Machine Learning

A Bayesian and Optimization Perspective

Academic Press, 2015

Sergios Theodoridis[1]

[1]Dept. of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece.

Winter 2015, Version II

Chapter 6
The Least-Squares Family

Least-Squares Linear Regression: A Geometric Perspective

- Our starting point is the regression model. Condider a set of observations,

$$y_n = \boldsymbol{\theta}^T \boldsymbol{x}_n + \eta_n, \ n = 1, 2, \ldots, N, \ y_n \in \mathbb{R}, \ \boldsymbol{x}_n \in \mathbb{R}^l, \ \boldsymbol{\theta} \in \mathbb{R}^l,$$

  where η_n denotes the (unobserved) values of a zero mean noise source.

- The task is to obtain an estimate of the unknown parameter vector, θ, so that

$$\hat{\boldsymbol{\theta}}_{LS} = \arg\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} (y_n - \boldsymbol{\theta}^T \boldsymbol{x}_n)^2.$$

  We have assumed that our data have been centered around their sample means; alternatively, the intercept, θ_0, can be absorbed in $\boldsymbol{\theta}$ with a corresponding increase in the dimensionality of $\boldsymbol{x}_n$.

- Define,

$$\boldsymbol{y} = [y_1, \ldots, y_N]^T \in \mathbb{R}^N, \quad X := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N \times l}.$$

  Then, the optimizing task can equivalently be written as,

$$\hat{\boldsymbol{\theta}}_{LS} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{e}\|^2, \text{ where } \boldsymbol{e} := \boldsymbol{y} - X\boldsymbol{\theta},$$

  and $\|\cdot\|^2$ denotes the square Euclidean norm, which measures the distance between the respective vectors in $\mathbb{R}^N$, i.e., $\boldsymbol{y}$ and $X\boldsymbol{\theta}$.

Least-Squares Linear Regression: A Geometric Perspective

- Our starting point is the regression model. Condider a set of observations,

$$y_n = \boldsymbol{\theta}^T \boldsymbol{x}_n + \eta_n, \ n = 1, 2, \ldots, N, \ y_n \in \mathbb{R}, \ \boldsymbol{x}_n \in \mathbb{R}^l, \ \boldsymbol{\theta} \in \mathbb{R}^l,$$

  where η_n denotes the (unobserved) values of a zero mean noise source.
- The task is to obtain an estimate of the unknown parameter vector, $\boldsymbol{\theta}$, so that

$$\hat{\boldsymbol{\theta}}_{LS} = \arg\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} (y_n - \boldsymbol{\theta}^T \boldsymbol{x}_n)^2.$$

  We have assumed that our data have been centered around their sample means; alternatively, the intercept, θ_0, can be absorbed in $\boldsymbol{\theta}$ with a corresponding increase in the dimensionality of $\boldsymbol{x}_n$.
- Define,

$$\boldsymbol{y} = [y_1, \ldots, y_N]^T \in \mathbb{R}^N, \quad X := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N \times l}.$$

  Then, the optimizing task can equivalently be written as,

$$\hat{\boldsymbol{\theta}}_{LS} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{e}\|^2, \text{ where } \boldsymbol{e} := \boldsymbol{y} - X\boldsymbol{\theta},$$

  and $\|\cdot\|^2$ denotes the square Euclidean norm, which measures the distance between the respective vectors in $\mathbb{R}^N$, i.e., $\boldsymbol{y}$ and $X\boldsymbol{\theta}$.

Least-Squares Linear Regression: A Geometric Perspective

- Our starting point is the regression model. Condider a set of observations,

$$y_n = \boldsymbol{\theta}^T \boldsymbol{x}_n + \eta_n, \ n = 1, 2, \ldots, N, \ y_n \in \mathbb{R}, \ \boldsymbol{x}_n \in \mathbb{R}^l, \ \boldsymbol{\theta} \in \mathbb{R}^l,$$

  where η_n denotes the (unobserved) values of a zero mean noise source.
- The task is to obtain an estimate of the unknown parameter vector, $\boldsymbol{\theta}$, so that

$$\hat{\boldsymbol{\theta}}_{LS} = \arg\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} (y_n - \boldsymbol{\theta}^T \boldsymbol{x}_n)^2.$$

  We have assumed that our data have been centered around their sample means; alternatively, the intercept, θ_0, can be absorbed in $\boldsymbol{\theta}$ with a corresponding increase in the dimensionality of $\boldsymbol{x}_n$.
- Define,

$$\boldsymbol{y} = [y_1, \ldots, y_N]^T \in \mathbb{R}^N, \quad X := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N \times l}.$$

  Then, the optimizing task can equivalently be written as,

$$\hat{\boldsymbol{\theta}}_{LS} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{e}\|^2, \text{ where } \boldsymbol{e} := \boldsymbol{y} - X\boldsymbol{\theta},$$

  and $\|\cdot\|^2$ denotes the square Euclidean norm, which measures the distance between the respective vectors in $\mathbb{R}^N$, i.e., $\boldsymbol{y}$ and $X\boldsymbol{\theta}$.

Least-Squares Linear Regression: A Geometric Perspective

- Let us denote as $\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c \in \mathbb{R}^N$ the columns of X, i.e.,

$$X = [\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c].$$

Then we can write,

$$\hat{\boldsymbol{y}} := X\boldsymbol{\theta} = \sum_{i=1}^{l} \theta_i \boldsymbol{x}_i^c, \text{ and } \boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}.$$

Obviously, $\hat{\boldsymbol{y}}$ represents a vector that lies in the span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$. Thus, our task is equivalent with selecting $\boldsymbol{\theta}$ so that the error vector between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ to have **minimum norm**.

- According to the Euclidean theorem of orthogonality, this is achieved if $\hat{\boldsymbol{y}}$ is chosen as the **orthogonal projection** of $\boldsymbol{y}$ onto the span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$.

Least-Squares Linear Regression: A Geometric Perspective

- Let us denote as $\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c \in \mathbb{R}^N$ the columns of X, i.e.,

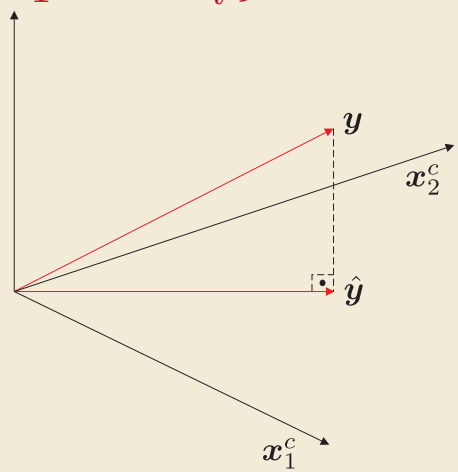
$$X = [\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c].$$

Then we can write,

$$\hat{\boldsymbol{y}} := X\boldsymbol{\theta} = \sum_{i=1}^{l} \theta_i \boldsymbol{x}_i^c, \text{ and } \boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}.$$

Obviously, $\hat{\boldsymbol{y}}$ represents a vector that lies in the span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$. Thus, our task is equivalent with selecting $\boldsymbol{\theta}$ so that the error vector between $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ to have **minimum norm**.

- According to the Euclidean theorem of orthogonality, this is achieved if $\hat{\boldsymbol{y}}$ is chosen as the **orthogonal projection** of $\boldsymbol{y}$ onto the span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$.



The LS estimate is chosen so that $\hat{\boldsymbol{y}}$ to be the orthogonal projection of $\boldsymbol{y}$ onto the span $\{\boldsymbol{x}_1^c, \boldsymbol{x}_2^c\}$; that is, the columns of X.

Least-Squares Linear Regression: A Geometric Perspective

- From the theory of projections, it is known that the projection of a vector $\boldsymbol{y}$ onto the space spanned by the columns of an $\mathbb{R}^{N\times l}$ matrix, X, is given by

$$\hat{\boldsymbol{y}} = X(X^TX)^{-1}X^T\boldsymbol{y},$$

  assuming that X^TX is invertible.

