Big Data Processing: Homework 5

凌康伟 5140219295

May 23, 2017

1 程序说明

`knn.py` KNN 算法实现。

`perceptron.py` Perceptron 算法实现。

`analysis.ipynb` 分析及决策面作图。

2 结果截图

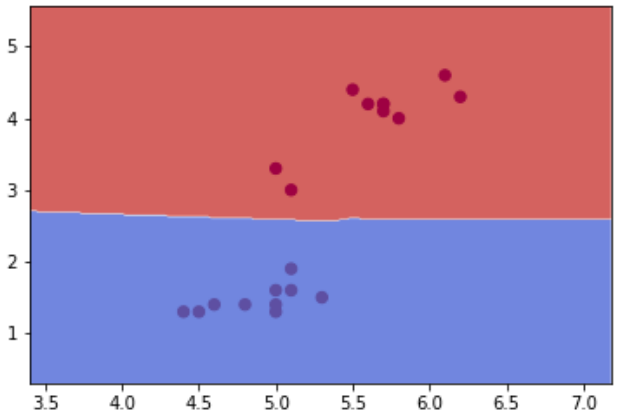

Figure 1: KNN - 1 nearest neighbor

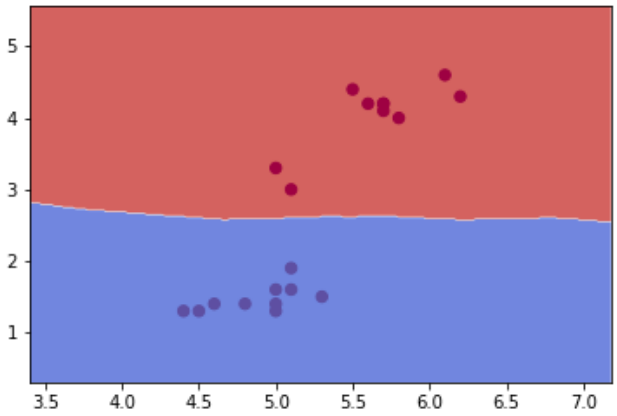

Figure 2: KNN - 3 nearest neighbor

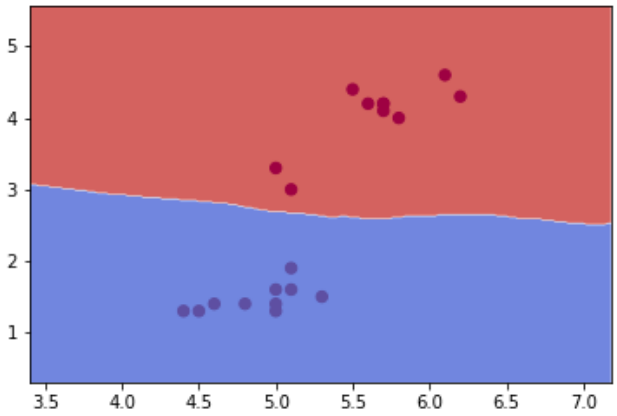

Figure 3: KNN - 6 nearest neighbor

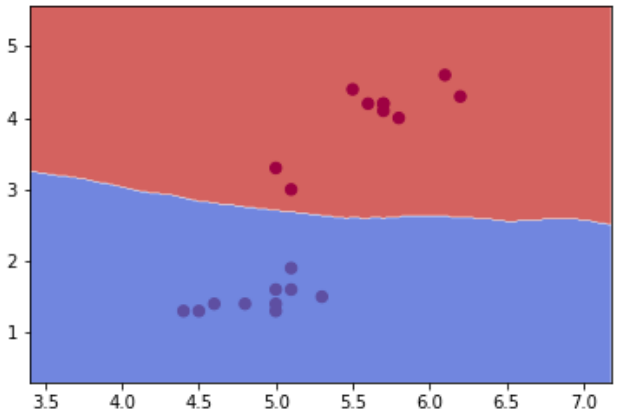

Figure 4: KNN - 10 nearest neighbor

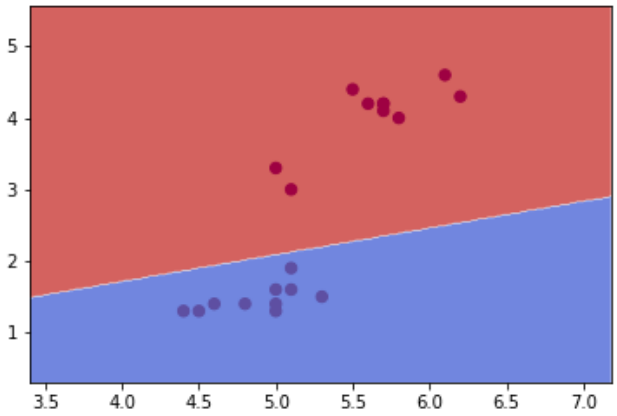

Figure 5: Perceptron

3 结果分析

从以上截图可以知道，无论是 KNN 算法还是 Percetron 算法，在提供的数据集上的分类效果都非常好，均达到了 100% 的准确率，但是分类面有很大差别。

