



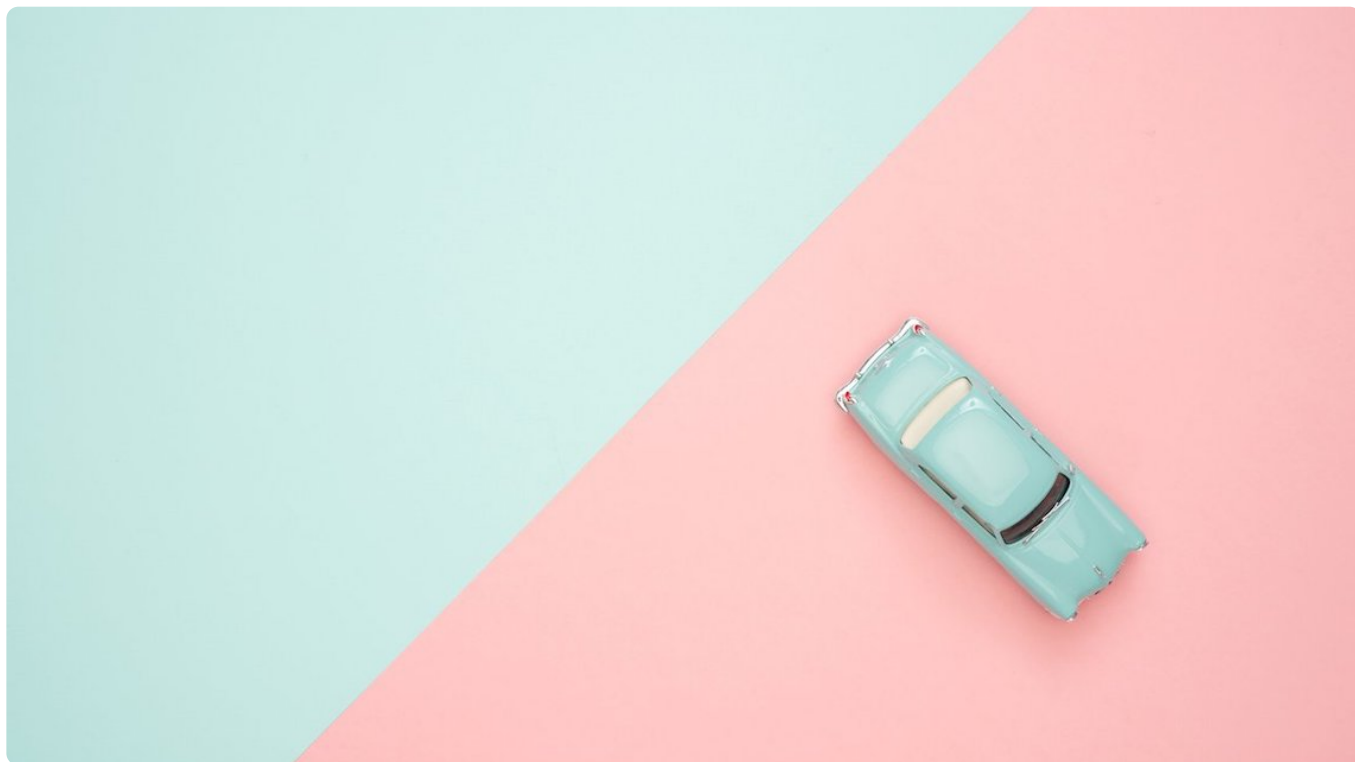
下载APP



01 | 导读：如何在机器学习中运用线性代数工具？

2020-07-27 朱维刚

重学线性代数


[进入课程 >](#)**讲述：朱维刚**

时长 07:32 大小 6.91M

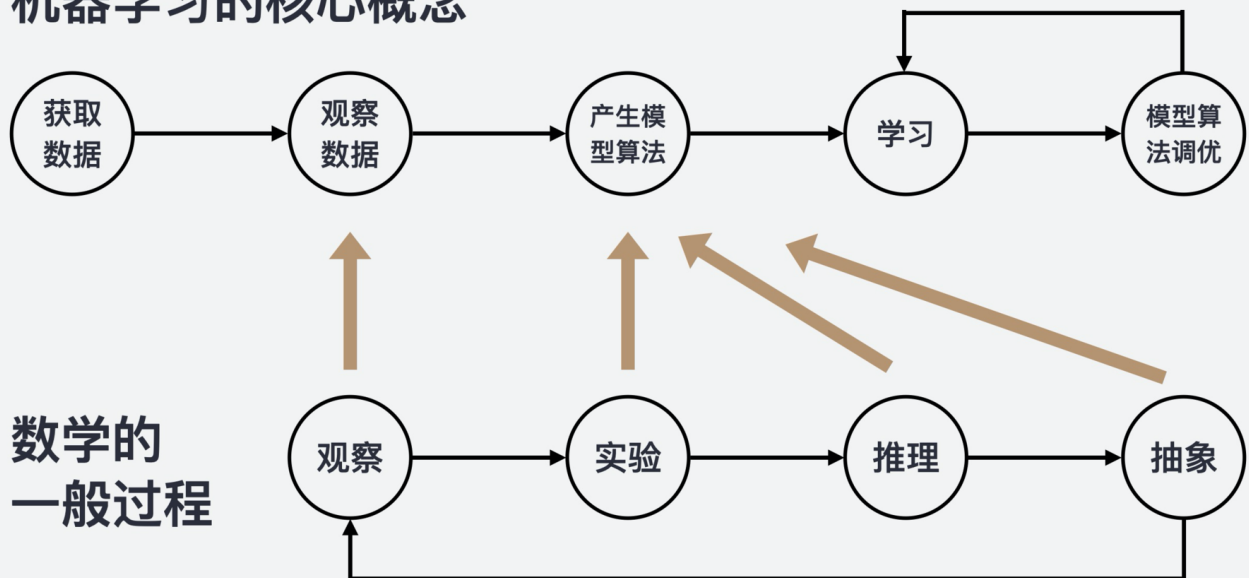


你好，我是朱维刚。欢迎你跟我一起重学线性代数！

在开篇词中，我和你大致讲过我自己的经历，从 2006 年开始到现在 14 年的时间里，我都专注于机器学习领域。对于**线性代数**在机器学习中的应用，我非常了解。而这也是线性代数最主要的应用场景之一。因此，今天第一节课，我想先和你聊一聊，如何在机器学习中运用线性代数工具，在我们开始自下而上的学习之前，先从上层来看一看。

