机器学习之：异常检测

王尼玛 · 1 年前

转自博客：infosec-wiki.com/?...

一、关于异常检测

异常检测（outlier detection）在以下场景：

- 数据预处理
- 病毒木马检测
- 工业制造产品检测
- 网络流量检测

等，有着重要的作用。由于在以上场景中，异常的数据量都是很少的一部分，因此诸如：SVM、逻辑回归等分类算法，都不适用，因为：

> 监督学习算法适用于有大量的正向样本，也有大量的负向样本，有足够的样本让算法去学习其特征，且未来新出现的样本与训练样本分布一致。

以下是异常检测和监督学习相关算法的适用范围：

> 异常检测：信用卡诈骗、制造业产品异常检测、数据中心机器异常检测、入侵检测
>
> 监督学习：垃圾邮件识别、新闻分类

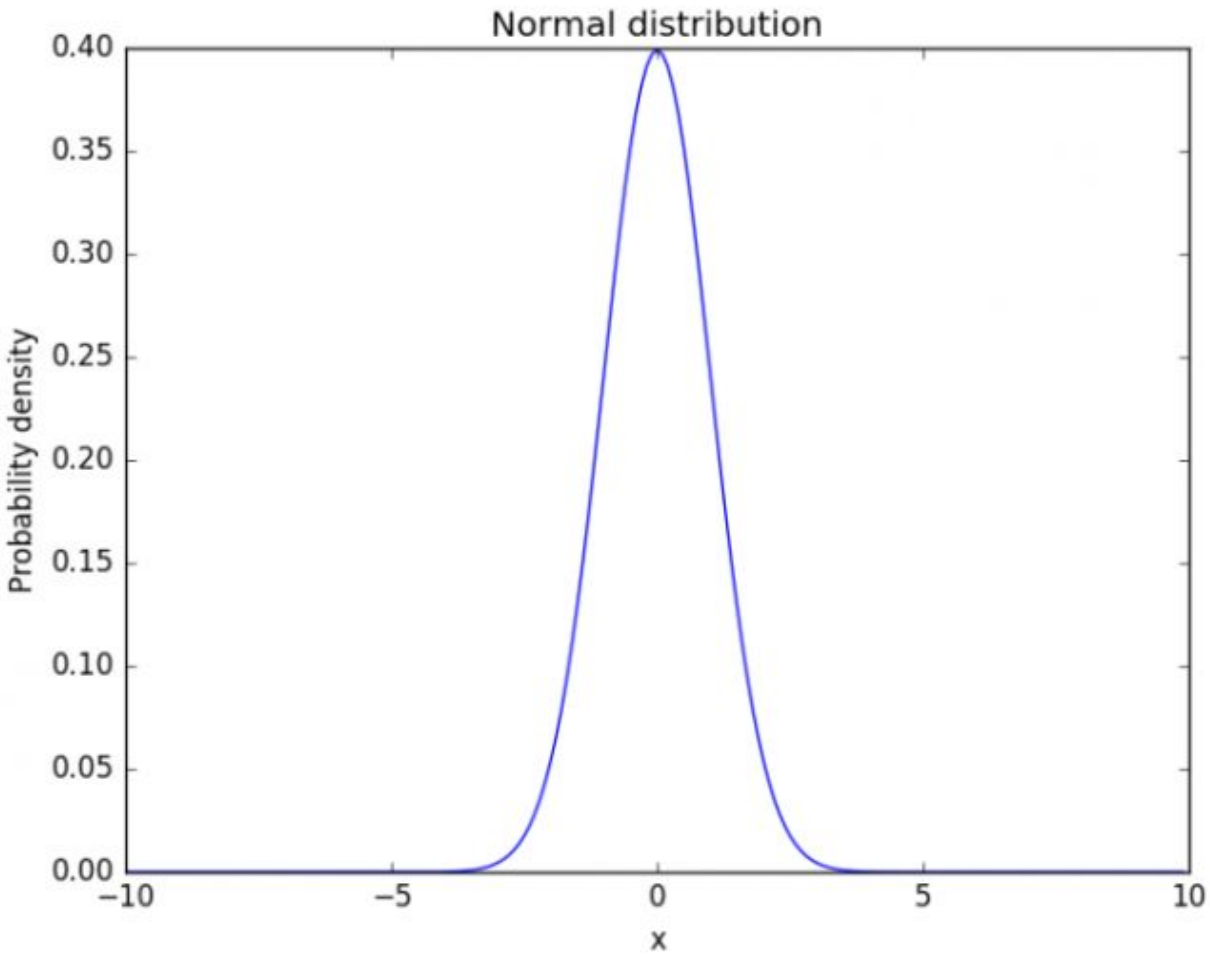
二、异常检测算法

1. 基于统计与数据分布

假设数据集应满足正态分布（Normal Distribution），即：

$$P(x;\mu,\sigma) = \frac{1}{\sqrt{\quad}} \exp\left(-\frac{(x-\mu)^2}{\quad}\right), \qquad x \in [-\infty;\infty]$$

知



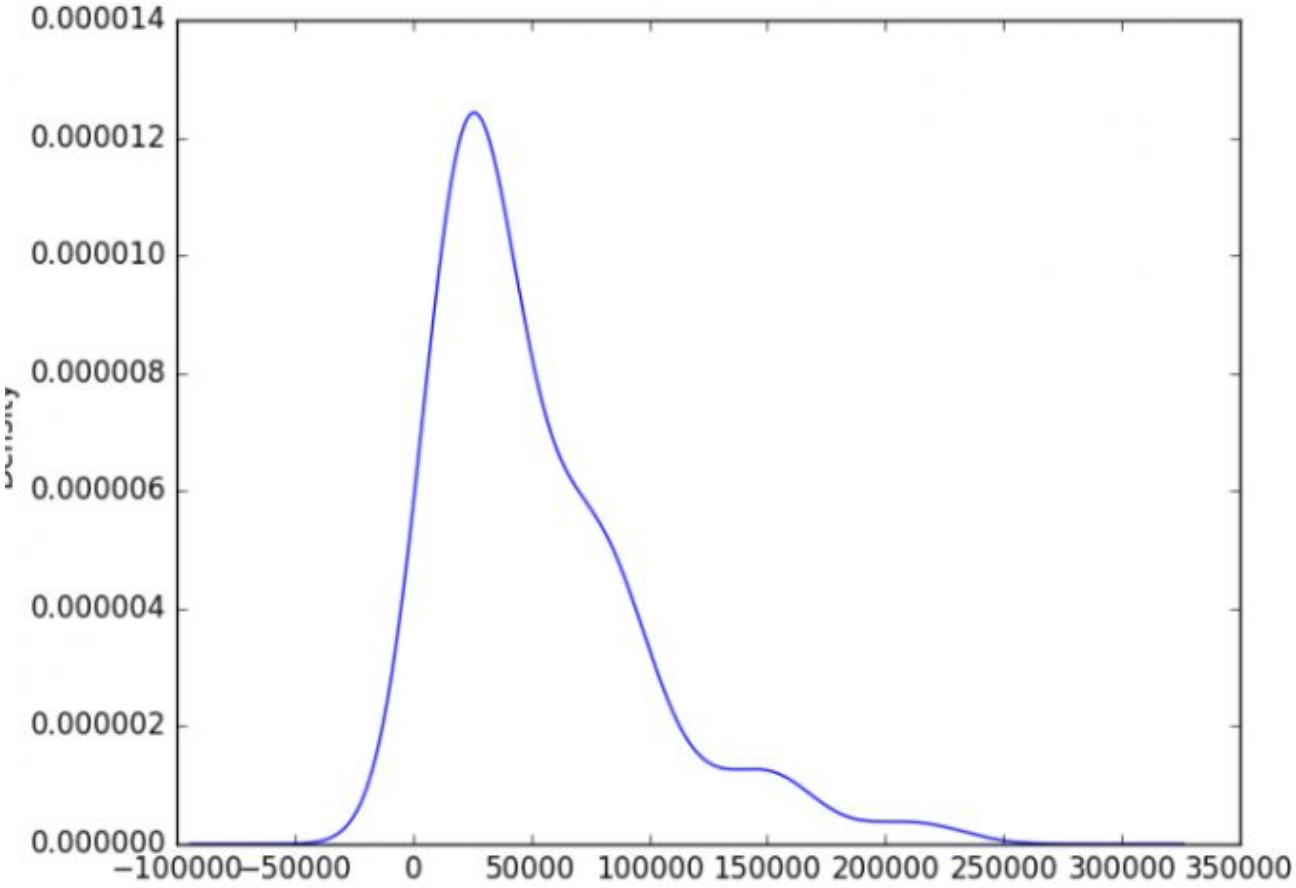
当满足上图训练数据的正态分布，如果x的值大于4或者小于-4，都可以认为是异常值。

