Linear Regression on a Combined Cycle Power Plant (CCPP) data

Dataset description

The dataset contains 5281 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) of the plant.

A combined cycle power plant (CCPP) is composed of gas turbines (GT), steam turbines (ST) and heat recovery steam generators. In a CCPP, the electricity is generated by gas and steam turbines, which are combined in one cycle, and is transferred from one turbine to another. While the Vacuum has effect on the Steam Turbine, the other three of the ambient variables effect the GT performance.

Split data in training, validation and test sets

Given m total data, keep m_t data as training data, $m_{val} := m_{tv} - m_t$ as validation data and $m_{test} := m - m_{val} - m_t = m - m_{tv}$. For instance one can take $m_t = m/3$ of the data as training, $m_{val} = m/3$ validation and $m_{test} = m/3$ as testing. Let us define as define

- S_t the training data set
- S_{val} the validation data set
- S_{test} the testing data set

The reason for this splitting is as follows:

TRAINING DATA: The training data are used to compute the empirical loss

$$L_S(h) = \frac{1}{m_t} \sum_{z_i \in S_t} \ell(h, z_i) \tag{1}$$

which is used to estimate $\hat{h}$ (ERM) in a given model class $\mathcal{H}$, i.e.:

$$\hat{h} = \arg\min_{h \in \mathcal{H}} L_S(h) \tag{2}$$

VALIDATION DATA: When different model classes are present (e.g. of different complexity, such as linear regression which uses a different number d of regressors $x_1, \ldots, x_d$), one has to choose which one is the "best" complexity.
Let $\mathcal{H}_d$ be the space of models as a function of the complexity d and let

$$\hat{h}_d = \arg\min_{h \in \mathcal{H}_d} L_S(h) \tag{3}$$

One can estimate the generalization error for model $\hat{h}_d$ as follows:

$$L_{\mathcal{D}}(\hat{h}_d) \simeq \frac{1}{m_{val}} \sum_{z_i \in S_{val}} \ell(\hat{h}_d, z_i) \tag{4}$$

and then choose the complexity which achieves the best estimate of the generalization error

$$\hat{d} := \arg\min_d \frac{1}{m_{val}} \sum_{z_i \in S_{val}} \ell(\hat{h}_d, z_i) \tag{5}$$

TESTING DATA: Last, the test data set can be used to estimate the performance of the final estimated model
$\hat{h}_{\hat{d}}$, using:

$$L_{\mathcal{D}}(\hat{h}_{\hat{d}}) \simeq \frac{1}{m_{test}} \sum_{z_i \in S_{test}} \ell(\hat{h}_{\hat{d}}, z_i) \tag{6}$$

Let's split the data in training, validation and test sets (as $m_t = m_{val} = \lfloor \frac{m}{3} \rfloor$, $m_{test} = m - m_t - m_{val}$)

Data Normalization

It is common practice in Statistics and Machine Learning to scale the data (= each variable) so that it is centered (zero mean) and has standard deviation equal to 1. This helps in terms of numerical conditioning for the (inverse) problems of estimating the model (the coefficients of the linear regression in this case), as well as to give the same scale to all the coefficients.

NOTE: Data normalization is achieved by means of a preprocessing function: the scaler.
If you want to normalize you data to have zero (empirical) mean and unit (empirical) variance the scaler will simply be a function that takes each data point, remove the empirical mean and then divide the result by the empirical variance. In this way you get a new normalized dataset.

NORMALIZATION and TRAINING PROCEDURE:

1. You are given a training dataset, you need to find the proper scaler in order to preprocess it (e.g. use empirical mean and variance, if you want to apply the normalization by mean and variance, you could also use a different normalization for example based on the max and min)
2. Apply the scaler to get the preprocessed training dataset
3. Fit a model in your preferred class on the normalized dataset
4. Now you need to test your trained model on new data in order to get an estimate of the generalization error. Note you have trained you model using normalized data, this means that the model expects data that have been normalized (according to the same preprocessing step as the training data). This means that you need to apply the same scaler (that exploits the empirical mean and variance computed from the training dataset) to the test dataset (the same holds true for the validation dataset).

See next cell (note that the mean and std for the validation and test sets are not 0 and 1 respectively!)

Remark: Both input x and output y can be normalized!

