机器学习Python实战

Jane 上海成趣信息科技有限公司

- 什么是机器学习?
- 机器学习的3个C
- 开发机器学习应用的步骤

- 监督学习：用朴素贝叶斯、最近邻居法（kNN）分类
- 监督学习：用回归分析做预测
- 无监督学习：用k均值实现聚类

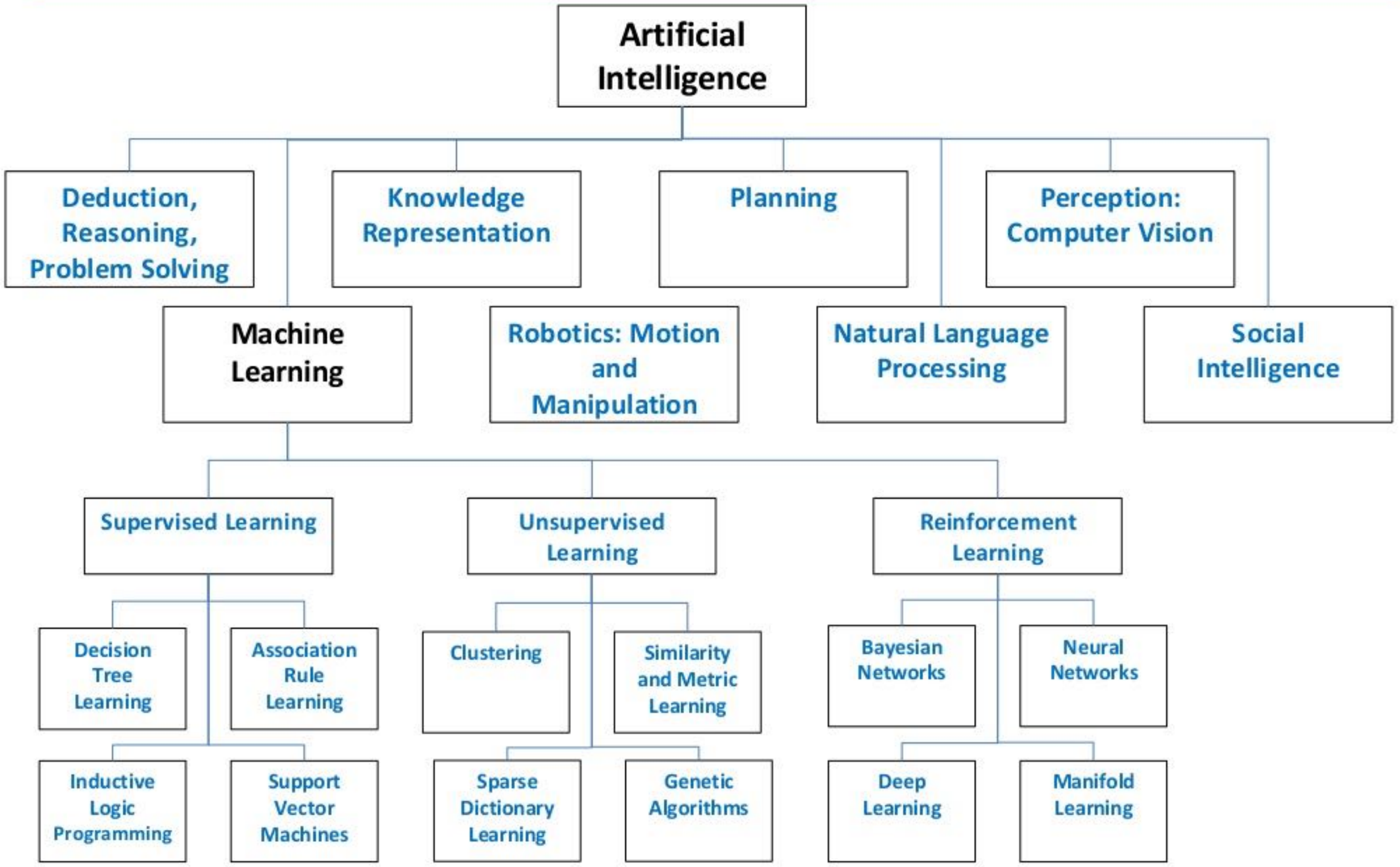
课前准备：

1. 安装Python 2.6以上版本均可。

