Data Analysis and Model Classification
Guidesheet VIII: Regularized regression

Fumiaki Iwane Ping-Keng Jao Bastien Orset Julien Rechenmann
Ricardo Chavarriaga José del R. Millán

November 19, 2018

Introduction

At this point of the course, you should already learned that problems can occur when trying to train models with many parameters with a few samples. In this guidesheet, you will learn techniques to train regression models, specifically using regularization terms based on L_1 and L_2 normalization constraints. As last week, we will use the mean square error (MSE, `immse` in MATLAB) to evaluate our models. For this we will use only 5% of the samples for training and the rest for testing. This is not a smart design choice, but we will use it here for pedagogical purposes to clearly understand the concept of regularization and its effects when data is scarce.

The L_2 norm is nothing else than the euclidian norm, defined as $||x||_2 = \sqrt{\sum_{i=1}^{p} x_i^2}$, where p is the dimension of the vector $x = |x_1, \dots x_p|^T$. The L_1 norm is the sum of the absolute values of each elements of the vector x: $||x||_1 = \sum_{i=1}^{p} |x_i|$. You can use the MATLAB commands `norm(x,1)` and `norm(x,2)` to compute these L_1 and L_2 norms.

Regression

The parameters of the standard regression are found through error minimization; specifically by solving $\min_\beta ||y - X\beta - \beta_0||_2$ minimization where y are the labels (`PosX` and `PosY`), X is the training data and β is the vector of regression coefficients, with β_0 being the *intercept* (bias).

Hands on

The regressed model follows perfectly PosX and PosY. The error on the train is e^-32 while it is big on the test

- Apply regression without PCA. Use linear regression with an intercept (`X=[I FM]`, where `I` is a vector of 1s and `FM` the feature matrix/training data) and compute the test error (as you did last week). Is it a good mean square error? Plot the output compared to the labels.
- How many parameters do you have to estimate when using regression in this problem? Compare the number of parameters to the number of samples available for training.

We have to estimate a number of parameters equal to the number of features. The #features is 960, the #samples is 643 (5% of the initial value). So we have 300 less samples compared to features.

LASSO

LASSO adds a L_1 regularization constraint to the definition of the regression coefficients:

$$\min_\beta ||y - X\beta - \beta_0||_2 \quad s.t. \quad \lambda||\beta||_1 \leq 1 \tag{1}$$

The constraint means that the L_1 norm of the weight vector (β) times λ must not exceed 1 (meaning that the norm must not exceed $\frac{1}{\lambda}$). This regularization type leads to sparse weight vectors (meaning that a lot of the coefficients will be zeros). The larger the λ, the sparser the resulting solution will become.

MATLAB provides the `lasso` function, and includes a cross-validation option to automatically select the λ that gives the smallest cross-validation (mean square) error. Have a (thorough) look at the MATLAB help for this function. Use a 10-fold cross-validation and set the *hyperparameter* `lambda = logspace(-10,0,15)` for example. The `lasso` function takes care of adding the intercept so you can feed it the training data directly. The function outputs two variables: `B` and `FitInfo`. `B` is a matrix containing the regression weights for all λs. The `FitInfo` structure contains the intercept and the mean square error (MSE) for each λ.

Hands on

- Apply LASSO on the training data. For each λ value, what is the number of non zero β weights? How does this number change as λ increases?
- Plot the cross-validation MSE for each λ and interpret it. For better visualization you can use a logarithmic scale using `semilogx(lambda,FitInfo.MSE)`.
- For the λ value corresponding to the best MSE, use the corresponding β and *intercept* to regress the test data (`PosX`, `PosY`). Plot the data and compute the test MSE. Happy?

Elastic nets

LASSO has two limitations: firstly, when $N < p$, it only sets a maximum of N out of p weights to a non-zero value (N: number of samples and p: number of features). This is not so much of a problem in our example, since p is not so much larger than N, but can be problematic, when $N << p$. The second limitation is much more problematic in our case: when groups of features are correlated, LASSO will only select one of these features since it optimizes sparsity. To cope with these limitations, we can add a second constrain on the L_2 norm. This method is called Elastic nets. The two constrains will be weighted by an additional factor $0 < \alpha \leq 1$:

$$c(\beta) = \alpha||\beta||_1 + \frac{1-\alpha}{2}||\beta||_2^2 \tag{2}$$

You can see that when $\alpha = 1$, elastic nets reduce to LASSO. Use the same parameters as in the previous section, except for $\alpha = 0.5$ instead of 1. Of course, α is also a *hyperparameter*. What is the appropriate way to set it up?

Hands on

- Apply an elastic net on the training data, choose the relevant arguments for the `lasso` function. Use For each λ value, what is the number of non zero β weights? How does it evolve with λ increasing. For each λ value, compare the number of non-zero weights between LASSO and elastic nets.
- For the λ value corresponding to the best MSE, use the corresponding β and *intercept* to regress the test data (`PosX`, `PosY`). Plot the data and compute the test MSE. Again, compare with the last section.
- Optimize your hyperparameters. Will you use a cross-validation or a single training-test split? How will you estimate performance of a model which optimizes hyperparameters?