# 适合 NumPy MLP 的数据集

以下是 **适合用 NumPy 手动实现 MLP（多层感知机）** 的实验数据集推荐，涵盖 **分类、回归、二分类、多分类** 场景，均具备 **数据量适中、特征简单、标签明确** 的特点，无需复杂预处理，可直接用于验证 MLP 的核心逻辑（前向传播、反向传播、梯度下降）。

## 一、内置轻量数据集（无需额外下载，NumPy 可直接生成/加载）

### 1. 二分类数据集

#### （1） moons（月牙形数据集）

- **特点**：非线性可分（两个月牙形簇），适合验证 MLP 的“非线性拟合能力”（单层感知机无法解决，MLP 加隐藏层可解决）。
- **数据规模**：默认 1000 个样本，2 个特征（x1, x2），标签为 0/1。
- **NumPy 生成代码**：

```
import numpy as np
from sklearn.datasets import make_moons

# 生成数据集（noise 控制噪声，random_state 固定随机种子）
X, y = make_moons(n_samples=1000, noise=0.1, random_state=42)
# 标签 reshape 为(1000, 1)（适配 MLP 输出维度）
y = y.reshape(-1, 1)
# 数据归一化（可选，加速训练）
X = (X - X.mean(axis=0)) / X.std(axis=0)

print("X shape:", X.shape)  # (1000, 2)
print("y shape:", y.shape)  # (1000, 1)
```

### （2） circles（环形数据集）

- **特点**：同心圆结构，非线性可分，比 moons 更考验 MLP 的特征映射能力。
- **NumPy 生成代码**：

```
from sklearn.datasets import make_circles


X, y = make_circles(n_samples=1000, noise=0.05, factor=0.5,
random_state=42)
y = y.reshape(-1, 1)
X = (X - X.mean(axis=0)) / X.std(axis=0)


print("X shape:", X.shape)  # (1000, 2)
print("y shape:", y.shape)  # (1000, 1)
```

### （3）逻辑回归经典数据集（人工生成）

- **特点**：线性可分/弱非线性，适合快速验证 MLP 的基本训练流程（如梯度下降收敛、准确率提升）。
- **NumPy 生成代码**：

```
# 生成两类正态分布数据
np.random.seed(42)
# 类别 0：均值(0,0)，方差 0.5；类别 1：均值(2,2)，方差 0.5
X0 = np.random.multivariate_normal(mean=[0, 0], cov=[[0.5, 0], [0,
0.5]], size=500)
X1 = np.random.multivariate_normal(mean=[2, 2], cov=[[0.5, 0], [0,
0.5]], size=500)
X = np.vstack([X0, X1])  # 合并特征 (1000, 2)
y = np.hstack([np.zeros(500), np.ones(500)]).reshape(-1, 1)  # 标
签 (1000, 1)


# 打乱数据
shuffle_idx = np.random.permutation(len(X))
```

```
X, y = X[shuffle_idx], y[shuffle_idx]
```

## 2. 多分类数据集

### （1） iris（鸢尾花数据集）

- **特点**：经典多分类任务，3 个类别（鸢尾花品种），4 个数值特征，数据量小（150 样本），适合快速验证 MLP 的多分类逻辑（softmax 输出层+交叉熵损失）。
- **NumPy 加载代码**：

```
from sklearn.datasets import load_iris

# 加载数据
iris = load_iris()
X = iris.data  # 特征 (150, 4)：花萼长度、花萼宽度、花瓣长度、花瓣宽度
y = iris.target  # 标签 (150,)：0=山鸢尾，1=变色鸢尾，2=维吉尼亚鸢尾

# 标签独热编码（适配多分类交叉熵损失）
y_onehot = np.eye(3)[y]  # (150, 3)

# 数据归一化（关键：特征量纲不同，需标准化）
X = (X - X.mean(axis=0)) / X.std(axis=0)

print("X shape:", X.shape)        # (150, 4)
print("y_onehot shape:", y_onehot.shape)  # (150, 3)
```

