Web Traffic Time Series Forecasting

1 Overview

Goal To forecast the future values of multiple time series. In this competition the challenge is **to predict the return of a stock, given the history of the past few days.**

Evaluation Method: SMAPE(Symmetric mean absolute percentage error)。

Highlight two stages of competition, second part is based on real-time data.

2 Data

train_*.csv - contains traffic data. This a csv file where each row corresponds to a particular article and each column correspond to a particular date. Some entries are missing data. The page names contain the *Wikipedia project (e.g. en.wikipedia.org), type of access (e.g. desktop) and type of agent (e.g. spider).*

In other words, each article name has the following format: 'name_project_access_agent' (e.g. 'AKB48_zh.wikipedia.org_all-access_spider').

key_*.csv - gives the mapping between the page names and the shortened Id column used for prediction

sample_submission_*.csv - a submission file showing the correct format

```
In [5]: train_set.index
Out[5]: RangeIndex(start=0, stop=145063, step=1)

In [7]: train_set.columns
Out[7]:
Index(['Page', '2015-07-01', '2015-07-02', '2015-07-03', '2015-07-04',
       '2015-07-05', '2015-07-06', '2015-07-07', '2015-07-08', '2015-07-09',
       ...
       '2016-12-22', '2016-12-23', '2016-12-24', '2016-12-25', '2016-12-26',
       '2016-12-27', '2016-12-28', '2016-12-29', '2016-12-30', '2016-12-31'],
      dtype='object', length=551)
```

so 550 daily traffic data of 145063 Wikepedia sites for the training set

3 Top Solution Top 1 Solution with Code, Top 2 Discussion, Top 6 With Code

Key Methods: RNN seq2seq, 就这些github上已经非常全了，以下主要概括Top 1的solution

3.1 Information Sources for Prediction

- Local features. If we see a trend, we expect that it will continue **(AutoRegressive model)**, if

we see a traffic spike, it will gradually decay **(Moving Average model)**, if wee see more traffic on holidays, we expect to have more traffic on holidays in the future **(seasonal model)**.
- Global features. If we look to autocorrelation plot, we'll notice strong year-to-year autocorrelation and some quarter-to-quarter autocorrelation.

The good model should use both global and local features, combining them in a intelligent way.

3.2 Model Selection - RNN seq2seq

1. RNN can be thought as a natural extension of well-studied ARIMA(Autoregressive integrated moving average) models, but much more flexible and expressive.
2. RNN is non-parametric, that's greatly **simplifies learning**. Imagine working with different ARIMA parameters for 145K timeseries.
3. Any exogenous feature (numerical or categorical, time-dependent or series-dependent) can be easily injected into the model
4. seq2seq seems natural for this task: we predict next values, conditioning on joint probability of previous values, including our past predictions. Use of past predictions stabilizes the model, it learns to be conservative, because error accumulates on each step, and extreme prediction at one step can ruin prediction quality for all subsequent steps.
5. Deep Learning is all the hype nowadays —> RNN。。。

3.3 Feature Engineering

I tried to be minimalistic, because **RNN is powerful enough to discover and learn features on its own.** Model feature list:

- *pageviews* (spelled as 'hits' in the model code, because of my web-analytics background). Raw values transformed by log1p() to get more-or-less normal intra-series values distribution, instead of skewed one.
- *agent*, *country*, *site* - these features are extracted from page urls and one-hot encoded
- *day of week* - to capture weekly seasonality
- *year-to-year autocorrelation*, *quarter-to-quarter autocorrelation* - to capture yearly and quarterly seasonality strength.
- *page popularity* - High traffic and low traffic pages have different traffic change patterns, this feature (median of pageviews) helps to capture traffic scale. This scale information is lost in a *pageviews* feature, because each pageviews series independently normalized to zero mean and unit variance.
- ***lagged pageviews* - I'll describe this feature later**