2. 安装scikit-learn 0.18.1 安装链接为 http://scikit-learn.org/stable/install.html

3. 下载代码，下载链接为 https://github.com/pbharrin/machinelearninginaction

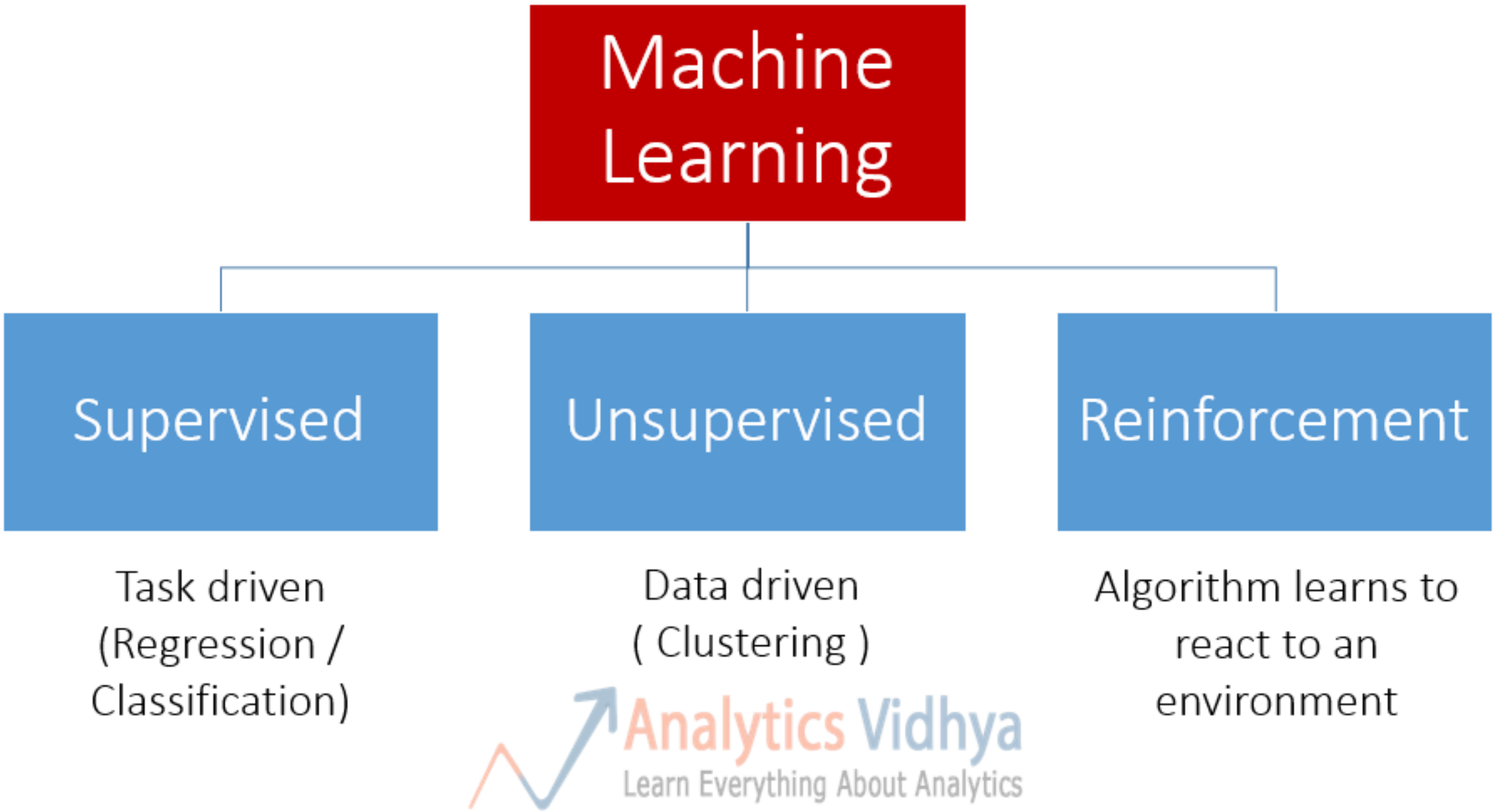
Artificial Intelligence / Machine Learning Classification



