Machine Learning - FMAN45
Assignment 1, spring 2019

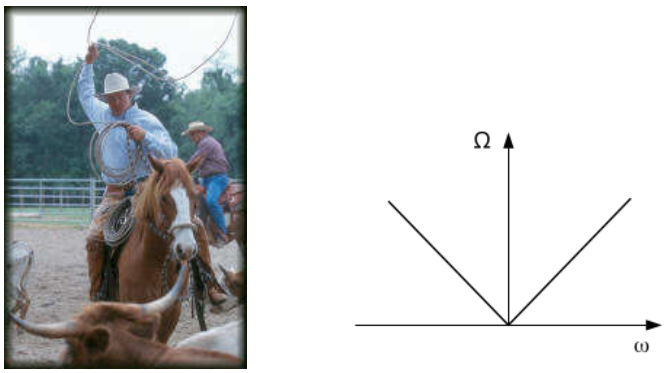
Penalized Regression via the Lasso
Hyperparameter-learning via K-fold Cross-validation
Denoising of an Audio Excerpt

Hicham Mohamad
hsmo@kth.se

April 18, 2019

1 Introduction

In this assignment, we are going to work on the machine learning algorithm, `linear regression`, that involves the noisy dataset `A1_data.mat` consisting of N = 50 data points. We first use the so called `Least Absolute Shrinkage and Selection Operator` or LASSO to penalize the regression and solve the minimization problem using a coordinate-wise approach. Then we need to implement the `cyclic coordinate descent` algorithm and run it on the same dataset with a `K-fold cross-validation` scheme for the Lasso estimator. Finally, we will perform denoising of a noisy recording of piano music using a K-fold cross-validation scheme for the multi-frame audio excerpt.



(a) (b)

Figure 1: a) The Lasso as an english term. b) The lasso regularization term.

2 Objectives

The objectives of this assignment are to:

1. Penalize the regression problem using the coordinate-wise LASSO minimizer.
2. Implement a cyclic coordinate descent solver.
3. Implement a K-fold cross-validation scheme for the LASSO solver.
4. Implement a K-fold cross-validation scheme for the multi-frame audio excerpt.
5. Denoise the test data `Ttest` using the provided function `lasso_denoise()`.

3 Background

An agent/model is learning if it improves its performance on future tasks. Starting from a collection of `input-output pairs`, it can learn a function that predicts the output for `new inputs`. Machine learning algorithms can be classified along two main lines: `supervised` and `unsupervised` classification.

The most common supervised learning tasks are *regression* (predicting values) and *classification* (predicting classes). In supervised learning the agent observes some example input–output pairs and learns a function that maps from input to output. In supervised learning, given a `training set` of N example input-output pairs

$$(x_1, y_1), (x_2, y_2), \cdots (x_N, y_N)$$

where each y_i was generated by an unknown function $y = f(x)$, we need to discover a function h that approximates the true function f. Here x and y can be any value and they need not to be numbers. The function h is a `hypothesis`. Learning is a search through the space of possible hypotheses for one that will perform well, *even on new examples beyond the training set.* To measure the *accuracy* of a hypothesis we give it a `test set` of examples that are distinct from the training set. In this case, we say a hypothesis `generalizes` well if it correctly predicts the value of y for novel examples. Sometimes the function f is stochastic - it is not strictly a function of x, and here we need to learn a conditional probability distribution, $\mathbf{P}(Y|x)$.

When the output y is one of a finite set of values, the learning problem si called **classification**, and is called Boolean or binary classification if there are only two values. When y is a number, such as tomorrow's temperature, the learning problem is called **regression**.

In general, there is a `tradeoff` between complex hypotheses that fit the training data well and simpler hypotheses that may generalize better.

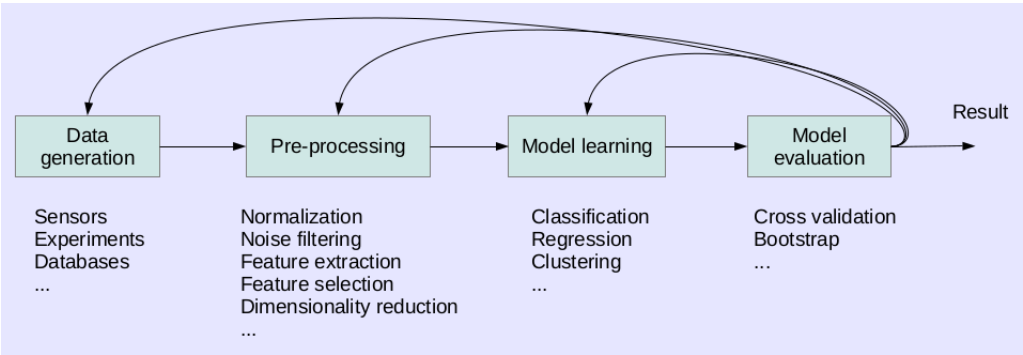


Figure 2: Typical learning process

As shown in Figure 2, we can distinguish between many tasks in machine learning, such as classification, regression, clustering, sequence prediction, imputation of missing values etc, where classification problems are very popular.

