# ICT for Health Laboratory
# Regression on Parkinson data using Gaussian processes – Lab #2

Monica Visintin

Politecnico di Torino


November 11th 2021

# Data preparation [1]

- As in Lab #1, load the Parkinson's disease dataset, remove the unwanted features, shuffle the data, get the training, and test datasets. To simplify, keep only the following features: total UPDRS (regressand), motor UPDRS, age, PPE (3 regressors).
- 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%.
- As in Lab # 1 find mean and standard deviation of each feature in the training dataset, and normalize the entire dataset using these means and standard deviations.
- As in Lab # 1 extract the regressand, i.e. total UPDRS, for the training, validation and test datasets.

# Data preparation [2]

- The order of these 3 initial operations can be changed as you like, but the final result must be that 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. Use Ndarrays, not Pandas dataframes.

# Goal [1]

We want to regress total UPDRS from the regressors (initially only motor UPDRS, age, PPE, later all the features used in Lab #1) using Gaussian processes:

$$Y = g(\mathbf{X}) + \nu$$

where $g(\mathbf{X})$ is the Gaussian process and $\nu$ the Gaussian measure error. The relevant formulas are the following:

- $N \times N$ covariance matrix $\mathbf{R}_{Y,N}$ in position $n, k$ has value:

$$\mathbf{R}_{Y,N}(n,k) = \theta \exp\left(-\frac{\|\mathbf{x}_n - \mathbf{x}_k\|^2}{2r^2}\right) + \sigma_\nu^2 \delta_{n,k}, \quad n, k \in [1, N]$$

- $N \times N$ covariance matrix $\mathbf{R}_{Y,N}$ is written as

$$\mathbf{R}_{Y,N} = \begin{bmatrix} \mathbf{R}_{Y,N-1} & \mathbf{k} \\ \mathbf{k}^T & d \end{bmatrix}$$

# Goal [2]

- The pdf of $Y_N$ for regressors $X = x_N$, given the measured values $\mathbf{y} = [y_1, \ldots, y_{N-1}]$ is

$$f_{Y_N|\mathbf{y}}(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$

$$\mu = \hat{y}_N = \mathbf{k}^T \mathbf{R}_{Y,N-1}^{-1} \mathbf{y}$$

$$\sigma^2 = d - \mathbf{k}^T \mathbf{R}_{Y,N-1}^{-1} \mathbf{k}$$

- In the above equations, couples $(\mathbf{x}_n, y_n)$ for $n = 1, \ldots, N-1$ belong to the training dataset, couple $(\mathbf{x}_N, Y_N)$ belongs to the test or validation datasets. Note that $Y_N$ is the random variable whose pdf we want to estimate, $y_N$ is its true value, $\hat{y}_N$ is its estimated value.
- **Hyperparameters $\theta, r, \sigma_\nu^2$ must be found using the validation dataset, i.e. using $(\mathbf{x}_N, Y_N)$ from the validation dataset.**

# Initial procedure to check that the system works [1]

In the model, initially set the autocorrelation hyperparameters $r^2 = 3$, $\theta = 1$ and $\sigma_\nu^2 = 0.001$.

1. Pick all the rows of `X_val` and perform the following steps for each of the rows. Let `x` be the $k$-th row of `X_val` and set `N=10` (`N` has the same meaning as $N$ in the slides):

```
x=X_val_norm[k,:]
```

2. Find the square distance between vector `x` and each of the rows of `X_train`, call the vector with the square distances `dist2`

```
A=X_train_norm-np.ones((Ntr,1))*x
dist2=np.sum(A**2,axis=1)
```

   (`Ntr` is the number of rows in `X_train`).

3. Sort `dist2` and take the indexes of the smallest `N-1` square distances.

```
ii=np.argsort(dist2)
ii=ii[0:N-1];
```

# Initial procedure to check that the system works [2]

Generate a submatrix of `X_train` that stores the `N-1` row vectors closer to `x` and add `x` as the last row (you get a matrix `Z` with `N` rows and `F` features).

```
6       refX=X_train_norm[ii,:]
7       Z=np.vstack((refX,x))
```

The first `N-1` datapoints you have in `Z` are the training datapoints that are more informative when you want to find the total UPDRS corresponding to vector `x`.

