|  |
| --- |
| **机器学习** |
| **XX案例名** |
| 主 研 人：  参 研 人： |
| 审 核 人：  方 向：工业算法案例研究  版 本 号：A |
|  |

**版本更新**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **更新描述** | **作者** |
| 2018/10/31 | A | 初稿 | 李琦 |

**选题表格**

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 竞赛名 | 竞赛背景描述（50字以内） | 类型（分类/回归） |
| 2018/10/31 | XX | 预测维基百科页面的未来流量。竞赛重点是预测多个时间序列的未来价值问题 | 回归 |

目录

[1. 背景描述](#_Toc15333_WPSOffice_Level1) [3](#_Toc15333_WPSOffice_Level1)

[1.1 竞赛赛题描述](#_Toc30925_WPSOffice_Level2) [4](#_Toc30925_WPSOffice_Level2)

[1.2 评估指标描述](#_Toc5630_WPSOffice_Level2) [4](#_Toc5630_WPSOffice_Level2)

[2. 数据来源及描述性统计分析](#_Toc30925_WPSOffice_Level1) [4](#_Toc30925_WPSOffice_Level1)

[2.1 大赛数据来源](#_Toc22205_WPSOffice_Level2) [4](#_Toc22205_WPSOffice_Level2)

[2.2 数据的描述性统计](#_Toc19007_WPSOffice_Level2) [4](#_Toc19007_WPSOffice_Level2)

[2.2.1 数据基本情况描述：](#_Toc30925_WPSOffice_Level3) [4](#_Toc30925_WPSOffice_Level3)

[2.2.2 数据字段介绍：](#_Toc5630_WPSOffice_Level3) [5](#_Toc5630_WPSOffice_Level3)

[2.2.3 数据描述性统计](#_Toc22205_WPSOffice_Level3) [6](#_Toc22205_WPSOffice_Level3)

[3. 优秀算法思路 12](#_Toc27744_WPSOffice_Level1)

[3.1 方案一](#_Toc4714_WPSOffice_Level2) [12](#_Toc4714_WPSOffice_Level2)

[3.1.1 方案一数据预处理及特征工程部分方案](#_Toc19007_WPSOffice_Level3) [13](#_Toc19007_WPSOffice_Level3)

[3.1.2 方案一模型设计、建立部分方案](#_Toc4714_WPSOffice_Level3) [14](#_Toc4714_WPSOffice_Level3)

[3.1.3 方案一结果、排名等](#_Toc17931_WPSOffice_Level3) [20](#_Toc17931_WPSOffice_Level3)

[3.1.4 方案一算法流程图](#_Toc27185_WPSOffice_Level3) [20](#_Toc27185_WPSOffice_Level3)

[3.2 方案二](#_Toc17931_WPSOffice_Level2) [22](#_Toc17931_WPSOffice_Level2)

[3.2.1 方案二数据预处理、特征工程、建模](#_Toc22096_WPSOffice_Level3) [22](#_Toc22096_WPSOffice_Level3)

[3.1.2方案二结果、排名等](#_Toc1835_WPSOffice_Level3) [25](#_Toc1835_WPSOffice_Level3)

[3.1.3 方案二算法流程图](#_Toc22566_WPSOffice_Level3) [26](#_Toc22566_WPSOffice_Level3)

[3.3 方案三（简述）](#_Toc27185_WPSOffice_Level2) [27](#_Toc27185_WPSOffice_Level2)

[3.3.1 方案三数据预处理及特征工程部分方案](#_Toc12546_WPSOffice_Level3) [27](#_Toc12546_WPSOffice_Level3)

[3.3.2 方案三模型设计、建立部分方案](#_Toc8113_WPSOffice_Level3) [27](#_Toc8113_WPSOffice_Level3)

[3.3.3 方案三结果、排名等](#_Toc4559_WPSOffice_Level3) [28](#_Toc4559_WPSOffice_Level3)

[4. 算法比较 2](#_Toc27744_WPSOffice_Level1)9

[5. 总结与展望 3](#_Toc27744_WPSOffice_Level1)1

[5.1 总结](#_Toc22096_WPSOffice_Level2) [31](#_Toc22096_WPSOffice_Level2)

[5.2 建模思路](#_Toc1835_WPSOffice_Level2) [31](#_Toc1835_WPSOffice_Level2)

1. 背景描述

本次竞赛的重点是预测多个时间序列的未来价值问题，因为它一直是该领域最具挑战性的问题之一。

从生物数据，金融市场，天气预报到音频和视频处理等许多关键的现实问题都出现了连续或时间观测。时间序列领域包含许多不同的问题，从分析和推理到分类和预测。您可以做些什么来帮助预测未来的观点？

**1.1 竞赛赛题描述**

我们的目标是在测试参与者设计的最先进方法的竞争中，针对大约145,000条维基百科文章预测未来网络流量的问题。

本次比赛分为两个阶段，涉及对未来实际事件的预测。将有一个培训阶段，在此阶段，排行榜基于历史数据，然后是参与者对未来真实事件进行评分的阶段。

您可以完全自由地制作预测：例如，使用单变量与多变量模型，使用元数据（文章标识符），分层时间序列建模（针对不同类型的流量），数据增强（例如，使用Google趋势数据到扩展数据集），异常和异常值检测和清理，缺失值插补的不同策略，以及更多类型的方法。

**1.2 评估指标描述**

##### [SMAPE](https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error) （Symmetric mean absolute percentage error对称平均绝对百分误差）

[SMAPE](https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error)在预测值与实际值之间。当实际值和预测值都为0时，定义SMAPE = 0



1. 数据来源及描述性统计分析
   1. **大赛数据来源**

谷歌公司提供

下面是数据的超链接。

<https://www.kaggle.com/c/6768/download-all>

**2.2 数据的描述性统计**

**2.2.1 数据基本情况描述：**

训练数据集由大约145k个时间序列组成。从2015年7月1日到2016年12月31日，每个时间序列代表了不同维基百科文章的每日浏览量。培训阶段的排行榜以2017年1月1日至2017年3月1日的流量为基准。

第二阶段将使用截至2017年9月1日的培训数据。比赛的最终排名将基于数据集中每篇文章在2017年9月13日至2017年11月13日之间的每日浏览量预测。

对于每个时间序列，将向您提供文章的名称以及该时间序列表示的流量类型(all、mobile、desktop、spider)。您可以使用此元数据和任何其他公开可用的数据进行预测。不幸的是，此数据集的数据源无法区分零流量值和缺失流量值。缺失值可能意味着流量为零，或者当天的数据不可用。

为了减小提交文件的大小，每个页面和日期组合都有一个更短的Id。key文件中给出了页面名称和提交Id之间的映射。

**2.2.2 数据字段介绍：**

**表2-2 train\_1数据表字段介绍**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| Page | 不同维基百科文章 | 字符串 | 0% |
| 2015-07-01~  2016-12-31 | 每日浏览量 | 离散 | 14% |

**表2-2 train\_2数据表字段介绍**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| Page | 不同维基百科文章 | 字符串 | 0% |
| 2015-07-01~  2017-09-10 | 每日浏览量 | 离散 | 14% |

**表2-3 key\_1数据表字段介绍**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| Page | 不同维基百科文章 | 字符串 | 0% |
| 2015-07-01~  2016-12-31 | 每日浏览量 | 离散 | 14% |

**表2-4 key\_2数据表字段介绍**

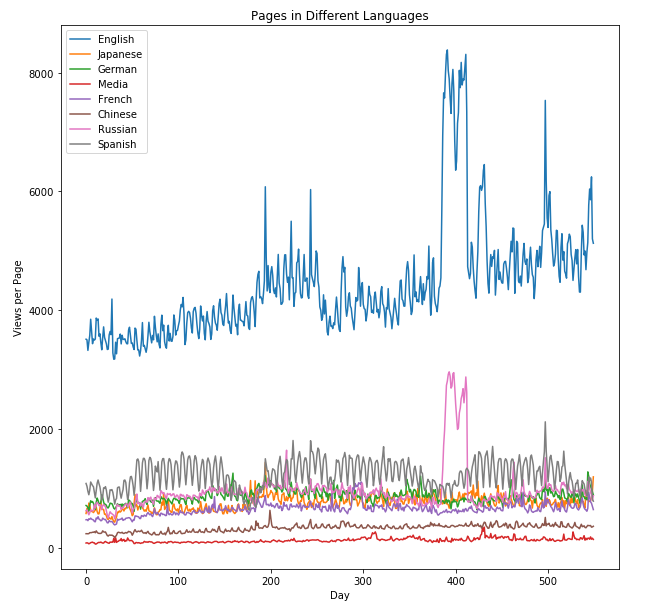
|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| Page | 不同维基百科文章 | 字符串 | 0% |
| 2015-07-01~  2016-12-31 | 每日浏览量 | 离散 | 14% |

**表2-5 train\_1数据表字段介绍**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 变量含义 | 变量类型 | 缺失率 |
| Page | 不同维基百科文章 | 字符串 | 0% |
| 2015-07-01~  2016-12-31 | 每日浏览量 | 离散 | 14% |