以下以“600680”股票成交量为例：

```
import tushare
from matplotlib import pyplot as plt

df = tushare.get_hist_data("600680")
v = df[-90: ].volume
v.plot("kde")
plt.show()
```

近三个月，成交量大于200000就可以认为发生了异常（天量，嗯，要注意风险了……）



算法示例：

Anomaly detection algorithm

1. Choose features x_i that you think might be indicative of anomalous examples. $\{x^{(1)}, \ldots, x^{(m)}\}$
2. Fit parameters $\mu_1, \ldots, \mu_n, \sigma_1^2, \ldots, \sigma_n^2$

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \mu_j)^2$$

3. Given new example x, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

Anomaly if $p(x) < \varepsilon$

2. 箱线图分析

箱线图，不做过多说明了：

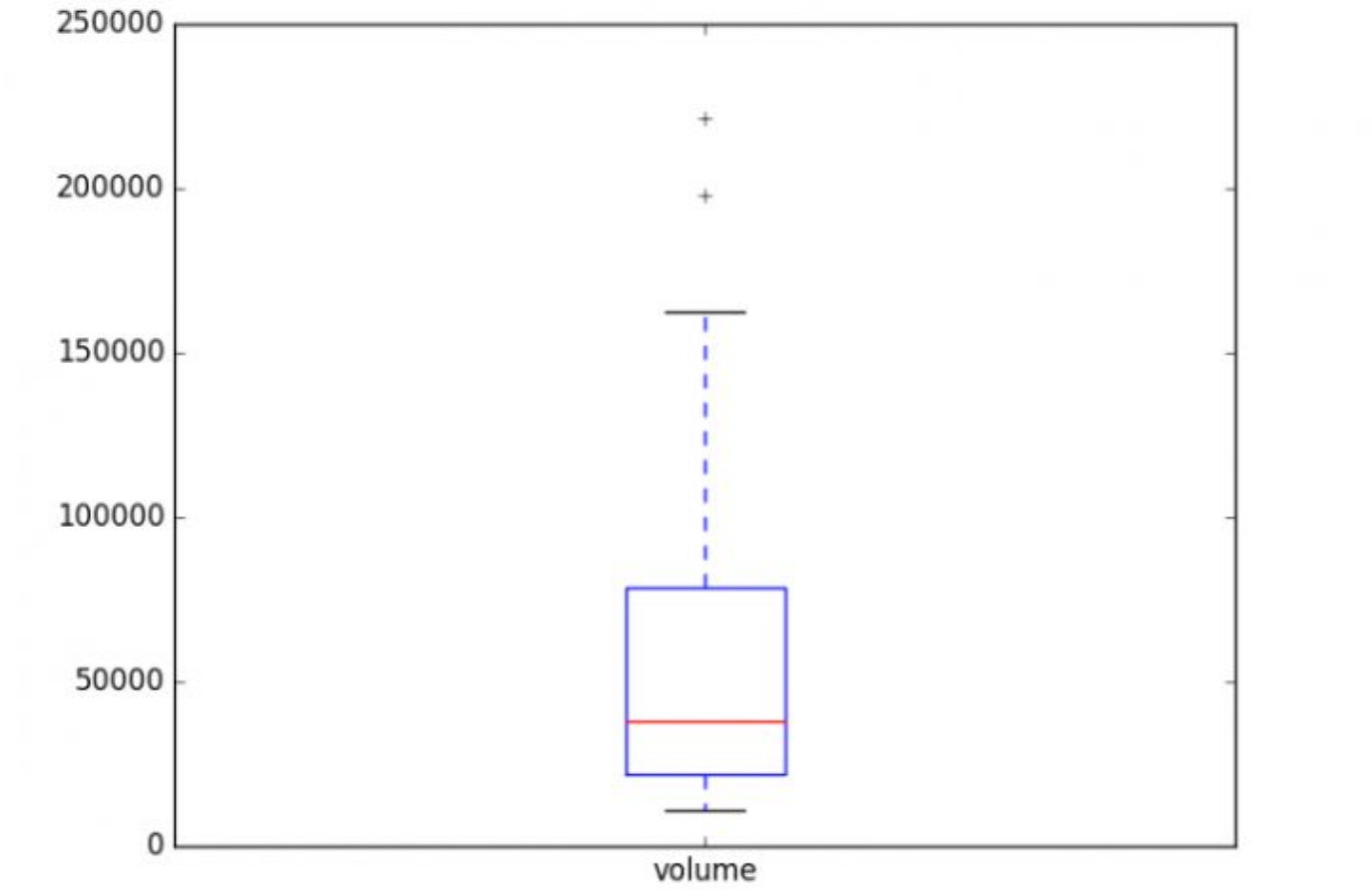
```
import tushare
from matplotlib import pyplot as plt

df = tushare.get_hist_data("600680")
v = df[-90: ].volume
v.plot("kde")
plt.show()
```