我们都知道，“数据”是机器学习的前提，机器学习的第一步就是要进行**数据**的收集、处理和特征提取；而**模型**就是通过数据来学习的算法；**学习**则是一个循环过程，一个在数据中寻找模式，并不停调优模型参数的过程。那我们就从机器学习的三个核心概念：数据、模型和学习说起。

机器学习的核心概念



极客时间

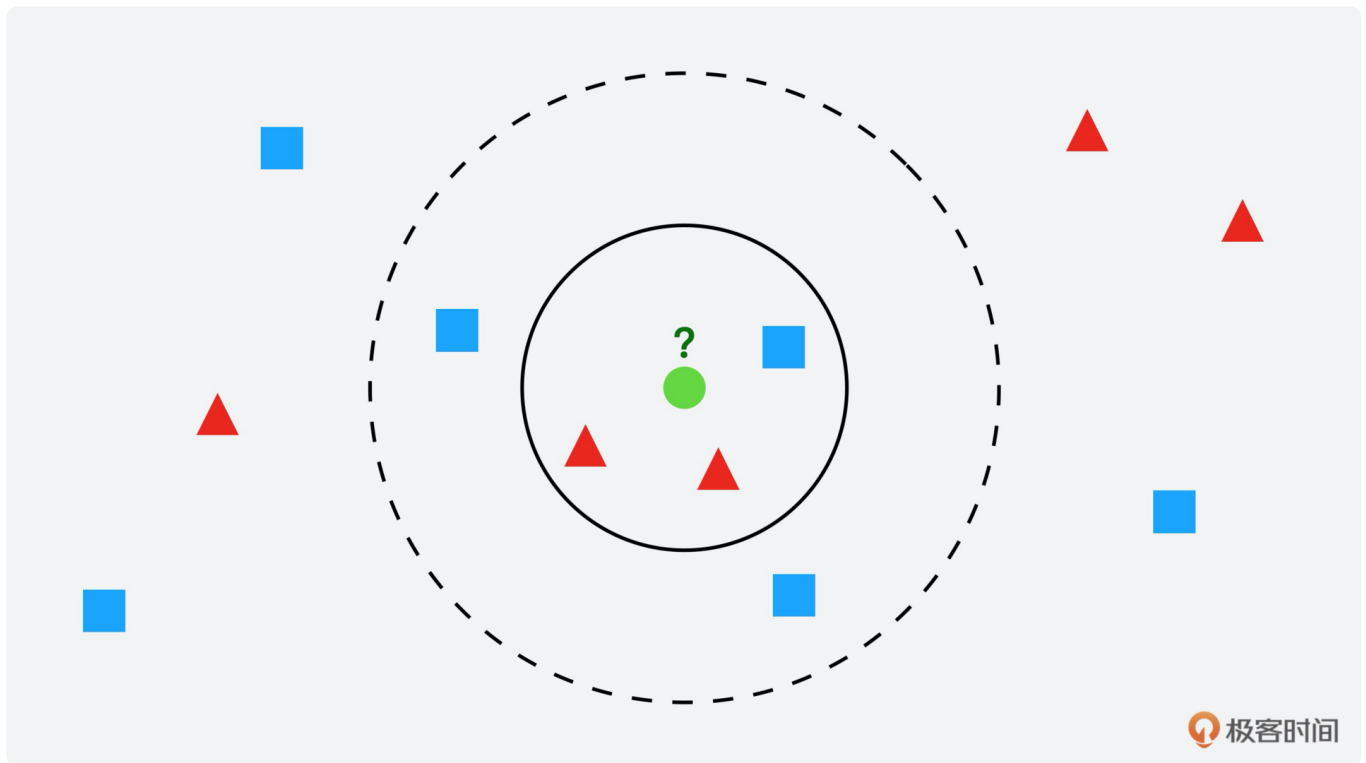
你看，不论是模型，还是学习，都涉及数据，而数据加上模型和学习，就是数学的一般过程了，也就是：观察、实验、推理和抽象。所以，我认为学好数学，不仅有利于理解复杂的机器学习系统，还能调优算法参数，甚至能帮助你创建新的机器学习解决方案。