Model Training

The model is trained (= estimated) minimizing the empirical error

$$L_S(h) := \frac{1}{m_t} \sum_{z_i \in S_t} \ell(h, z_i) \tag{7}$$

When the loss function is the quadratic loss

$$\ell(h,z) := (y - h(x))^2 \tag{8}$$

we define the Residual Sum of Squares (RSS) as

$$RSS(h) := \sum_{z_i \in S_t} \ell(h, z_i) = \sum_{z_i \in S_t} (y_i - h(x_i))^2 \tag{9}$$

so that the training error becomes

$$L_S(h) = \frac{RSS(h)}{m_t} \tag{10}$$

We recal that, for linear models with scalar output we have $h(x) =< w, x >$ and the Empirical error $L_S(h)$ can be written
in terms of the vector of parameters w in the form

$$L_S(w) = \frac{1}{m_t} \|Y - Xw\|^2 \tag{11}$$

where Y and X are the matrices whose $i-$th row are, respectively, the output data y_i and the input vectors $x_i^\top$.

The least squares solution is given by the expression

$$\hat{w} = \arg\min_w L_S(w) = (X^\top X)^{-1} X^\top Y \tag{12}$$

When the matrix $X^\top X$ is not invertible, the solution can be computed using the Moore-Penrose pseudoinverse $(X^\top X)^\dagger$ of $(X^\top X)$

$$\hat{w} = (X^\top X)^\dagger X^\top Y \tag{13}$$

The Moore-Penrose pseudoinverse $A^\dagger$ of a matrix $A \in \mathbb{R}^{m \times n}$ can be expressed in terms of the Singular Value Decomposition (SVD) as follows:

Let $A \in \mathbb{R}^{m \times n}$ be of rank $r \leq \min(n, m)$ and let

$$A = USV^\top \tag{14}$$

be the singular value decomposition of A where

$$S = \text{diag}\{s_1, s_2, .., s_r\} \tag{15}$$

Then

$$A^\dagger = VS^{-1}U^\top \tag{16}$$

In practice some of the singular values may be very small (e.g. $< 1e - 12$). Therefore it makes sense to
first approximate the matrix A truncating the SVD and then using the pseudoinverse formula.

More specifically, let us postulate that, given a threshold T_h (e.g $= 1e - 12$), we have $\sigma_i < T_h$, for $i = \hat{r} + 1, .., r$. Then we can approximate (by SVD truncation) A using:

$$A = USV^\top = U\,\text{diag}\{s_1, s_2, .., s_r\}\,V^\top \simeq \hat{A}_r = U\,\text{diag}\{s_1, s_2, .., s_{\hat{r}}, 0, .., 0\}\,V^\top \tag{17}$$

So that

$$A^\dagger \simeq \hat{A}_r^\dagger := V\,\text{diag}\{1/s_1, 1/s_2, .., 1/s_{\hat{r}}, 0, .., 0\}\,U^\top \tag{18}$$

TO DO [2 - 5]: compute the linear regression coefficients directly (using pseudo inverse) and using np.linalg.lstsq from scikitlearn.

Data prediction

Compute the output predictions on both training and test set and compute the Residual Sum of Sqaures (RSS) defined above, the Emprical Loss and the quantity R^2 where

$$R^2 = 1 - \frac{\sum_{z_i \in S_t}(y_i - \hat{y}_i(x_i))^2}{\sum_{z_i \in S_t}(y_i - \bar{y})^2} \qquad \bar{y} = \frac{1}{m_t}\sum_{z_i \in S_t} y_i \tag{19}$$

R^2 is the so-called "Coefficient of determination" (COD).

Note: The above COD is computed on the training dataset, the formula for the test dataset is similar (mutatis mutandis).

TO DO 7: Answer in ths cell (you do not need more than 5-7 lines):

1- What is the intuition behind the definition of COD, would you prefer it high or low?

2- What results have been achieved using the linear model in this dataset (compare the metrics we used both on
training and test datasets). Is this what you expected? Is this a good model to describe the dataset?

3- Compare the output prediction and the distributions of errors (this plot might be the clearest),
what do you observe?

Confidence intervals for output predictions

In the following we will use d to represent the number of paraemters of the linear model (which may contain the bias b).

