针对时间序列分类（TSC）的深度模型

Deep Learning for Time Series Classification[1]

Xinyi Li

2019 年 11 月 18 日

[1] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963.

Overview

问题背景描述

Strong Baseline

TSC 社区生态

Different Learning Tasks

Multi-variate

MTS

- different measurements of the **same** instance
- **high correlation**
- feeding features

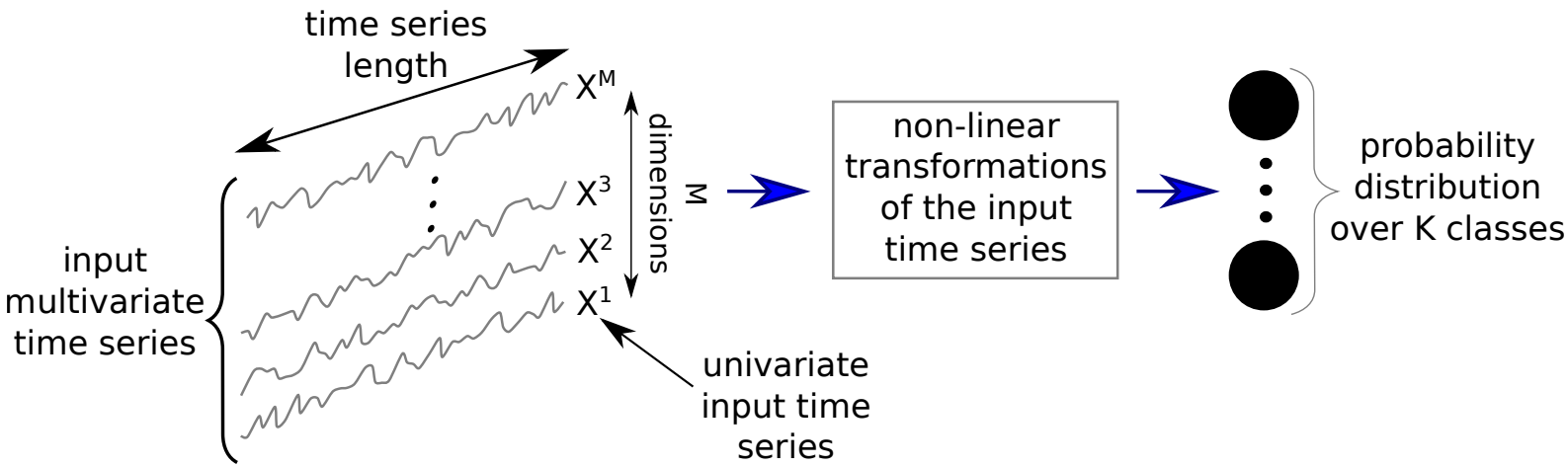
Univariate

panel data

- the same measurements on **different** instances
- i.i.d. assumption
- feeding sku/store/...

Problem Description

univariance/multi-variance/panel



HIVE-COTE[2]:

SOTA classic algorithm[3]: Collective of Transformation based **Ensembles** (COTE) with a Hierarchical Vote system

1. Elastic Ensemble(**EE**): combination of 1-NN classifiers using different measurements
2. Shapelet Transform Ensemble(**ST**): top k shaplets (independent phase short pattern)
3. Bag-of-SFA-Symbols (**BOSS**) Ensemble: shapelets based on presence or absence
4. Time Series Forest (**TSF**): trained on selected $3\sqrt{m}$ features
5. Random Interval Features (**RIF**): spectral component of *Flat-COTE*

[2] Jason Lines, Sarah Taylor, and Anthony Bagnall. "Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification". In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE. 2016, pp. 1041–1046.

[3] Anthony Bagnall et al. "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances". In: *Data Mining and Knowledge Discovery* 31.3 (2017), pp. 606–660.

HIVE-COTE Algorithm[4]

ensemble of ensembles

Refs to Flat-COTE

Algorithm 1 ProportionalEnsemble(*classifiers*, *train*, *test*)

```
1:  trainAccs = ∅;
2:  for i ← 1 to |classifiers| do
3:      trainAccs_i = loocv(train, classifiers[i])
4:      classifiers_i.buildClassifier(train)
5:  testPreds = ∅
6:  for i ← 1 to |test| do
7:      votes = ∅
8:      bsfWeight = −1;
9:      bsfClass = −1;
10:     for c ← 1 to |classifiers| do
11:         p = classifiers_c.classify(test_i)
12:         votes_p = votes_p + trainAccs_c;
13:         if votes_p > bsfWeight then
14:             bsfWeight = votes_p
15:             bsfClass = p
16:         testPreds_i = bsfClass
17: return testPreds
```

