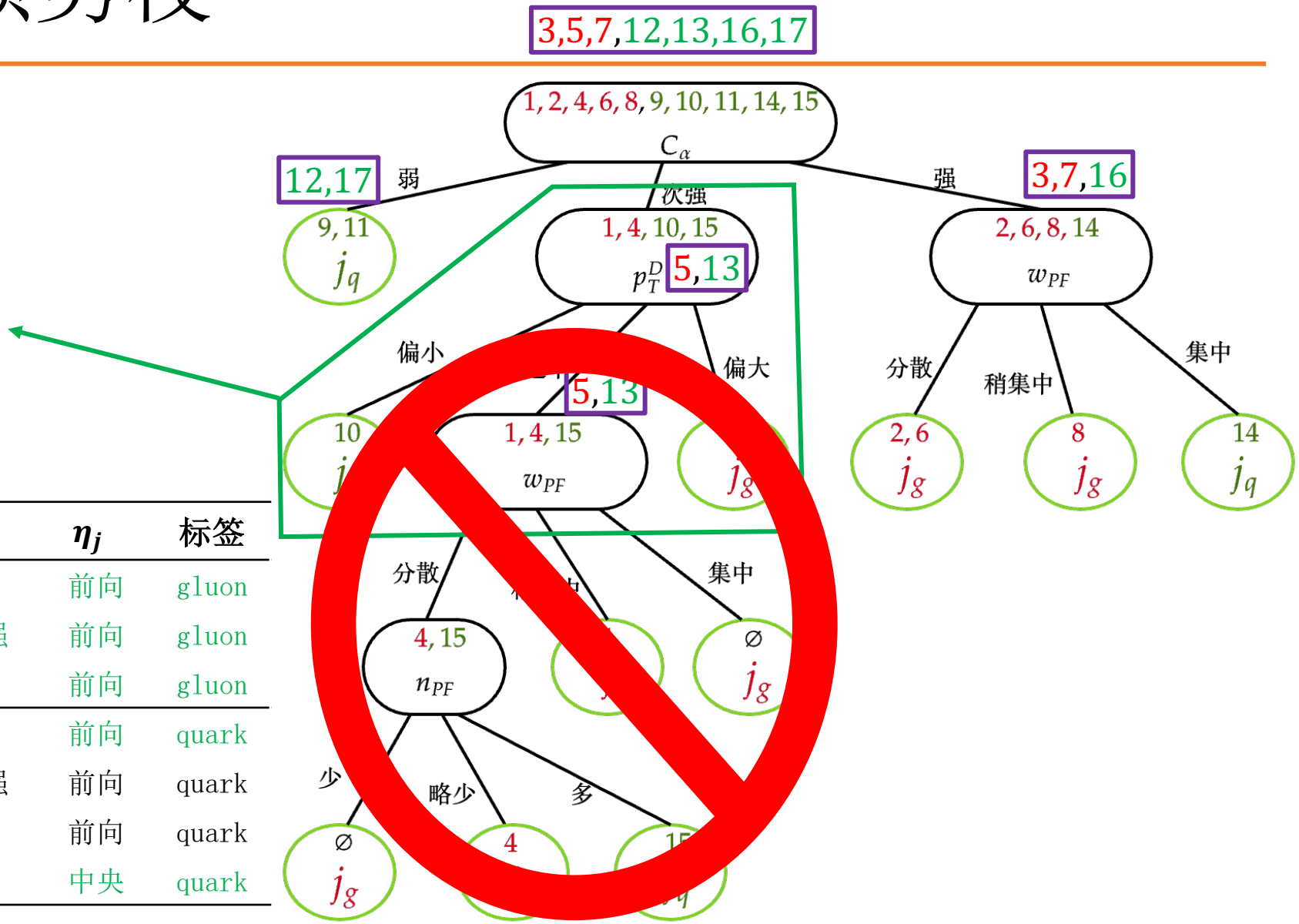


编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 – 预剪枝

- 验证集剪枝
  - 预剪枝

	验证集精度
划分前	5/7=71.4%
划分后	5/7=71.4%

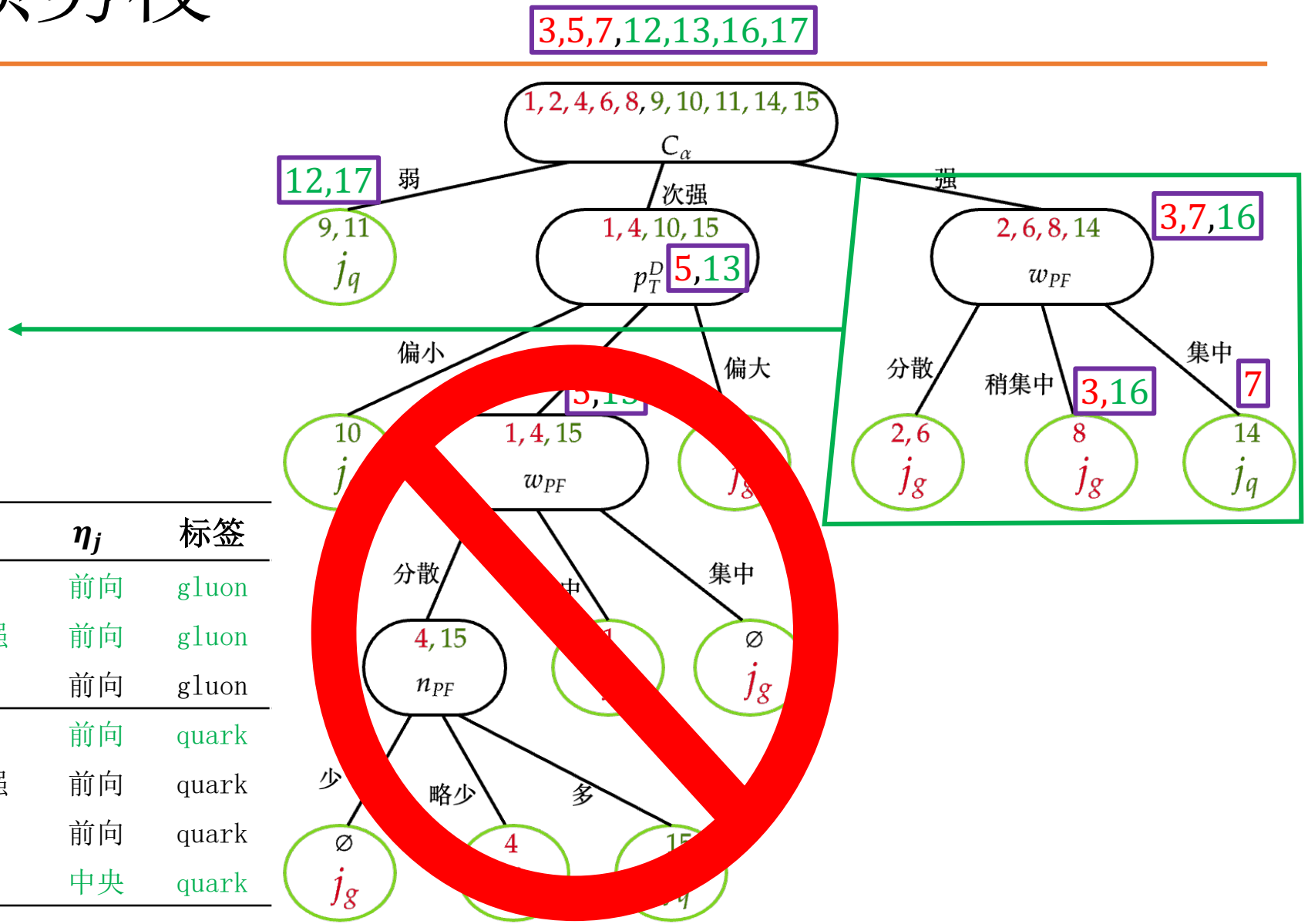


编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 – 预剪枝

- 验证集剪枝
  - 预剪枝

	验证集精度
划分前	5/7=71.4%
划分后	4/7=57.1%

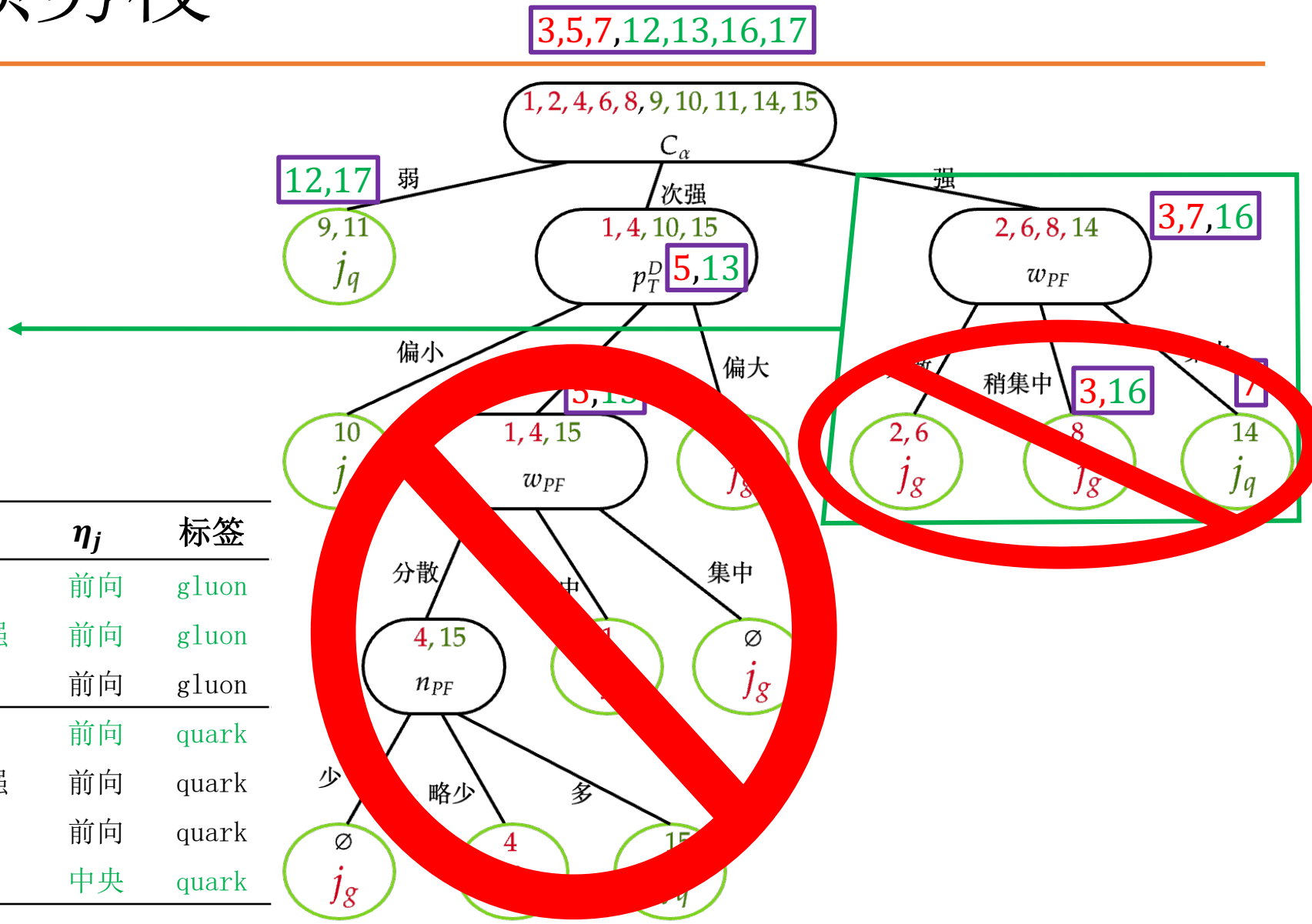


编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 - 预剪枝

- 验证集剪枝
  - 预剪枝

	验证集精度
划分前	5/7=71.4%
划分后	4/7=57.1%



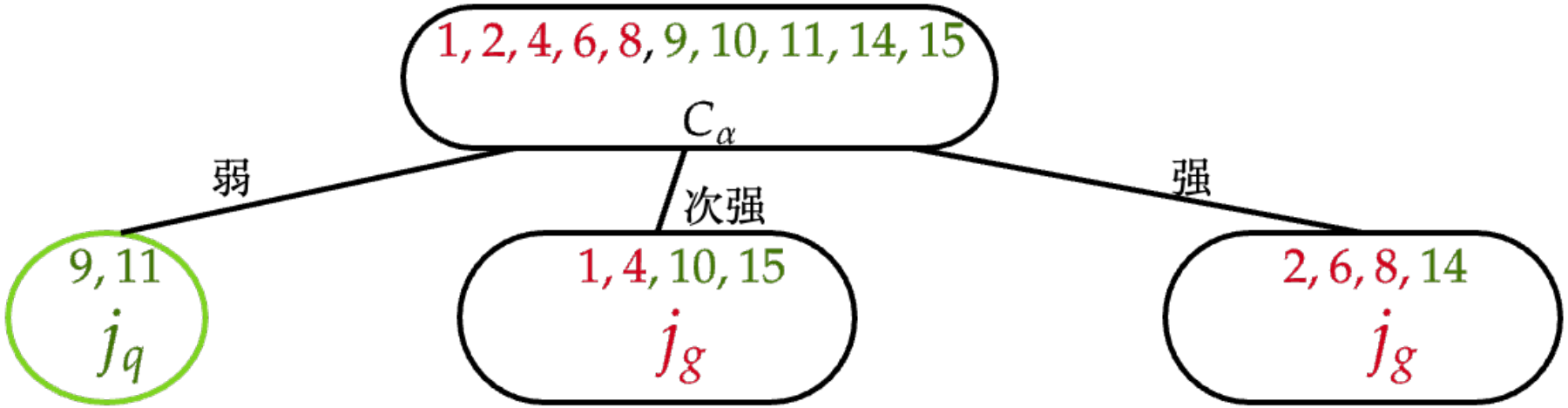
编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 - 预剪枝

- 验证集剪枝
  - 预剪枝
    - 减少训练开支 😃
    - 欠拟合 😩

编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

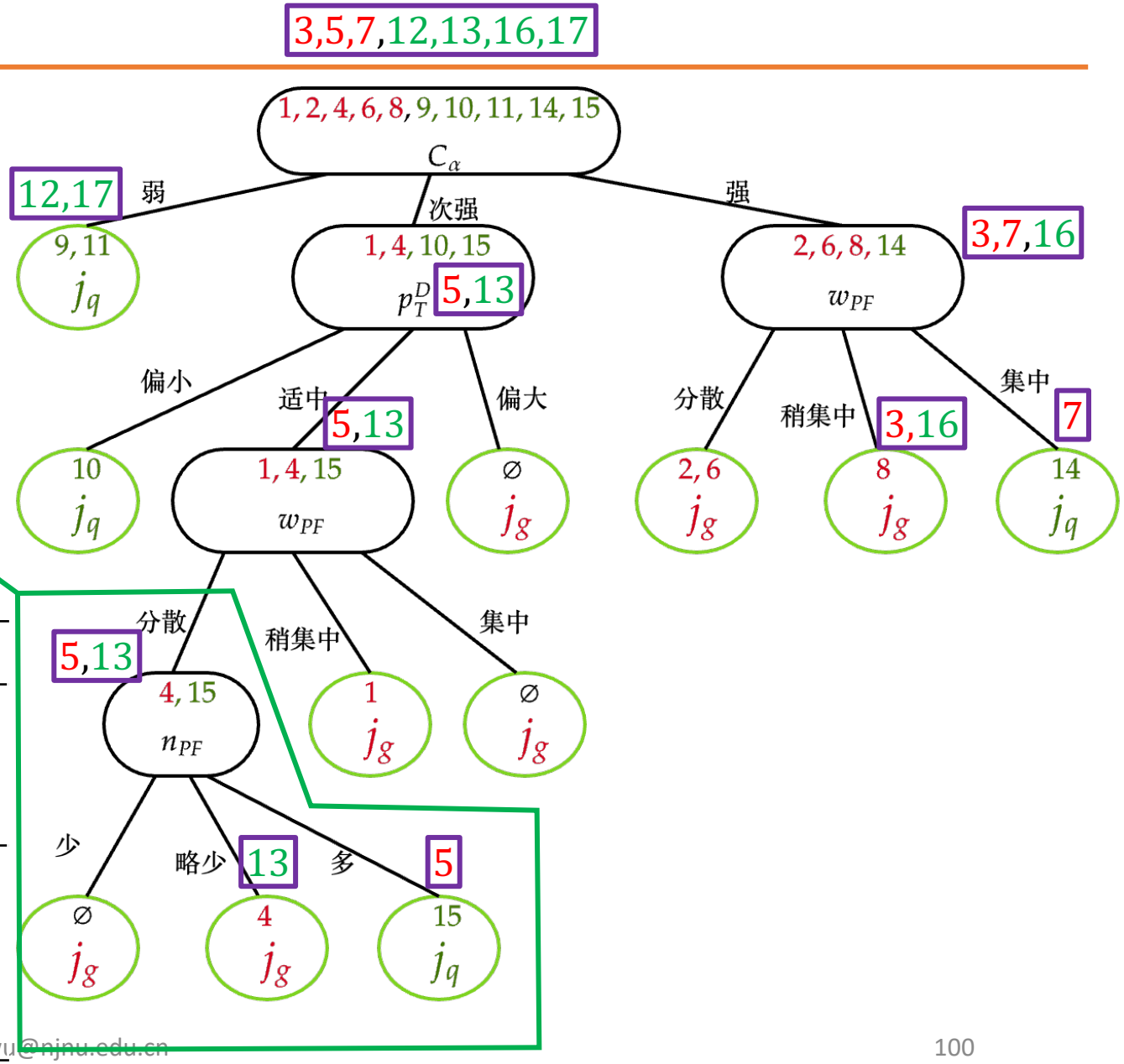
- 决策树桩(Stump)



