Neurocomputing 000 (2017) 1–4

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Original software publication

Gaussian kernel smooth regression with topology learning neural networks and Python implementation

Zhiyang Xiang, Zhu Xiao, Dong Wang*, Jianhua Xiao

College of Computer Science and Electronics Engineering, Hunan University, Changsha 410082, China

ARTICLE INFO

Article history:
Received 19 July 2016
Revised 11 November 2016
Accepted 16 January 2017
Available online xxx

Keywords:
Kernel density estimation
Semi-supervised regression
Python

ABSTRACT

Topology learning neural networks such as Growing Neural Gas (GNG) and Self-Organizing Incremental Neural Network (SOINN) are online clustering methods. With GNG and SOINN implemented as basic learners, this software completes two machine learning tasks, namely density estimation and regression. A kernel density estimation framework is implemented to transform the topology learning neural networks into density estimation methods. Besides, a kernel smoother to implement supervised and semi-supervised regression is devised. Moreover, the implemented frameworks can be used to transform other clustering methods into density estimation, supervised regression and semi-supervised regression.

Software metadata	
(executable) Software metadata description	
Current software version	0.1
Permanent link to executables of this version	https://github.com/Neurocomputing/NEUCOM-D-16-02468
Legal Software License	BSD-3
Computing platform/Operating System	BSD, Linux, OS X, Microsoft Windows, Unix-like.
Installation requirements & dependencies	Python 2.7, numpy 1.09+, sklearn 0.15+, python-graph 1.8
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://sbxzy.github.io
Support email for questions	lordxzy@qq.com

Code metadata	
Code metadata description	
Current code version	0.1
Permanent link to code/repository used of this code version	https://github.com/Neurocomputing/NEUCOM-D-16-02468
Legal Code License	BSD-3
Code versioning system used	none
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python 2.7, numpy 1.09+, sklearn 0.15+, python-graph 1.8
If available Link to developer documentation/manual	https://sbxzy.github.io
Support email for questions	lordxzy@qq.com

1. Introduction

The topology learning neural networks of Growing Neural Gas (GNG) and Self-Organizing Incremental Neural Networks (SOINN) are further developments of the famous Self-Organizing Map (SOM). Unfortunately, among the mainstream machine learning softwares such as Sklearn [1], there is no such implementations. In addition, there has not been an publicly available implementation of the improved SOINN [2].

Semi-supervised learning is an active research area. There are great efforts in classification algorithm researches, but for semi-supervised regression, there is not enough attention.

The main contributions of the work are as follows.

* Corresponding author.
E-mail address: wangd@hnu.edu.cn (D. Wang).

http://dx.doi.org/10.1016/j.neucom.2017.01.051
0925-2312/

1. A novel semi-supervised regression framework called Semi-Supervised Learning Gaussian Kernel Smoother (SSL-GKS) is proposed.
2. Kernel density estimation based on GNG and SOINN is implemented.
3. The proposed framework can be used in combine with any clustering methods for semi-supervised regression.

2. Problems and background

From statistical point of view, the regression learning task is equivalent to modeling the joint distribution of explanatory and response variables. According to kernel density estimation (KDE) [3], joint distribution of explanatory variables X and response variables Y can be represented by weights of clustering centers $\mathbf{W} = \{W_i\}$, where $W_i \in \mathbb{R}^d$ and index i are used for iteration of all cluster centers, $\mathbf{c} = \{c_i\}$ is the distribution of clustering centers, and s is the smooth parameter. The learning task of KDE can be written as $P(X, Y|\mathbf{W}, \mathbf{c}, s)$. According to Nadaraya–Watson estimator [3] and its multivariate case in [4], clustering based regression task, i.e. $P(Y|X; \mathbf{W}, \mathbf{c}, s)$, can be written as

$$\hat{Y} = f(X) = \sum_i \frac{c_i Y_i w_i(X)}{\sum_j c_j w_j(X)} \tag{1}$$

$$w_i(X) = \frac{1}{\sqrt{(2\pi)^d |\mathbf{H}|}} \exp(-0.5(X - W_i)^T \mathbf{H}^{-1}(X - W_i)) \tag{2}$$

d is the number of dimensions, and $\mathbf{H}$ is a diagonal matrix with squares of bandwidth h along the diagonal calculated as the variance of dataset along each dimension multiplied by a smooth parameter: $h_k^2 = s\delta_k^2$. In KDE, s is non-parametrically chosen. However, in regression a different strategy must be employed for that regression should emphasize on fitting the test dataset while KDE only fits the training dataset.

We propose a semi-supervised learning process to address this problem. Denote a labeled dataset as $\mathcal{D}_L = \{X_L, Y_L\}$, and unlabeled dataset $\mathcal{D}_U = \{X_U, Y_U\}$, and clustering weights on them $\mathbf{W}_L$ and $\mathbf{W}_U$ respectively, and clustering weights of the explanatory variables as $\mathbf{W}_L^X$ and $\mathbf{W}_U^X$. The SSL learning purpose can be formulated as $P(Y|X; \mathbf{W}_L \bigcup \mathbf{W}_U, \mathbf{c}, s)$. $\mathbf{W}_U$ is unknown, while the similarity relationships between $\mathbf{W}_L$ and $\mathbf{W}_U$ can be approximated by $\mathbf{W}_L^X$ and $\mathbf{W}_U^X$. By using $\mathbf{W}_L^X$ and $\mathbf{W}_U^X$ to calculate a smooth parameter that fits the complete dataset rather than the training dataset only, we propose the following framework

$$P(Y|X; \mathbf{W}_L, \mathbf{c}, s) = P(Y|X; \mathbf{W}_L, \mathbf{c}, s) P(s|\mathbf{W}_L^X, \mathbf{W}_U^X) \tag{3}$$

Since joint distributions can be learned by topology preserving and topology learning neural networks [5,6], regression can also be accomplished by such neural networks. In this work, we implement the Gaussian kernel smoother of Eq. (1), and the SSL framework in Eq. (3), and two topology learning neural networks, namely GNG and SOINN, to accomplish the regression task.