- The Moore-Penrose pseudo-inverse: Let X be a tall matrix; its pseudo inverse matrix is defined by

$$X^\dagger := (X^TX)^{-1}X^T.$$

- Employing the previous definition, we can write

$$\hat{\boldsymbol{\theta}}_{LS} = X^\dagger \boldsymbol{y}.$$

- Note that the pseudo-inverse is a generalization of the notion of the inverse of a square matrix. Indeed, if X is square, then it is readily seen that the pseudo inverse coincides with X^{-1}. For complex data, the only difference is that transposition is replaced by the Hermitian one.

Least-Squares Linear Regression: A Geometric Perspective

- From the theory of projections, it is known that the projection of a vector $\boldsymbol{y}$ onto the space spanned by the columns of an $\mathbb{R}^{N\times l}$ matrix, X, is given by

$$\hat{\boldsymbol{y}} = X(X^TX)^{-1}X^T\boldsymbol{y},$$

  assuming that X^TX is invertible.

- The Moore-Penrose pseudo-inverse: Let X be a tall matrix; its pseudo inverse matrix is defined by

$$X^\dagger := (X^TX)^{-1}X^T.$$

- Employing the previous definition, we can write

$$\hat{\boldsymbol{\theta}}_{LS} = X^\dagger \boldsymbol{y}.$$

- Note that the pseudo-inverse is a generalization of the notion of the inverse of a square matrix. Indeed, if X is square, then it is readily seen that the pseudo inverse coincides with X^{-1}. For complex data, the only difference is that transposition is replaced by the Hermitian one.

Least-Squares Linear Regression: A Geometric Perspective

- From the theory of projections, it is known that the projection of a vector $\boldsymbol{y}$ onto the space spanned by the columns of an $\mathbb{R}^{N\times l}$ matrix, X, is given by

$$\hat{\boldsymbol{y}} = X(X^TX)^{-1}X^T\boldsymbol{y},$$

  assuming that X^TX is invertible.

- The Moore-Penrose pseudo-inverse: Let X be a tall matrix; its pseudo inverse matrix is defined by

$$X^\dagger := (X^TX)^{-1}X^T.$$

- Employing the previous definition, we can write

$$\hat{\boldsymbol{\theta}}_{LS} = X^\dagger\boldsymbol{y}.$$

- Note that the pseudo-inverse is a generalization of the notion of the inverse of a square matrix. Indeed, if X is square, then it is readily seen that the pseudo inverse coincides with X^{-1}. For complex data, the only difference is that transposition is replaced by the Hermitian one.

Least-Squares Linear Regression: A Geometric Perspective

- From the theory of projections, it is known that the projection of a vector $\boldsymbol{y}$ onto the space spanned by the columns of an $\mathbb{R}^{N\times l}$ matrix, X, is given by

$$\hat{\boldsymbol{y}} = X(X^TX)^{-1}X^T\boldsymbol{y},$$

  assuming that X^TX is invertible.

- The Moore-Penrose pseudo-inverse: Let X be a tall matrix; its pseudo inverse matrix is defined by

$$X^\dagger := (X^TX)^{-1}X^T.$$

- Employing the previous definition, we can write

$$\hat{\boldsymbol{\theta}}_{LS} = X^\dagger\boldsymbol{y}.$$

- Note that the pseudo-inverse is a generalization of the notion of the inverse of a square matrix. Indeed, if X is square, then it is readily seen that the pseudo inverse coincides with X^{-1}. For complex data, the only difference is that transposition is replaced by the Hermitian one.

Statistical Properties of the LS Estimator

- Assume that there exists a true (yet unknown) parameter/weight vector, $\boldsymbol{\theta}_o$, that generates the output (dependent) random variables (stacked in a random vector $\mathbf{y} \in \mathbb{R}^N$), according to the model,

$$\mathbf{y} = X\boldsymbol{\theta}_o + \boldsymbol{\eta},$$

where $\boldsymbol{\eta}$ is a zero mean noise vector. Observe that, we have assumed that X is fixed and not random; that is, the randomness underlying the output variables, $\mathbf{y}$, is due solely to the noise. Under the previously stated assumptions, the following properties hold:

- The LS Estimator is Unbiased: The LS estimator for the parameters is given by,

$$\begin{aligned} \hat{\theta}_{LS} &= (X^T X)^{-1} X^T \mathbf{y}, \\ &= (X^T X)^{-1} X^T (X\boldsymbol{\theta}_o + \boldsymbol{\eta}) = \boldsymbol{\theta}_o + (X^T X)^{-1} X^T \boldsymbol{\eta}, \quad (1) \end{aligned}$$

or

$$\mathbb{E}[\hat{\theta}_{LS}] = \boldsymbol{\theta}_o + (X^T X)^{-1} X^T \mathbb{E}[\boldsymbol{\eta}] = \boldsymbol{\theta}_o,$$

which proves the claim.

Statistical Properties of the LS Estimator

- Assume that there exists a true (yet unknown) parameter/weight vector, $\boldsymbol{\theta}_o$, that generates the output (dependent) random variables (stacked in a random vector $\mathbf{y} \in \mathbb{R}^N$), according to the model,

$$\mathbf{y} = X\boldsymbol{\theta}_o + \boldsymbol{\eta},$$

where $\boldsymbol{\eta}$ is a zero mean noise vector. Observe that, we have assumed that X is fixed and not random; that is, the randomness underlying the output variables, $\mathbf{y}$, is due solely to the noise. Under the previously stated assumptions, the following properties hold:

- The LS Estimator is Unbiased: The LS estimator for the parameters is given by,

$$\begin{aligned} \hat{\theta}_{LS} &= (X^T X)^{-1} X^T \mathbf{y}, \\ &= (X^T X)^{-1} X^T (X\boldsymbol{\theta}_o + \boldsymbol{\eta}) = \boldsymbol{\theta}_o + (X^T X)^{-1} X^T \boldsymbol{\eta}, \quad (1) \end{aligned}$$

or

$$\mathbb{E}[\hat{\theta}_{LS}] = \boldsymbol{\theta}_o + (X^T X)^{-1} X^T \mathbb{E}[\boldsymbol{\eta}] = \boldsymbol{\theta}_o,$$

which proves the claim.

Statistical Properties of the LS Estimator

- Covariance Matrix of the LS Estimator: Let, in addition to the previously adopted assumptions, that

$$\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^T] = \sigma_\eta^2 I.$$

That is, the source generating the noise samples is white. By the definition of the covariance matrix, we get

$$\Sigma_{\hat{\theta}_{LS}} = \mathbb{E}\Big[(\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o)(\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o)^T\Big],$$

and substituting $\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o$ from (1), we obtain

$$\begin{aligned}
\Sigma_{\hat{\theta}_{LS}} &= \mathbb{E}\Big[(X^TX)^{-1}X^T\boldsymbol{\eta}\boldsymbol{\eta}^TX(X^TX)^{-1}\Big] \\
&= (X^TX)^{-1}X^T\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^T]X(X^TX)^{-1} \\
&= \sigma_\eta^2(X^TX)^{-1}. \qquad (2)
\end{aligned}$$

- Note that, for large values of N, we can write

$$X^TX = \sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T \approx N\Sigma_x, \text{ where } \Sigma_x := \mathbb{E}[\mathbf{x}_n\mathbf{x}_n^T] \approx \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T$$

- Thus, for large values of N, we can write

$$\Sigma_{\hat{\theta}_{LS}} \approx \frac{\sigma_\eta^2}{N}\Sigma_x^{-1}.$$

Statistical Properties of the LS Estimator

- Covariance Matrix of the LS Estimator: Let, in addition to the previously adopted assumptions, that

$$\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^T] = \sigma_\eta^2 I.$$

That is, the source generating the noise samples is white. By the definition of the covariance matrix, we get

$$\Sigma_{\hat{\theta}_{LS}} = \mathbb{E}\Big[(\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o)(\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o)^T\Big],$$

and substituting $\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o$ from (1), we obtain

$$\begin{aligned} \Sigma_{\hat{\theta}_{LS}} &= \mathbb{E}\Big[(X^TX)^{-1}X^T\boldsymbol{\eta}\boldsymbol{\eta}^TX(X^TX)^{-1}\Big] \\ &= (X^TX)^{-1}X^T\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^T]X(X^TX)^{-1} \\ &= \sigma_\eta^2(X^TX)^{-1}. \end{aligned} \tag{2}$$

- Note that, for large values of N, we can write

$$X^TX = \sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T \approx N\Sigma_x, \text{ where } \Sigma_x := \mathbb{E}[\mathbf{x}_n\mathbf{x}_n^T] \approx \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T$$