对于 KNN 算法，得到的分界面呈不规则形状。当然，这是由于数据集的具体分布而造成的，同时这也说明 KNN 算法能对数据集有比较好的刻画能力。另一方面，KNN 的分界面也受具体多少近邻的选择。KNN 中 K 的取值越小，分类器的抗噪声干扰能力就越弱。

对于 Perceptron 算法，分界面是线性的，这一点由定义也可以得出（超平面）。与 KNN 算法的结果比较可以发现，其得出的分界面并不是最优的，因为一旦对训练集中数据分类没有错误后，就不再继续优化了，因此泛化能力比较弱。下面分别是分界面与原数据集的分布图。

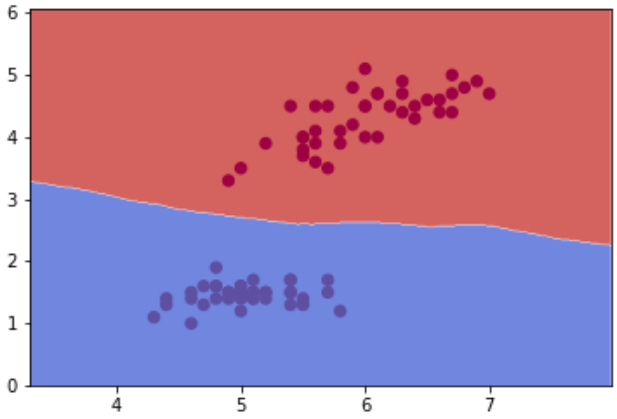

Figure 6: KNN - 10

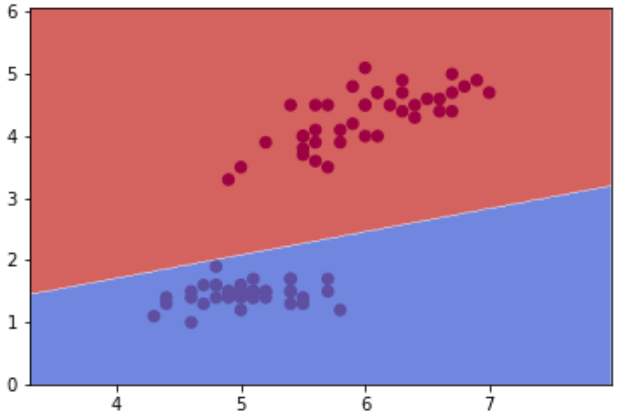

Figure 7: Perceptron

同时，KNN 算法和 Perceptron 算法的训练与分类用的时间也有很大差别。准确来说，KNN 算法不存在训练的步骤 (不包括对数据集的预处理等)，但是每次分类都需要对整个数据集遍历。而 Perceptron 算法一旦收敛后，就可以不再需要之前的训练数据了，其分类所需要的时间比较少，而且稳定。

总的来说，KNN 算法适用于以下场景：

- 数据复杂性非常高，非线性可分
- 数据集相对不大（performance - time tradeoff)

其局限性体现在：

- 计算代价高

- 无法解决样本不平衡问题
- 需要大量内存（存储 + 计算）

而 Perceptron 算法则适用于以下场景：

- 数据线性可分性非常好
- 数据规模大
- 在线学习

其局限性体现在：

- 模型的表达能力差
- 泛化能力一般