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 20 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
- 深度模型结构
  - MLP/DNN
  - FCN
  - ResNet
  - Encoder
  - MCNN
  - Time Le-Net
  - MCDCNN
  - Time-CNN
  - TWIESN

Different Learning Tasks

Multi-variate

MTS

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

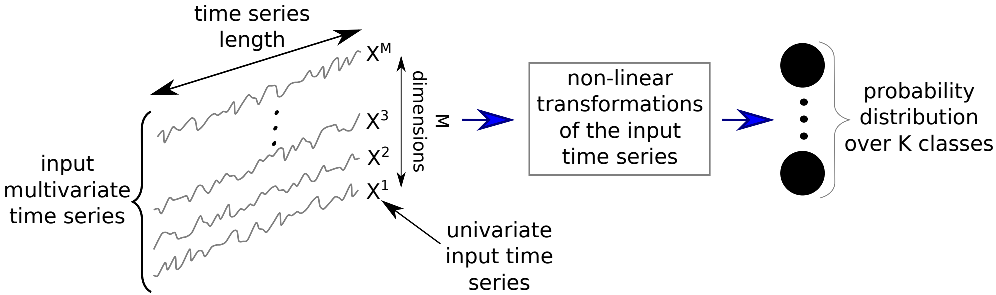
Univariate

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: ref to Flat-COTE

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

```
trainAccs = ∅;
for i ← 1 to |classifiers| do
    trainAccs_i = loocv(train, classifiers[i])
    classifiers_i.buildClassifier(train)
testPreds = ∅
for i ← 1 to |test| do
    votes = ∅
    bsfWeight = −1;
    bsfClass = −1;
    for c ← 1 to |classifiers| do
        p = classifiers_c.classify(test_i)
        votes_p = votes_p + trainAccs_c;
        if votes_p > bsfWeight then
            bsfWeight = votes_p
            bsfClass = p
    testPreds_i = bsfClass
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

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

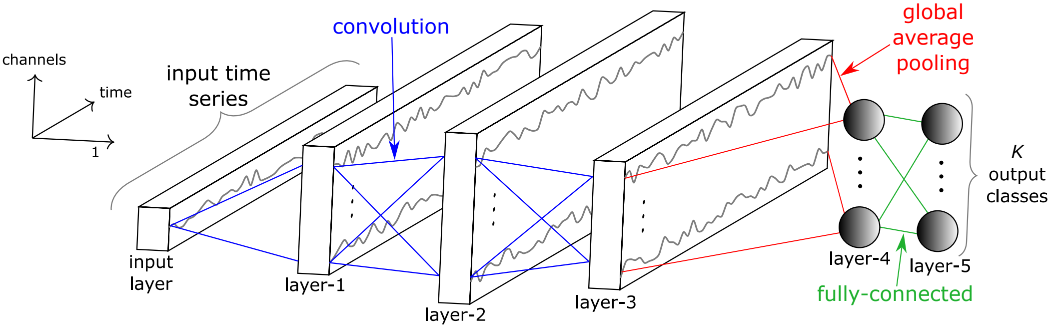
Objective loss: categorical cross entropy

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

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

FCN[6]

Fully Convolutional Neural Network



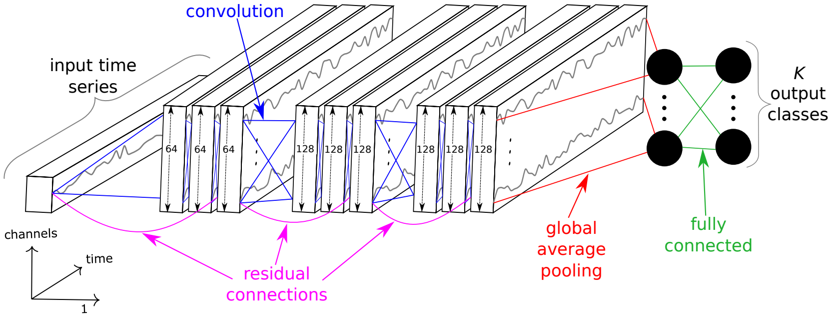
convolution layer ($\forall$ time stamp t shares filter ω with length l)

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

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

ResNet[7]

Residual Network



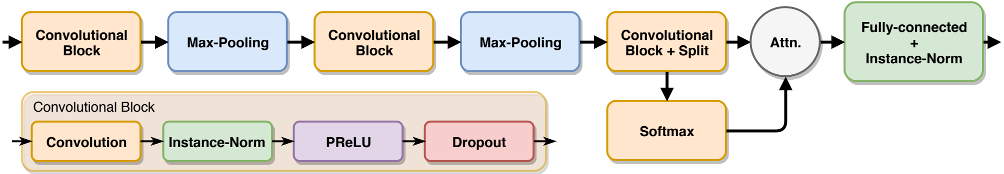
[7] Zhiguang Wang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline". In: *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 1578–1585.

Encoder[8]