3.1 Capacity, overfitting and underfitting

The central challenge of machine learning is to perform well on *new previously unseen data*, not part of the training. A model with such ability is said to generalize.

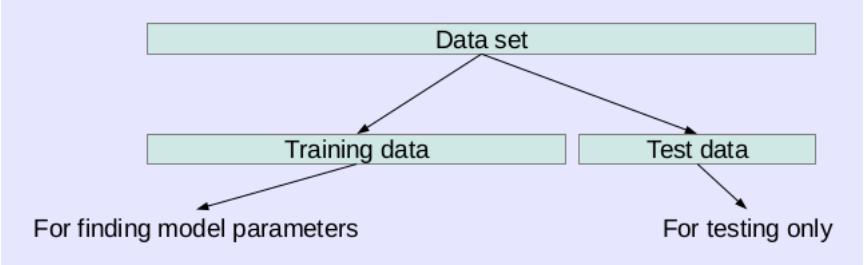


Figure 3: Training data and test data.

To evaluate the performance of the machine learning algorithm, one often uses during training an `error measure` instead of a performance measure. We expect that `test error` $\geq$ `training error`, and the goal is to minimize training error. Thus, we need to minimize the `gap` between test error and trianing error as small as possible, as illustrated in Figure 4.

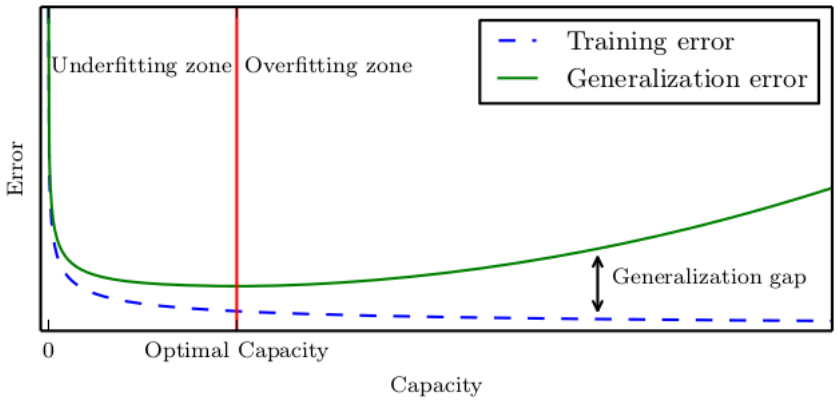


Figure 4: capacity, overfitting and underfitting.

A model can be said to possess a certain **capacity**, i.e. the ability to fit a range of different functions. With small capacity we can only fit a limited number of functions and increasing the capacity means fitting a larger set of functions. It may be controlled by choosing the hypothesis space (representational capacity) or by the number of trainable parameters.
In practice, other factors also affect a model's capacity, such as, e.g., optimization method and imperfection, and how much data is used for training (effective capacity). In statistical learning, capacity is quantified by the **Vapnik-Chervonenkis dimension** (VC-dimension):

$$\text{test error} \;\geq\; \text{training error} \;+\sqrt{\frac{h + h\log(2N/h) - \log(p/4)}{N}} \tag{1}$$

where N = size of training set, h = VC dimension of the model class = complexity and p = uppper bound on probability that this bound fails. As we can see, The discrepancy between test and training error grows when the VC dimension increases, and decreases when the number of training examples increase. If we train models of different complexities, we should ideally select the one that minimizes the type of bound above.

3.2 Evaluation - Cross validation

In overfitting situation, we want an efficient way for modifying the capacity of an ML model to find balance between underfitting and overfitting in order to reduce the **generalization error**. One of the various approaches is cross-validation method which is a way to estimate the generalization error, where the estimate's uncertainty is taken into account. In this method, we split (randomly) the dataset into K subsets of (approximate) equal size. Then we train K models on K slightly different training sets (in grey) and validate on K different validation sets (in red). Figure 5 illustrates how the different training and validation sets are defined.

Figure 5: The K-fold cross validation method, where $K = 5$ can mitigate the overfit. Each of the K validation parts are used as a validation set when training on the remaining parts.

.

In this way, we notice that each example serves double duty — as training CROSS-VALIDATION data and test data. Popular values for k are 5 and 10 which is enough to give an estimate that is statistically

likely to be accurate, at a cost of 5 to 10 times longer computation time. The extreme is k = n, also known as `leave-one-out cross-validation` or LOOCV.

3.3 Loss function

In machine learning it is traditional to express utilities by means of a loss function. The loss function $L(x, y, \hat{y})$ is defined as the amount of utility lost by predicting $h(x) = \hat{y}$ when the correct answer is $f(x) = y$. Thus, we note that $L(y, y)$ is always zero; by definition there is no loss when you guess exactly right. In general small errors are better than large ones; two functions that implement that idea are the `absolute value of the difference`, called the L_1 loss, and the `square of the difference` (called the L_2 loss):

$$\text{Absolute value loss: } L_1(y, \hat{y}) = |y - \hat{y}|$$

$$\text{Squared error loss: } L_2(y, \hat{y}) = (y - \hat{y})^2$$

3.4 Regularization

In the case we have **correlated features** wich result in large values of $\mathbf{w}$, regularization is used to penalize them. We replace the loss function by a **cost function**

$$Cost(\mathbf{w}) = Loss(\mathbf{w}) + \lambda\Omega(\mathbf{w}) \tag{2}$$

where Ω is the regularization term and λ control the amount of regularization. We can consider different types of Ω term, as the L2 norm (weight decay) and the L1 norm (Lasso).

