Imperial College London

Department of Mathematics

Title of the Thesis

FIRSTNAME LASTNAME

CID: 01234567

Supervised by SUPERVISORNAME and COSUPERVISORNAME

1 May 2022

Submitted in partial fulfilment of the requirements for the MSc in Statistics of Imperial College London

Abstract

ABSTRACT GOES HERE

Acknowledgements

ANY ACKNOWLEDGEMENTS GO HERE

Contents

Notation

$\boldsymbol{X}$ is a matrix

y is a vector

Abbreviations

DAIN	Deep Adaptive Input Normalization
RDAIN	Robust Deep Adaptive Input Normalization
EDAIN	Extended Deep Adaptive Input Normalization
BIN	Bilinear Input Normalization

1 Introduction

The introduction section goes here[1].

[1]Tip: write this section last.

2 Background

TODO: introduction to this chapter

2.1 Deep learning

TODO: write details

2.1.1 Sequence models

2.2 Data preprocessing

TODO: introduction

2.2.1 Static distribution transformations

2.2.2 Adaptive distribution transformations

DAIN

The Deep Adaptive Input Normalization (DAIN) method...

RDAIN

BiN

2.3 Normalizing flows

TODO: write details

3 Methods

TODO: introduction to this chapter

3.1 EDAIN

My first contribution is the Extended Deep Adaptive Input Normalization (EDAIN) layer. This adaptive preprocessing layer is inspired by the likes of DAIN_REF and BIN_REF, but unlike the aforementioned methods, the EDAIN layer also supports normalizing the data in a *batch-agnostic* fashion, whereas the DAIN, Robust Deep Adaptive Input Normalization (RDAIN) and Bilinear Input Normalization (BIN) layers are all *batch-aware*. Additionally, the EDAIN layer extends the other layers with two new operations: An outlier removal operation that is designed to reduce the negative impact of high-tail observations, as well as a power-transform operation that is designed to transform non-normal data to be more normal.

Notation

Let $\{\mathbf{X}^{(i)} \in \mathbb{R}^{d\times T}; i = 1, \ldots, N\}$ denote a set of N multivariate time-series, each composed of T d-dimensional feature vectors. We also let $\mathbf{x}_t^{(i)} \in \mathbb{R}^d$, where $t = 1, \ldots, T$, denote the tth feature vector at time-step t in the time-series.

Architecture

An overview of the layer's architecture is shown in figure TODO_FIG. First, outlier removal. Then scale. Then shift. Then power transform. This order because TODO.

TODO: explain comparison to DAIN, RDAIN and BIN, and how not batch-aware, but rather aims to learn characteristics of global distribution, as it's the global distribution that might be very irregular...

Outlier removal

Handling outliers and extreme values in the dataset can increase predictive performance if done correctly (citation needed). Two common ways of doing this are omission and winsorization (Nyitrai and Virág, 2019). With the former, observations that are deemed to be extreme are simply removed during training. With the latter, all the data is still used, but observations lying outside a certain number of standard deviation from the mean, or below or above certain percentiles, are *clamped down* to be closer to the mean or median of the data. For example, if winsorizing data using 3 standard deviation, all values less than $\mu - 3\sigma$ are set to be exactly $\mu - 3\sigma$. Similarly, the values above $\mu + 3\sigma$ are *clamped* to this value. Winsorization can also be done using percentiles, where common boundaries are the first and fifth percentiles Nyitrai and Virág (2019). However, the type of winsorization, as well as the number of standard deviation or percentiles to use, might depend on the dataset. Additionally, it might not be necessary to winsorize the data at all if the outliers turn out to not negatively affect performance. All this introduces more hyperparameters to tune during modelling. The outlier removal presented here aims to automatically both determine whether winsorization is necessary for a particular feature, and determine the threshold at which to apply winsorization.

