13 | 模型学习和预测：如何检查流数据异常?

今天，我们来讨论流数据中有关模型学习和预测的问题。相比前面讲述的四类算法，模型学习和预测，是一个内容更加丰富，也需要更多数据科学专业知识的领域。它的交叉性非常强，往往在具体实践时，需要工程人员（如 Java 工程师和大数据工程师）和数据人员（如数据科学家和数据分析师）紧密的配合，才能实现一个可以切实落地的方案。

所以，如果你是一个 Java 工程师或者大数据工程师，那么掌握在流数据上进行模型学习和预测的基本原理和通用方法，是与数据人员更加高效配合，并设计出最佳落地方案的基础。

数据人员在为数据建模时，通常有两种不同的思路，一种是统计学习模型，另一种是机器学习模型。

- 统计学习模型以统计分析为基础，偏向于挖掘数据内部产生的机制，它更加注重模型和数据的可解释性。比如，使用泊松分布来预测网站访问次数，用ARIMA 模型（自回归移动平均模型）来进行时间序列预测等。
- 而机器学习模型，则是以各种机器学习方法为基础，偏向于用历史数据来预测未来数据，更加注重模型的预测效果。比如，用决策树来进行信贷风险评估，用人工神经网络进行人脸检测等。

而如果是从业务使用场景的角度看，在流数据上进行模型学习和预测，主要目标通常可以归为两类，**一类是进行异常检测，另一类则是预测未来**。

所以，今天我们就分别讲解下，如何用统计学习模型进行异常检测，以及如何用机器学习模型预测未来。期望通过今天的课程，帮助你理解在流数据上，进行异常检测或者预测未来的基本原理和通用方法。

针对流数据的模型学习和预测

随着流计算技术的普及，越来越多原本针对离线数据的统计和机器学习方法，也开始被用于流数据。

比如在传统风控系统中，模型的训练是通过离线批处理进行的。比如，将每天线上新到的数据先保存到数据库或 Hadoop 里，然后选择一个一天中业务低峰的时间段，通过批处理的方式，训练出新的模型和参数，之后再将新模型和参数运用到线上。

但在移动互联网时代，交易发生得越来越频繁，不仅数据量大，为了保证用户体验，还需要能够实时处理。因此，为了更加及时地应对模型参数的变化，也为了避免数据的积压，风控系统越来越多地开始采用流计算技术，实现风控模型的实时在线训练和更新。

再比如在异常检测中，我们会在线统计和估计变量的分布参数，然后根据训练出的分布模型，判断变量之后的取值是否属于异常。这种同时在线更新和预测的做法，在流计算应用中也越来越常见。

总的来说，在数据流上进行模型学习，并根据模型做出判断或预测，是将统计学习和机器学习的理论方法，推广应用在流数据上的结果。**流数据不断输入模型学习算法，实时更新模型参数，在线训练得到模型，能够更加及时和真切地描述当时的状况。**

使用统计学习模型进行异常检测

我们先来看统计学习模型的问题。在使用统计学习模型建模时，其核心思路就是，选择一个分布模型，然后使用样本数据估计出这个分布模型的参数。不同的分布模型，确定参数的方法不同。

比如，我们经常使用的分布有 0-1 分布、二项分布、多项式分布、泊松分布、均匀分布、正态分布和指数分布等。这些分布我们在大学时期的《概率论与数理统计》中都学过。如果你正需要使用这块的知识的话，建议可以回顾下大学时的教材。

拿正态分布来说，它的分布模型如下：

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

而我们要估计的参数，就是均值 μ 和标准差 σ：

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_i$$

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N} (x_i - \mu)^2$$

其中，$x_1, x_2, \ldots, x_N$ 是样本数据。

所以，当我们选择使用的统计模型是正态分布时，就只需要估计出均值 μ 和标准差 σ 即可。

而针对在流数据上计算**均值**和**方差**的方法，我们已经在模块 3 第 10 课中，讲解有关“**时间维度聚合值**”计算的内容时详细讨论过了。

你看，是不是很神奇！我们一番操作后，就将一个统计学习模型的问题，转化为了我们前面详细讨论过的流数据处理问题。

其实，像其他的 0-1 分布、二项分布、多项式分布、泊松分布、均匀分布、正态分布和指数分布等，也都可以采用类似的方法。换言之，最后对这些分布参数的估计，也都会转化为各种类似于计数、均值、方差等统计值的计算。