Having estimated the extended set of coefficients $\hat{w}$ (remember this is the outcome of a random variable), and given a new location x_0, the output prediction has the form

$$\hat{y}_0 := x_0^\top \hat{w} \tag{20}$$

and postulating $X^T X$ is invertible and that

$$y_0 = x_0^\top w + \epsilon_0 \quad \epsilon_0 \sim \mathcal{N}(0, \sigma^2) \tag{21}$$

where $w := \mathbb{E}[\hat{w}]$ (due to the invertibility assumption). You can think the last assumption is not that far from being true: remember the bell shaped errors we plotted before (very very close to be a gaussian)! We would like to compute a confidence interval on the output prediction or equivalently for the estimation error

$$\tilde{y}_0 := \hat{y}_0 - y_0 \tag{22}$$

Using the equations above we have that

$$\tilde{y}_0 = x_0^\top(\hat{w} - w) - \epsilon_0 \tag{23}$$

where ϵ_0 and $\hat{w}$ are uncorrelated (since x_0 is a new input location $\hat{w}$ does not depend on x_0). It thus follows that (x_0 is a deterministic quantity)

$$\tilde{y}_0 \sim \mathcal{N}(0, x_0^\top Var\{\hat{w}\} x_0 + \sigma^2) \tag{24}$$

where

$$x_0^\top Var\{\hat{w}\} x_0 = \sigma^2 x_0^T (X^T X)^{-1} x_0 \tag{25}$$

Try to compute the variance of the random variable $\hat{w}$ (write its equation and then substitute $Y = Xw + E$ with $E \sim \mathcal{N}(0, \sigma^2 I)$).

Note: We do not know σ^2, we must estimate it from the data!

Using the results we have seen in class (extended to the case of unknown variance) we have that the interval

$$[-\Delta_0, +\Delta_0] \qquad \Delta_0 := \hat{\sigma}\, t_{1-\frac{\alpha}{2}}(m_t - d - 1)\sqrt{x_0^\top (X^\top X)^{-1} x_0 + 1} \tag{26}$$

where

$$t_{1-\frac{\alpha}{2}}(m_t - d - 1) \tag{27}$$

is the $1 - \frac{\alpha}{2}$ percentile of the Student's t-distribution, and

$$\hat{\sigma}^2 := \frac{1}{m_t - d - 1} \sum_{i \in S_t} (y_i - \hat{w}^\top x_i)^2. \tag{28}$$

The interval $[-\Delta_0, +\Delta_0]$ satisfies the condition

$$\mathbb{P}[\tilde{y}_0 \in [-\Delta_0, +\Delta_0]] = \mathbb{P}[\hat{y}_0 - y_0 \in [-\Delta_0, +\Delta_0]] = 1 - \alpha \tag{29}$$

Equivalently we can write

$$\mathbb{P}[y_0 \in [\hat{y}_0 - \Delta_0, \hat{y}_0 + \Delta_0]] = 1 - \alpha \tag{30}$$

Note that in the latter equation both y_0 AND the interval $[\hat{y}_0 - \Delta_0, \hat{y}_0 + \Delta_0]$ are random. The probability is to be understood with repect to both the choice of the traning set (Y, X) as well as on the choice of y_0.
With some abuse of terminology we shall say that $[\hat{y}_0 - \Delta_0, \hat{y}_0 + \Delta_0]$ is a confidence interval of level $1 - \alpha$ for y_0.

Why do we need the validation dataset?
Up to now we have not used it! But up to now we have not tried to find the best hyper-parameters for our linear model.
Now we are going to find the best subset of features (and therefore the best coefficients) according to RSS or COD evaluated using the validation.
In this very simple case we are going to use a brute force approach, we will try every single possible combination of features and evaluate the trained model using the validation dataset.

Since our model now depends on the validation dataset we should not use the validation dataset to evaluate the generalization capability of the new model, that is why we need a "new" dataset (that we have not used to optimize our model): the test dataset.

In this last part of the HW we will have a look at the running time of the methods we have written and compare them sklearn implementation.
We are going to test what how the running time is increasing as a function of the number of data and the number of features we use in our linear model.

TO DO 9: Answer in the next cell (you do not need more than 5-7 lines):

1- What is the effect on running time due to the change in dataset size? Why?

2- What is the effect on running time due to the change in features size? Why?

3- What are the most time consuming computations we need to perform to compute the LS solution?