4 Penalized regression via the LASSO

Linear Regression is usually the first machine learning algorithm that every data scientist comes across. It is a simple model but everyone needs to master it as it lays the foundation for other machine learning algorithms. Where can Linear Regression be used? It is a very powerful technique and can be used to understand the factors that influence **profitability**. For example, it can be used to forecast sales in the coming months by analyzing the sales data for previous months. It can also be used to gain various insights about **customer behaviour**.

In this assignment, we need to work with a type of penalized regression, called LASSO, which stands for Least Absolute Shrinkage and Selection Operator. This method is used when the explanatory variable, or weights, $\mathbf{w} \in R^M$ is sparse, having few non-zero coordinates. The LASSO solves the minimization problem

$$\min_w \frac{1}{2}\|\mathbf{t} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\| \tag{3}$$

where $\lambda \leq 0$ is a capacity **hyperparameter**.

Notations - Hypothesis of Linear Regression

In general, the linear regression model can be represented by the following equation

$$\mathbf{Y} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

and the purpose is to estimate the weights $\mathbf{w} \in \mathbb{R}^M$ given a linear relationship to the response variable, data, $\mathbf{Y} \in \mathbb{R}^N$, defined by the regression matrix $\mathbf{X} \in \mathbb{R}^{N \times M}$.

- Y is the predicted value
- w_0 is the bias term.
- $w_1, \cdots, w_n$ are the model parameters, weights or explanatory variables, to be estimated.
- $x_1, x_2, \cdots, x_n$ are the feature values or regressors.

The above hypothesis can also be represented by

$$\mathbf{Y} = \mathbf{w}^T\mathbf{X}$$

where $\mathbf{w}$ is the model's **parameter vector** including the bias term w_0 and $\mathbf{X}$ is the **feature vector** with $x_0 = 1$. Linear regression and neural networks use the training data to estimate a fixed set of parameters $\mathbf{w}$. That defines our hypothesis $h_{\mathbf{w}}(\mathbf{x})$, and *at that point we can throw away the training data, because they are all summarized by* $\mathbf{w}$. A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a **parametric model**. A **nonparametric model** is one that cannot be characterized by a bounded set of parameters.

4.1 Task 1 - Solving a sequence of optimization problems

Because there is no closed-form solution for the LASSO minimization problem in 3, we need to consider the coordinate-wise approach. It consists of optimizing iteratively only one coordinate in $\mathbf{w}$ at a time, keeping the others fixed at their most recent value. In this way, at iteration j the **coordinate descent** minimizer of x_i has the **closed-form solution**

$$\hat{w}_i^{(j)} = \begin{cases} \frac{\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}}{\mathbf{x}_i^T\mathbf{x}_i \quad |\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}|}(|\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}| - \lambda) & \text{when} \quad |\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}| > \lambda \\ \\ 0 \qquad \text{when} \quad |\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}| \leq \lambda \end{cases} \tag{4}$$

where

$$\mathbf{r}_i^{(j-1)} = \mathbf{t} - \sum_{l<i} \mathbf{x}_l\hat{w}_l^{(j)} - \sum_{l>i} \mathbf{x}_l\hat{w}_l^{(j-1)} \tag{5}$$

For solving this sequence of optimization problems and verify the closed-form solution in 4, we need to differentiate the objective in the following minimization problem

$$\min_{w_i} \frac{1}{2}\|\mathbf{r}_i - \mathbf{x}_i w_i\|^2 + \lambda|w_i| \tag{6}$$

where $\mathbf{x}_i$ denotes the regression matrix and

$$\mathbf{r}_i = \mathbf{t} - \sum_{l\neq i} \mathbf{x}_l w_l$$

denotes the residual vector without the effect of the i:th regressor. The objective in Equation 6 is minimized when its partial derivatives with respect to w_i are zero. Thus, differentiating implies that

$$\begin{aligned} \frac{\partial}{\partial w_i}[\frac{1}{2}(\mathbf{r}_i - \mathbf{x}_i w_i)^T(\mathbf{r}_i - \mathbf{x}_i w_i)] + \lambda\frac{\partial}{\partial w_i}|w_i| = \\ \frac{\partial}{\partial w_i}[\frac{1}{2}(\mathbf{r}_i^T\mathbf{r}_i - 2\mathbf{r}_i^T\mathbf{x}_i w_i + (\mathbf{x}_i w_i)^T(\mathbf{x}_i w_i))] + \lambda\frac{\partial}{\partial w_i}|w_i| = \\ -\mathbf{r}_i^T\mathbf{x}_i + (\mathbf{x}_i^T\mathbf{x}_i)w_i + \lambda\frac{w_i}{|w_i|} = 0 \end{aligned} \tag{7}$$