Hybrid deep CNN based on FCN

Modified from FCN

- GAP layer $\rightarrow$ attention layer (careful design for pre-train)
- normalization for each Conv layer output:
  1. ReLU $\rightarrow$ PReLU activation function (+ parameter)
  2. + dropout regularization
  3. + max pooling



[8] Joan Serrà, Santiago Pascual, and Alexandros Karatzoglou. "Towards a Universal Neural Network Encoder for Time Series.". In: *CCIA*. 2018, pp. 120–129.

MCNN[9]

Multi-scale Convolutional Neural Network

Similar to Traditional CNN:

1. 2 Conv layer (with max pooling)
2. 1 Fully-connected layer
3. final softmax layer

Heavy data pre-preprocessing step:

Window Slicing(WS)

for data augmentation:

1. slide a window over raw input
2. extract subsequences

Before training, $\forall$ subsequence

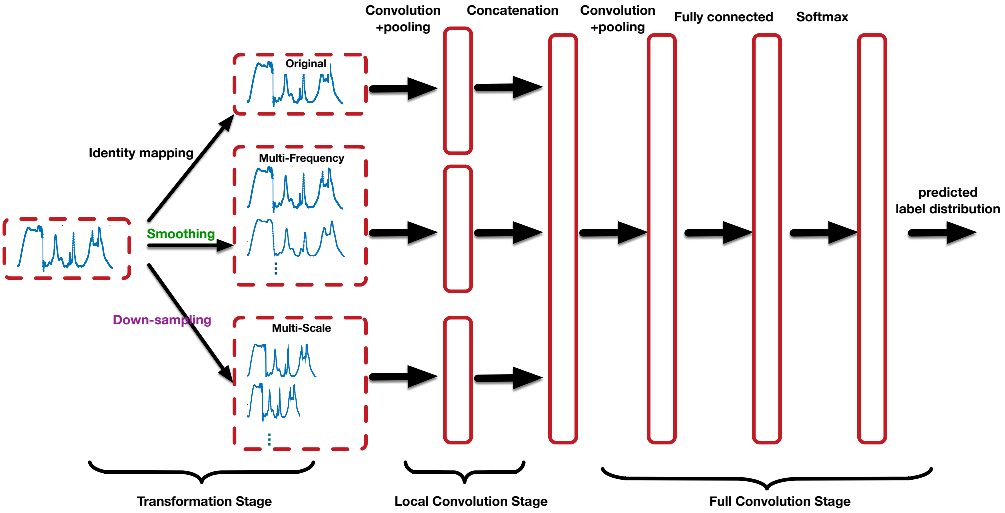
Transformations (parallel)

1. identity mapping
   - keep unchanged $\rightarrow$ 1-st Conv
2. down-sampling
   - $\rightarrow$ different shorter lengths subsequences
   - $\rightarrow$ 1-st Conv
3. smoothing:
   - equal length one
   - $\rightarrow$ 1-st Conv
   - $\rightarrow$ 2-nd Conv

[9] Zhicheng Cui, Wenlin Chen, and Yixin Chen. "Multi-scale convolutional neural networks for time series classification". In: *arXiv preprint arXiv:1603.06995* (2016).

MCNN

Framework



Time Le-Net[10]

Inspired by Le-Net[11],like CNN: 2 Conv + FC + final softmax

Compared to FCN:
GAP $\rightarrow$ FC

Local max pooling

- take max in a local pooling
- $+$ invariance to small perturbations
- shorten a time series

Still #parameters $\uparrow$ #invariance $\downarrow$

Data augmentation to prevent overfitting
especially on relatively small datasets

Window Slicing(WS)

$=$ method used in MCNN

Window Warping (WW)

For a time series with length l

