import paddle.v2 as pdl

导入库

import paddle.v2 as paddle

import paddle.v2.dataset.uci\_housing as uci\_housing

导入库

# 模型初始化

paddle.init(use\_gpu=False,trainer\_count=1)

不使用GPU，一个训练器

# 数据

x = paddle.layer.data(name='x',type=paddle.data\_type.dense\_vector(13))

y\_predict = paddle.layer.fc(input=x, size=1, act=paddle.activation.Linear())

全连接层，act：激活函数，

y = paddle.layer.data(name='y', type=paddle.data\_type.dense\_vector(1))

cost = paddle.layer.mse\_cost(input=y\_predict, label=y)

损失函数，mse：均方差，input，label参数不要弄反了

#参数，优化器，训练器

parameters = paddle.parameters.create(cost)

optimizer = paddle.optimizer.Momentum(momentum=0)

模型简单，不需要向量。

trainer = paddle.trainer.SGD(cost=cost, parameters=parameters, update\_equation=optimizer)

SGD：随机梯度下降

词到向量，word2vec，用genism库。

机器学习分类：

监督学习

分类

决策树

支持向量机

回归

线性回归 LinearRegression

非线性回归 GBRT

无监督学习

聚类 clustering

词嵌入 word embeding

监督学习是要根据已知数据集X和Y寻找出映射关系f

机器学习：

从数据中产生模型

关键概念：

假设函数hypothesis function

用数学的方法描述自变量X和因变量Y之间的关系

线性假设函数

损失函数cost function

用数学的方法衡量假设函数预测结果与真实值之间的“误差”

MSE(mean square error)均方差损失函数

交叉熵(cross entropy)

优化算法（梯度下降gradient descent）

学习率，不能太大，不能太小。

三种梯度下降优化框架：

批量梯度下降BGD，每次迭代数据量：全量样本，

随机梯度下降SGD，一个样本，

小批量梯度下降MBGD，部分样本，

多种优化算法

Momentum

adadelta

adam

adagrad

数据预处理

归一化

dataset

reader，对所有的数据做归一化，结果：\_reader，每次返回一条数据，把数据集中

reader-creator

reader-decorator

reader = paddle.batch( paddle.reader.shuffle (uci\_housing.train(),buf\_size=500),batch\_size=2 )

reader：从数据集里面，一条一条的读数据

shuffle\_reader：从reader里面读取一个固定大小的数据进行一个本地的随机化

batch\_reader：从batch\_reader里面读取一个固定大小的batch\_size的数据，送入训练器进行每一步的迭代和训练。

假设函数

线性回归本质上就是一个采用线性激活函数的全连接层。

找到最合适的θ的组合，

x = paddle.layer.data(name=’x’, type=paddle.data\_type.dense\_vector(13))

y\_predict = paddle.layer.fc(input=x, size=1, act=paddle.activation.Linear())

全连接网络：fc\_layer

激活函数：线性激活函数

损失函数

均方误差

标注数据

y\_label=paddle.layer.data(name=’y\_label’, type=paddle.data\_type.dense\_vector(1))

损失函数

cost=paddle.layer.mse\_cost(input=y\_predict, label=y\_label)

梯度下降

optimizer=paddle.optimizer.Momentum(momentum=0, learning\_rate=0.01)

learning-rate，学习率

构造SGD trainer

trainer = paddle.trainer.SGD(cost = cost, parameters=parameters,update\_eauqtion=optimizer)

提前构造 cost parameters optimizer

开始训练，指定训练30次

trainer.train(reader=batch\_reader,feeding=feeding,event\_handler=event\_handler,num\_passes=30)

应用模型

应用训练出来的模型进行预测inference

1、生成测试数据

test\_data\_creator = paddle.dataset.uci\_housint.test()

test\_data = []

for item in test\_data\_creator():

test\_data.apppend(item[0],))

if len(test\_data) == 5:

break

2、预测inference

probs = paddle.infer(output\_layer=y\_predict, parameters=parameters, input=test\_data)

for data in probs:

print data

线性回归优点：

形式简单，易于建模

可演变成其他复杂模型

可解释性好

数据集操作：paddle.dataset / paddle.reader

全连接层：paddle.layer.fc()

线性激活函数：paddle.activation.Linear()

均方差损失函数：paddle.layer.mse\_cost()

参数优化器：paddle.optimizer.Momentum()

梯度下降训练器：paddle.trainer.SGD()

识别数字