从机器学习到线性代数

那机器学习和线性代数之间到底有着怎样的关系呢？我想，用一个实际的机器学习算法的例子来解释，你可能更容易搞清楚。接下来，我使用 KNN (K-Nearest Neighbor, K 最近邻分类算法) 来让你简单了解一下机器学习，以及它和线性代数之间的关系。

之所以选 KNN 分类算法，因为它是一个理论上比较成熟的方法，也是最简单的机器学习算法之一。这个方法的思路是：如果一个样本在特征空间中的 K 个最相似（即特征空间中最邻近）的样本中的大多数属于某一个类别，则该样本也属于这个类别。

这里有个前提，KNN 算法中，所选择的“邻居”都是已经正确分类的对象。KNN 分类算法在分类决策上只依据最邻近的一个或者几个样本的类别，来决定待分样本所属的类别。我们通过图来理解的话或许更容易一些。



假设图片中那个绿色圆就要是我们要决策的对象，那么根据 KNN 算法它属于哪一类？是红色三角形还是蓝色四方形？

如果 $K=3$ （实线圆），也就是包含离绿色圆最近的 3 个，由于红色三角形所占比例为 $2/3$ ，绿色圆就属于红色三角形那个类。但如果 $K=5$ （虚线圆），就是包含离绿色圆最近的 5 个，由于蓝色四方形比例为 $3/5$ ，绿色圆就属于蓝色四方形那个类。


鸢尾花分类问题中的线性代数

通过前面这个小例子，你应该已经理解了 KNN 算法的概念。那么接下来，我们就试着使用 KNN 在给定鸢尾花特征值的情况下，给鸢尾花做花种分类，带你来实际看一下线性代数在这里起到的作用。

特别说明一下，**鸢尾花分类问题**是一个国际上通用的案例，一般都被作为机器学习入门来使用，所以它的数据集也是公开的。


1. 数据集的收集、加载和分析

首先，我们要做的是数据集的收集、加载和分析，你也可以点击 [这里](#) 下载原始数据集，来看看原始数据长什么样，下面是获取和加载数据的代码，sklearn 数据集已经包含了样本数据，你可以直接用。

 复制代码


```
1 import pandas as pd
2
3 from sklearn import datasets
4 iris = datasets.load_iris()
5
6 species = [iris.target_names[x] for x in iris.target]
7
8 iris = pd.DataFrame(iris['data'], columns = ['Sepal_Length', 'Sepal_Width', 'P
9
10 iris['Species'] = species
```

从显示的结果，我们能够看出鸢尾花有四个特征：花萼的长、宽和花瓣的长、宽。我们来看下这四个特征的数据类型：

 复制代码

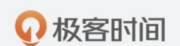
```
1 iris.dtypes
2 Sepal_Length    float64
3 Sepal_Width     float64
4 Petal_Length    float64
5 Petal_Width     float64
6 Species         object
7 dtype: object
```

这些特征都是数值型，而且标签 Species 表示的是花种，是一个字符串类型的变量。我们继续看一下鸢尾花的分类统计：

 复制代码

```
1 iris['count'] = 1
2 iris[['Species', 'count']].groupby('Species').count()
```

