pandas + 机器学习

2016-08-20 Programming 从零到一 pandas python 机器学习

几次参加国内的机器学习竞赛时, 都使用 pandas 作为数据处理库. 很遗憾一直没有取得很好的成绩, 但是既然有了尝试, 就把一点点实际的经验记录下来.

数据 io

输入方面, `pd.read_csv()` 和 `pd.read_table()` 是最常用的两个命令[1]. 常用的参数有:

- `header` 是否有标签行
- `names=[column names]` 指定列名称

导入的数据是 pandas.DataFrame 对象

输出时, 常用 `pd.to_csv()`:

- `header`
- `index` 是否输出 index, 通常都不需要(=False).

数据整理和运算

得到数据后, 经常需要做整理和处理操作, pandas 可以很方便地处理 DataFrame [2]

`df.loc[row, col]` 或者 `df.iloc[row,col]` 可以**选择**某些行/列 操作(i/o)

Content

- `df.loc[:, 'column_name']` == `df.column_name`

```
training_set.loc[:,'y'] = np.log10(1.001 - training_set.key_index_x)  # 赋值建立 log 变换后的新列
df['mins'] = df['time_delta'].astype('timedelta64[m]')
df_num['time'] = df_num['time'].apply(lambda x: pd.to_datetime(x, errors='coerce'))  # 赋值回原列,
注意这里用的 apply(function) 是一个比较通用的方法

time0 = df.iloc[0,:].time
df['time_delta'] = df['time'].apply(lambda x: x - time0)
```

- `df.loc` 还可使用限定条件:

```
training_set.loc[training_set.key_index_x > 0.85]  # 选择 key_index_x > 0.85 的行
```

- `df.iloc` 限定使用序号, 性能更好

```
top50 = training_set.sort_values(by='y').iloc[:50,:]
least50 = training_set.sort_values(by='y').iloc[-50:,:]  # 按 y 列排序之后取首尾
training_set.iloc[[4557, 4558, 5362, 5363, 5884, 5885, 5886, 5887, 5888, 5889, 5890,5891, 5892,
5893],:]  # 使用列表查看某些行的数据
```

`df.loc` 的输出常常用于后续的输入. 如使用 `series.tolist()` 或 `df.as_matrix()`

element-wise 操作 通常可以直接进行(如上面取对数的例子)

`df.describe()` 可以给出统计值(包括 count mean std min max等) 配合可视化可以很快理解数据分布

缺失值

`df.dropna()`, `df.fillna()` 提供了方便的处理缺失值的方法.

如果缺失值比例不大, 最简单的处理方法就是直接舍弃:

Content

```
df.dropna(how='any')
```

pivot/groupby

透视表/汇总也是很重要的数据整理操作, 也常常是可视化之前的准备:

```
# 用index 指定行, columns 指定列, values 指定值
df.pivot(index='mins', columns='product_no', values='value').plot()

grouped = training_set.groupby([training_set.draft_param1,
                                training_set.y_bin])
grouped.count()
```

也可使用 `grouped.mean()`, 甚至同时用多个函数, 获得类似 `df.describe()` 的效果

```
grouped = training_set.groupby('draft_param1')
# grouped.aggregate(np.mean)
print(grouped.y.agg([np.count_nonzero, np.mean, np.std])) ~~~
         count_nonzero      mean       std draft_param1 0                      25.0 -1.223018
0.448874 1                    239.0 -1.518699  0.410279 2                       35.0 -1.557707
0.357107 3                   4959.0 -1.504268  0.393509 4                      530.0 -1.438326
0.325457 5                   3017.0 -1.489401  0.322244 ~~~
```

可视化

使用 `DataFrame.plot()`

```
training_set.plot.scatter('draft_param1', 'y')  # 散点图
training_set.hist(column=['key_index_x'],bins=100)  # 只用一行命令就可以查看一列的统计分布
```

Content

```
draft_training.hist(column=['key_index','draft_param1','draft_param2','draft_param3'],bins=10)  #
一次查看多列数据也没问题
```

seaborn[3] 也提供了直接绘制 DataFrame 的方法, 可以绘制 violinplot, boxplot 等更复杂的图

```
import seaborn as sns
sns.violinplot(x="draft_param1", y="y",
               data=training_set)
```

传入 scikit-learn

scikit-learn 的输入是 `numpy.array`, 因此使用 `DataFrame.as_matrix(columns=[features])` 可以选择特征列输入 scikit-learn

```
X = training_set.as_matrix(columns=['param1', 'param2',
        'param3', 'param4',
        'param5','param6','param7',
        'param8','param9','draft_param1',
        'draft_param2','draft_param3','draft_param4',
        'draft_param5','draft_param6','draft_param7',
        'draft_param9', 'draft_param10','draft_param11'])
y = training_set.as_matrix(columns=['y']).ravel()

from sklearn.cross_validation import KFold
kf = KFold(len(training_set), n_folds=5)

from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor(n_estimators=100, max_depth=10)

for k, (train, test) in enumerate(kf):
```

Content

```
    clf.fit(X[train], y[train])
    print("[fold {}], training score {:.5f}, test score {:.5f}".
          format(k, clf.score(X[train], y[train]), clf.score(X[test], y[test])))
```

小结

pandas 作为 python 科学计算和数据分析生态链中的重要部分, 提供了方便的数据结构和相关处理函数, 也可轻松与 numpy, scikit-learn 等库集成为完整的 pipeline.

pandas 的文档[4], 以及 10 minutes to pandas[5], 比本文更加详尽. 遇到问题时, 利用google 搜索关键字 (如 pandas pivottable, pandas scatter plot) 往往比翻文档更快(通常第一个是文档, 第二个开始是 stackoverflow 的问题和答案).

References

1. IO Tools (Text, CSV, HDF5, …) — pandas 0.18.1 documentation ↩
2. Indexing and Selecting Data — pandas 0.18.1 documentation ↩
3. Seaborn: statistical data visualization — seaborn 0.7.1 documentation ↩
4. pandas: powerful Python data analysis toolkit — pandas 0.18.1 documentation ↩
5. 10 Minutes to pandas — pandas 0.18.1 documentation ↩

Similar Posts

Content

- chaos to MoinMoin
- OpenCV auto 2048
- opencv-python3-pyenv
- 在 AWS Elastic Beanstalk 部署 Django 应用

上一篇 在 AWS Elastic Beanstalk 部署 Django 应用

下一篇 opencv-python3-pyenv

Comments

Content