- Thus, for large values of N, we can write

$$\Sigma_{\hat{\theta}_{LS}} \approx \frac{\sigma_\eta^2}{N}\Sigma_x^{-1}.$$

Statistical Properties of the LS Estimator

- Covariance Matrix of the LS Estimator: Let, in addition to the previously adopted assumptions, that

$$\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^T] = \sigma_\eta^2 I.$$

That is, the source generating the noise samples is white. By the definition of the covariance matrix, we get

$$\Sigma_{\hat{\theta}_{LS}} = \mathbb{E}\Big[(\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o)(\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o)^T\Big],$$

and substituting $\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_o$ from (1), we obtain

$$\begin{aligned} \Sigma_{\hat{\theta}_{LS}} &= \mathbb{E}\Big[(X^TX)^{-1}X^T\boldsymbol{\eta}\boldsymbol{\eta}^TX(X^TX)^{-1}\Big] \\ &= (X^TX)^{-1}X^T\mathbb{E}[\boldsymbol{\eta}\boldsymbol{\eta}^T]X(X^TX)^{-1} \\ &= \sigma_\eta^2(X^TX)^{-1}. \end{aligned} \tag{2}$$

- Note that, for large values of N, we can write

$$X^TX = \sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T \approx N\Sigma_x, \text{ where } \Sigma_x := \mathbb{E}[\mathbf{x}_n\mathbf{x}_n^T] \approx \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^T$$

- Thus, for large values of N, we can write

$$\Sigma_{\hat{\theta}_{LS}} \approx \frac{\sigma_\eta^2}{N}\Sigma_x^{-1}.$$

Statistical Properties of the LS Estimator

- In other words, under the adopted assumptions, the LS estimator is not only unbiased but its covariance matrix tends asymptotically to zero. That is, with high probability, the estimate $\hat{\boldsymbol{\theta}}_{LS}$, which is obtained via a large number of measurements, will be close to the true value, $\boldsymbol{\theta}_o$.
- Viewing it slightly differently, note that the LS solution tends to the MSE solution, which is discussed in Chapter 3. Indeed, for the case of centered data,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T = \Sigma_x,$$

and

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n y_n = \mathbb{E}[\mathrm{xy}] = \boldsymbol{p}.$$

Moreover, we know that for the linear regression modeling case, the normal equations, $\Sigma_{\mathrm{x}}\boldsymbol{\theta} = \boldsymbol{p}$, result in the solution $\boldsymbol{\theta} = \boldsymbol{\theta}_o$.

- The LS Estimator is BLUE in the Presence of White Noise: Let $\hat{\theta}$ be any other **linear unbiased** estimator. Under the white noise assumption, the following holds true:

$$\mathbb{E}\Big[(\hat{\theta} - \boldsymbol{\theta}_o)^T(\hat{\theta} - \boldsymbol{\theta}_o)\Big] \geq \mathbb{E}\Big[(\hat{\theta}_{LS} - \boldsymbol{\theta}_o)^T(\hat{\theta}_{LS} - \boldsymbol{\theta}_o)\Big].$$

Statistical Properties of the LS Estimator

- In other words, under the adopted assumptions, the LS estimator is not only unbiased but its covariance matrix tends asymptotically to zero. That is, with high probability, the estimate $\hat{\boldsymbol{\theta}}_{LS}$, which is obtained via a large number of measurements, will be close to the true value, $\boldsymbol{\theta}_o$.
- Viewing it slightly differently, note that the LS solution tends to the MSE solution, which is discussed in Chapter 3. Indeed, for the case of centered data,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T = \Sigma_x,$$

and

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n y_n = \mathbb{E}[\mathbf{xy}] = \boldsymbol{p}.$$

Moreover, we know that for the linear regression modeling case, the normal equations, $\Sigma_{\mathrm{x}}\boldsymbol{\theta} = \boldsymbol{p}$, result in the solution $\boldsymbol{\theta} = \boldsymbol{\theta}_o$.

- The LS Estimator is BLUE in the Presence of White Noise: Let $\hat{\theta}$ be any other **linear unbiased** estimator. Under the white noise assumption, the following holds true:

$$\mathbb{E}\Big[(\hat{\theta} - \boldsymbol{\theta}_o)^T(\hat{\theta} - \boldsymbol{\theta}_o)\Big] \geq \mathbb{E}\Big[(\hat{\theta}_{LS} - \boldsymbol{\theta}_o)^T(\hat{\theta}_{LS} - \boldsymbol{\theta}_o)\Big].$$

Statistical Properties of the LS Estimator

- In other words, under the adopted assumptions, the LS estimator is not only unbiased but its covariance matrix tends asymptotically to zero. That is, with high probability, the estimate $\hat{\boldsymbol{\theta}}_{LS}$, which is obtained via a large number of measurements, will be close to the true value, $\boldsymbol{\theta}_o$.
- Viewing it slightly differently, note that the LS solution tends to the MSE solution, which is discussed in Chapter 3. Indeed, for the case of centered data,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^T = \Sigma_x,$$

and

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n y_n = \mathbb{E}[\mathbf{xy}] = \boldsymbol{p}.$$

Moreover, we know that for the linear regression modeling case, the normal equations, $\Sigma_{\mathrm{x}}\boldsymbol{\theta} = \boldsymbol{p}$, result in the solution $\boldsymbol{\theta} = \boldsymbol{\theta}_o$.

- The LS Estimator is BLUE in the Presence of White Noise: Let $\hat{\theta}$ be any other **linear unbiased** estimator. Under the white noise assumption, the following holds true:

$$\mathbb{E}\Big[(\hat{\theta} - \boldsymbol{\theta}_o)^T(\hat{\theta} - \boldsymbol{\theta}_o)\Big] \geq \mathbb{E}\Big[(\hat{\theta}_{LS} - \boldsymbol{\theta}_o)^T(\hat{\theta}_{LS} - \boldsymbol{\theta}_o)\Big].$$

Statistical Properties of the LS Estimator

- **Proof**: Indeed, from the respective definitions we have

$$\hat{\boldsymbol{\theta}} := H\mathbf{y} \Rightarrow \hat{\boldsymbol{\theta}} = H(X\boldsymbol{\theta}_o + \boldsymbol{\eta}) = HX\boldsymbol{\theta}_o + H\boldsymbol{\eta}.$$

  However, since $\hat{\boldsymbol{\theta}}$ is unbiased, then the previous equation implies that, $HX = I$ and

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o = H\boldsymbol{\eta}.$$

- Thus,

$$\begin{aligned} \Sigma_{\hat{\theta}} &:= \mathbb{E}\Big[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o)(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o)^T\Big] \\ &= \sigma_n^2 H H^T, \end{aligned}$$

- However, taking into account that $HX = I$, it is easily checked out that

$$\sigma_n^2 H H^T = \sigma_n^2 (H - X^\dagger)(H - X^\dagger)^T + \sigma_n^2 (X^T X)^{-1},$$

  where $X^\dagger$ is the respective pseudo-inverse matrix,

- Since $\sigma_n^2 (H - X^\dagger)(H - X^\dagger)^T$ is a semidefinite matrix, its trace is also nonnegative and the claim has been proved, i.e.,

$$\text{trace}\{\sigma_n^2 H H^T\} \geq \text{trace}\{\sigma_n^2 (X^T X)^{-1}\}.$$

  with equality only if $H = X^\dagger = (X^T X)^{-1} X^T$.

Statistical Properties of the LS Estimator

- **Proof**: Indeed, from the respective definitions we have

$$\hat{\boldsymbol{\theta}} := H\mathbf{y} \Rightarrow \hat{\boldsymbol{\theta}} = H(X\boldsymbol{\theta}_o + \boldsymbol{\eta}) = HX\boldsymbol{\theta}_o + H\boldsymbol{\eta}.$$

  However, since $\hat{\boldsymbol{\theta}}$ is unbiased, then the previous equation implies that, $HX = I$ and

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o = H\boldsymbol{\eta}.$$

- Thus,

$$\begin{aligned} \Sigma_{\hat{\theta}} &:= \mathbb{E}\Big[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o)(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o)^T\Big] \\ &= \sigma_n^2 HH^T, \end{aligned}$$

- However, taking into account that $HX = I$, it is easily checked out that

$$\sigma_n^2 HH^T = \sigma_n^2 (H - X^\dagger)(H - X^\dagger)^T + \sigma_n^2 (X^T X)^{-1},$$

  where $X^\dagger$ is the respective pseudo-inverse matrix,

- Since $\sigma_n^2 (H - X^\dagger)(H - X^\dagger)^T$ is a semidefinite matrix, its trace is also nonnegative and the claim has been proved, i.e.,

$$\text{trace}\{\sigma_n^2 HH^T\} \geq \text{trace}\{\sigma_n^2 (X^T X)^{-1}\}.$$

  with equality only if $H = X^\dagger = (X^T X)^{-1} X^T$.