As mentioned in the text, for $w_i \neq 0$, its derivative is the sign of w_i , i.e.

$$\frac{\partial}{\partial w_i}|w_i| = \frac{w_i}{|w_i|}$$

Now after differentiating, we can solve for w_i and get the resulting equation

$$(\mathbf{x}_i^T\mathbf{x}_i)w_i + \lambda\frac{w_i}{|w_i|} = \mathbf{r}_i^T\mathbf{x}_i = w_i(\mathbf{x}_i^T\mathbf{x}_i + \frac{\lambda}{|w_i|}) \tag{8}$$

Now we should get rid of the w_i and solve for $|w_i|$ first by computing the absolute value of the left side term and that of right side in Equation 8, i.e.

$$|\mathbf{r}_i^T\mathbf{x}_i| = |w_i(\mathbf{x}_i^T\mathbf{x}_i + \frac{\lambda}{|w_i|})| = |w_i|\mathbf{x}_i^T\mathbf{x}_i + \lambda \quad \implies \quad |w_i| = \frac{|\mathbf{r}_i^T\mathbf{x}_i| - \lambda}{\mathbf{x}_i^T\mathbf{x}_i} \tag{9}$$

then, by solving for w_i in Equation 8, we get the following:

$$\mathbf{r}_i^T\mathbf{x}_i = w_i(\mathbf{x}_i^T\mathbf{x}_i + \frac{\lambda}{|w_i|}) \implies \mathbf{r}_i^T\mathbf{x}_i|w_i| = w_i(\mathbf{x}_i^T\mathbf{x}_i|w_i| + \lambda) \implies w_i = \frac{\mathbf{r}_i^T\mathbf{x}_i|w_i|}{\mathbf{x}_i^T\mathbf{x}_i|w_i| + \lambda} \tag{10}$$

Finally, by plugging 9 back into w_i in Equation 10 we get

$$w_i = \frac{\mathbf{r}_i^T\mathbf{x}_i(\frac{|\mathbf{r}_i^T\mathbf{x}_i|-\lambda}{\mathbf{x}_i^T\mathbf{x}_i})}{|\mathbf{r}_i^T\mathbf{x}_i| - \lambda + \lambda} = \frac{\mathbf{r}_i^T\mathbf{x}_i}{|\mathbf{r}_i^T\mathbf{x}_i|\mathbf{x}_i^T\mathbf{x}_i}(|\mathbf{r}_i^T\mathbf{x}_i| - \lambda) \tag{11}$$

As we can see, we get the same solution as the first line in 4.

4.2 Task 2 - Convergence of the coordinate descent solver

In this task, we need to show that, for an **orthogonal** regression matrix, $\hat{w}_i^{(j)}$ does not depend on previous estimates $\hat{\mathbf{w}}$ but only on $\mathbf{t}$, $\mathbf{x}_i$ and λ.

Here we should remember that by considering the simplified case where the regression matrix is an orthonormal basis we can write the following

$$\mathbf{X}^T\mathbf{X} = \mathbf{I}_N \implies \mathbf{x}_i^T\mathbf{x}_j = \begin{cases} 1 & \text{when } i = j \\ 0 & \text{when } i \neq j \end{cases} \tag{12}$$

Starting from Equation 4 and taking in consideration the fact in 12, we can write

$$\hat{w}_i^{(j)} = \begin{cases} \pm(|\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}| - \lambda) & \text{when} \quad |\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}| > \lambda \\ \\ 0 \qquad \text{when} & |\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}| \leq \lambda \end{cases} \tag{13}$$

Now by plugging the value of $\mathbf{r}_i$ in 5 into the first line, we write

$$\begin{aligned} \hat{w}_i^{(j)} &= \pm(|\mathbf{x}_i^T\mathbf{r}_i^{(j-1)}| - \lambda) \\ &= \pm(|\mathbf{x}_i^T(\mathbf{t} - \sum_{l<i}\mathbf{x}_l\hat{w}_l^{(j)} - \sum_{l>i}\mathbf{x}_l\hat{w}_l^{(j-1)})| - \lambda) \\ &= \pm(|\mathbf{x}_i^T\mathbf{t} - \sum_{l<i}\mathbf{x}_i^T\mathbf{x}_l\hat{w}_l^{(j)} - \sum_{l>i}\mathbf{x}_i^T\mathbf{x}_l\hat{w}_l^{(j-1)})| - \lambda) \\ &= \pm(|\mathbf{x}_i^T\mathbf{t}| - \lambda) \end{aligned} \tag{14}$$

where the sums are reduced to zero because of the orthogonality when indices are different. So we get the expression that shows that $\hat{w}_i^{(j)}$ does not depend on previous estimates but only on $\mathbf{t}$, $\mathbf{x}_i$ and λ, and we can say that

$$\hat{w}_i^{(2)} - \hat{w}_i^{(1)} = 0, \quad \forall i$$

.

