Machine Learning: Assignment 2

Karsten Standnes - STNKAR012

October 2017

1 Introduction

In this assignment I'll be looking at the performance of different models when fitting a data created by a unknown mode. In "Task 1" two linear models with an offset is fitted to dataset and the relative performance is measured. "Task 2" looks at making legendre based model and fit them to a dataset made from a sinus function. Different values for λ, the regularization parameter is used. In "Task 3" a set of facial images are processed and used to produce eigenfaces. This task evolves around eigenfaces and how they can be used to reconstruct a picture.

Contents

2 Task 1

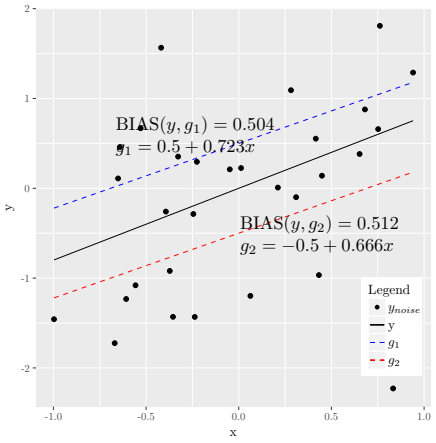
In this task we look at the linear function of the form:

$$\begin{aligned} &y_i = 0.8x_i + \epsilon_i; \quad -1 \leq x \leq 1, i = 1, ..., N \\ &\text{where: } \epsilon_i \sim \text{Normal}(0, 1) \end{aligned} \tag{1}$$

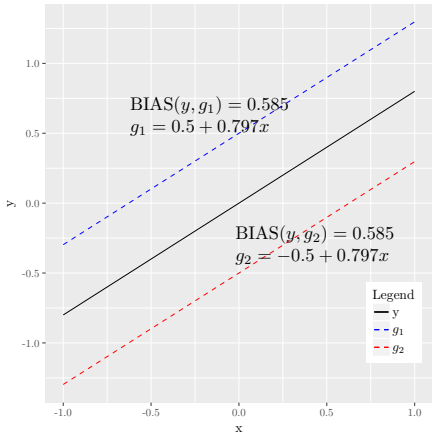
2.1 Task 1 i

First a dataset of size $N = 30$ is created from using equation 1 using a set of x values generated from the Uniform$(-1, 1)$. Then two models are attempted fit two the dataset. The models fitted are:

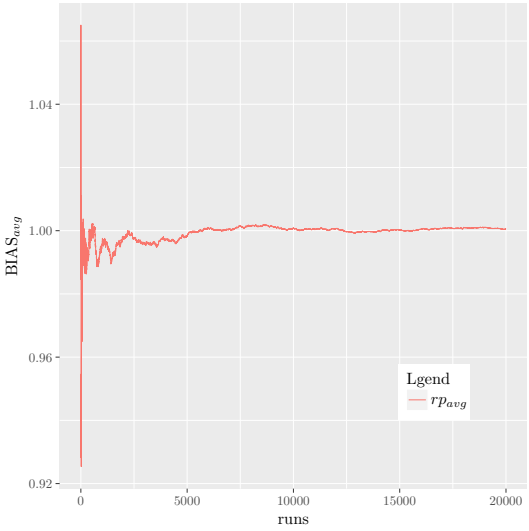
$$\begin{aligned} g_1(x) &= 0.5 + b_1 x \\ g_2(x) &= -0.5 + b_2 x \end{aligned}$$



(a) g_1 and g_2 fitted to the noisy data.



(b) Average fitted g_1 and g_2 over 20000 data sets.



(c) Relative bias $\frac{\text{BIAS}(y,g_1)}{\text{BIAS}(y,g_1)}$ for 1 to 20000 runs.

Figure 1: .

In figure 1a we see the underlining function together with a dataset created from equation 1 and the linear functions $g_1(x)$ and $g_2(x)$ fitted to the data. Both functions in this case performs fairly well with pretty similar slope, with $g_1(x)$ performing slightly better. This being the result from only one dataset means that it can't be viewed as representative for the performance of the functions. To get a better picture, 20000 dataset are created and averaged over. Figure 1b shows $g_1(x)$ and $g_2(x)$ after 20000 runs. We can see that the bias and slope is the same for both giving a relative performance (rp) equal to 1. In figure 1c below the relative performance is plotted as a function of average relative bias after 1 to 20000 runs. After around 5000 runs the relative bias starts to converge towards 1, meaning the models perform equally well. The functions have a constant ± 0.5 making it impossible to perfectly fit the underlying function. Intuitively the best models will have the slope with a offset of ± 0.5. This is confirmed when the slope of the functions converge to $b_1, b_2 \approx 0.8$, the slope of the original underlying function.