Statistical Properties of the LS Estimator

- **Proof**: Indeed, from the respective definitions we have

$$\hat{\boldsymbol{\theta}} := H\mathbf{y} \Rightarrow \hat{\boldsymbol{\theta}} = H(X\boldsymbol{\theta}_o + \boldsymbol{\eta}) = HX\boldsymbol{\theta}_o + H\boldsymbol{\eta}.$$

  However, since $\hat{\boldsymbol{\theta}}$ is unbiased, then the previous equation implies that, $HX = I$ and

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o = H\boldsymbol{\eta}.$$

- Thus,

$$\begin{aligned} \Sigma_{\hat{\theta}} &:= \mathbb{E}\Big[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o)(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o)^T\Big] \\ &= \sigma_n^2 HH^T, \end{aligned}$$

- However, taking into account that $HX = I$, it is easily checked out that

$$\sigma_n^2 HH^T = \sigma_n^2 (H - X^\dagger)(H - X^\dagger)^T + \sigma_n^2 (X^T X)^{-1},$$

  where $X^\dagger$ is the respective pseudo-inverse matrix,

- Since $\sigma_n^2 (H - X^\dagger)(H - X^\dagger)^T$ is a semidefinite matrix, its trace is also nonnegative and the claim has been proved, i.e.,

$$\text{trace}\{\sigma_n^2 HH^T\} \geq \text{trace}\{\sigma_n^2 (X^T X)^{-1}\}.$$

  with equality only if $H = X^\dagger = (X^T X)^{-1} X^T$.

Statistical Properties of the LS Estimator

- **Proof**: Indeed, from the respective definitions we have

$$\hat{\boldsymbol{\theta}} := H\mathbf{y} \Rightarrow \hat{\boldsymbol{\theta}} = H(X\boldsymbol{\theta}_o + \boldsymbol{\eta}) = HX\boldsymbol{\theta}_o + H\boldsymbol{\eta}.$$

  However, since $\hat{\boldsymbol{\theta}}$ is unbiased, then the previous equation implies that, $HX = I$ and

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o = H\boldsymbol{\eta}.$$

- Thus,

$$\begin{aligned} \Sigma_{\hat{\theta}} &:= \mathbb{E}\Big[(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o)(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_o)^T\Big] \\ &= \sigma_n^2 HH^T, \end{aligned}$$

- However, taking into account that $HX = I$, it is easily checked out that

$$\sigma_n^2 HH^T = \sigma_n^2 (H - X^\dagger)(H - X^\dagger)^T + \sigma_n^2 (X^T X)^{-1},$$

  where $X^\dagger$ is the respective pseudo-inverse matrix,

- Since $\sigma_n^2 (H - X^\dagger)(H - X^\dagger)^T$ is a semidefinite matrix, its trace is also nonnegative and the claim has been proved, i.e.,

$$\text{trace}\{\sigma_n^2 HH^T\} \geq \text{trace}\{\sigma_n^2 (X^T X)^{-1}\}.$$

  with equality only if $H = X^\dagger = (X^T X)^{-1} X^T$.

Statistical Properties of the LS Estimator

- The LS Estimator Achieves the Cramer-Rao Bound for White Gaussian Noise: The concept of the Cramer-Rao lower bound is discussed in Chapter 3. There, it has been stated that, under the zero mean **Gaussian** noise with covariance matrix Σ_n, the efficient estimator is given by

$$\hat{\boldsymbol{\theta}} = (X^T \Sigma_n^{-1} X)^{-1} X^T \Sigma_n^{-1} \mathbf{y},$$

  which for $\Sigma_n = \sigma_n^2 I$ coincides with the LS estimator.

- In other words, under the white Gaussian noise assumption, the LS estimator becomes Minimum Variance Unbiased Estimator. This is a strong result. No other unbiased estimator (not necessarily linear) will do better than the LS one. Note that this result holds true not asymptotically, but also for finite number of samples N. If one wishes to decrease further the Mean Square Error (MSE), then a biased estimator, e.g., via regularization, has to be considered.

Statistical Properties of the LS Estimator

- The LS Estimator Achieves the Cramer-Rao Bound for White Gaussian Noise: The concept of the Cramer-Rao lower bound is discussed in Chapter 3. There, it has been stated that, under the zero mean **Gaussian** noise with covariance matrix Σ_n, the efficient estimator is given by

$$\hat{\boldsymbol{\theta}} = (X^T \Sigma_n^{-1} X)^{-1} X^T \Sigma_n^{-1} \mathbf{y},$$

  which for $\Sigma_n = \sigma_n^2 I$ coincides with the LS estimator.

- In other words, under the white Gaussian noise assumption, the LS estimator becomes Minimum Variance Unbiased Estimator. This is a strong result. No other unbiased estimator (not necessarily linear) will do better than the LS one. Note that this result holds true not asymptotically, but also for finite number of samples N. If one wishes to decrease further the Mean Square Error (MSE), then a biased estimator, e.g., via regularization, has to be considered.

Statistical Properties of the LS Estimator

- Asymptotic Distribution of the LS Estimator: We have already seen that the LS estimator is unbiased and that its covariance matrix is (approximately, for large values of N) inversely proportional to N. Thus, as $N \longrightarrow \infty$, the variance around the true value, $\boldsymbol{\theta}_o$, is becoming increasingly small.
- Furthermore, there is a stronger result, which provides the distribution of the LS estimator for large values of N. Under some general assumptions, e.g., independence of successive observation vectors and that the white noise source is independent of the input, and mobilizing the central limit theorem, it can be shown, that

$$\sqrt{N}(\hat{\theta}_{LS} - \boldsymbol{\theta}_0) \longrightarrow \mathcal{N}(\mathbf{0}, \sigma_\eta^2 \Sigma_x^{-1}),$$

where the limit is meant to be in distribution. Alternatively, for large values of N, we can write that

$$\hat{\theta}_{LS} \sim \mathcal{N}(\boldsymbol{\theta}_0, \frac{\sigma_\eta^2}{N} \Sigma_x^{-1}).$$

In other words, the LS parameter estimator is asymptotically distributed according to the normal distribution.

Statistical Properties of the LS Estimator

- Asymptotic Distribution of the LS Estimator: We have already seen that the LS estimator is unbiased and that its covariance matrix is (approximately, for large values of N) inversely proportional to N. Thus, as $N \longrightarrow \infty$, the variance around the true value, $\boldsymbol{\theta}_o$, is becoming increasingly small.
- Furthermore, there is a stronger result, which provides the distribution of the LS estimator for large values of N. Under some general assumptions, e.g., independence of successive observation vectors and that the white noise source is independent of the input, and mobilizing the central limit theorem, it can be shown, that

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}_0) \longrightarrow \mathcal{N}(\mathbf{0}, \sigma_\eta^2 \Sigma_x^{-1}),$$

where the limit is meant to be in distribution. Alternatively, for large values of N, we can write that

$$\hat{\boldsymbol{\theta}}_{LS} \sim \mathcal{N}(\boldsymbol{\theta}_0, \frac{\sigma_\eta^2}{N} \Sigma_x^{-1}).$$

In other words, the LS parameter estimator is asymptotically distributed according to the normal distribution.