参数随时间变化的统计模型

上面我们已经知道了如何确定统计学习中分布模型参数的方法。不过，这里我们还需要处理一个流数据场景下经常遇到的问题，也就是模型本身随时间变化而变化的。

在传统针对离线数据做统计分析时，构建模型的重要目的，是解释既有数据，因此模型估计出来的参数是不变的。但是在针对流数据进行统计建模时，虽然确定分布参数依旧重要，但多了另外一个任务，即需要处理分布参数随时间变化而变化的问题。

说到分布模型的参数会随时间变化的问题，一个很常见的例子就是，一家商店晚上的客流量，一般会比早上多，然后周末的客流量，也会比工作日的客流量多。如果我们用柏松分布，来对每个小时的客流量建模，那很明显这个柏松分布的期望，是随着时间在变化的。

所以，当在流数据上构建统计学习模型时，模型通常会包含两个层面：

- 一个层面是，随机变量在一段时间窗口内的概率分布函数；
- 另一个层面则是，概率分布函数的参数是随时间变化的变量。

比如，考虑了期望随时间变化的柏松分布，就重新定义为如下的公式：

$$P(X=k|\lambda(t))=\frac{\lambda(t)^k}{k!}*e^{-\lambda(t)},k=0,1,2,\cdots$$

其中，$\lambda(t)$ 是 t 时刻对 λ 值的估计。

那具体怎样进行估计呢？估计 λ(t) 本身比较简单，因为对于柏松分布而言，其期望就是 λ(t) 的无偏估计。所以，我们同样可以采用第 10 课时中计算“时间维度聚合值”的方式，计算出 λ(t) 的均值作为期望值即可。

这里稍微还需要考虑下的是，应该怎样更新这个估计值。可以有两种更新方法：

- 一种是，逐事件更新。也就是每来一个新数据就重新估计一次；
- 另一种是，定周期更新。也就是每隔一段时间重新估计一次，比如每小时重新估计一次。

这两种方式都是不错的选择，你可以根据具体业务场景，选择其中一种方法即可。

现在，我们已经估计出了统计学习模型的参数。换言之，我们已经训练出我们设定的统计学习模型了。那我们拿这个模型有什么用呢？

前面我们已经说过，我们在流数据上进行模型学习的主要目的，无非就是异常检测，或者预测未来。异常检测和预测未来本质上是一样的数学原理，无非使用的形式不同而已。

所以，接下来我们就重点讨论如何进行异常检测的问题。而一旦说到“检测”，我们就不得不说到在统计检测中，最基础也是最重要的 P-value 检验方法了。

P-value 检验

那什么是 P-value 检验呢？

以小明和小花抛硬币为例。在开始抛硬币前，小花押“字”朝上，小明押“花”朝上。确定之后，小明从口袋拿出一个硬币抛了 1 次，结果是“花”朝上。小花不服，要求再来一局……然后就是，反反复复抛了 10 次。但结果很令人惊讶，10 次抛硬币的结果中，有 9 次“花”朝上，只有 1 次“字”朝上。对于这个结果，小花更加不服气了，觉得小明的硬币一定是个“假”硬币。

那怎样科学地判断小明的硬币是“真”硬币还是“假”硬币呢？这里，就可以用到 P-value 检验的方法了。

首先，我们假定硬币是“真”的，也就是“字”朝上的概率和“花”朝上的概率都是 0.5 ，那么抛 10 次硬币，只有不超过 1 次“字”朝上的概率，就是下面的计算公式：

$$C(10,0)*0.5^0*(1-0.5)^{10}+C(10,1)*0.5^1*(1-0.5)^9=\frac{11}{1024}\approx 0.01$$

这么一算，不超过 1 次“字”朝上的概率，只有区区百分之一左右。小花当然可以理直气壮地，怀疑小明对硬币做了手脚。

在上面的这个例子中， 0.01 就是所谓的 P-value 。由于 P-value 很小，故而可以推翻我们前面的“真”硬币假设。这就是 P-Value 检验的方法。

不过，当我们把统计学习模型，运用在实时异常检测时， P-value 又有了一层新的含义。

比如，统计页面浏览量的场景。根据过往经验和历史数据的统计，我们认为某个页面每秒钟的访问次数，应该符合期望为 6 的泊松分布。可是实际计算的结果，却显示当前这一秒，该页面的访问量达到了 16 次。