2.1. The proposed SSL-GKS framework

Using $H = diag(s\delta_1^2, s\delta_2^2, \ldots, s\delta_k^2, \ldots)$ to rewrite the KDE equation:

$$P(X|s) = \sum_i \frac{1}{\sqrt{(2\pi)^d s^d \prod_k \delta_k^2}} \exp(-0.5 s^{-1}(X - W_i)^T \mathbf{\Delta} (X - W_i)) \tag{4}$$

where $\mathbf{\Delta} = diag(\delta_1^{-2}, \delta_2^{-2}, \ldots, \delta_k^{-2}, \ldots)$. Assume that a cluster center $W_U \in \mathbf{W}_U^X$ is generated by the most likely (training set) cluster center. Since each cluster center represents the center of a Gaussian distribution with the same covariance matrix $\mathbf{H}$, we select the Mahalanobis distance to calculate the similarity between cluster centers.

Then the likelihood function of the cluster centers generated by test dataset can be written as $L(s|\mathbf{W}_L^X, \mathbf{W}_U^X) = \prod_{W_U \in \mathbf{W}_U^X} \prod_{i=1}^K P(W_U|W(i); s)$ where $W(i)$ means the ith nearest cluster center (generated by training set) to W_U. Replacing P with Gaussian kernel, the log-likelihood function becomes $L(s|\mathbf{W}_U^X) = \sum_{W_U \in \mathbf{W}_U^X} \sum_{i=1}^K (\ln \frac{c_i}{\sqrt{(2\pi)^d \prod_k \delta_k^2}} - 0.5d \ln s - 0.5 s^{-1} \mathcal{X}_i)$ where $\mathcal{X}_i$ is the Mahalanobis distance $(W_U - W(i))^T \mathbf{\Delta} (W_U - W(i))$ between W_U and its ith nearest cluster center. Then, from $\frac{\partial L(s|X(t))}{\partial s} = 0$ we have

$$s = \frac{\sum_{W_U \in \mathbf{W}_U^X} \sum_{i=1}^K \mathcal{X}_i}{K|\mathbf{W}_U^X|d} \tag{5}$$

If the smooth parameter s is specified by the user rather than Eq. (5), the bandwidth will simply be calculated as $h = \hat{\delta}|N|^s$ [7]. In this case, the regression is only supervised rather than SSL. Rewriting the regression function in Eq. (1), we have the final regression function

$$f(X(t)) = \sum_{W_i \in N} \frac{c_i Y_i \exp(-0.5 s^{-1}(X - W_i)^T \mathbf{\Delta} (X - W_i))}{\sum_{W_j \in N} c_j \exp(-0.5 s^{-1}(X - W_j)^T \mathbf{\Delta} (X - W_j))} \tag{6}$$

Note that in Eq. (5) only $L(s|\mathbf{W}_U^X)$ is maximized. To maximize $L(s|\mathcal{D}_L, \{X_U\})$, i.e. to adjust smooth parameter according to the complete dataset, the user will have to modify $\mathcal{D}_U$ to make it include the training dataset as $\mathcal{D}_L \subset \mathcal{D}_U$.

3. Software framework

3.1. Software architecture

The software is composed of 5 parts. (1) **'utils.py'**: Supporting utilities for csv file reading and Python dict operations. (2) **'isoxnn2.py' and 'gng2.py'**: GNG and SOINN algorithms. (3) **'ui_isoinn.py' and 'ui_gng.py')** Programming interfaces for GNG and SOINN. (4) **'gks.py'**: SSL-GKS implementation (5) **'reg_inn.py' and 'reg_gng.py'**: Regression programming interfaces.

3.2. Software functionalities

Main functionalities are implemented by 4 Python classes listed below.

1. class pygks.gks.GKS: By implementing Eq. (6), weights of clustering centers and their distribution is constructed for the regression purpose.
2. class pygks.ui_pygks.data_block: SOINN training and results visualization and topology features generation.
3. class pygks.reg_gng.GNGregressor: Supervised and semi-supervised regression with GNG and SSL-GKS.
4. class pygks.reg_inn.ISOINNregressor: Supervised and semi-supervised regression with SOINN and SSL-GKS.

4. Empirical results

There are two sets of experiments on 6 datasets. First, the typical SSL setting is employed, where comparison results are on varying labeled datasets with labeling percentages growing. Second, we move to a real application, namely the traffic flow prediction. The datasets are downloaded from the Caltrans Performance measurement Systems (PEMS) database [8]. 6 days of data beginning from Nov. 24th, 2014 are chosen as training datasets, and the data from Nov. 30th, 2014 as the testing datasets. The selected detector numbers are 'I880S', 'SR120E', 'US101N' and 'I5N'. 'wine' and

Please cite this article as: Z. Xiang et al., Gaussian kernel smooth regression with topology learning neural networks and Python implementation, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.01.051