[4] https://github.com/TonyBagnall/py-hive-cote

Libraries/Implements/Community

sktime[5] & its extensions

Sktime

- based on classic models (shallow)
- `scikit-learn` interface compatible

Sktime-dl

- use `Keras` to implement all 9 **SOTA** deep models above
- 暂时**不能**直接安装（MacOS）

UEA & UCR Time Series Classification Repository

- 128 TSC datasets + 30 MTS datasets
- Collect a bunch of **classic** algorithms

[5] Markus Löning et al. *sktime: A Unified Interface for Machine Learning with Time Series*. 2019. eprint: `arXiv:1909.07872`.

MLP/DNN

Multi Layer Perceptrons/Fully-Connected(FC) Network

The Simplest DNN (e.g. `keras.Layers.Dense`)

$$\mathbf{X}_{i+1} = \sigma(\mathbf{W}_i \mathbf{X}_i + b_i)$$

Final (l-th) layer activate function: softmax

$$\hat{y}_k(\mathbf{X}_{l-1}) = (e^{\mathbf{W}_k \mathbf{X}_{l-1} + b_k}) / (\sum_{i=1}^{K} e^{\mathbf{W}_i \mathbf{X}_{l-1} + b_i})$$

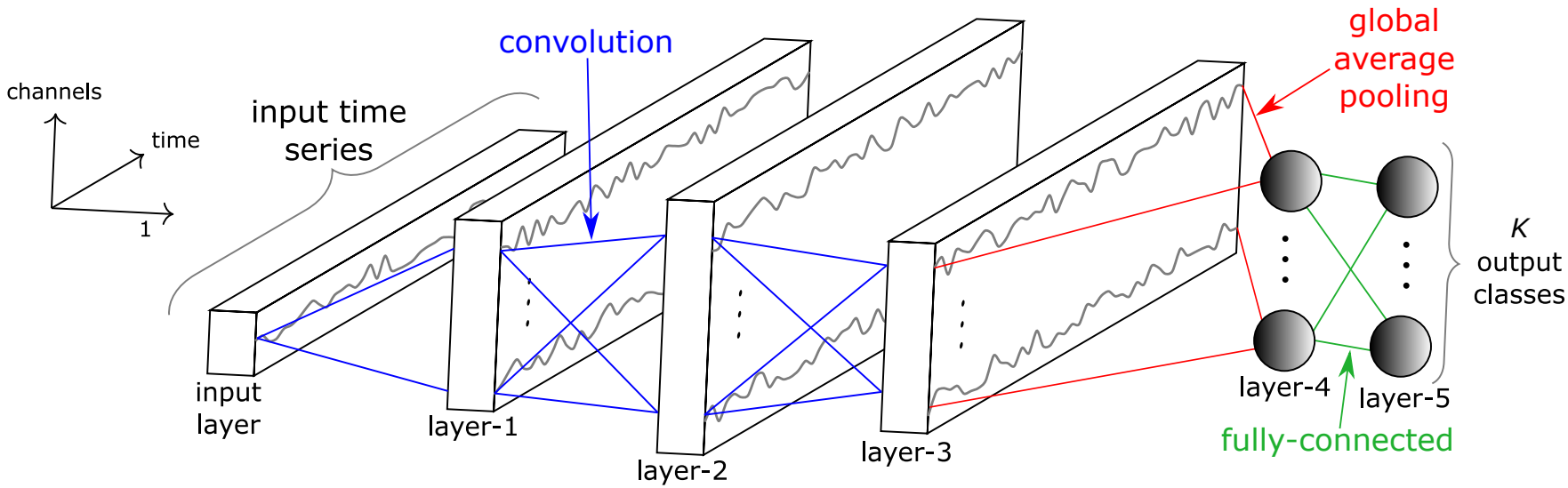
Objective loss: categorical cross entropy

$$Loss(\mathbf{X}) = -\sum_{i=1}^{K} y_i \log \hat{y}_i$$

minimized to learn the weights using **gradient descent** method

FCNN[6]

Fully Convolutional Neural Network



convolution layer ($\forall$ time stamp t shares filter ω with length l)

$$\mathbf{C}_t = \sigma(\omega * \mathbf{X}_{t-l/2:t+l/2} + b) | \forall t \in [1, T]$$

[6] John Cristian Borges Gamboa. "Deep learning for time-series analysis". In: *arXiv preprint arXiv:1701.01887* (2017).

Thanks

All codes, slides and papers available

li-xin-yi/deep_time_series_share_slide