昵称：rubbninja
园龄：1年11个月
粉丝：18
关注：3
+加关注

< 2017年4月 >

日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签
机器学习(13)
学习笔记(12)
室内定位(7)
室内定位系列(7)
卡尔曼滤波(2)
C++(2)
ipython(1)
ipython notebook(1)
Java(1)
python(1)
更多

室内定位系列（四）——位置指纹法的实现（测试各种机器学习分类器）

位置指纹法中最常用的算法是k最近邻（kNN）。本文的目的学习一下python机器学习scikit-learn的使用，尝试了各种常见的机器学习分类器，比较它们在位置指纹法中的定位效果。

导入数据

数据来源说明：http://www.cnblogs.com/rubbninja/p/6118430.html

```
# 导入数据
import numpy as np
import scipy.io as scio
offline_data = scio.loadmat('offline_data_random.mat')
online_data = scio.loadmat('online_data.mat')
offline_location, offline_rss = offline_data['offline_location'],
offline_data['offline_rss']
trace, rss = online_data['trace'][0:1000, :], online_data['rss'][0:1000, :]
del offline_data
del online_data
```

```
# 定位准确度定义
def accuracy(predictions, labels):
    return np.mean(np.sqrt(np.sum((predictions - labels)**2, 1)))
```

knn回归

```
# knn回归
from sklearn import neighbors
knn_reg = neighbors.KNeighborsRegressor(40, weights='uniform', metric='euclidean')
%time knn_reg.fit(offline_rss, offline_location)
%time predictions = knn_reg.predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, "m"
```

```
Wall time: 92 ms
Wall time: 182 ms
accuracy:  2.24421479398 m
```

Logistic regression （逻辑斯蒂回归）

```
# 逻辑斯蒂回归是用来分类的
labels = np.round(offline_location[:, 0]/100.0) * 100 + np.round(offline_location[:,
1]/100.0)
from sklearn.linear_model import LogisticRegressionCV
clf_l2_LR_cv = LogisticRegressionCV(Cs=20, penalty='l2', tol=0.001)
predict_labels = clf_l2_LR.fit(offline_rss, labels).predict(rss)
x = np.floor(predict_labels/100.0)
y = predict_labels - x * 100
predictions = np.column_stack((x, y)) * 100
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, 'm'
```

```
accuracy:  3.08581348591 m
```

Support Vector Machine for Regression （支持向量机）

```
from sklearn import svm
clf_x = svm.SVR(C=1000, gamma=0.01)
clf_y = svm.SVR(C=1000, gamma=0.01)
%time clf_x.fit(offline_rss, offline_location[:, 0])
%time clf_y.fit(offline_rss, offline_location[:, 1])
%time x = clf_x.predict(rss)
%time y = clf_y.predict(rss)
predictions = np.column_stack((x, y))
```

随笔档案

2017年1月 (1)

2016年12月 (4)

2016年11月 (2)

2016年2月 (2)

2016年1月 (2)

2015年11月 (3)

2015年10月 (4)

2015年7月 (3)

2015年5月 (2)

C++基础

最新评论

1. Re:室内定位系列（五）——目标跟踪（卡尔曼滤波）

博主可以加我微信吗？ 我的名字叫吴志国，微信号：18813157365

--gguo_2017

2. Re:室内定位系列（五）——目标跟踪（卡尔曼滤波）

@gguo_2017不错哦！...

--rubbninja

3. Re:室内定位系列（五）——目标跟踪（卡尔曼滤波）

您好！我是北邮研二在读学生，研究方向室内定位，目前有个国内室内定位交流群，主要成员都是在读的研究生和博士生，还有部分的国外研究生博士生。如果博主有兴趣，欢迎加入！希望和博主多交流学习。室内定位交流群号.....

--gguo_2017

4. Re:室内定位系列（一）——WiFi位置指纹（译）

@king-blues没有。很多分类器都可以用来代替最基本的knn，但感觉没这个必要。后续会尝试各种分类器来做这个，不过都用现成的包。...

--rubbninja

```
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, "m"
```

```
Wall time: 9min 27s
Wall time: 12min 42s
Wall time: 1.06 s
Wall time: 1.05 s
accuracy:  2.2468400825 m
```

Support Vector Machine for Classification（支持向量机）

```
from sklearn import svm
labels = np.round(offline_location[:, 0]/100.0) * 100 + np.round(offline_location[:,
1]/100.0)
clf_svc = svm.SVC(C=1000, tol=0.01, gamma=0.001)
%time clf_svc.fit(offline_rss, labels)
%time predict_labels = clf_svc.predict(rss)
x = np.floor(predict_labels/100.0)
y = predict_labels - x * 100
predictions = np.column_stack((x, y)) * 100
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, 'm'
```

```
Wall time: 1min 16s
Wall time: 15 s
accuracy:  2.50931890608 m
```

random forest regressor（随机森林）

```
from sklearn.ensemble import RandomForestRegressor
estimator = RandomForestRegressor(n_estimators=150)
%time estimator.fit(offline_rss, offline_location)
%time predictions = estimator.predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, 'm'
```

```
Wall time: 58.6 s
Wall time: 196 ms
accuracy:  2.20778352008 m
```

random forest classifier（随机森林）

```
from sklearn.ensemble import RandomForestClassifier
labels = np.round(offline_location[:, 0]/100.0) * 100 + np.round(offline_location[:,
1]/100.0)
estimator = RandomForestClassifier(n_estimators=20, max_features=None, max_depth=20) # 内存受限，tree的数量有点少
%time estimator.fit(offline_rss, labels)
%time predict_labels = estimator.predict(rss)
x = np.floor(predict_labels/100.0)
y = predict_labels - x * 100
predictions = np.column_stack((x, y)) * 100
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, 'm'
```

```
Wall time: 39.6 s
Wall time: 113 ms
accuracy:  2.56860790666 m
```