决策树 - 后剪枝

- 后剪枝

	验证集精度
剪枝前	3/7=42.9%
剪枝后	4/7=57.1%

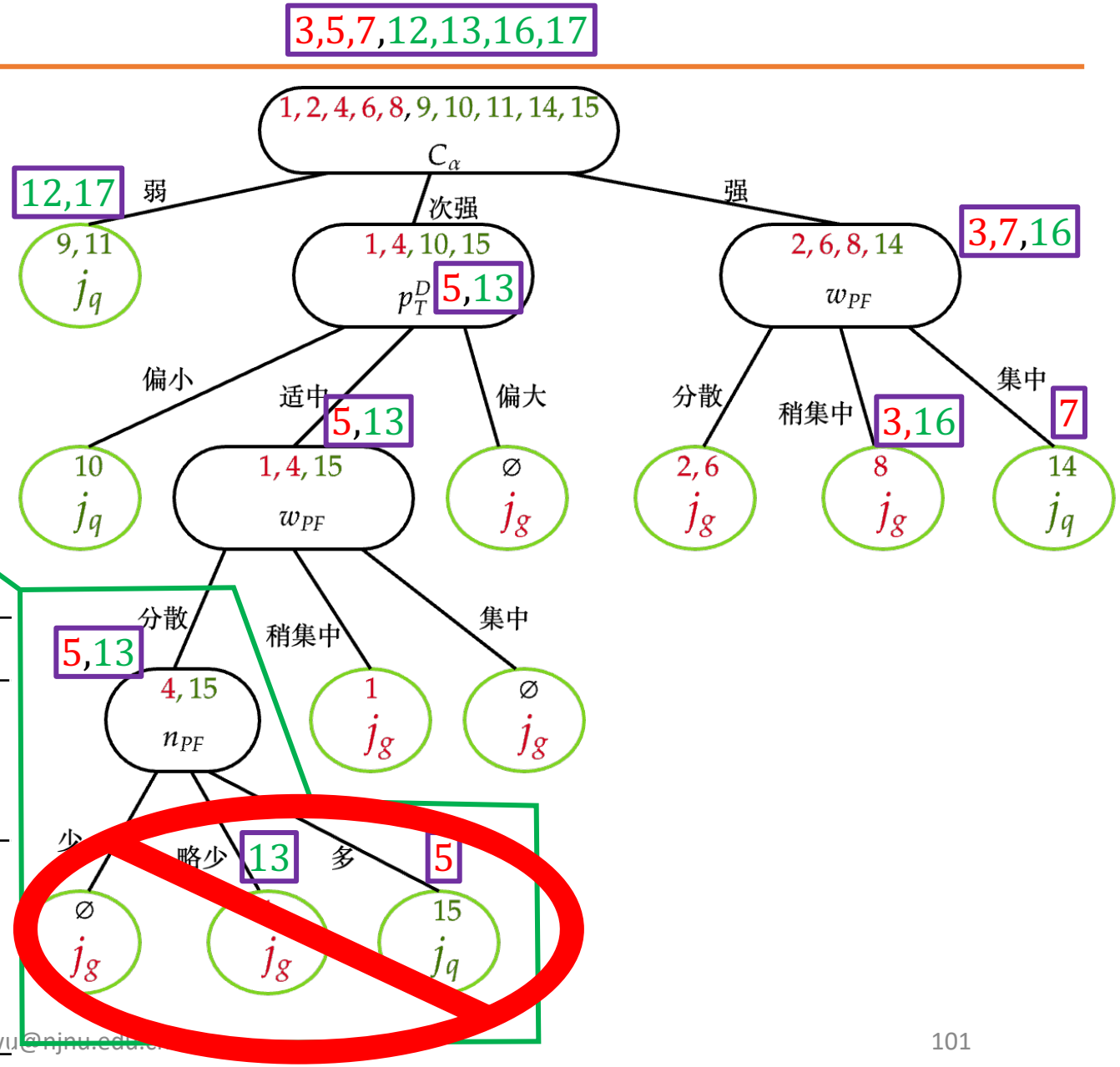


编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 - 后剪枝

- 后剪枝

	验证集精度
剪枝前	3/7=42.9%
剪枝后	4/7=57.1%

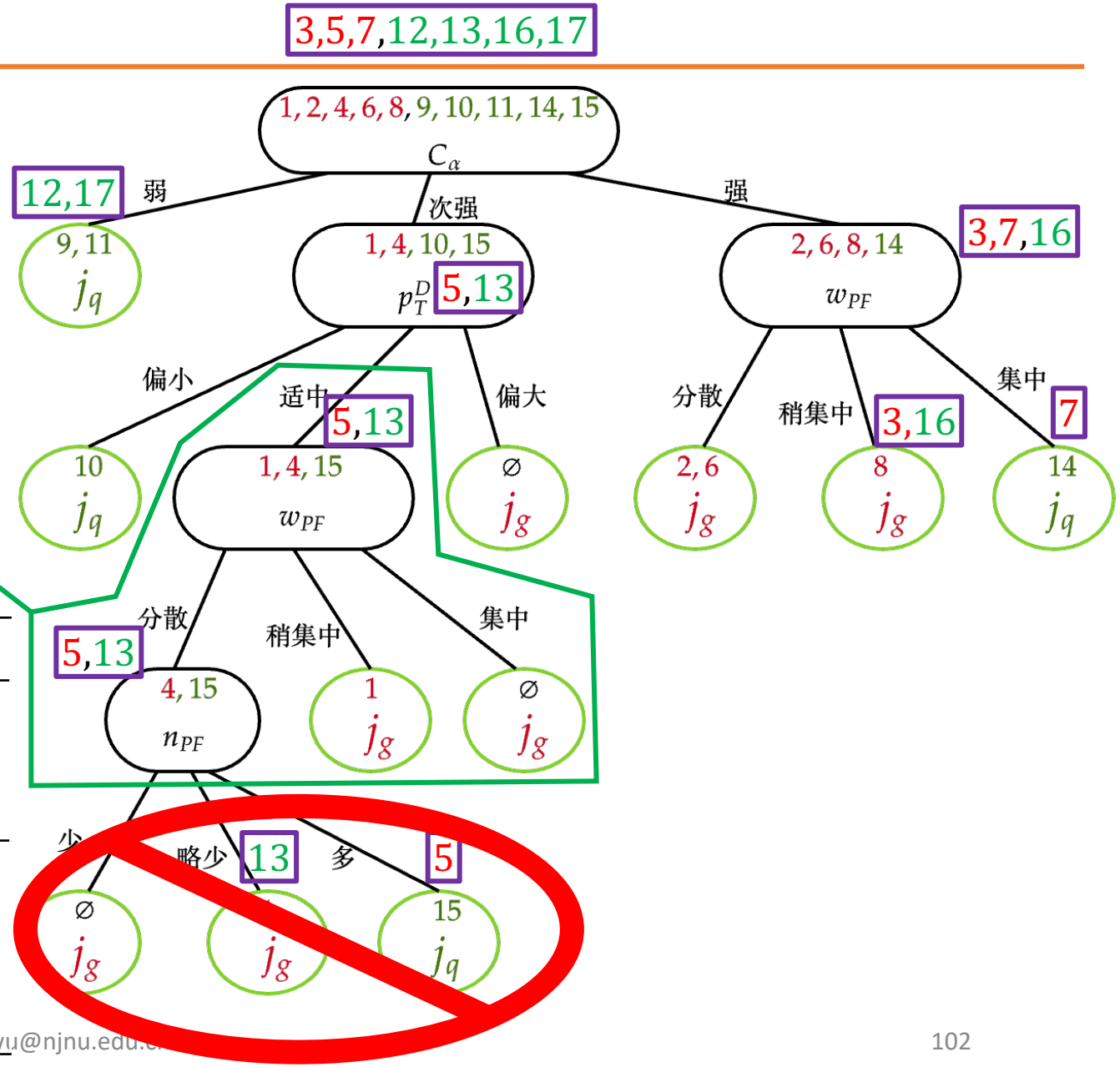


编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 – 后剪枝

- 后剪枝

	验证集精度
剪枝前	4/7=57.1%
剪枝后	4/7=57.1%

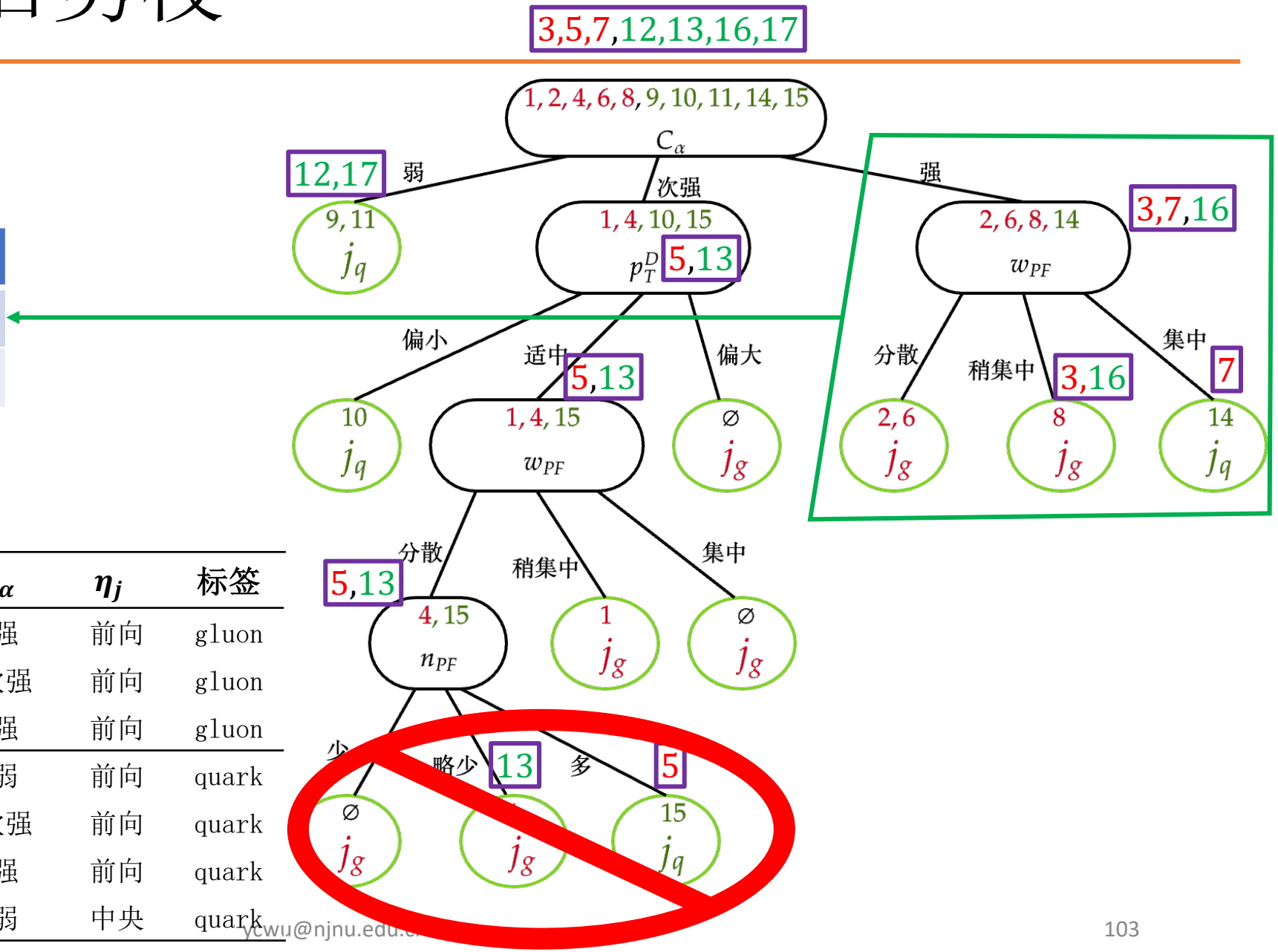


编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 - 后剪枝

- 后剪枝

	验证集精度
剪枝前	4/7=57.1%
剪枝后	5/7=71.4%

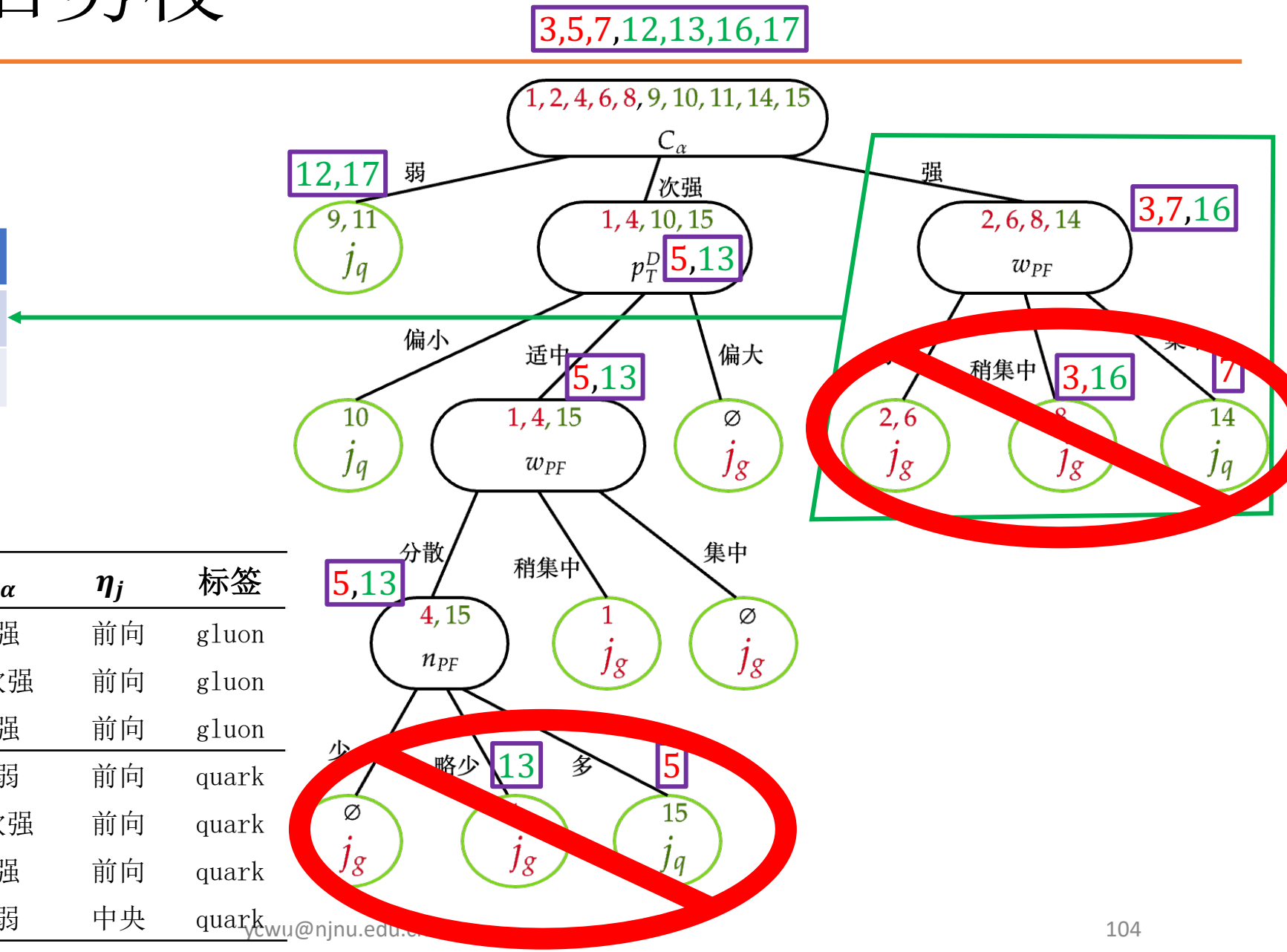


编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

决策树 – 后剪枝

- 后剪枝

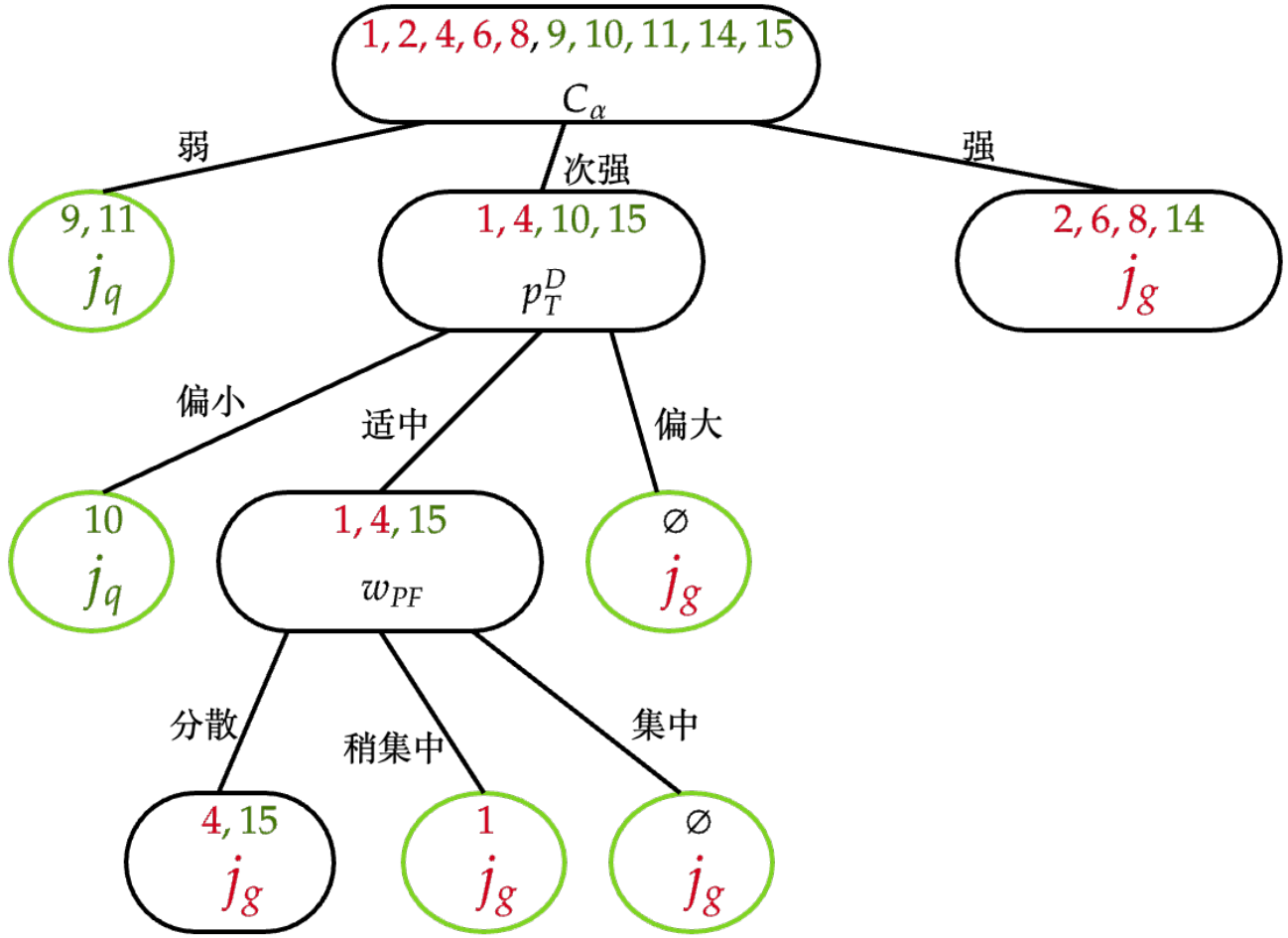
	验证集精度
剪枝前	4/7=57.1%
剪枝后	5/7=71.4%



编号	w_{PF}	p_T^D	x_{hard}	n_{PF}	C_α	η_j	标签
3	稍集中	偏小	中	多	强	前向	gluon
5	分散	适中	低	多	次强	前向	gluon
7	集中	偏小	低	多	强	前向	gluon
12	集中	偏大	高	少	弱	前向	quark
13	分散	适中	中	略少	次强	前向	quark
16	稍集中	适中	低	略少	强	前向	quark
17	集中	偏小	低	少	弱	中央	quark

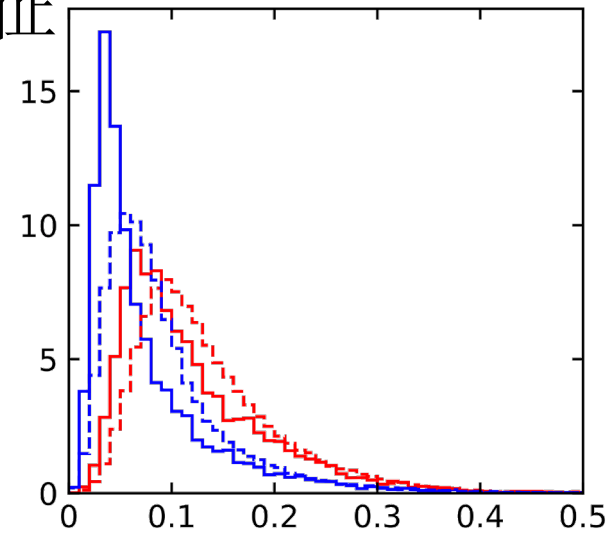
决策树 – 后剪枝

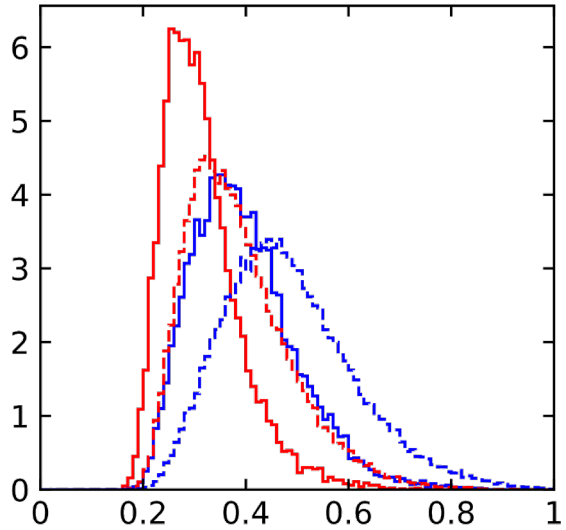
- 验证集剪枝
  - 后剪枝
    - 较小的欠拟合风险 😀
    - 更高的计算开销 😩

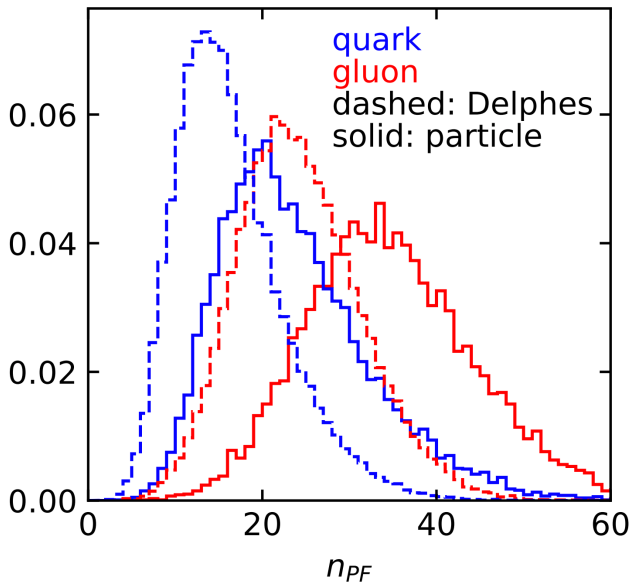


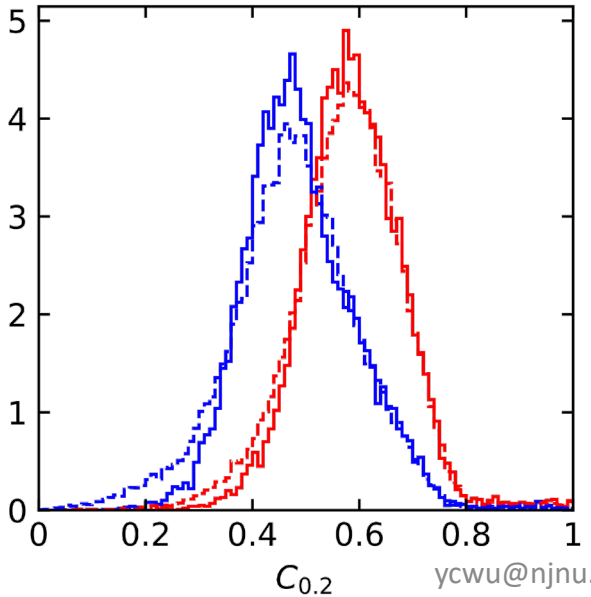
决策树 - 连续值

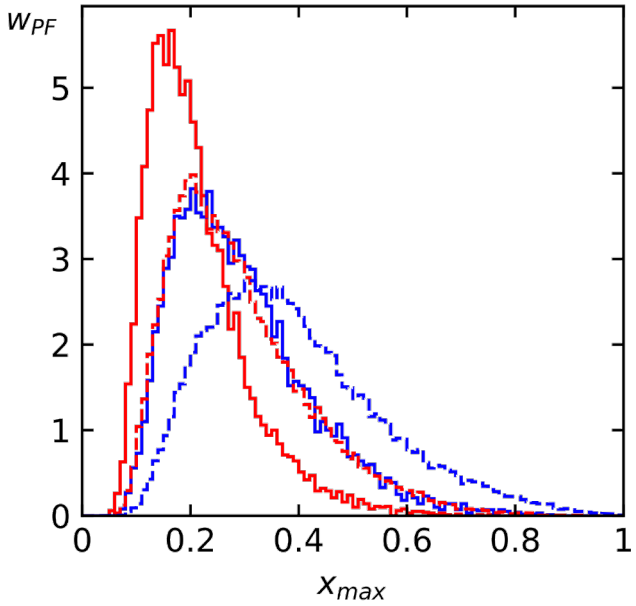
- quark/gluon-jet tagging的特征
  - 连续取值
    - 连续特征离散化
      - 二分法 - （C4.5算法中的方案）











决策树 – 连续值

- 连续特征离散化
  - 二分法 – C4.5算法中的方案
- 考虑数据集D中的某个连续取值的特征a
  - 特征a在数据集D上有n_a个取值：$\{a_1, a_2, \cdots, a_{n_a}\}$ – 从小到大排列
    - 已经离散化了
  - 划分点的候选者

$$T^a = \left\{ \frac{a_i + a_{i+1}}{2} \middle| 1 \le i \le n_a - 1 \right\}$$

    - 每个划分点，将数据集D划分为两个子集：$\{D_a^{t-}, D_a^{t+}\}$（$a < t$、$a > t$）
  - 最优划分的选择 – 信息增益/增益率/基尼指数

$$\Delta\mathcal{E}(D, a) = \max_{t \in T^a} \left(\mathcal{E}(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_a^{t\lambda}|}{|D|} \mathcal{E}\left(D_a^{t\lambda}\right) \right)$$

回归树

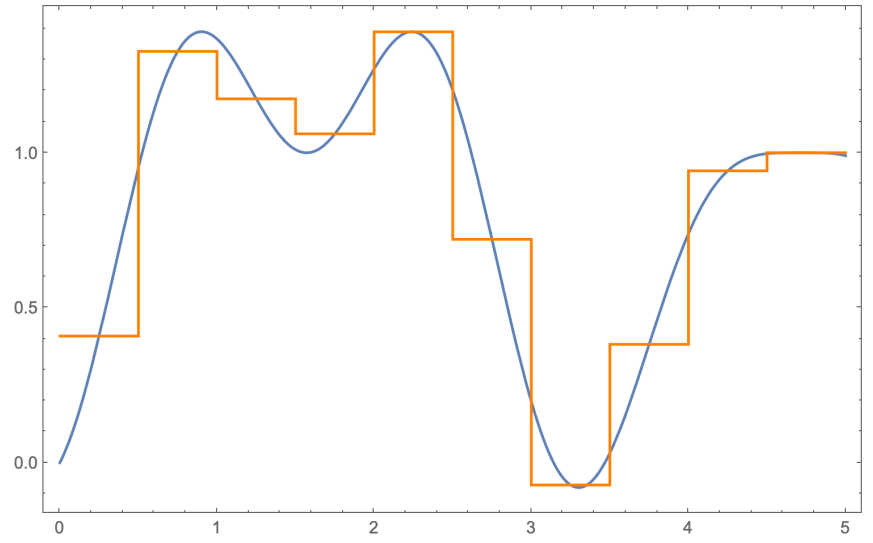
- 不止于分类 – 回归
  - 数据集：$D = \{(\vec{x}_1, y_1), \cdots, (\vec{x}_m, y_m)\}$
  - 如何用树状结构描述一个“连续”函数？
    - 分块描述
- 假设已经将输入空间划分为M个独立子区域$\{R_1, \cdots, R_M\}$
  - 每个子区域R_k有一个固定输出c_k
  - 完整的模型：

$$f(\vec{x}) = \sum_{k=1}^{M} c_k \mathbb{I}(\vec{x} \in R_k)$$

    - 预测误差 $\epsilon = \left(y - f(\vec{x})\right)^2$

$$c_k = \arg\min_c \sum_{x_i \in R_k} \left(y_i - f(\vec{x}_i)\right)^2 = \arg\min_c \sum_{x_i \in R_k} (y_i - c)^2 = \bar{y}_k$$

  - 如何划分区域？



在子区域R_k中的样本的y的平均值

回归树

- 区域划分 – 二叉划分
  - 对输入的第ℓ分量（切分变量）以及某一切分点s可以划分两个区域

$$R^-(\ell,s)=\{\vec{x}|\vec{x}^{\ell}\le s\},\qquad R^+(\ell,s)=\{\vec{x}|\vec{x}^{\ell}>s\}$$

  - 那么对某个节点上的数据寻找最优切分变量和切分点

$$(\ell^*,s^*)=\arg\min_{\ell,s}\left[\min_{c^-}\sum_{\vec{x}_i\in R^-(\ell,s)}(y_i-c^-)^2+\min_{c^+}\sum_{\vec{x}_i\in R^+(\ell,s)}(y_i-c^+)^2\right]$$

    - 以及得到相应区域内的模型预测值：

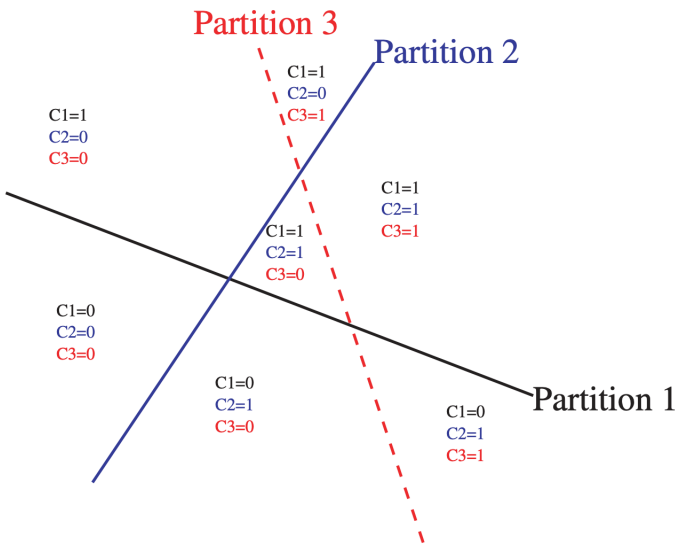
$$c^-=ave\left(y_i|\vec{x}_i\in R^-(\ell^*,s^*)\right),\qquad c^+=ave\left(y_i|\vec{x}_i\in R^+(\ell^*,s^*)\right)$$

  - 对每个区域重复进行划分，直到满足约定的停止条件
    - 节点内样本具有一样的输出、节点样本数、误差下降值

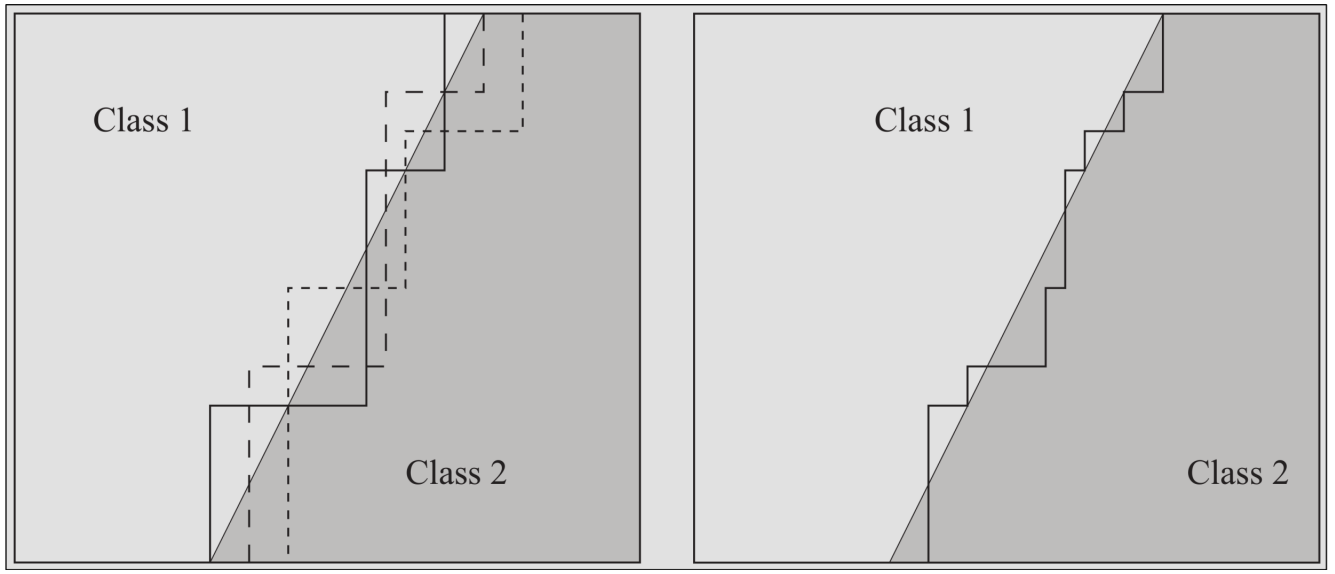
集成学习

集成学习

- 集成学习
  - 通过构建并结合多个模型来完成任务
    - 三个臭皮匠顶个诸葛亮



(a)

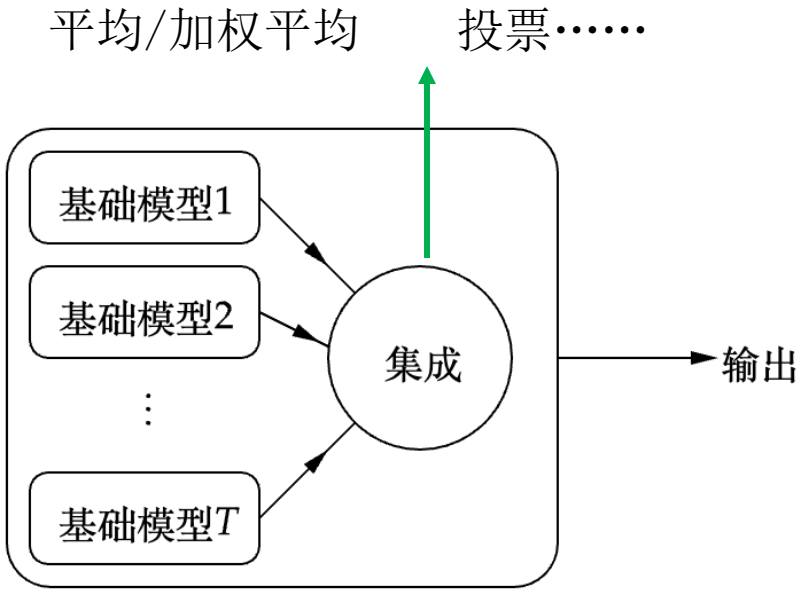


(b)

arXiv: 2206.09645

集成学习

- 强学习模型 vs. 弱学习模型
  - 正确率很高 vs. 正确率只比随机猜测要好一点
- 弱模型比强模型更容易获得
  - 集成学习 – 从弱模型到强模型
    - 基础模型：
      - 不能太差 – 弱模型 $>50\%$ 精度
      - 多样性



	样本1	样本2	样本3
h_1	✓	✓	×
h_2	×	✓	✓
h_3	✓	×	✓
集成	✓	✓	✓

集成提升性能

	样本1	样本2	样本3
h_1	✓	✓	×
h_2	✓	✓	×
h_3	✓	✓	×
集成	✓	✓	×

集成不起作用

	样本1	样本2	样本3
h_1	✓	×	×
h_2	×	✓	×
h_3	×	×	✓
集成	×	×	×

集成降低性能

集成学习

- 集成学习的效果
  - 二分类情景：$y \in \{-1, +1\}$，真实模型：$y = f(\vec{x})$
  - 假设基础模型h_i的错误率为ϵ

$$P(h_i(\vec{x}) \neq f(\vec{x})) = \epsilon$$

  - 投票法集成总共$T = 2k + 1$个基础模型$\{h_1, \cdots, h_T\}$

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

  - 集成模型的错误率
    - 小于等于k个基础模型预测正确

$$P\left(H(\vec{x}) \neq f(\vec{x})\right) = \sum_{i=0}^{k} C_T^i (1-\epsilon)^i \epsilon^{T-i} \leq e^{-\frac{1}{2}T(1-2\epsilon)^2}$$

霍夫丁不等式

没考虑模型间关联

集成学习

- 集成学习的效果
  - k个回归模型$\{h_1, \cdots, h_k\}$，h_i在样本上的误差ϵ_i
  - 假设这个误差服从一个多维(k维)正态分布
    - 均值为0，方差$\mathbb{E}[\epsilon_i^2] = \sigma^2$，协方差$\mathbb{E}[\epsilon_i \epsilon_j] = \rho^2$
  - 如果考虑集成的强模型是弱模型的平均，集成模型的误差是$\frac{1}{k}\sum_i \epsilon_i$
  - 因此，集成模型误差平方的期望是

$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \frac{1}{k^2}\mathbb{E}\left[\sum_i \epsilon_i^2 + \sum_i \sum_{j \neq i} \epsilon_i \epsilon_j\right] = \frac{1}{k^2}\sum_i \mathbb{E}[\epsilon_i^2] + \frac{1}{k^2}\sum_i \sum_{j \neq i} \mathbb{E}[\epsilon_i \epsilon_j] = \frac{1}{k}\sigma^2 + \frac{k-1}{k}\rho^2$$

$$= \begin{cases} \frac{\sigma^2}{k}, & \rho^2 = 0 \\ \sigma^2, & \rho^2 = \sigma^2 \end{cases}$$