**2.2.3 数据描述性统计**

**流量是否受到页面语言的影响?**



由于维基百科是一家总部位于美国的网站，因此可以预料到，英文网页的浏览量要高得多。

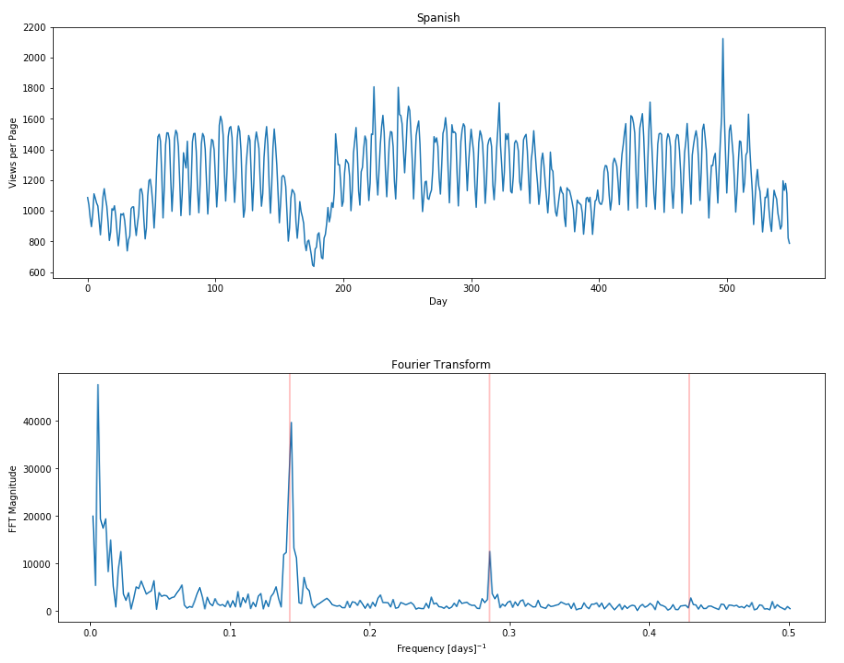
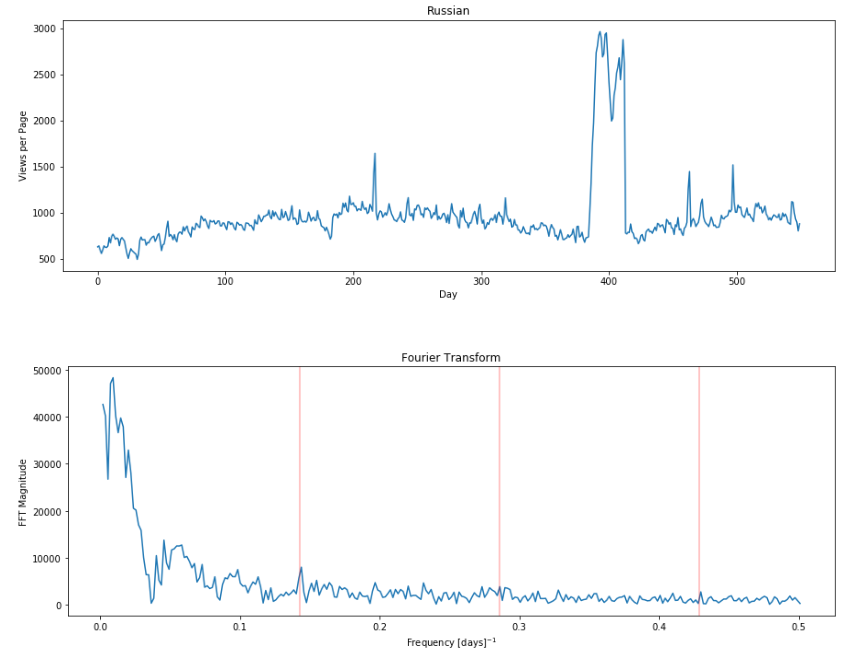
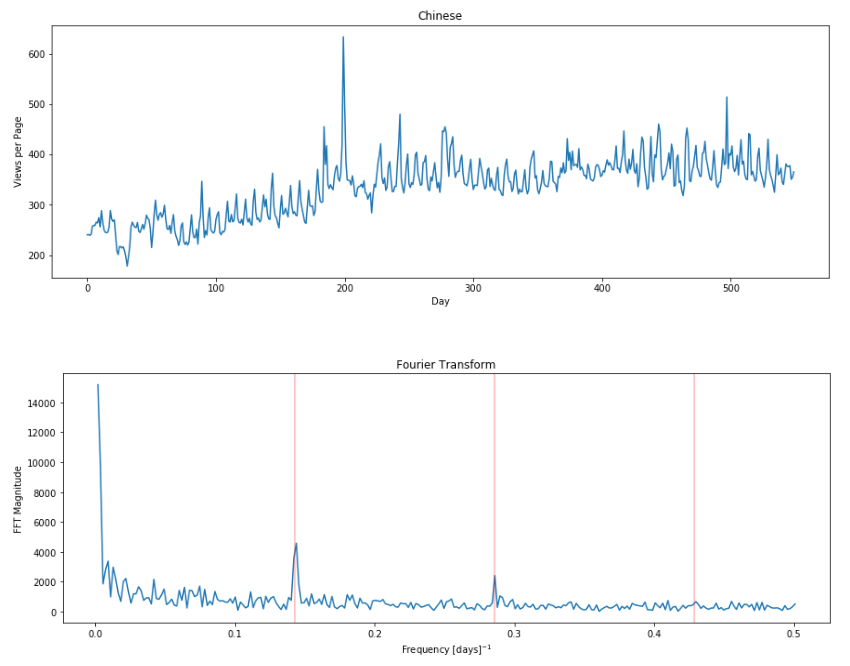
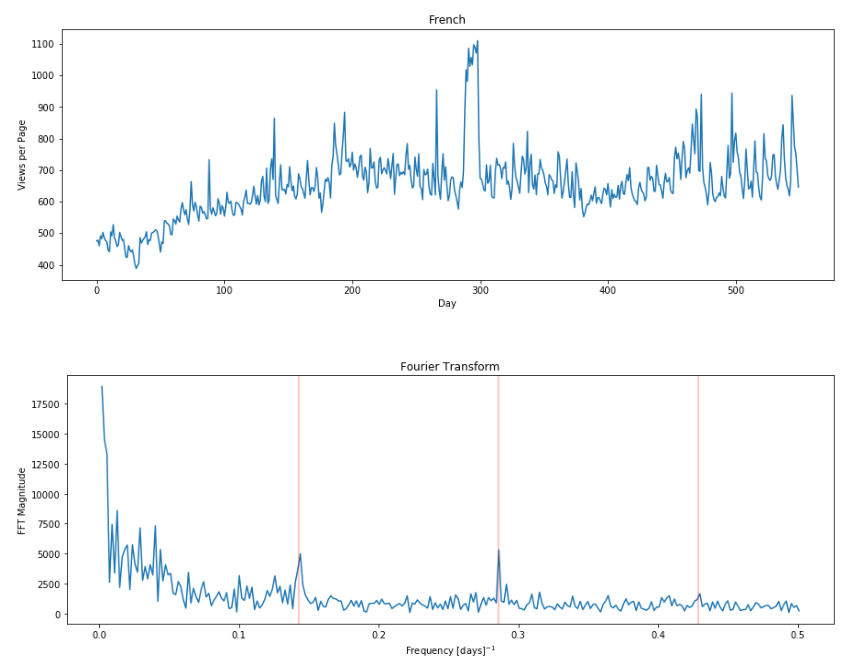
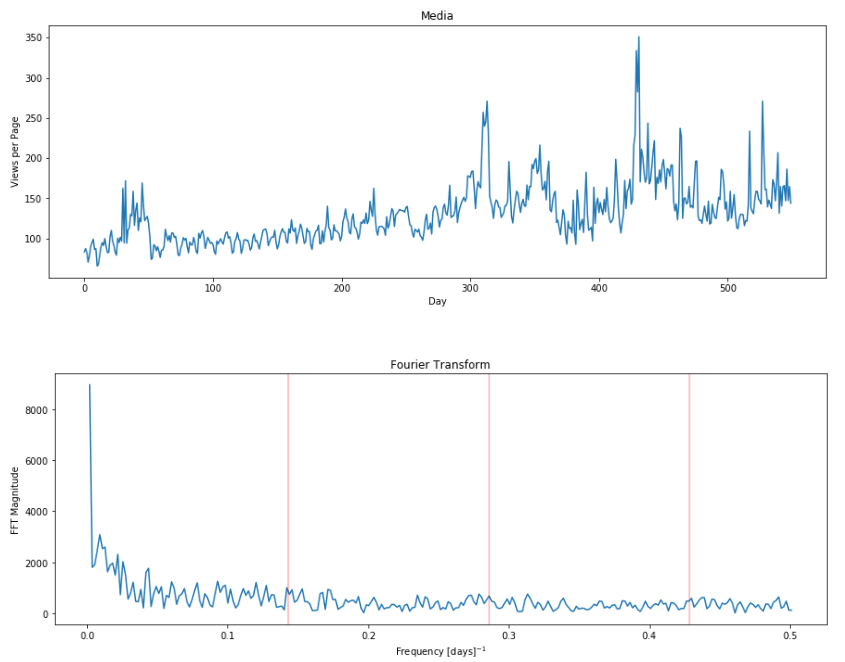
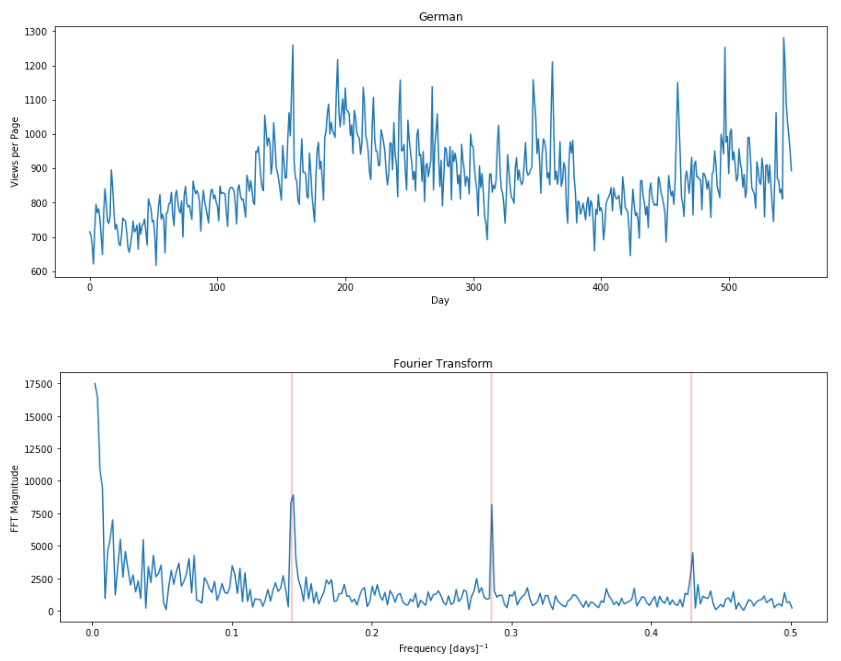
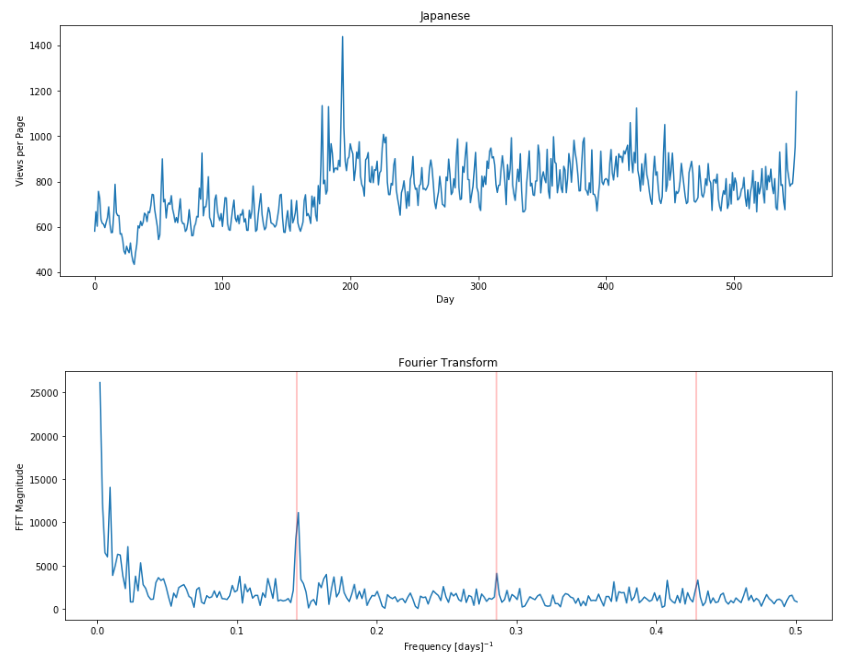
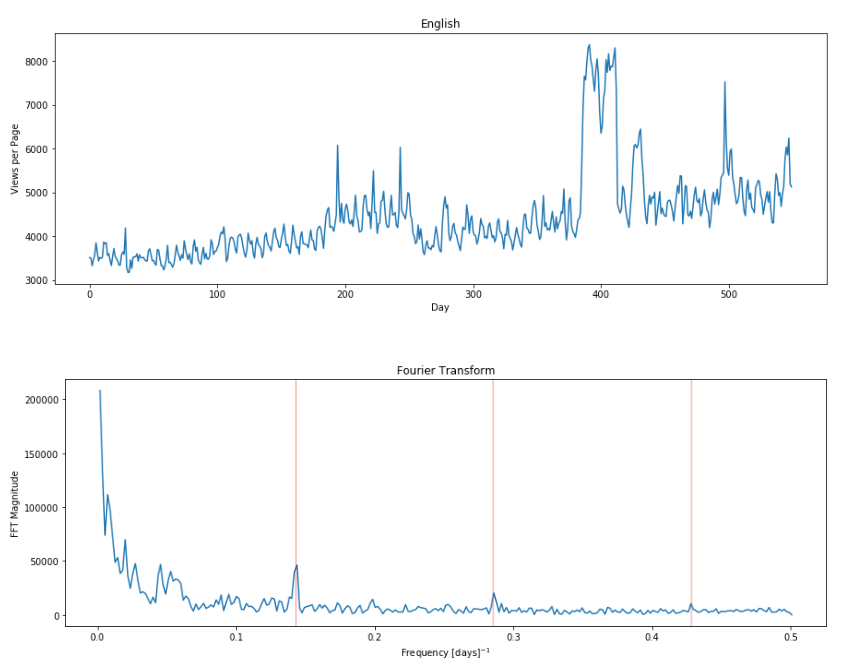
英语和俄语的图表显示，在第400天左右(2016年8月左右)出现了非常大的峰值，而英语数据在2016年晚些时候出现了几个峰值。我的猜测是，这既是8月份夏季奥运会的影响，也是美国大选的影响。