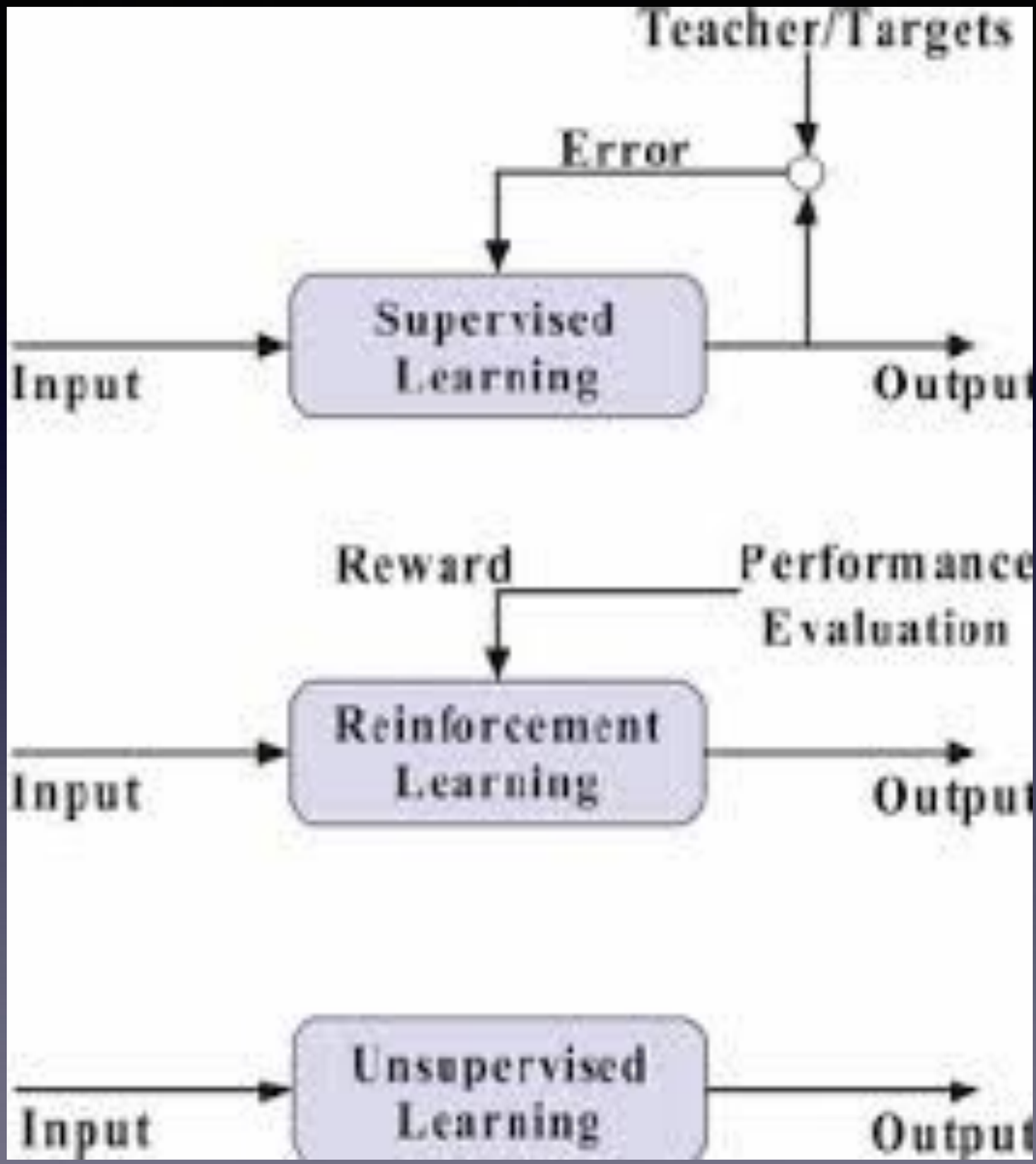
什么是机器学习?

- 机器学习理论：设计和分析一些让计算机可以像人一样自动学习的算法
- 机器学习算法：从数据中自动分析获得规律，并利用规律对未知数据进行预测的算法
- 机器学习的三个要素：
  - 数据
  - 模型
  - 算法

Types of Machine Learning

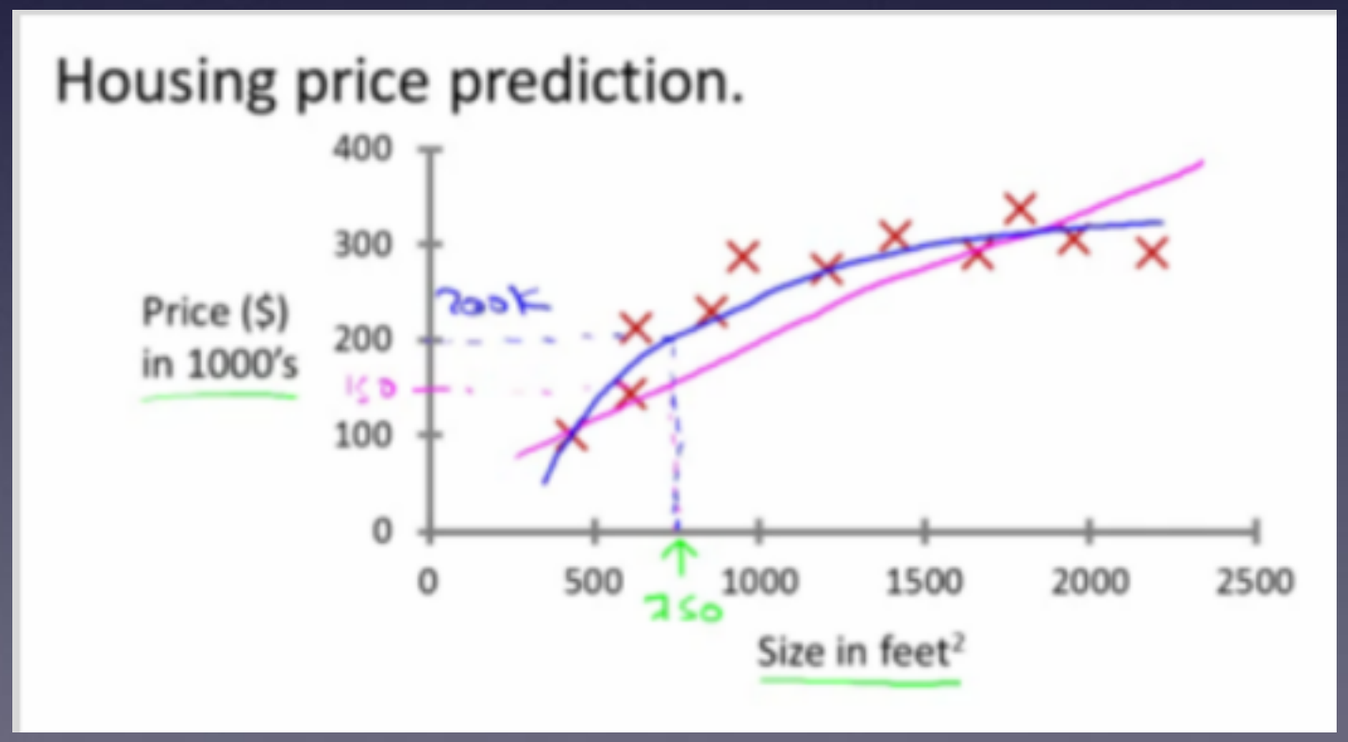


机器学习


Teacher/Targets
Error
Supervised Learning
Input
Output
Reward
Performance Evaluation
Reinforcement Learning
Input
Output
Unsupervised Learning
Input
Output

