

线性回归

让我们从经典的线性回归（Linear Regression [1]）模型开始这份教程。在这一章里，你将使用真实的数据集建立起一个房价预测模型，并且了解到机器学习中的若干重要概念。

背景介绍

给定一个大小为 n 的数据集 $\{y_i, x_{i1}, \dots, x_{id}\}_{i=1}^n$ ，其中 x_{i1}, \dots, x_{id} 是第 i 个样本 d 个属性上的取值， y_i 是该样本待预测的目标。线性回归模型假设目标 y_i 可以被属性间的线性组合描述，即

$$y_i = \omega_1 x_{i1} + \omega_2 x_{i2} + \dots + \omega_d x_{id} + b, i = 1, \dots, n$$

例如，在我们将要建模的房价预测问题里， x_{ij} 是描述房子 i 的各种属性（比如房间的个数、周围学校和医院的个数、交通状况等），而 y_i 是房屋的价格。

初看起来，这个假设实在过于简单了，变量间的真实关系很难是线性的。但由于线性回归模型有形式简单和易于建模分析的优点，它在实际问题中得到了大量的应用。很多经典的统计学习、机器学习书籍[2,3,4]也选择对线性模型独立成章重点讲解。

效果展示

我们使用从[UCI Housing Data Set](#)获得的波士顿房价数据集进行模型的训练和预测。下面的散点图展示了使用模型对部分房屋价格进行的预测。其中，每个点的横坐标表示同一类房屋真实价格的中位数，纵坐标表示线性回归模型根据特征预测的结果，当二者值完全相等的时候就会落在虚线上。所以模型预测得越准确，则点离虚线越近。

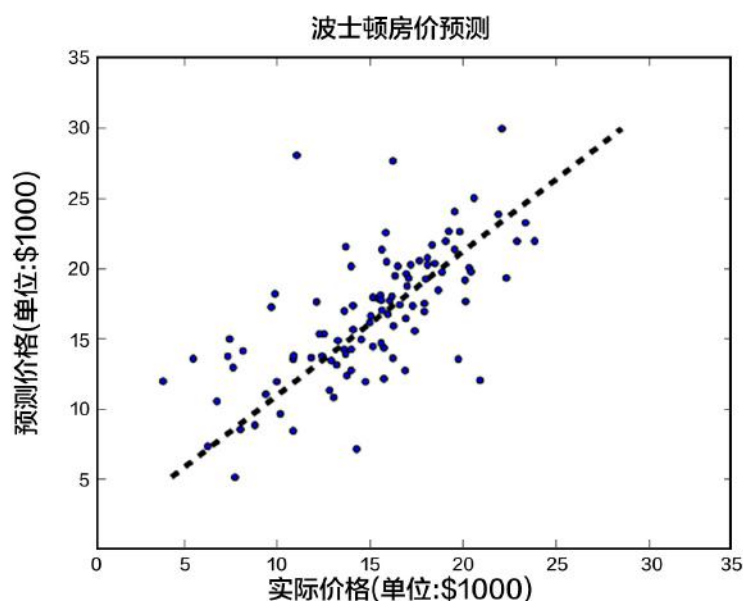


图1. 预测值 V.S. 真实值

模型概览

模型定义

在波士顿房价数据集中，和房屋相关的值共有14个：前13个用来描述房屋相关的各种信息，即模型中的 x_i ；最后一个值为我们要预测的该类房屋价格的中位数，即模型中的 y_i 。因此，我们的模型就可以表示成：

$$\hat{Y} = \omega_1 X_1 + \omega_2 X_2 + \dots + \omega_{13} X_{13} + b$$

\hat{Y} 表示模型的预测结果，用来和真实值 Y 区分。模型要学习的参数即： $\omega_1, \dots, \omega_{13}, b$ 。

建立模型后，我们需要给模型一个优化目标，使得学到的参数能够让预测值 \hat{Y} 尽可能地接近真实值 Y 。这里我们引入损失函数（[Loss Function](#)，或 Cost Function）这个概念。输入任意一个数据样本的目标值 y_i 和模型给出的预测值 \hat{y}_i ，损失函数输出一个非负的实值。这个实质通常用来反映模型误差的大小。