在200天左右的英语数据中也有一个奇怪的特征。

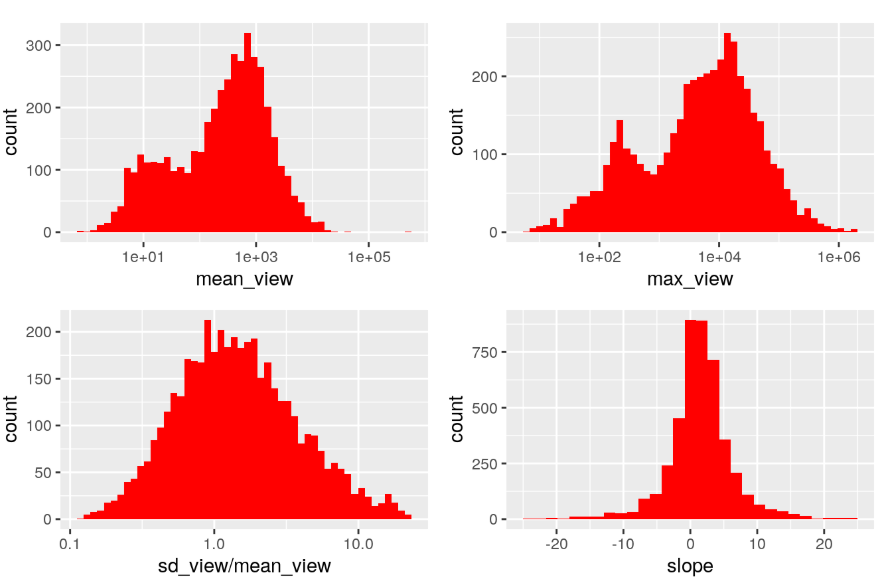
西班牙的数据也很有趣。这里有一个明显的周期结构，有一个大约1周的快速周期，似乎每6个月左右就有一个明显的下降。

**周期结构和FFTs**

因为看起来这里有一些周期结构，将把它们分别画出来，这样规模就更明显。除了单独的图，还将研究快速傅里叶变换(FFT)的大小。FFT的峰值显示了周期信号中最强的频率。



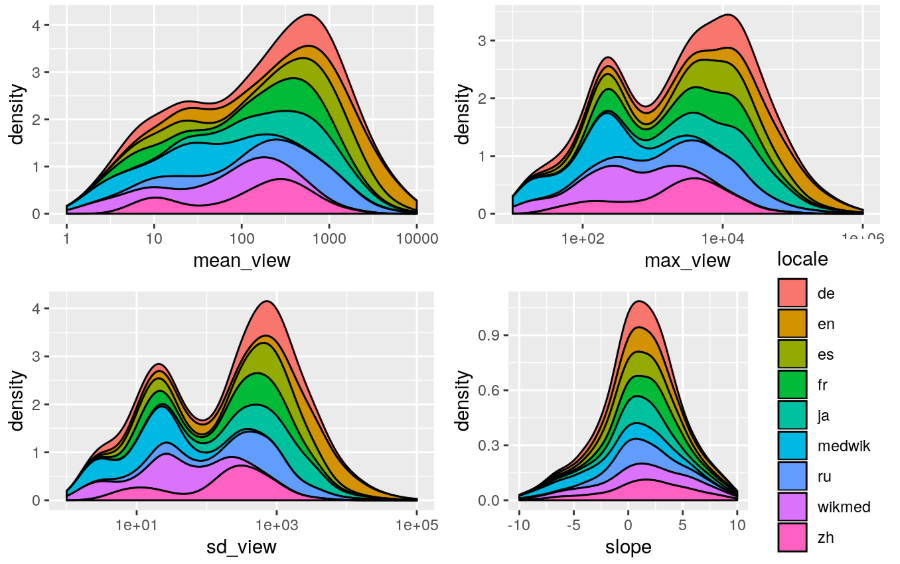
让我们来研究一下我们构建的参数空间。(分布的全局形状不应受抽样的影响。)首先，我们绘制主要参数的直方图:



我们发现:

1. 平均浏览量呈明显的双峰分布，峰值在10 - 200-300之间。类似的情况也适用于最大的观看次数，尽管这里的第一个峰值(大约200次)非常窄。第二高峰的中心在一万米以上。
2. 标准差的分布(除以平均值)倾向于更高的值，峰值数量更多或变异性趋势更强。这些将是更具挑战性的观测结果。
3. 边坡的分布是合理对称的，中心明显高于零。

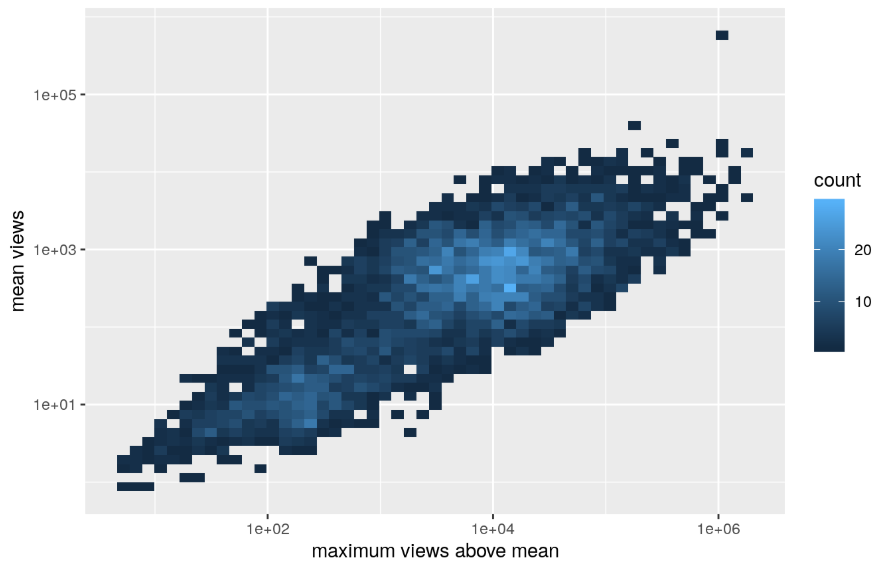
我们把它按地区分开，重点放在密度上：



我们发现:

1. 中文页面(zh，粉红色)略有不同，但明显不同于其他页面。平均和最大视图更低，也更少的变化。与其它曲线相比，它们的斜率分布更广，但也更倾向于正值。
2. 最大浏览量在200-300左右的峰值在法语页面上最明显(fr，用蓝绿色表示)。
3. 英文页面(en，芥末色)有最高的平均值和最大的浏览量，这并不奇怪。