### （2） blobs（人工多分类簇）

- **特点**：可自定义类别数、特征数，数据分布均匀，适合测试 MLP 在不同类别数下的性能。
- **NumPy 生成代码**：

```
from sklearn.datasets import make_blobs
```

```
# 生成 4 分类数据集，5 个特征，1000 样本
X, y = make_blobs(
    n_samples=1000, n_features=5, centers=4,
    cluster_std=1.0, random_state=42
)
y_onehot = np.eye(4)[y]  # 独热编码 (1000, 4)
X = (X - X.mean(axis=0)) / X.std(axis=0)  # 归一化

print("X shape:", X.shape)        # (1000, 5)
print("y_onehot shape:", y_onehot.shape)  # (1000, 4)
```

## 3. 回归数据集

### （1） 人工非线性回归数据

- **特点**：基于正弦函数+噪声生成，适合验证 MLP 的回归拟合能力（MSE 损失）。
- **NumPy 生成代码**：

```
np.random.seed(42)
X = np.linspace(-np.pi, np.pi, 1000).reshape(-1, 1)  # 特征 (1000, 1)
y = np.sin(X) + 0.1 * np.random.randn(*X.shape)  # 标签：正弦函数+噪声 (1000, 1)

# 数据归一化
X = (X - X.mean(axis=0)) / X.std(axis=0)

print("X shape:", X.shape)  # (1000, 1)
print("y shape:", y.shape)  # (1000, 1)
```

### （2） diabetes（糖尿病数据集）

- **特点**：真实回归任务，10 个特征（如年龄、体重、血压），目标是预测糖尿病进

展指标，适合测试 MLP 在真实数据上的回归性能。

- **NumPy 加载代码**:

```
from sklearn.datasets import load_diabetes

diabetes = load_diabetes()
X = diabetes.data  # 特征 (442, 10)
y = diabetes.target.reshape(-1, 1)  # 标签 (442, 1)：糖尿病进展值

# 归一化（回归任务对特征缩放敏感）
X = (X - X.mean(axis=0)) / X.std(axis=0)

print("X shape:", X.shape)  # (442, 10)
print("y shape:", y.shape)  # (442, 1)
```

## 二、数据集选择建议（按实验目标）

| 实验目标 | 推荐数据集 | MLP 配置要点 |
| --- | --- | --- |
| **验证 MLP 非线性拟合能力** | moons / circles | 隐藏层≥1 层（如 2 层：2→10→1），激活函数用 sigmoid/tanh |
| **验证多分类逻辑（softmax）** | iris / blobs（3-5 类） | 输出层维度=类别数，损失函数用交叉熵 |
| **验证回归能力** | 正弦人工数据 / diabetes | 输出层维度=1，损失函数用 MSE，激活函数用恒等映射 |
| **快速调试代码（线性可分）** | 人工二分类正态数据 | 隐藏层可 1 层（2→5→1），快速收敛验证 |
| **测试高维特征适配性** | blobs（n_features=10） | 调整隐藏层神经元数（如 |

| | | |
|---|---|---|
| | | 10→20→4） |

## 三、NumPy MLP 适配关键注意事项

1. **数据维度**：特征 X 需为`(样本数，特征数)`，标签 y（分类）需独热编码为`(样本数，类别数)`，回归标签为`(样本数，1)`。
2. **数据归一化**：必须对特征标准化（`(X - mean)/std`），否则不同量纲特征会导致梯度爆炸/消失，训练不收敛。
3. **标签格式**：二分类可用 `sigmoid` 输出+二元交叉熵，多分类必须用 `softmax` 输出 + categorical 交叉熵（标签独热编码）。
4. **数据集划分**：手动划分训练集/测试集（如 8:2），避免过拟合：

```
# 划分训练集（80%）和测试集（20%）
split_idx = int(0.8 * len(X))
X_train, X_test = X[:split_idx], X[split_idx:]
y_train, y_test = y[:split_idx], y[split_idx:]
```

这些数据集均无需复杂依赖，仅需 NumPy+sklearn（用于生成/加载，可替换为纯 NumPy 手动生成），完美匹配“用 NumPy 手动实现 MLP”的实验场景，可快速验证前向传播、反向传播、参数更新等核心逻辑。