The Recursive Least-Squares Algorithm

- Our focus now turns in presenting a celebrated online algorithm for obtaining the LS solution. To this end, the special structure of $X^T X$ will be taken into account, that leads to substantial computational savings.
- Moreover, when dealing with time recursive (online) techniques, one can also care for time variations of the statistical properties of the involved data. In our formulation, we will allow for such applications and the LS cost will be slightly modified so that to be able to accommodate time varying environments.
- We are going to bring into our notation explicitly the time index, n. Also, we will assume that the time starts at $n = 0$ and the received observations are $(y_n, \boldsymbol{x}_n),\ n = 0, 1, 2, \ldots$. To this end, let us denote the input matrix, at time n, as

$$X_n^T = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n].$$

- The Exponentially Weighted Least-Squares (EWLS) cost function is defined as,

$$J(\boldsymbol{\theta}) := \sum_{i=0}^{n} \beta^{n-i}(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2, \tag{3}$$

where $\beta,\ 0 < \beta \leq 1$ is a user-defined parameter, very close to unity.

The Recursive Least-Squares Algorithm

- Our focus now turns in presenting a celebrated online algorithm for obtaining the LS solution. To this end, the special structure of X^TX will be taken into account, that leads to substantial computational savings.
- Moreover, when dealing with time recursive (online) techniques, one can also care for time variations of the statistical properties of the involved data. In our formulation, we will allow for such applications and the LS cost will be slightly modified so that to be able to accommodate time varying environments.
- We are going to bring into our notation explicitly the time index, n. Also, we will assume that the time starts at $n = 0$ and the received observations are $(y_n, \boldsymbol{x}_n),\ n = 0, 1, 2, \ldots$. To this end, let us denote the input matrix, at time n, as

$$X_n^T = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n].$$

- The Exponentially Weighted Least-Squares (EWLS) cost function is defined as,

$$J(\boldsymbol{\theta}) := \sum_{i=0}^{n} \beta^{n-i}(y_i - \boldsymbol{\theta}^T\boldsymbol{x}_i)^2 + \lambda\beta^{n+1}\|\boldsymbol{\theta}\|^2, \tag{3}$$

where $\beta,\ 0 < \beta \leq 1$ is a user-defined parameter, very close to unity.

The Recursive Least-Squares Algorithm

- Our focus now turns in presenting a celebrated online algorithm for obtaining the LS solution. To this end, the special structure of $X^T X$ will be taken into account, that leads to substantial computational savings.
- Moreover, when dealing with time recursive (online) techniques, one can also care for time variations of the statistical properties of the involved data. In our formulation, we will allow for such applications and the LS cost will be slightly modified so that to be able to accommodate time varying environments.
- We are going to bring into our notation explicitly the time index, n. Also, we will assume that the time starts at $n = 0$ and the received observations are $(y_n, \boldsymbol{x}_n),\ n = 0, 1, 2, \ldots$. To this end, let us denote the input matrix, at time n, as

$$X_n^T = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n].$$

- The Exponentially Weighted Least-Squares (EWLS) cost function is defined as,

$$J(\boldsymbol{\theta}) := \sum_{i=0}^{n} \beta^{n-i}(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2, \tag{3}$$

where $\beta,\ 0 < \beta \leq 1$ is a user-defined parameter, very close to unity.

The Recursive Least-Squares Algorithm

- Our focus now turns in presenting a celebrated online algorithm for obtaining the LS solution. To this end, the special structure of $X^T X$ will be taken into account, that leads to substantial computational savings.
- Moreover, when dealing with time recursive (online) techniques, one can also care for time variations of the statistical properties of the involved data. In our formulation, we will allow for such applications and the LS cost will be slightly modified so that to be able to accommodate time varying environments.
- We are going to bring into our notation explicitly the time index, n. Also, we will assume that the time starts at $n = 0$ and the received observations are $(y_n, \boldsymbol{x}_n),\ n = 0, 1, 2, \ldots$. To this end, let us denote the input matrix, at time n, as

$$X_n^T = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n].$$

- The Exponentially Weighted Least-Squares (EWLS) cost function is defined as,

$$J(\boldsymbol{\theta}) := \sum_{i=0}^{n} \beta^{n-i}(y_i - \boldsymbol{\theta}^T \boldsymbol{x}_i)^2 + \lambda \beta^{n+1} \|\boldsymbol{\theta}\|^2, \tag{3}$$

where $\beta,\ 0 < \beta \leq 1$ is a user-defined parameter, very close to unity.

The Recursive Least-Squares Algorithm

- The exponentially weighted squared error cost function involves two user-defined parameters, namely:
  - The forgetting factor, $0 < \beta \leq 1$. The purpose of its presence is to assist the cost function to slowly forget past data samples, by weighting heavier the more recent observations. This will equip the algorithm with the agility to track changes.
  - The regularization-related parameter λ. Starting from time $n = 0$, we are forced to introduce regularization. During the initial period, i.e., $n < l - 1$, the corresponding system of equations will be underdetermined and $X_n^T X_n$ is **not** invertible. Indeed, we have that,

$$X_n^T X_n = \sum_{i=0}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T.$$

In other words, $X_n^T X_n$ is the sum of **rank one** matrices. Hence, for $n < l - 1$ its rank is necessarily less than l, and it cannot be inverted. For larger values of n, it can become full rank, provided that at least l of the input vectors are linearly independent, which is usually assumed to be the case. For large values of n, regularization is not necessary; this is the reason of the presence of β^{n+1}, which tends to zero.

The Recursive Least-Squares Algorithm

- The exponentially weighted squared error cost function involves two user-defined parameters, namely:
  - The forgetting factor, $0 < \beta \leq 1$. The purpose of its presence is to assist the cost function to slowly forget past data samples, by weighting heavier the more recent observations. This will equip the algorithm with the agility to track changes.
  - The regularization-related parameter λ. Starting from time $n = 0$, we are forced to introduce regularization. During the initial period, i.e., $n < l - 1$, the corresponding system of equations will be underdetermined and $X_n^T X_n$ is **not** invertible. Indeed, we have that,

$$X_n^T X_n = \sum_{i=0}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T.$$

In other words, $X_n^T X_n$ is the sum of **rank one** matrices. Hence, for $n < l - 1$ its rank is necessarily less than l, and it cannot be inverted. For larger values of n, it can become full rank, provided that at least l of the input vectors are linearly independent, which is usually assumed to be the case. For large values of n, regularization is not necessary; this is the reason of the presence of β^{n+1}, which tends to zero.

The Recursive Least-Squares Algorithm

- The exponentially weighted squared error cost function involves two user-defined parameters, namely:
  - The forgetting factor, $0 < \beta \leq 1$. The purpose of its presence is to assist the cost function to slowly forget past data samples, by weighting heavier the more recent observations. This will equip the algorithm with the agility to track changes.
  - The regularization-related parameter λ. Starting from time $n = 0$, we are forced to introduce regularization. During the initial period, i.e., $n < l - 1$, the corresponding system of equations will be underdetermined and $X_n^T X_n$ is **not** invertible. Indeed, we have that,

$$X_n^T X_n = \sum_{i=0}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T.$$

In other words, $X_n^T X_n$ is the sum of **rank one** matrices. Hence, for $n < l - 1$ its rank is necessarily less than l, and it cannot be inverted. For larger values of n, it can become full rank, provided that at least l of the input vectors are linearly independent, which is usually assumed to be the case. For large values of n, regularization is not necessary; this is the reason of the presence of β^{n+1}, which tends to zero.

The Recursive Least-Squares Algorithm

- Minimizing the EWLS cost results in,

$$\Phi_n \boldsymbol{\theta}_n = \boldsymbol{p}_n,$$

where,

$$\Phi_n = \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i \boldsymbol{x}_i^T + \lambda \beta^{n+1} I,$$

and

$$\boldsymbol{p}_n = \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i y_i,$$

which for $\beta = 1$ coincides with the ridge regression.

- Time-Iterative computations of Φ_n, $\boldsymbol{p}_n$: It turns out that

$$P_n := \Phi_n^{-1}, \tag{4}$$

$$P_n = \beta^{-1} P_{n-1} - \beta^{-1} K_n \boldsymbol{x}_n^T P_{n-1}, \tag{5}$$

$$K_n := \frac{\beta^{-1} P_{n-1} \boldsymbol{x}_n}{1 + \beta^{-1} \boldsymbol{x}_n^T P_{n-1} \boldsymbol{x}_n}. \tag{6}$$

K_n is known as the Kalman gain.