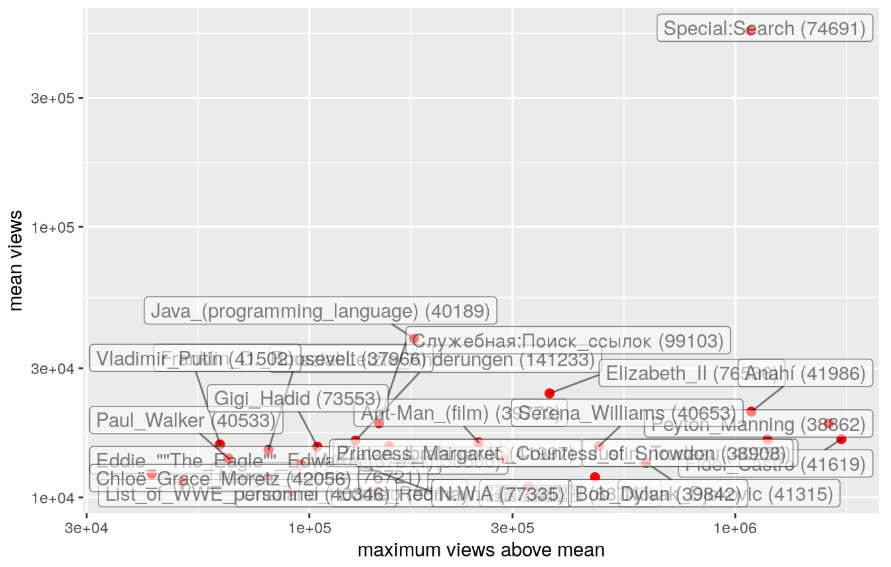
接下来，我们将研究binned二维直方图。



我们发现:

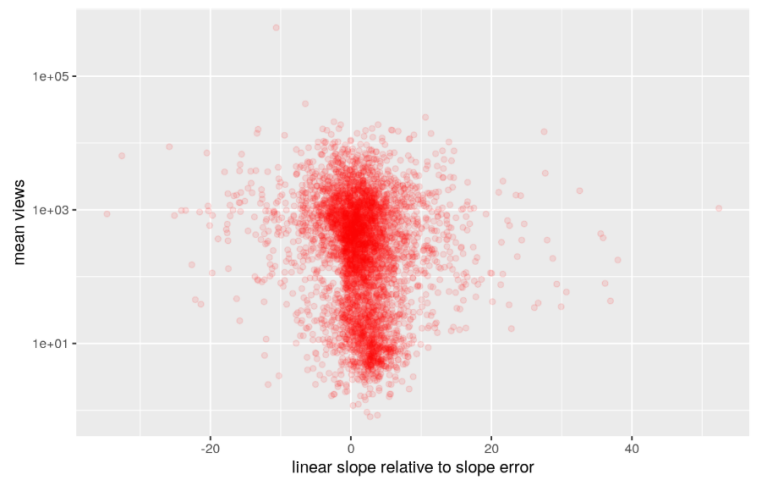
平均视图和最大视图之间存在明显的相关性。同样在这里，我们又发现了我们在单个直方图中识别出的两个簇峰。一些异常值和异常群是显而易见的。

让我们放大到右上角(括号中的数字是行号):



在这里，我们找到了一些主页和其他元页面(在完整的数据集中)。

另一个问题:视图中的(假设的)线性变化是否取决于视图的总数?



我们发现，具有较高平均浏览量的文章在其线性趋势中有更多的可变性。然而，这可能是由于我们的坡度正常化，这将减少有效坡度为低视图计数。但是，这不应影响到观测结果，即低视图物品的坡度平均略高于高视图物品的坡度。当然，这种效应可能是由观察峰值引起的，但我认为这些峰值是随机分布的。

1. 优秀算法思路

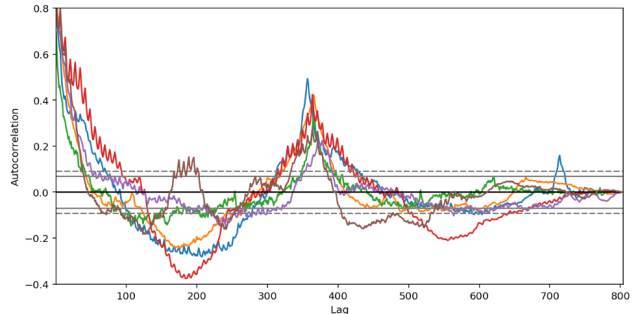
**3.1 方案一**

代码：<https://github.com/Arturus/kaggle-web-traffic>

下面将简要介绍 Artur Suilin 如何修正 GRU 以完成网站流量时序预测竞赛。

预测有两个主要的信息源：

1. 局部特征。我们看到一个趋势时，希望它会继续（自回归模型）朝这个趋势发展；看到流量峰值时，知道它将逐渐衰减（滑动平均模型）；看到假期交通流量增加，就知道以后的假期也会出现流量增加（季节模型）。
2. 全局特征。如果我们查看自相关（autocorrelation）函数图，就会注意到年与年之间强大的自相关和季节间的自相关。



**3.1.1** 方案一数据预处理及特征工程部分方案

**特征工程：**

RNN 足够强大来发现和学习自身特征。模型的特征列表如下：

1.pageviews：原始值经过 log1p() 的转换得到几乎正态的时序内值分布，而不是偏态分布。

2.agent, country, site：这些特征从网页URL中提取，然后经过 One-Hot 编码。

3.day of week：捕捉每周的季节效应。

4.year-to-year autocorrelation, quarter-to-quarter autocorrelation：捕捉各年、各季度的季节效应。

5.page popularity：高流量和低流量页面具有不同的流量变化模式，该特征（pageviews 的中间值）帮助捕捉流量规模。pageviews 特征丢失了规模信息，因为每个 pageviews 序列被独立归一化至零均值和单位方差。

6.lagged pageviews：之后将具体介绍。

**特征预处理：**

1.所有特征（包括 One-Hot 编码的特征）被归一化至零均值和单位方差。每个 pageviews 序列被独立归一化。

2.时间依赖特征（自相关性、国家等）被「拉伸」至时序长度，即每天重复使用 tf.tile() 命令。

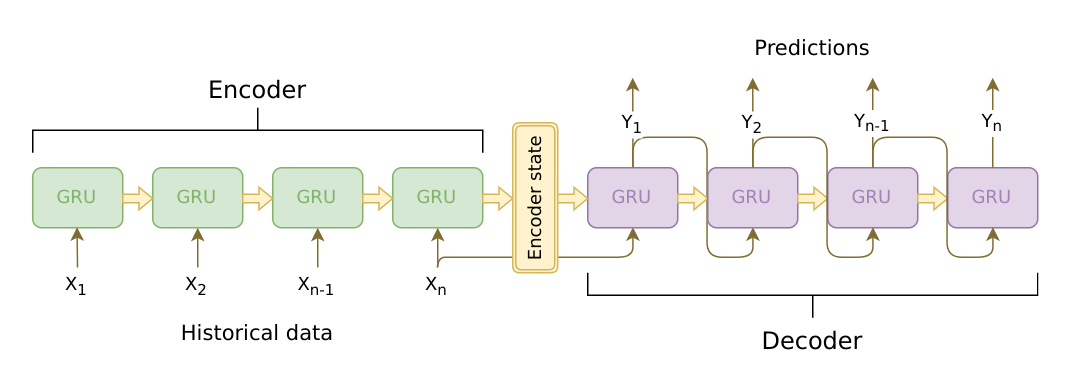
3.模型在来自初始时序的随机固定长度样本上进行训练。例如，如果初始时序长度是 600 天，我们使用 200 天的样本进行训练，那么我们可以在前 400 天中随意选择开始采样的样本。

4.采样工作是一种有效的数据增强机制：训练代码在每一步随机选择每次时序的开始点，生成无限量的几乎不重复的数据。

**3.1.2** 方案一模型设计、建立部分方案

## 模型核心

模型有两个主要部分：编码器和解码器。



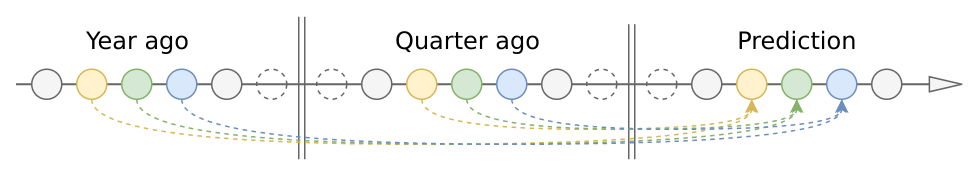
## 编码器为 cuDNN GRU，cuDNN 要比 TensorFlow 的 RNNCells 快大约 5 到 10 倍，但代价就是使用起来不太方便，且文档也不够完善。

## 解码器为 TF GRUBlockCell，该 API 封装在 tf.while\_loop() 中。循环体内的代码从上一步获得预测，并加入到当前时间步的输入特征中。

**处理长时间序列**

LSTM/GRU 对于相对较短的序列（100-300 项以内）来说是非常好的解决方案。但对于较长的序列来说，LSTM/GRU 仍然有效，只不过会逐渐遗忘较早时间步所包含的信息。Kaggle 竞赛的时间序列长达 700 多天，所以我们需要找一些方法来「加强」GRU 的记忆力

我们第一个方法先是考虑使用一些注意力机制。注意力机制可以将过去较长距离的有用信息保留到当前 RNN 单元中。对于我们的问题，最简单高效的注意力方法是使用固定权重的滑动窗口注意力机制。它在较长距离的过去时间步上有两个重要的点（考虑长期的季节性），即 1 年前和 1 个季度前。



我们可以采用 current\_day - 365 和 current\_day - 90 这两个时间点的编码器输出，并将它们馈送到全连接层以降低维度，并将结果加入到解码器的输入特征中。这个解决方案虽然简单却大大降低了预测误差。

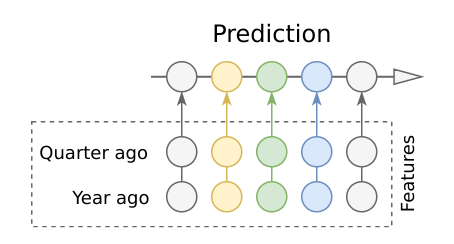
随后我们将重要的点与它们的近邻求均值，并借此减少噪声和补偿不均匀的间隔（闰年和不同长度的月份）：attn\_365 = 0.25 \* day\_364 + 0.5 \* day\_365 + 0.25 \* day\_366。