3.4 Feature Preprocessing

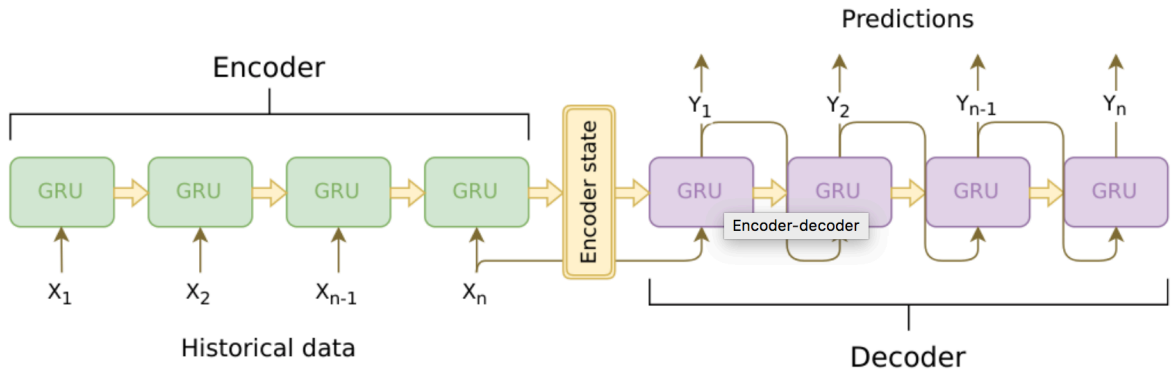
- All features (including one-hot encoded) are **normalized** to zero mean and unit variance. Each *pageviews* series normalized independently.
- Time-independent features (autocorrelations, country, etc) are "stretched" to timeseries length i.e. repeated for each day by `tf.tile()` command.
- Model trains on random fixed-length samples from original timeseries. For example, if original timeseries length is 600 days, and we use 200-day samples for training, we'll have

a choice of 400 days to start the sample.

This sampling works as effective data augmentation mechanism: training code randomly chooses starting point for each timeseries on each step, generating endless stream of almost non-repeating data.

3.5 Model Core

Model has two main parts: encoder and decoder.

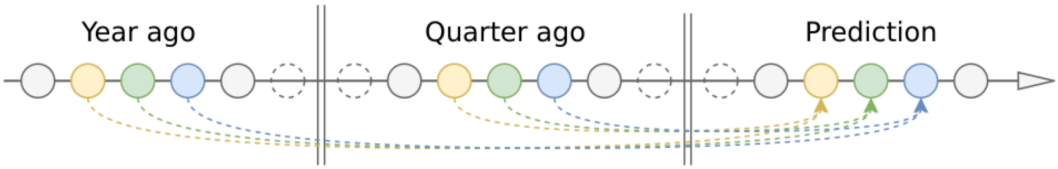


Encoder is cuDNN GRU. cuDNN works much faster (5x-10x) than native Tensorflow RNNCells, at the cost of some inconvenience to use and poor documentation.

Decoder is TF `GRUBlockCell`, wrapped in `tf.while_loop()` construct. Code inside the loop gets prediction from previous step and appends it to the input features for current step.

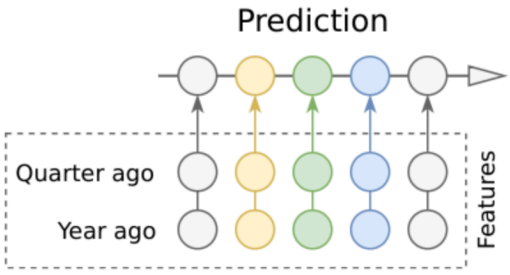
3.6 Working with long timeseries

- LSTM/GRU能够很好的处理短序列的问题(100-300items)，但比赛中涉及到超过700天的数据，因此作者采用了一些方法来'strengthen' GRU memory.
  - First trial is called 'attention' method:



    I can just take encoder outputs from `current_day - 365` and `current_day - 90` timepoints, pass them through FC layer to reduce dimensionality and append result to input features for decoder. This solution, despite of being simple, **considerably lowered prediction error.**
  - Then, to reduce the noise: `attn_365 = 0.25 * day_364 + 0.5 * day_365 + 0.25 * day_366`
  - Then I realized that `0.25,0.5,0.25` is a 1D convolutional kernel (length=3) and I can automatically learn bigger kernel to detect important points in a past.（非常6）
  - Ended up with a monstrous attention mechanism not classical method such as **(Bahdanau or Luong attention)**, because of calculation cost.

- 但其实最好的public score反倒是最简单的模型：



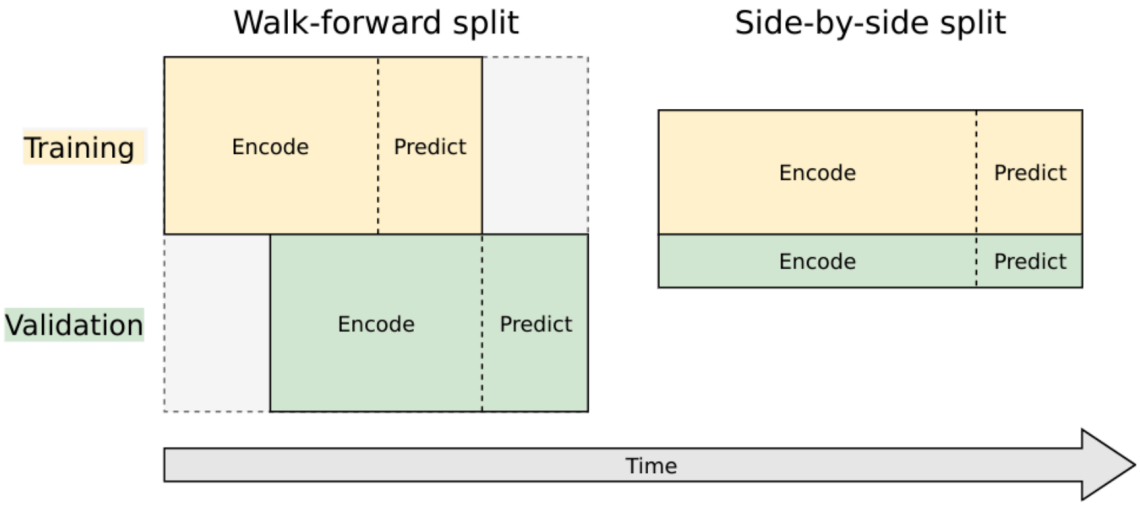
- 另外发现60-90day的encoder表现也还okay

3.7 Redefine Loss Function

- SMAPE (target loss for competition) can't be used directly, because of unstable behavior near zero values (loss is a step function if truth value is zero, and not defined, if predicted value is also zero).
- Candidates include *differentiable SMAPE* and *MAE*

3.8 Training and validation

- COCOB optimizer
- Prefer walk-forward split rather than side-by-side split



3.9 Reducing model variance

- Performance varies from training steps and seed.
- Solutions:
  - But I know approximate region where model is (possibly) trained well enough, but (possibly) not started to overfit. I decided to set this optimal region bounds to 10500..11500 training steps and save 10 checkpoints from each 100th step in this region.
  - Similarly, I decided to train 3 models on different seeds and save checkpoints from each model. So I have 30 checkpoints total.
  - One widely known method for reducing variance and improving model performance is SGD averaging (ASGD). Method is very simple and well supported in Tensorflow -

we have to maintain moving averages of network weights during training and use these averaged weights, instead of original ones, during inference.
    - 然后 AVERAGE！！！

3.10 Hyperparameter Tuning

略略略。。。

4 Comment

solution set非常丰富，1st place有着详细的code和readme，用的是RNN，有一些地方feature extraction不清晰，可以先做。