ELEC2870 Project
Automated Electrocardiogram Prediction

BERK SEVILMIS

SHRUTHI NARASIMHE GOWDA

INTRODUCTION

The purpose of this project is to develop several regression models, namely linear regression, kNN regression and RBF network, to predict one beat of an ECG signal (corresponding to 140 samples) from a given one wave (corresponding to 50 samples) of the signal. Training data having 50 consecutive values of an ECG time signal is provided to train regression models which are able to predict the 51st value. The overall performances of the models are then evaluated by the mean absolute error over 140 consecutive predictions in which the predictions are done sequentially.

Linear Regression

Linear regression model assumes that the relationship between the dependent variable y, the output, and the D dimensional independent variable x, the input vector, is linear. Hence, the analytical expression of the output takes the form:

$$y = \mathbf{x}\mathbf{w} + w_0$$

where **w** denotes the weight column vector multiplying each corresponding dimensions of x whereas w_0 denotes an additive bias. w_0 is needed because x=**0** is not necessarily give y=0. In order to obtain a more compact representation of the input-output relationship, the additive bias is included in the weight vector and each input vector **x** is concatenated with 1.

For a given input vector set and weights the resultant output vector will be calculated as follows:

$$\begin{pmatrix} 1 & x_1^1 & \dots & x_1^D \\ 1 & x_2^1 & \dots & x_2^D \\ 1 & x_p^1 & \dots & x_p^D \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \cdot \\ \cdot \\ w_D \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_p \end{pmatrix}$$

X w = y

In order to fit a model, MSE criterion is used as:

- The MSE possesses the very satisfying properties of convexity, symmetry, and differentiability and the optimization of parameters with MSE yields closed-form analytical solutions. Even when this is not the case, iterative numerical optimization procedures are often easy to formulate, since the gradient and the Hessian matrix of the MSE are easy to compute.
- It is the natural way to define the energy of the error signal.

MSE expression is given by:

$$E(w|X,T) = \frac{1}{P}\sum_{p=1}^{P}(y_p - t_p)^2 = \frac{1}{P}\|\boldsymbol{y} - \boldsymbol{t}\|^2 = \frac{1}{P}[\boldsymbol{Xw} - \boldsymbol{t}]^T[\boldsymbol{Xw} - \boldsymbol{t}]$$

where the activation is linear, σ(z)=z

Minimising the MSE yields the following expression for the weights:

$$E = \frac{1}{P}[Xw - t]^T[Xw - t]$$

$$E = \frac{1}{P}[w^T X^T - t^T][Xw - t]$$

$$E = \frac{1}{P}[w^T X^T Xw - w^T X^T t - t^T Xw + t^T t]$$

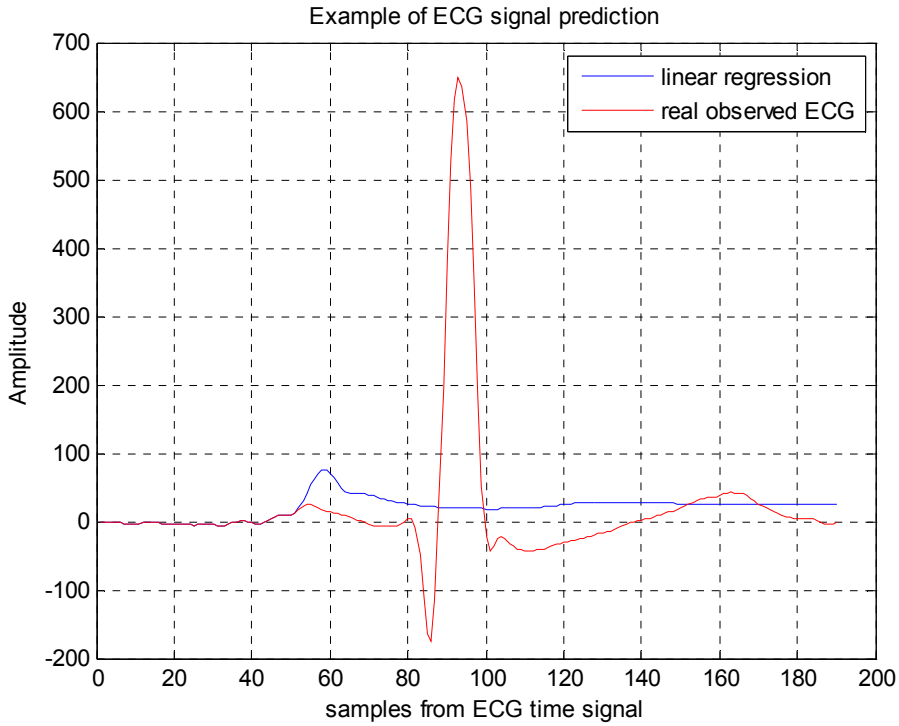
$$\nabla_{\mathrm{w}}\mathrm{E} = 0$$

$$\nabla_{\mathrm{w}}\mathrm{E} = \frac{1}{P}[w^T X^T X + w^T X^T X - t^T X - t^T X] = 0$$

$$\nabla_{\mathrm{w}}\mathrm{E} = \frac{2}{P}[w^T X^T X - t^T X] = 0$$

$$\mathrm{w} = (X^T X)^{-1} X^T t$$

With weights calculated using pseudo-inverse method, sequential prediction is done to obtain 140 predictions for each test input. Below shown is a result obtained over a given test input:



K-Nearest Neighbour (KNN) Regression

Methodology

- For a given input vector from X_test data set, Euclidean distances to each vector in X_train are calculated. The distances are sorted and the closest K minimum input vectors from X_train are considered.
- The respective T_train samples of these K closest input vectors are averaged. This provides the first predicted value. (51st value of the resulting time series).
- For the next prediction, input vector from X_test is shifted to include the elements from 2 to 51 where 51 corresponds to the value that has been predicted previously.
- Sequential prediction is continued till 140 predictions have been made.
- This provides the first row of the output y. The method is iterated for all rows (20 times) of the X_test data set to obtain the output matrix.

Meta Parameter Optimisation:

The kNN method depends solely on the parameter k, which determines the number of input vectors being used in determining the target. K is optimised by using the bootstrap method.

Bootstrap Method

$$E_{gen} = E_{S,S} + \frac{1}{K}\sum_{k=1}^{K}\left(E_{S^{*k},P^k} - E_{S^{*k},S^{*k}}\right)$$

$$S = training\ data\ set\ ,\ S^* = bootstrap\ samples\ ,\ T^* = target\ set\ corresponding\ to\ S^*$$
$$S^*\ and\ T^*\ have\ the\ same\ size\ as\ S\ and\ T$$

In order to estimate generalization error, two terms are considered. First term, $E_{S,S}$, is the mean square error of training. The second term is called optimism and is utilised to avoid overfitting. In order to obtain optimism, bootstrap samples are generated by randomly picking indices from the training set. Replicated indices are allowed which results in only some of the data being used to train the network. When the network trained by bootstrap samples is used to test the training data, the network encounters new samples. The resulting MSE , E_{S^{*k},P^k} , helps correcting the overfitting by providing a measure of how well the network behaves for new inputs.

kNN Bootstrap Method

Consider the following variables :

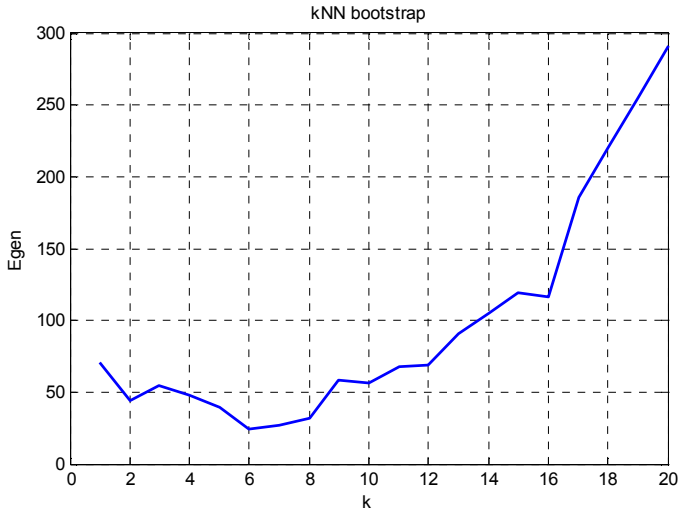
For a given set of values of k

Data
X_Reference
T_Reference
T_Comparison

- Compare **Data** with **X_Reference** and get indices of minimun distances.
- Find the first k values of these indices in **T_Reference** and calculate the mean.
- Compare with **T_Comparison** and find the error.
- Egen is calculated for each values of k and a matrix is generated to select the best k value.