那这究竟是正常流量波动，还是系统受到了攻击呢？

根据柏松分布，我们可以计算出一秒钟内，页面访问量超过 16 次的概率为万分之五左右（P(X ≥ 16) = 0.0005）。这个概率很小，意味着这秒钟的页面访问量，和我们的预期并不相符。

但这一次，我们并不是像之前 P-value 检验中那样拒绝假设，而是反过来，断定一秒钟 16 次的页面访问量是异常行为，这也预示着我们的系统可能正受到攻击。

总的来说，使用 P-Value 检测异常的过程，就是先确定一个分布模型，然后计算在这个模型下，一件事情发生的概率。如果这个概率很小，就意味着“事出反常必有妖”，也就是我们所说的“异常”了。

至此，我们就可以总结下针对流数据进行异常检测的完整过程了：

- 首先，针对实际的业务场景，结合业务专家或数据人员的经验，选择一个合适的分布模型。比如，前面我们在统计网站访问人数时，使用了泊松分布；
- 其次，分析分布模型需要估计的参数。比如，柏松分布需要估计的参数就是期望，或者说是均值；
- 然后，由于流数据场景下，模型的参数通常是随时间而变化的，所以需要在线训练和更新模型参数。比如，对于均值和方差，可以使用课时 10 中“时间维度聚合值”计算的方法；
- 接着，当估计出模型的参数后，我们就得到了完整的分布模型，从而可以根据这个分布模型，计算一个新到的数据在这个分布模型下的 P-value 值。如果 P-value 很小（一般小于 5%），我们就认为这个新到的数据是一个“异常”了；
- 之后，就是不断重复在线训练和更新模型参数、针对 P-value 进行异常检测的过程了。

以上就是针对流数据进行异常检测的一般过程了。或许以后你在工作中，数据人员会针对业务需要选择更加复杂的统计模型。但是，它们的实现方法与上面描述的过程基本上是一致的。所以，希望你能够掌握这里的方法。

使用机器学习模型进行预测

接下来，我们再来看另一类学习模型，也就是“机器学习模型”。

相比统计学习模型，使用机器学习的方法构建模型，有个极大的好处，也就是我们不需要对数据内在的产生机理，有任何的先验知识。基本上，你只需要**准备好模型**，以及模型的输入**特征向量**，然后确定一个要最优化的**目标函数**，就可以让机器自动去发现输入数据和输出结果之间的对应关系了。最后，训练好的模型就可用于分类或预测。

这么讲可能有些抽象，我们来看个机器学习领域最典型的例子，也就是使用人工神经网络识别手写数字。当使用神经网络识别手写数字时，以手写数字图片的 28 * 28 像素矩阵作为**特征向量**，然后设置好**神经网络各层的结构（包括将输入层设置为 784 个节点分别对应输入图片的 28 * 28 个像素，将隐藏层设置为 20 个节点，将输出层为 10 个节点分别对应 0～9 这 10 个数字）。那么优化目标**就应该是每个手写数字图片在经过神经网络后，最终唯一激活与图片中数字相对应的输出层节点。

当确定好**特征向量**、**神经网络模型**和**优化目标**这三者后，就可以用一些标记好数值的手写数字图片作为样本，开始训练神经网络。训练好的神经网络，当输入一个写有 1 的图片时，它就会激活代表数字 1 的那个输出节点。这样，也就完成了对未知手写数字图片所代表真实数值的预测了。