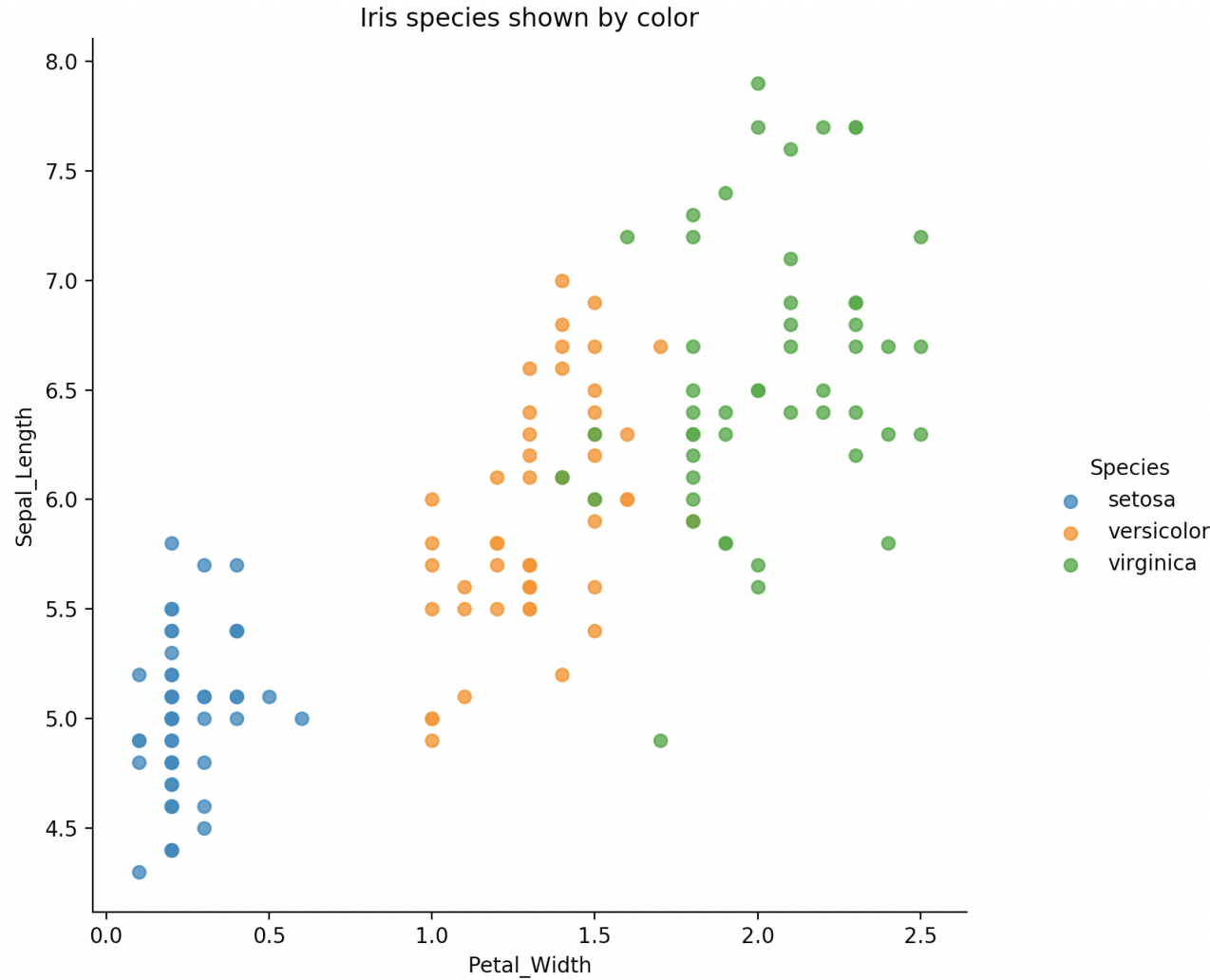
Species	count
setosa	50
versicolor	50
virginica	50