The Recursive Least-Squares Algorithm

- Minimizing the EWLS cost results in,

$$\Phi_n \boldsymbol{\theta}_n = \boldsymbol{p}_n,$$

where,

$$\Phi_n = \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i \boldsymbol{x}_i^T + \lambda \beta^{n+1} I,$$

and

$$\boldsymbol{p}_n = \sum_{i=0}^{n} \beta^{n-i} \boldsymbol{x}_i y_i,$$

which for $\beta = 1$ coincides with the ridge regression.

- Time-Iterative computations of Φ_n, $\boldsymbol{p}_n$: It turns out that

$$P_n := \Phi_n^{-1}, \tag{4}$$

$$P_n = \beta^{-1} P_{n-1} - \beta^{-1} K_n \boldsymbol{x}_n^T P_{n-1}, \tag{5}$$

$$K_n := \frac{\beta^{-1} P_{n-1} \boldsymbol{x}_n}{1 + \beta^{-1} \boldsymbol{x}_n^T P_{n-1} \boldsymbol{x}_n}. \tag{6}$$

K_n is known as the Kalman gain.

The Recursive Least-Squares Algorithm

- **Proof**: By the respective definitions, we have that

$$\Phi_n = \beta\Phi_{n-1} + \boldsymbol{x}_n\boldsymbol{x}_n^T,$$

and

$$\boldsymbol{p}_n = \beta\boldsymbol{p}_{n-1} + \boldsymbol{x}_n y_n.$$

- Recall Woodburry's matrix inversion formula,

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}.$$

Plugging it in the first of the above, we obtain (5) and (6).

- Also, rearranging the terms in (6), we get

$$K_n = \left(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^T P_{n-1}\right)\boldsymbol{x}_n,$$

and taking into account (5) results in

$$K_n = P_n\boldsymbol{x}_n.$$

- Time Updating of $\boldsymbol{\theta}_n$: Following similar arguments as before, we obtain

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n,$$

where

$$e_n := y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n.$$

The Recursive Least-Squares Algorithm

- **Proof**: By the respective definitions, we have that

$$\Phi_n = \beta\Phi_{n-1} + \boldsymbol{x}_n\boldsymbol{x}_n^T,$$

and

$$\boldsymbol{p}_n = \beta\boldsymbol{p}_{n-1} + \boldsymbol{x}_n y_n.$$

- Recall Woodburry's matrix inversion formula,

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}.$$

Plugging it in the first of the above, we obtain (5) and (6).

- Also, rearranging the terms in (6), we get

$$K_n = \left(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^T P_{n-1}\right)\boldsymbol{x}_n,$$

and taking into account (5) results in

$$K_n = P_n\boldsymbol{x}_n.$$

- Time Updating of $\boldsymbol{\theta}_n$: Following similar arguments as before, we obtain

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n,$$

where

$$e_n := y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n.$$

The Recursive Least-Squares Algorithm

- **Proof**: By the respective definitions, we have that

$$\Phi_n = \beta\Phi_{n-1} + \boldsymbol{x}_n\boldsymbol{x}_n^T,$$

and

$$\boldsymbol{p}_n = \beta\boldsymbol{p}_{n-1} + \boldsymbol{x}_n y_n.$$

- Recall Woodburry's matrix inversion formula,

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}.$$

Plugging it in the first of the above, we obtain (5) and (6).

- Also, rearranging the terms in (6), we get

$$K_n = \left(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^T P_{n-1}\right)\boldsymbol{x}_n,$$

and taking into account (5) results in

$$K_n = P_n\boldsymbol{x}_n.$$

- Time Updating of $\boldsymbol{\theta}_n$: Following similar arguments as before, we obtain

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n,$$

where

$$e_n := y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n.$$

The Recursive Least-Squares Algorithm

- **Proof**: By the respective definitions, we have that

$$\Phi_n = \beta\Phi_{n-1} + \boldsymbol{x}_n\boldsymbol{x}_n^T,$$

and

$$\boldsymbol{p}_n = \beta\boldsymbol{p}_{n-1} + \boldsymbol{x}_n y_n.$$

- Recall Woodburry's matrix inversion formula,

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}.$$

Plugging it in the first of the above, we obtain (5) and (6).

- Also, rearranging the terms in (6), we get

$$K_n = \left(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^T P_{n-1}\right)\boldsymbol{x}_n,$$

and taking into account (5) results in

$$K_n = P_n\boldsymbol{x}_n.$$

- Time Updating of $\boldsymbol{\theta}_n$: Following similar arguments as before, we obtain

$$\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n,$$

where

$$e_n := y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n.$$

The Recursive Least-Squares Algorithm

- **Proof**: By the respective definitions, we have that

$$\begin{aligned}\boldsymbol{\theta}_n &= \Phi_n^{-1}\boldsymbol{p}_n = \left(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^T P_{n-1}\right)\beta\boldsymbol{p}_{n-1} + \\ & P_n\boldsymbol{x}_n y_n \\ &= \boldsymbol{\theta}_{n-1} - K_n\boldsymbol{x}_n^T\boldsymbol{\theta}_{n-1} + K_n y_n \\ &= \boldsymbol{\theta}_{n-1} - K_n\left(y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n\right),\end{aligned}$$

  which proves the claim.

- All the necessary recursions have been derived. Note that the essence behind the derivations lies in a) expressing the quantities at time n in terms of their counterparts at time $n-1$ and b) the use of the matrix inversion lemma.

- The last time update recursion for the LS solution, has the typical form of

  New=Old+Error-Related Correction Term.

The Recursive Least-Squares Algorithm

- **Proof**: By the respective definitions, we have that

$$\begin{aligned}\boldsymbol{\theta}_n &= \Phi_n^{-1}\boldsymbol{p}_n = \left(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^T P_{n-1}\right)\beta\boldsymbol{p}_{n-1} + \\ & P_n\boldsymbol{x}_n y_n \\ &= \boldsymbol{\theta}_{n-1} - K_n\boldsymbol{x}_n^T\boldsymbol{\theta}_{n-1} + K_n y_n \\ &= \boldsymbol{\theta}_{n-1} - K_n\left(y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n\right),\end{aligned}$$

  which proves the claim.

- All the necessary recursions have been derived. Note that the essence behind the derivations lies in a) expressing the quantities at time n in terms of their counterparts at time $n-1$ and b) the use of the matrix inversion lemma.

- The last time update recursion for the LS solution, has the typical form of

$$\text{New}=\text{Old}+\text{Error-Related Correction Term}.$$

The Recursive Least-Squares Algorithm

- **Proof**: By the respective definitions, we have that

$$\begin{aligned}\boldsymbol{\theta}_n &= \Phi_n^{-1}\boldsymbol{p}_n = \Big(\beta^{-1}P_{n-1} - \beta^{-1}K_n\boldsymbol{x}_n^T P_{n-1}\Big)\beta\boldsymbol{p}_{n-1} + \\ & P_n\boldsymbol{x}_n y_n \\ &= \boldsymbol{\theta}_{n-1} - K_n\boldsymbol{x}_n^T\boldsymbol{\theta}_{n-1} + K_n y_n \\ &= \boldsymbol{\theta}_{n-1} - K_n\big(y_n - \boldsymbol{\theta}_{n-1}^T\boldsymbol{x}_n\big),\end{aligned}$$

  which proves the claim.

- All the necessary recursions have been derived. Note that the essence behind the derivations lies in a) expressing the quantities at time n in terms of their counterparts at time $n-1$ and b) the use of the matrix inversion lemma.

- The last time update recursion for the LS solution, has the typical form of

  New=Old+Error-Related Correction Term.

- **The RLS Algorithm**
  - Initialize
    - $\boldsymbol{\theta}_{-1} = \mathbf{0}$; any other value is also possible.
    - $P_{-1} = \lambda^{-1} I$; λ a user-defined variable.
    - Select β; close to 1.
  - **For** $n = 0, 1, \ldots$ **Do**
    - $e_n = y_n - \boldsymbol{\theta}_{n-1}^T \boldsymbol{x}_n$
    - $\boldsymbol{z}_n = P_{n-1} \boldsymbol{x}_n$
    - $K_n = \frac{\boldsymbol{z}_n}{\beta + \boldsymbol{x}_n^T \boldsymbol{z}_n}$
    - $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n$
    - $P_n = \beta^{-1} P_{n-1} - \beta^{-1} K_n \boldsymbol{z}_n^T$
  - **End For**
- The complexity of the RLS algorithm is of the order $\mathcal{O}(l^2)$ per iteration, due to the matrix-product operations. That is, there is an order of magnitude difference compared to the LMS and the other gradient descent-based schemes. In other words, the RLS does not scale well with dimensionality. Fast versions of the RLS algorithm, of complexity $\mathcal{O}(l)$, have also been derived for the case where the input is a random processes and are briefly discussed in the text.