图：


```
import tushare
from matplotlib import pyplot as plt

df = tushare.get_hist_data("600680")
v = df[-90: ].volume
v.plot("kde")
plt.show()
```



大体可以知道，该股票在成交量少于20000，或者成交量大于80000，就应该提高警惕啦！

3. 基于距离/密度

典型的算法是：“局部异常因子算法-Local Outlier Factor”，该算法通过引入“k-distance，第k距离”、“k-distance neighborhood，第k距离邻域”、“reach-distance，可达距离”、以及“local reachability density，局部可达密度”和“local outlier factor，局部离群因子”，来发现异常点，详情可参考：异常点/离群点检测算法--LOF - wangyibo0201的博客 - 博客频道 - CSDN.NET

4. 基于划分思想

典型的算法是 “孤立森林，Isolation Forest”，其思想是：

> 假设我们用一个随机超平面来切割（split）数据空间（data space），切一次可以生成两个子空间（想象拿刀切蛋糕一分为二）。之后我们再继续用一个随机超平面来切割每个子空间，循环下去，直到每子空间里面只有一个数据点为止。直观上来讲，我们可以发现那些密度很高的簇是可以被切很多次才会停止切割，但是那些密度很低的点很容易很早的就停到一个子空间

Algorithm 1 $iTree(X, e, h)$

Input: X - input data; e - current height; h - height limit.
Output: an $iTree$.