监督学习

- 从给定的训练数据集中学习出一个函数，根据函数，对新的输入数据预测输出结果。训练集包括输入和输出，即特征和目标。
- 需要人工标记样本，需要较大的人力成本。
- 例子：
  - 回归分析
  - 统计分类
- 模型分类
  - 生成式模型
  - 判别式模型



无监督学习

- 训练集没有人为标注的结果
- 例子
  - 用户分组
  - 新闻分类

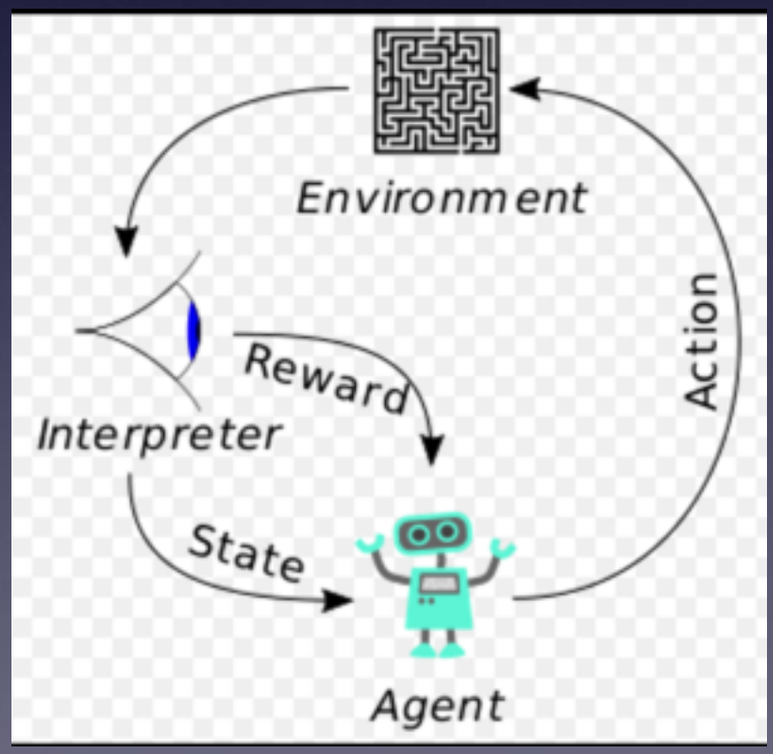
半监督学习

- 介于监督学习和无监督学习之间，有少量的标记数据与大量的无标记数据
- 师傅领进门，成功靠个人

增强学习

- 通过观察来学习如何动作。每个动作都会对环境有所影响，学习对象根据观察到的周围环境的反馈来做出判断。

- 例子：
  - 自动驾驶
  - 和对手玩游戏



机器学习的3个C

- Collaborative filtering 协同过滤 推荐
- Clustering 聚类
- Classification 分类

协同过滤

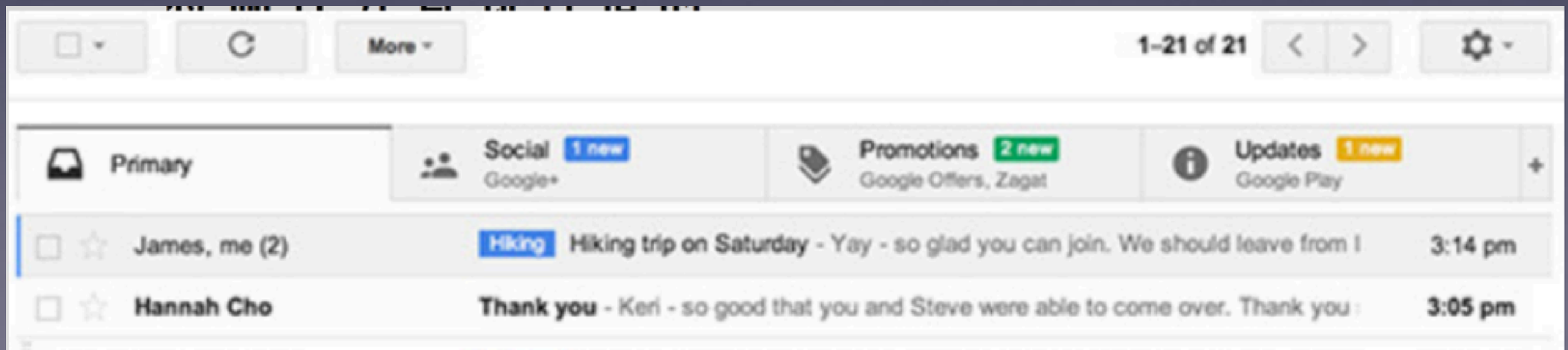
- 基于一些信息进行过滤
- 应用于推荐
  - 基于条目的过滤：条目之间的相似性
  - 基于用户的过滤：用户之间的相似性
  - 基于内容的过滤：内容之间的相似性