$\rho^2 = 0$ ← 基础模型无关联

$\rho^2 = \sigma^2$ ← 基础模型强关联

集成学习

- 集成学习的效果

$$P\left(H(\vec{x}) \neq f(\vec{x})\right) \leq e^{-\frac{1}{2}T(1-2\epsilon)^2}$$

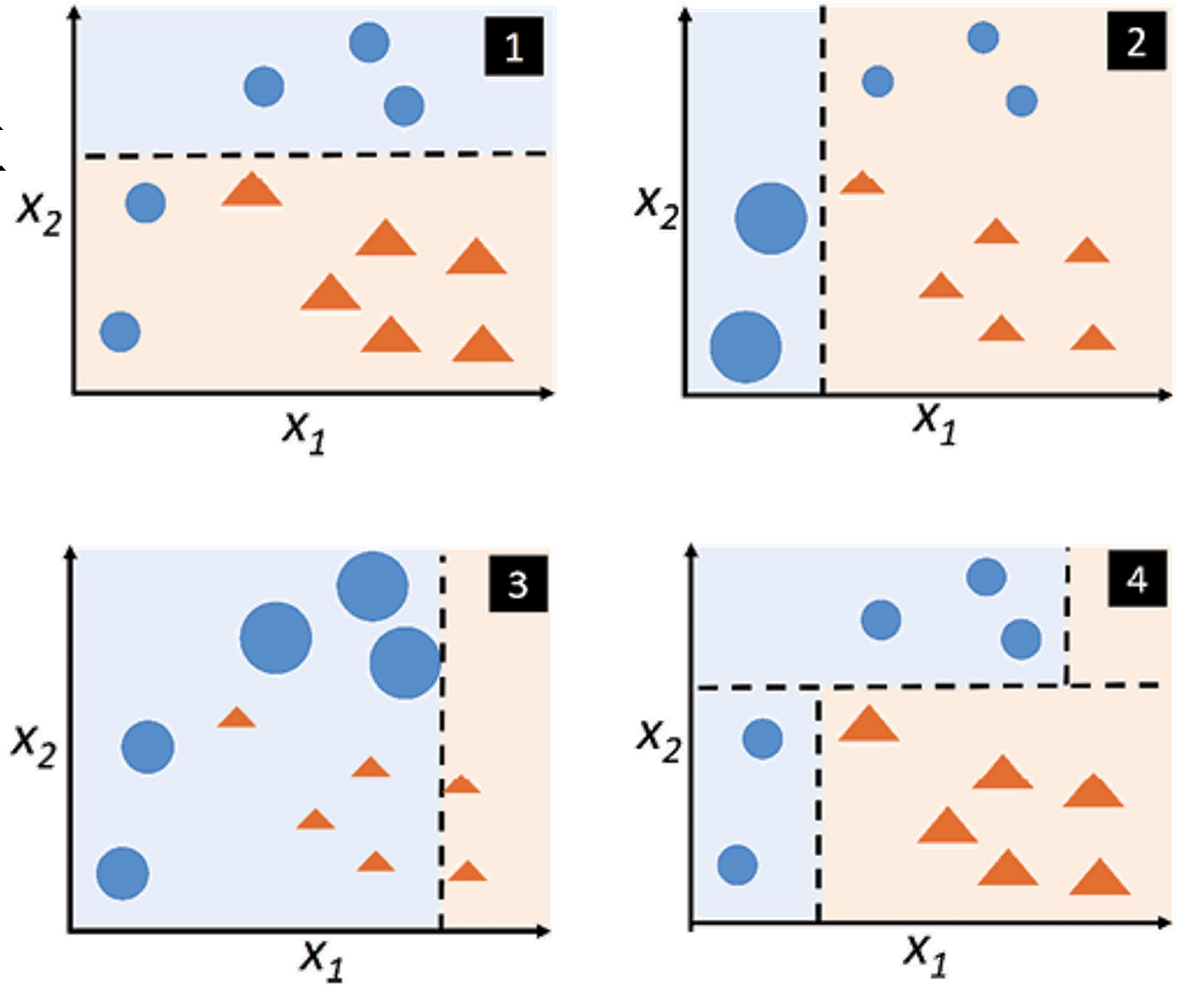
$$\mathbb{E}\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)^2\right] = \begin{cases} \frac{\sigma^2}{k}, & \rho^2 = 0 \\ \sigma^2, & \rho^2 = \sigma^2 \end{cases}$$

  - 基础模型之间弱关联
    - 大量有一定效果并且不同的基础模型

- 集成学习方法
  - 基础模型之间强依赖 - 串行
    - Boosting - AdaBoost
  - 基础模型之间弱依赖 - 并行
    - Bagging/Random Forest

集成学习 - Boosting

- Boosting算法
  - 递归产生所有基础模型
  - 根据上一个基础模型调整数据权重
    - 预测错误的数据增加权重
    - 基于权重调整的数据训练新模型
  - 所有基础模型的加权叠加
  - 每个基础模型可以很弱
- AdaBoost
  - Adaptive Boosting
- Gradient Boosting
- XGB
  - eXtreme Gradient Boosting



集成学习 - AdaBoost

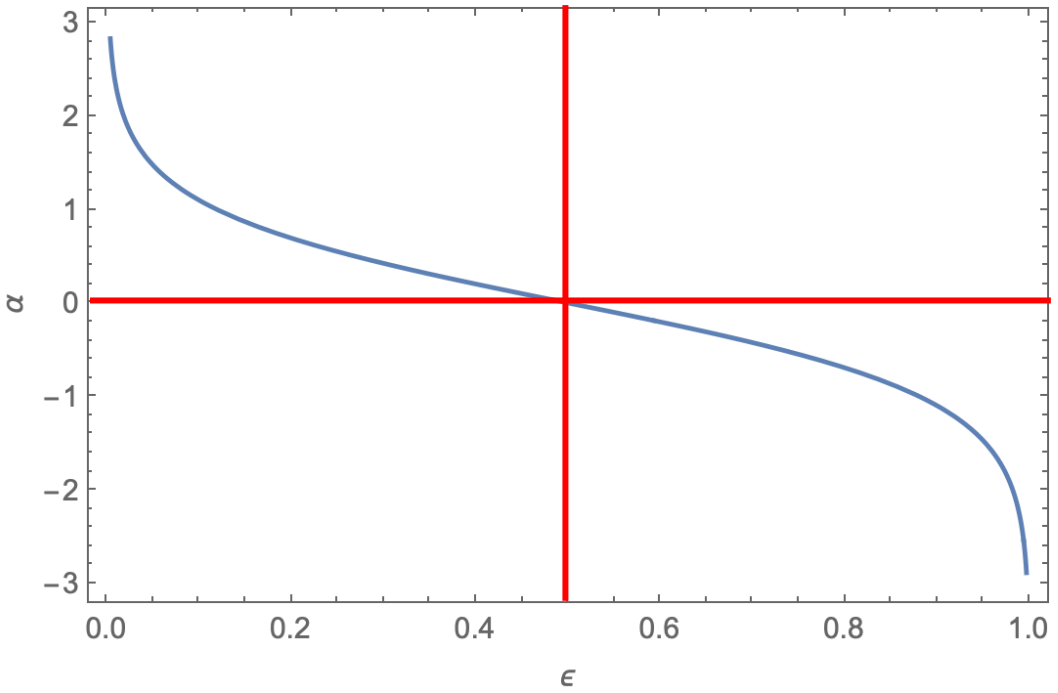
- AdaBoost算法
  - 二分类数据集$D=\{(\vec{x}_1,y_1),\cdots,(\vec{x}_m,\vec{y}_m)\},\ |D|=m,\ y_i\in\mathcal{Y}=\{-1,+1\}$
  - 初始化数据集权重：$W_1=\{w_{11},\cdots,w_{1m}\},\ w_{1i}=\frac{1}{m}$
  - 递归产生基础模型$\ell=1,\cdots,T$
    - 基于已经得到的数据集权重W_ℓ，学习得到基础模型h_ℓ
      - 基础模型h_ℓ最小化带权重W_ℓ的数据集上的分类误差
        - $\epsilon_\ell=\sum_{i=1}^{m}w_{\ell i}\,\mathbb{I}(h_\ell(x_i)\neq y_i)$
    - 计算基础模型h_ℓ的权重系数
      - $\alpha_\ell=\frac{1}{2}\ln\frac{1-\epsilon_\ell}{\epsilon_\ell}$
    - 产生新的数据集权重：$W_{\ell+1}=\{w_{\ell+1,1},\cdots,w_{\ell+1,m}\}$
      - $w_{\ell+1,i}=\frac{w_{\ell i}}{N_\ell}e^{-\alpha_\ell y_i h_\ell(\vec{x}_i)},\quad N_\ell=\sum_{i=1}^{m}w_{\ell i}\,e^{-\alpha_\ell y_i h_\ell(\vec{x}_i)}$
  - 得到最终的集成模型：

$$H(\vec{x})=sign\big(h(\vec{x})\big)=sign\left(\sum_{\ell=1}^{T}\alpha_\ell h_\ell(\vec{x})\right)$$

集成学习 - AdaBoost

- 二分类AdaBoost算法中的权重系数
  - 基础模型的权重
    - 误差越小的模型权重越大
      - 在最终模型中的作用越大
    - $\epsilon > 0.5$?

$$\alpha_\ell = \frac{1}{2}\ln\frac{1-\epsilon_\ell}{\epsilon_\ell}$$



  - 数据集的权重
    - 分类正确的样本 - 权重减小
    - 分类错误的样本 - 权重增加
    - 数据集本身没有任何改变，权重改变 - 产生足够弱模型

$$w_{\ell+1,i} = \frac{w_{\ell i}}{N_\ell} e^{-\alpha_\ell y_i h_\ell(\vec{x}_i)} = \frac{w_{\ell i}}{N_\ell} \times \begin{cases} e^{-\alpha_\ell}, & h_\ell(\vec{x}_i) = y_i \\ e^{+\alpha_\ell}, & h_\ell(\vec{x}_i) \neq y_i \end{cases}$$

集成学习 – AdaBoost

- AdaBoost算法例子
  - 二叉树桩 – 基础模型

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
W_1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
$H(x)$	1	1	1	-1	-1	-1	-1	-1	-1	-1

  - 初始化数据权重，学习第一个模型
    - 连续取值的特征 – 划分点：$\{0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5\}$
    - 得到误差，计算模型权重

$$\epsilon_1 = w_{17} + w_{18} + w_{19} = 0.3$$

$$h_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x \geq 2.5 \end{cases} \qquad \alpha_1 = \frac{1}{2}\ln\frac{1-\epsilon_1}{\epsilon_1} \approx 0.4236$$

$$H(x) = sign\big(\alpha_1 h_1(x)\big)$$

集成学习 - AdaBoost

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
W_1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
W_2	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.1667	0.1667	0.1667	0.07143
$H(x)$	1	1	1	1	1	1	1	1	1	-1

- 更新数据权重 $\Longrightarrow$ W_2 $\Longrightarrow$ 训练第二个基础模型

$$h_2(x) = \begin{cases} 1, & x < 8.5 \\ -1, & x \geq 8.5 \end{cases}$$

$$\epsilon_2 = w_{24} + w_{25} + w_{26} = 0.2143$$

$$\alpha_2 = \frac{1}{2}\ln\frac{1-\epsilon_2}{\epsilon_2} \approx 0.6496$$

$$H(x) = sign\left(0.4236h_1(x) + 0.6496h_2(x)\right)$$

集成学习 – AdaBoost

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
W_1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
W_2	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.1667	0.1667	0.1667	0.07143
W_3	0.04545	0.04545	0.04545	0.1667	0.1667	0.1667	0.1061	0.1061	0.1061	0.04545
$H(x)$	1	1	1	-1	-1	-1	1	1	1	-1

- 更新数据权重 $\Longrightarrow$ W_3 $\Longrightarrow$ 训练第三个基础模型

$$h_3(x) = \begin{cases} 1, & x \geq 5.5 \\ -1, & x < 5.5 \end{cases}$$

$$\epsilon_3 = w_{31} + w_{32} + w_{33} + w_{3,10} = 0.1818$$

$$\alpha_3 = \frac{1}{2}\ln\frac{1-\epsilon_3}{\epsilon_3} \approx 0.7520$$

$$H(x) = sign\left(0.4236h_1(x) + 0.6496h_2(x) + 0.7520h_3(x)\right)$$

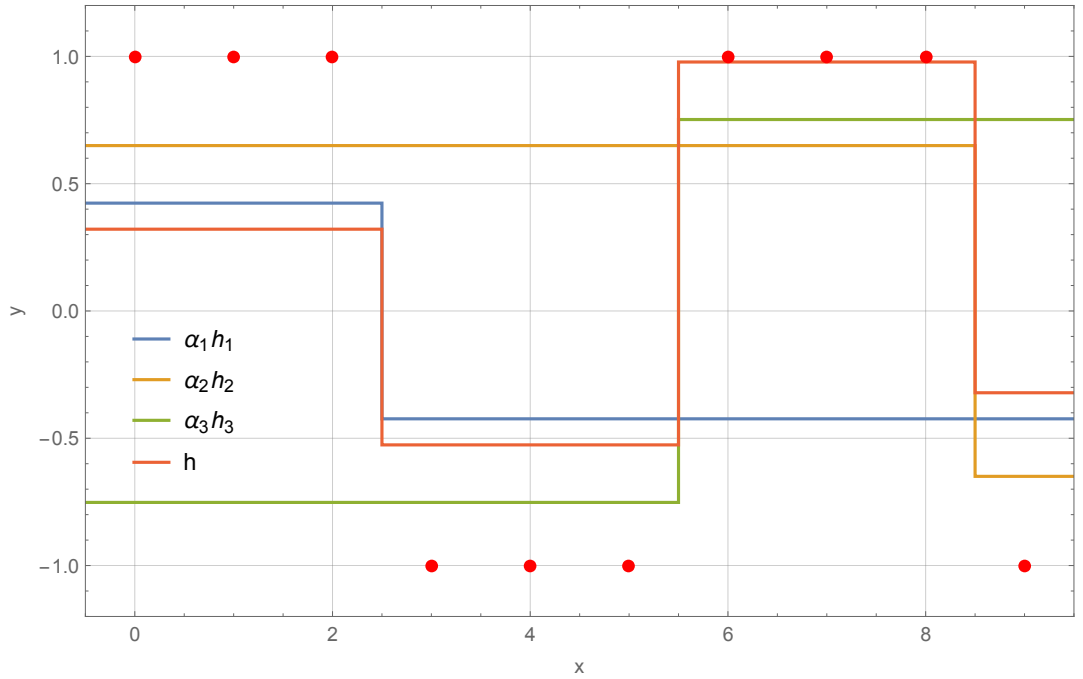
集成学习 – AdaBoost

- AdaBoost

$$h_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x \geq 2.5 \end{cases} \qquad \alpha_1 \approx 0.4236$$

$$h_2(x) = \begin{cases} 1, & x < 8.5 \\ -1, & x \geq 8.5 \end{cases} \qquad \alpha_2 \approx 0.6496$$

$$h_3(x) = \begin{cases} 1, & x \geq 5.5 \\ -1, & x < 5.5 \end{cases} \qquad \alpha_3 \approx 0.7520$$



$$H(x) = sign\left(0.4236h_1(x) + 0.6496h_2(x) + 0.7520h_3(x)\right)$$

集成学习 – AdaBoost

- AdaBoost的训练误差
  - 有上界

$$\epsilon = \frac{1}{N}\sum_{i=1}^{m} \mathbb{I}(H(\vec{x}_i) \neq y_i) \leq \frac{1}{N}\sum_{i}^{m} e^{-y_i h(\vec{x}_i)} = \prod_{\ell=1}^{T} N_\ell$$

$$\left(\mathbb{I}(H(\vec{x}_i) \neq y_i) = \begin{cases} 0 \\ 1 \end{cases}\right) \leq e^{-y_i h(\vec{x}_i)}$$

$$w_{\ell+1,i} = \frac{w_{\ell i}}{N_\ell} e^{-\alpha_\ell y_i h_\ell(\vec{x}_i)}$$

$$\frac{1}{N}\sum_{i=1}^{m} e^{-y_i h(\vec{x}_i)} = \frac{1}{N}\sum_{i}^{m} e^{-y_i \sum_{\ell=1}^{T} \alpha_\ell h_\ell(\vec{x}_i)} \xrightarrow{w_{1i}=\frac{1}{N}} \sum_{i}^{m} w_{1i} e^{-y_i \alpha_1 h_1(\vec{x}_i)} e^{-y_i \sum_{\ell=2}^{T} \alpha_\ell h_\ell(\vec{x}_i)}$$

$$= \sum_{i}^{m} N_1 w_{2i} e^{-y_i \alpha_2 h_2(\vec{x}_i)} e^{-y_i \sum_{\ell=3}^{T} \alpha_\ell h_\ell(\vec{x}_i)} = \cdots = \prod_{\ell=1}^{T} N_\ell$$

$$N_\ell = \sum_{i=1}^{m} w_{\ell i}\, e^{-\alpha_\ell y_i h_\ell(\vec{x}_i)}$$

集成学习 – AdaBoost

- AdaBoost的训练误差
  - 有上界

$$\epsilon = \frac{1}{N}\sum_{i=1}^{m} \mathbb{I}(H(\vec{x}_i) \neq y_i) \leq \frac{1}{N}\sum_{i}^{m} e^{-y_i h(\vec{x}_i)} = \prod_{\ell=1}^{T} N_\ell$$

  - 上界可控

$$N_\ell = \sum_{i=1}^{m} w_{\ell i}\, e^{-\alpha_\ell y_i h_\ell(\vec{x}_i)} = \sum_{y_i = h_\ell(\vec{x}_i)} w_{\ell i} e^{-\alpha_\ell} + \sum_{y_i \neq h_\ell(\vec{x}_i)} w_{\ell i} e^{\alpha_\ell}$$

$$= (1-\epsilon_\ell)e^{-\alpha_\ell} + \epsilon_\ell e^{\alpha_\ell} = 2\sqrt{(1-\epsilon_\ell)\epsilon_\ell} = \sqrt{1-4\delta_\ell^2} \leq e^{-2\delta_\ell^2} \leq 1$$

$$\alpha_\ell = \frac{1}{2}\ln\frac{1-\epsilon_\ell}{\epsilon_\ell}$$

$$\delta_\ell = \frac{1}{2} - \epsilon_\ell$$

    - 如果每个基础模型都比随机模型好一点$\delta_\ell \geq \delta$

$$\epsilon = \prod_{\ell}^{T} N_\ell \leq e^{-2\sum_{\ell=1}^{T}\delta_\ell^2} \leq e^{-2T\delta^2}$$

集成学习 – 加法模型和前向分步算法

- 加法模型

$$F(\vec{x}) = \sum_{\ell=1}^{T} f_\ell(\vec{x}, \vec{c}_\ell)$$

  - 待学习的模型参数 – $\{\vec{c}_1, \cdots, \vec{c}_\ell\}$
  - 基础模型f_ℓ视任务确定

- 学习加法模型 – 最小化损失函数

$$\min_{\{\vec{c}_\ell\}} \sum_{i}^{N} L\big(y_i, F(\vec{x}_i, \{\vec{c}_\ell\})\big)$$

  - 前向分步算法
    - 考虑模型序列：$\{F^1, F^2, \cdots, F^T\}$ $\quad F^k(\vec{x}) = \sum_{\ell=1}^{k} f_\ell(\vec{x}, \vec{c}_\ell)$
      - 从F^1开始从前往后依次优化$\vec{c}_k$

$$\min_{\vec{c}_k} \sum_{i}^{N} L\left(y_i, F^{k-1}(\vec{x}_i, \{\vec{c}_1, \cdots, \vec{c}_{k-1}\}) + f_k(\vec{x}_i, \vec{c}_k)\right)$$

集成学习 - 回归提升树

- AdaBoost算法是前向分步加法算法的特例 - 针对分类问题
  - 损失函数是指数函数$L\big(y,f(x)\big)=e^{-yf(x)}$
  - 此时比较容易得到对每个基础模型的优化
    - 可以证明AdaBoost中对基础模型的选择正是符合最小化指数损失函数的模型
- 回归树 - 平方误差函数
  - 也比较容易得到针对回归树的提升模型
  - 数据集$D=\{(\vec{x}_1,y_1),\cdots,(\vec{x}_m,y_m)\}$
    - 划分出M个子区域$\{R_1,\cdots,R_M\}$，以及每个区域对应的常数$\{c_1,\cdots,c_M\}$
      - 通过最小化平方误差

$$h(\vec{x};\Theta)=\sum_{k=1}^{M}c_k\mathbb{I}(\vec{x}\in R_k)$$

    - 参数$\Theta=\{(R_1,c_1),\cdots,(R_M,c_M)\}$

集成学习 – 回归提升树

- 回归树(或其他基础模型) + Boosting：前向分步
  - 始化模型：$H^0(\vec{x}) = 0$
  - 考虑递归：$\ell = 1, \cdots, T$
    - $H^{\ell}(\vec{x}) = H^{\ell-1}(\vec{x}) + h(\vec{x}; \Theta_{\ell})$
    - 其中$h(\vec{x}; \Theta_{\ell})$通过最小化相应的损失函数来确定

$$\Theta_{\ell} = \arg\min_{\Theta} \sum_{i=1}^{m} L\left(y_i, H^{\ell-1}(\vec{x}_i) + h(\vec{x}_i; \Theta)\right)$$

    - 如果考虑平方损失函数 $L\big(y, \hat{y} = H(x)\big) = (y - \hat{y})^2$

$$\hat{y}_i^{\ell-1} = H^{\ell}(\vec{x}_i)$$

$$\hat{y}_i = h(\vec{x}_i; \Theta)$$

$$L\left(y_i, H^{\ell-1}(\vec{x}_i) + h(\vec{x}_i; \Theta)\right) = \left(y_i - \hat{y}_i^{\ell-1} - \hat{y}_i\right)^2 = \left(r_i^{\ell} - \hat{y}_i\right)^2$$

残差更大的数据：
- 模型拟合差
- 平方损失函数中天然贡献大

第ℓ轮回归树$h(\vec{x}; \Theta_{\ell})$真实拟合的目标 – 上一轮模型$H^{\ell-1}(\vec{x})$与真实目标之间的残差

集成学习 – 回归提升树

- 回归提升树例子

x	1	2	3	4	5	6	7	8	9	10
y	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05

  - 基础模型考虑回归树桩 – 切分变量 – x
  - 第一步：切分点$\{1.5,\ 2.5,\ 3.5,\ 4.5,\ 5.5,\ 6.5,\ 7.5,\ 8.5,\ 9.5\}$
    - 切分点$s = 1.5$

$$c^- = 5.56, \qquad c^+ = \frac{1}{9}(5.70 + 5.91 + 6.40 + 6.80 + 7.05 + 8.90 + 8.70 + 9.00 + 9.05) = 7.50$$

$$L = (5.56 - 5.56)^2 + (5.70 - 7.50)^2 + \cdots + (9.05 - 7.50)^2 = 15.72$$

s	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
L	15.72	12.08	8.37	5.78	3.91	1.93	8.01	11.74	15.74

$$\Theta_1 = \{s = 6.5, c^- = 6.24, c^+ = 8.91\}$$

$$h(x;\Theta_1) = \begin{cases} 6.24, & x \le 6.5 \\ 8.91, & x > 6.5 \end{cases} \qquad H^1 = h(x;\Theta_1)$$

集成学习 - 回归提升树

- 回归提升例子

x	1	2	3	4	5	6	7	8	9	10
$r_1 = y$	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05
r_2	-0.68	-0.54	-0.33	0.16	0.56	0.81	-0.01	-0.21	0.09	0.14

- 计算残差 - 用于第二轮回归树训练

s	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
L	1.42	1.01	0.80	1.14	1.67	1.93	1.93	1.90	1.91

$$\Theta_2 = \{s = 3.5, c^- = -0.51, c^+ = 0.22\}$$

$$h(x;\Theta_2) = \begin{cases} -0.51, & x \le 3.5 \\ 0.22, & x > 3.5 \end{cases}$$

$$H^2 = h(x;\Theta_1) + h(x;\Theta_2) = \begin{cases} 5.73, & x \le 3.5 \\ 6.46, & 3.5 < x \le 6.5 \\ 9.13, & 6.5 < x \end{cases}$$

集成学习 – 回归提升树

- 回归提升树例子

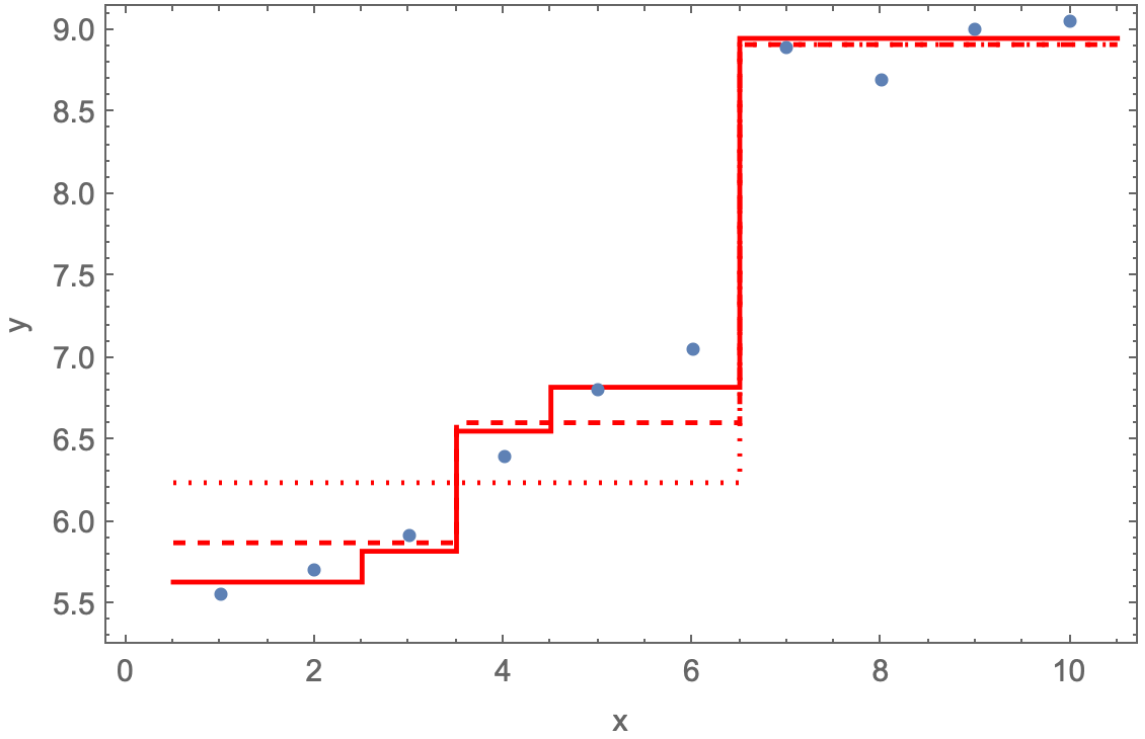
$$h(x;\Theta_3)=\begin{cases}0.15, & x\le 6.5\\ -0.22, & x>6.5\end{cases},\qquad L=0.48$$

$$h(x;\Theta_4)=\begin{cases}-0.16, & x\le 4.5\\ 0.11, & x>4.5\end{cases},\qquad L=0.31$$

$$h(x;\Theta_5)=\begin{cases}0.07, & x\le 6.5\\ -0.11, & x>6.5\end{cases},\qquad L=0.23$$