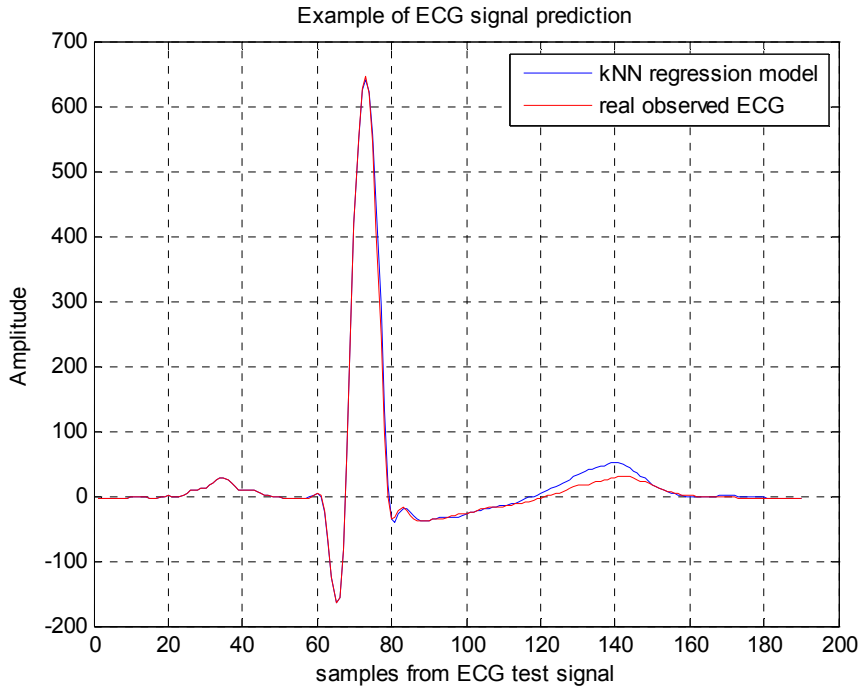
	$E_{S,S}$	$E_{S^*,P}$	E_{S^*,S^*}
Data	X_train	X_train	S^*
X_Reference	X_train	S^*	S^*
T_Reference	T_train	T^*	T^*
T_Comparison	T_train	T_train	T^*

kNN bootstrap vector										
K	1	2	3	4	5	6	7	8	9	10
Egen	70.304	44.2878	54.1659	48.0148	39.6675	23.8395	26.7302	32.2003	58.3121	56.8976
K	11	12	13	14	15	16	17	18	19	20
Egen	67.486	68.3866	90.4513	104.7531	119.1012	116.3925	185.8871	219.7217	254.9042	290.5448



K=6 gives the minimum error.

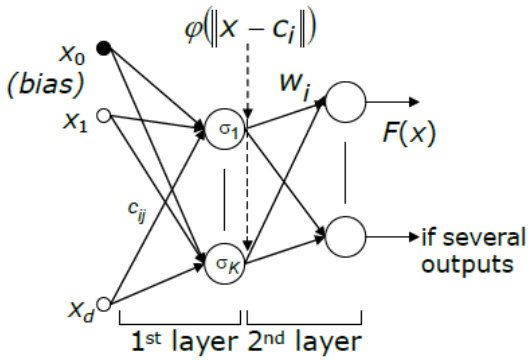
Performing kNN regression using K=6.



Neural Model Regression

The Neural model chosen here is Radial Basis Function Network (RBFN). Radial basis function network (RBFN) was chosen because of its advantages over traditional multilayer perceptron (MLP), namely faster convergence, smaller extrapolation errors, and higher reliability. Also RBF network has more meta-parameters than MLP so it is more flexible to tune, it is more resilient against a small training set than an MLP hence providing better results.

RBFN is a class of single hidden layer feed forward networks where the activation functions for hidden units are defined as radially symmetric basis functions such as the Gaussian function.



