Introduction to regression: LWPR Locally weighted projection regression

Luis Montesano

MLSS 2011, September 8

Abstract

This lab is intended for people who have recently started to work in ML. We will briefly review linear regression and partial square (PLS). Next, we will describe Locally Weighted Projection Regression (LWPR), an efficient method for non-linear regression based on local linear models and reduction of dimensionality. We will use the the MATLAB implementation available at http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/

1 Linear Regression Models

Regression is a supervised learning problem whose objective is to predict the value of an output variable $\mathbf{y} \in \mathbb{R}^m$ from an input variable $\mathbf{x} \in \mathbb{R}^n$. The simplest type of regression is linear regression where the prediction is obtained from a linear combination of the input variable

$$\mathbf{y} = \mathbf{w}^T \mathbf{x}^* \tag{1}$$

with $\mathbf{x}^* = (\mathbf{x}^T 1)$.

Given a set of N examples $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, the Least Square solution to this problem has the following closed form solution

$$\mathbf{w}^{ml} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} \tag{2}$$

with

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^* \\ \mathbf{x}_2^* \\ \cdots \\ \mathbf{x}_N^* \end{pmatrix}, \qquad \mathbf{Y} = \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \cdots \\ \mathbf{y}_n \end{pmatrix} \tag{3}$$

This model is equivalent to the maximum likelihood solution of a linear model with Gaussian noise.

2 Partial Least Squares

Partial Least Squares (PLS) is a bilinear regression technique similar to principal component regression. The latter is often used when input data features are (close to) co-linear to regress on the principal components space of the input variables. PLS, instead, jointly projects the input and output spaces to find the direction in $\mathbf{X}$ that explains the maximum variance direction in $\mathbf{Y}$. PLS is useful when the matrix of predictors has more variables than observations, and when there is multicollinearity among $\mathbf{X}$ values.

- Generate a dataset with redundant input features (see demoPLS.m for a 4D example. embeded in 2D)

  % Generate input features

```
X1 = (rand(n,2)-.5)*2;
R = randn(d);
R = R'*R;
R = orth(R);
X = [X1 zeros(n,d-2)]*R;
% Generate output response (linear or non-linear)
y= ...
```

- Fit a linear regression (implement it yourself based on Eq. 2). What should be the output? Why?
- Use PCA to select input features (e.g. use princomp) and check the results obtained with different numbers of components.
- Use the *regress* function in Matlab.
- Use PLS software in Matlab (use the function to select the appropriate number of components Compute_NcompPLS). Compare the prediction error, the parameters of the three different methods.

3 Locally Weighted Projected Regression

In many real applications, data cannot be modelled with a linear model and more complex models have to be used. LWPR is one such a model where a combination of weighted local linear models are learned from data to approximate non-linear functions. The prediction of LWPR takes the form of a weighted combination of local models

$$\hat{y} = \frac{\sum_{k=1}^{K} w_k \hat{y}_k}{\sum_{k=1}^{K} w_k} \tag{4}$$

where $\hat{y}_k$ is the prediction of local model k and w_k is the weight associated to this model. Local models are linear models spanned along the directions provided by PLS

$$y_k = \beta_k \mathbf{z} \tag{5}$$

where $\mathbf{z}$ represents the projected inputs. The weight w_k represents the region where the model k is active, also known as recepteive field. It is computed using a Gaussian kernel paremeterized by the distance metric $\mathbf{D}_k$

$$w_k = \exp(-0.5(\mathbf{x} - \mu_k)^T \mathbf{D}_k (\mathbf{x} - \mu_k)). \tag{6}$$

Therefore, the parameters of the model include the number of the models K, the local model parameters β_k and the distance metric $\mathbf{D}_k$. In order to learn these parameters from data, a cross-correlation scheme is used to minimize the predicted residual sums of squares

$$J = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} \frac{w_k (y - \hat{y}_i)^2}{(1 - w_i \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i)} + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2 \tag{7}$$

using stochstic gradient descent.

4 The LWPR library

The purpose of the lab is to get familiar with the LWPR library described in

http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/

Details on the LWPR method as well as on practical issues can be found at this web page and may help you during the lab. We will use the Matlab version of the LWPR library. You can download it directly from the authors webpage.

4.1 Matlab code

- We are first going to run the 1D example of the LWPR library (the script is very well documented). It (1D_example.m)
  - Generates a 1D dataset with training and test points on a regular grid over the
  - Create the LWPR structure,
  - Tunes the input parameters for the dataset,
    (look at: http://www.ipab.inf.ed.ac.uk/slmc/software/lwpr/lwpr_doc.pdf)
    1. Specifies number of dimension,
    2. Sets the initial distance matrix,
    3. Sets the component wise normalization,
    4. Sets the Learning rate
    5. Runs the learning loop K times. At each time, loop over the whole dataset feeding each example at the time to the function *lwpr_update*
    6. Predicts values at the test points (*lwpr_predict*)
- Modify the parameters and evaluate the impact on the results
- Inspect the 2D examples
- Now create your own script to run the previous 4D example with redundant inputs for a linear and non-linear 1D response
- EXTRA WORK: Implement and online version of the LWPR algorithm using stochastic gradient descent. Compute the initial solution based on a subset of the trainining data and next incorporate the rest of the training instances.
- EXTRA WORK: Feed the data to the GP library with a simple kernel (Gaussian) and a simple likelihood (Gaussian). Compare the results. To do this, use script trySimpleGP.m that uses the GP software available

  http://www.gaussianprocess.org/gpml/

  For more hands-on work with GPs there is a specific laboratory on kernel methods next week.

4.2 Robot data

We next use the robot dataset used to illustrate both LWPR and GPs.

- The dataset is available at http://www.gaussianprocess.org/gpml/data/ in two separated matlab .mat files with the training and test data.
- Use LWPR and GPs to predict one of the output torques. Compare the prediction error for both methods after tunning the parameters.

If you feel like, test any dataset you may have or download from the web. A good place to look for data is http://archive.ics.uci.edu/ml/.

References

[1] Sethu Vijayakumar, Aaron D'Souza and Stefan Schaal. Incremental Online Learning in High Dimensions. Neural Computation, vol. 17, no. 12, pp. 2602-2634 (2005).

[2] C.E. Rasmussen and C. K. I. Williams, Gaussian processes for machine learning, MIT Press 2006.