但随后我们意识到 0.25、0.5、0.25 是一个一维卷积核（length=3），我们可以自动学习更大的卷积核以检测过去重要的点。

最后，我们构建了一个非常大的注意力机制，它会查看每一个时间序列的「指纹」（指纹由较小的卷积网络产生），并决定应该注意哪些点和为较大卷积核生成权重。这个应用于解码器输出的较大卷积核会为每一个预测的日期生成一个注意力特征。虽然最后没有使用这种方法，但这个注意力机制仍然保留在代码中，读者可以在模型代码中找到它。

注意，我们并没有使用经典的注意力方案（Bahdanau 或 Luong 注意力机制），因为经典的注意力机制应该在每个预测步上使用所有的历史数据点从头开始计算，因此这种方法对于较长时间序列（约两年的天数）来说太耗时间了。所以我们的方案将会对所有数据点进行一次卷积，对所有预测时间步使用相同的注意力权重（这也是缺点），这样的方案计算起来要快很多。

因为我们对注意力机制的复杂度感到不太满意，因此我们试图完全移除注意力机制，并将一年前、半年前、一季度前重要的数据点作为编码器和解码器的附加特征。这样的结果是非常令人惊讶的，甚至在预测质量方面都要比带注意力机制的模型略胜一筹。因此我们最好的公开分数都是仅使用滞后（lagged）数据点实现的，它们都没有使用注意力机制。

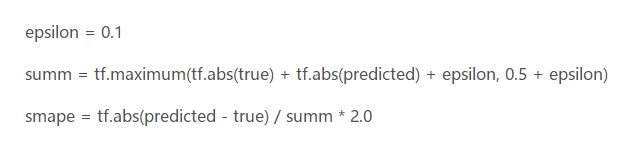


## 滞后数据点另一个重要的优势是，模型可以使用更短的编码器而不需要担心损失过去的信息，因为这些信息现在明确地包含在特征中。在采用这种方法后，即使我们编码器的长度是 60 到 90 天，结果也是完全可以接受的，而以前需要 300-400 天的长度才能获得相同的性能。此外，更短的编码器就等于更快速的训练和更少的信息损失。

**损失和正则化**

SMAPE（竞赛用的目标损失函数）因其在零值周围不稳定的行为而无法直接使用（当真值为零的时候，损失函数是阶跃函数；预测值也为零的时候，则损失函数不确定）。

我使用经平滑处理的可微 SMAPE 变体，它在所有实数上都表现良好：



另一个选择是在 log1p(data) 上的 MAE 损失函数，它很平滑，且训练目标与 SMAPE 非常接近。

最终预测取最接近的整数，负面预测取零。

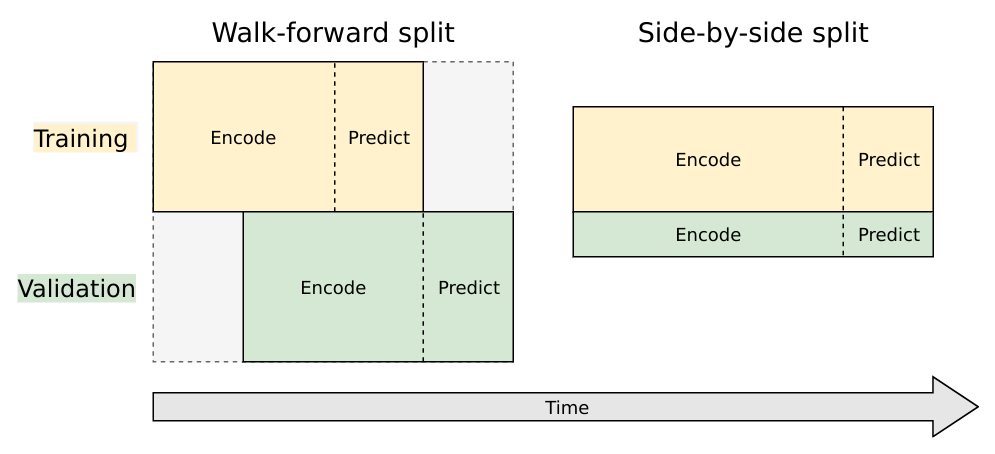
我尝试使用论文《Regularizing RNNs by Stabilizing Activations》中经正则化的 RNN 激活值，因为 cuDNN GRU 的内部权重无法直接正则化（也可能是我没有找到正确的方法）。稳性损失（Stability loss）不起作用，激活损失可以为较小损失权重（1e-06..1e-05）带来少许改进。

**训练和验证**

我使用 COCOB 优化器（详见论文《Training Deep Networks without Learning Rates Through Coin Betting》）结合梯度截断进行训练。COCOB 尝试预测每个训练步的最优学习率，因此我完全不必调整学习率。它的收敛速度也比传统的基于动量的优化器快得多，尤其是在第一个 epoch 上，可以让我及早停止不成功的实验。

有两种方式可以将时序分割为训练和验证数据集：

1. Walk-forward 分割。这实际上不是分割：我们在完整数据集上训练和验证，但使用不同的时间跨度。验证用的时间跨度比训练用时间跨度前移一个预测间隔。
2. Side-by-side 分割。这是主流机器学习传统的分割模型。数据集被分割成两个独立的部分，一个用于训练，另一个用于验证。



## 两种方式我都试了，但对于这个任务来说 Walk-forward 更好，因为它与竞赛目标直接相关：使用历史值预测未来值。但是该分割破坏了时序结尾的数据点，使得训练准确预测未来的模型变得困难。

## 具体来说：比如，我们有 300 天的历史数据，想预测接下来 100 天的数据。如果我们选择 walk-forward 分割，我们必须使用前 100 天的数据用于真实训练，后面 100 天的数据用于训练模式的预测（运行解码器、计算损失），再后面 100 天的数据用于验证，最后 100 天用于对未来值真正进行预测。因此，我们实际上可以使用 1/3 的数据点来训练，最后一个训练数据点和第一个预测数据点之间隔了 200 天。间隔太大了，因为一旦我们离开某个训练数据，预测质量将出现指数级下降（不确定性增加）。使用 100 天差距训练的模型预测质量相对较好。

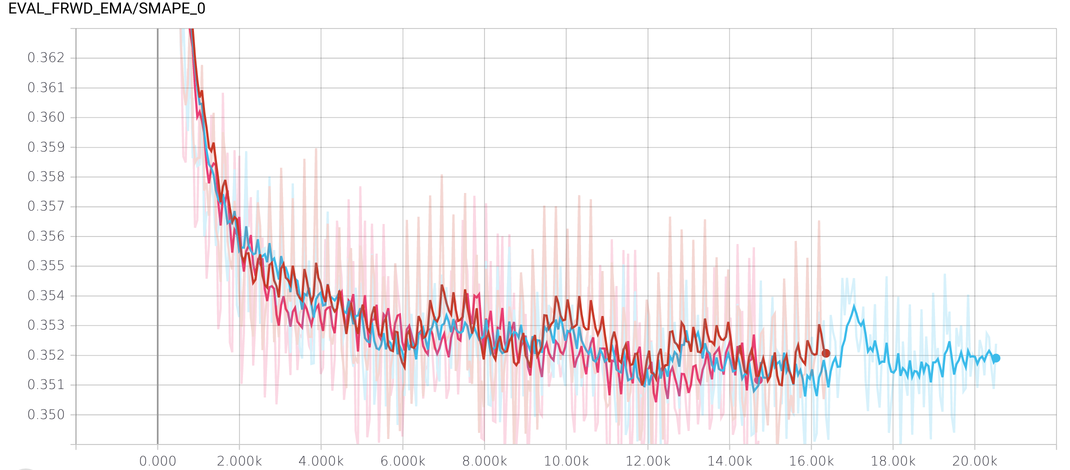
## Side-by-side 分割所需要的计算力更少，因为它在端点时并不会消耗数据点。但是对于我们的数据，模型在验证集上的性能与在训练集上的性能是强相关的，并且与将来的实际模型性能几乎不具有相关性。换而言之，并行分割对于我们的问题基本上是没有什么作用的，它只是复制了在训练数据集上观察到的模型损失。

## 我仅使用验证集（带有前向分步分割）进行模型调优，预测未来数值的最终模型只是在盲目的模式中进行训练，没有使用任何验证集。

## 减少模型方差

优于强噪音数据的输入，模型不可避免地具有高方差。坦白讲，我很惊讶 RNN 居然从噪音数据中学习到了东西。

在不同 seed 上训练的相同模型具有不同的表现，有时模型甚至在「不幸」的 seed 上变得发散。训练期间，表现也会逐步地发生很大波动。依靠纯粹的运气很难赢得比赛，因此我决定采取行动降低方差。



1. 我不知道哪个训练步骤最适合预测未来（但前数据的验证结果与未来数据的结果只有弱相关关系），所以我不能使用提前停止。但是我知道近似区域，其中模型（可能）进行了充分训练，但（可能）没有开始过拟合。我决定把这个最佳区域设置为 10500 到 11500 次迭代区间内，并且从这个区域的每第 10000 个步骤保存 10 个检查点。
2. 相似地，我决定在不同 seed 上训练 3 个模型，并从每个模型中保存检查点。因此我一共有 30 个检查点。
3. 降低方差、提升模型性能的一个众所周知的方法是 ASGD（SGD 平均）。它很简单，并在 TensorFlow 中得到很好的支持。我们必须在训练期间保持网络权重的移动平均值，并在推断中使用这些平均权重，而不是原来的权重。

三个模型的结合表现不错（在每个检查点上使用平均模型权重的 30 个检查点的平均预测）。我在排行榜上（针对未来数据）获得了相较于历史数据上的验证大致相同的 SMAPE 误差

理论上讲，你也可以把前两种方法用作集成学习，但我主要用其降低方差。

**超参数调节**

很多模型参数（层的数量、深度，激活函数，dropout 系数等）能够（并且应该）被调节从而获得更优的模型表现。手动调节乏味且费时，所以我决定自动化该过程，并使用 SMAC3 搜索超参数。下面是 SMAC3 的一些优势：