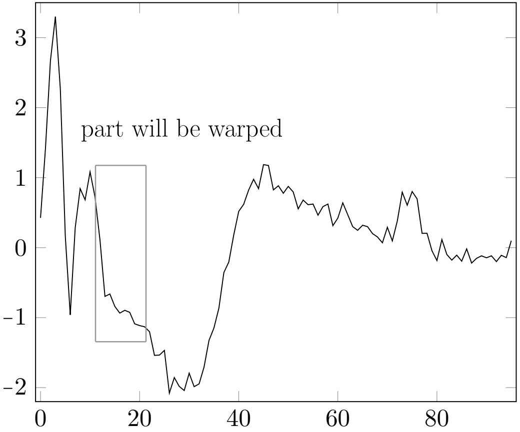
1. dilate $(\times 2) \rightarrow 2l$
2. squeeze $(\times \frac{1}{2}) \rightarrow \frac{1}{2}l$

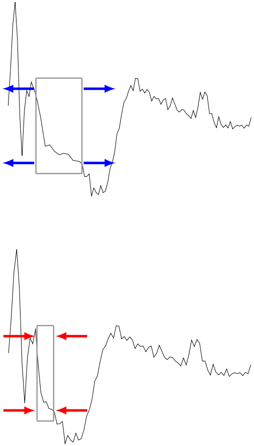
[10] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. "Data augmentation for time series classification using convolutional neural networks". In: 2016.

[11] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

Window Warping (WW)

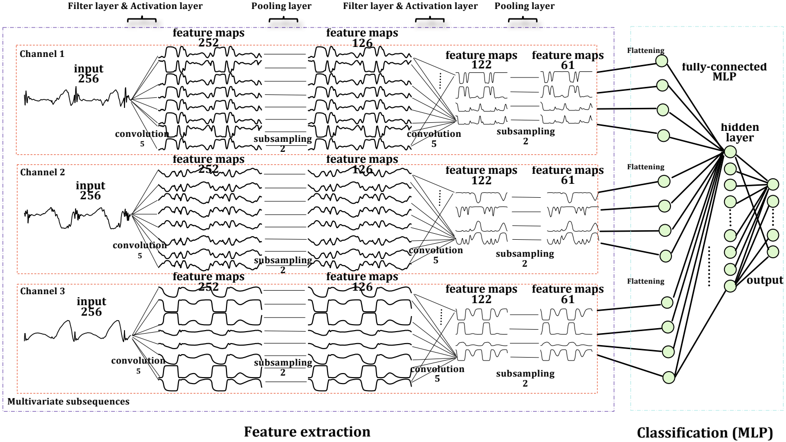
Then feed all data (length=l,$2l$,$\frac{1}{2}l$) into network





MCDCNN[12]

Multi Channel Deep Convolutional Neural Network: **Independent** Conv specified for **MTS**



[12] Yi Zheng et al. "Time series classification using multi-channels deep convolutional neural networks". In: *International Conference on Web-Age Information Management*. Springer. 2014, pp. 298–310.

Time-CNN[13]

both for univariate and multivariate

Main differences compared to previous models:

1. loss function: categorical cross-entropy → MSE
2. activate function of final layer: softmax → sigmoid

$$\sum_{i=1}^{K} p(\hat{Y}_i) \neq 1$$

3. throughout CNN: local max pooling → local average pooling

Compared to some models:

- apply 1 Conv for all dimensions (unlike MCDCNN)
- Conv directly fully connected to final layer (modify FCN: GAP → FC)

[13] Bendong Zhao et al. "Convolutional neural networks for time series classification". In: *Journal of Systems Engineering and Electronics* 28.1 (2017), pp. 162–169.

TWIESN[14]

Time Warping Invariant Echo State Network

[14] Pattreeya Tanisaro and Gunther Heidemann. "Time series classification using time warping invariant echo state networks". In: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2016, pp. 831–836.

Thanks

All codes, slides and papers available

li-xin-yi/deep_time_series_share_slide