对于线性回归模型来讲，最常见的损失函数就是均方误差（Mean Squared Error，[MSE](#)）了，它的形式是：

$$MSE = \frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$$

即对于一个大小为 n 的测试集， MSE 是 n 个数据预测结果误差平方的均值。

训练过程

定义好模型结构之后，我们要通过以下几个步骤进行模型训练

1. 初始化参数，其中包括权重 ω_i 和偏置 b ，对其进行初始化（如0均值，1方差）。
2. 网络正向传播计算网络输出和损失函数。
3. 根据损失函数进行反向误差传播（[backpropagation](#)），将网络误差从输出层依次向前传递，并更新网络中的参数。
4. 重复2~3步骤，直至网络训练误差达到规定的程度或训练轮次达到设定值。

数据准备

执行以下命令来准备数据:

```
1. cd data && python prepare_data.py
```

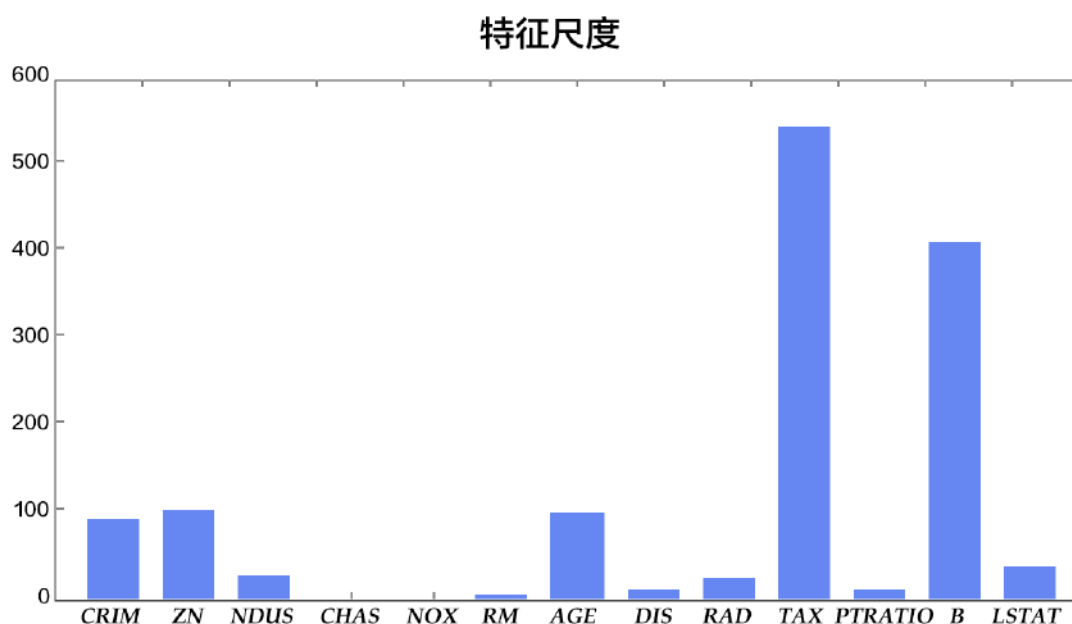


图2. 各维属性的取值范围

整理训练集与测试集

我们将数据集分割为两份：一份用于调整模型的参数，即进行模型的训练，模型在这份数据集上的误差被称为**训练误差**；另外一份被用来测试，模型在这份数据集上的误差被称为**测试误差**。我们训练模型的目的是为了通过从训练数据中找到规律来预测未知的新数据，所以测试误差是更能反映模型表现的指标。分割数据的比例要考虑到两个因素：更多的训练数据会降低参数估计的方差，从而得到更可信的模型；而更多的测试数据会降低测试误差的方差，从而得到更可信的测试误差。一种常见的分割比例为**8 : 2**，感兴趣的读者朋友们也可以尝试不同的设置来观察这两种误差的变化。

执行如下命令可以分割数据集，并将训练集和测试集的地址分别写入train.list 和 test.list两个文件中，供PaddlePaddle读取。

```
1. python prepare_data.py -r 0.8 #默认使用8:2的比例进行分割
```