1. 支持条件参数（例如，为每层联合调节层数和 dropout；如果 n\_layers > 1，第二层上的 dropout 将被调节）
2. 明确处理模型方差。SMAC 在不同种子上训练每个模型的若干个实例，如果实例在相同种子上训练还要对比模型。如果它在所有相同种子上优于另一个模型，则该模型获胜。

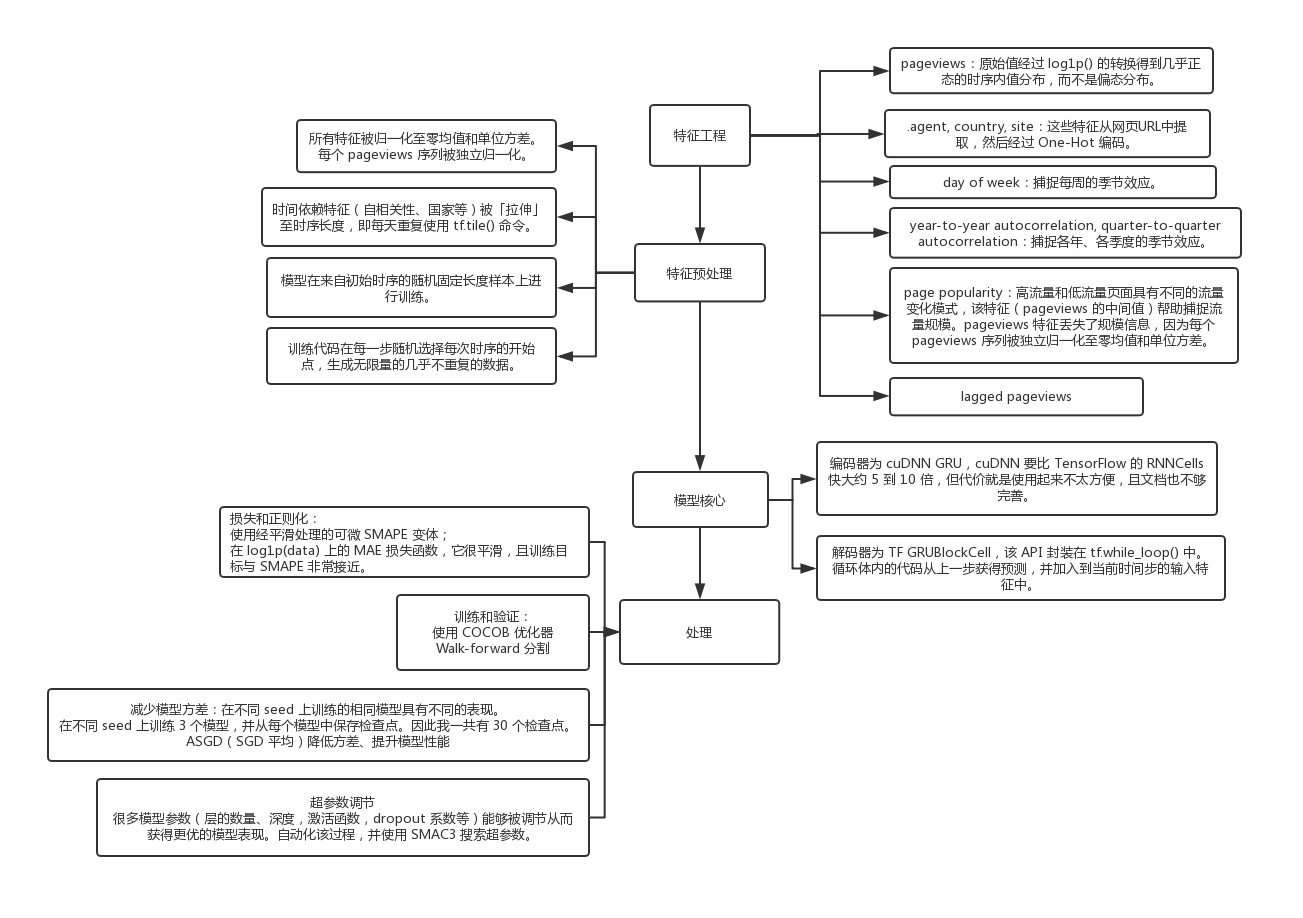
与我的期望相反，超参数搜索并没有建立定义明确的全局最小。所有的最佳模型大致具有相同的性能，但参数不同。可能 RNN 模型对于这个任务来说太具有表现力了，并且最好的模型得分更多依赖于模型架构上的数据信噪比。不管怎样，最好的参数设置依然可以在 hparams.py 文件中找到。

**3.1.3** 方案一结果、排名等

结果：35.48065

排名：1

**3.1.4** 方案一算法流程图

****

**图3-1**

**3.2 方案二**

代码：<https://github.com/jfpuget/Kaggle/tree/master/WebTrafficPrediction>

**3.2.1** 方案二数据预处理、特征工程、建模

基于5个想法。

1. **使用数据的年度季节性。**

有一个每周的季节性，对于一些项目来说非常强大，例如es.wikipedia.com，并且几乎不存在于其他一些项目。但更重要的是，每年都有巨大的季节性。

因此，我决定使用cv训练我的模型来自12个月前的数据。对于第1阶段，我在2016年1月1日至2016年3月1日期间作为测试集进行了培训，构建了2015年7月至12月期间的视图。然后我根据2016年7月至12月的相同特征预测了价值。对于第2阶段，我使用2016年9月13日的62天进行了培训，并且相对于2016年9月10日的数据构建了相同的功能。然后预测使用相同的特征构建截至2017年9月10日的数据。为了消除年度趋势并保持季节性，我减去了训练集和测试集的最近20周列车数据的中位数。一旦中位数被减去，价值范围每年都是可比的。这涉及XGBoost无法真正推断的事实（参见[https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/38352）](https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/38352)" \t "https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/_blank)。

我使用5倍CV，其中数据按页面分割：给定页面的所有数据必须在同一折叠中，否则模型会过度拟合。我根据提交的预测采用了每个折叠训练模型的中位数。预测值四舍五入为最接近的整数，负值设置为0。

这第一个想法真的是一个赌注，我绝对不知道它在第二阶段的效果如何。

1. **不使用RMSE，使用log1p转换数据来近似SMAPE。**

SMAPE是关于忽略异常值。使用所有视图数据的log1p变换是忽略异常值的第一步。第二步是为我使用的三种算法中的每一种使用正确的目标函数：

广义线性模型。我使用了sklearn Huber回归量。这是一个线性回归，其中rmse（如在OLS中）被Huber损失取代。这对异常值的敏感度低于OLS。使用5倍CV，在第1阶段公共LB中得到43.9分。

XGBoost。为MAE定义了一个自定义目标函数（后来由Arthur Suilin提出）。基本上，如果预测高于真值，则梯度为正，如果预测低于真值，则梯度为负，如果预测低于真值，则为0。我使用了具有足够高值的常量hessian（真正的hessian在任何地方都是null，除非预测等于真值）。代码很简单（因为我正在从内存中重新输入，可能会出现拼写错误）

from numba import jit

@jit

def grad(preds, dtrain):

labels = dtrain.get\_labels()

n = preds.shape[0]

grad = np.empty(n)

hess = 500 \* np.ones(n)

for i in range(n):

diff = preds[i] - labels[i]

if diff > 0:

grad[i] = 200

elif diff < 0:

grad[i] = -200

else:

grad[i] = 0

return grad, hess

我后来通过预先分配数组来优化它。通过cv选择常数200，学习率为0.1。较低的值导致收敛非常缓慢。较高的值导致CV评分较差。XGBoost设置是保守的，最*小子*权重设置为400.通过CV选择400。

我用它获得了第5阶段公共LB 42.6，使用5倍CV。由于自定义目标函数（以及自定义评估函数），此模型的主要问题是其训练时间为12小时。内置目标的训练速度提高了10倍，但结果却更差。我后来发现，通过对1/10的数据进行训练，我可以获得相同的准确度。这是满足第2阶段截止日期的关键！

神经网络。我使用了在Keras中以这种方式定义的剪辑版MAE：

将keras.backend导入为K.

def smape *error（y* true，y *pred）：返回K.mean（K.cplip（K.abs（y* pred - y\_true），0.0,1.0），轴= -1）

通过CV选择剪切阈值。我用它获得了第4阶段公共LB 41.8，使用5倍CV。NN在下面更详细地描述。

1. **摆脱异常值。**

我找到的最好的方法是使用我最好的单个模型提交并逐行计算它的smape，并丢弃smape高于1.25的行。通过CV选择1.25阈值。这使我的XGBoost模型公共LB得分从42.6提高到42.2。

1. **通过训练Keras预测的残差和与我的XGBoost模型相同的特征加上来自Huber回归量和Keras模型的折叠预测来整合xgboost中的所有内容。**

我使用了自定义目标的略微修改版本，其中渐变与误差的平方成反比。这是SMAPE的实际梯度。这使得比分仅从41.8降至41.4。

1. **是使用中位作为特征而不是原始值（受到非常有效的公共内核的启发）**

在过去8到12周内，特征主要是每周中位数，具体取决于模型，加上较大时期的中位数。稍后加入中位数延迟一周，中位数差异，中位数最大值。

对于Huber回归量和XGBoost，其余特征是：星期几（星期一= 0到星期日= 6），捕获每周季节性，测试集中的星期排名，捕获每周演变和项目（最高网址）。使用访问和代理对XGBoost没有帮助，但它确实对NN有所帮助。几乎没有其他功能，但它们的影响非常小。无论如何，我在比赛的最后几天删除了大部分。

对于NN，我没有明确使用星期几和星期排名。我宁愿用它们来定义输出指数：无论我在哪一年，第三个星期二的输出都处于相同的位置。这是良好预测的关键，因为它可以预测每周模式。项目和访问/代理是一个热编码，中位数是log1p转换数据。我没有对数据进行标准化，因为平均值已接近0，方差接近1。

NN是前馈网络，每层有200,200,100,200个小区。输入再次与第一层的输出连接。我不知道为什么，但这提高了准确性。激活是relu，除了最后一个是线性的。几乎所有层都使用0.5的丢失（通过CV选择0.5）。我还对中间层使用了批量标准化。使用adam优化器和上面定义的损失函数编译模型。