聚类

- 将相似的对象通过静态分类的方法分成不同的组别或者更多的子集
- 应用于
  - 寻找同类新闻、博客和文件
  - 给网站用户分组
- 分类方法
  - 结构性聚类
  - 分散型聚类

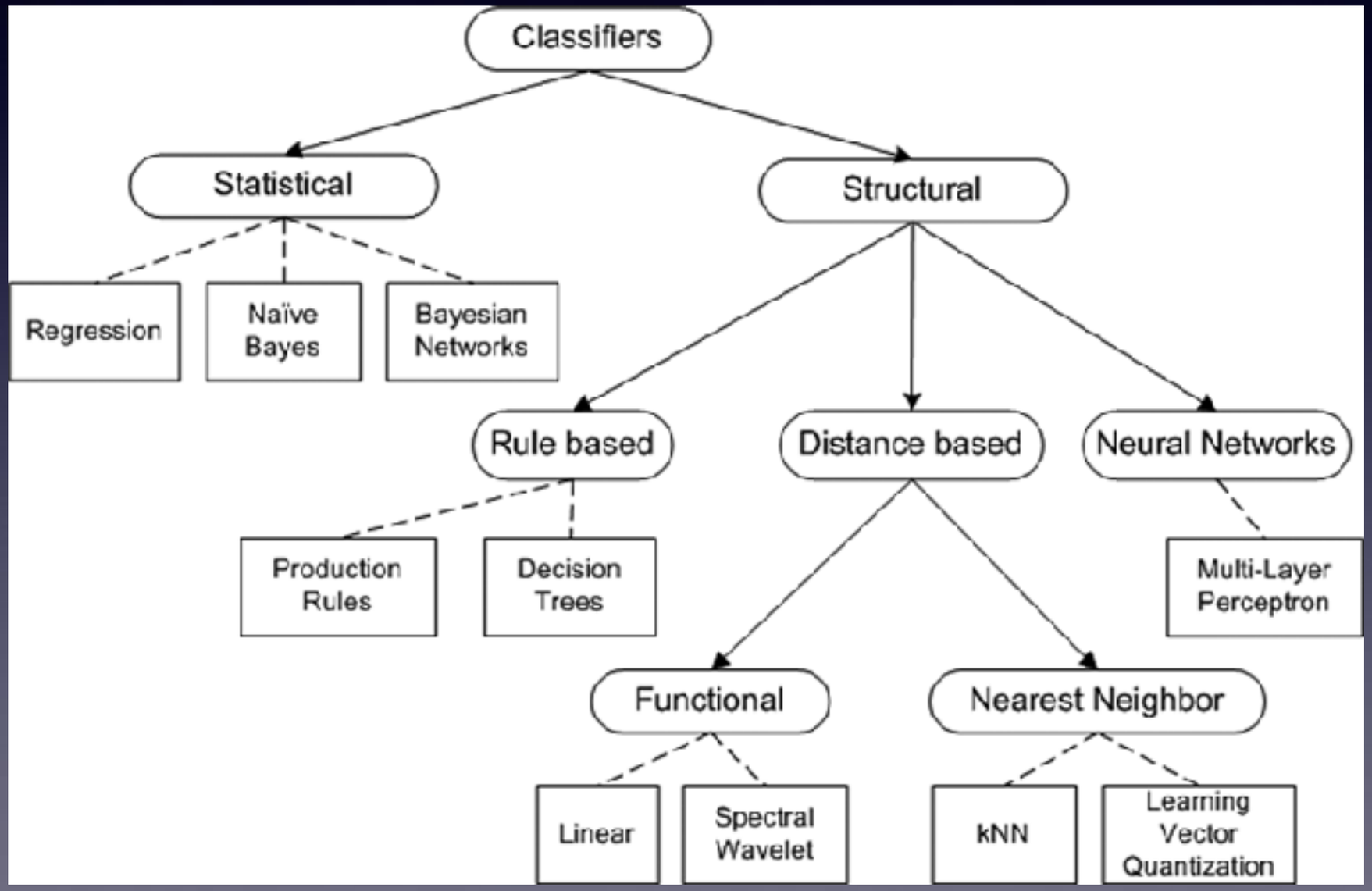
分类

- 识别样本所属的类别
- 应用案例
  - 过滤垃圾邮件
  - 将邮件分置放到合适的文件夹
  - 区分保险欺诈用户与正常用户



分类类别

- 结构分类算法
  - 基于规则的算法
  - 基于距离的算法
  - 神经网络算法
- 统计分类算法
  - 逻辑回归算法
  - 朴素贝叶斯
  - 贝叶斯网