4.3 Task 3 - LASSO estimate's bias

In this task, we assume that the data $\mathbf{t}$ is truly a noisy (stochastic) example from the hypothesis space defined by the regression matrix, that is

$$\mathbf{t} = \mathbf{X}\mathbf{w}^* + \mathbf{e}, \qquad \text{where } \mathbf{e} \in N(\mathbf{0}_N, \sigma\mathbf{I}_N) \tag{15}$$

In this way, we need to show that by estimating $\mathbf{w}$ using LASSO, the bias induced into it, for an orthogonal regression matrix and noisy $\mathbf{t}$, is given by

$$\lim_{\sigma=0}\mathbb{E}(\hat{w}_i^{(1)} - w_i^*) = \begin{cases} -\lambda & \text{when } w_i^* > \lambda \\ -w_i^*, & \text{when } |w_i^*| \leq \lambda \\ -\lambda & \text{when } w_i^* < -\lambda \end{cases} \tag{16}$$

By considering the first line in 4 as two seperate cases, we can write that when $\mathbf{x}_i^T \mathbf{r}_i^{(j-1)} > \lambda$:

$$\begin{aligned}
\mathbb{E}(\hat{w}_i^{(1)} - w_i^*) &= E(\frac{\mathbf{x}_i^T \mathbf{r}_i^{(0)} - \lambda}{\mathbf{x}_i^T \mathbf{x}_i} - w_i^*) \\
&= \mathbb{E}(\mathbf{x}_i^T \mathbf{r}_i^{(0)} - \lambda - w_i^*) \\
&= \mathbb{E}(\mathbf{x}_i^T (\mathbf{t} - \sum_{l<i} \mathbf{x}_l \hat{w}_l^{(1)} - \sum_{l>i} \mathbf{x}_l \hat{w}_l^{(0)}) - \lambda - w_i^*) \\
&= \mathbb{E}(\mathbf{x}_i^T \mathbf{t} - \lambda - w_i^*) \\
&= \mathbb{E}(\mathbf{x}_i^T (\mathbf{x}_i w_i^* + \mathbf{e}) - \lambda - w_i^*) \\
&= \mathbb{E}(w_i^* + \mathbf{x}_i^T \mathbf{e} - \lambda - w_i^*) \\
&= \mathbb{E}(\mathbf{x}_i^T \mathbf{e}) - \lambda
\end{aligned} \tag{17}$$

Here we can conclude that because of the Gaussian distribution noise (white noise):

$$\mathbf{e} \in N(\mathbf{0}_N, \sigma \mathbf{I}_N) \implies \lim_{\sigma=0} \mathbb{E}(\hat{w}_i^{(1)} - w_i^*) = -\lambda$$

In similar reasoning, we can show for the second case when $\mathbf{x}_i^T \mathbf{r}_i^{(j-1)} < \lambda$, but now we have

$$\hat{w}_i^{(1)} = \frac{\lambda - \mathbf{x}_i^T \mathbf{r}_i^{(0)}}{\mathbf{x}_i^T \mathbf{x}_i}$$

$$\begin{aligned}
\mathbb{E}(\hat{w}_i^{(1)} - w_i^*) &= E(\frac{\lambda - \mathbf{x}_i^T \mathbf{r}_i^{(0)}}{\mathbf{x}_i^T \mathbf{x}_i} - w_i^*) \\
&= \mathbb{E}(\lambda - \mathbf{x}_i^T \mathbf{r}_i^{(0)} - w_i^*) \\
&= \mathbb{E}(\lambda - \mathbf{x}_i^T (\mathbf{t} - \sum_{l<i} \mathbf{x}_l \hat{w}_l^{(1)} - \sum_{l>i} \mathbf{x}_l \hat{w}_l^{(0)}) - w_i^*) \\
&= \mathbb{E}(\lambda - \mathbf{x}_i^T \mathbf{t} - w_i^*) \\
&= \mathbb{E}(\lambda - \mathbf{x}_i^T (\mathbf{x}_i w_i^* + \mathbf{e}) - w_i^*) \\
&= \mathbb{E}(\lambda - w_i^* + \mathbf{x}_i^T \mathbf{e} - w_i^*) \\
&= \mathbb{E}(\mathbf{x}_i^T \mathbf{e}) + \lambda - 2w_i^*
\end{aligned} \tag{18}$$

Here we can conclude that because of the Gaussian distribution noise (white noise):

$$\mathbf{e} \in N(\mathbf{0}_N, \sigma \mathbf{I}_N) \implies \lim_{\sigma=0} \mathbb{E}(\hat{w}_i^{(1)} - w_i^*) = \lambda - 2w_i^*$$

For the third case, i.e. when $\hat{w}_i^{(1)} = 0$ we get

$$\lim_{\sigma=0} \mathbb{E}(\hat{w}_i^{(1)} - w_i^*) = \mathbb{E}(-w_i^*) = -w_i^*$$

Looking at 16, we can see the fact that the bias is governed by the selection of λ.