The RLS Algorithm

- **The RLS Algorithm**
  - Initialize
    - $\boldsymbol{\theta}_{-1} = \mathbf{0}$; any other value is also possible.
    - $P_{-1} = \lambda^{-1} I$; λ a user-defined variable.
    - Select β; close to 1.
  - **For** $n = 0, 1, \ldots$ **Do**
    - $e_n = y_n - \boldsymbol{\theta}_{n-1}^T \boldsymbol{x}_n$
    - $\boldsymbol{z}_n = P_{n-1} \boldsymbol{x}_n$
    - $K_n = \frac{\boldsymbol{z}_n}{\beta + \boldsymbol{x}_n^T \boldsymbol{z}_n}$
    - $\boldsymbol{\theta}_n = \boldsymbol{\theta}_{n-1} + K_n e_n$
    - $P_n = \beta^{-1} P_{n-1} - \beta^{-1} K_n \boldsymbol{z}_n^T$
  - **End For**
- The complexity of the RLS algorithm is of the order $\mathcal{O}(l^2)$ per iteration, due to the matrix-product operations. That is, there is an order of magnitude difference compared to the LMS and the other gradient descent-based schemes. In other words, the RLS does not scale well with dimensionality. Fast versions of the RLS algorithm, of complexity $\mathcal{O}(l)$, have also been derived for the case where the input is a random processes and are briefly discussed in the text.

The RLS Algorithm

- The RLS algorithm shares similar numerical behaviour with the Kalman filter, which is discussed in Chapter 4. P_n may loose its positive definite and symmetric nature, which then leads the algorithm to divergence. To remedy such a tendency, a number of versions have been developed and discussed in the text.
- The choice of λ in the initialization step needs special consideration. The related theoretical analysis suggests that λ has a direct influence on the convergence speed and it should be chosen so that to be a small positive for high Signal-to-Noise (SNR) ratios and a large positive constant for low SNRs.
- The main advantage of the RLS is that it converges to the steady state **much faster** than the LMS and the rest of the members of the gradient-descent family. This can be justified by the fact that the RLS can been seen as an offspring of Newton's iterative optimization method.

The RLS Algorithm

- The RLS algorithm shares similar numerical behaviour with the Kalman filter, which is discussed in Chapter 4. P_n may loose its positive definite and symmetric nature, which then leads the algorithm to divergence. To remedy such a tendency, a number of versions have been developed and discussed in the text.
- The choice of λ in the initialization step needs special consideration. The related theoretical analysis suggests that λ has a direct influence on the convergence speed and it should be chosen so that to be a small positive for high Signal-to-Noise (SNR) ratios and a large positive constant for low SNRs.
- The main advantage of the RLS is that it converges to the steady state **much faster** than the LMS and the rest of the members of the gradient-descent family. This can be justified by the fact that the RLS can been seen as an offspring of Newton's iterative optimization method.

The RLS Algorithm

- The RLS algorithm shares similar numerical behaviour with the Kalman filter, which is discussed in Chapter 4. P_n may loose its positive definite and symmetric nature, which then leads the algorithm to divergence. To remedy such a tendency, a number of versions have been developed and discussed in the text.
- The choice of λ in the initialization step needs special consideration. The related theoretical analysis suggests that λ has a direct influence on the convergence speed and it should be chosen so that to be a small positive for high Signal-to-Noise (SNR) ratios and a large positive constant for low SNRs.
- The main advantage of the RLS is that it converges to the steady state **much faster** than the LMS and the rest of the members of the gradient-descent family. This can be justified by the fact that the RLS can been seen as an offspring of Newton's iterative optimization method.

Newton's Iterative Minimization Method

- The steepest descent optimization scheme is discussed in Chapter 5. There, it is stated that the members of this family exhibit linear convergence rate and a heavy dependence on the condition number of the Hessian matrix associated with the cost function. The heart of the gradient descent schemes beats around a first order Taylor's expansion of the cost function.
- Newton's method is a way to overcome this dependence on the condition number and at the same time improve upon the rate of convergence towards the solution. Let us proceed with a second order Taylor's expansion of the cost, i.e.,

$$J\left(\boldsymbol{\theta}^{(i-1)}+\Delta\boldsymbol{\theta}^{(i)}\right) = J\left(\boldsymbol{\theta}^{(i-1)}\right)+\left(\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{T}\Delta\boldsymbol{\theta}^{(i)}+\frac{1}{2}\left(\Delta\boldsymbol{\theta}^{(i)}\right)^{T}\nabla^{2}J\left(\boldsymbol{\theta}^{(i-1)}\right)\Delta\boldsymbol{\theta}^{(i)}.$$

- Assuming $\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)$ to be positive definite (this is always the case if $J(\boldsymbol{\theta})$ is a strictly convex function), the above is a convex quadratic function w.r. to the step $\Delta\boldsymbol{\theta}^{(i)}$; the latter is computed so that to minimize the above second order approximation.

Newton's Iterative Minimization Method

- The steepest descent optimization scheme is discussed in Chapter 5. There, it is stated that the members of this family exhibit linear convergence rate and a heavy dependence on the condition number of the Hessian matrix associated with the cost function. The heart of the gradient descent schemes beats around a first order Taylor's expansion of the cost function.
- Newton's method is a way to overcome this dependence on the condition number and at the same time improve upon the rate of convergence towards the solution. Let us proceed with a second order Taylor's expansion of the cost, i.e.,

$$J\left(\boldsymbol{\theta}^{(i-1)}+\Delta\boldsymbol{\theta}^{(i)}\right) = J\left(\boldsymbol{\theta}^{(i-1)}\right)+\left(\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{T}\Delta\boldsymbol{\theta}^{(i)}+\frac{1}{2}\left(\Delta\boldsymbol{\theta}^{(i)}\right)^{T}\nabla^{2}J\left(\boldsymbol{\theta}^{(i-1)}\right)\Delta\boldsymbol{\theta}^{(i)}.$$

- Assuming $\nabla^2 J\left(\theta^{(i-1)}\right)$ to be positive definite (this is always the case if $J(\theta)$ is a strictly convex function), the above is a convex quadratic function w.r. to the step $\Delta\theta^{(i)}$; the latter is computed so that to minimize the above second order approximation.

Newton's Iterative Minimization Method

- The steepest descent optimization scheme is discussed in Chapter 5. There, it is stated that the members of this family exhibit linear convergence rate and a heavy dependence on the condition number of the Hessian matrix associated with the cost function. The heart of the gradient descent schemes beats around a first order Taylor's expansion of the cost function.
- Newton's method is a way to overcome this dependence on the condition number and at the same time improve upon the rate of convergence towards the solution. Let us proceed with a second order Taylor's expansion of the cost, i.e.,

$$\begin{aligned} J\left(\boldsymbol{\theta}^{(i-1)}+\Delta\boldsymbol{\theta}^{(i)}\right) &= J\left(\boldsymbol{\theta}^{(i-1)}\right)+\left(\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{T}\Delta\boldsymbol{\theta}^{(i)}+ \\ &\quad \frac{1}{2}\left(\Delta\boldsymbol{\theta}^{(i)}\right)^{T}\nabla^{2}J\left(\boldsymbol{\theta}^{(i-1)}\right)\Delta\boldsymbol{\theta}^{(i)}. \end{aligned}$$

- Assuming $\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)$ to be positive definite (this is always the case if $J(\boldsymbol{\theta})$ is a strictly convex function), the above is a convex quadratic function w.r. to the step $\Delta\boldsymbol{\theta}^{(i)}$; the latter is computed so that to minimize the above second order approximation.

Newton's Iterative Minimization Method

- The minimum results by equating the corresponding gradient to $\mathbf{0}$, i.e.,

$$\Delta\boldsymbol{\theta}^{(i)} = -\left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1}\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right).$$

Note that this is indeed a descent direction, because

$$\nabla^T J\left(\boldsymbol{\theta}^{(i-1)}\right)\Delta\boldsymbol{\theta}^{(i)} = -\nabla^T J\left(\boldsymbol{\theta}^{(i-1)}\right)\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right) < 0,$$

due to the positive definite nature of the Hessian. Equality to zero is achieved only at a minimum.