4. Find the square distance between each couple of row vectors in `Z`: you must get a real symmetric matrix `D` with size `N` $\times$ `N`, with zeros in the main diagonal. Element `D[i,k]` must be the square distance between the vector in row `i` and the vector in row `k`. Note that

$$\|\mathbf{x}_k - \mathbf{x}_i\|^2 = \|\mathbf{x}_k\|^2 + \|\mathbf{x}_i\|^2 - 2\mathbf{x}_k^T\mathbf{x}_i$$

# Initial procedure to check that the system works [3]

```
8     sc=np.dot(Z,Z.T)# dot products
9     e=np.diagonal(sc).reshape(N,1)# square norms
10    D=e+e.T-2*sc# matrix with the square distances
```

5. Generate the covariance matrix `R_N` (Gaussian formula that depends on the square distances just found); also this matrix is `N` $\times$ `N`:

```
11    R_N=np.exp(-D/2/r2)+s2*np.identity(N)#covariance matrix
```

where `s2` corresponds to $\sigma_\nu^2$, initially set to 0.001, and `r2` corresponds to $r^2$, initially set to 3.

6. Define submatrix `R_Nm1` (top left submatrix of `R_N`, size $(N-1) \times (N-1)$), vector `K`, scalar `d` (as in the slides):

```
12    R_Nm1=R_N[0:N-1,0:N-1]
13    K=R_N[0:N-1,N-1]
14    d=R_N[N-1,N-1]
```

# Initial procedure to check that the system works [4]

7. Generate the (normalized) UPDRS value corresponding to `x`, i.e. the estimate of `y_val[i]`, using the formula of $\mu$ in the slides. Note that you need the column vector with the normalized total UPDRS of the first `N-1` points in `Z` to build vector `y` (vector $\mathbf{y}$ in the slides):

```
15      C=np.linalg.inv(R_Nm1)
16      refY=y_train_norm[ii]
17      mu=K.T@C@refY
18      sigma2=d-K.T@C@K
19      sigma=np.sqrt(sigma2)
20      yhat_val_norm[k]=mu
```

   Of course array `yhat_val_norm` must have been generated (as empy or zero) before the last line (otherwise you get an error message).

8. Verify that you get reasonable results by plotting, as usual, the estimate of `y_val_norm` versus `y_val_norm`.

# Procedure to set the hyerparameters [1]

- Note that, since the dataset was normalized, the autocorrelation $R_Y(\mathbf{0})$ (i.e. the values on the main diagonal of the covariance matrix) are equal to 1 and it is not necessary to optimize this value (i.e. $\theta = 1$).
- Parameters to be set are: `r2` and `s2`. They must be set so that**the mean square value of `y_val-yhat_val` (i.e. the validation mean square error) is minimized**. Equivalently you can minimize the mean square value of `y_val_norm-yhat_val_norm`. Use a set of values for `r2`, a set of values for `s2`, measure the validation mean square error, find the minimum.
- Don't ask the instructor about the range that you must consider for `r2` and `s2`, use common sense, find a solution, you are a master student.
- Moreover you should also have to correctly set `N`: we cannot use `N` equal to the number of datapoints in the training set (the covariance matrix would be too large), but `N=10` might be too small. However, use $N = 10$, it gives reasonable results.

# Procedure to set the hyerparameters [2]

- You should also try and see what happens if the features used for this regression are not just motor UPDRS, age and PPE, but keep only these regressors, they give reasonable results.

# Procedure to set the hyerparameters [3]

- In the overall you must find reasonable values of the hyperparameters

  `r2, s2`.

  keeping $N = 10$ and the suggested regressors.

# Testing

Once `r2, s2` have been optimized for the validation dataset, measure the following for the **test dataset**

1. the mean, the standard deviation, the mean square value of the estimation error for the un-normalized total UPDRS
2. the histogram of the un-normalized estimation error
3. the plot of the estimated total UPDRS versus the true one with the errorbars (use 3 times the value of `sigma` generated in line 19 of the code in the previous slides, but remember to un-normalize this value)
4. the value of $R^2$

so that you can compare the results obtained with the Gaussian process regression with the results obtained with linear regression LLS in Lab #1. Note that you must run again Lab #1 because **you must use exactly the same training data and exactly the same test data for LLS and GPR**. In particular validation dataset is used only for GPR, it is not used at all for LLS.

Note that each time you run your code you might get different results, since shuffling of the data is random. Use your PoliTo ID number to set the seed for the generation of random variables in Python. Each student shall have his/her own results.

**Report due within 14 days** (11.59 PM). Download the Latex file from the class Dropbox. **Maximum number of pages: 6**.