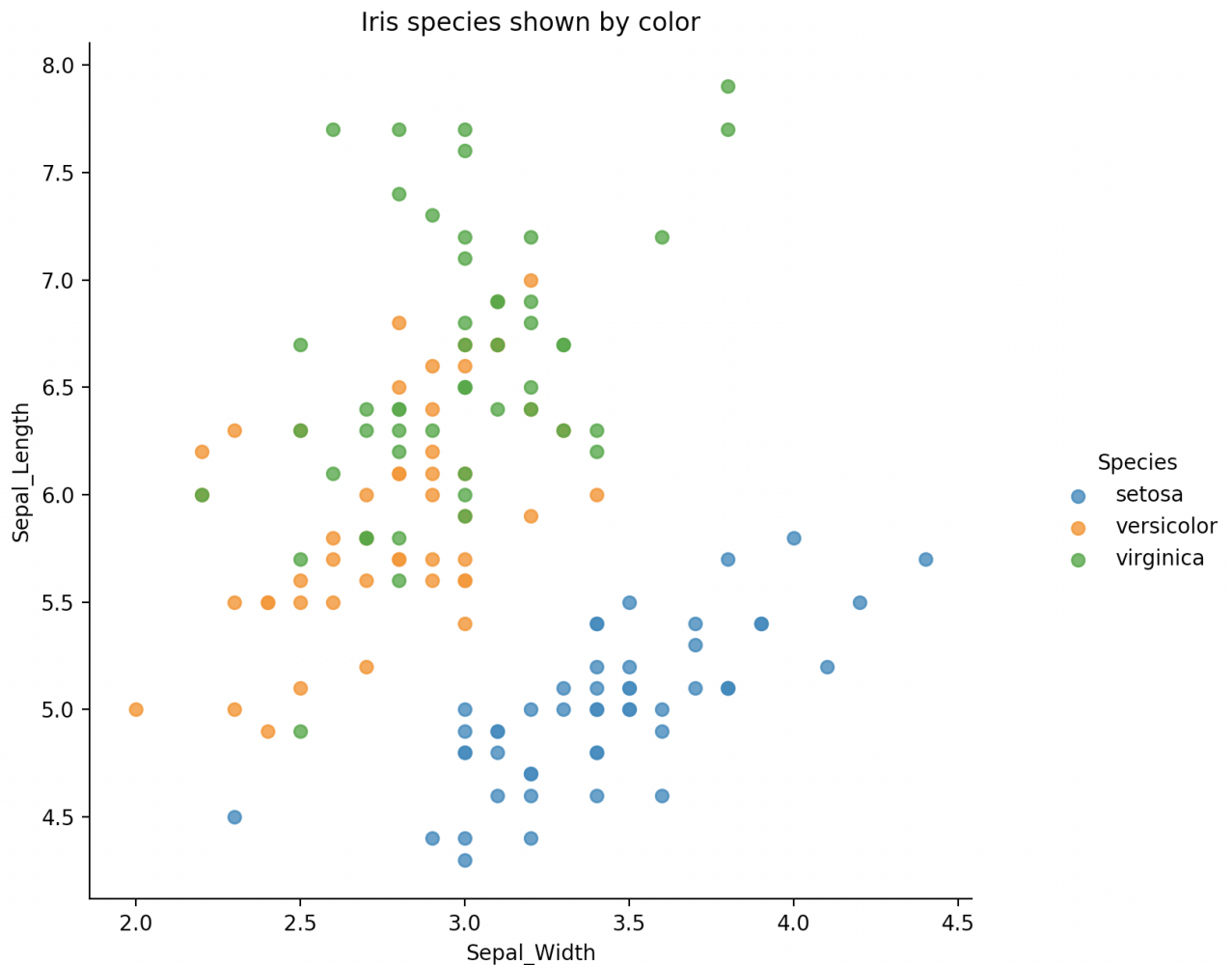


这里我们直接能够看到，鸢尾花有三个花种，每个种类有 50 个实例，或者说 50 条数据，我们再用图来更直观地显示这三种鸢尾花。

[复制代码](#)

```
1 %matplotlib inline
2
3 def plot_iris(iris, col1, col2):
4     import seaborn as sns
5     import matplotlib.pyplot as plt
6
7     sns.lmplot(x = col1, y = col2,
8               data = iris,
9               hue = "Species",
10              fit_reg = False)
11
12     plt.xlabel(col1)
13
14     plt.ylabel(col2)
15
16     plt.title('Iris species shown by color')
17
18     plt.show()
19
20 plot_iris(iris, 'Petal_Width', 'Sepal_Length')
21
22 plot_iris(iris, 'Sepal_Width', 'Sepal_Length')
```





蓝、黄、绿，这三种颜色分别代表了三种鸢尾花，显示还是很清楚的。

2. 数据集的准备

接下来的第二步就是数据集的准备了。在训练任何机器学习模型前，数据准备都相当重要，这里也要涉及两步准备。

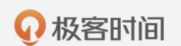
第一步，特征数值标准化。如果我们不做标准化，后果就是大数值特征会主宰模型训练，这会导致更有意义的小数值特征被忽略。这里我们用 Z Score 标准化，使每一类特征平均值为 0，方差为 1.0，我们可以通过代码实现来看下效果。

复制代码

```
1 from sklearn.preprocessing import scale
2
3 import pandas as pd
4
5 num_cols = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']
6
7 iris_scaled = scale(iris[num_cols])
```

```
8 iris_scaled = pd.DataFrame(iris_scaled, columns = num_cols)
9
10 print(iris_scaled.describe().round(3))
11
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	150.000	150.000	150.000	150.000
mean	-0.000	-0.000	-0.000	-0.000
std	1.003	1.003	1.003	1.003
min	-1.870	-2.434	-1.568	-1.447
25%	-0.901	-0.592	-1.227	-1.184
50%	-0.053	-0.132	0.336	0.133
75%	0.675	0.559	0.763	0.791
max	2.492	3.091	1.786	1.712

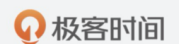


你可以看到，每一列平均值为 0，标准差大约是 1.0。为了分类需要，我们用字典把花种从字符串类型转换成数字表示。

[复制代码](#)

