Lab 5：线性分类器

本次作业将以“猫狗识别”任务为例，用代码实现线性分类器的图像分类任务。

猫狗图片都为彩色图片，来自2013年的猫狗分类大赛：https://www.kaggle.com/c/dogs-vs-cats/data (https://www.kaggle.com/c/dogs-vs-cats/data)

原数据集有25000张图片，其中猫狗各12500张。为了节省运算时间，本次作业只提供其中的3000张图片（猫狗各1500张）作为训练集，又提供另外200张（猫狗各100张）作为测试数据。如果有兴趣使用全数据集来做测试，可以自行前往上述链接下载。

请将support.py文件存放于本notebook文件所在的文件夹中。请安装以下库：pandas、keras、tensorflow、sklearn、skimage。

In [ ]:
```
import numpy as np
import keras.utils as image
import cv2
from pathlib import Path
from sklearn.model_selection import GridSearchCV, train_test_split
from skimage.io import imread
import pandas as pd
import os
from builtins import range
import warnings
import support
warnings.filterwarnings('ignore')
```

第一步：创建批量化读取彩色图片的函数

传入3个参数：图片文件夹的路径、图片的色彩通道、输出图片的空间维度。

其中，channel为图片的色彩通道，取值范围为0-2。该函数每次能提取图片的其中一个色彩通道。

dimension为输出图片的空间维度，用tuple（宽、高）表示

In [ ]:
```
def load_image_files(container_path, channel, dimension=(50,50)):
    # 获得图片路径，该路径下有不同的文件夹，每个文件夹储存某个类别的图片（只存储图片，不存储其他文件）
    image_dir = Path(container_path)
    # 识别路径下的所有文件夹
    folders = [directory for directory in image_dir.iterdir() if directory.is_dir()]
    # 按照路径下的文件夹，得知图片的类别
    categories = [fo.name for fo in folders]
    descr = "图片分类识别数据集"

    count = 0
    #总的img数组
    all_img = []

    # 遍历该路径下的文件夹
    for i, direc in enumerate(folders):
        #print(direc)
        # 遍历文件夹中的图片文件
        for file in direc.iterdir():
            count += 1
            # 读取一张图片
            img = cv2.imread(str(file))
            # 将图片的宽和高转化为输入的dimension定义的宽和高
            img_pred = cv2.resize(img, dimension, interpolation=cv2.INTER_AREA)
            # 提取某一个色彩通道
            img_pred = img_pred[:, :, channel]
            # 将上一步提取的图片色彩通道层变成numpy数组
            img_pred = image.img_to_array(img_pred)
            # 针对255做归一化
            img_pred = img_pred / 255
            # 将当前的numpy数组添加到总的img数组里
            all_img.append(img_pred)
    # 将总的img数组转为numpy数组
    X = np.array(all_img)
    return X
```

第二步：将图片合成为图片数据矩阵X

In [ ]:
```
# 计算有多少张图片
n_img = support.image_count("lab5_img/train")
# 组建图片矩阵X
# 该矩阵中，包含了1000张图片，每张图片的维度被定义为50*50*3
dimension = (50,50)
X = np.zeros((n_img, dimension[0], dimension[1], 3))
# 提取路径下所有图片的R色彩通道
X0 = load_image_files("lab5_img/train", 0, dimension)
# 提取路径下所有图片的G色彩通道
X1 = load_image_files("lab5_img/train", 1, dimension)
# 提取路径下所有图片的B色彩通道
X2 = load_image_files("lab5_img/train", 2, dimension)
X[:, :, :, 0] = X0[:, :, :, 0]
X[:, :, :, 1] = X1[:, :, :, 0]
X[:, :, :, 2] = X2[:, :, :, 0]
print("数据矩阵的维度: "+str(X.shape))
```

第三步：创建标签向量y

In [ ]:
```
n_img_dogs = support.image_count_category("lab5_img/train/dogs")
n_img_cats = support.image_count_category("lab5_img/train/cats")
# 分别创建狗和猫的标签向量
y0 = np.zeros(n_img_dogs) #狗的标签设为0
y1 = np.ones(n_img_cats) #猫的标签设为1
# 合并标签向量，组成y
y = []
y = np.concatenate((y0, y1), axis=0)

print(X.shape)
print(y.shape)
```

选做：将数据集划分为训练集、验证集和测试集

因为本次作业不涉及到调参，因此不需要验证集。测试集已经单独提供，因此不需要测试集，因此本步骤可以跳过。

如果跳过本步骤，请将上一步的X矩阵重新赋值到X_train，把标签向量y赋值到y_train，并在后面的代码中删除所有和X_val、X_test、y_val、y_test相关的内容。

