朴素贝叶斯算法与K近邻算法
Naïve Bayes Algorithms

目录

1.1 朴素贝叶斯法的学习与分类

考虑一个分类问题，我们希望根据动物的某些特征（$X = (x_1, x_2, \ldots, x_n)$）来区分猫（y = 1）和狗（y = 0）。

- **判别模型**
  - 找到将猫和狗分开的决策边界或分类原则。
  - 为了分类一只新动物，判别模型会检查它落在决策边界的哪一边，并直接做出决定。
  - 直接估计后验概率 $p(y|x)$ 。
- **生成模型**
  - 分别学习猫和狗的特征模型。
  - 要对新动物进行分类，将其与猫/狗模型进行匹配，并查看它看起来更像哪个模型。
  - 估计先验概率 $p(y)$ 和条件概率 $p(x|y)$，根据贝叶斯定理计算后验概率 $p(y|x)$ 。

1.1 朴素贝叶斯法的学习与分类

朴素贝叶斯

思想：**朴素贝叶斯假设，又称条件独立性假设**

对于特征$X = (x_1, x_2, \dots, x_n)$，满足 $x_i \perp x_j \mid y \ (i \neq j)$

$$p(X \mid y) = p(x_1, x_2, \dots, x_n \mid y) = \prod_{j=1}^{n} p(x_j \mid y)$$

Motivation：简化运算

条件独立假设，用于分类的特征在分类模型确定的条件下是条件独立的。

1.1 朴素贝叶斯法的学习与分类

朴素贝叶斯法

思想：**朴素贝叶斯假设，又称条件独立性假设**

做法：根据**贝叶斯定理**来估计每个类别的**后验概率**。

$$p(y|x)=\frac{p(x,y)}{p(x)}=\frac{p(x|y)p(y)}{p(x)}=\frac{p(x|y)p(y)}{\sum_i p(x|y_i)p(y_i)}\propto p(x|y)p(y)$$

对于样本X,朴素贝叶斯法的目标是找到

$$y=\arg\max_y p(y|X)=\arg\max_y \frac{p(X,\ y)}{p(X)}=\arg\max_y p(X|y)p(y)$$
$$=\arg\max_y \prod_i p(x_i|y)\,p(y)$$

作为X的分类结果。

1.2例子

例 4.1 试由表 4.1 的训练数据学习一个朴素贝叶斯分类器并确定 $x = (2, S)^{\mathrm{T}}$ 的类标记 y。表中 $X^{(1)}$，$X^{(2)}$ 为特征，取值的集合分别为 $A_1 = \{1, 2, 3\}$，$A_2 = \{S, M, L\}$，Y 为类标记，$Y \in C = \{1, -1\}$。

表 4.1　训练数据

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$X^{(1)}$	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
$X^{(2)}$	S	M	M	S	S	S	M	M	L	L	L	M	M	L	L
Y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

1.2例子

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$X^{(1)}$	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
$X^{(2)}$	S	M	M	S	S	S	M	M	L	L	L	M	M	L	L
Y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

$$P(Y=1)=\frac{9}{15},\quad P(Y=-1)=\frac{6}{15}$$

$$P(X^{(1)}=1|Y=1)=\frac{2}{9},\quad P(X^{(1)}=2|Y=1)=\frac{3}{9},\quad P(X^{(1)}=3|Y=1)=\frac{4}{9}$$

$$P(X^{(2)}=S|Y=1)=\frac{1}{9},\quad P(X^{(2)}=M|Y=1)=\frac{4}{9},\quad P(X^{(2)}=L|Y=1)=\frac{4}{9}$$

$$P(X^{(1)}=1|Y=-1)=\frac{3}{6},\quad P(X^{(1)}=2|Y=-1)=\frac{2}{6},\quad P(X^{(1)}=3|Y=-1)=\frac{1}{6}$$

$$P(X^{(2)}=S|Y=-1)=\frac{3}{6},\quad P(X^{(2)}=M|Y=-1)=\frac{2}{6},\quad P(X^{(2)}=L|Y=-1)=\frac{1}{6}$$