```
1 levels = {'setosa':0, 'versicolor':1, 'virginica':2}
2
3 iris_scaled['Species'] = [levels[x] for x in iris['Species']]
4
5 iris_scaled.head()
```


	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
0	-0.900681	1.019004	-1.340227	-1.315444	0
1	-1.143017	-0.131979	-1.340227	-1.315444	0
2	-1.385353	0.328414	-1.397064	-1.315444	0
3	-1.506521	0.098217	-1.283389	-1.315444	0
4	-1.021849	1.249201	-1.340227	-1.315444	0



第二步，把数据集随机分割成样本训练集和评估数据集，训练集用来训练 KNN 模型，评估集用来测试和评估 KNN 的分类结果。

[复制代码](#)

```
1 from sklearn.model_selection import train_test_split
2
3 import numpy as np
4
5 np.random.seed(3456)
6
7 iris_split = train_test_split(np.asmatrix(iris_scaled), test_size = 75)
8
9 iris_train_features = iris_split[0][:, :4]
10
11 iris_train_labels = np.ravel(iris_split[0][:, 4])
12
13 iris_test_features = iris_split[1][:, :4]
14
15 iris_test_labels = np.ravel(iris_split[1][:, 4])
16
17 print(iris_train_features.shape)
18
19 print(iris_train_labels.shape)
20
21 print(iris_test_features.shape)
22
23 print(iris_test_labels.shape)
```


通过代码，我们得到了下面这样的结果。

 复制代码

```
1 (75, 4)
2 (75,)
3 (75, 4)
4 (75,)
```

3. 训练模型


数据准备好后，就是第三步训练模型了。这里我们使用 K=3 来训练 KNN 模型，当然你也可以调整这个参数来进行观察和调优。

 复制代码

```
1 from sklearn.neighbors import KNeighborsClassifier
2
3 KNN_mod = KNeighborsClassifier(n_neighbors = 3)
4
5 KNN_mod.fit(iris_train_features, iris_train_labels)
```

4. 模型测试

执行 KNN 训练后，我们来到了最后一步，模型测试，这里我们使用测试集来测试模型。


 复制代码