5 Hyperparameter-learning via K-fold cross-validation

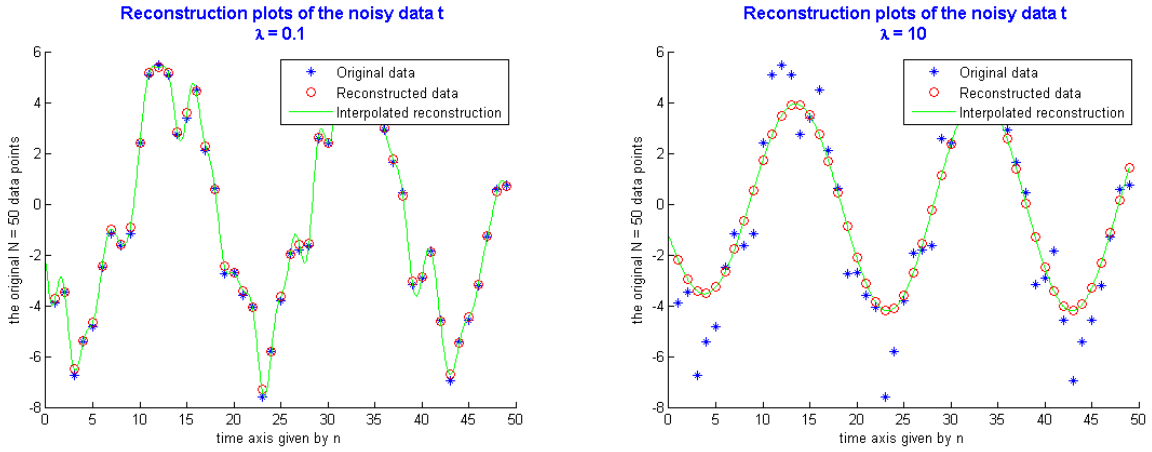
In this section, we consider LASSO regression for the noisy data $\mathbf{t}$, where the regression matrix consists of 500 candidate sine and cosine pairs, i.e.,

$$\begin{aligned}
\mathbf{X} &= [\mathbf{X}_1 \quad \mathbf{X}_2 \quad \cdots \quad \mathbf{X}_{500}] \\
\mathbf{X}_i &= [\mathbf{u}_i \quad \mathbf{v}_i] \\
\mathbf{u}_i &= [\sin(2\pi f_i 0) \quad \sin(2\pi f_i 0) \quad \cdots \quad \sin(2\pi f_i (N-1))]^T \\
\mathbf{v}_i &= [\cos(2\pi f_i 0) \quad \cos(2\pi f_i 0) \quad \cdots \quad \cos(2\pi f_i (N-1))]^T
\end{aligned} \tag{19}$$

and where f_i are the 500 frequency candidates, equally spaced on the interval [0.02, 0.48].

5.1 Task 4 - Implementation of the cyclic coordinate descent for LASSO solution

In this task, we need to Implement a cyclic coordinate descent solver for the coordinate-wise LASSO solution in 4, using data $\mathbf{t}$ and regression matrix $\mathbf{X}$ in `A1_data.mat`. After implementing the the function `lasso_ccd()`, three variants of reconstruction plots, one for each of the values of the hyperparameter $\lambda \in \{0.1, 10, 0.5\}$ are made and shown in Figures 6 and 7.



(a) (b)

Figure 6: a) The reconstruction plot for the hyperparameter $\lambda = 0.1$. b) The reconstruction plot for the hyperparameter $\lambda = 10$.

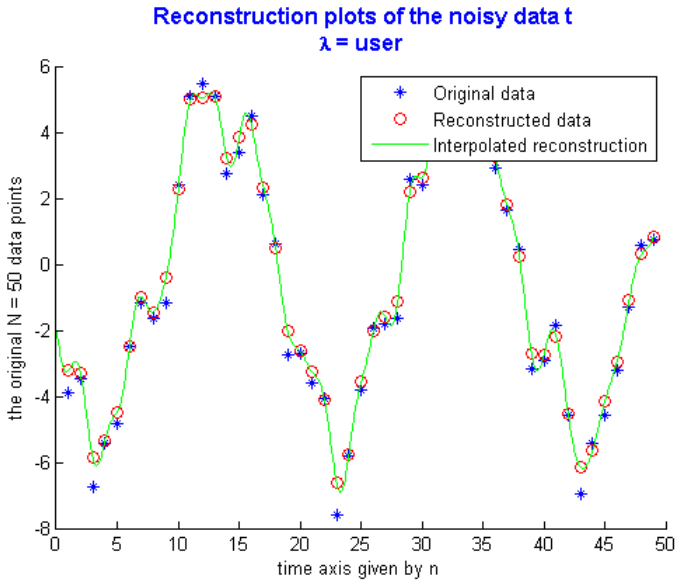


Figure 7: The reconstruction plot for the hyperparameter $\lambda = 0.5$

	$\lambda = 0.1$	$\lambda = 10$	$\lambda = 0.5$
nb of non-zero coordinates	80	0	25

Table 1: The number of non-zero coordinates for the values of the hyperparameter $\lambda \in \{0.1, 10, 0.5\}$.

