ML

Python token ghp\_Cmi4z2vjccBMzxjQepLxkTKdc3kUHK0lzDM2

数据 模型 目标

数据(猫猫狗狗的照片) 🡪 目标(减速)

1.Typical work flow ： 定义问题

2.获取数据

3.training

4.评估

4-1 benchmark 去评估

4-2 更好

5.上线 deployment

6．监控

7. 持续提升 & go to step2

环境： machine learning training

编程语言 & library & IDE(PyCharm)

Library🡪 conda (神经网络) & torch

Version control 🡪 git (github)

Create，上传，rollback

**MNIST** 识别手写字母的CV算法

#Iris classification

Numpy & Pytorych

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

装CONDA 装GitHub

Version control 回滚

2个问题 都是python的特殊语法

下划线写在前面的function

@property 是什么意思

@property 是 Python 中的一个装饰器，用来将一个方法“伪装成”属性，让你可以像访问属性一样调用它，而不需要加括号

Google drive OR Dropbox 别人可以上传

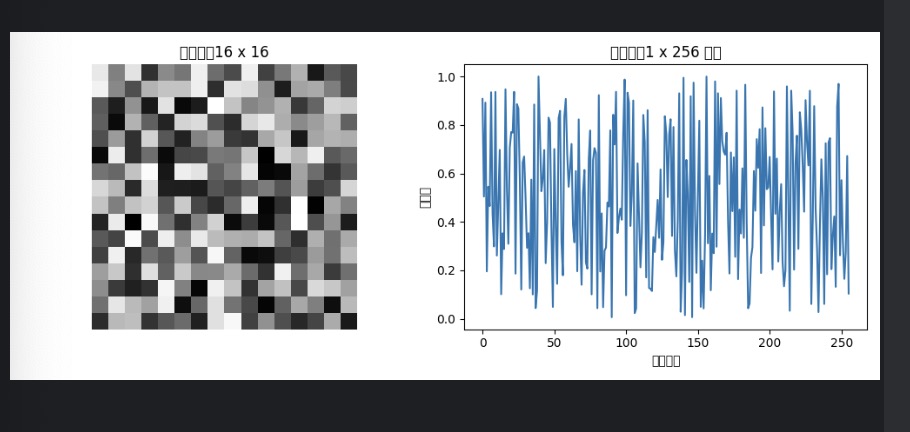
PyCharm 里USPS每一行读懂什么意思

明天早上

可视化 Flatten 过程可视化 Flatten 过程：

torch.Size([1, 16, 16])

torch.Size([1, 256])



**“让他先乱猜数字是几，然后跟 label 进行对照，然后他自己学会其中规律，自主判断”**

没错，这就是深度学习中最基础也是最重要的“监督学习”思想。

“清空 → 反向传播 → 更新”

optimizer.zero\_grad()

loss.backward() ？ why这么神奇 全自动？

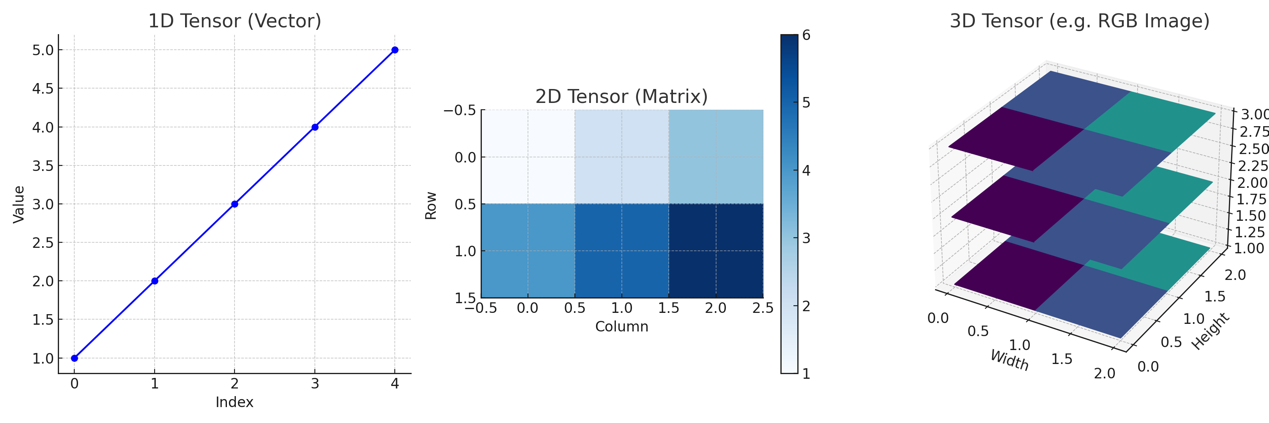
optimizer.step()