我使用5倍Cv用于NN批量大小很大：4096.我发现增加批量大小既减少了培训时间，又提高了准确性。训练进行了200个时期，在所有折叠中每10个时期计算实际训练SMAPE，并使用所有预测的最佳时期的中位数进行提交。对于每个折叠，我训练了1个模型，并取了他们预测的中位数。由于批量大，每个型号在我的GTX 1080 Ti上需要几秒钟。

我尝试了CNN和RNN（LSTM和Seq2Seq），但没有得到与简单FF网络一样好的结果。毫不奇怪，因为这是我第一次使用深度学习。我很想向那些成功使用CNN或RNN的人学习。

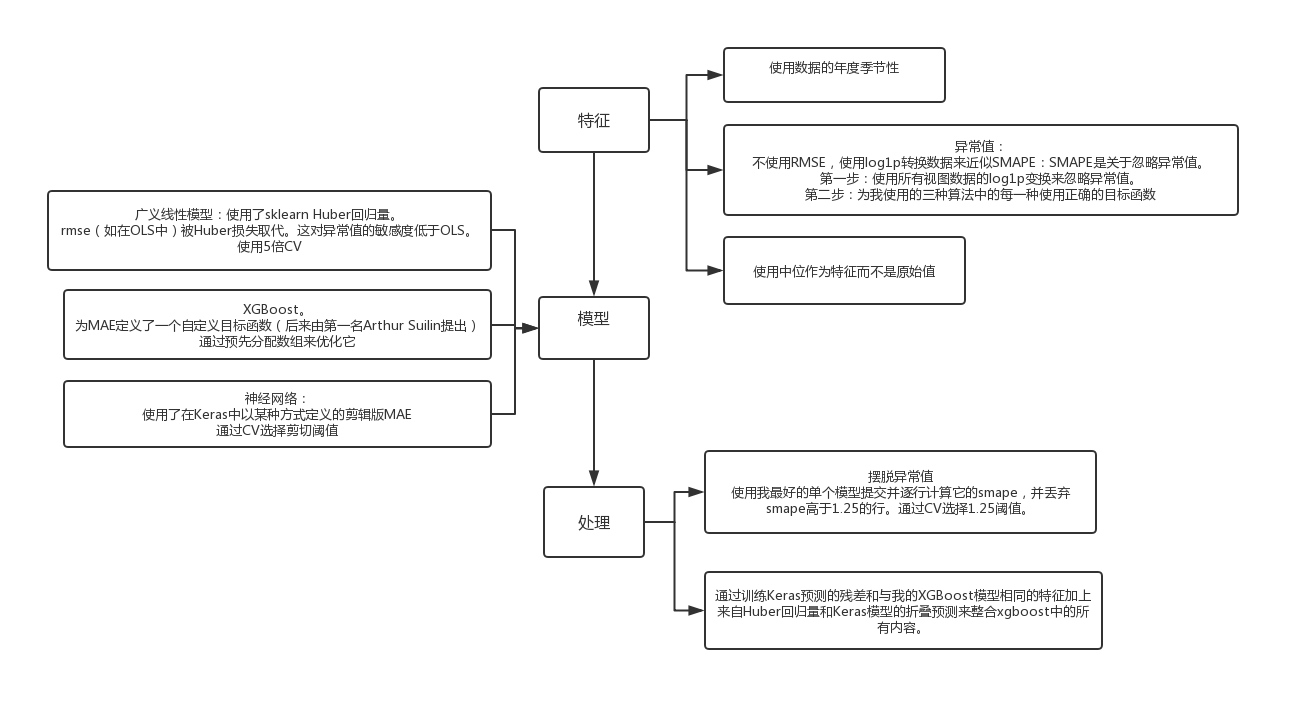
我尝试的另一件事是合奏XGBoost模型。它产生了0.3的改进，但由于数据可用性和截止日期之间缺乏时间，我无法在第2阶段提交时运行它。

**3.1.2**方案二结果、排名等

结果：36.78499

排名：2

**3.1.3** 方案二算法流程图



**3.3 方案三（简述）**

能够实现一个能够产生合理预测的卷积模型。不幸的是，我没有时间妥善处理nans，所以我在以nan结尾的页面上的所有预测都很不好。大多数其他预测在大多数情况下看起来都很好。总的来说，我认为这是解决这个问题的一种非常有前景的方法。

代码：<https://github.com/sjvasquez/web-traffic-forecasting>

**3.3.1** 方案三数据预处理及特征工程部分方案

未描述

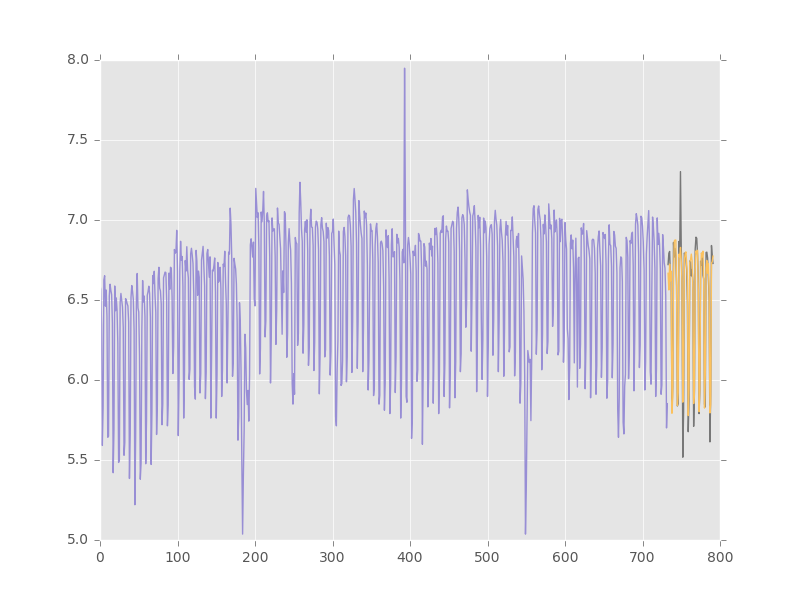
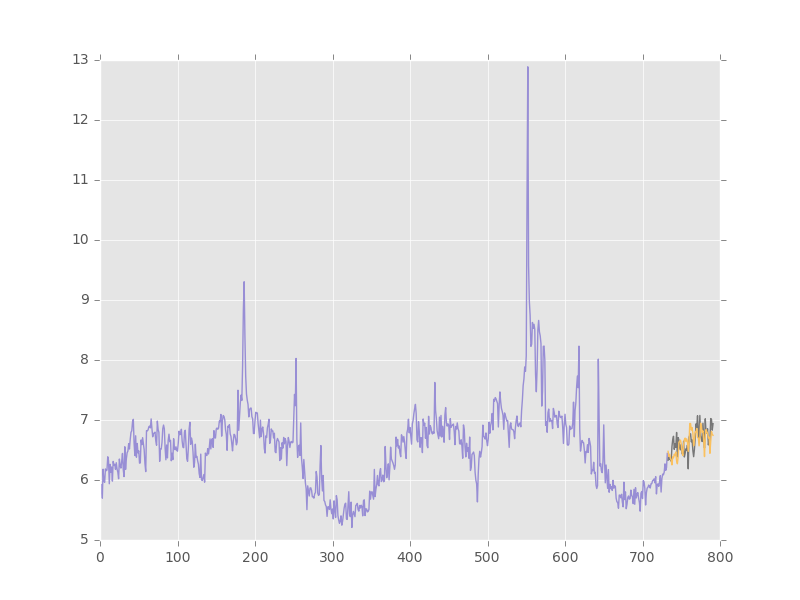
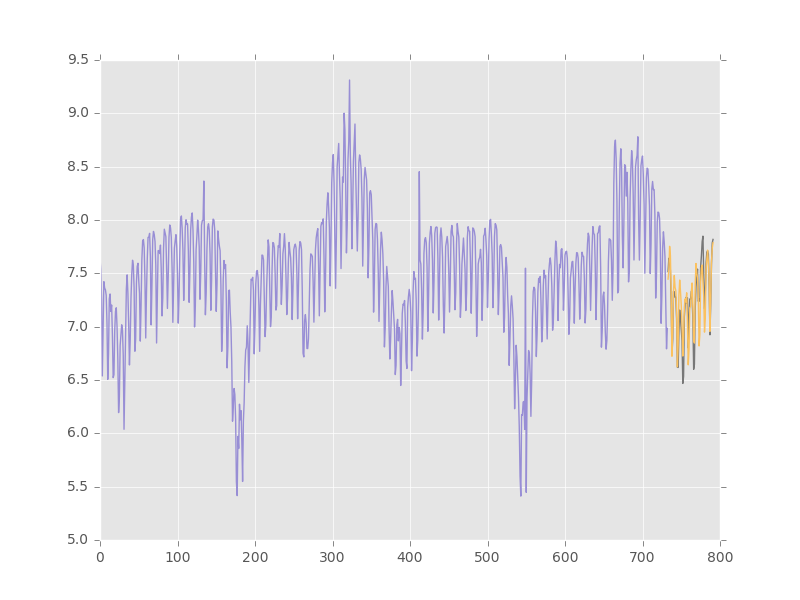
**3.3.2** 方案三模型设计、建立部分方案