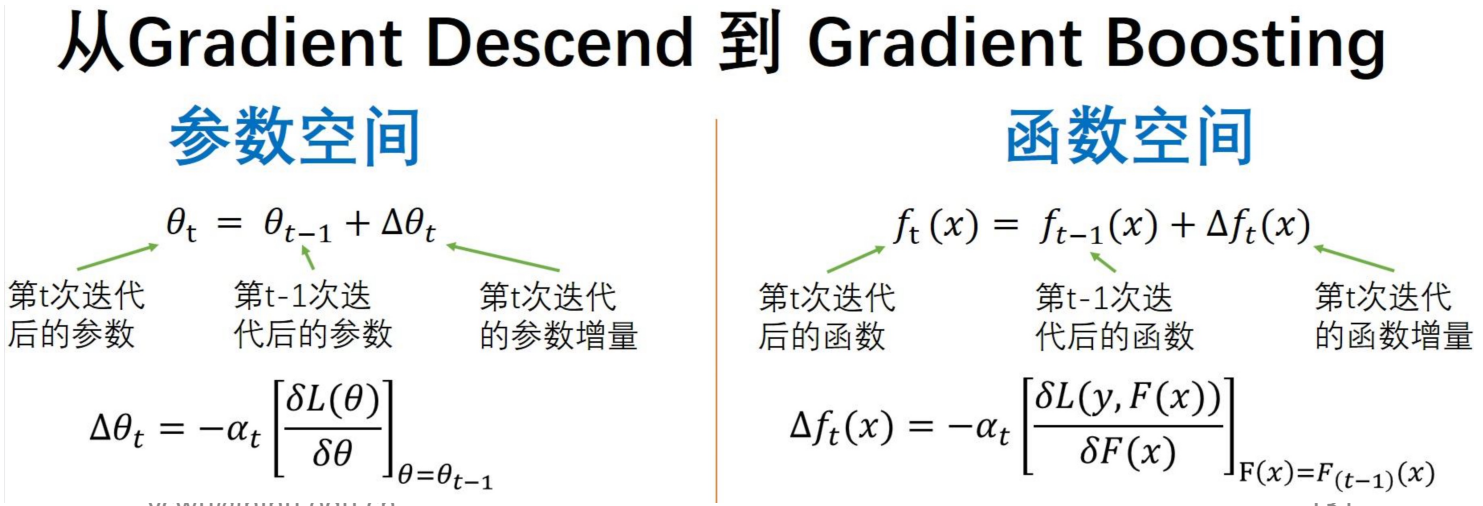
$$h(x;\Theta_6)=\begin{cases}-0.15, & x\le 2.5\\ 0.04, & x>2.5\end{cases},\qquad L=0.17$$

$$H^6(x)=\sum_{\ell} h(x;\Theta_\ell)=\begin{cases}5.63, & x\le 2.5\\ 5.82, & 2.5<x\le 3.5\\ 6.56, & 3.5<x\le 4.5\\ 6.83, & 4.5<x\le 6.5\\ 8.95, & 6.5\le x\end{cases}$$



集成学习 - 梯度提升Gradient Boosting

- 前向分步加法算法
  - 损失函数
    - 指数函数$L(y, f(x)) = e^{-yf(x)}$
    - 平方误差函数$L(y, f(x)) = (y - f(x))^2$
  - 此时比较容易得到对每个基础模型的优化
- 一般情况的损失函数$L(y, \hat{y} = f(x))$
  - 考虑损失函数对模型函数的“梯度”
    - 利用梯度更新模型函数



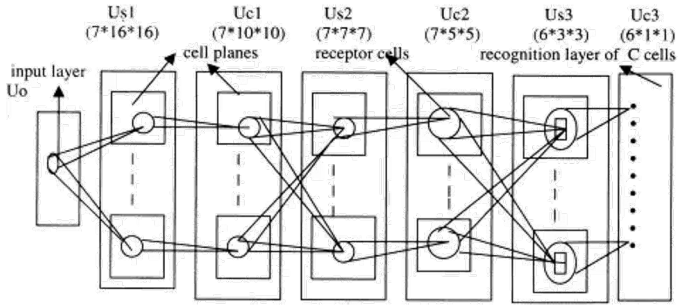
集成学习 - Bagging/Random Forest

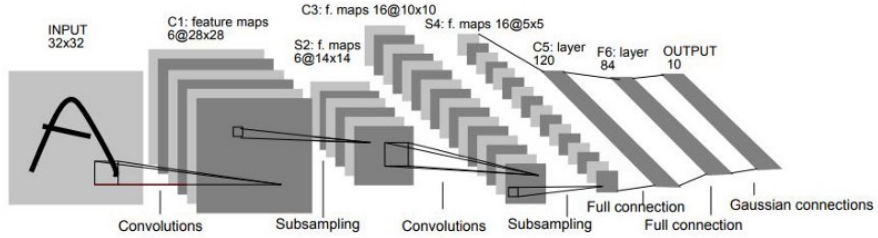
- 并行式集成学习方法
  - 基础模型之前没有依赖关系
- Bagging - Bootstrap Aggregating
  - Bootstrap sampling
    - 有放回的采样 - 有约36.8%的数据不出现在采样数据集中
    - 数据集的多样性 - 基础模型的多样性
- Random Forest
  - 决策树为基础模型
  - 训练数据的随机性(Bagging) + 决策树节点可用特征随机性

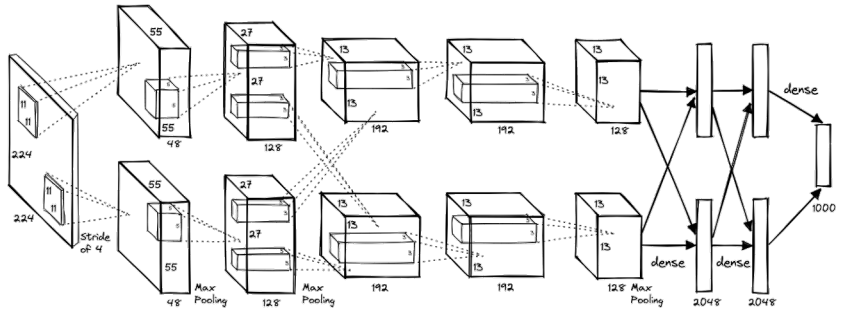
卷积神经网络CNN

卷积神经网络 - CNN

- 卷积神经网络：使用了“卷积”运算的神经网络
- 发展：
  - 1968：David Hubel和Torsten Wiesel视觉系统的研究
    - 1981年诺贝尔生理学或医学奖
  - 1980：Kunihiko Fukushima原始卷积网络的概念
  - 1998：LeNet-5，
    - MNIST手写数字识别
  - 2012：AlexNet，8层
    - ImageNet比赛获胜
  - 2015：ResNet，152层，残差
    - 首次在ImageNet分类中超越人类(3.57% vs. 5.1%)
- 特点
  - 稀疏连接、参数共享、等变表示

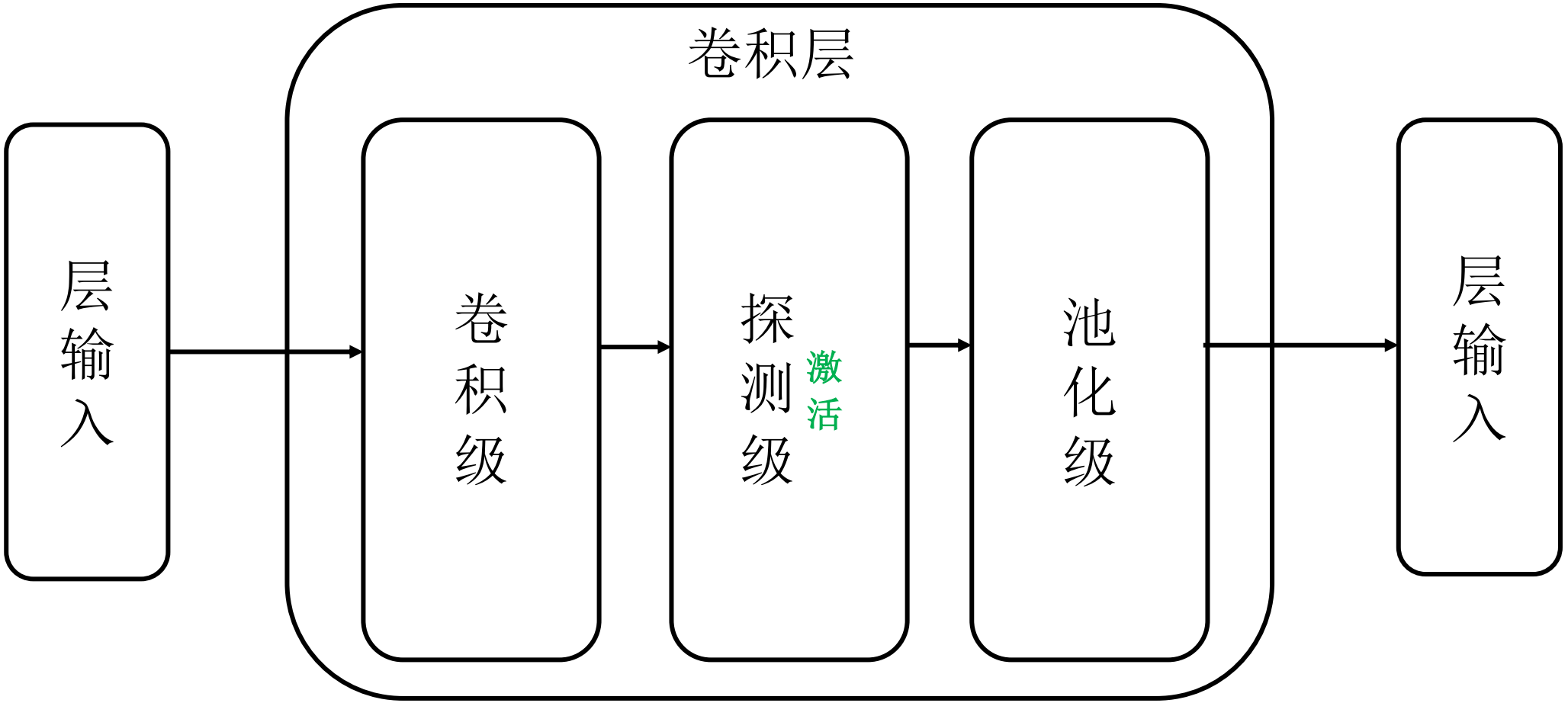






基本结构

- 卷积神经网络基本结构



卷积级

- 卷积级/层
  - 一维卷积运算

$$s(t) = \int x(a)w(t-a)da \qquad s(t) = \sum_{a} x(a)w(t-a)$$

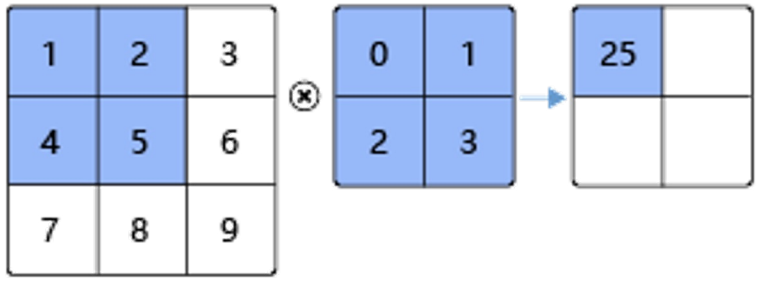
  - 神经网络中的“卷积”

$$S(i,j) = \sum_{m,n} I(i+m, j+n)K(m,n)$$

    - 也可以看作矩阵乘法 – 线性变换/仿射变换
      - 非常稀疏（很多零）
      - 参数共享（很多位置的元素是绑定的）

卷积操作

- 卷积级/层
  - 卷积核大小
    - $h_k \times w_k$



(a) 0x1+1x2+2x4+3x5=25

(b) 0x2+1x3+2x5+3x6=31

- 输出数据大小

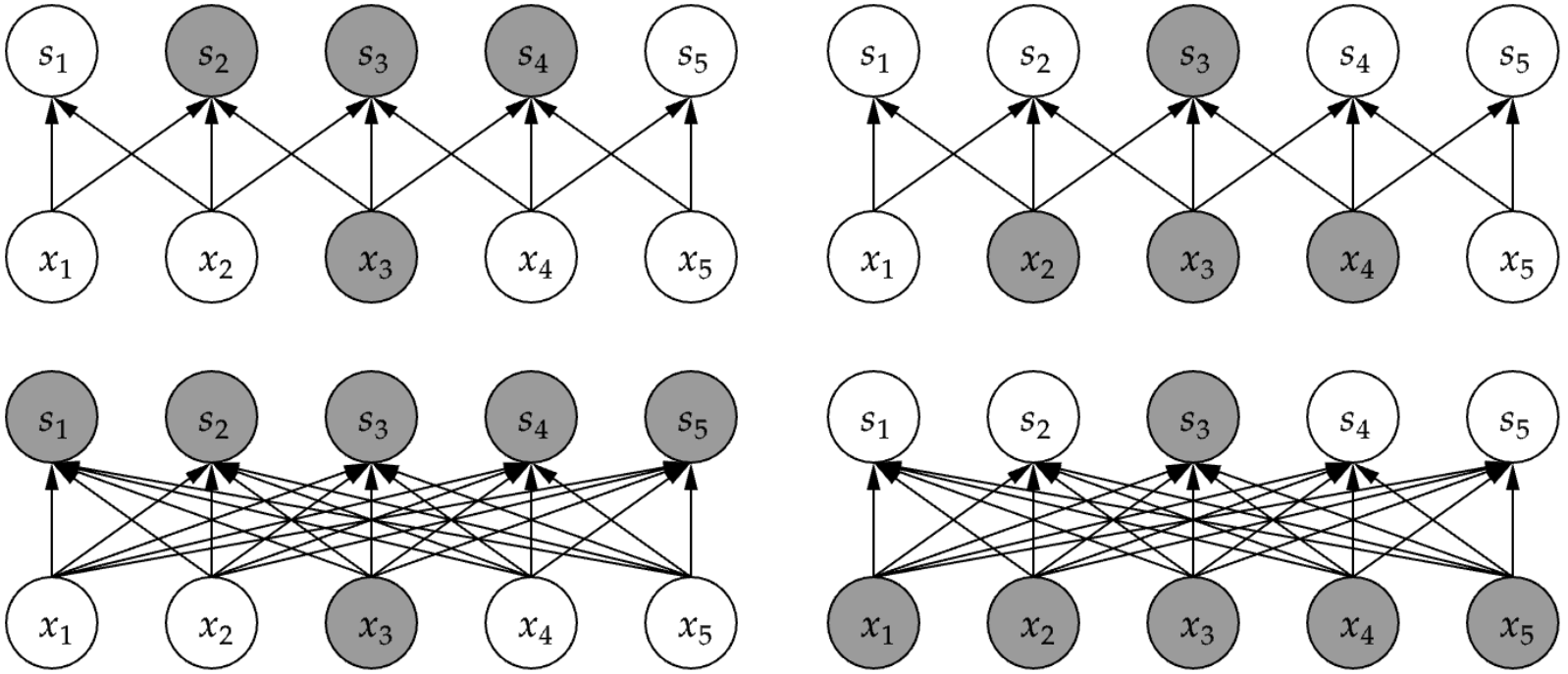
$$h_o = h_i - h_k + 1$$
$$w_o = w_i - w_k + 1$$

(c) 0x4+1x5+2x7+3x8=43

(d) 0x5+1x6+2x8+3x9=49

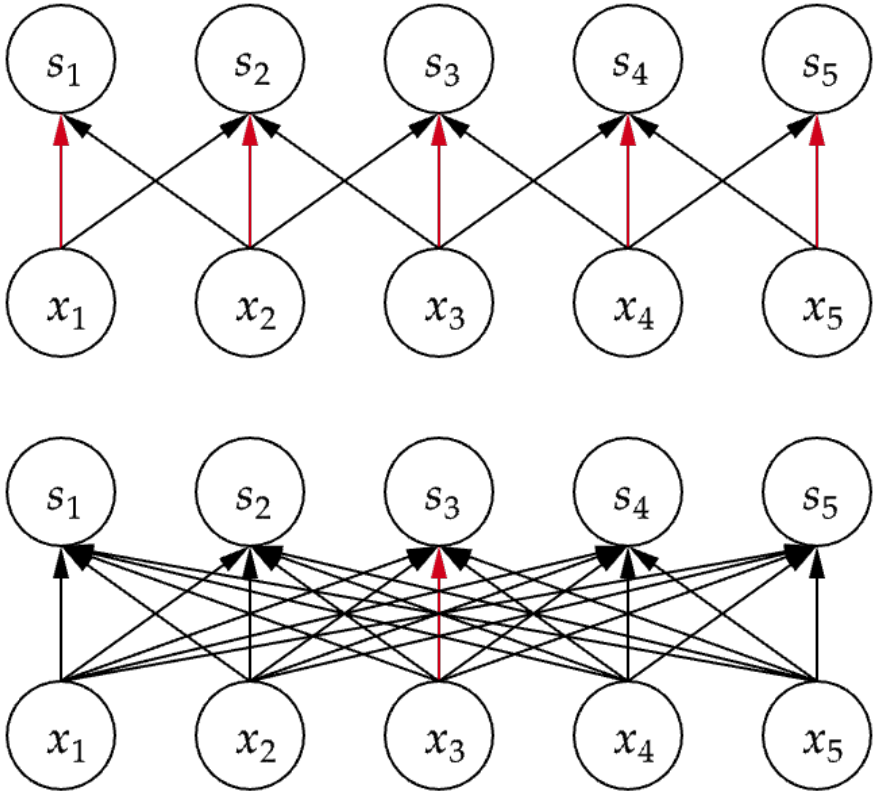
卷积 vs. 全连接

- 卷积 vs. 全连接
  - 稀疏连接，参数共享，等变表示
- 稀疏连接
  - 卷积核大小 vs. 输入/输出大小



卷积 vs. 全连接

- 参数共享
  - 参数数目：k vs. $m \times n$
- 等变表示
  - 对输入的平移 = 对输出的平移



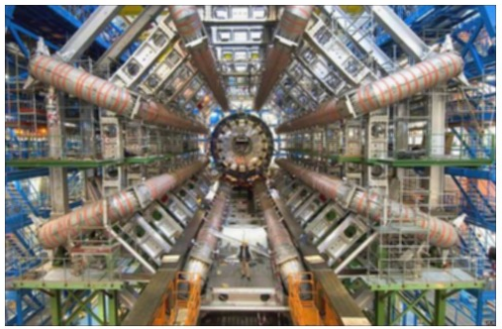
卷积核的作用

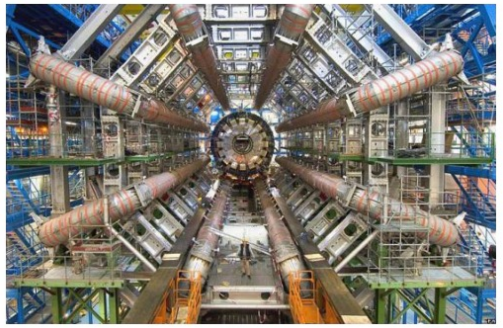
- 卷积核

  https://graphics.stanford.edu/courses/cs178/applets/convolution.html

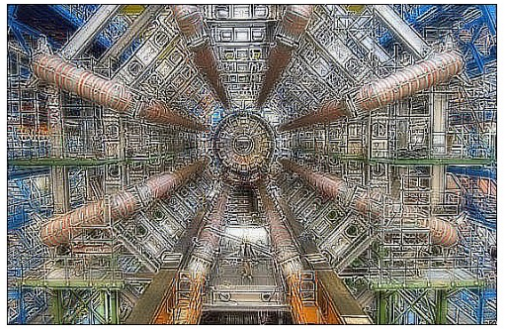
$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

高斯模糊

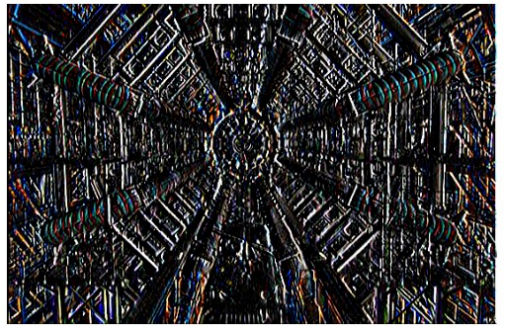




$$\begin{bmatrix}1 & 1 & 1\\ 1 & -7 & 1\\ 1 & 1 & 1\end{bmatrix}$$ 锐化

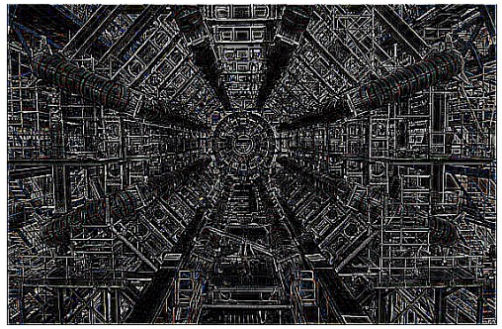


$$\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}$$



边缘检测

$$\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix}$$

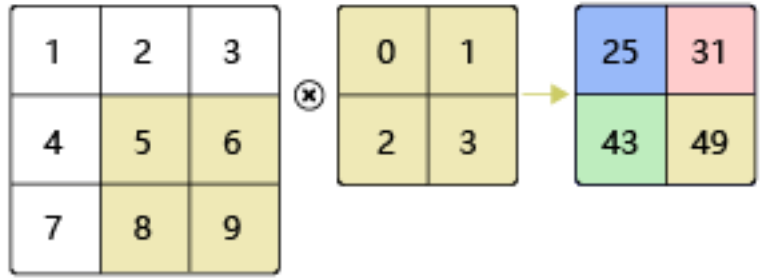


$$\begin{bmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{bmatrix}$$

填充和步幅

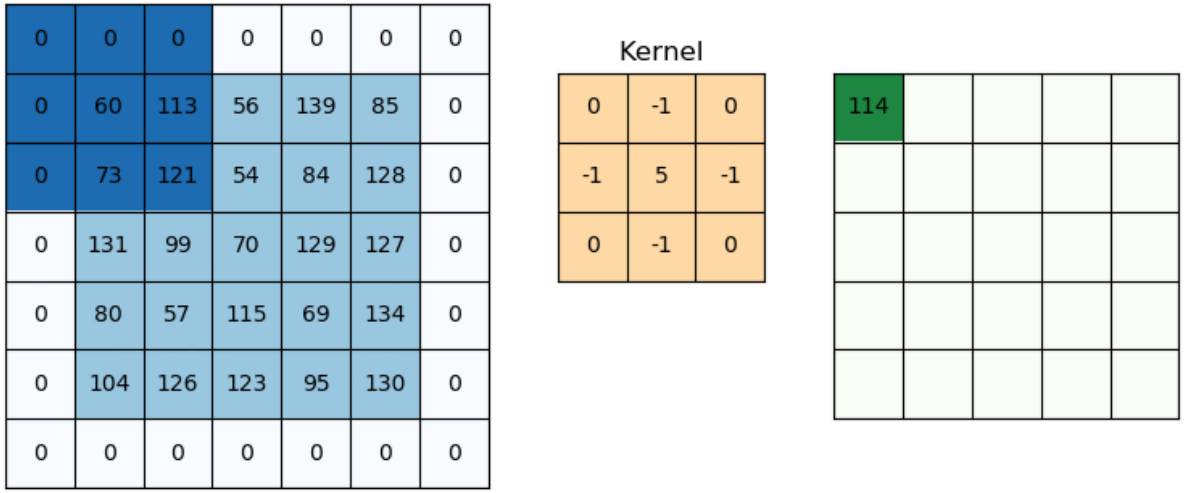
- 卷积级/层中的填充(padding)和步幅(stride)
  - 填充

$$h_o = h_i - h_k + 1$$
$$w_o = w_i - w_k + 1$$



- $h_k \geq 2, w_k \geq 2 \Rightarrow h_o < h_i, \quad w_o < w_i$
  - 正常情况，输出图片会逐渐变小
- 在输入图片周围填充像素

$$h_o = h_i + h_p - h_k + 1$$
$$w_o = w_i + w_p - w_k + 1$$

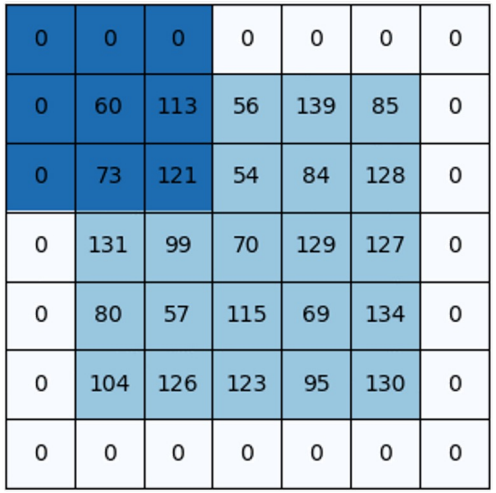


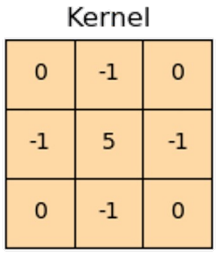
- 一般选择 $h_p = h_k - 1,$
  $w_p = w_k - 1,$
  - h_k, w_k 为奇数

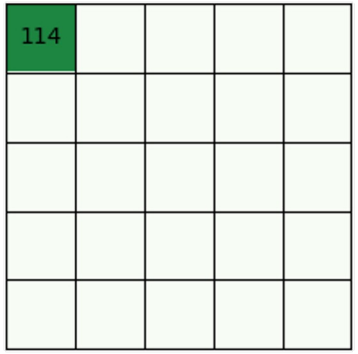
填充和步幅

- 卷积级/层中的填充(padding)和步幅(stride)
  - 步幅
    - 卷积核每次移动的行数和列数

$$h_s = w_s = 1$$



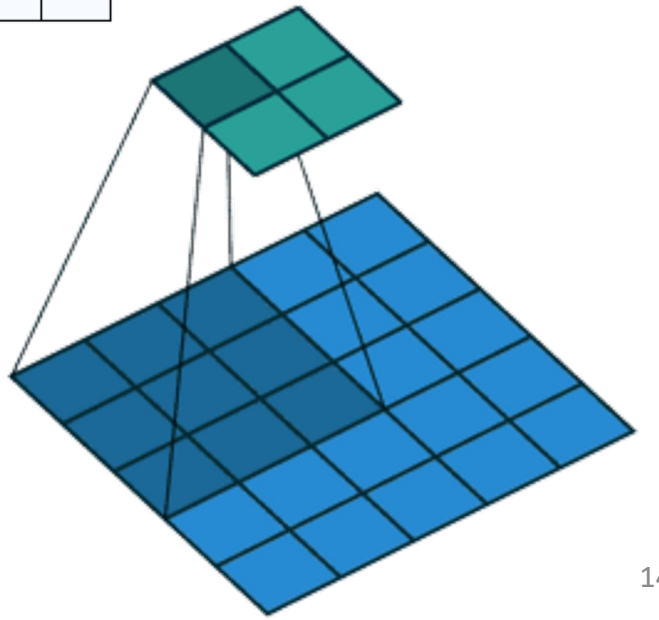




$$h_o = \left\lfloor \frac{h_i + h_p + h_s - h_k}{h_s} \right\rfloor$$

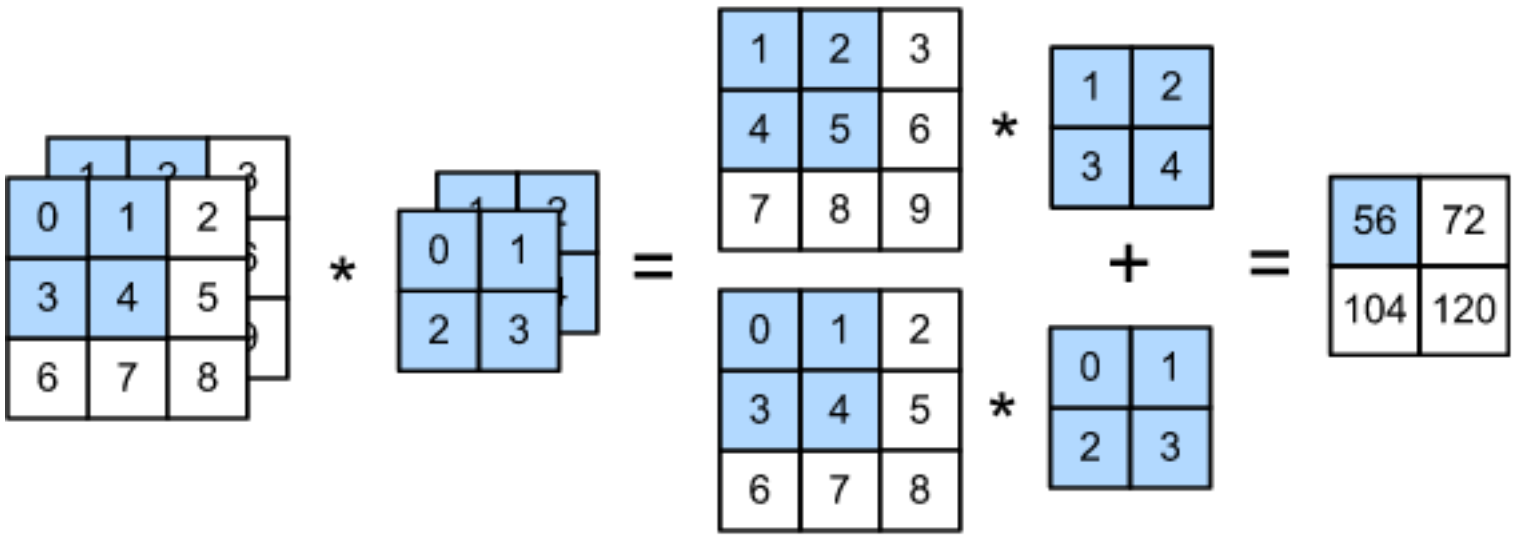
$$w_o = \left\lfloor \frac{w_i + w_p + w_s - w_k}{w_s} \right\rfloor$$