下图 1 就是这么一个神经网络识别数字的简单示意图。

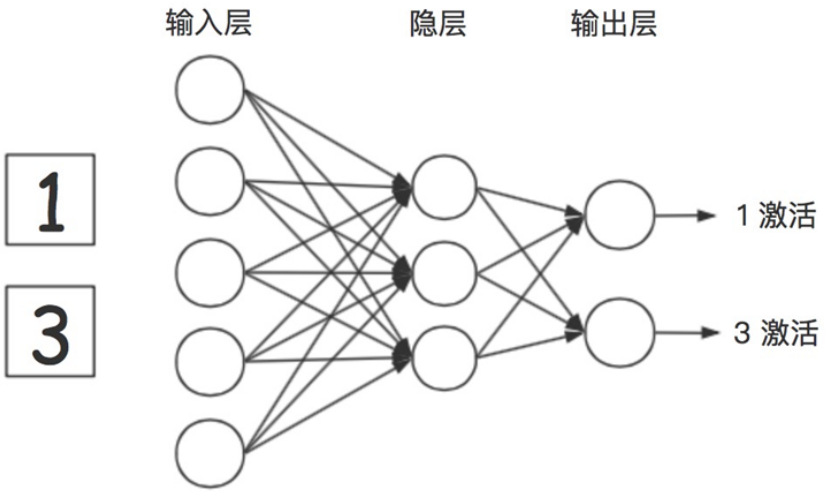

图 1 神经网络识别数字简单示意图

一般来说，除了神经网络外，我们经常使用的机器学习模型，还有线性回归、逻辑回归、朴素贝叶斯、决策树和随机森林等。不过为了简单起见，接下来我就以最基本的**线性回归模型**为例，讲解一下在流数据上，是如何进行机器学习的。

当然，这里还是需要说一句，虽然简单，但其实线性回归模型，以及由它衍生出的广义线性回归模型，比如逻辑回归，在经济学和金融领域的使用是非常广泛的，甚至很多时候它会是唯一选择。原因无它，正是因为线性回归模型简单，并且能够很好地解释输入（也就是自变量）和输出（也就是因变量）之间的因果关系，所以它更为大家所接受。而其他机器学习模型，可能效果比线性回归好，但是因为不能很好地解释输入和输出之间的因果关系，模型判断出错后不好解释，所以也就只能作罢。

回到线性回归模型上来。在计量经济学里，经常需要对“时间序列”进行分析，比如根据过去几年的人口数量，来预测未来的人口变化趋势。而“**时间序列**”，在我们技术人的眼里，说白了就是**流数据**。所以，现在我们就尝试用线性回归模型，来对上证指数收盘价组成的时间序列进行分析，以期望能够预测出下一个交易日的上证指数。

那具体怎么做呢？这里，我们选取了 2003 年全年的上证指数收盘价，作为分析的流数据。我们的目标是，用最近 10 个交易日的收盘价，预测下一个交易日的收盘价。

所以线性回归模型的输入，就是最近 10 个交易日的收盘价。而输出，则是下一个交易日的收盘价。

下面就是实现线性回归模型，训练和预测的代码（参考完整代码）。

```
int numberOfVariables = 10;
UpdatingMultipleLinearRegression rm = new MillerUpdatingRegression(numberOfVariables, true);
Queue<Double> xPrices = new LinkedList<>();
double[] predictPrices = new double[prices.length];
for (int i = 0; i < prices.length; i++) {
    double price = prices[i];
    // 用于训练和预测的数据量不足，所以跳过继续执行
    if (i < numberOfVariables) {
        xPrices.add(price);
        predictPrices[i] = 0;
        continue;
    }
    if (i <= numberOfVariables * 2 + 1) {
        // 用于预测的数据量不足，所以跳过继续执行
        predictPrices[i] = 0;
    } else {
        // 根据模型进行预测
        double params[] = rm.regress().getParameterEstimates();
        List<Double> xpList = new LinkedList<>();
        xpList.add(1d);
        xpList.addAll(xPrices);
        double[] x_p = ArrayUtils.toPrimitive(xpList.toArray(new Double[0]));
        double y_p = new ArrayRealVector(x_p).dotProduct(new ArrayRealVector(params));
        predictPrices[i] = y_p;
    }
    // 更新模型
    double[] x = ArrayUtils.toPrimitive(xPrices.toArray(new Double[0]));
    double y = price;
    rm.addObservation(x, y);
    xPrices.add(price);
    xPrices.remove();
}
```

可以看到，在上面的代码中，我们主要使用了能够增量更新训练的线性回归模型，也就是 MillerUpdatingRegression 类。其中，在进行训练和预测时，我们是这样做的。

首先，按照时间顺序，依次将每天的收盘价 price 和前 10 个交易日的收盘价 xPrices 分别作为线性回归模型的因变量和自变量，构成一组观察记录。

然后，将这组观察记录，通过 addObservation 方法更新到模型中去。

接着，使用 regress 函数获得线性回归模型的参数。这里，在 regress 函数内部计算线性回归模型参数的过程，就是模型训练的过程。

最后，用 dotProduct 函数，将过去 10 个交易日的收盘价 xPrices 与训练出的模型参数，进行点乘计算，就得到了下一个交易日的收盘价预测值 y_p 。