```
1 iris_test = pd.DataFrame(iris_test_features, columns = num_cols)
2
3 iris_test['predicted'] = KNN_mod.predict(iris_test_features)
4
5 iris_test['correct'] = [1 if x == z else 0 for x, z in zip(iris_test['predicted'], iris_test['actual'])]
6
7 accuracy = 100.0 * float(sum(iris_test['correct'])) / float(iris_test.shape[0])
8
9 print(accuracy)
```

 复制代码

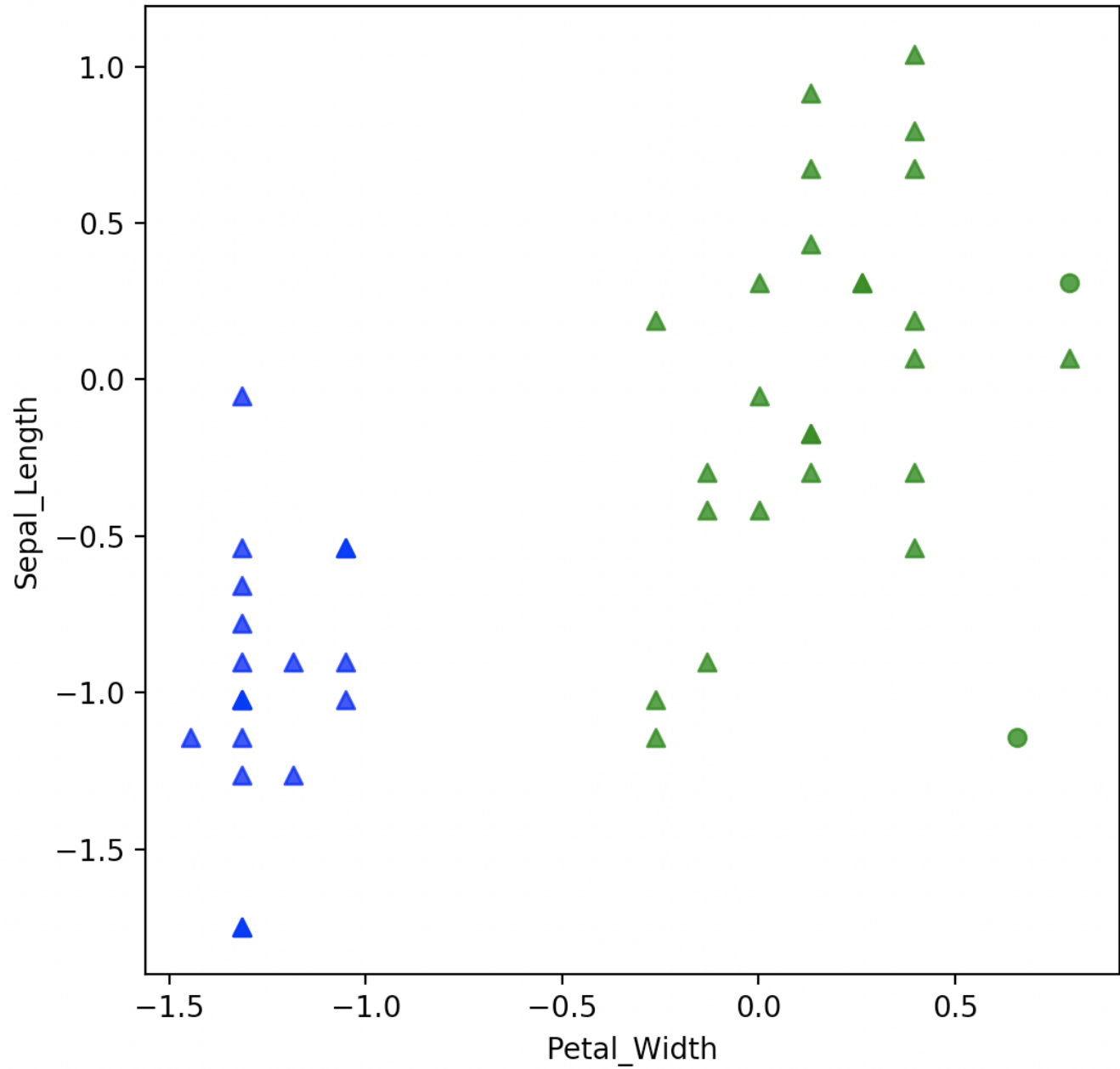
```
1 96.0
```

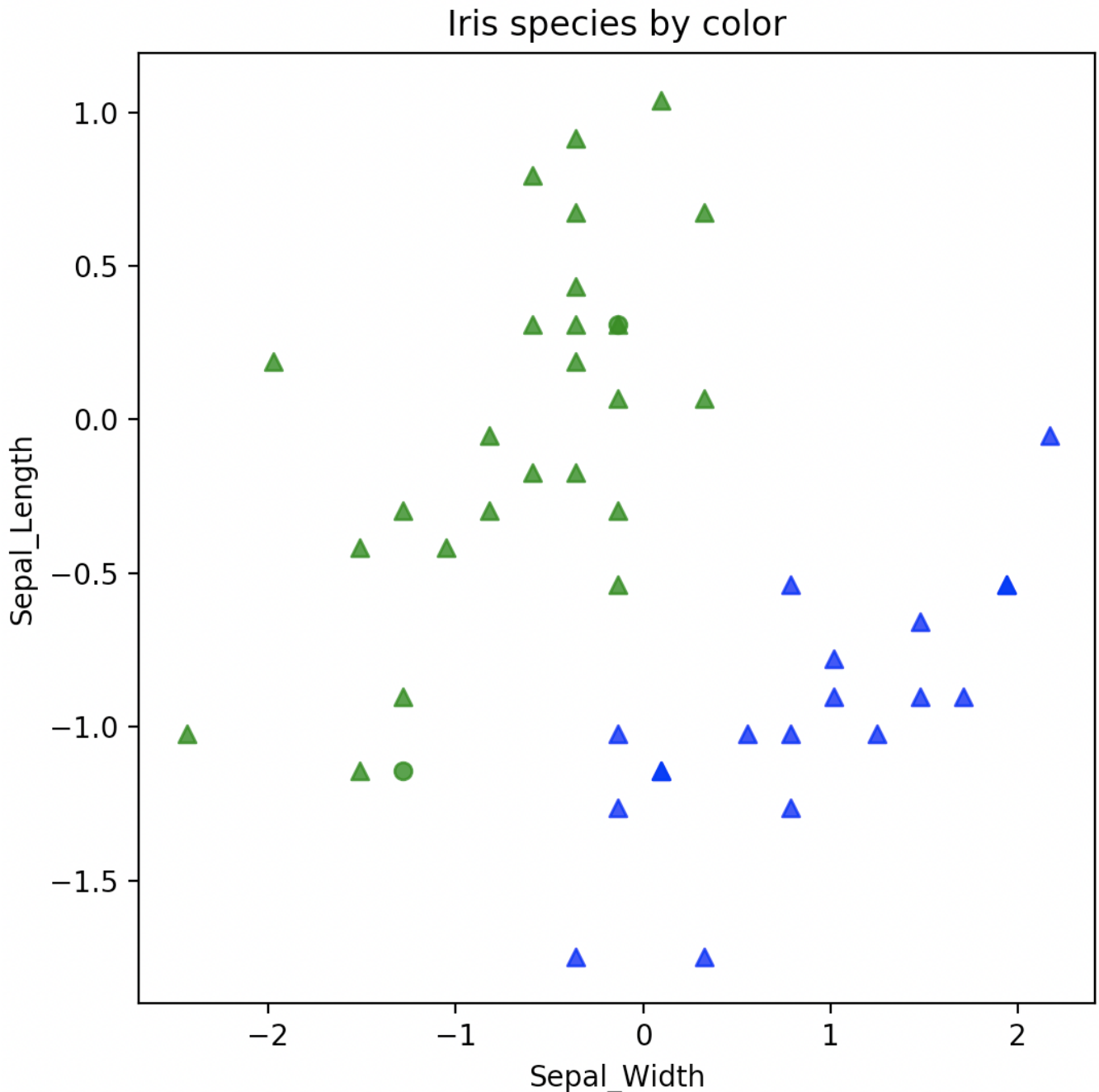
最终，我们得到的准确率是 96.0，说明了 KNN 的训练模型不错，适用这类场景。我们通过代码把其中的两个分类 setosa 和 versicolor 打印出来看看。

 复制代码