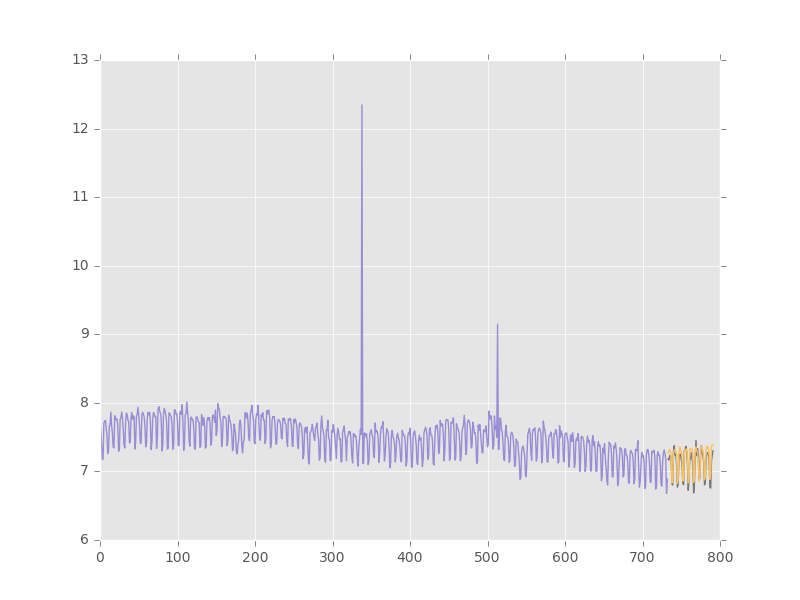
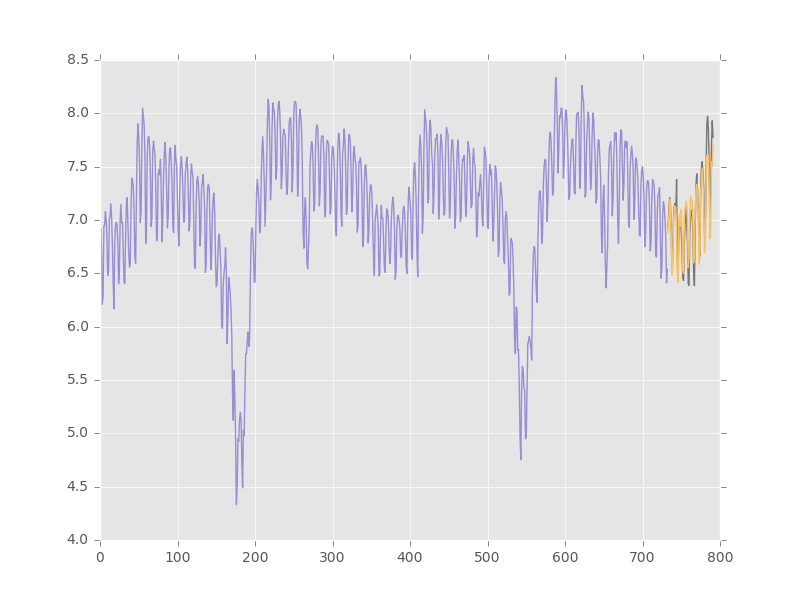
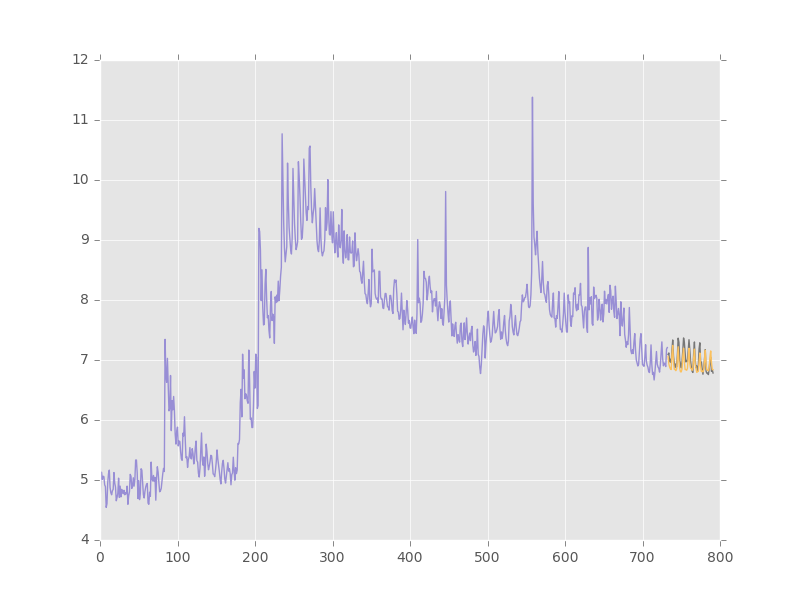
使用单个神经网络对所有145k时间序列进行建模。模型架构类似于WaveNet，由一堆扩张的因果卷积组成，如下[图](https://deepmind.com/blog/wavenet-generative-model-raw-audio/)所示（可双击查看动态图）。



进行了一些修改以调整模型以生成整个预测范围（64天）的连贯预测。使用下一步预测训练WaveNet，因此在没有条件信息的情况下模型生成长序列时误差会累积。为了解决这个问题，我们对模型进行了训练，以最大限度地减少64步拆开时的损失。我们采用序列到序列方法。这允许解码器在生成长序列时处理累积噪声。

下面是一些示例预测，用于演示网络可以捕获的一些模式。预测值为黄色，地面实况值（未在训练或验证中使用）以灰色显示。对y轴进行对数变换。

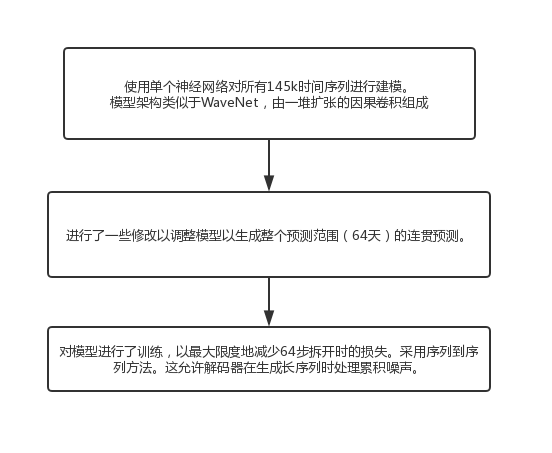




**3.3.3** 方案三结果、排名等

结果：37.42000

排名：6

**3.3.4** 方案三算法流程图

4.算法比较

描述几种算法的基本情况及效果对比，以表格形式：

例如： 特征工程和建模部分可以描述建模用到的基础算法名称（PCA、LSTM、时间衍生等）基本库指用到的基本库（sklearn等）

**表4-1 算法比较**

|  | **评估指标**  **（**[SMAPE](https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error)**）** | **特征工程** | **基础算法** | **基本库** |
| --- | --- | --- | --- | --- |
| **算法1** | 35.48065 | 转换成正态的时序内值分布；  one-hot编码；季节效应；  中间值；滞后特征 | CuDNN GRU | Pandas  Numpy  sklearn |
| **算法2** | 36.78499 | 年度季节性；异常值 | 广义线性模型；XGBoost；  神经网络 | Pandas  Numpy  sklearn |
| **算法3** | 37.42000 | 未描述 | 网络 | Pandas  Numpy  sklearn |

**算法一：GRU**

**作者自述：**

我决定使用 RNN seq2seq 模型进行预测，原因如下：

1. RNN 可以作为 ARIMA 模型的自然扩展，但比 ARIMA 更灵活，更具表达性。
2. RNN 是非参数的，大大简化了学习。想象一下对 145K 时序使用不同的 ARIMA 参数。
3. 任何外源性的特征（数值或类别、时间依赖或序列依赖）都可以轻松注入该模型。
4. seq2seq 天然适合该任务：我们根据先前值（包括先前预测）的联合概率

（joint probability）预测下一个值。使用先前预测可保持模型稳定，因为误差会在每一步累积，如果某一步出现极端预测，则有可能毁了所有后续步的预测质量。

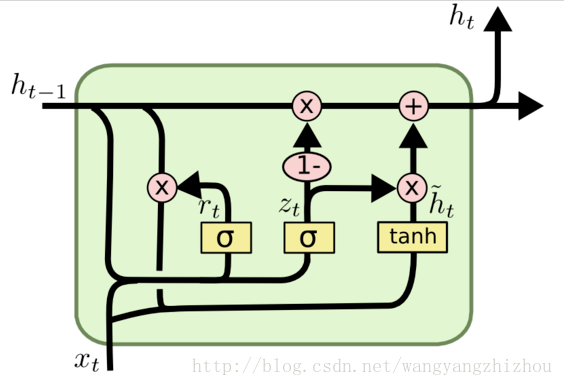
1. 现在的深度学习出现了太多的炒作。

**GRU概述**

GRU即Gated Recurrent Unit。为了克服RNN无法很好处理远距离依赖提出了LSTM，而GRU则是LSTM的一个变体，当然LSTM还有有很多其他的变体。

GRU是LSTM网络的一种效果很好的变体，它较LSTM网络的结构更加简单，而且效果也很好，因此也是当前非常流形的一种网络。GRU既然是LSTM的变体，因此也是可以解决RNN网络中的长依赖问题。

　　在LSTM中引入了三个门函数：输入门、遗忘门和输出门来控制输入值、记忆值和输出值。而在GRU模型中只有两个门：分别是更新门和重置门。具体结构如下图所示：



图中的zt和rt分别表示更新门和重置门。更新门用于控制前一时刻的状态信息被带入到当前状态中的程度，更新门的值越大说明前一时刻的状态信息带入越多。重置门控制前一状态有多少信息被写入到当前的候选集 h~th~t 上，重置门越小，前一状态的信息被写入的越少。

**算法二：**

运用了三种算法，线性回归、XGBoost与神经网络来处理时间序列。

在辨别异常值时可以选择最好的模型来进行最优选择从而获得最优处理。

在最终出结果时，通过训练Keras预测的残差和与XGBoost模型相同的特征加上来自Huber回归量和Keras模型的折叠预测来整合xgboost中的所有内容，从而提升性能。

**算法三：**

使用单个神经网络对所有145k时间序列进行建模。模型架构类似于WaveNet，由一堆扩张的因果卷积组成。

但前面提到：RNN无法很好处理远距离依赖。下面是选手的一些做法：

为了生成整个预测范围（64天）的连贯预测，使用下一步预测训练WaveNet，因此在没有条件信息的情况下模型生成长序列时误差会累积。为了解决这个问题，我们对模型进行了训练，以最大限度地减少64步拆开时的损失。我们采用序列到序列方法。这允许解码器在生成长序列时处理累积噪声。

5. 总结与展望

**5.1 总结**

此次竞赛主要是处理时序问题，通过提供的以前的网站流量来预测以后的。

特征主要是寻找各个网站流量的变化规律，找出影响它们变化背后的特征，从而在预测时能更周到的考虑到这些因素，进行更精准的预测。

主要难度有如何处理异常值的问题。

选择算法主要考虑如何对长时间序列进行学习其变化规律，又要求其对将来的预测既对季节性变化规律有体现，又能符合近期的变化规律。

**5.2 建模思路**

寻找变化规律，选择特征

数据归一化

使用GRU处理时序序列

SMAPE变体评估

优化模型，调节参数