For input vector $\mathbf{x} \in \mathbb{R}^d$, the adaptive outlier removal operation is defined as:

$$\boldsymbol{\alpha}' \odot \underbrace{\left(\boldsymbol{\beta}' \odot \tanh\left\{(\mathbf{x} - \hat{\boldsymbol{\mu}}) \oslash \boldsymbol{\beta}'\right\} + \hat{\boldsymbol{\mu}}\right)}_{\text{smooth adaptive centred winsorization}} + \underbrace{(1 - \boldsymbol{\alpha}') \odot \mathbf{x}}_{\text{residual connection}}, \tag{3.1}$$

where $\odot$ is the element-wise multiplication, $\oslash$ is element-wise division, $\boldsymbol{\alpha}' \in [0,1]^d$ is a parameter controlling how winsorization to apply to each feature, and $\boldsymbol{\beta}' \in [\beta_{\min}, \infty)^d$ controls the winsorization threshold for each feature. The $\hat{\boldsymbol{\mu}}$ parameter is an estimate of the mean of the data, and is used to ensure the winsorization is centred. When setting the EDAIN layer in *batch-aware* mode, it is simply the mean of the batch:

$$\hat{\mu}_k = \frac{1}{|\mathcal{B}|T} \sum_{i \in \mathcal{B}} \sum_{t=1}^{T} x_{j,k}^{(i)}, \tag{3.2}$$

while if using the *batch-agnostic* mode, it is iteratively updated using a cumulative moving average estimate at each forward pass of the layer. This is to better approximate the global mean of the data. The unknown parameters of the model are $\boldsymbol{\alpha} \in \mathbb{R}^d$ and $\boldsymbol{\beta} \in \mathbb{R}^d$, and they are transformed into the constrained parameters $\boldsymbol{\alpha}'$ and $\boldsymbol{\beta}'$, as used in eq. (3.1) through the following element-wise mappings:

$$\boldsymbol{\alpha}' = \frac{e^{\boldsymbol{\alpha}}}{1 + e^{\boldsymbol{\alpha}}} \qquad \boldsymbol{\beta}' = \beta_{\min} + e^{\boldsymbol{\beta}}. \tag{3.3}$$

For ease of notation, we let $\mathbf{W}_1 = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ denote the $2d$ unknown parameters that are optimised for the adaptive outlier removal layer.

TODO: plots of curve for different parameters

Scale and shift

The adaptive shift and scale layer, combined, simply performs the operation

$$(\mathbf{x} \oplus \boldsymbol{\gamma}) \odot \boldsymbol{\lambda}, \tag{3.4}$$

with input $\mathbf{x}$ and unknown parameters $\boldsymbol{\gamma} \in \mathbb{R}^d$ and $\boldsymbol{\lambda} \in (0, \infty)^d$ when the EDAIN is set to batch-agnostic mode. This makes the scale-and-shift layer a generalised version of Z-score scaling, or standard scaling, as setting

$$\boldsymbol{\gamma} := -\frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbf{x}_t^{(i)} \tag{3.5}$$

and

$$\boldsymbol{\lambda} := \left(\frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\mathbf{x}_t^{(i)} \oplus \boldsymbol{\gamma} \right)^2 \right)^{-1} \tag{3.6}$$

makes the operation in eq. (3.4) equivalent to Z-score scaling. This *batch-agnostic* mode is useful if the distribution is similar across batches and constitute a global unimodal distribution that should be centred.

However, some datasets might have multiple modes arising from significantly different data generation mechanisms. Attempting to scale and shift each batch to a global mean and standard deviation might hurt performance in such cases. Instead, CITE_AUTHOR_DAIN propose basing the scale and shift on a *summary representation* of the current batch, allowing each batch to be normalized according the specific mode that batch of data might have come from. This gives

$$(\mathbf{x} \oplus [\boldsymbol{\gamma} \odot \mu_{\mathbf{x}}]) \odot [\boldsymbol{\lambda} \odot \sigma_{\mathbf{x}}], \tag{3.7}$$

where the summary representations $\sigma_{\mathbf{x}}$ and $\mu_{\mathbf{x}}$ are computed through reduction of the temporal dimension for each observation:

$$\mu_{\mathbf{x}}^{(i)} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{x}_t^{(i)} \in \mathbb{R}^d \tag{3.8}$$

$$\sigma_{\mathbf{x}}^{(i)} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} \left(\mathbf{x}_t^{(i)} - \mu_{\mathbf{x}}^{(i)} \right)^2} \in \mathbb{R}^d. \tag{3.9}$$

With this mode, it is difficult for the layer to generalise Z-score scaling, but it becomes more able to normalize in a *mode-aware* fashion.