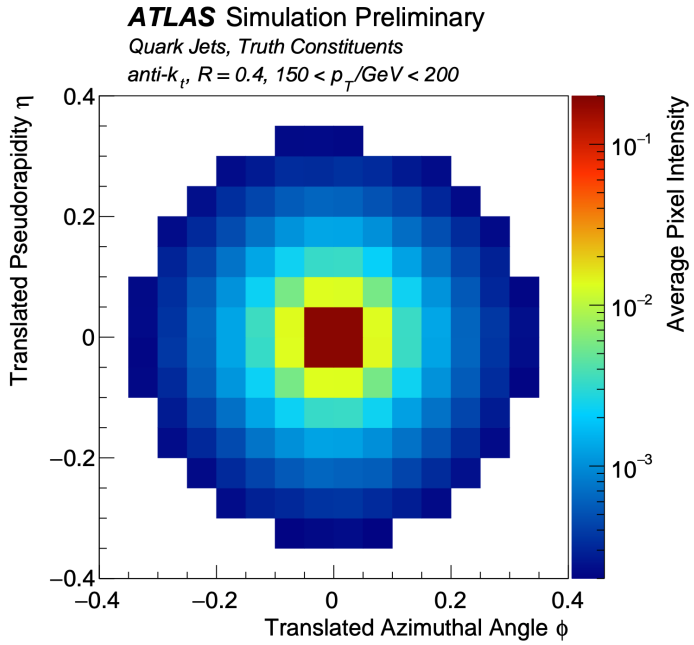
$$h_s = w_s = 2$$



多通道输入输出

- 卷积级/层 - 多输入/输出通道（channel）
  - 一张图片除了宽/高之外 - 还有RGB三个颜色通道
  - 即使图片一开始只有一个通道，也可以人为制造多通道
- 多输入通道
  - 输入通道数 = 每组卷积核数目

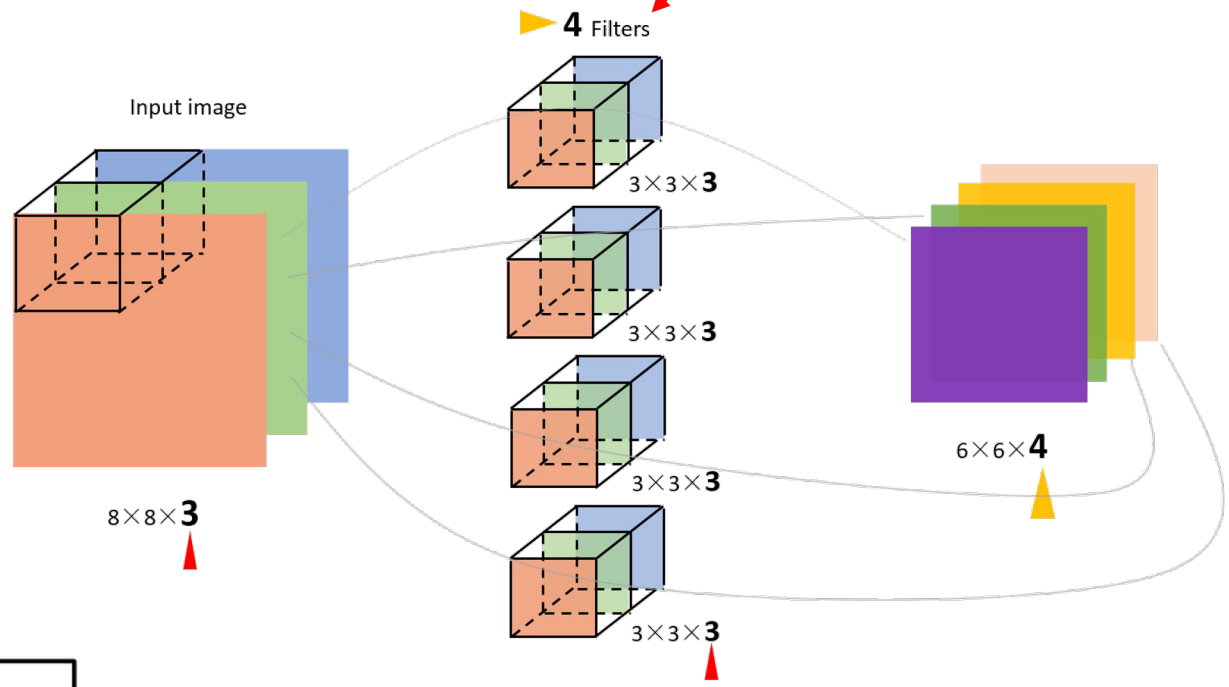




多通道输入输出

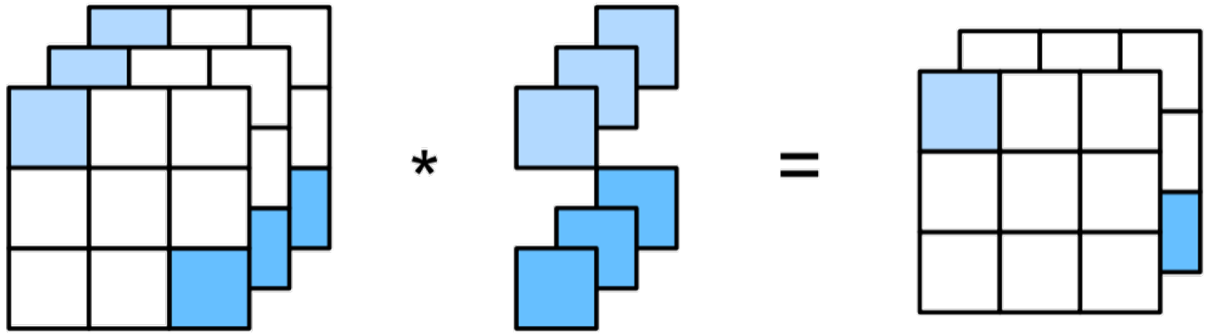
- 多通道输出
  - 输出通道数 = 卷积核的组数

用于探测输入数据中不同的特征



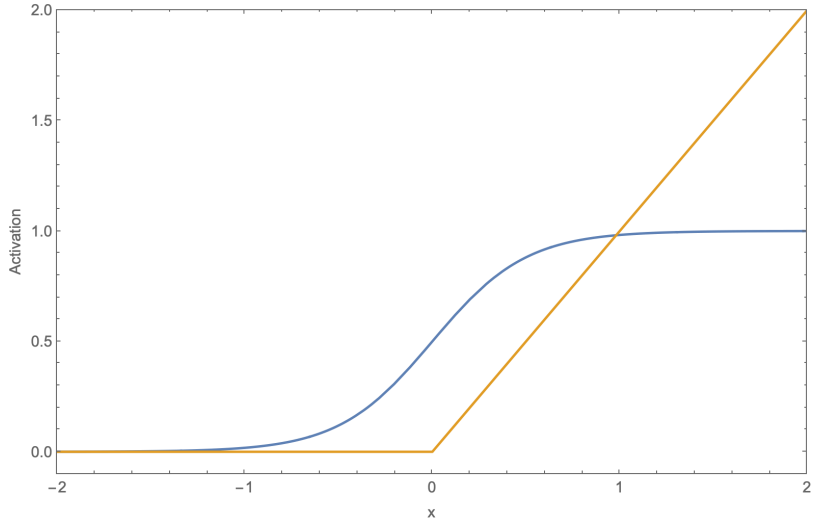
- 1×1卷积
  - 等价为保持宽高的全连接层
  - 调整通道数目

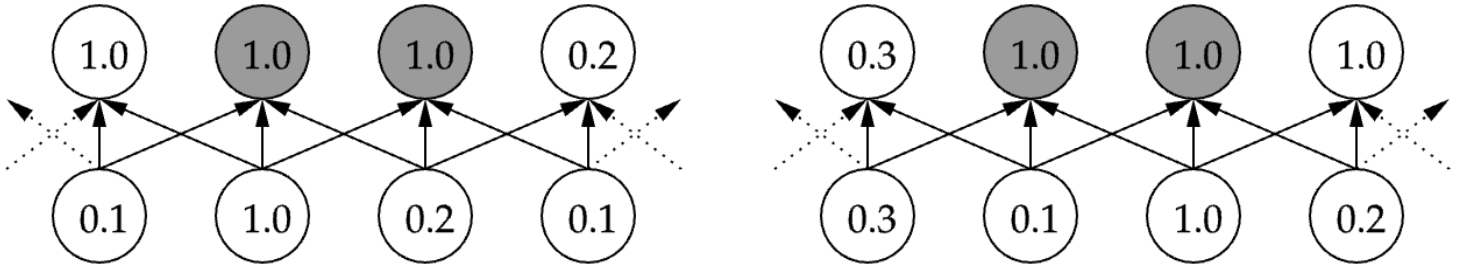
3通道 → 2通道



激活和池化

- 探测级 - 激活函数 - 非线性
  - LeNet：tanh
  - AlexNet：ReLU
    - 计算更简单
    - 有利于避免梯度消失
- 池化级(pooling)
  - 相邻输出的总体统计特征代替相应位置的输出
    - 最大值、平均值、加权平均……
  - 近似局部平移不变
    - 更关心特征是否出现，不关心特征出现在什么位置





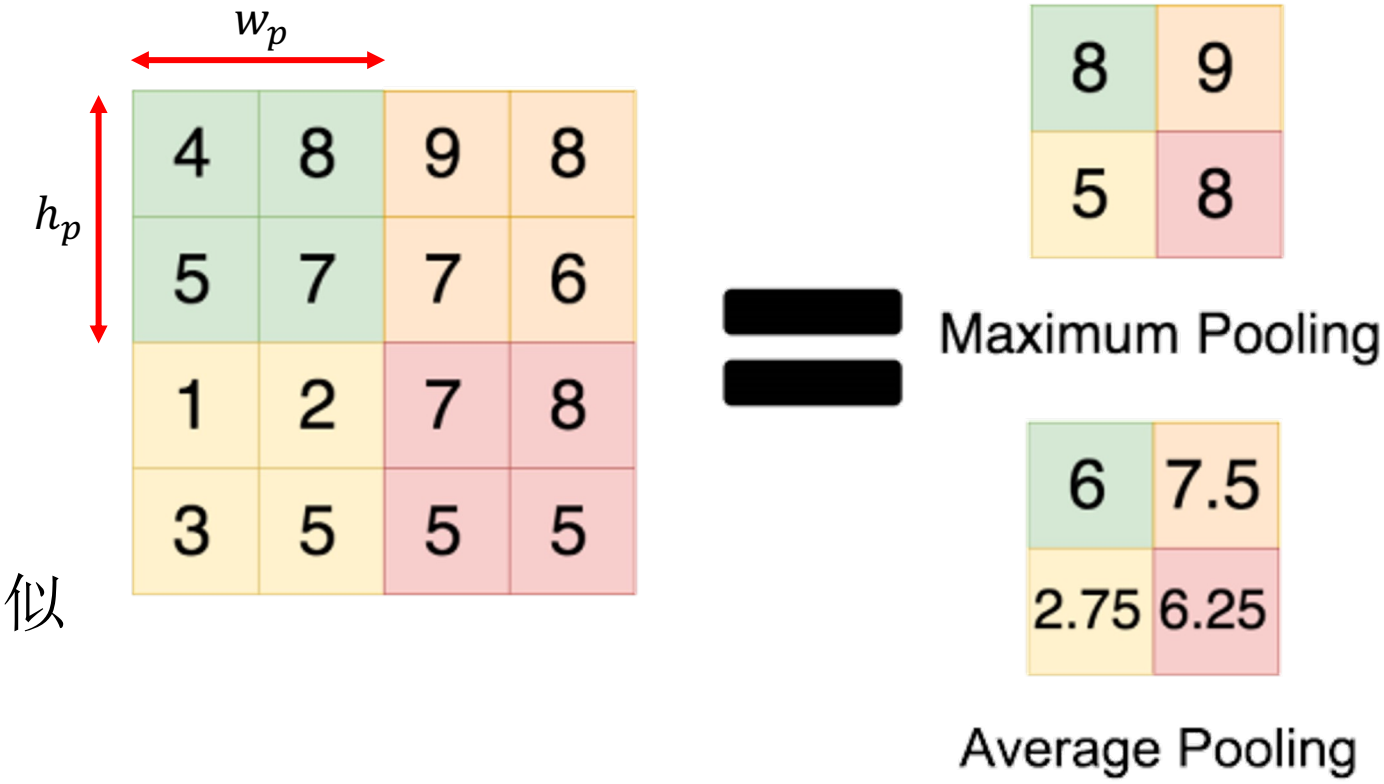
池化

- 池化操作
  - 最大池化 vs. 平均池化
  - 池化窗口大小

$$h_p \times w_p$$

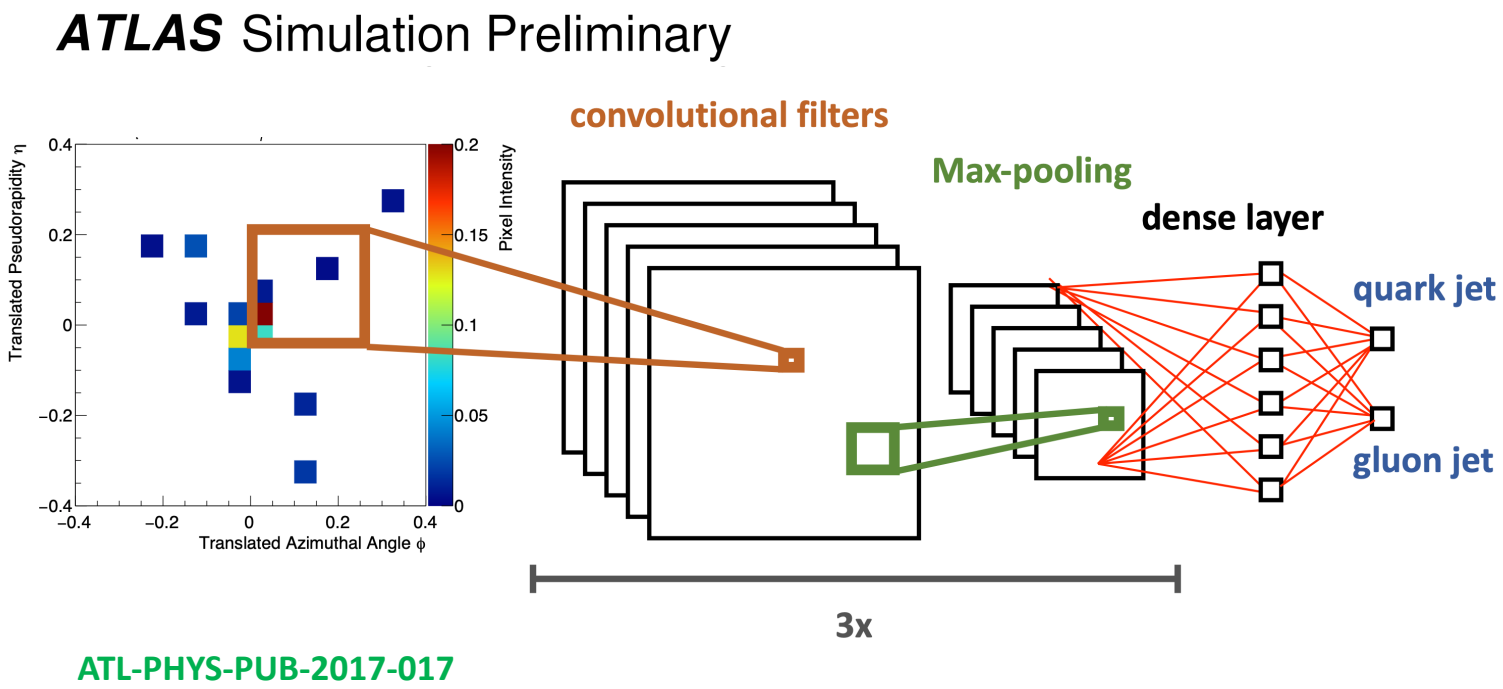
- 池化的填充和步幅
  - 同卷积类似
  - 输入、输出之间维数关系也类似

- 多通道
  - 池化过程中：输出通道 = 输入通道

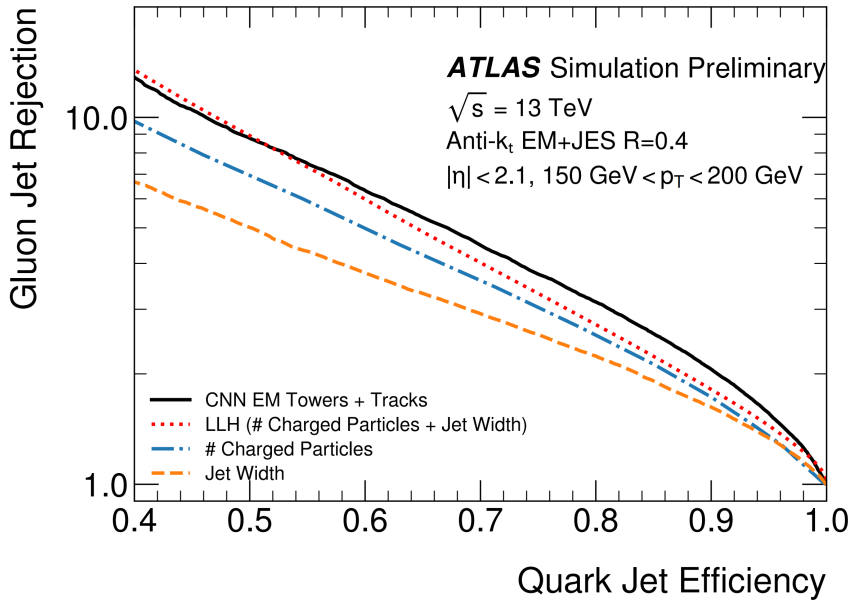


卷积神经网络

- Quark-gluon jet tagging



ATL-PHYS-PUB-2017-017

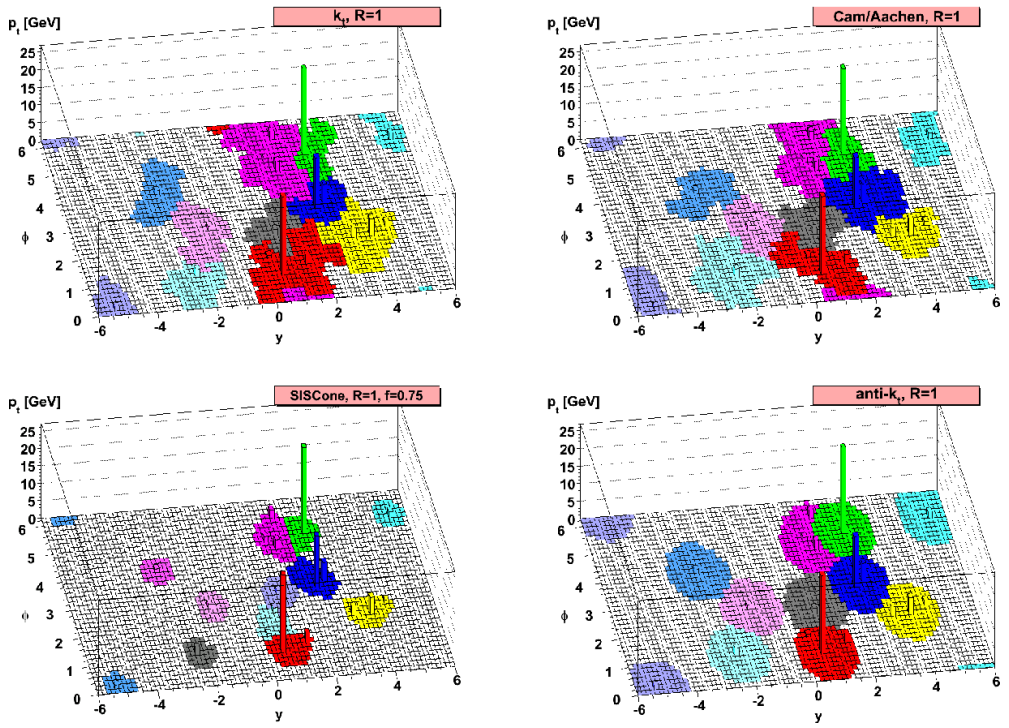


聚类-Clustering

聚类

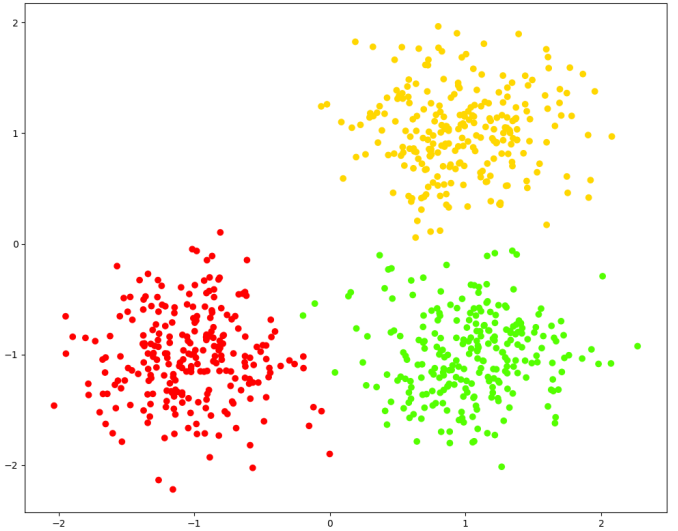
- 无监督学习 – 样本无标签信息
  - 通过对样本的学习来揭示数据的一些内在结构/性质
- 聚类：
  - 数据集$D = \{\vec{x}_1, \cdots, \vec{x}_m\}$ – 无标签的样本
  - 将数据集划分为k个互不相交的“类/簇”
    - $\{C_1, \cdots, C_k\}$, $C_i \cap_{i \neq j} C_j = \emptyset$
  - 每个样本只属于一个类：$\vec{x}_i \in C_{\lambda_i}$
    - 类标记：$\lambda_i \in \{1, \cdots, k\}$
    - 聚类结果给出数据样本的类标记：$\{\lambda_1, \cdots, \lambda_m\}$
- 性能度量：
  - 分类是好是坏？
    - 对结果的评估
    - 聚类过程的优化目标

Different Jet clustering algorithm



聚类性能度量

- 物以类聚，人以群分
  - 类内部“相似度” – 越高越好
  - 类之间“相似度” – 越低越好
- 考虑聚类的结果：$\mathcal{C} = \{C_1, \cdots, C_k\}$



$$\bar{d}(C_\ell) = \frac{2}{|C_\ell|(|C_\ell|-1)} \sum_{1 \le i < j \le |C_\ell|} dist(\vec{x}_i, \vec{x}_j)$$

C_ℓ内样本间的平均距离：越小越好

$$d_{max}(C_\ell) = \max_{1 \le i < j \le |C_\ell|} dist(\vec{x}_i, \vec{x}_j)$$

C_ℓ内样本间最大距离：越小越好

$$d_{min}(C_\ell, C_p) = \min_{\vec{x}_i \in C_\ell, \vec{x}_j \in C_p} dist(\vec{x}_i, \vec{x}_j)$$

C_ℓ, C_p两类样本间最小距离：越大越好

$$d_c(C_\ell, C_p) = dist(\vec{\mu}_\ell, \vec{\mu}_p)$$

C_ℓ, C_p两个类中心的距离：越大越好

类中心：$\vec{\mu}_\ell = \frac{1}{|C_\ell|} \sum_{\vec{x}_i \in C_\ell} \vec{x}_i$

聚类性能度量

- 考虑聚类的结果：$\mathcal{C} = \{C_1, \cdots, C_k\}$

$$\bar{d}(C_\ell) = \frac{2}{|C_\ell|(|C_\ell|-1)} \sum_{1 \le i < j \le |C_\ell|} dist(\vec{x}_i, \vec{x}_j)$$

C_ℓ内样本间的平均距离：越小越好

$$d_{max}(C_\ell) = \max_{1 \le i < j \le |C_\ell|} dist(\vec{x}_i, \vec{x}_j)$$

C_ℓ内样本间最大距离：越小越好

$$d_{min}(C_\ell, C_p) = \min_{\vec{x}_i \in C_\ell, \vec{x}_j \in C_p} dist(\vec{x}_i, \vec{x}_j)$$

C_ℓ, C_p两类样本间最小距离：越大越好

$$d_c(C_\ell, C_p) = dist(\vec{\mu}_\ell, \vec{\mu}_p)$$

C_ℓ, C_p两个类中心的距离：越大越好

- 性能度量指标

$$\text{DBI} = \frac{1}{k} \sum_{\ell=1}^{k} \max_{p \ne \ell} \left(\frac{\bar{d}(C_\ell) + \bar{d}(C_p)}{d_c(C_\ell, C_p)} \right)$$

平均聚集度：越小越好

$$\text{DI} = \min_{\ell} \left\{ \min_{p} \left(\frac{d_{min}(C_\ell, C_p)}{\max_q d_{max}(C_q)} \right) \right\}$$

最大类内距离归一的最小类间距离：越大越好

样本间距离

- $dist(\vec{x}_i, \vec{x}_j)$：距离度量
  - 非负：$dist(\vec{x}_i, \vec{x}_j) \geq 0$
  - 同一性（只有自己和自己是零距离的）：$dist(\vec{x}_i, \vec{x}_j) = 0 \Leftrightarrow \vec{x}_i = \vec{x}_j$
  - 对称性：$dist(\vec{x}_i, \vec{x}_j) = dist(\vec{x}_j, \vec{x}_i)$
- 闵可夫斯基距离

$$dist_p(\vec{x}_i, \vec{x}_j) = \left(\sum_{\alpha=1}^{n} |x_{i\alpha} - x_{j\alpha}|^p\right)^{\frac{1}{p}}$$

  - 曼哈顿距离：$p = 1$
  - 欧氏距离：$p = 2$
  - 切比雪夫距离：$p = \infty$
    - $dist_\infty(\vec{x}_i, \vec{x}_j) = \max_\alpha |x_{i\alpha} - x_{j\alpha}|$

