UNIVERSIDAD LOYOLA ANDALUCÍA

Departament of Quantitative Methods

Diversity in Extreme Learning Machine ensembles

Carlos Perales-González[1]

[1]PhD student from Universidad Loyola Andalucía

Thesis directors: Francisco Fernández-Navarro, David Becerra-Alonso, Mariano Carbonero-Ruz

Overview

Machine learning

Machine learning: field of computer science that uses statistical techniques to give computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed [1].
Machine learning is the study of pattern recognition.

- Unsupervised: inferring a hidden structure from unlabeled data. I.e.: Clustering, autoencoders neural networks, ...
- Supervised: learning a function that maps from features to target based on training data. I.e.: SVM, Ridge regression, decision tree, ...
  1. Classification. Target is a category.
  2. Regression. Target is a real value.

Supervised classification

Algorithms minimize the error of classification. How?

- Linear methods.
- Nonlinear methods.
  1. Pure nonlinear methods. I.e.: Decision trees.
  2. Kernels. I.e.: RBF, polynomial.

How is the error function?

- Heuristic. Constrain programming, huge and hard loss functions. I.e.: genetic or greedy algorithms.
- Analytic. Simpler algorithms, based on convex optimization. I.e.: SVM, ELM.

Convex optimization is fast. But **strong assumption**: relation between features and target is convex. **Solution**: ensembles.

ELM I

Extreme Learning Machine for classification, a.k.a. Ridge classification [2], [3], [4].

$$f(\mathbf{x}) = \mathbf{h}'(\mathbf{x})\,\boldsymbol{\beta}, \tag{1}$$

where

- $\mathbf{x} \in \mathbb{R}^m$ is the vector of attributes, m is the dimension of the input space.
- $\boldsymbol{\beta} = (\boldsymbol{\beta}_j, j = 1, \ldots, J) \in \mathbb{R}^{d \times J}$ is ELM matrix.
- $\mathbf{h} : \mathbb{R}^m \to \mathbb{R}^d$ is the mapping function and d is the number of hidden nodes (the dimension of the transformed space).

ELM II

Learning problem: Let us also denote $\mathbf{H}$ as $\mathbf{H} = \left(h'\left(\mathbf{x}_i\right), \, i = 1, \ldots, n\right) \in \mathbb{R}^{n \times d}$ as the transformation of the training set and $\mathbf{Y} \in \mathbb{R}^{n \times J}$ is the labels "1-of-J" encoded.

$$\min_{\beta \in \mathbb{R}^{d \times J}} \left(\|\beta\|^2 + C \|\mathbf{H}\beta - \mathbf{Y}\|^2 \right), \tag{2}$$

where $C \in \mathbb{R}^+$ is a cross-validated hyper-parameter.

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}'\mathbf{H} \right)^{-1} \mathbf{H}'\mathbf{Y} \tag{3}$$

ELM III

It uses:

- Matrix programming.
- Use Tikhonov regularization.

Machine learning problems are not always solvable. To avoid inverse matrix problems, regularization term appears.
Different mapping functions h haven been proposed

- Single Hidden Layer ELM, or Neural ELM.
- Kernel ELM, or Kernel Ridge classification.

Proposed solution to avoid convex assumption: ensembles.

Bagging and Boosting

Bagging (bootstrap aggregating) is a learning method for generating several versions of a base learner by selecting some subsets from the training set and using these as new learning sets [5]. Random, different classifiers.

Boosting is a family of machine learning meta-algorithms which focus on combining base learners over several iterations and generate a weighted majority hypothesis [6].

Our proposal I

Explicit diverse ensembles for classification. This was sent to the International Conference on Hybrid Artificial Intelligent Systems (HAIS) 2018 and accepted.

$$\min_{\beta^l \in \mathbb{R}^{d \times J}} \frac{1}{2} \left(\left\| \beta^l \right\|^2 + C \left\| \mathbf{H}\beta^l - \mathbf{Y} \right\|^2 + \left(D + \frac{n}{s} \right) \sum_{j=1}^{J} \sum_{k=1}^{l-1} \left\langle \beta_j^l, \mathbf{u}_j^k \right\rangle^2 \right) \tag{4}$$

where:

- $\mathbf{u}^k \in \mathbb{R}^{d \times J}$ is the column-by-column normalized β^k from the iteration k of the ensemble.
- $D > 0$ is a hyper-parameter like C.