Power transform

Many real-world datasets exhibit significant skewness, which is often treated using power transformations (citation needed). The most common transformation is the Box-Cox transformation, but this is only valid for positive values, so it is not applicable to most real-world datasets TODO_CITE_BOX_COX. An alternative is a transformation proposed by TODO_CITE_YEO_Johnson, who proposed to following transformation:

$$f_{\mathrm{YJ}}(x) = \begin{cases} \frac{(x+1)^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0, x \geq 0; \\ \log(x+1), & \text{if } \lambda = 0, x \geq 0; \\ \frac{(1-x)^{2-\lambda} - 1}{\lambda - 2}, & \text{if } \lambda \neq 2, x < 0; \\ -\log(1-x), & \text{if } \lambda = 2, x < 0. \end{cases} \tag{3.10}$$

Like the Box-Cox transformation, transformation f_{YJ} only has one unknown parameter, λ, but it works for any $x \in \mathbb{R}$, not just positive values.

The power transform layer simply applies the transformation in eq. (3.10) along each dimension of the input, that is for each $i = 1, \dots, N$ and $t = 1, \dots, T$,

$$\left[\text{power\_transform}\left(\mathbf{x}_t^{(i)}\right)\right]_j = f_{\mathrm{YJ}}(x_{t,j}^{(i)}), \quad j = 1, \dots, d. \tag{3.11}$$

The unknown parameters is the vector $\boldsymbol{\lambda} \in \mathbb{R}^d$.

Optimising the parameters

Can simply feed values forward through the 4 layers, as shown in TODO_FIG. Additionally, weights are updated through standard gradient descent simultaneously while training the model

$$\Delta(\mathbf{W}_\alpha, \beta) = -\eta \left(\eta_{pt} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_a}, \dots \right) \tag{3.12}$$

Similarly to DAIN_REF, convergence is unstable if use same learning rate for RNN model and preprocessing layer weights, so seperate learning rates for each parameter, and these are set using hyperparameter tuning search, described later in section TODO.

3.2 EDAIN-KL

TODO: write details

3.3 PREPMIX-CAPS

TODO: write details

4 Results

TODO: introduction to this chapter

4.1 Evaluation methodology

Small introduction

4.1.1 Sequence model architecture

4.1.2 Fitting the models

Mention scheduling, early stopping, optimizer used, learning rate etc.

4.1.3 Tuning adaptive preprocessing model hyperparameters

Details on the tuning for all the methods presented

4.1.4 Evaluation metrics

4.1.5 Cross-validation

4.2 Simulation study

Small introduction, including motivation

4.2.1 Multivariate time-series data generation algorithm

4.2.2 Negative effects of irregularly-distributed data

4.2.3 Preprocessing method experiments

4.3 American Express default prediction dataset

4.3.1 Description

4.3.2 Preprocessing method experiments

5 Discussion

TODO: introduction to this chapter

5.1 EDAIN

5.2 EDAIN-KL

5.3 PREPMIX-CAPS

6 Conclusion

6.1 Summary

Conclusion goes here.

6.2 Main contributions

6.3 Future work

References

Tamás Nyitrai and Miklós Virág. The effects of handling outliers on the performance of bankruptcy prediction models. *Socio-Economic Planning Sciences*, 67:34–42, 2019. ISSN 0038-0121. doi: https://doi.org/10.1016/j.seps.2018.08.004. URL https://www.sciencedirect.com/science/article/pii/S003801211730232X.