标题：预测房价
机器学习通常把现实问题建模成一个函数求极值问题，而当函数非常复杂时，该极值没有办法得到显式的解析解（没法通过显式的公式计算得到，例如简单的二次函数极值就可以通过公式得到），这时就会使用一种常用的数值计算方法——梯度下降。梯度就是表示某一函数在该点处的方向导数沿着该方向取得最大值。如果把目标函数的图像想象成一座山的话，那么梯度就是山最陡峭的地方，每次沿着最陡峭的地方走一小步，一步一步就可以走到低点。（如果是凸函数，就会走到最低点，凸函数和非凸函数的区别大家可以网上搜索，简单的理解就是一口光滑的锅和一口凹凸不平的锅，因为凹凸不平所以可能陷入到谋一个坑里。我们这里使用的目标函数是凸函数。）形式化的来说，如果目标函数是$f(w)$，其中w是n维向量，梯度下降的流程如下：

$$
\begin{aligned}
&\text{初始化}w\text{向量，这里我们选择用}0\text{向量初始化：}w^0 = \mathbf{0} \\
&\text{while } f(w^{t-1}) - f(w^t) > \epsilon \text{ 或者未达到设置的最大迭代次数} \text{max\_iter}: \\
&w^t = w^{t-1} - \alpha \nabla f(w)
\end{aligned}
$$

其中ϵ表示算法停止需要满足的精度这里我们可以设置为1e-5，$\nabla f(w)$是$f(w)$的梯度，α一般称作步长或学习率控制每一步跨多大。大家可以在https://zhuanlan.zhihu.com/p/107782332查看到一个简单的例子。

这里我们需要解决的是预测1970波斯顿房价问题，我们的预测依据是以下13个指标：

```
CRIM - per capita crime rate by town
ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS - proportion of non-retail business acres per town.
CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
NOX - nitric oxides concentration (parts per 10 million)
RM - average number of rooms per dwelling
AGE - proportion of owner-occupied units built prior to 1940
DIS - weighted distances to five Boston employment centres
RAD - index of accessibility to radial highways
TAX - full-value property-tax rate per $10,000
PTRATIO - pupil-teacher ratio by town
B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT - % lower status of the population
```

我们提供了train.txt，valid.txt，test.txt三个文件用于模型的训练，挑选参数α、max_iter和最终测试（搜索机器学习训练集，验证集也称作开发集与测试集了解更多）。每个文件的格式都是LibSVM格式，每一行的内容如下所示：

```
20.1 1:-0.997757 2:-0.4 3:-0.672287 4:-1 5:-0.823045 6:-0.0289327 7:0.281153
8:-0.0530968 9:-0.565217 10:-0.568702 11:-0.148936 12:0.988502 13:-0.411148
```

每一行的开头是样本真实的房价（单位是千美元），13个特征通过特征id加“:”加特征数值呈现，在本题的计算中id和“:”没有用处。读取数据文件，将特征和真实房价转换为二维列表x和一维列表y。

我们将预测问题建模成简单的线性回顾问题，即当输入第i个样本特征向量x^i时我们的预测结果是$h(x^i) = \sum_{j=1}^{13} w_j * x_j^i$，我们的目标函数$f(w) = \frac{1}{2}\sum_{i=1}^{n}(\sum_{j=1}^{13} w_j * x_j^i - y^i)^2$其中$n$表示训练样本数量，$y$表示训练样本的真实房价，所以目标函数表达含义是所有训练样本预测结果与真实结果的平方和（注意线性回归是有解析解，但是这里要求使用梯度下降方法），每个特征的偏导数为$\partial \frac{f(w)}{w_j} = \sum_{i=1}^{n}(\sum_{j=1}^{13} w_j * x_j^i - y^i) * x_j^i$。

提示：训练集最小损失在308左右。

```
# 读取数据
def read_libsvm(name):
    # 返回二维列表x和一维列表y
    # 如果熟悉Numpy的同学也可以使用
    return x, y

# 线性模型
class LinearModel(object):
    # 初始化权重列表w和特征个数d
    def __init__(xxxxx):

    # 计算样本i的预测值
    def h(xxxxx):

# 梯度下降
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX

# 使用 pip install matplotlib 安装matplotlib
# idx是从0到t的迭代次数列表，mses是对应迭代次数的训练损失
import matplotlib.pyplot as plt
plt.plot(idx, mses)
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.xlabel('迭代次数')
plt.ylabel('训练损失')
plt.show()
```