Our proposal II

Hence, β_j^l could be obtained analytically as:

$$\beta_j^l = \left(\frac{\mathbf{I}}{C} + \mathbf{H}'\mathbf{H} + \frac{1}{C}\left(D + \frac{n}{s}\right)\mathbf{M}_j^l \right)^{-1} \mathbf{H}'\mathbf{Y}_j \quad j = 1, \ldots, J \quad (5)$$

where $\mathbf{M}_j^l$ is defined as

$$\mathbf{M}_j^l \equiv \sum_{k=1}^{l-1} \mathbf{u}_j^k \left(\mathbf{u}_j^k\right)' \quad (6)$$

Results I

	Accuracy (Acc)			
	DELM	AELM	BRELM	NCELM
car	**0.929711**	0.834618	0.901805	*0.905111*
winequality-red	**0.853687**	*0.840085*	0.839670	0.837363
ERA	**0.829479**	0.822201	0.828019	*0.828428*
LEV	**0.836345**	0.786404	0.792371	*0.798220*
SWD	**0.787940**	*0.764487*	0.759893	0.760442
newthyroid	**0.932035**	0.817172	0.812035	*0.819509*
automobile	**0.867376**	0.834618	0.841636	*0.846499*
squash-stored	0.694286	**0.751429**	0.694063	*0.711937*
squash-unstored	*0.814286*	**0.830952**	0.813810	0.812381
pasture	**0.833333**	0.766667	0.811111	*0.826667*

Results II

	Diversity (d)			
	DELM	AELM	BRELM	NCELM
car	**0.999206**	0.180213	0.176621	*0.181212*
winequality-red	**0.926890**	0.152451	0.124054	*0.185804*
ERA	**0.968917**	0.138991	0.143748	*0.156551*
LEV	**0.992860**	0.098013	0.089886	*0.133830*
SWD	**0.98095**3	0.130116	*0.138222*	0.137884
newthyroid	**0.886023**	0.043061	0.040141	*0.057340*
automobile	**0.932272**	0.314529	0.311612	*0.317588*
squash-stored	**0.668662**	0.216839	0.181023	*0.217834*
squash-unstored	**0.568838**	0.130116	0.145780	*0.155101*
pasture	0.081884	*0.181297*	0.175300	**0.187400**

Motivation

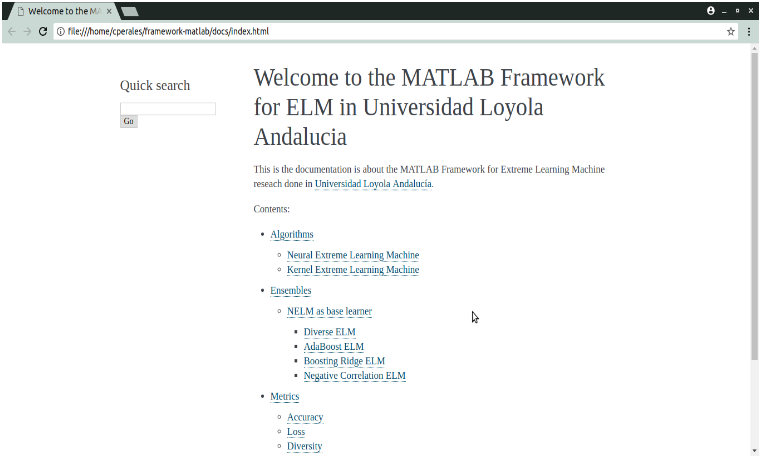
- Solve convex assumption, looking for local convexities.
- Sensitivity analysis. It helps to establish a ranking of good solutions.
- Diverse solutions are looked for in a explicit way.

Work in progress

Improving our proposal to HAIS 2018.

- More ensembles to compare against.
- More data sets.
- Minimizing runtime.
- Sensitivity analysis.

Framework in MATLAB (1)



Quick search

Go

Welcome to the MATLAB Framework for ELM in Universidad Loyola Andalucia

This is the documentation is about the MATLAB Framework for Extreme Learning Machine reseach done in Universidad Loyola Andalucía.

Contents:

- Algorithms
  - Neural Extreme Learning Machine
  - Kernel Extreme Learning Machine
- Ensembles
  - NELM as base learner
    - Diverse ELM
    - AdaBoost ELM
    - Boosting Ridge ELM
    - Negative Correlation ELM
- Metrics
  - Accuracy
  - Loss
  - Diversity

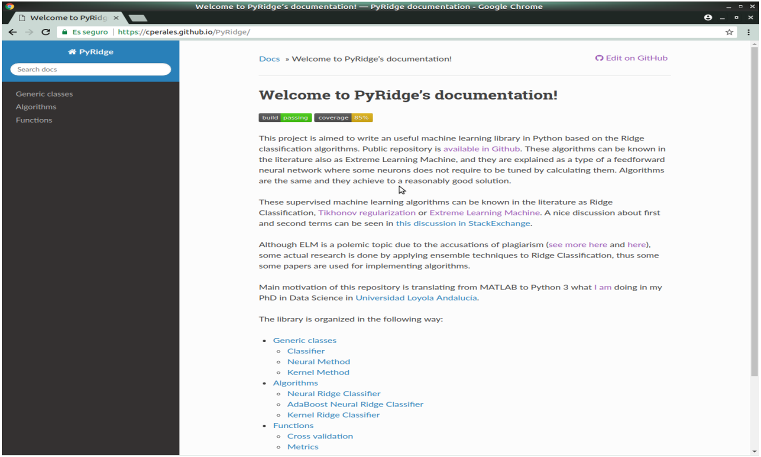
Framework in MATLAB (2)