下图 1 就是将上证指数收盘价的实际曲线与预测曲线进行比较的结果。

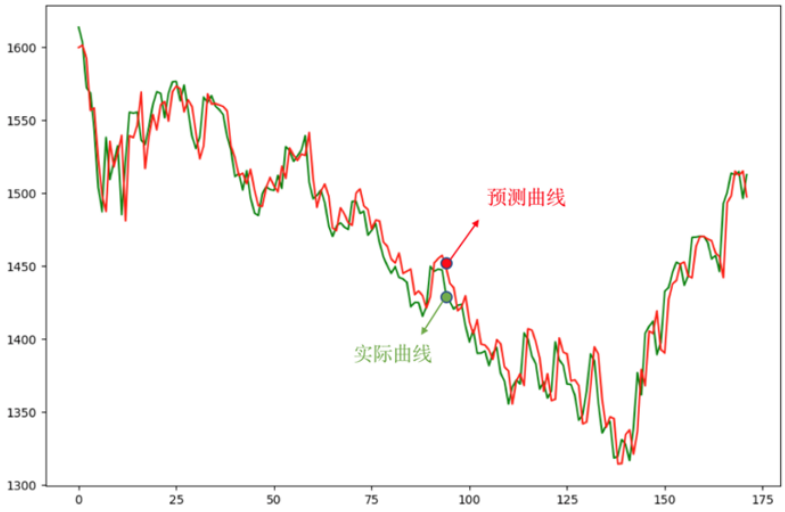

图 1 使用线性回归模型预测下一个交易日的上证指数

从上图可以看出，预测收盘价曲线，能够比较好地跟随真实收盘价曲线变化的趋势。但是却并不能非常好地预测真实收盘价曲线的突变。我们对比预测曲线和实际曲线可以看到，预测收盘价曲线总是比真实收盘价曲线慢半拍。这也意味着，我们并不能指望，用这个线性回归模型从上证指数的变化中获利。

当然，这里用线性回归模型预测上证指数，只是一个演示性的例子。我们不能用线性回归模型预测股价。主要原因还是因为股价的变化本身是非线性的，有很多非理性的因素在影响着股价，比如市场情绪、突发事件等。正是因为这些非理性的因素，让线性模型不能非常好地预测这些“意外”的变化。毕竟，如果真这么容易预测股市走向的话，人人都成巴菲特了。

不过，线性回归模型也有其更加适用的场景。比如在风控系统中，可以用一种广义线性回归模型，也就是逻辑回归，来构建风控模型。逻辑回归训练出来的模型，能够非常好地转化为评分卡，被经济和金融领域的分析人员广泛使用。

小结

今天，我们讲解了在流数据上进行模型和机器学习的通用性方法。针对流数据的异常检测，以及根据流数据进行预测，已经在越来越多的业务场景被使用，所以掌握这种通用性方法是非常必要的。

总的来说，今天课程的内容交叉性比较强。因为对于传统的 Java 工程师和大数据工程师来说，他们不太会涉及具体数学模型的设计工作，而对于传统的数据科学家或数据分析师来说，他们工作的重点又会放在建模以及模型效果分析上面，不太会直接针对流数据做工程化实现。所以，如果有一个这两者都懂的开发人员的话，就可以将工程人员和数据人员的工作有效地组织起来。目前，这种人才是非常稀缺的，所以希望你能够成为这方面的人才，这也是今天课程的另一个重要目标。

另外，我详细讲解的几个模型，都是日常开发中相对基础但又广泛使用的模型。如果在后续的工作中，数据人员采用了更加复杂的模型，而你又发现很难将他们的模型改成针对流数据的算法的话，那么你可以退而求其次选择使用 Lambda 架构。我们在 11 课时中计算关联图谱时，就使用过 Lambda 架构，在后面的课程中还会再专门讲解 Lambda 架构。

最后，留一个问题给你思考。假设现在你准备做量化交易，那能否使用 Flink 来做量化交易呢？使用 Flink 做量化交易又会有什么优势或劣势呢？可以将你的想法和问题写到留言区。

下面是本课内容的脑图，以便于你理解。