对于给定的 $x=(2,S)^{\mathrm{T}}$ 计算：

$$P(Y=1)P(X^{(1)}=2|Y=1)P(X^{(2)}=S|Y=1)=\frac{9}{15}\cdot\frac{3}{9}\cdot\frac{1}{9}=\frac{1}{45}$$

$$P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)=\frac{6}{15}\cdot\frac{2}{6}\cdot\frac{3}{6}=\frac{1}{15}$$

因为 $P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)$ 最大，所以 $y=-1$。 ■

1.3 贝叶斯估计

思考：在前面的分类算法中，如果测试样例中的特征没有在训练集中出现会造成什么结果？

会影响到后验概率的计算结果，使分类产生偏差。解决这一问题的方法是采用**贝叶斯估计**。具体地，估计特征x_k的条件概率为：

$$p(x_k|y_i) = \frac{C(x_k, y_i) + \lambda}{C(y_i) + K(x_k)\lambda}$$

估计y_i的概率计算为：

$$p(y_i) = \frac{C(y_i) + \lambda}{N + K(y_i)\lambda}$$

式中C表示符合条件的样本个数，$K(x)$为特征x的取值种类数，$\lambda \geq 0$。等价于在随机变量各个取值的频数上赋予一个正数$\lambda \geq 0$。当$\lambda = 0$时就是极大似然估计。一般取$\lambda = 1$，这时称为**拉普拉斯平滑** (Laplacian smoothing)。

1.3 贝叶斯估计例子

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$X^{(1)}$	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
$X^{(2)}$	S	M	M	S	S	S	M	M	L	L	L	M	M	L	L
Y	-1	-1	1	1	-1	-1	-1	1	1	1	1	1	1	1	-1

$$P(Y=1)=\frac{10}{17},\quad P(Y=-1)=\frac{7}{17}$$

$$P(X^{(1)}=1|Y=1)=\frac{3}{12},\quad P(X^{(1)}=2|Y=1)=\frac{4}{12},\quad P(X^{(1)}=3|Y=1)=\frac{5}{12}$$

$$P(X^{(2)}=S|Y=1)=\frac{2}{12},\quad P(X^{(2)}=M|Y=1)=\frac{5}{12},\quad P(X^{(2)}=L|Y=1)=\frac{5}{12}$$

$$P(X^{(1)}=1|Y=-1)=\frac{4}{9},\quad P(X^{(1)}=2|Y=-1)=\frac{3}{9},\quad P(X^{(1)}=3|Y=-1)=\frac{2}{9}$$

$$P(X^{(2)}=S|Y=-1)=\frac{4}{9},\quad P(X^{(2)}=M|Y=-1)=\frac{3}{9},\quad P(X^{(2)}=L|Y=-1)=\frac{2}{9}$$

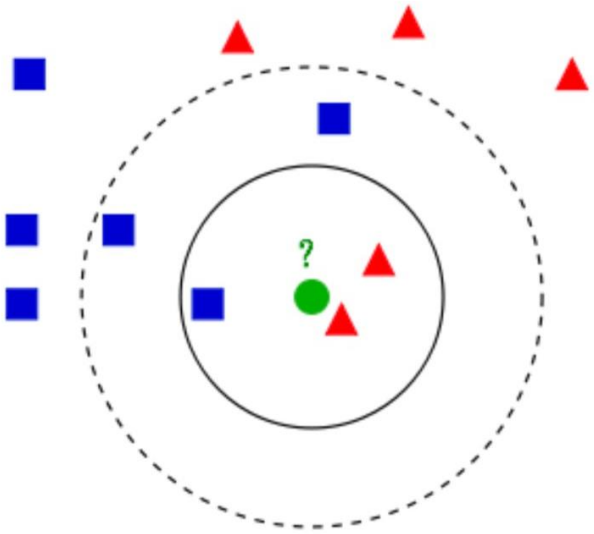
$$P(Y=1)P(X^{(1)}=2|Y=1)P(X^{(2)}=S|Y=1)=\frac{10}{17}\cdot\frac{4}{12}\cdot\frac{2}{12}=\frac{5}{153}=0.0327$$

$$P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)=\frac{7}{17}\cdot\frac{3}{9}\cdot\frac{4}{9}=\frac{28}{459}=0.0610$$