2.2 Task 1 ii

By using the model in equation 1, 10000 datasets of size $N = 30$ are simulated. Each is set is split into 21 combinations of validation and training sets, where the validation set is of the size i and the training set of the size $30 - i$, with $i \in [5, 25]$. For each of the set the models $g_1(x)$ and $g_2(x)$ are attempted fit to the training data and validated on the validation set. For each dataset and i the best of the two models is selected. For each model selected the error measures E_{out} and E_{val} is calculated and the average over all the datasets is taken to give the errors for each combination of training and validation set. Below the averaged best for each i is plotted.

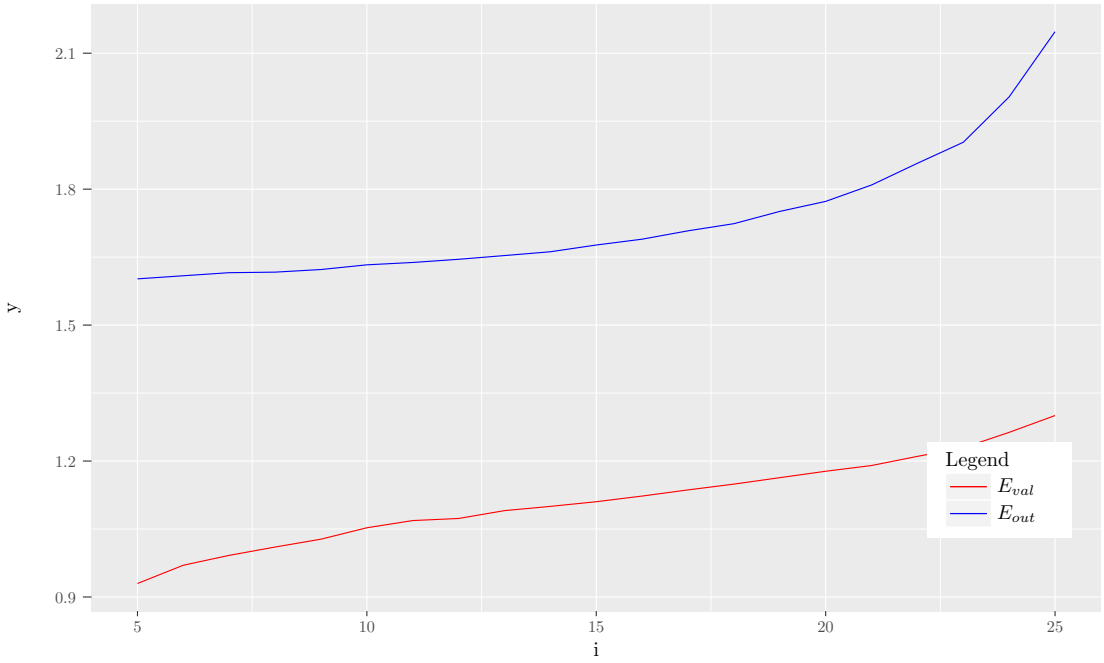


Figure 2: Error measures E_{out} and E_{val} are plotted for different sizes of training and validation sets.

In figure 2 it is a clear difference between the value of E_{val}(use MSE) and E_{out}(use BIAS) which is pretty stable for different sizes of the training and validation set. We can see that both the errors for increases for increasing i. E_{out} starts with incrementing slowly, then more rapidly around $i \sim 15 - 20$, while E_{val} has a more steady increase with a faster increase in the start ($i < 10$). We can deduce from this that E_{val} will consequently give a lower value than E_{out} and that decreasing the training set in advantage for increasing the validation set give a worse performance, at least for datasets of size $N = 30$. This makes sense since the less data the models have available during

training, the more likely it is to fit the noise of the data. Then increasing the validation set does not help other than giving a more precise measure of the performance. That being said, having a large enough validation set is also important to keep up the precision of the measure of error. Here the conclusions of it being a disadvantage to decrease the training set can be drawn using the average of 10000 sets of data.

3 Task 2

In "Task 2" we look at the function of the form:

$$\begin{aligned} &y_i = \sin(\pi x_i) + \epsilon_i; \quad -1 \leq x \leq 1, i = 1, ..., N \\ &\text{where: } \epsilon_i \sim \text{Normal}(0, 1) \end{aligned} \tag{2}$$

For each of the subtasks a dataset/datasets of size $N = 50$ with $x \sim \text{Uniform}(-1, 1)$.