Question：

张量tensor是什么，他跟数列array，或者矩阵matrix区别是什么

tensor：是 **0 维、1 维、2 维、3 维、n 维都可以的**通用表达

**array 和 matrix 是 tensor 的“子集”**  
PyTorch 统一都叫它 tensor，因为它支持任意维度。



git add X.py

git commit -m “balabala in X.py”

git push origin main



每个 GitHub 仓库（repository）可以托管一个项目。

本地项目通过 git remote add origin <远程地址> 与某个 GitHub 仓库绑定。

你一个 GitHub 账号可以有很多仓库，但一个本地 Git 项目**通常只连接一个远程 origin**。

**…or create a new repository on the command line**

echo "# Garbage" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/Air-ZZH/Garbage.git

git push -u origin main

**push an existing repository from the command line**

git remote add origin https://github.com/Air-ZZH/Garbage.git

git branch -M main

git push -u origin main

**Git 回滚**是指你对代码做出修改或提交之后，想撤销

文件已经被我删除，但我想恢复

git log -- python03\_garbage.py

出现 commit c89e1d410ab97474832a11359c42737ffcfafdd1 🡪也就是说：**你删除了它，但它曾经存在过，现在可以恢复**

git checkout c89e1d4 -- python03\_garbage.py 🡪 你从那个提交中单独把 python03\_garbage.py 文件“拽”回来

✅ 情境 1：**改了代码还没提交，想撤销**

git restore python03\_garbage.py

会把这个文件还原成上一次 commit 的版本

print('balabala')  
print(123)

执行后变回print('balabala')

✅ 情景 2：已经 commit（甚至 push）了，想“撤销”这次提交

git log –oneline

69f539f (**HEAD -> main**, **origin/main**) whatever

git revert 69f539f –-no-edit

周日 11pm

IRIS dataset 表格类数据shape dtype **跑通**

Tensor #属性shape dtype strides

Dtype 用什么精度来表示 默认float32，2^ 32，第一位表示+-，剩下表示大小

用的bit越多，刻画的越仔细，尤其用在[-1,1] 占用32个内存

32位—32 bit--32位 🡪 4 byte

Byte 字节 1 byte = 8 bit，多少K，兆，指的都是字节，不是字符

了解 字节 和字符(bit) 的概念

Tensor 🡪 Shape 4个

Epoch 是访问数据集的次数，epoch = 5，整个数据集跑了5次（测试集永远只跑1次）

Iteration 迭代次数 = Sample size / Batch size

Batch size 一批的大小

Sample size 训练集的大小

Case study:

Sample size 7000, batch size 64, 一个epoch里迭代次数 = 7000 / 64 约=100

5个epoch就是500次

Generally speaking, Train accuracy 要高于 test accuracy

看什么时候模型学不到什么知识了，仅仅是记住这个答案了 🡪 过拟合

没有拟合到知识，拟合到了不该拟合到的噪声，可能无脑都选A

过拟合的话，test accuracy 会下降

欠拟合：没学到位，epoch = 1的时候就是欠拟合

IRIS Classification

three classes of flowers, **Versicolor, Setosa, Virginica**

each class contains 4 features, ‘Sepal length’, ‘Sepal width’, ‘Petal length’, ‘Petal width’.

algorithm called “Support vector machine”. sVMs are one of the most robust classifications methods.

**Hyperplane（超平面）就是把空间“切开”的一条线或一个面**，它是用来**把不同类别的数据分开的“边界”**，n 维空间中的一个 n−1 维的“切割面”， 支持向量机（SVM）中：

你想把两类数据区分开（比如正类和负类），SVM 就会找一条**最优的 hyperplane**，这个 hyperplane 会**最大限度地分开两类数据**，并让它们之间的距离（margin）尽可能大。

SVM algorithm finds the points closest to the line from both classes. These points are known as support vectors. Then it computes the distance between the line and support vectors. This distance is called the margin. The main goal is to maximize the margin. The hyperplane which has the maximum margin is known as the optimal hyperplane. SVM mainly supports binary classification natively，multiclass classification可以反复使用SVM.

Numpy 🡪 computational operations.

Matplotlib and seaborn 🡪 data visualization

Pandas help to load data from various sources like local storage, database, excel file, CSV file, etc.

Stochastic Gradient Descent（SGD）

前向传播（forward pass）y\_pred = model(X\_train) 计算模型的输出。

计算损失（loss）：loss = loss\_fn(y\_pred, y\_train) 评估预测与标签的差距。

反向传播（backward pass）：loss.backward( ) 计算梯度。

参数更新optimizer.step( )用 SGD 或 Adam 等优化器更新权重。