开发机器学习应用程序的步骤

1. 搜集数据
2. 准备输入数据
3. 分析输入数据
4. 训练算法
5. 测试算法
6. 使用算法

代码来源

- https://github.com/pbharrin/machinelearninginaction

监督学习：分类

- 朴素贝叶斯

- 最近邻居法 k-NN

朴素贝叶斯

- 概率推理 与确定性推理相对应
- 基于独立假设，假设样本每个特征与其他特征不相关
- 依靠精准的自然概率模型

朴素贝叶斯优缺点

- 优点：在数据较少的情况下有小，可以处理多类别问题。
- 缺点：对于输入数据的准备方式较为敏感。
- 数据类型：标称型数据。

朴素贝叶斯概率模型

条件概率 $p(C|F_1)$ 事件C在另外一个事件F_1已经发生条件下的发生概率。读作“在F_1条件下C的概率”

条件概率模型 $p(C|F_1,\dots,F_n)=p(C)p(F_1,\dots,F_n|C)/p(F_1,\dots,F_n)$

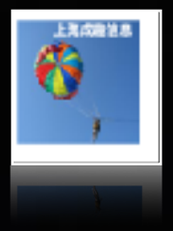
问题变为求 $p(C)p(F_1,\dots,F_n|C)$

对朴素贝叶斯分类器

$p(C)p(F_1,\dots,F_n|C) = p(C)(p(F_1|C)p(F_2|C)\dots p(F_n|C)$

练习：使用朴素贝叶斯进行文档分类

1. 搜集数据：生成文本文件
2. 准备输入数据：数值型或者布尔型数据
3. 分析输入数据：如果有大量特征时，采用直方图效果好于绘制特征
4. 训练算法：计算不同的独立特征的条件概率
5. 测试算法：计算错误率
6. 使用算法：适用于任何的分类场景



阅读内容

- 机器学习实战第四章4.5,4.6
- 重点：
  - 训练算法：从词向量计算概率
  - 测试算法：根据现实情况修改分类器

示例：分类电子邮件

- 收集数据：提供文本文件
- 准备数据：解析成词条向量
- 分析数据：检查词条确保解析正确
- 训练算法：使用trainNB0（）函数
- 测试算法：使用classifyNB（），构建新的测试函数计算错误率
- 使用算法：构建分类文档，输出错分文档

使用算法

- >>> bayes.spamTest()
- 输出10封随机选择的电子邮件上的分类错误率
- 可能会出现的错误：将垃圾邮件误判为正常邮件

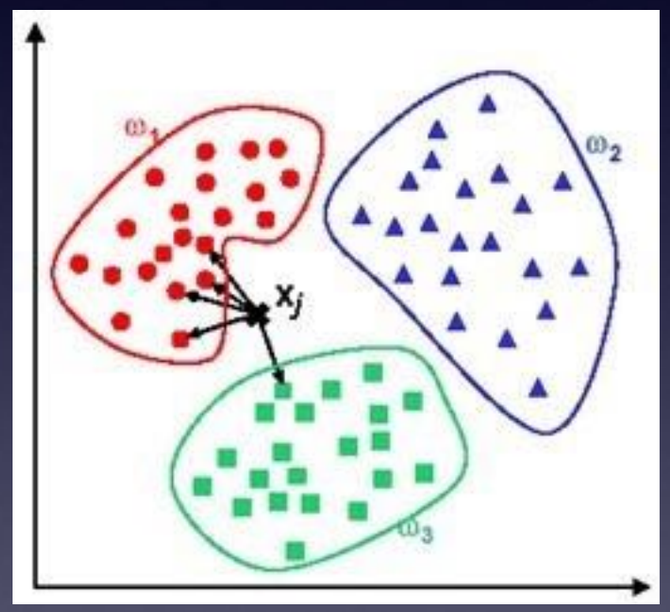
```
>>> import bayes
>>> bayes.spamTest()
the error rate is:  0.0
>>> bayes.spamTest()
the error rate is:  0.0
>>> bayes.spamTest()
the error rate is:  0.0
>>> bayes.spamTest()
the error rate is:  0.0
>>> bayes.spamTest()
the error rate is:  0.0
>>> bayes.spamTest()
the error rate is:  0.0
>>> bayes.spamTest()
classification error ['yeah', 'ready', 'may', 'not', 'here', 'because', 'jar', 'jar', 'has', 'plane', 'tickets', 'germany', 'for']
classification error ['home', 'based', 'business', 'opportunity', 'knocking', 'your', 'door', 'don', 'rude', 'and', 'let', 'this',
'chance', 'you', 'can', 'earn', 'great', 'income', 'and', 'find', 'your', 'financial', 'life', 'transformed', 'learn', 'more',
'here', 'your', 'success', 'work', 'from', 'home', 'finder', 'experts']
classification error ['benoit', 'mandelbrot', '1924', '2010', 'benoit', 'mandelbrot', '1924', '2010', 'wilmott', 'team', 'benoit',
'mandelbrot', 'the', 'mathematician', 'the', 'father', 'fractal', 'mathematics', 'and', 'advocate', 'more', 'sophisticated',
'modelling', 'quantitative', 'finance', 'died', '14th', 'october', '2010', 'aged', 'wilmott', 'magazine', 'has', 'often',
'featured', 'mandelbrot', 'his', 'ideas', 'and', 'the', 'work', 'others', 'inspired', 'his', 'fundamental', 'insights', 'you',
'must', 'logged', 'view', 'these', 'articles', 'from', 'past', 'issues', 'wilmott', 'magazine']
the error rate is:  0.3
```

最近邻居法kNN

- 我们向朋友们征求意见的方法是?