由于 $P(Y=-1)P(X^{(1)}=2|Y=-1)P(X^{(2)}=S|Y=-1)$ 最大，所以 $y=-1$。 ■

2. 机器学习算法--KNN

- K-近邻(KNN)算法—KNN处理分类问题



半径大小 表示 K值大小

k-nearest neighbours classifier:

$$f(q) = maj\left(g\left(\Phi_{X,k}(q)\right)\right)$$

其中:

$\Phi_{X,k}(q)$：返回训练集X中距离q最近的k个样本

$g(\cdot)$：返回（训练）样本的标签

$maj(\cdot)$：返回众数

2. 机器学习算法--KNN

- **K-近邻(KNN)算法—KNN处理分类问题:步骤**

Document number	I	buy	an	apple	...	friend	has	emotion
train 1	1	1	1	1	...	0	0	happy
train 2	1	0	0	1	...	0	0	happy
train 3	0	0	0	1	...	0	0	sadness
test 1	0	0	1	1	...	1	1	?

2. 相似度计算：计算test1与每个train的距离

欧氏距离：

$$d(train1, test1) = \sqrt{(1-0)^2 + (1-0)^2 + \cdots + (0-1)^2} = \sqrt{6};$$

$$d(train2, test1) = \sqrt{(1-0)^2 + (1-0)^2 + \cdots + (0-1)^2} = \sqrt{8};$$

$$d(train3, test1) = \sqrt{(0-0)^2 + (0-0)^2 + \cdots + (0-1)^2} = \sqrt{9};$$

（也可以使用其他距离度量方式）

3. 类别计算：最相似的k个样本之标签的众数

若k=1，test1的标签即为train1的标签happy；

若k=3，test1的标签为train1,train2,train3的标签中数量较多的，即为happy。

6. 机器学习算法--KNN

- K-近邻(KNN)算法—KNN参数设置
- 采用不同的距离度量方式(见下一页)
- 通过验证集对参数(k值)进行调优
  - 如果k值取的过大，学习的参考样本更多，会引入更多的噪音，所以可能存在欠拟合的情况；
  - 如果k值取的过小，参考样本少，容易出现过拟合的情况
  - 关于k的经验公式：一般取$k=\sqrt{N}$，N为训练集实例个数，大家可以尝试一下
- 权重归一化

Name	Formula	Explain
Standard score	$X' = \frac{X-\mu}{\sigma}$	μ is the mean and σ is the standard deviation
Feature scaling	$X' = \frac{X-X_{min}}{X_{max}-X_{min}}$	X_{min} is the min value and X_{max} is the max value

6. 机器学习算法--KNN

- **K-近邻(KNN)算法—不同的度量公式**

距离公式：

L_p距离(所有距离的总公式)：

$$L_p(x_i, x_j)=\left\{\sum_{l=1}^{n}\left|x_i^{(l)}-x_j^{(l)}\right|^p\right\}^{\frac{1}{p}}$$

$p=1$：曼哈顿距离；

$p=2$：欧氏距离，最常见。

例 3.1 已知二维空间的 3 个点 $x_1=(1,1)^{\mathrm{T}}$, $x_2=(5,1)^{\mathrm{T}}$, $x_3=(4,4)^{\mathrm{T}}$，试求在 p 取不同值时，L_p 距离下 x_1 的最近邻点。

解 因为 x_1 和 x_2 只有第一维的值不同，所以 p 为任何值时，$L_p(x_1,x_2)=4$。而

$$L_1(x_1,x_3)=6,\quad L_2(x_1,x_3)=4.24,\quad L_3(x_1,x_3)=3.78,\quad L_4(x_1,x_3)=3.57$$

于是得到：p 等于 1 或 2 时，x_2 是 x_1 的最近邻点；p 大于等于 3 时，x_3 是 x_1 的最近邻点。 ■

余弦相似度：

$$\cos\left(\vec{A},\vec{B}\right)=\frac{\vec{A}\cdot\vec{B}}{\left|\vec{A}\right|\left|\vec{B}\right|}$$，**其中$\vec{A}$和$\vec{B}$表示两个文本特征向量；**