An example RBFN

Each hidden unit forms a localized receptive field in the input space X, whose centroid is located on c with width σ. The fraction of overlap between each hidden unit and its neighbors is decided by the width σ such that a smooth interpolation over the input space is allowed.

$$F(x) = \sum_{i=1}^{K} w_i \phi(\|x - c_i\|) \quad ; \quad \phi(\|x - c_i\|) = exp\left(-\frac{\|x - c_i\|^2}{2{\sigma_i}^2}\right)$$

Model selection is very important and relies on the selection of the meta parameters.

- number of kernels (number of hidden units)→ N
- smoothness factor (width σ)→ W

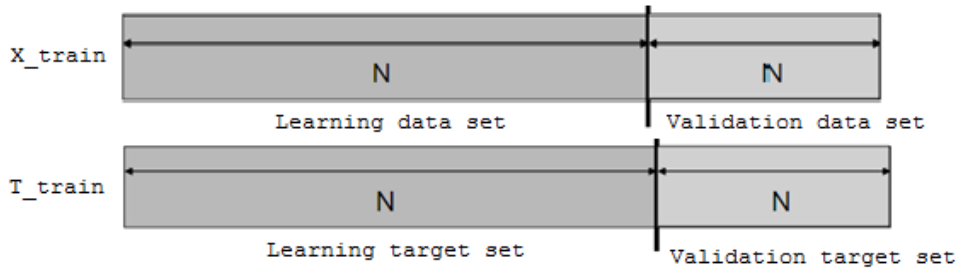
Weights and centers are obtained from vector quantization during training of the RBFN. Learning can be performed only after knowing these parameters. Various methods have been used to determine these meta parameters.

Meta Parameters Optimisation:

Cross Validation Method

The data set is split into two parts: learning set and validation set.

$$\hat{E}_{gen} = \frac{1}{K}\sum_{k=1}^{K}\frac{\sum_{t\in VS^k}\left(\hat{y}_t^{\,k} - y_t\right)^2}{N_2}$$



- The RBFN model is trained with learning data set and tested with validation data set. Generalization error is averaged over K observations. Before each observation, the data set is shuffled. K is taken to be 10 while the number of samples in the validation set is 100.

Bootstrap Method

Bootstrap is also tested along with cross validation because bootstrap has nearly unbiased estimates of predictive accuracy that are of low variance and it also regulates the over fitting thus giving better results.

- $E_{S,S}$: for every value of N and W, the RBFN model is trained with X_train and the MSE of the training is obtained.

S^* and $T^*$are bootstrap samples.

- E_{S^{*k},P^k}: for every value of N and W, the RBFN model is trained with S^* , tested with X_train and the MSE is obtained.
- $E_{S^{*k},S^{*k}}$: for every value of N and W, the RBFN model is trained with S^* and the MSE of the training is obtained.

Egen is calculated for each N and W and the [N,W] pair giving the minimum Egen is selected as the model parameters. 10 recursive iterations have been performed for calculating optimism.

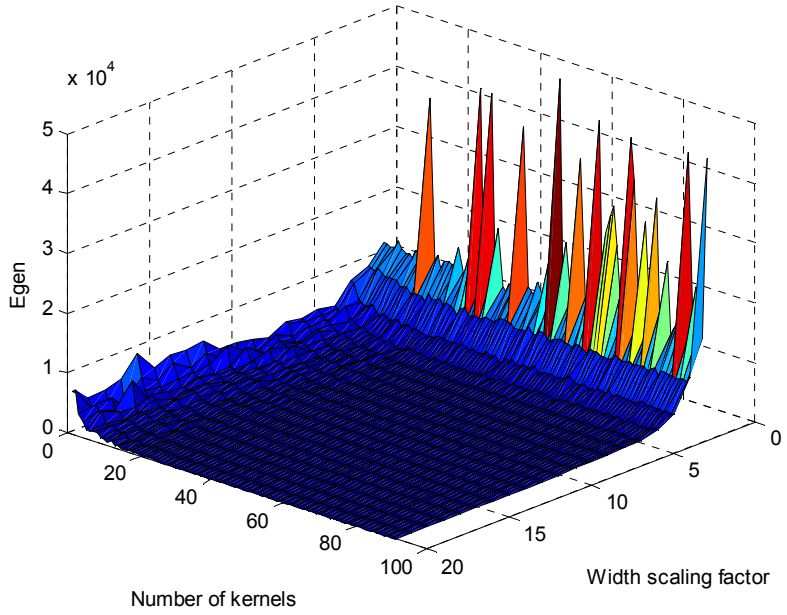
Bootstrap-632 Method

In Bootstrap-632 method the optimism is only calculated on the complementary set of the bootstrap samples. The generalization estimate is given as:

$$\widehat{E_{gen}} = 0.368E_{S,S} + 0.632\widehat{optimism}$$

Due to computational complexity, bootstrap-632 method is only applied for the given range of values: N= 90 to 100 and W= 1 to 10.