```
1 levels = {0:'setosa', 1:'versicolor', 2:'virginica'}
2
3 iris_test['Species'] = [levels[x] for x in iris_test['predicted']]
4
5 markers = {1:'^', 0:'o'}
6
7 colors = {'setosa':'blue', 'versicolor':'green',}
8
9 def plot_shapes(df, col1,col2, markers, colors):
10     import matplotlib.pyplot as plt
11     import seaborn as sns
12
13     ax = plt.figure(figsize=(6, 6)).gca() # define plot axis
14
15     for m in markers: # iterate over marker dictionary keys
16         for c in colors: # iterate over color dictionary keys
17             df_temp = df[(df['correct'] == m) & (df['Species'] == c)]
18             sns.regplot(x = col1, y = col2,
19                         data = df_temp,
20                         fit_reg = False,
21                         scatter_kws={'color': colors[c]},
22                         marker = markers[m],
23                         ax = ax)
24     plt.xlabel(col1)
25     plt.ylabel(col2)
26     plt.title('Iris species by color')
27     return 'Done'
28
29 plot_shapes(iris_test, 'Petal_Width', 'Sepal_Length', markers, colors)
30 plot_shapes(iris_test, 'Sepal_Width', 'Sepal_Length', markers, colors)
```

Iris species by color





从显示的效果来说，分类还是挺明显的，熟悉了最基础的机器学习过程后，你可能会问，讲了半天，线性代数到底在哪里呢？关键就在 `KNeighborsClassifier` 模块上，这个模型算法的实现背后，其实用到了线性代数的核心原理。

首先，因为每种鸢尾花都有四个特征：花萼的长、宽和花瓣的长、宽，所以每条数据都是四维向量。

接着，量化样本之间的相似度，也就是计算向量之间的距离。而向量之间距离的运算有很多方式，比如：曼哈顿距离、欧式距离、切比雪夫距离、闵可夫斯基距离等等。其中，欧式距离你应该很熟悉了，因为我们初中都学过，在二维平面上计算两点之间的距离公式：

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

扩展到我们实例中的四维向量，也是同样的算法。

你看，这就是线性代数在机器学习中的一种应用场景。KNN 是一种监督学习算法，因为在样本集中有分类信息，通过计算距离来衡量样本之间相似度，算法简单，易于理解和实现。还有另一种机器学习算法是无监督学习，底层的数学原理其实也是差不多的，总的思想就是“物以类聚”。

现在，你是不是有一种豁然开朗的感觉？终于看到了线性代数原来那么有意义，而且再简单的公式也是美的。

本节小结

好了，到这里导读这一讲就结束了，最后我再总结一下前面讲解的内容。

这一讲我使用机器学习的监督学习算法 KNN，在给定鸢尾花特征值的情况下，给鸢尾花做花种分类，让你了解机器学习最基本的过程外，能够真正了解其背后的线性代数真相，为你进入后面课程的学习提供一个感性的认知。

机器学习中用到的线性代数知识点比比皆是，而且往往软件架构上看上去复杂的事情，在数学上反而很简单，希望你在学习了这门课程后，能够多从数学角度出发去构思解决问题的方案。

同时，欢迎你在留言区说说自己对机器学习的理解，也可以分享一下自己的线性代数学习经历，如果你有所收获，也欢迎你把这篇文章分享给你的朋友。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 开篇词 | 从今天起，学会线性代数

下一篇 02 | 基本概念：线性代数研究的到底是什么问题？

精选留言 (1)

写留言



ququwowo

2020-07-28

最近刚开始学拿Stan/pymc3做贝叶斯机器学习，很多情况下需将模型re-parameterize (主要搞成non-centered)，方便MCMC采后验。几乎总用到cholesky factorization，希望老师后面能讲到。

作者回复: 你好，ququwowo，对于pymc3我不熟悉，可以尝试一下变分推理，有机会也可以另外做个专题来探讨大规模概率编程。



4