余弦值作为衡量两个个体间差异的大小的度量

为正且值越大，表示两个文本差距越小，为负代表差距越大，请大家自行脑补两个向量余弦值

6. 机器学习算法--KNN

- **6.1 K-近邻(KNN)算法—KNN算法效率**
- 假设训练集有N个样本，测试集有M个样本，每个样本是一个V维的向量。

- 如果使用线性搜索的话，那么k-NN的时间花销就是O(N*M*V)。

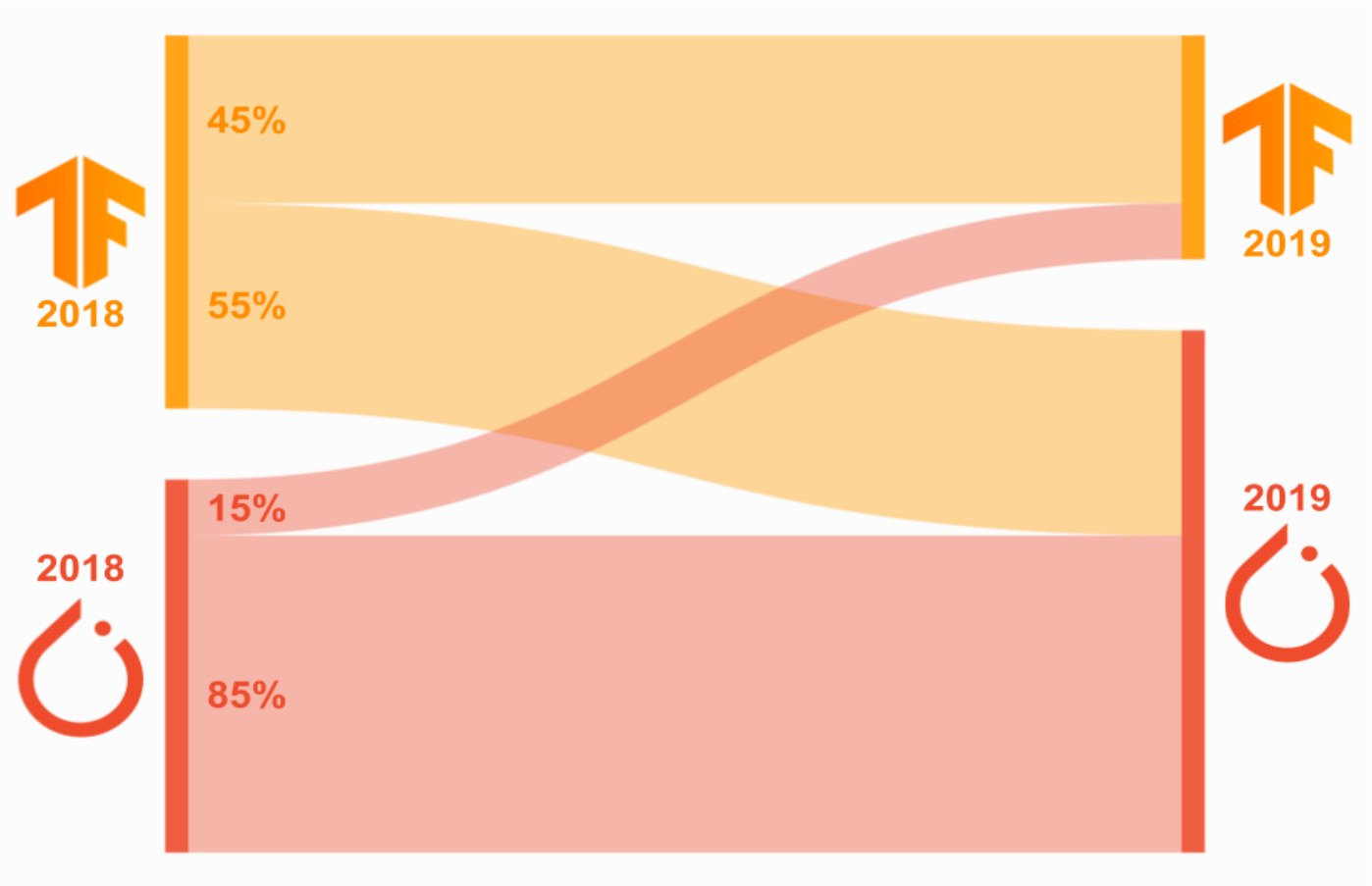
- 改善：KD树(感兴趣可自行尝试)