$p \geq 1$时，满足三角不等式

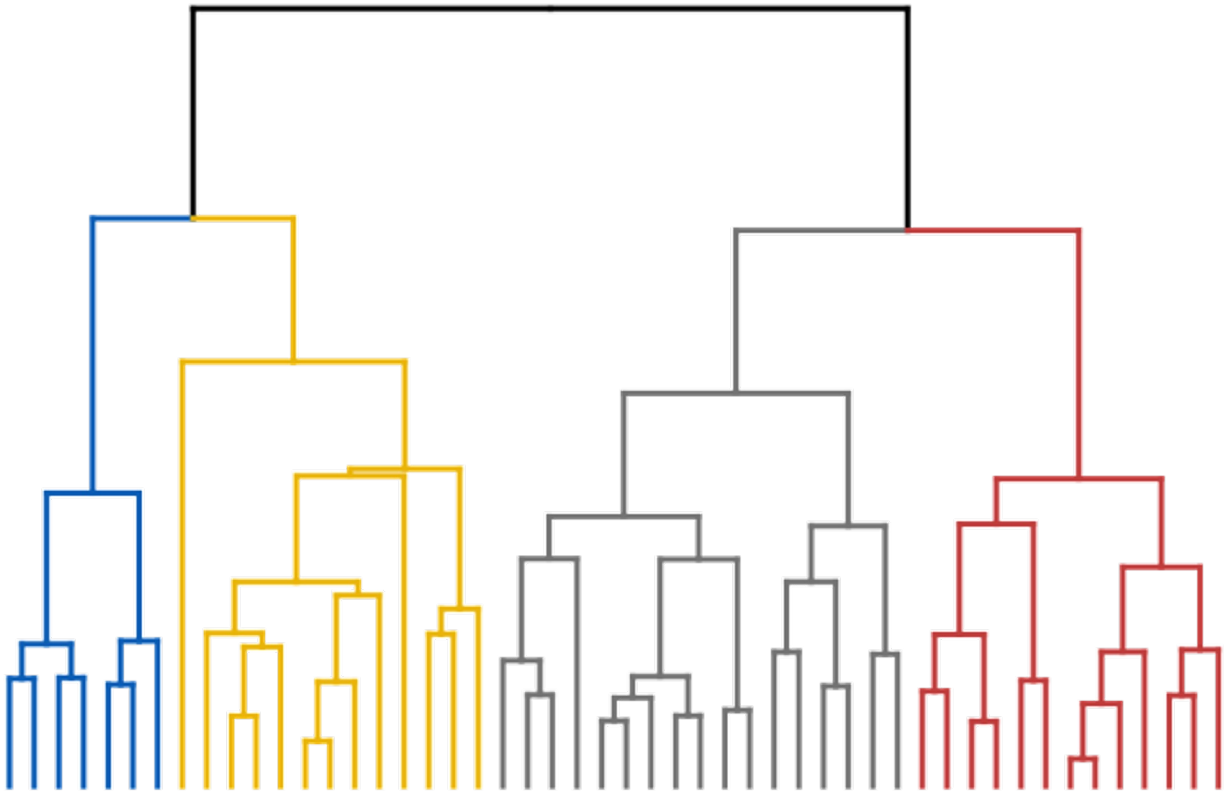
$$dist(\vec{x}_i, \vec{x}_j) \leq dist(\vec{x}_i, \vec{x}_o) + dist(\vec{x}_j, \vec{x}_o)$$

加权距离：

$$dist_{wp}(\vec{x}_i, \vec{x}_j) = \left(\sum_{a}^{n} w_a |x_{ia} - x_{ja}|^p\right)^{\frac{1}{p}}$$

层次聚类

- Hierarchy
  - 聚合(agglomerative) vs. 分裂(divisive)
- 聚合：
  - 给定数据集$D = \{\vec{x}_1, \cdots, \vec{x}_m\}$
  - 初始将每个样本分到单独一个类
  - 按照一定规则合并两个类
    - 比如距离最小
  - 直到满足停止条件
    - 类的数目达到阈值
    - 类的直径达到阈值



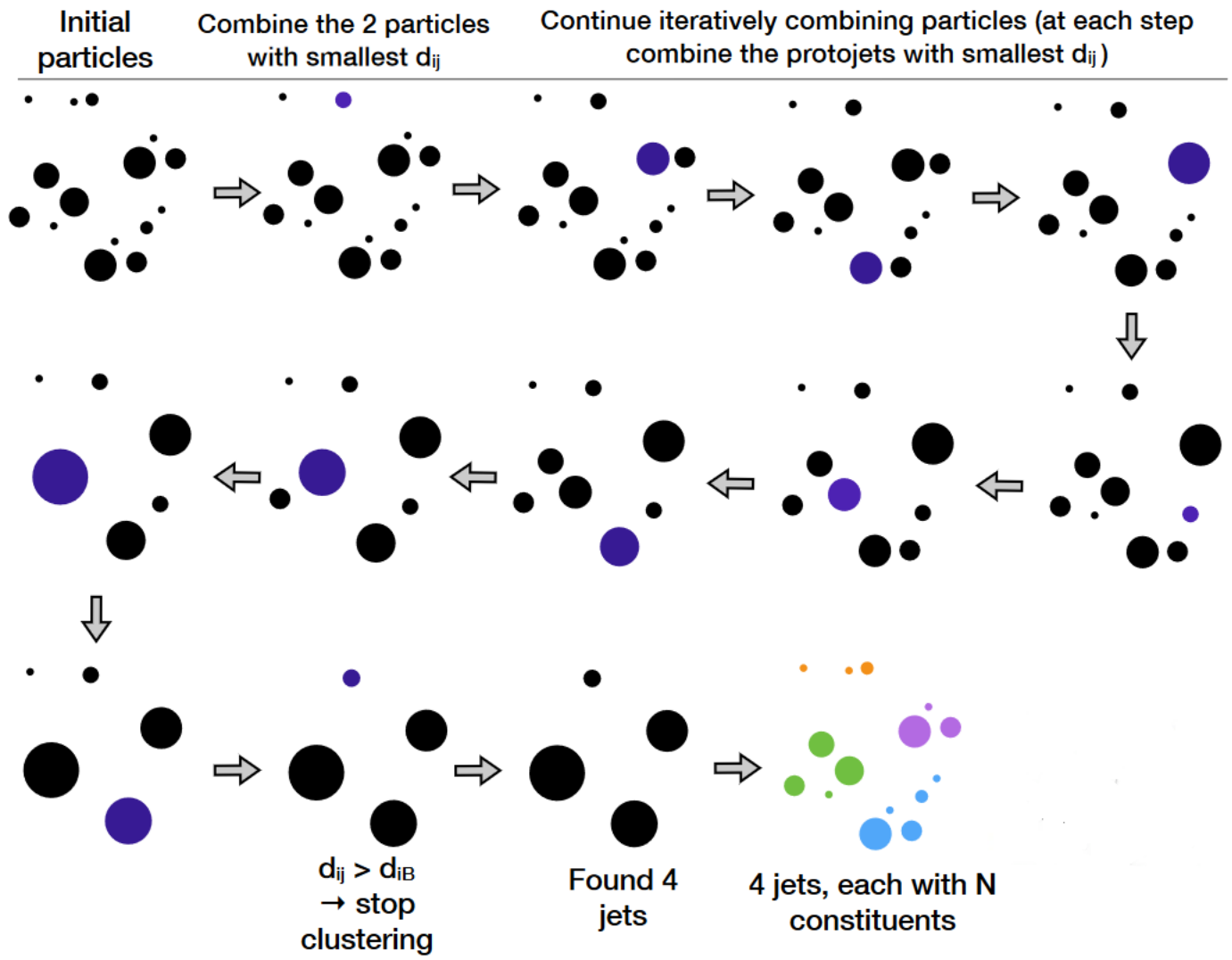
Jet Sequential Clustering Algorithm:

$$d_{ij} = \min\left(p_{Ti}^{2k}, p_{Tj}^{2k}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{2k}$$

- $k = 1$: k_t algorithm
- $k = 0$: Cambridge/Aachen algorithm
- $k = -1$: anti-k_t algorithm

层次聚类

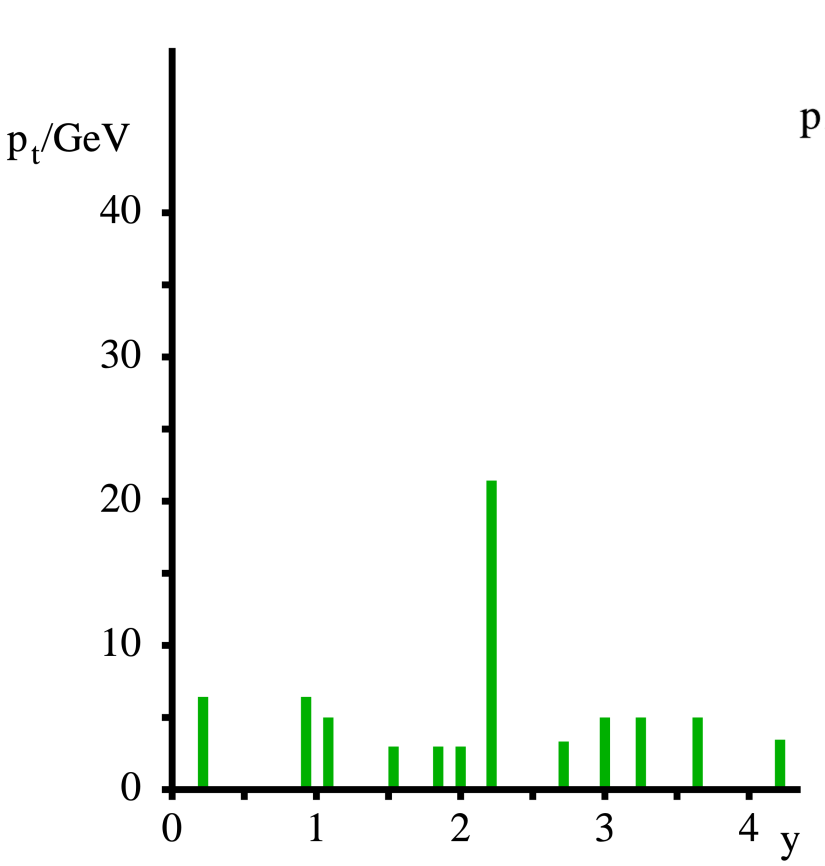
- Jet Clustering

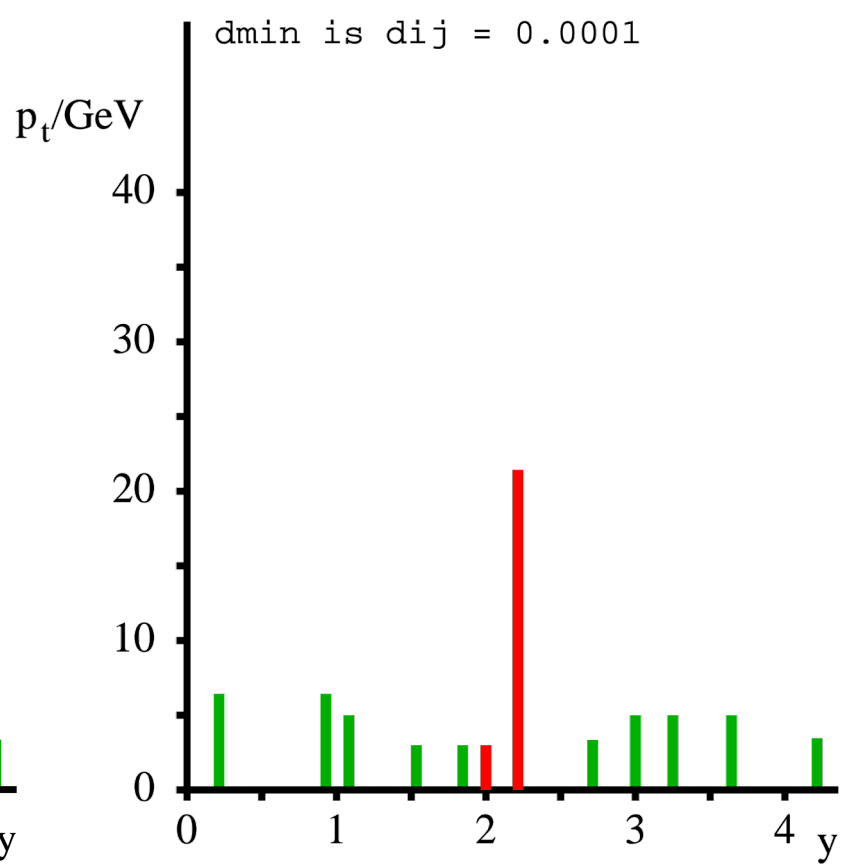


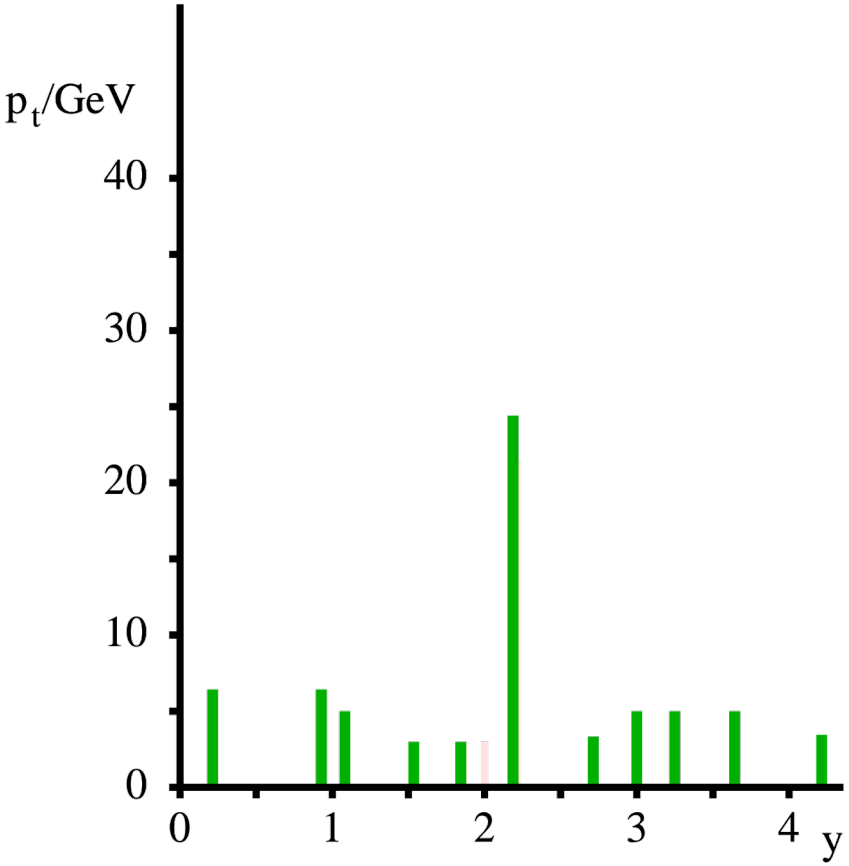
层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$





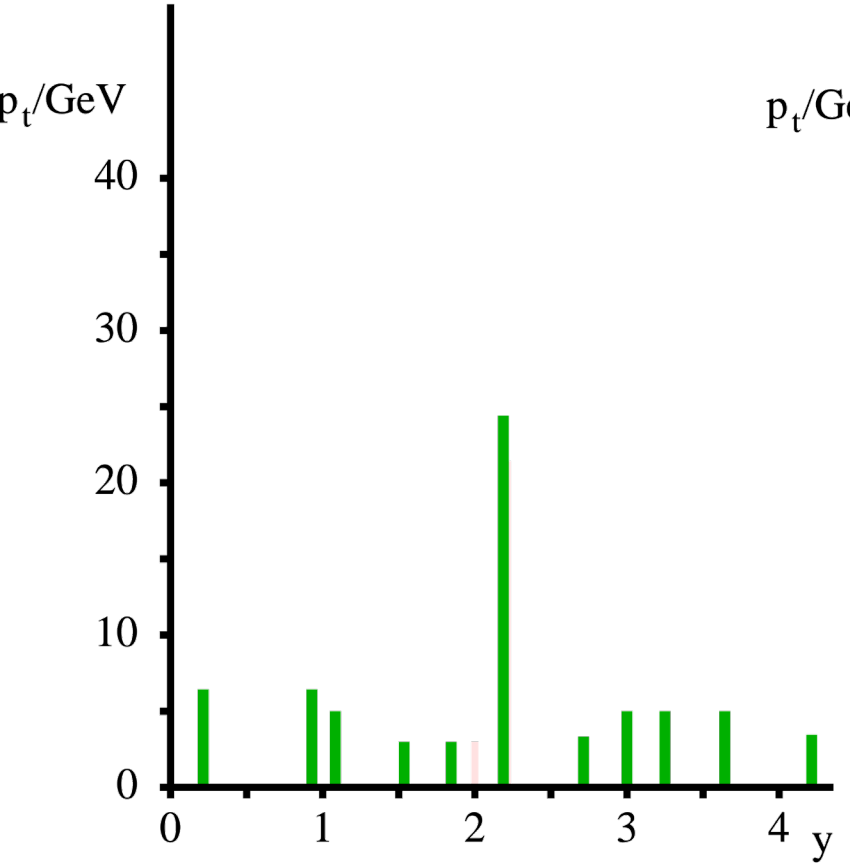


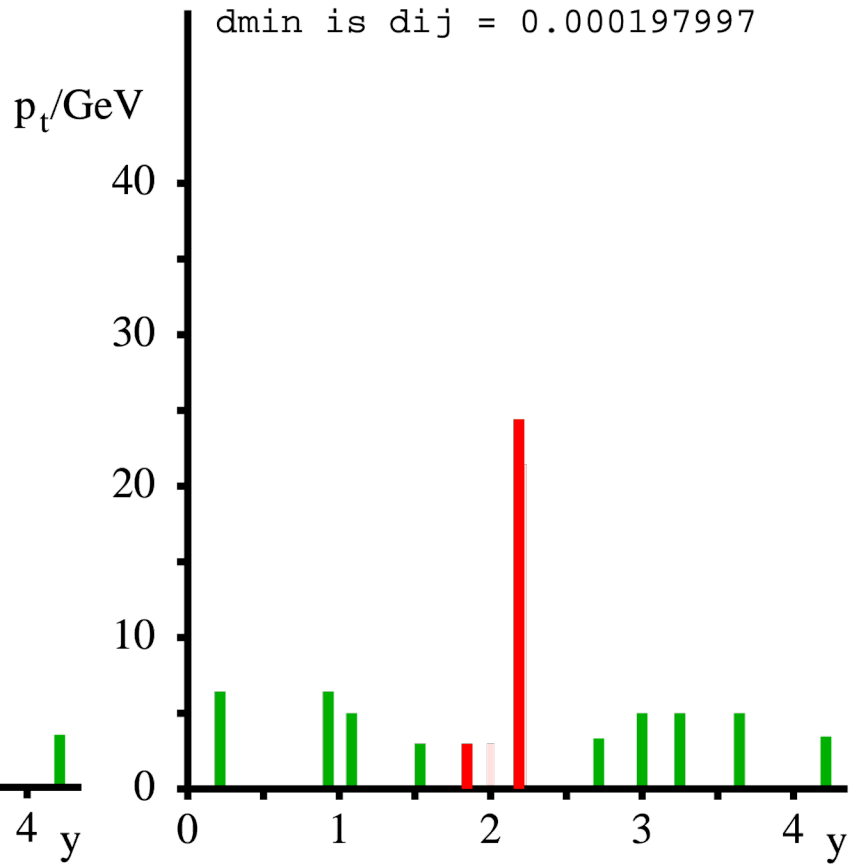
Example by Gavin Salam

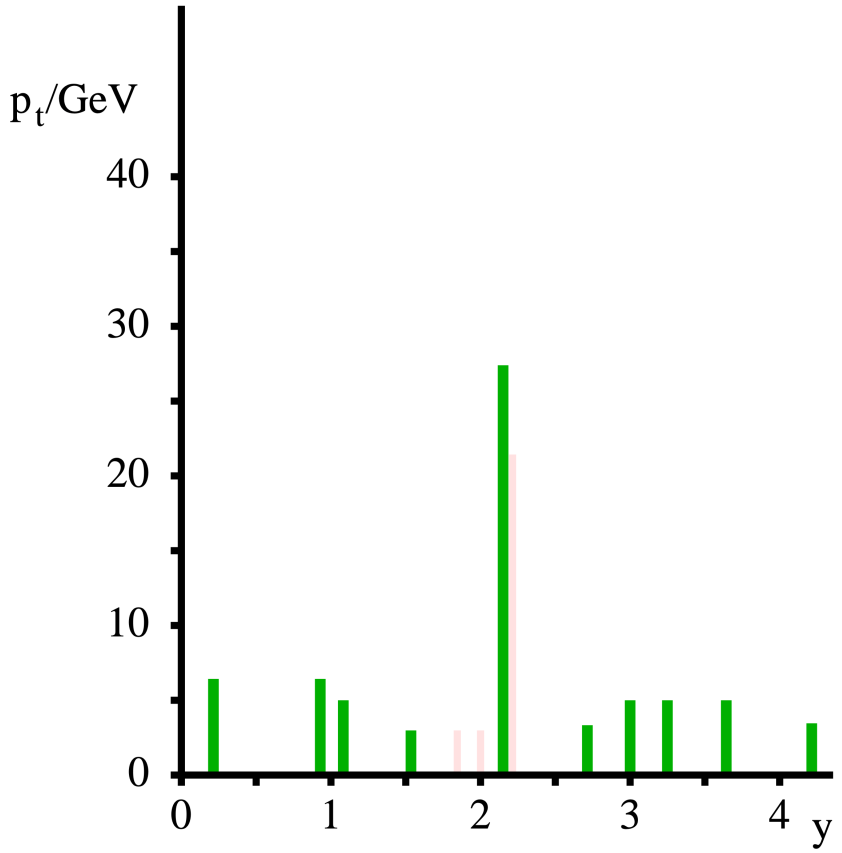
层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$





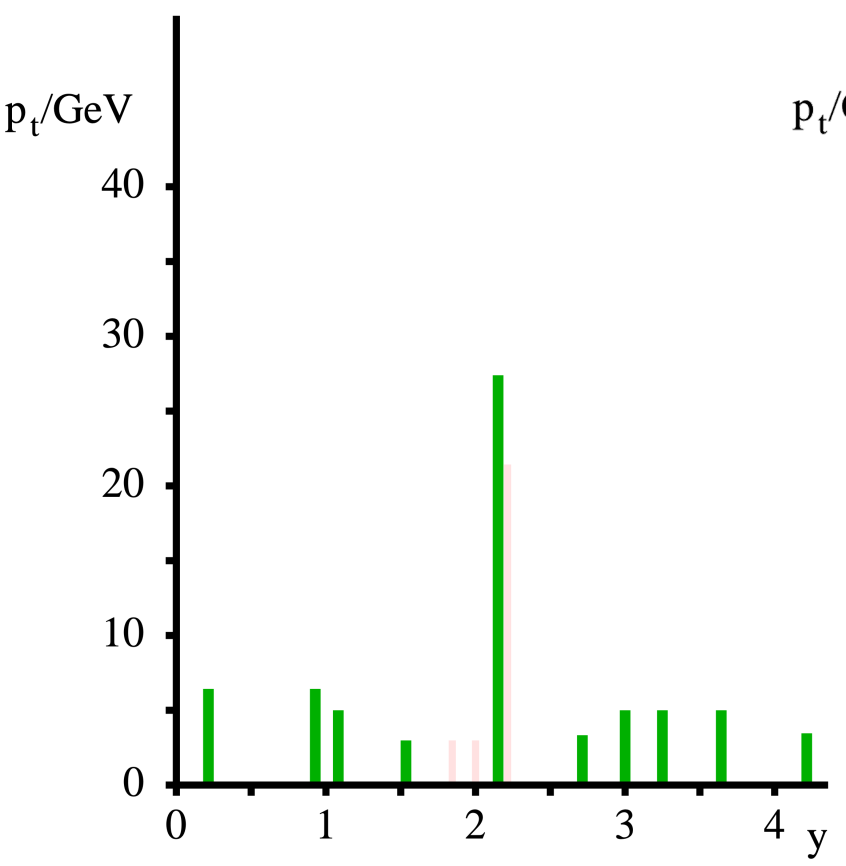


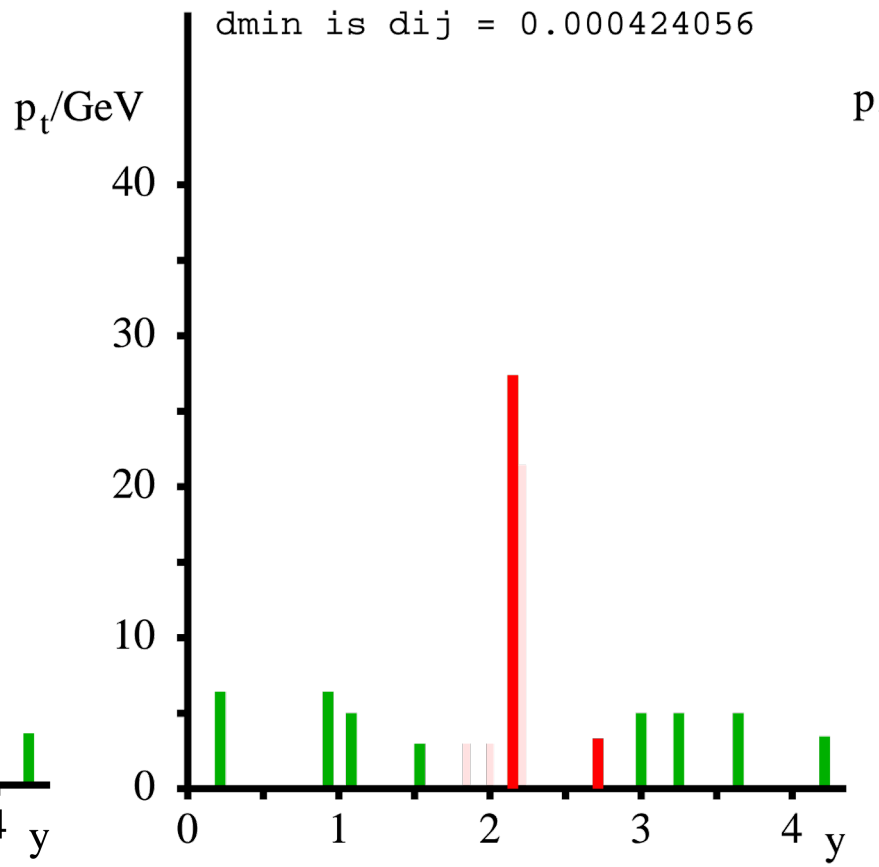
Example by Gavin Salam

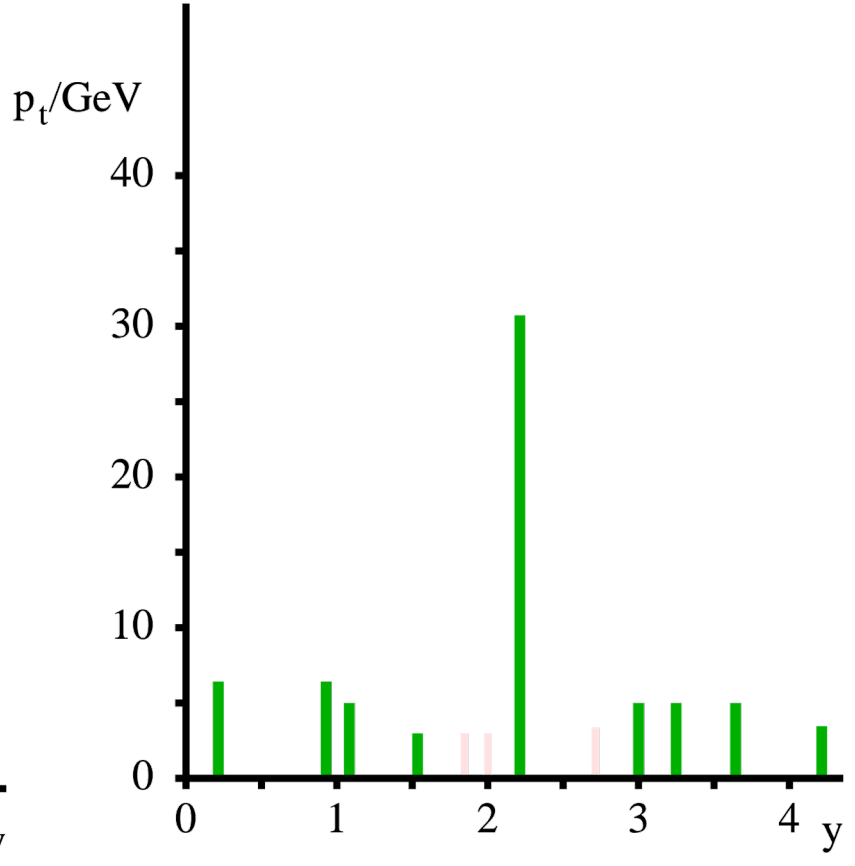
层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$