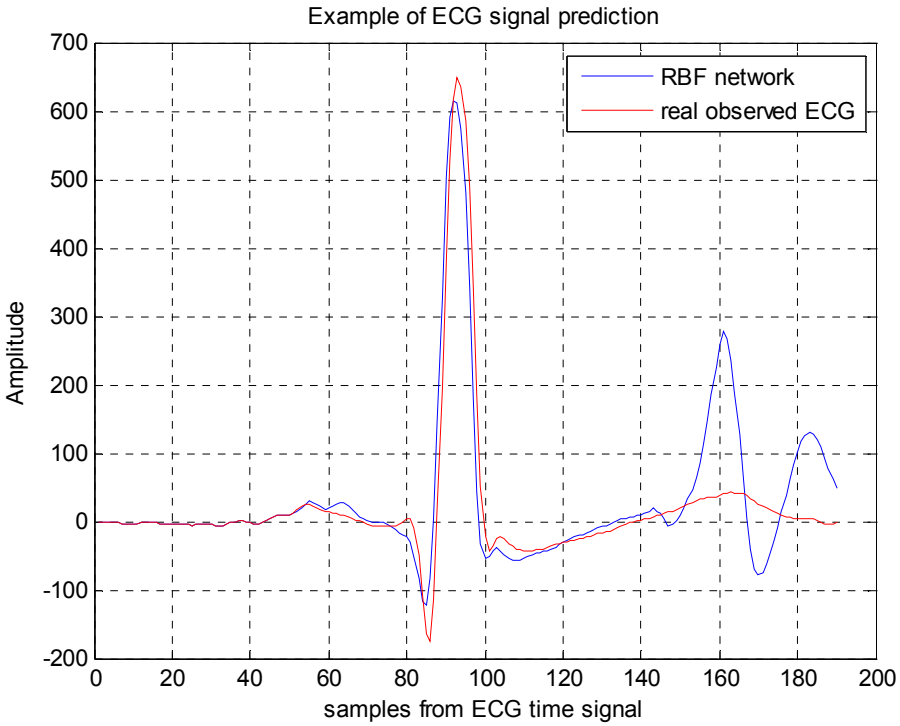
Below shown is the surface plot of the bootstrap method for hidden units changing from 10 to 100 and width scaling factor changing from 0.5 to 10 with increments of 0.5. Results obtained from the abovementioned parameter optimization methods are attached as .mat files along with the codes.



The meta-parameters returned from the above methods are tabulated below:

Method	Number of kernels	Width scaling factor
Cross-validation	93	10
Bootstrap	92	10
Bootstrap 632	95	10

Since the bootstrap 632 was tested for a narrow range of values and it is known that bootstrap outperforms cross validation (as was proven later also), results from bootstrap method are used for model selection.



Mean absolute error comparison of all the three methods are shown below:

Method	Mean Absolute Error
Linear Regression	61.2827
kNN Regression	13.0140
RBF network	98.2226

From the above table, the error from RBFN seems worse than linear regression. The reasoning is that even though the RBFN is able to predict the peak properly (as can be seen from the above figure), most of the time it is shifted which in turn results in a very high Mean Absolute Error whereas the linear regression is not able to predict the peak at all.

Note: For the RBFN case, after the meta-parameters were obtained from bootstrap, a trial and error search near these parameters yielded N=94 and W=4 to be the best parameters resulting in a Mean Absolute Error of 44.0321. However, none of the meta-parameter optimization methods were able to return these parameters.

CONCLUSION

In this project, three different machine learning tools have been employed to predict one beat of an ECG signal. It has been noticed that among all the methods, kNN regression gives best results along with being computationally efficient. Linear regression was unable to predict properly. RBFN was able to predict P waves, T waves and QRS complexes but most of the time was not synchronising with the real ECG signal. Exhaustive methods were implemented to find the optimal meta-parameters in both kNN regression and RBF networks.