实验任务

- 用朴素贝叶斯算法和KNN算法完成分类任务。
- 详见【第6次作业.pdf】
- 本次作业作为练习，不需要提交

PyTorch V.S. TensorFlow

AI 顶会统计：2018年使用 TensorFlow 的研究者有 55% 转向使用 PyTorch，而使用 PyTorch 的研究者有 85% 选择继续使用 PyTorch。PyTorch 框架因其简洁易上手等特点近年来深受研究机构人员青睐，TensorFlow 则在大型工程项目中使用较多。



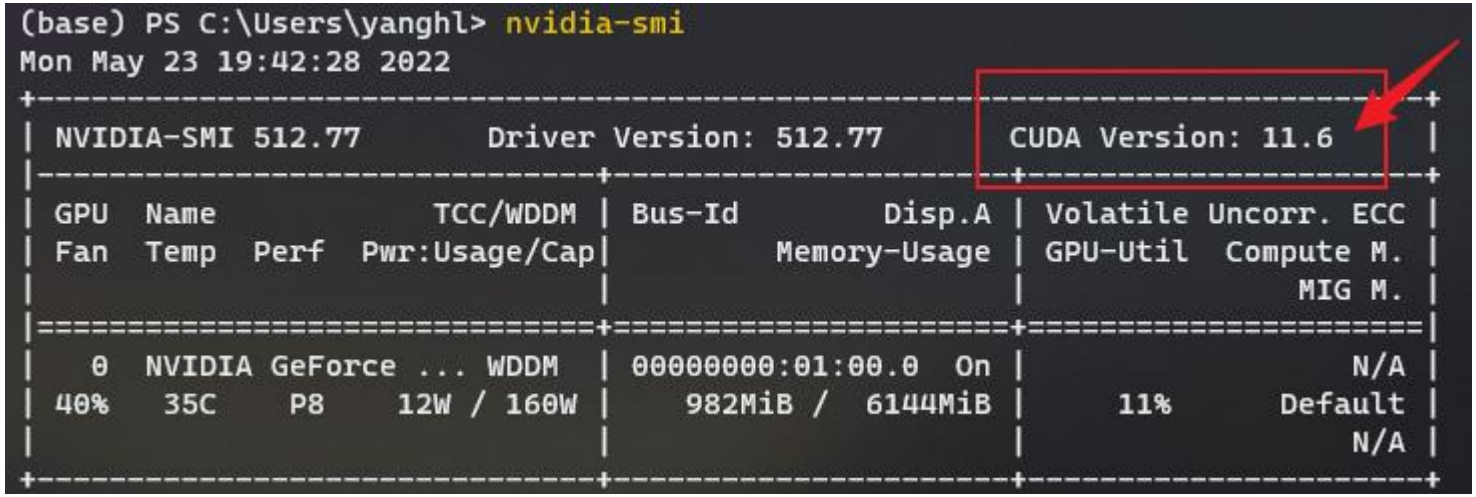
1.1 PyTorch 安装

官网：https://pytorch.org

CPU 版安装（无 Nvidia 显卡）：直接在官网主页选择配置，然后复制生成的 Command 粘贴到终端运行（注意需提前激活 Python 虚拟环境）。

1.1 PyTorch 安装

GPU 版安装：先在终端使用 nvidia-smi 命令查看当前显卡支持的 CUDA 版本



然后在主页选择 CUDA xx.x 生成 Command 命令，PyTorch CUDA 的版本不能高于显卡支持的 CUDA 版本。

Linux 和 MacOS 同理，在主页选择对应 OS 选项即可。

1.1 PyTorch 安装

安装完成后验证是否安装成功：在终端键入 python，进入 python 交互环境。

- CPU 版直接执行 import torch，如无报错即安装成功。
- GPU 版执行 import torch 后，执行 torch.cuda.is_available()，如返回 True 说明 GPU 版 PyTorch 安装成功。

```
(pytorch) PS C:\Users\yanghl> python
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.is_available()
True
```

PyTorch 安装到此结束，更多内容可参考官方文档：

https://pytorch.org/docs/stable/index.html