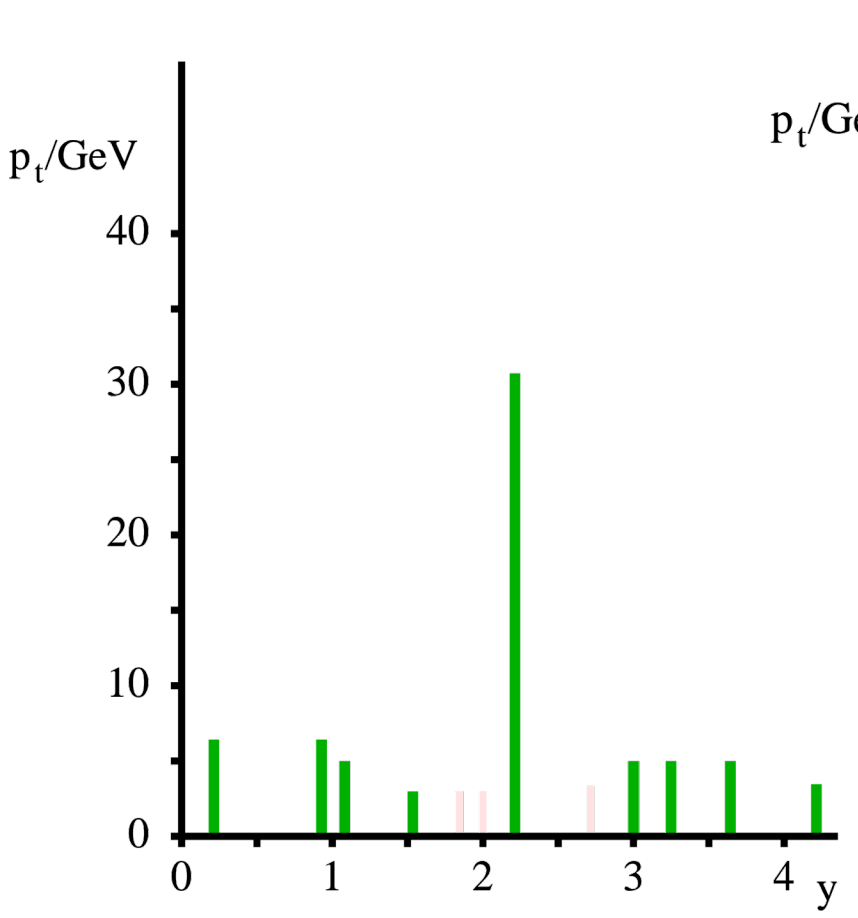


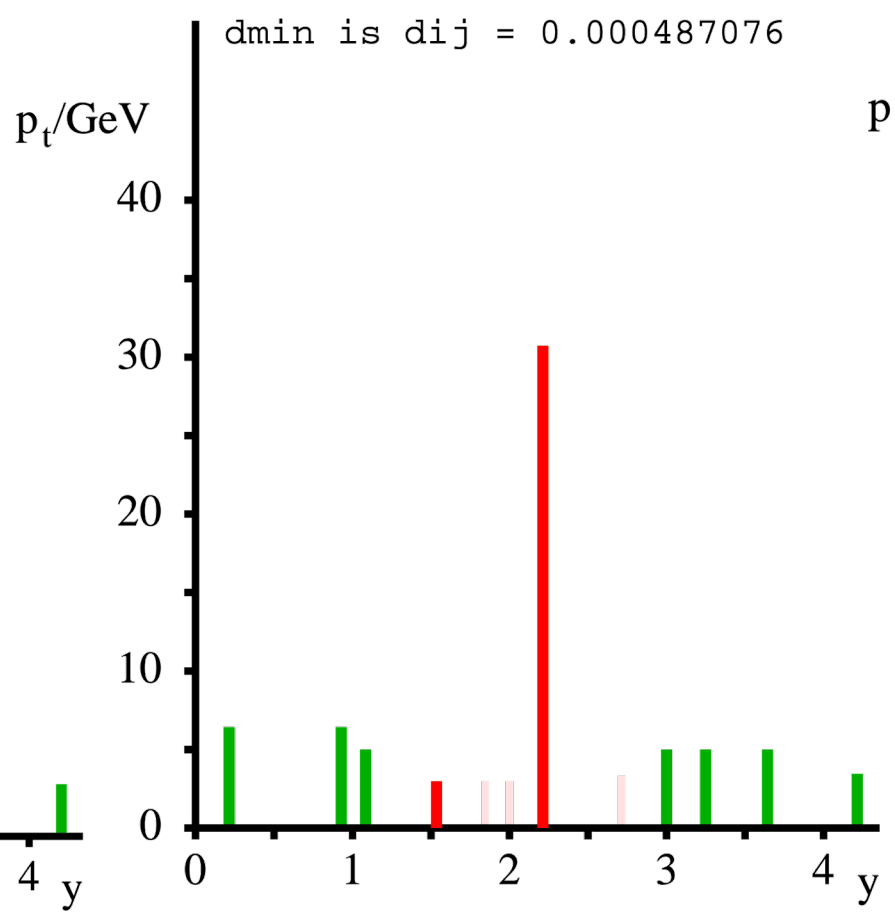
Example by Gavin Salam

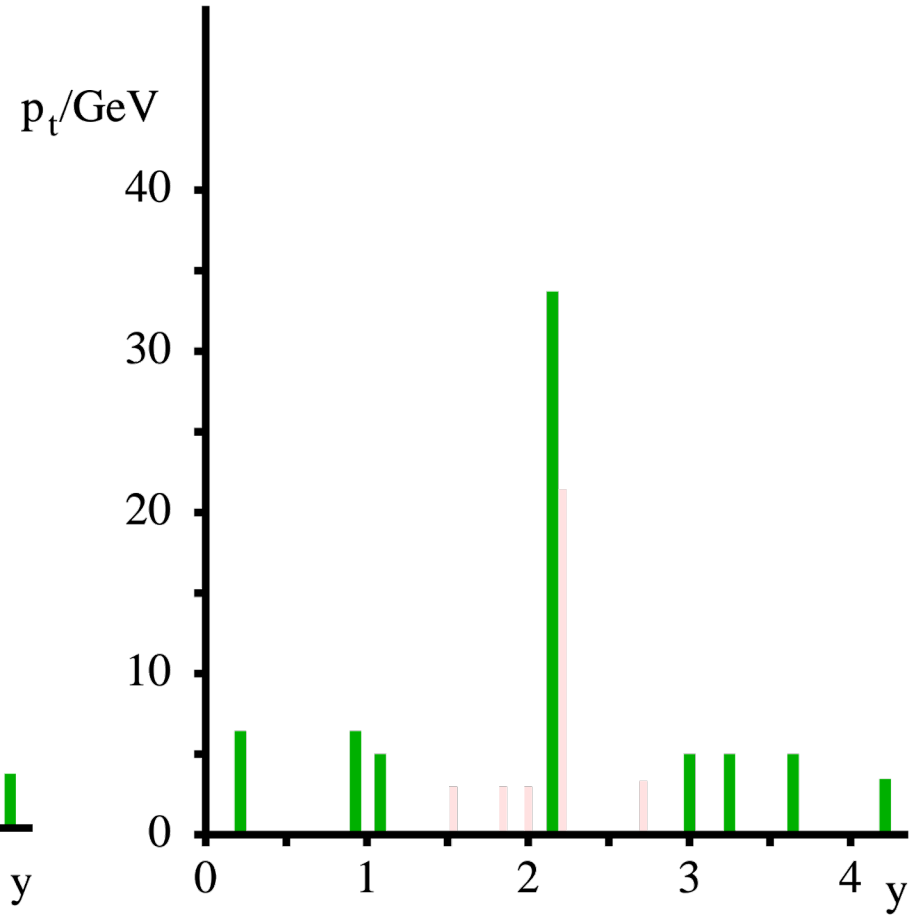
层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$





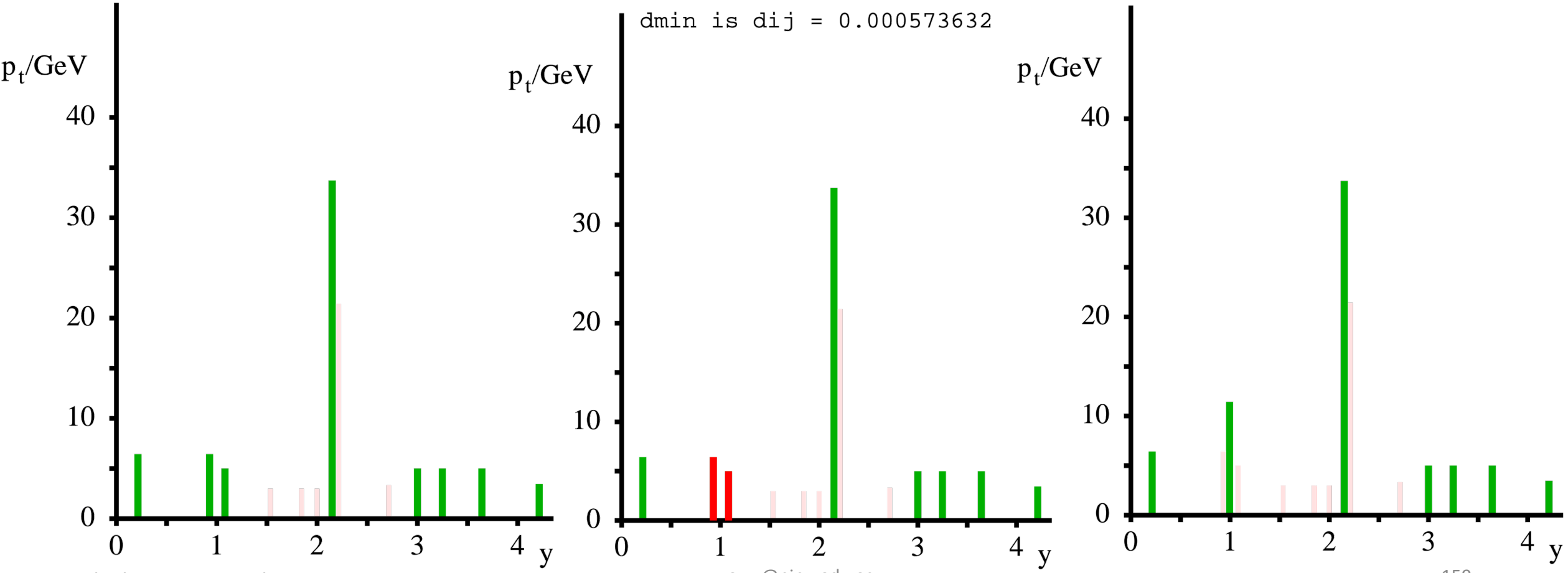


Example by Gavin Salam

层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$

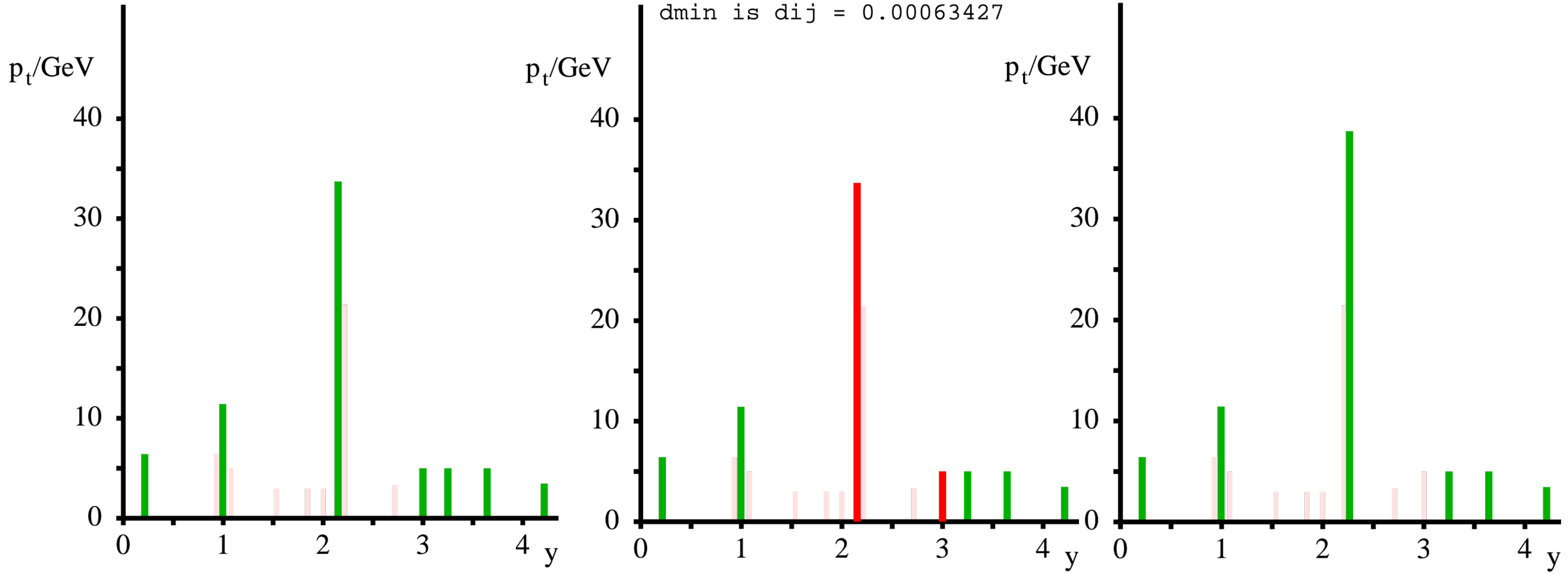


Example by Gavin Salam

层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$

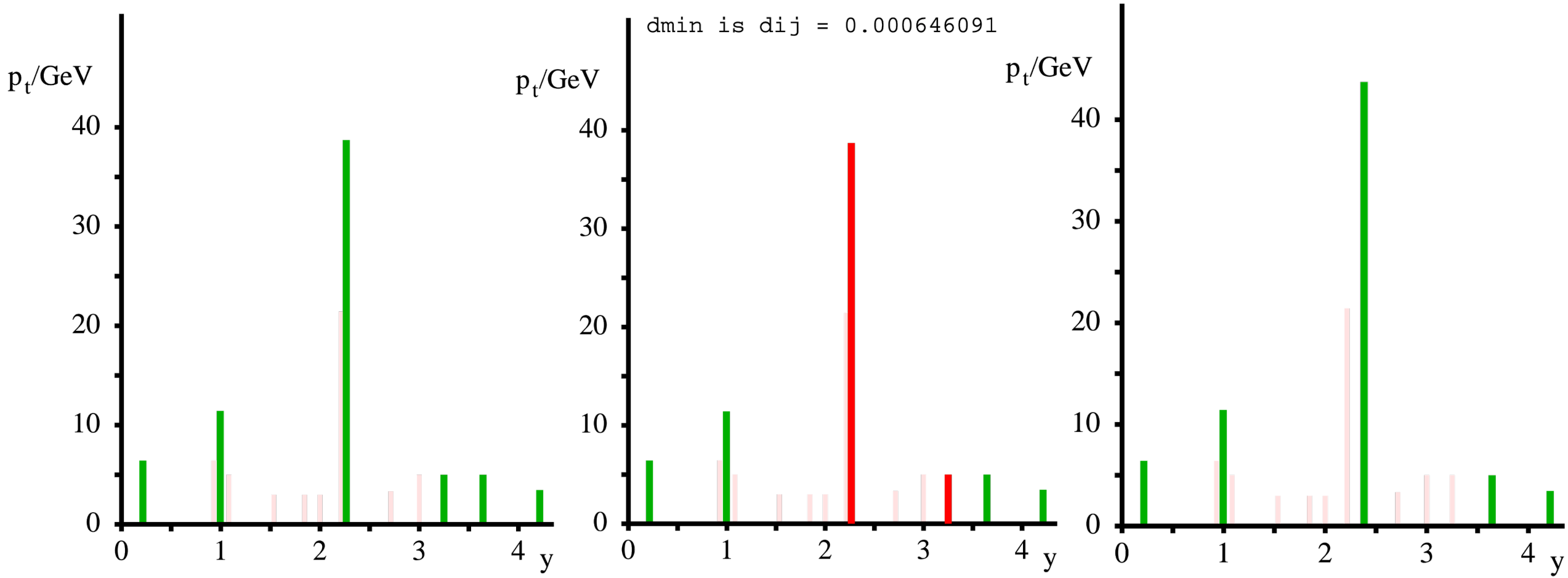


Example by Gavin Salam

层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$

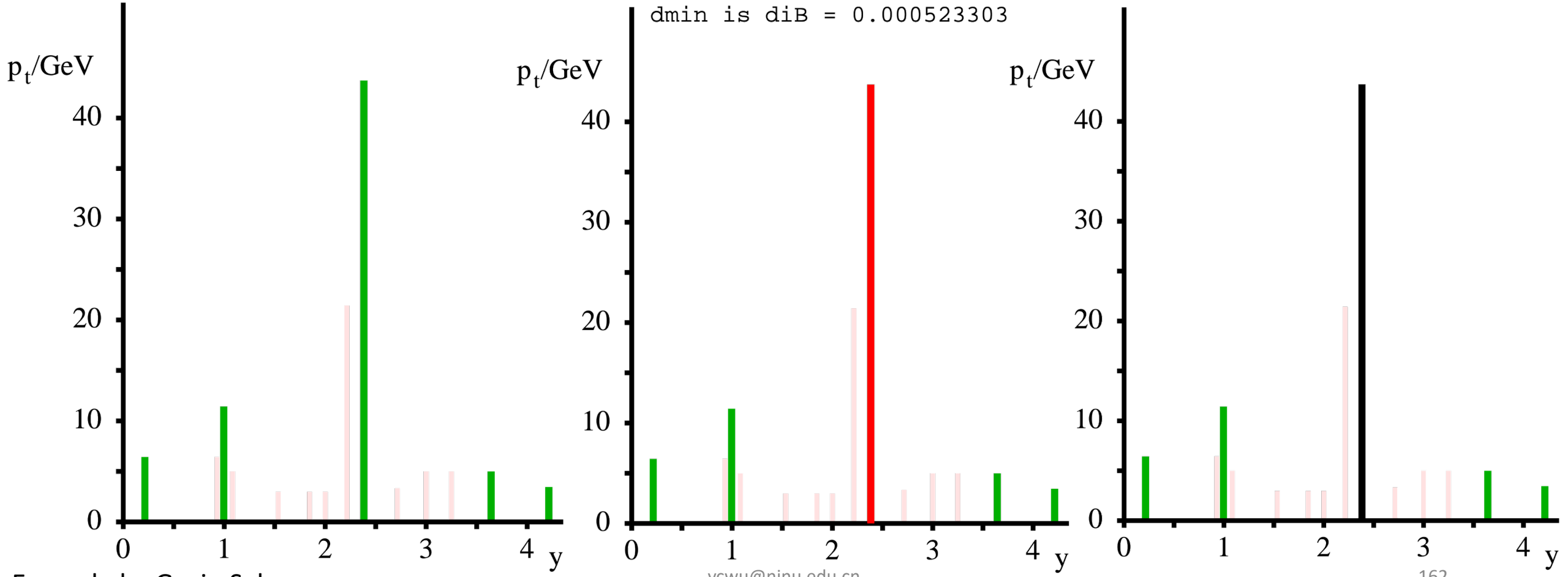


Example by Gavin Salam

层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$

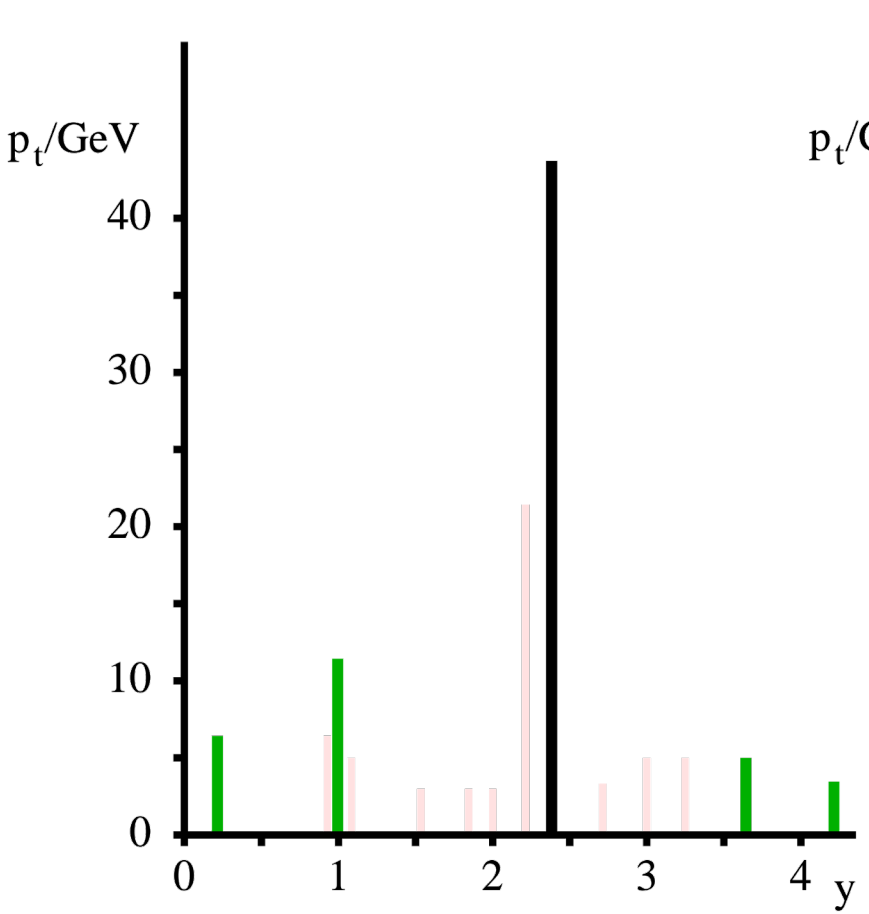


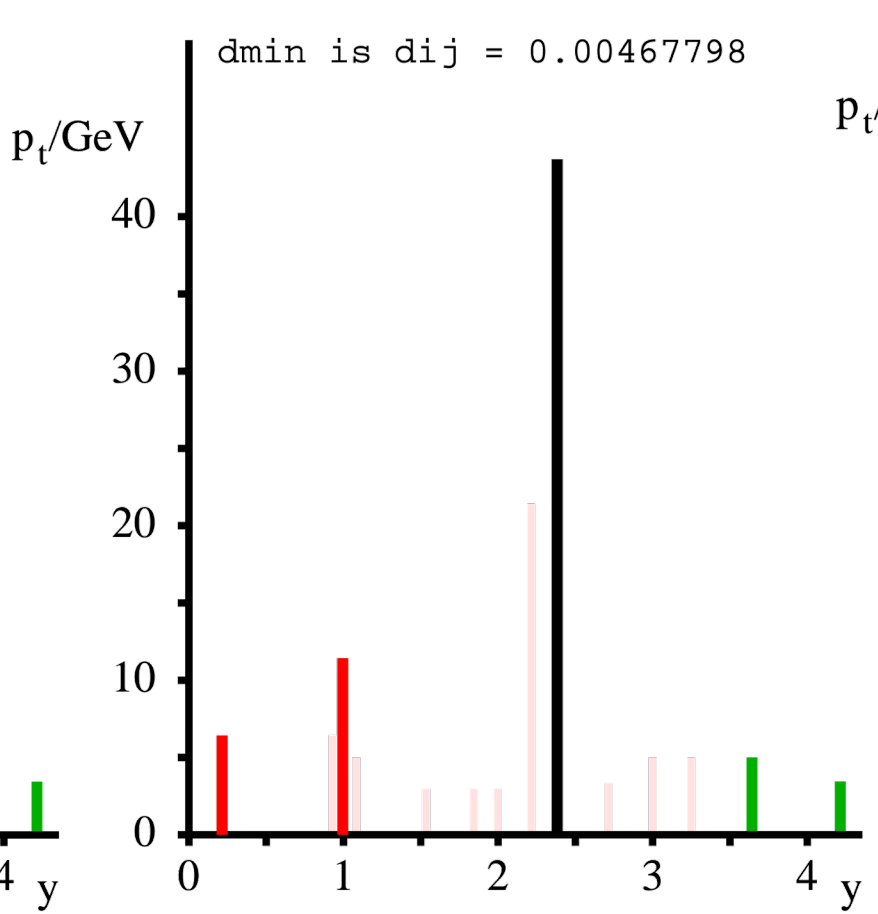
Example by Gavin Salam

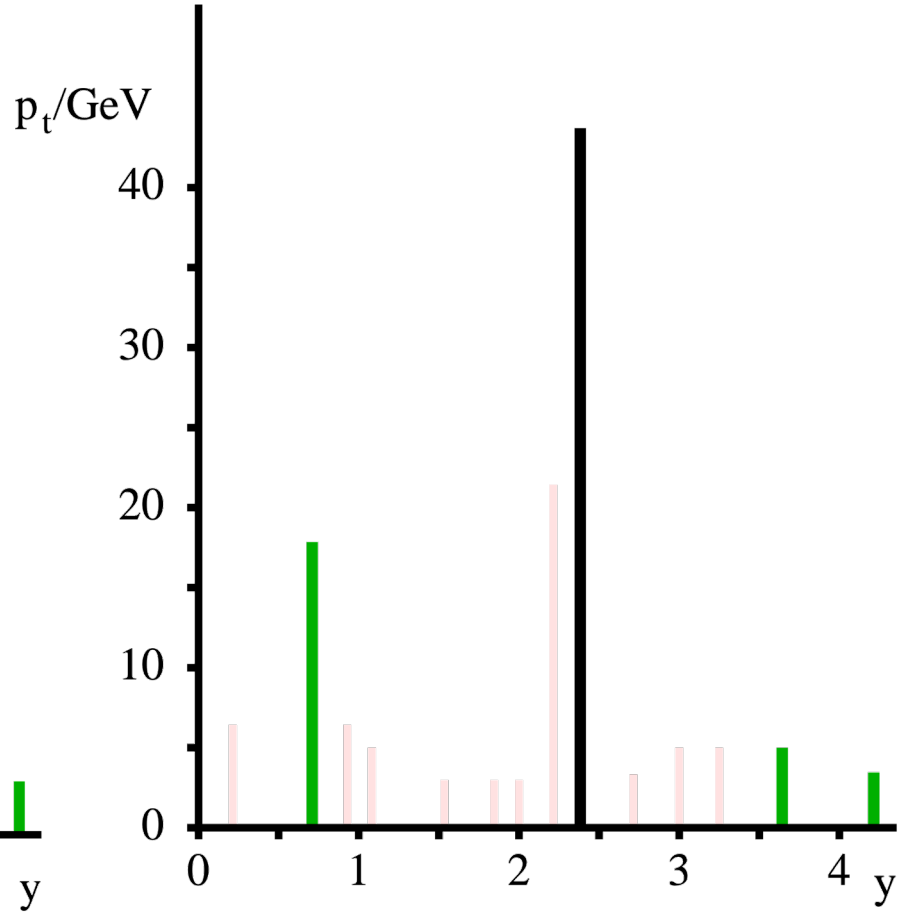
层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$