5.2 algorithm - K-fold cross-validation for LASSO

The LASSO's penalty term promotes **sparsity** in w by setting coordinates with small magnitude to zero, thereby reducing the **degrees of freedom** for the problem. The choice of the hyperparameter λ thus becomes critical. To select λ optimally, one may use the K-fold cross-validation scheme. As discussed

above, in this approach, the training data is randomly split in K non-overlapping and equally sized folds, i.e. subsets, where (K-1) folds are used for estimation, and the remaining fold is used for validation.

To measure the performance of the estimated model when used to predict on the validation data, the `root mean squared error` on validation `RMSEest` term

$$RMSE_{val}(\lambda_j) = \sqrt{K^{-1} \sum_{n=1} N_{val}^{-1} \quad \|\mathbf{t}(I_k) - \mathbf{X}(I_k)\hat{\mathbf{w}}^{[k]}(\lambda_j)\|^2} \tag{20}$$

where

$$\hat{\lambda} = \lambda_p, \quad p = \arg\min_j RMSE_{val}(\lambda_j)$$

Algorithm 1 K-fold cross-validation for LASSO

Select grid of hyperparameters $\lambda_j \geq 0$, $\forall j$, and $K > 1$.
Intitialize $SE_{\text{val}}(k, \lambda_j) = 0$, $\forall k, j$.
Randomize validation indices $\mathcal{I}_k$, $\forall k$.
for k = 1, ..., K **do**
  for j = 1, ..., J **do**
    Calculate LASSO estimate $\hat{\mathbf{w}}(\lambda_j)^{[k]}$ on the k:th fold's estimation data.
    Set $SE_{\text{val}}(k, \lambda_j) = N_{\text{val}}^{-1} \left\|\mathbf{t}(\mathcal{I}_k) - \mathbf{X}(\mathcal{I}_k)\hat{\mathbf{w}}^{[k]}(\lambda_j)\right\|_2^2$.
  end for
end for
Calculate $RMSE_{\text{val}}(\lambda_j) = \sqrt{K^{-1} \sum_{k=1}^{K} SE_{\text{val}}(k, \lambda_j)}$.
Select $\hat{\lambda}$ as the λ_j which minimizes $RMSE_{\text{val}}$.

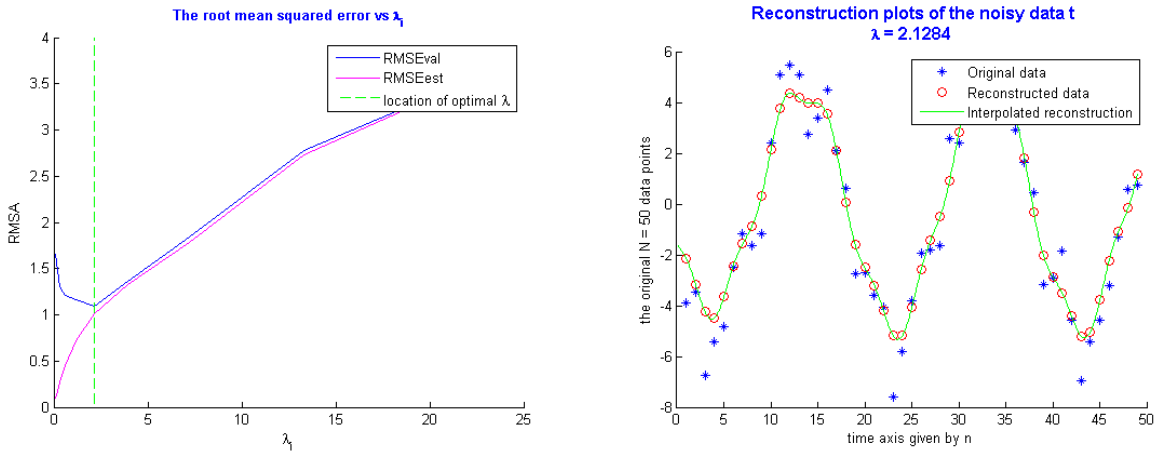
Figure 8: The algorithm that outlines our implementation of a K-fold cross-validation scheme for the LASSO estimator

To display the generalization gap, the corresponding root mean squared error on estimation data, RMSEest becomes

$$RMSE_{est}(\lambda_j) = \sqrt{K^{-1} \sum_{n=1} N_{est}^{-1} \quad \|\mathbf{t}(I_k^c) - \mathbf{X}(I_k^c)\hat{\mathbf{w}}^{[k]}(\lambda_j)\|^2} \tag{21}$$

5.3 Task 5 - Implementation the K-fold cross-validation for LASSO solver

In this task, the function `lasso_cv` is implemented according the algorithm illustrated in Figure 8. A plot illustrating the RMSEval and RMSEest is shown in Figure 9a.



(a) (b)

Figure 9: a) plots illustrating the RMSEval and RMSEest vs. λ_j and showing the vertical line at the location of the optimal λ. b) The reconstruction plot for the optimal hyperparameter $\hat{\lambda}$.

6 Denoising of an audio excerpt

In this section, we need to perform denoising of a noisy recording of piano music using the LASSO and a regression matrix of sine and cosine pairs. The idea is thus that, using a hypothesis space with frequency functions, and by virtue of the sparse estimates obtained using the LASSO, only the strong narrowband components in the music will be modeled, while the broadband noise is cancelled.