1: **if** $e \geqslant h$ OR $|X| \leqslant 1$ **then**
2: **return** $exNode\{Size \leftarrow |X|\}$;
3: **else**
4: Randomly select an attribute q;
5: Randomly select a split point p between min and max values of attribute q in X;
6: $X_l \leftarrow filter(X, q < p)$, $X_r \leftarrow filter(X, q \geqslant p)$;
7: **return** $inNode\{ Left \leftarrow iTree(X_l, e+1, h),$
$Right \leftarrow iTree(X_r, e+1, h),$
$SplitAttr \leftarrow q, SplitValue \leftarrow p\}$;
8: **end if**

详情见：

iForest （ Isolation Forest ）孤立森林 异常检测 入门篇IsolationForest example

示例代码：

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import IsolationForest

rng = np.random.RandomState(42)

# Generate train data
X = 0.3 * rng.randn(100, 2)
X_train = np.r_[X + 1, X - 3, X - 5, X + 6]
# Generate some regular novel observations
X = 0.3 * rng.randn(20, 2)
X_test = np.r_[X + 1, X - 3, X - 5, X + 6]
# Generate some abnormal novel observations
X_outliers = rng.uniform(low=-8, high=8, size=(20, 2))

# fit the model
clf = IsolationForest(max_samples=100*2, random_state=rng)
clf.fit(X_train)
y_pred_train = clf.predict(X_train)
y_pred_test = clf.predict(X_test)
y_pred_outliers = clf.predict(X_outliers)
```

```
xx, yy = np.meshgrid(np.linspace(-8, 8, 50), np.linspace(-8, 8, 50))
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.title("IsolationForest")
plt.contourf(xx, yy, Z, cmap=plt.cm.Blues_r)

b1 = plt.scatter(X_train[:, 0], X_train[:, 1], c='white')
b2 = plt.scatter(X_test[:, 0], X_test[:, 1], c='green')
c = plt.scatter(X_outliers[:, 0], X_outliers[:, 1], c='red')
plt.axis('tight')
plt.xlim((-8, 8))
plt.ylim((-8, 8))
plt.legend([b1, b2, c],
           ["training observations",
            "new regular observations", "new abnormal observations"],
           loc="upper left")
plt.show()
```

结果如下，其中：红色即为异常点，白色是训练集，绿色是测试数据

注意：孤立森林不适用于特别高维的数据。由于每次切数据空间都是随机选取一个维度，建完树后仍然有大量的维度信息没有被使用，导致算法可靠性降低。高维空间还可能存在大量噪音维度或无关维度（irrelevant attributes），影响树的构建。孤立森林算法具有线性时间复杂度。因为是ensemble的方法，所以可以用在含有海量数据的数据集上面。通常树的数量越多，算法越稳定。由于每棵树都是互相独立生成的，因此可以部署在大规模分布式系统上来加速运算。

5. 其他算法

包括：One-class SVM 以及 Elliptic Envelope 等。

参考：2.7. Novelty and Outlier Detection

6. 值得一提

这些算法里面，孤立森林和局部异常因子算法相比之下，效果是最好的。

文章参考资料：

Isolation ForestLOF – Identifying Density-Based Local Outlierskamidox.com异常检测（anomaly detection）如何在Python中实现这五类强大的概率分布 - Python - 伯乐在线博客频道 - CSDN.NET2.7. Novelty and Outlier DetectioniForest（Isolation Forest）孤立森林 异常检测
入门篇异常点检测算法（一）
异常值检测算法（二）
异常点检测算法（三）
异常点检测算法综述

机器学习 异常处理

熊亭

你好，我想请教一个问题，如果没有训练数据和测试数据，就是我拿到一组数据我只是要对这组数据要做异常点检测，我还能用iforest吗，就是说我也不知道这组数据里面哪些是异常点啊

8 个月前

王尼玛（作者） 回复 **熊亭** 查看对话

iforest本来就是无监督学习

8 个月前

熊亭 回复 **王尼玛**（作者） 查看对话

可是我在调用sklearn里面提供的方法iforest时，把待检测数据输入进去，每次运行后显示的结果都不相同，不知道是为什么呢

8 个月前

任远 回复 **熊亭** 查看对话

这个算法不是带有随机性么...

6 个月前

steven 回复 **熊亭** 查看对话

iforest是无监督的学习方法，训练模型不需要类标。如果你拥有一批无类标的数据，可以训练模型，只是不知道训练好的模型的效果如何

3 个月前