3.1 Task 2 i

Simulate a dataset using equation **??** and plot it together with the underlying model:

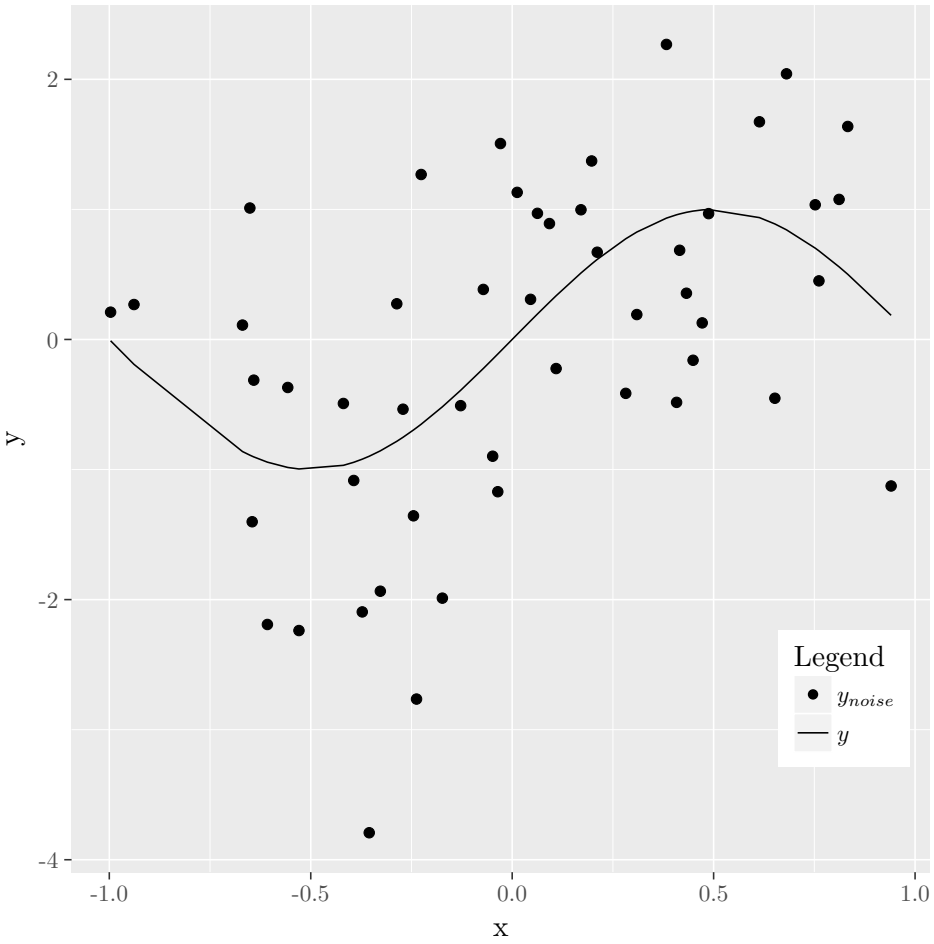


Figure 3: Underlying function with simulated dataset using equation 2

3.2 Task 2 ii

In this subtask two models are fitted to the noisy data, the models are based on

$$y_i = \sum_{q=0}^{Q_f=10} \beta_q L_q(x) \tag{3}$$

,$L_q(x)$ being the Legendre polynomial of q-th order. The Legendre polynomial is given by:

$$L(x) = 2^q \sum_{k=0}^{q} x^k \binom{q}{k} \binom{\frac{q+k-1}{2}}{q} \tag{4}$$

. The two models have a regularization parameter λ equal to 0 and 5 for the models. Using linearization to fit the models to the noisy data:

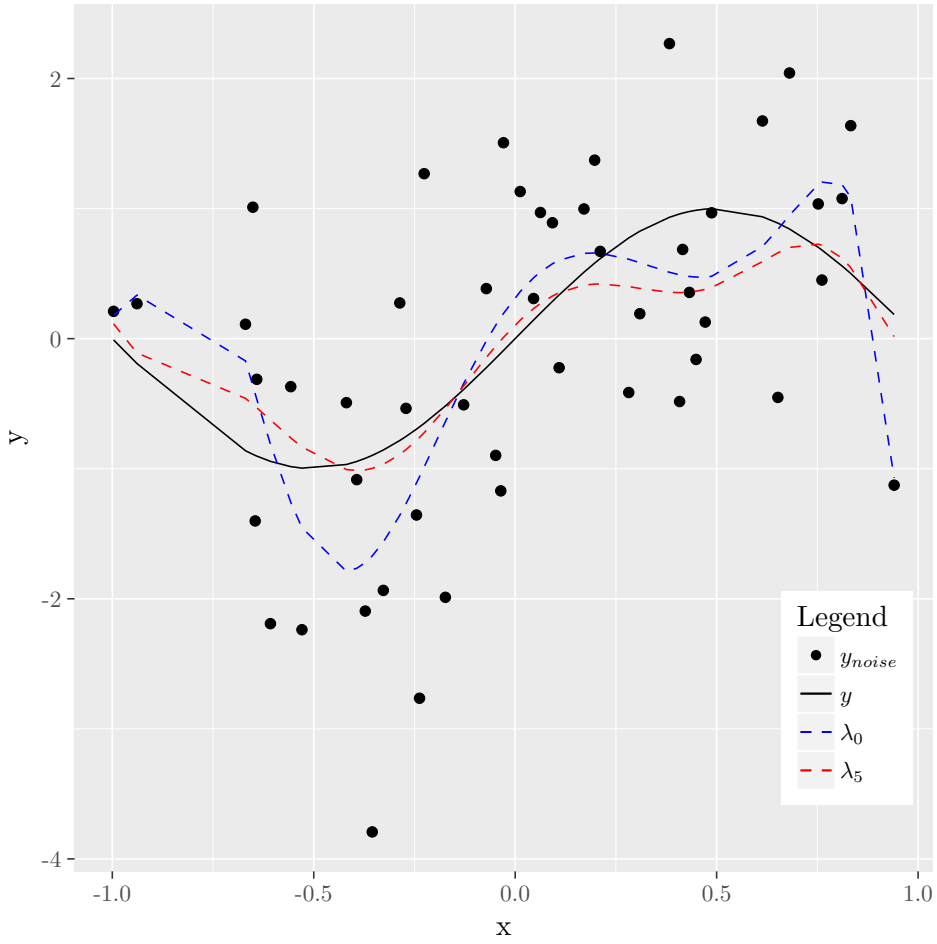
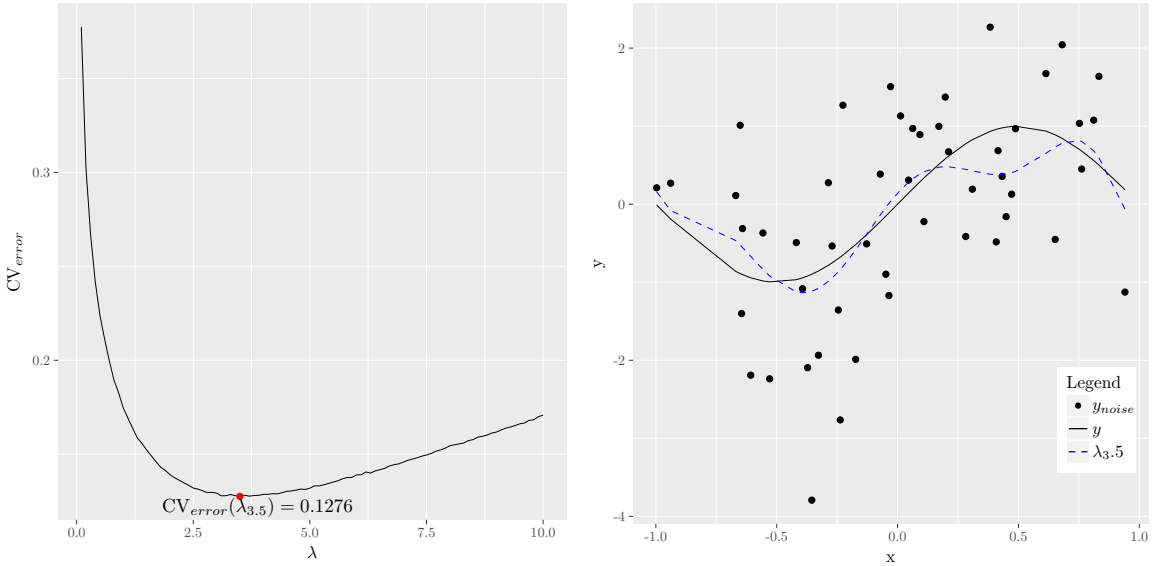


Figure 4: Two legendre models are fitted to the dataset with different values for λ.

From the figure 4 we can see that the model with a higher regularization parameter give a result that is much more similar to the underlying function. The model with $\lambda = 0$ has much more rapid change in direction, leading to it quickly missing the underlying function. This indicate that regularization increase the precision of the model compared to no regularization, at least for regularization parameter $\lambda = 5$.