Thus we need to implement a K-fold cross-validation scheme for the **multi-frame** audio excerpt `Ttrain` in order to find one optimal $\hat{\lambda}$ for all frames.

6.1 Narrowband and broadband sounds

In general, the sounds of importance that we perceive in our daily lives are narrowband, meaning that for any short (< 100 ms) sequence, the spectral representation is sparse, i.e., it may be modeled using only a small number of pure frequencies. This is typically true for, e.g., speech and music. As a contrast, sounds of small importance, or even nuisance sounds, are often broadband, such as, e.g., noise and interference, containing many or most frequencies across the spectrum.

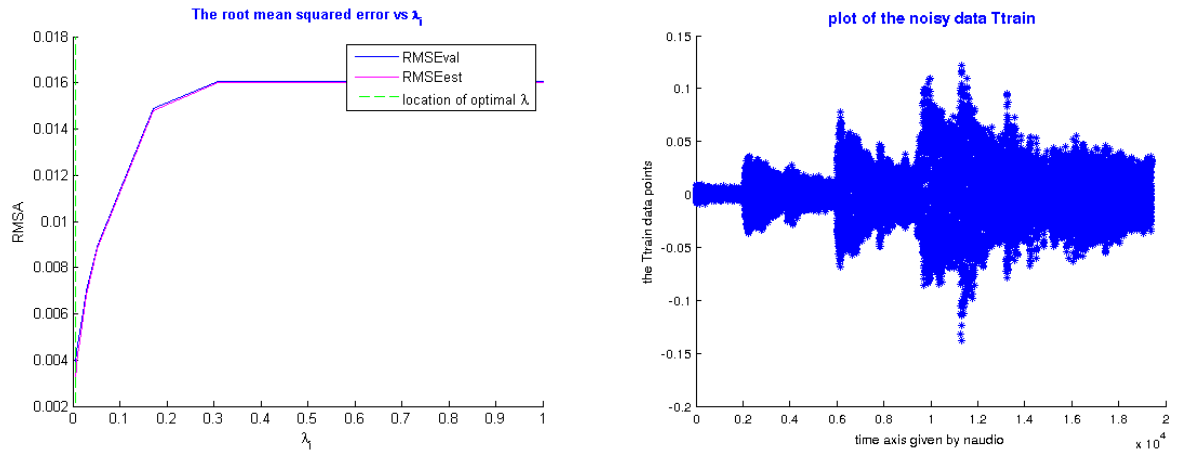
6.2 An excerpt of piano music

The data archive `A1_data.mat` contains an excerpt of piano music, 5 seconds in total, divided into two data sets, `Ttrain`, and `Ttest`, where the former is for estimation and validation, while the latter is used only for testing (not validation).

The training data is much too large (and non-stationary) to be modeled in one regression problem. Instead, *the training data is to be divided into frames 40 ms long.* For each frame, a (possibly unique) LASSO solution may be obtained, which if used to reconstruct the data can be listened to. The same regression matrix, `Xaudio` is used for all frames.

6.3 Task 6 - Implementation the K-fold cross-validation for the multi-frame audio excerpt

After implementing the function `multiframe_lasso_cv()`, and running it on the multi-frame Ttrain we obtained the results illustrated with the plots in Figure 10a.



(a) (b)

Figure 10: a) plots illustrating the RMSEval and RMSEest vs. λ_j for the multi-frame audio excerpt Ttrain and showing the vertical line at the location of the optimal $\hat{\lambda} = 0.005$. b) Plot for Ttrain data.

6.4 Task 7 - Denoising the test data

In this task, we should denoise teh test data `Ttest` by running the provided function `lasso_denoise()` on the obtained $\hat{\lambda}$, we have trained in Task 6. By listening to the output, using the Matlab-function `soundsc(Ttest,fs)`, where fs is the sampling rate, we notice the music became better denoised. The results are saved in the file `denoised_audio` and included in the submission.

References

[1] Bishop, C. M.: Pattern Recognition and Machine Learning. Springer, 2006. Online version available at https://www.microsoft.com/en-us/research/people/cmbishop/#!prml-book (March 2019).

[2] Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep Learning.* MIT Press, 2016, ISBN: 9780262035613. HTLM version available at https://www.deeplearningbook.org/ (March 2019).

[3] R. Sutton, A.G. Barto: Reinforcement Learning, An Introduction. MIT Press, 2017. Draft available at http://incompleteideas.net/sutton/book/the-book-2nd.html (March 2019).

[4] T. Hastie, R. Tibshirani, J. Friedman: The elements of statistical learning, data mining, inference and prediction, 2nd edition, Springer, 2009, online version available at https://web.stanford.edu/~hastie/Papers/ESLII.pdf (March 2019).

[5] Andrew Ng, Stanford course CS229: Machine Learning. Available online at cs229.stanford.edu/notes/cs229-notes1.pdf

[6] Roger Grosse, UToronto course CSC 411 Fall 2018: Machine Learning and Data Mining. Available online at http://www.cs.toronto.edu/~rgrosse/courses/csc411_f18/