最近邻居法k-NN

- 寻找品味相似的朋友，转换成数学的方式，我们找到离目标点最近的**K**个点，这些点就是**K**近邻。



- 算法：
  - 寻找与目标最近的k个点
  - 分析近邻的类别，根据少数服从多数的原则，为目标点定义类别

kNN算法

- 优点：精度高、对异常值不敏感、无数据输入假定。
- 缺点：计算复杂度高、空间复杂度高。
- 适用数据范围：数值型和标称型。

如何用kNN改善约会网站的配对效果?

- 三种类型的人
  - 不喜欢的人
  - 魅力一般的人
  - 极具魅力的人
- 如何更好地将匹配对象划分到确切的分类中?
- 如何根据三种样本数据类型决定人的类型?

阅读内容

- 机器学习实战 第二章 2.1，2.2节
- 重点：
  - 归一化数据
  - 使用kNN将每组数据划分到某个类中
  - 如何测试分类器?

练习：在约会网站上使用kNN算法

1. 搜集数据：生成文本文件
2. 准备输入数据：使用Python解析文本文件
3. 分析输入数据：使用Matplotlib画二维扩散图
4. 训练算法：不适用
5. 测试算法：使用部分数据作为测试样本
6. 使用算法：产生简单的命令行程序，输入特征数据以判断对方是否为自己喜欢的类型。

1.准备数据：解析数据

- >>> reload(kNN)
- >>> datingDataMat,datingLabels = kNN.file2matrix('datingTestSet.txt')
- >>> datingDataMat
- >>> datingLabels[0:20]

2.分析数据：创建散点图

- >>> from numpy import *
- >>>import matplotlib
- >>> import matplotlib.pyplot as plt
- >>> fig = plt.figure()
- >>> ax =fig.add_subplot(111)
- >>> ax.scatter(datingDataMat[:,1],datingDataMat[:,2]) -> ax.scatter(datingDataMat[:,1],datingDataMat[:,2], 15.0*array(datingLabels),15.0*array(datingLabels))
- >>> plt.show()

3.准备数据：归一化数值

- >>> reload(kNN)
- >>> normMat, ranges, minVals = kNN.autuoNorm(datingDataMat)
- >>> normMat
- >>> ranges
- >>> minVals

4.测试算法：验证分类器

- >>> kNN.datingClassTest()

5.使用算法：分类

- 根据信息，预测是否会喜欢对方呢?

- >>> kNN.classifyPerson()

线性回归

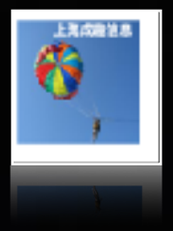
- 优点：结果易于理解，计算简单
- 缺点：对非线性数据拟合不好
- 适用数据类型：数值型和标称型数据

回归的指标

- 方差：模型之间的差异
- 偏差：模型预测值和数据之间的差异

回归分析做预测

- 收集数据：采用任何方法收集数据
- 准备数据：标称型数据将被转成二值型数据
- 分析数据：绘制出数据的可视化二维图
- 训练算法：找到回归系数
- 测试算法：使用R2或者预测值和数据的拟合度，分析模型的效果
- 使用算法：使用回归，在给定输入的时候预测出数值



阅读内容

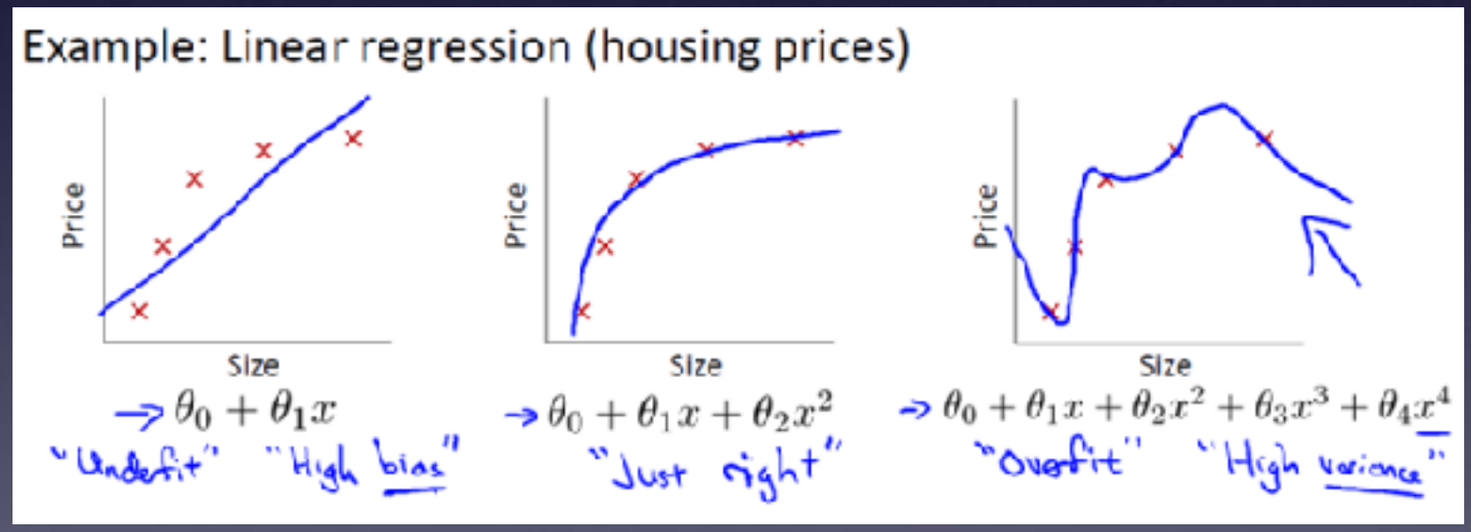
- 机器学习实战 第八章 8.1-8.3节
- 重点：
  - 局部加权线性回归
  - 选择核的类型