Linear Regression（线性回归）

```
from sklearn.linear_model import LinearRegression
predictions = LinearRegression().fit(offline_rss, offline_location).predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, 'm'
```

```
accuracy:  3.83239841667 m
```

Ridge Regression（岭回归）

```
from sklearn.linear_model import RidgeCV
clf = RidgeCV(alphas=np.logspace(-4, 4, 10))
predictions = clf.fit(offline_rss, offline_location).predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, 'm'
```

5. Re:室内定位系列（一）——WiFi位置指纹（译）

你好，请问目前是否用到PSO训练ANN的方式来作指纹定位算法。

--king-blues

阅读排行榜

1. 室内定位系列（一）——WiFi位置指纹（译）(1789)

2. 统计信号处理-简单看看克拉美罗界(1016)

3. Tensorflow使用环境配置(896)

4. 室内定位系列（〇）——从人耳听觉定位原理到室内定位技术(689)

5. 室内定位系列（五）——目标跟踪（卡尔曼滤波）(634)

评论排行榜

1. 统计信号处理-简单看看克拉美罗界(3)

2. 室内定位系列（五）——目标跟踪（卡尔曼滤波）(3)

3. 室内定位系列（一）——WiFi位置指纹（译）(2)

推荐排行榜

1. 统计信号处理-简单看看克拉美罗界(2)

2. 室内定位系列（一）——WiFi位置指纹（译）(2)

3. 使用Java练习算法常用的基本操作(1)

4. 室内定位系列（〇）——从人耳听觉定位原理到室内定位技术(1)

5. 小技能——markdown(1)

```
accuracy:  3.83255676918 m
```

Lasso回归

```
from sklearn.linear_model import MultiTaskLassoCV
clf = MultiTaskLassoCV(alphas=np.logspace(-4, 4, 10))
predictions = clf.fit(offline_rss, offline_location).predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, 'm'
```

```
accuracy:  3.83244688001 m
```

Elastic Net（弹性网回归）

```
from sklearn.linear_model import MultiTaskElasticNetCV
clf = MultiTaskElasticNetCV(alphas=np.logspace(-4, 4, 10))
predictions = clf.fit(offline_rss, offline_location).predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, 'm'
```

```
accuracy:  3.832486036 m
```

Bayesian Ridge Regression（贝叶斯岭回归）

```
from sklearn.linear_model import BayesianRidge
from sklearn.multioutput import MultiOutputRegressor
clf = MultiOutputRegressor(BayesianRidge())
predictions = clf.fit(offline_rss, offline_location).predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, "m"
```

```
accuracy:  3.83243319129 m
```

Gradient Boosting for regression（梯度提升）

```
from sklearn import ensemble
from sklearn.multioutput import MultiOutputRegressor
clf = MultiOutputRegressor(ensemble.GradientBoostingRegressor(n_estimators=100,
max_depth=10))
%time clf.fit(offline_rss, offline_location)
%time predictions = clf.predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, "m"
```

```
Wall time: 43.4 s
Wall time: 17 ms
accuracy:  2.22100945095 m
```

Multi-layer Perceptron regressor（神经网络多层感知器）

```
from sklearn.neural_network import MLPRegressor
clf = MLPRegressor(hidden_layer_sizes=(100, 100))
%time clf.fit(offline_rss, offline_location)
%time predictions = clf.predict(rss)
acc = accuracy(predictions, trace)
print "accuracy: ", acc/100, "m"
```

```
Wall time: 1min 1s
Wall time: 6 ms
accuracy:  2.4517504109 m
```

总结

上面的几个线性回归模型显然效果太差，这里汇总一下其他的一些回归模型：

算法	定位精度
knn	2.24m
logistic regression	3.09m
support vector machine	2.25m
random forest	2.21m
Gradient Boosting for regression	2.22m
Multi-layer Perceptron regressor	2.45m

从大致的定位精度上看，KNN、SVM、RF、GBDT这四个模型比较好（上面很多算法并没有仔细地调参数，这个结果也比较粗略，神经网络完全不知道怎么去调...）。此外要注意的是，SVM训练速度慢，调参太麻烦，KNN进行预测时的时间复杂度应该是和训练数据量成正比的，从定位的实时性上应该不如RF和GBDT。

作者：rubbninja
出处：http://www.cnblogs.com/rubbninja/
关于作者：目前主要研究领域为机器学习与无线定位技术，欢迎讨论与指正！

标签：室内定位，机器学习，室内定位系列

rubbninja
关注 - 3
粉丝 - 18
+加关注

0 0

« 上一篇：室内定位系列（三）——位置指纹法的实现（KNN）
» 下一篇：室内定位系列（五）——目标跟踪（卡尔曼滤波）

posted @ 2016-12-16 14:49 rubbninja 阅读(385) 评论(0) 编辑 收藏