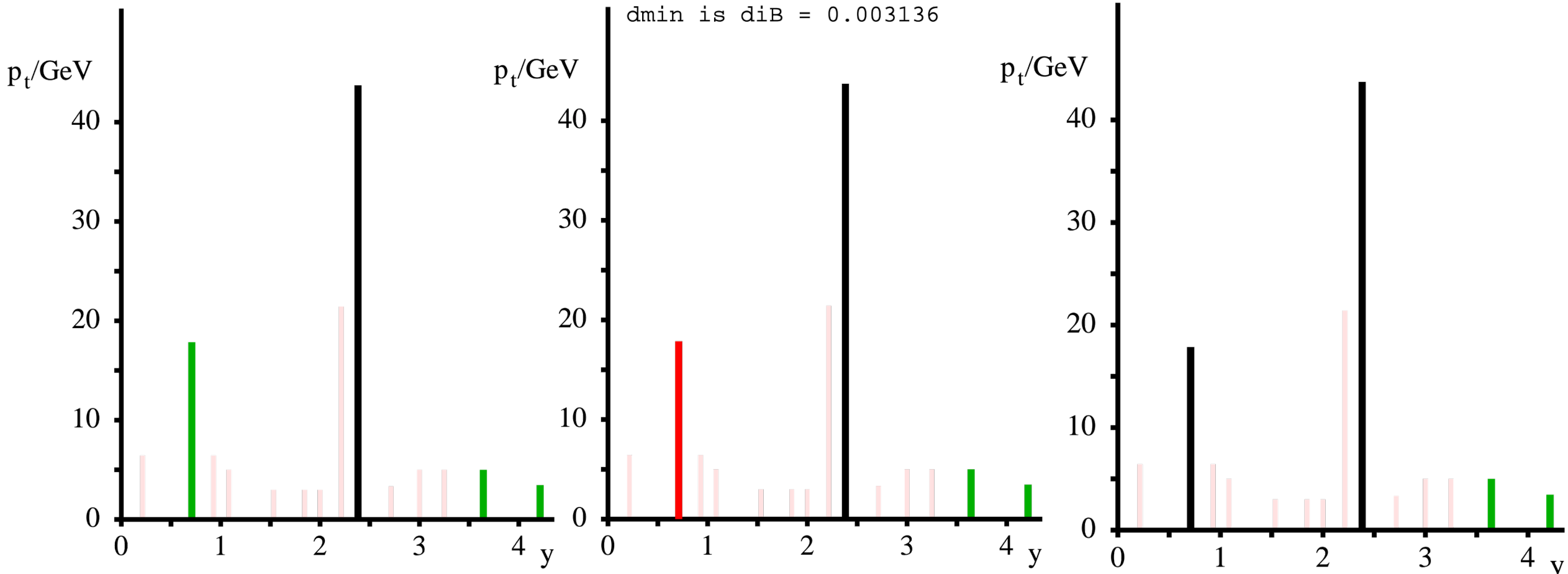


Example by Gavin Salam

层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$

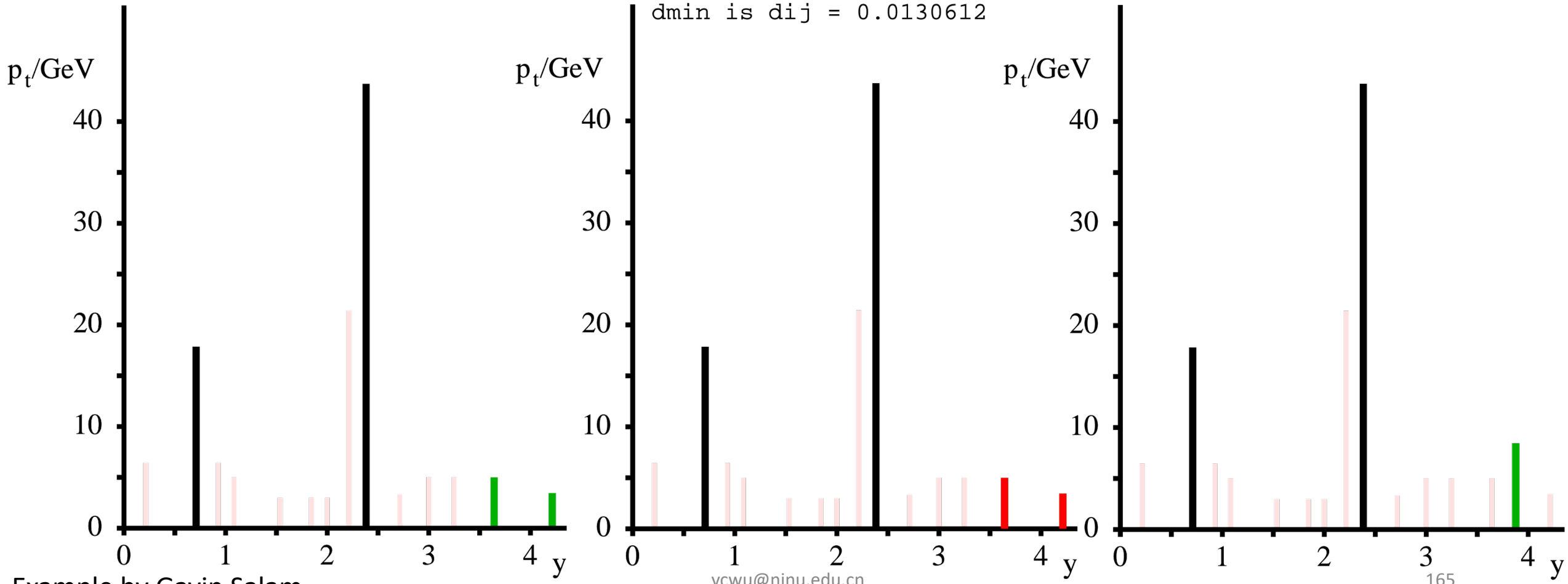


Example by Gavin Salam

层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$

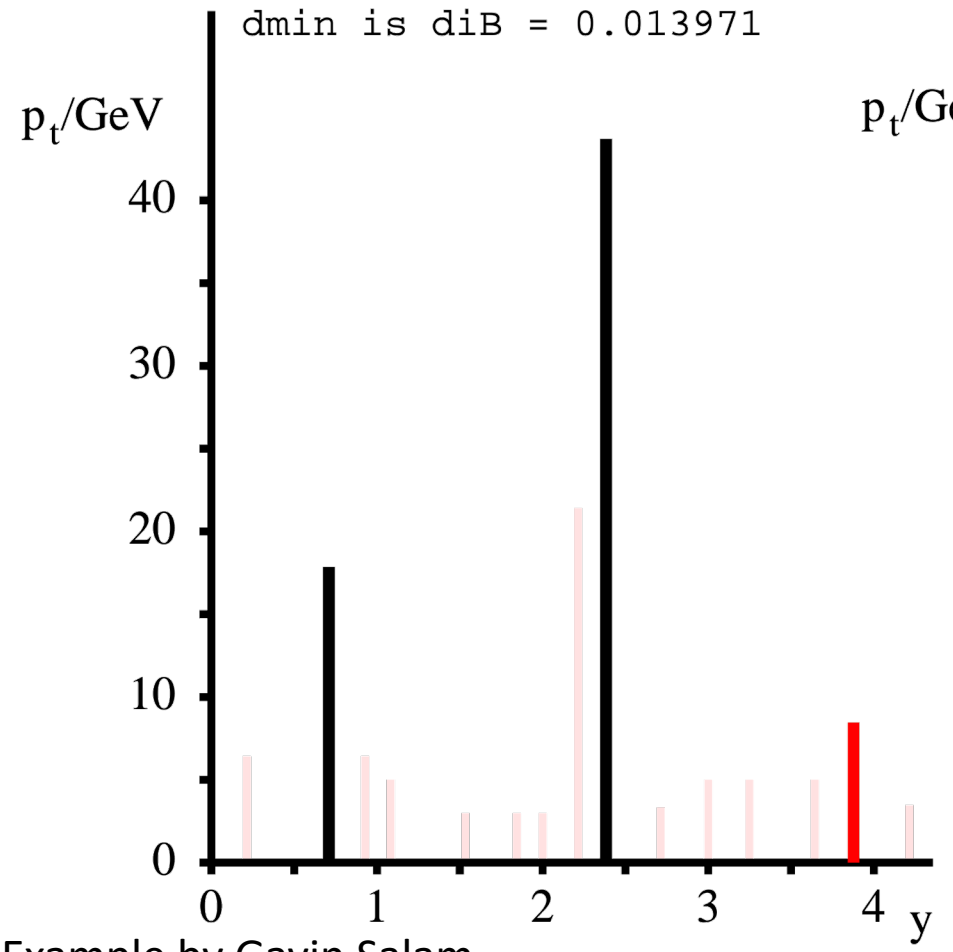


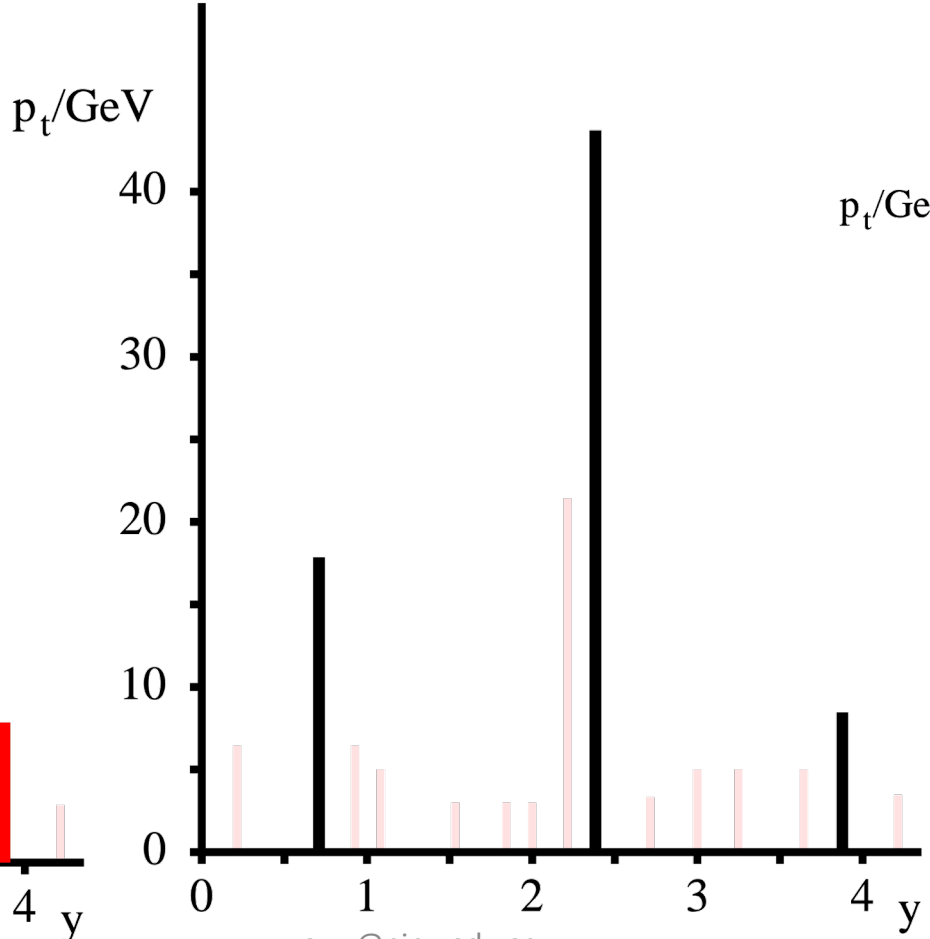
Example by Gavin Salam

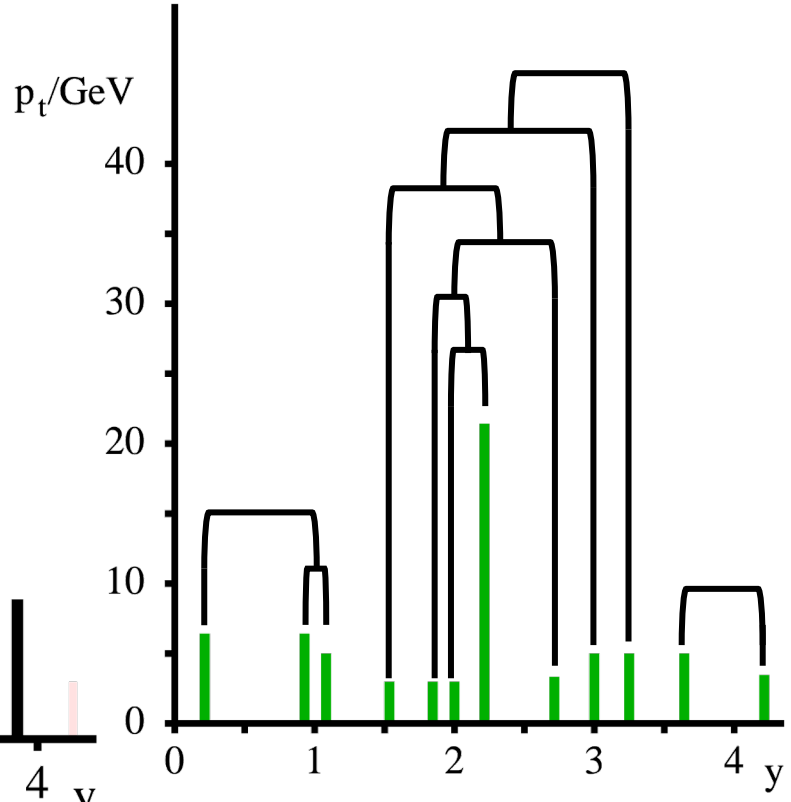
层次聚类

- Jet Clustering

$$d_{ij} = \min\left(p_{Ti}^{-2}, p_{Tj}^{-2}\right)\frac{\Delta R_{ij}^2}{R^2}, \qquad d_{iB} = p_{Ti}^{-2}$$







Example by Gavin Salam

k均值聚类 - k-means clustering

- 原型聚类 - prototype-based clustering
  - 聚类结构可以通过一组原型来描述
  - 初始化原型，迭代更新原型
- k均值聚类
  - 数据集：$D = \{\vec{x}_1, \cdots, \vec{x}_m\}$
  - 期望将数据集分为k $(k < m)$类：$\{C_1, \cdots, C_k\}$
  - 并且最小化平方误差

$$\epsilon = \sum_{\ell=1}^{k} \sum_{\vec{x}_i \in C_\ell} \left|\left|\vec{x}_i - \vec{\mu}_\ell\right|\right|_2^2$$

  - 指数级的可能性，迭代优化
    - 不保证一定能找到最优解

将m个样本分成k类的可能分法的数目

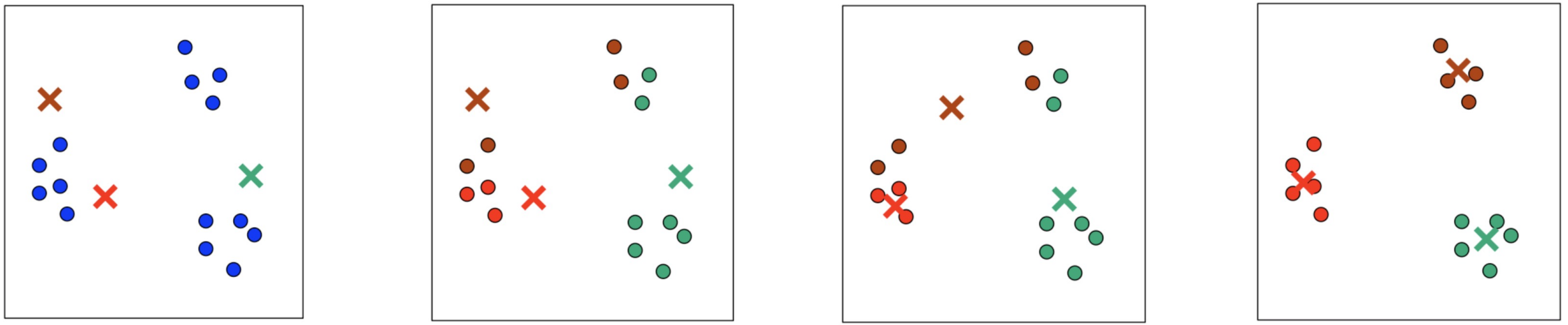
$$S(m,k) = \frac{1}{k!} \sum_{\ell=1}^{k} (-1)^{k-\ell} \binom{k}{\ell} \ell^m$$

k均值聚类

- 迭代方案：
  - 给定数据集$D = \{\vec{x}_1, \cdots, \vec{x}_m\}$和需分类数目$k$:$\{C_1, \cdots, C_k\}$
  - 随机从数据集中选择k个样本作为初始聚类中心$\mu^{(0)} = \left\{\vec{\mu}_1^{(0)}, \cdots, \vec{\mu}_k^{(0)}\right\}$
  - 迭代：$t = 0, \cdots\cdots$
    - 根据聚类中心进行聚类：
      - 对数据集中每个样本$\vec{x}_i$，计算样本到每个聚类中心的距离$d_{i\ell} = dist\left(\vec{x}_i, \vec{\mu}_\ell^{(t)}\right)$
      - 将样本指派到与其距离最近的聚类中心所属的类别：
        - $\lambda_i^{(t)} = \arg\min_\ell dist\left(\vec{x}_i, \vec{\mu}_\ell^{(t)}\right)$
    - 更新聚类中心
      - $\vec{\mu}_\ell^{(t+1)} = \frac{1}{|C_\ell|}\sum_{\vec{x}_i \in C_\ell} \vec{x}_i$
    - 如果迭代收敛或符合停止条件，停止迭代

Jet Cone Clustering Algorithm
- 从一些“种子”出发
- 在以种子为中心的圆锥内的粒子归为一个Jet
- 重新计算中心
- 直到找到稳定的归类(stable cone)
- 处理分类的重叠

k均值聚类

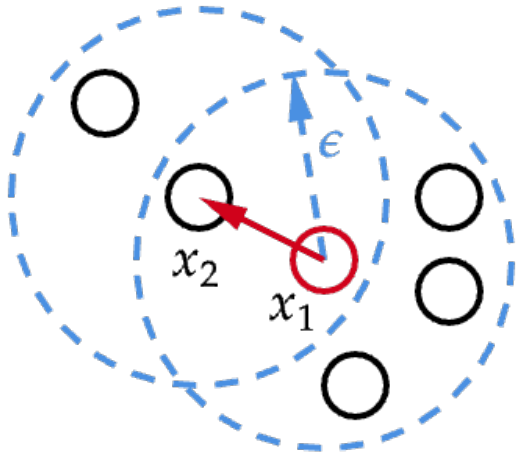


[Figures by E. Garrett-Mayer]

- 不保证全局最优
- 对初始聚类中心的选择比较敏感
- 需要提供类别数目k

密度聚类

- 聚类结构能通过样本分布的紧密程度来确定
- 给定数据集$D = \{\vec{x}_1, \cdots, \vec{x}_m\}$，以及邻域参数$(\epsilon, N_{th})$
  - ϵ–邻域：$D_\epsilon(\vec{x}_i) = \{\vec{x}_j | dist(\vec{x}_i, \vec{x}_j) \leq \epsilon, \vec{x}_j \in D\}$
  - 核心对象：$\vec{x}_i$是核心对象，如果$|D_\epsilon(\vec{x}_i)| \geq N_{th}$
  - 密度直达：如果$\vec{x}_j \in D_\epsilon(\vec{x}_i)$且$\vec{x}_i$是核心对象，则称$\vec{x}_j$可由$\vec{x}_i$密度直达
    - $\vec{x}_i \rightarrow \vec{x}_j$
    - 不对称；$\vec{x}_j$不一定是核心对象



$N_{th} = 5$

x_1是核心对象
x_2不是核心对象
x_2可由x_1密度直达

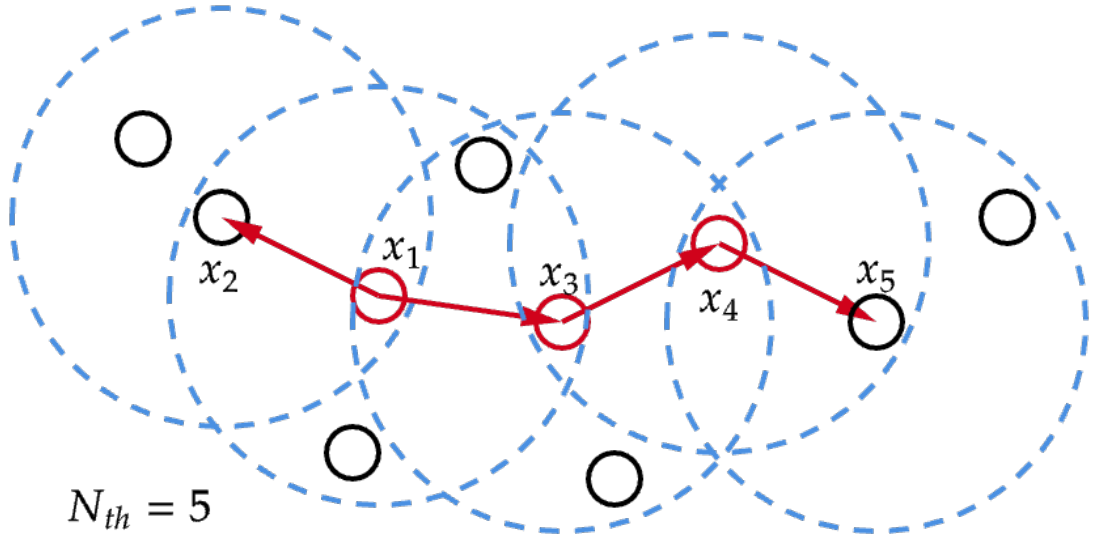
密度聚类

- 聚类结构能通过样本分布的紧密程度来确定
- 给定数据集$D = \{\vec{x}_1, \cdots, \vec{x}_m\}$，以及邻域参数$(\epsilon, N_{th})$
  - 密度可达：若存在一个样本序列：$\vec{p}_1 = \vec{x}_i, \vec{p}_2, \cdots \vec{p}_{n-1}, \vec{p}_n = \vec{x}_j$
    - 其中$\vec{p}_{k+1}$由$\vec{p}_k$密度直达，则称$\vec{x}_j$由$\vec{x}_i$密度可达
    - 满足传递性：$\vec{x}_i \Rightarrow \vec{x}_k$且$\vec{x}_k \Rightarrow \vec{x}_j$，则$\vec{x}_i \Rightarrow \vec{x}_j$
    - 不满足对称性（密度直达不对称）
  - 密度相连：$\vec{x}_i \sim \vec{x}_j$
    - $\vec{x}_k \Rightarrow \vec{x}_i$且$\vec{x}_k \Rightarrow \vec{x}_j$，则$\vec{x}_i$和$\vec{x}_j$密度相连

序列：$\{x_1, x_3, x_4, x_5\}$

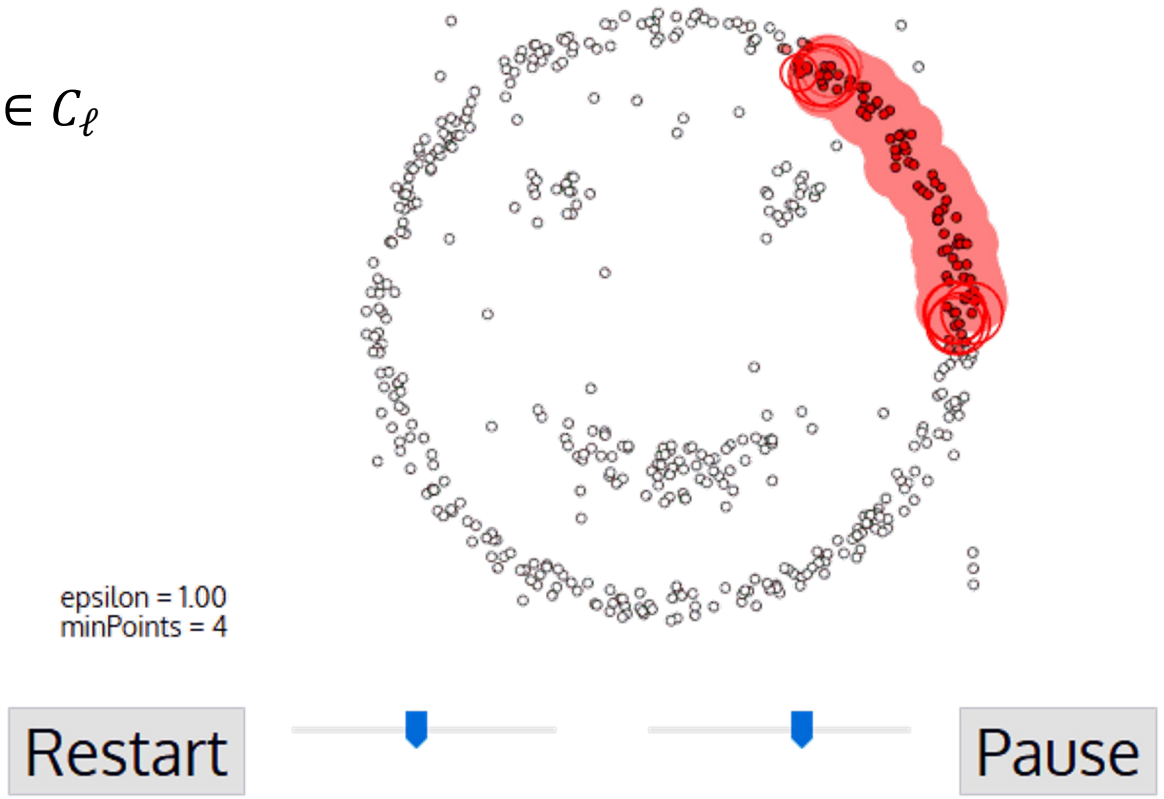
$x_1 \Rightarrow x_5$

$x_2 \sim x_5$



密度聚类

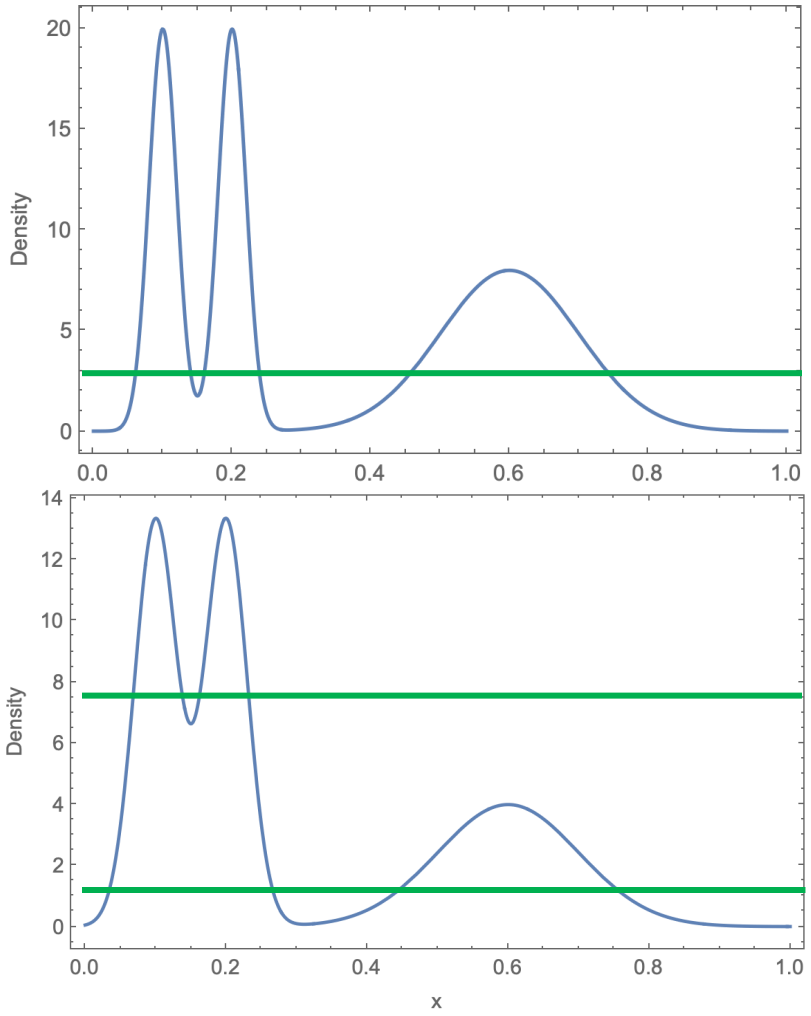
- DBSCAN算法
  - 类：由密度可达关系得到的最大的密度相连的样本集合
  - 每个类$C_\ell \subseteq D$是满足以下性质的非空子集：
    - 连接性：$\vec{x}_i, \vec{x}_j \in C_\ell$，则$\vec{x}_i \sim \vec{x}_j$
    - 最大性：$\vec{x}_i \in C_\ell$并且$\vec{x}_i \Rightarrow \vec{x}_j$，则$\vec{x}_j \in C_\ell$
- 如何找到这样的类？
  - 若$\vec{x}_i$是核心对象
  - 则$X_i = \{\vec{x}_j | \vec{x}_i \Rightarrow \vec{x}_j\}$是一个类
    - 很显然满足连接性和最大性
  - 以核心对象为种子
  - 找到所有其密度可达的点
  - 知道所有核心对象都已经归类
  - 会留下噪点



Credit: https://help.supermap.com/iDesktopX/1101/zh/tutorial/SpecialFeatures/Analyst/SpatialStatisticalAnalysis/DensityBasedClustering

密度聚类

- DBSCAN的问题
  - 全局的ϵ和N_{th}



按照“距离”排序 - OPTICS算法

- 距离依旧由ϵ, N_{th}来确定
- 从核心对象出发