常规工作中，拆分训练集、验证集和测试集的流程如下：

In [ ]:
```
# 将图片集拆分为训练集、验证集和测试集
# 定义每个数据集的大小
# 先2-8分为测试集和训练集
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.20)
# 再把测试集5-5分为测试集和验证集
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, random_state=42, test_size=0.5)
print("查看以下矩阵的维度：")
print("训练集 X_train 的矩阵维度: " + str(X_train.shape))
print("测试集 X_test 的矩阵维度: " + str(X_test.shape))
print("验证集 X_val 的矩阵维度: " + str(X_val.shape))
print("训练集标签 y_train 的矩阵维度: " + str(y_train.shape))
print("测试集标签 y_test 的矩阵维度: " + str(y_test.shape))
print("验证集标签 y_val 的矩阵维度: " + str(y_val.shape))
```

In [ ]:
```
#组建X_test, X_train, y_train, y_test等数据矩阵
#如果前面跳过了拆分，则只需要组建X_train和y_train

num_training = X_train.shape[0]
mask = list(range(num_training))
X_train = X_train[mask]
y_train = y_train[mask]

num_test = X_test.shape[0]
mask = list(range(num_test))
X_test = X_test[mask]
y_test = y_test[mask]

num_val = X_val.shape[0]
mask = list(range(num_val))
X_val = X_val[mask]
y_val = y_val[mask]

# 让3个X矩阵的每一行为一张图片的x向量
X_train = np.reshape(X_train, (X_train.shape[0], -1))
X_test = np.reshape(X_test, (X_test.shape[0], -1))
X_val = np.reshape(X_val, (X_val.shape[0], -1))
# 打印训练集、测试集和验证集的数据量，及每个数据的维度（50*50*3）
print("训练集、测试集和验证集的数据量及图片维度", X_train.shape, X_test.shape, X_val.shape)
# 打印训练集、测试集和验证集标签的数据量
print("标签向量的长度", y_train.shape, y_test.shape, y_val.shape)
```

任务1：做bias trick

对X_train、X_var、X_test三个矩阵做出bias trick的修改，即：对其中的每一张图片，在其向量的末尾添加1。

提示：X矩阵中每一行为一个图片的图像表示向量x。

In [ ]:
```
# 将代码写在此行以下

# 将代码写在此行以上，完成后运行整个cell
print("训练集、测试集和验证集的数据量，及bias trick后的图片维度",X_train.shape, X_test.shape, X_val.shape)
print("数据已经就绪")
```

对数据做0均值化的预处理

In [ ]:
```
# 0均值化
mean_image = np.mean(X_train, axis=0)
X_train -= mean_image
X_test -= mean_image
X_val -= mean_image
```

第四步（任务2）：损失函数

完成损失函数 loss_function(W,X,y,reg)的编写

它的传入参数如下：

- W：权值矩阵
- X：图像表示向量
- y：标签向量
- reg：正则项权重，相当于课件中的lambda

要求：返回总损失

总损失包括多类支持向量机损失(L)和L2正则损失(R)，它们之间的权重有reg决定，即：

总损失 = L + reg * R

提示：可能用到的中间量包括：

- Li ------- 每个样本的损失值
- c = W.shape[1] ------- 类别数
- d = X.shape[0] ------- 图片的维数
- f = X.dot(W) ------- 类别分数（权值向量和图片向量点乘）

In [ ]:
```
# 替换掉Pass，完成损失函数的变形
def loss_function(W, X, y, reg):
    pass
```

第五步：创建LinearClassifier类

任务3：预测函数

在下面的LinearClassifier类中，定义predict()函数。

该函数使用该线性分类器训练好的权重W，来预测X中所含图片的标签。

输入X:

包含数据的numpy数组，其维度为(N, D)，意思为：共有N个图片向量，每个图片向量的维数（向量长度）为D。

输出:

一个长度为N的一维数组，数组中的每个元素都是一个整数，表示对X中的每个图片向量的类别预测结果。其中，狗的标签为0，猫的标签为1

In [ ]:
```
# 替换predict(self, X)后的pass完成此函数

from __future__ import print_function
from builtins import object


class LinearClassifier(object):

    def __init__(self):
        self.W = None

    def train(self, X, y, learning_rate=1e-3, reg=1e-5, num_iters=100,
              batch_size=200, verbose=False):

        num_train, dim = X.shape
        num_classes = np.max(y) + 1 # y的值域为1到c-1，其中c为类别数
        if self.W is None:
            self.W = 0.001 * np.random.randn(dim, int(num_classes))
        # 随机梯度下降
        loss_history = []
        for it in range(num_iters):
            X_batch = None
            y_batch = None

            batch_indices = np.random.choice(num_train, batch_size, replace=False)
            X_batch = X[batch_indices]
            y_batch=y[batch_indices]

            # 评估损失和梯度值
            loss, grad = self.loss(X_batch, y_batch, reg)
            loss_history.append(loss)

            # 根据梯度和学习率，更新W权值
            self.W -= learning_rate*grad

            # 每50次迭代打印一次结果
            if verbose and it % 50 == 0:
                print('迭代次数 %d / %d: 损失值 %f' % (it, num_iters, loss))

        return loss_history

    def predict(self, X):
        """
        使用该线性分类器训练好的权重W，来预测X中所含图片的标签。

        输入:
        X: 包含图片数据的numpy数组，其维度为(N, D)，意思为：共有N个图片向量，每个图片向量的维数（向量长度）为D。

        输出:
        一个长度为N的一维数组，数组中的每个元素都是一个整数，表示对X中的每个图片向量的类别预测结果。其中，狗的标签为0，猫的标

        提示1：忽略偏置后的线性分类器对某张图片在每一个类别中的预测分数可表示为权重和图片向量的点乘，即score = W·X

        提示2：获得score向量后，需要识别score中的最高分对应的类别，如果是狗，则在输出数组的相应位置（图片的次序）设为0，
        如果是猫则设为1
        """
        pass


class LinearLoss(LinearClassifier):
    """ 调用loss_function计算损失 """

    def loss(self, X_batch, y_batch, reg):
        L = loss_function(self.W, X_batch, y_batch, reg)
        return support.gradient(L, self.W, X_batch, y_batch, reg)
```

第六步：训练

In [ ]:
```
# 训练模型，学习率0.005，正则项权重0.005，迭代1000次，小批量100
model_loss = LinearLoss()
loss_hist = model_loss.train(X_train, y_train, learning_rate=0.005, reg=0.005,
                      num_iters=1000, batch_size=100, verbose=True)
```

第七步：用200张图片进行测试

注意：本次作业仅为展示分类器的训练过程，因为使用数据较少，训练结果不一定可观。

In [ ]:
```
# 对100张狗的测试图片进行测试
n_cat = 0
n_dog = 0
for i in range(1, 101):
    file = os.path.join('lab5_img/test/dogs/dog_test'+str(i) + ".jpg")
    img = cv2.imread(file)
    img_pred = cv2.resize(img, (50, 50), interpolation=cv2.INTER_AREA)
    img_pred = image.img_to_array(img_pred)
    img_pred = img_pred / 255
    img_pred = np.reshape(img_pred, (1, 3 * img_pred.shape[0] * img_pred.shape[1]))
    img_pred = np.hstack([img_pred, np.ones((img_pred.shape[0], 1))])
    res = model_loss.predict(img_pred)

    if res[0] == 0:
        n_dog += 1
    else:
        n_cat += 1

print("分类结果: ")
print("狗: ", n_dog, "猫: ", n_cat)
```

In [ ]:
```
# 对100张猫的测试图片进行测试
n_cat = 0
n_dog = 0
for i in range(1, 101):
    file = os.path.join('lab5_img/test/cats/cat_test'+str(i) + ".jpg")
    img = cv2.imread(file)
    img_pred = cv2.resize(img, (50, 50), interpolation=cv2.INTER_AREA)
    img_pred = image.img_to_array(img_pred)
    img_pred = img_pred / 255
    img_pred = np.reshape(img_pred, (1, 3 * img_pred.shape[0] * img_pred.shape[1]))
    img_pred = np.hstack([img_pred, np.ones((img_pred.shape[0], 1))])
    res = model_loss.predict(img_pred)

    if res[0] == 0:
        n_dog += 1
    else:
        n_cat += 1

print("分类结果: ")
print("狗: ", n_dog, "猫: ", n_cat)
```

任务4：总结分类结果

假设狗类为positive、猫类为negative。

绘制confusion matrix并计算accuracy、precision、recall和F1 score。

In [ ]:
```
# 在这里完成任务4
```

任务5：手动调参

重新训练模型，手动调整学习率和正则项权重的设置，观察分类结果的变化

In [ ]:
```
# 在这里完成任务5，手动调整下面的学习率（learning_rate）及正则项权重（reg），观察模型性能的变化
model_loss = LinearLoss()
loss_hist = model_loss.train(X_train, y_train, learning_rate=0.005, reg=0.005,
                      num_iters=1000, batch_size=100, verbose=True)
```

在下面的Cell中，用comment的形式，对任务5的操作进行说明：你查看了哪些学习率和正则项权重的组合（至少3种组合）以及它们的训练效果有何不同。

In [ ]:
```
# 将回答写在此处
```

提交方式

本次作业有5个任务。完成所有cell的运行后，保存为ipynb和PDF格式（保留所有输出）。将导出的ipynb命名为“Lab5+姓名+学号.ipynb”，将导出的PDF命名为“Lab5+姓名+学号.pdf”，并将上述两个文件提交到学习通作业模块的相应位置（如果ipynb无法单独上传，请打包成zip格式）。 请独立完成练习，参考答案将在截止时间后公布。 截止时间：2024年6月5日23:59。超时1天之内将扣除5%的分数，超时1天以上将扣除10%的分数。