Editor - /home/cperales/framework-matlab/algorithm/NELM.m

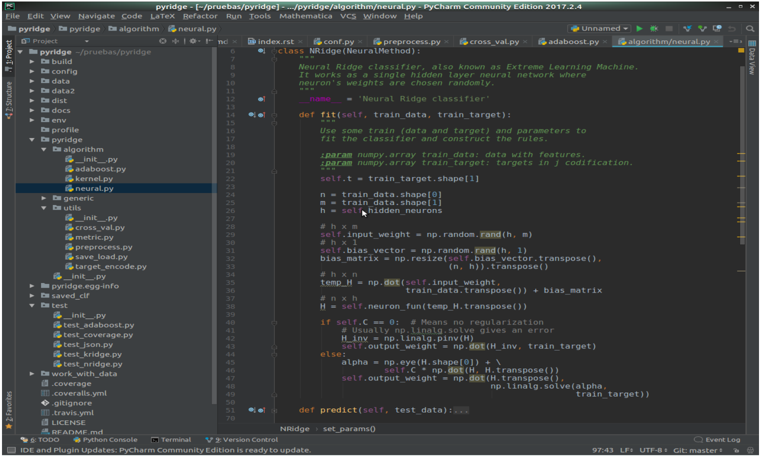
```matlab
classdef NELM < NMethod
    % Neural Extreme Learning Machine with a hidden layer

    properties

    end

    methods

        function obj = fit(obj, trainData, trainTarg, parameters)
            % :trainData: Data matrix n x m, with n instances and m features.
            % :trainTarg: Target J-encoded matrix n x j.
            % :parameters: Structure with the cross validated hyperparameters.

                    % Absorb the parameter structure
                    hiddenNeurons = parameters.hiddenNeurons;
                    if hiddenNeurons == 0
                        hiddenNeurons = size(trainTarg, 2);
                    end
                    C = parameters.C;

                    n = size(trainData,1);
                    m = size(trainData,2);
                    h = hiddenNeurons;

                    InputWeight = rand(h, m);
                    BiasVector = rand(h, 1);
                    ind = ones(n, 1);
                    BiasMatrix = BiasVector(:,ind);
                    tempH = InputWeight * trainData' + BiasMatrix;
                    H = obj.neuronFun(tempH');  % n x h
                    if C == 0  % No regularization
                        OutputWeight = H \ trainTarg; % h x t
                    else
                        alpha = (eye(size(H, 1)) ./ C + H * H') \ trainTarg;
                        OutputWeight = H' * alpha;  % h x t
                    end

                    % Store Information
                    obj.C = C;
                    obj.hiddenNeurons = hiddenNeurons;
                    obj.InputWeight = InputWeight;
                    obj.BiasVector = BiasVector;
                    obj.OutputWeight = OutputWeight;

        end

        function [testTargets] = predict(obj, testPatterns)
            % :testPattern: Data matrix n x m, with n instances to predict and m features.
```

Framework in Python (1)

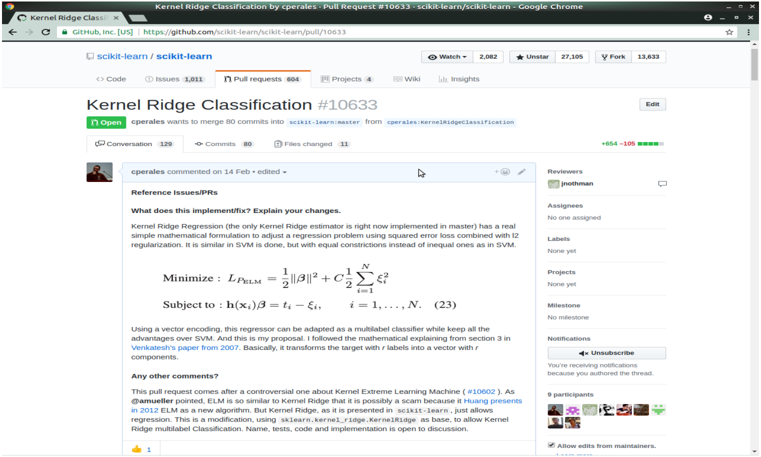
https://github.com/cperales/PyRidge



Framework in Python (2)



scikit-learn / sklearn



References I

- A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.
- A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.
- G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

References II

G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2012.

L. Bbeiman, "Bagging Predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

Y. Freund and R. E. Schapire, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.

END

THANK YOU