回归分析做预测

- 收集数据：采用任何方法收集数据
- 准备数据：标称型数据将被转成二值型数据
- 分析数据：绘制出数据的可视化二维图
- 训练算法：找到回归系数
- 测试算法：使用R2或者预测值和数据的拟合度，分析模型的效果
- 使用算法：使用回归，在给定输入的时候预测出数值

过拟合现象（Overfitting）

- 为了得到一致假设而使假设变得过度复杂
- 与样本拟合得很好，但是不能很好地预测实际情况
- 解决办法
  - 人工检查变量
  - 模型选择算法



练习四：预测鲍鱼的年龄

- 鲍鱼年龄可以从鲍鱼壳的层数推算得出。

1. 加载数据

```
>>> import regression
>>> abX,abY = regression.loadDataSet('abalone.txt')
>>> yHat01 = regression.lwlrTest(abX[0:99],abX[0:99],abY[0:99],0.1)
>>> yHat1 = regression.lwlrTest(abX[0:99],abX[0:99],abY[0:99],1)
>>> yHat10 =regression.lwlrTest(abX[0:99],abX[0:99],abY[0:99],10)
```

2. 计算误差大小

```
>>> regression.rssError(abY[0:99],yHat01.T)
56.78512629514465
>>> regression.rssError(abY[0:99],yHat1.T)
429.89056187032406
>>> regression.rssError(abY[0:99],yHat10.T)
549.11817088271073
```

回归案例：预测鲍鱼的年龄

- 为何不在所有数据集上使用最小的核呢？过拟合！！

- 利用新数据的结果

```
>>> yHat01 = regression.lwlrTest(abX[100:199],abX[0:99],abY[0:99],0.1)
>>> regression.rssError(abY[100:199],yHat01.T)
71808.011542448148
>>> yHat1 = regression.lwlrTest(abX[100:199],abX[0:99],abY[0:99],1)
>>> regression.rssError(abY[100:199],yHat1.T)
573.52614418958501
>>> yHat10 = regression.lwlrTest(abX[100:199],abX[0:99],abY[0:99],10)
>>> regression.rssError(abY[100:199],yHat10.T)
517.57119053831059
```

- 简单的线性回归

```
>>> ws = regression.standRegres(abX[0:99],abY[0:99])
>>> from numpy import *
>>> yHat = mat(abX[100:199])*ws
>>> regression.rssError(abY[100:199],yHat.T.A)
518.63631532408499
```

简单线性回归与局部加权

- 局部加权比普通线性回归更好
- 局部加权必须在整个数据集上运行，要保存所有的训练数据

无监督学习：k均值聚类算法

- 可以发现k个不同的簇，且每个簇的中心由簇中所含值的均值计算所得
- 优点：容易实现
- 缺点：可能收敛到局部最小值，在大规模数据集上收敛较慢
- 适用数据类型：数值型数据

无监督学习

K-均值算法的工作流程是这样的。首先，随机确定k个初始点作为质心。然后将数据集中的每个点分配到一个簇中，具体来讲，为每个点找距其最近的质心，并将其分配给该质心所对应的簇。这一步完成之后，每个簇的质心更新为该簇所有点的平均值。

上述过程的伪代码表示如下：

```
创建k个点作为起始质心（经常是随机选择）
当任意一个点的簇分配结果发生改变时
    对数据集中的每个数据点
        对每个质心
            计算质心与数据点之间的距离
        将数据点分配到距其最近的簇
    对每一个簇，计算簇中所有点的均值并将均值作为质心
```

k均值聚类的一般流程

- 收集数据：使用任意方法
- 准备数据：需要数值型数据计算距离，也将标称型数据映射为二值性数据再用于距离计算
- 分析数据：使用任意方法
- 训练算法：无此步骤，不需要训练
- 测试算法：应用聚类算法观察测试结果。使用误差平方和评价算法结果
- 使用算法：簇质心代表整个簇的数据做出决策

使用scikit-learn识别手写数字

- http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html

参考资料

- 机器学习实战
- http://scikit-learn.org/stable/index.html

杨晓春

上海成趣信息

科技有限公司

独立IT顾问

产品设计

技术开发、技术管理

人工智能、数据分析解决方案

物联网解决方案

医疗养老产品

DevHub开发者社区

DevHub开发者社区

分享、启发、探索

传播IT知识文化
陪伴探索者前行