ICT for Health
Laboratory # 2
Regression on Parkinson’s dataset, again

Monica Visintin

Politecnico di Torino

2024/25

Description

We want to regress again Total UPDRS as in laboratory # 1, and we will only use LLS algorithm. The difference is that we want to use only K nearest neighbours to train the model. We will use the validation dataset to find the "optimum" value of K.

Rationale

"Each function is linear if you zoom enough".
You know that Taylor's series expansion truncated at the first term (the linear part)

$$f(\mathbf{x}) \simeq f(\mathbf{x}_0) + \nabla_f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

is valid only for points $\mathbf{x}$ close to $\mathbf{x}_0$.
We want to apply the same principle to linear regression, we simply need to define "closeness" in an easy way. The easiest way is to use only K closest points to $\mathbf{x}_0$ to find the optimum weight vector $\mathbf{w}$ so that $f(\mathbf{x}_0) \simeq \mathbf{x}_0^T\mathbf{w}$. Here $\mathbf{x}_0 \in \mathbb{R}^F$ is the test point that stores the regressors and $f(\mathbf{x}_0)$ is the corresponding regressand, matrix $\mathbf{X}_{train}$ has K rows and F columns (instead of having N_{Tr} rows). Note that each of the test points will have its own weight vector $\mathbf{w}$ (so it would be more correct to call it $\mathbf{w}(\mathbf{x}_0)$).

Data preparation [1]

1. As in Lab #1, load the Parkinson's disease dataset, remove the unwanted features, shuffle the data, get the training, and test datasets. Use the following features as regressors: 'sex', 'test_time', 'motor_UPDRS', 'Jitter(%)', 'Jitter(Abs)', 'Jitter:RAP', 'Jitter:PPQ5', 'Jitter:DDP', 'Shimmer', 'Shimmer(dB)', 'Shimmer:APQ3', 'Shimmer:APQ5', 'Shimmer:APQ11', 'Shimmer:DDA', 'NHR', 'HNR', 'RPDE', 'DFA', 'PPE' (they are 19), and use 'total_UPDRS' as regressand.
2. As a difference with respect to Lab # 1, the data must be divided into training, **validation** and test datasets, according to the percentages: 50%, 25%, 25%. Note that with such a division, you technically have 75% of training data and 25% of test data; the training data are further divided into validation data (33%) and true training data (66%).
3. As in Lab # 1 find mean and standard deviation of each feature in the true training dataset (as if you did not know validation data and test data), and normalize the entire dataset using these means and standard deviations.
4. As in Lab # 1 extract the regressand, i.e. total UPDRS, for the training, validation and test datasets.

Data preparation [2]

5. At the end you have a matrix `X_train_norm`, a vector `y_train_norm`, a matrix `X_test_norm`, a vector `y_test_norm`, a matrix `X_val_norm` a vector `y_val_norm`, all normalized, with obvious meanings. For simplicity, use Numpy NdArrays, not Pandas dataframes

Fixed K

1. Set a suitable value for K (for example $k = 20$). For each $\mathbf{x}$ in the validation dataset:
   1.1 Find the distance (or square distance) between $\mathbf{x}$ and each of the N_{tr} points of the training dataset (use normal Numpy methods (`x**2` for the element-wise square of vector`x` and `np.sum`).
   1.2 Select the K points of the training dataset with the smallest distance from $\mathbf{x}$ (available methods are `np.sort` or `np.argsort`)
   1.3 Generate matrix $\mathbf{A}$ (K rows and F columns); generate column $\mathbf{y}$ (true values of total UPDRS for the K selected training points), generate

   $$\hat{\mathbf{w}} = \left(\mathbf{A}^T\mathbf{A} + \epsilon\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{y}$$

   where $\epsilon = 10^{-8}$ and $\mathbf{I}$ is the identity matrix (which shape?). Term $\epsilon\mathbf{I}$ is added to guarantee that it is possible to invert the matrix (see ridge regression).
   1.4 Evaluate the estimated UPDRS for the validation point $\mathbf{x}$ as

   $$\hat{y}(\mathbf{x}) = \mathbf{x}^T\hat{\mathbf{w}}$$

   and the estimation error as

   $$e(\mathbf{x}) = y(\mathbf{x}) - \hat{y}(\mathbf{x})$$

   where $y(\mathbf{x})$ is the true UPDRS value for $\mathbf{x}$. In the above linear algebra equations $\mathbf{x}$ is assumed to be a column vector.
2. Evaluate the mean square error for the validation dataset.

Optimization of K

1. Try and use the script as in the previous slide with some "extreme" values of K and store on a piece of paper the validation mean square error for the selected values of K.
2. Decide possible minimum and maximum values of K: K_{min} and K_{max}, respectively.
3. Generate an external loop letting K run from K_{min} to K_{max} with a chosen appropriate step.
4. For each value of K store the validation mean square error.
5. Plot the the validation mean square error versus K, and find K_{opt} that gives the minimum validation mean square error.

Test phase

1. Repeat what you did for the validation dataset, using this time the test dataset, using K_opt.
2. As in Lab # 1, for the test dataset measure the mean value, the standard deviation and the mean square value of the error, the correlation coefficient, the coefficient of determination R^2, the regression line, the estimation error histogram.
3. In the script for Lab # 2, add LLS regression using the training dataset made of 75% of the total points, so the test points are exactly the same, and measure the mean value, the standard deviation and the mean square value of the error, the correlation coefficient, the coefficient of determination R^2, the regression line, the estimation error histogram.
4. Compare the results you get with the standard LLS linear regression and the K nearest neighbor-LLS linear regression (note: if the test points are not exactly the same, the results cannot be compared).