在更复杂的模型训练过程中，我们往往还会多使用一种数据集：验证集。因为复杂的模型中常常还有一些超参数（[Hyperparameter](#)）需要调节，所以我们会尝试多种超参数的组合来分别训练多个模型，然后对比它们在验证集上的表现选择相对最好的一组超参数，最后才使用这组参数下训练的模型在测试集上评估测试误差。由于本章训练的模型比较简单，我们暂且忽略掉这个过程。

提供数据给PaddlePaddle

准备好数据之后，我们使用一个Python data provider来为PaddlePaddle的训练过程提供数据。一个 data provider 就是一个Python函数，它会被PaddlePaddle的训练过程调用。在这个例子里，只需要读取已经保存好的数据，然后一行一行地返回给PaddlePaddle的训练进程即可。

```
1. from paddle.trainer.PyDataProvider2 import *
2. import numpy as np
3. #定义数据的类型和维度
4. @provider(input_types=[dense_vector(13), dense_vector(1)])
5. def process(settings, input_file):
6.     data = np.load(input_file.strip())
7.     for row in data:
8.         yield row[:-1].tolist(), row[-1:].tolist()
```

模型配置说明

数据定义

首先，通过 `define_py_data_sources2` 来配置PaddlePaddle从上面的 `dataprovider.py` 里读入训练数据和测试数据。PaddlePaddle接受从命令行读入的配置信息，例如这里我们传入一个名为 `is_predict` 的变量来控制模型在训练和测试时的不同结构。

```
1.  from paddle.trainer_config_helpers import *
2.
3.  is_predict = get_config_arg('is_predict', bool, False)
4.
5.  define_py_data_sources2(
6.      train_list='data/train.list',
7.      test_list='data/test.list',
8.      module='dataprovider',
9.      obj='process')
```

算法配置

接着，指定模型优化算法的细节。由于线性回归模型比较简单，我们只要设置基本的 `batch_size` 即可，它指定每次更新参数的时候使用多少条数据计算梯度信息。

```
1.  settings(batch_size=2)
```

网络结构

最后，使用 `fc_layer` 和 `LinearActivation` 来表示线性回归的模型本身。

```
1.  #输入数据，13维的房屋信息
2.  x = data_layer(name='x', size=13)
3.
4.  y_predict = fc_layer(
5.      input=x,
6.      param_attr=ParamAttr(name='w'),
7.      size=1,
```

```
8.         act=LinearActivation(),
9.         bias_attr=ParamAttr(name='b'))
10.
11.     if not is_predict: #训练时，我们使用MSE，即regression_cost作为损失函数
12.         y = data_layer(name='y', size=1)
13.         cost = regression_cost(input=y_predict, label=y)
14.         outputs(cost) #训练时输出MSE来监控损失的变化
15.     else: #测试时，输出预测值
16.         outputs(y_predict)
```

训练模型

在对应代码的根目录下执行PaddlePaddle的命令行训练程序。这里指定模型配置文件为 `trainer_config.py`，训练30轮，结果保存在 `output` 路径下。

```
1.     ./train.sh
```

应用模型

现在来看下如何使用已经训练好的模型进行预测。

```
1.     python predict.py
```

这里默认使用 `output/pass-00029` 中保存的模型进行预测，并将数据中的房价与预测结果进行对比，结果保存在 `predictions.png` 中。

如果你想使用别的模型或者其它的数据进行预测，只要传入新的路径即可：

```
1.     python predict.py -m output/pass-00020 -t data/housing.test.npy
```

总结

在这章里，我们借助波士顿房价这一数据集，介绍了线性回归模型的基本概念，以及如何使用PaddlePaddle实现训练和测试的过程。很多的模型和技巧都是从简单的线性回归模型演化而

来，因此弄清楚线性模型的原理和局限非常重要。

参考文献

1. https://en.wikipedia.org/wiki/Linear_regression
2. Friedman J, Hastie T, Tibshirani R. The elements of statistical learning[M]. Springer, Berlin: Springer series in statistics, 2001.
3. Murphy K P. Machine learning: a probabilistic perspective[M]. MIT press, 2012.
4. Bishop C M. Pattern recognition[J]. Machine Learning, 2006, 128.



本教程由PaddlePaddle创作，采用[知识共享 署名-非商业性使用-相同方式共享 4.0 国际 许可协议](#)进行许可。