3.3 Task 2 iii

To try to find the best regularization parameter for the legendre based model we can use cross-validation. In this task 10-fold cross-validation is used, meaning the dataset is split into 10 folds where each fold is used as a validation set with the other nine as the training set. This is done for all folds, then the mean-squared-error is calculated for the results of all the folds. Below are two plots, figure 5a giving the best λ based on the CV-error for 1000 dataset and figure 5b showing a model fitted with the best found λ on the data from figure 4.



(a) λ_i giving the lowest CV-error over 1000 datasets

(b) Model fitted with best $\lambda_i = 3.5$.

Figure 5: Task 2 iii

From the results above we can clearly see that the CV-error drastically decrease in the start when increasing the value of λ, then it has stabilize while curving slightly around $\lambda \in [2.5, 5.0]$ before it gradually increases in a seemingly linear fashion. When plotting the function with the obtained best regularization parameter $\lambda = 3.5$, it performs quite similar as the model with $\lambda = 5$ in figure 4. Looking at the CV-error plot this seems very reasonable, giving that both values give a very similar error. It's hard to say, but it might also seem like the model with $\lambda = 5$ performs better on this dataset. This does not destroy the result that $\lambda = 3.5$ is the best in figure 5a since an *average best* does not guarantee a *always best*.

4 Task 3

Using a image set of 400 pictures of 40 people with 10 of each person, we look at eigenfaces and reconstructing faces from them using principal component analysis PCA.

4.1 Task 3 i

(a) Mean image/face

(b) Standard deviation image/face

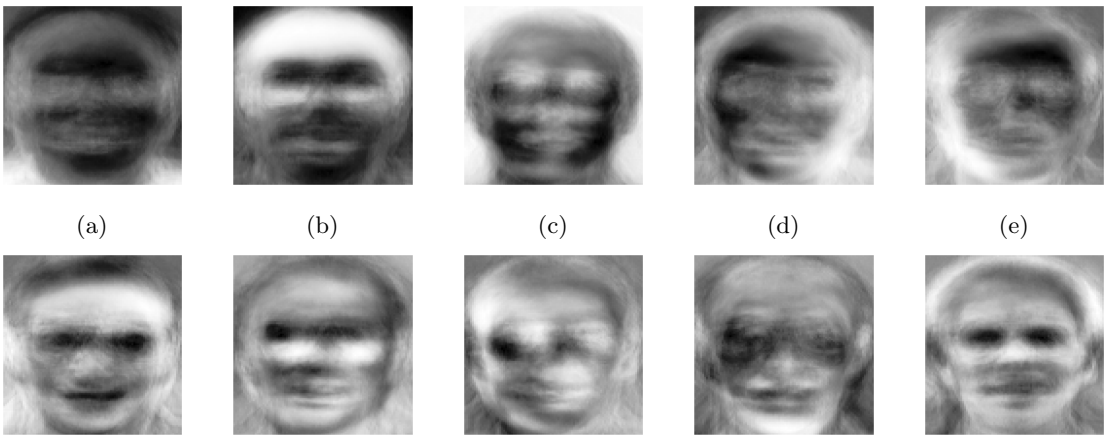
(c) Original image/face

(d) Scaled image/face

Figure 6: Task 2 i - SD and mean image/face of the set together with tha scaled and original image/face of i" *168.pgm*

4.2 Task 3 ii

Deriving the first ten eigenfaces by using the whole set of images:



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figure 7: Ten first eigenfaces based on image set.

4.3 Task 3 iii

In this subtask we will look at how it is possible to reconstruct a image using eigenfaces. Below the image "*115.pgm*" together with reconstruted images based on 5, 50 and 200 eigenfaces.

(a) Scaled image

(b) Reconstructed with 5 eigenfaces

(c) Reconstructed with 50 eigenfaces

(d) Reconstructed with 200

Figure 8: Scaled version of "*115.pgm*" together with reconstructed versions.

The purpose of using eigenfaces is to efficiently store and encode images. Eigenfaces are derived eigenvalues of the co-variance matrix of the set of images. By using eigenfaces it is possible to reduce the dimensionality by using a smaler set of images to represent the orginal set. The eigenvalues stores the variation in the set of face images and can be combined to reconstruct a specific image. As seen in figure 8 that as the number of eigenfaces increase the face shown starts to look more and more similar to the scaled version of the real face. Another interesting observation is that as the number of eigenfaces used increase each image has a lesser impact in improving the quality. The difference in increasing from 5 to 50 eigenfaces is much more dramatic than from 50 to 200 eigenfaces. This indicate that to get a recognizable image a small amount of eigenfaces is needed and increasing the number after that will only improve the detail

5 Appendix