- Thus, the iterative scheme takes the following form:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \mu_i\left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1}\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)$$

Newton's Iterative Minimization Method

- The minimum results by equating the corresponding gradient to $\mathbf{0}$, i.e.,

$$\Delta\boldsymbol{\theta}^{(i)} = -\left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1}\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right).$$

  Note that this is indeed a descent direction, because

$$\nabla^T J\left(\boldsymbol{\theta}^{(i-1)}\right)\Delta\boldsymbol{\theta}^{(i)} = -\nabla^T J\left(\boldsymbol{\theta}^{(i-1)}\right)\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right) < 0,$$

  due to the positive definite nature of the Hessian. Equality to zero is achieved only at a minimum.

- Thus, the iterative scheme takes the following form:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \mu_i\left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1}\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)$$

Newton's Iterative Minimization Method

- The minimum results by equating the corresponding gradient to $\mathbf{0}$, i.e.,

$$\Delta\boldsymbol{\theta}^{(i)} = -\left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1}\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right).$$

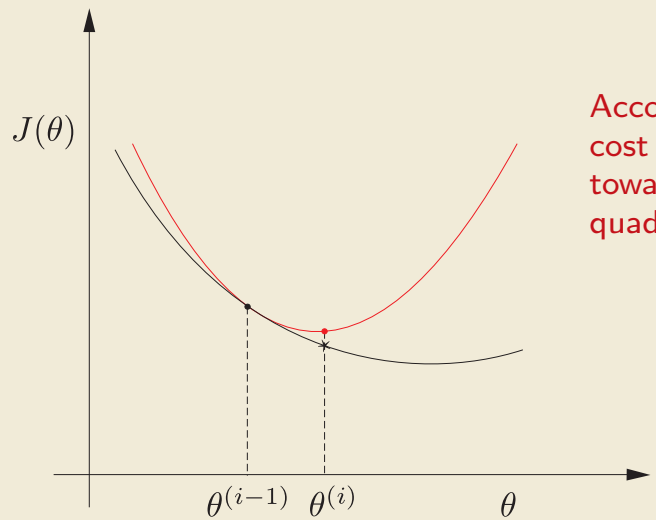
  Note that this is indeed a descent direction, because

$$\nabla^T J\left(\boldsymbol{\theta}^{(i-1)}\right)\Delta\boldsymbol{\theta}^{(i)} = -\nabla^T J\left(\boldsymbol{\theta}^{(i-1)}\right)\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right) < 0,$$

  due to the positive definite nature of the Hessian. Equality to zero is achieved only at a minimum.

- Thus, the iterative scheme takes the following form:

$$\boldsymbol{\theta}^{(i)} = \boldsymbol{\theta}^{(i-1)} - \mu_i\left(\nabla^2 J\left(\boldsymbol{\theta}^{(i-1)}\right)\right)^{-1}\nabla J\left(\boldsymbol{\theta}^{(i-1)}\right)$$



According to Newton's method, a local quadratic approximation of the cost function is considered, and the correction pushes the new estimate towards the minimum of this approximation. If the cost function is quadratic, then convergence can be achieved in one step.

Newton's Iterative Minimization Method

- The presence of the Hessian in the correction term remedies, to a large extent, the influence of the condition number of the Hessian matrix on the convergence.
- The convergence rate for Newton's method is, in general, high and it becomes quadratic close to the solution. Assuming $\boldsymbol{\theta}_*$ to be the minimum, quadratic convergence means that at each iteration, i, the deviation from the optimum value follows the pattern:

$$\ln \ln \frac{1}{||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*||^2} \propto i$$

- In contrast, for the linear convergence, the iterations approach the optimal according to:

$$\ln \frac{1}{||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*||^2} \propto i.$$

- The RLS algorithm can be rederived following Newton's iterative scheme applied to the MSE and adopting stochastic approximation arguments.

Newton's Iterative Minimization Method

- The presence of the Hessian in the correction term remedies, to a large extent, the influence of the condition number of the Hessian matrix on the convergence.
- The convergence rate for Newton's method is, in general, high and it becomes quadratic close to the solution. Assuming $\boldsymbol{\theta}_*$ to be the minimum, quadratic convergence means that at each iteration, i, the deviation from the optimum value follows the pattern:

$$\ln \ln \frac{1}{||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*||^2} \propto i$$

- In contrast, for the linear convergence, the iterations approach the optimal according to:

$$\ln \frac{1}{||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*||^2} \propto i.$$

- The RLS algorithm can be rederived following Newton's iterative scheme applied to the MSE and adopting stochastic approximation arguments.

Newton's Iterative Minimization Method

- The presence of the Hessian in the correction term remedies, to a large extent, the influence of the condition number of the Hessian matrix on the convergence.
- The convergence rate for Newton's method is, in general, high and it becomes quadratic close to the solution. Assuming $\boldsymbol{\theta}_*$ to be the minimum, quadratic convergence means that at each iteration, i, the deviation from the optimum value follows the pattern:

$$\ln \ln \frac{1}{||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*||^2} \propto i$$

- In contrast, for the linear convergence, the iterations approach the optimal according to:

$$\ln \frac{1}{||\boldsymbol{\theta}^{(i)} - \boldsymbol{\theta}_*||^2} \propto i.$$

- The RLS algorithm can be rederived following Newton's iterative scheme applied to the MSE and adopting stochastic approximation arguments.

Steady State Performance of the RLS

- Compared to the stochastic gradient techniques, we do not have to worry whether RLS converges and where it converges. The RLS computes the exact solution of the EWLS minimization task in an iterative way. Asymptotically and for $\beta = 1$, $\lambda = 0$) solves the MSE optimization task.
- However, we have to consider its steady state performance for $\beta \neq 1$. Even for the stationary case, $\beta \neq 1$ results in an excess MSE. To this end, we adopt the same setting as that which is followed in Chapter 5.
- We adopt the following model for generating the data,

$$y_n = \theta_{o,n}^T \mathbf{x}_n + \eta_n,$$

and

$$\theta_{o,n} = \theta_{o,n-1} + \boldsymbol{\omega}_n, \text{ with } \mathbb{E}[\boldsymbol{\omega}_n \boldsymbol{\omega}_n^T] = \Sigma_\omega.$$

where η_n are the noise samples, i.i.d drawn, of variance σ_η^2.

- A performance index is related to the excess MSE with respect to the optimal MSE estimator, for each time instant, n. Since the MSE estimator is the optimal one, it results in the minimum MSE error, J_{min}. The estimator associated with the RLS, minimizing the EWLS, will result in higher MSE, by an amount J_{exc}.

Steady State Performance of the RLS

- Compared to the stochastic gradient techniques, we do not have to worry whether RLS converges and where it converges. The RLS computes the exact solution of the EWLS minimization task in an iterative way. Asymptotically and for $\beta = 1$, $\lambda = 0$) solves the MSE optimization task.
- However, we have to consider its steady state performance for $\beta \neq 1$. Even for the stationary case, $\beta \neq 1$ results in an excess MSE. To this end, we adopt the same setting as that which is followed in Chapter 5.
- We adopt the following model for generating the data,

$$y_n = \theta_{o,n}^T \mathbf{x}_n + \eta_n,$$

and

$$\theta_{\mathbf{o,n}} = \theta_{\mathbf{o,n-1}} + \boldsymbol{\omega}_\mathbf{n}, \text{ with } \mathbb{E}[\boldsymbol{\omega}_\mathbf{n}\boldsymbol{\omega}_\mathbf{n}^\mathbf{T}] = \Sigma_\omega.$$

where η_n are the noise samples, i.i.d drawn, of variance σ_η^2.

- A performance index is related to the excess MSE with respect to the optimal MSE estimator, for each time instant, n. Since the MSE estimator is the optimal one, it results in the minimum MSE error, J_{min}. The estimator associated with the RLS, minimizing the EWLS, will result in higher MSE, by an amount J_{exc}.

Steady State Performance of the RLS

- Compared to the stochastic gradient techniques, we do not have to worry whether RLS converges and where it converges. The RLS computes the exact solution of the EWLS minimization task in an iterative way. Asymptotically and for $\beta = 1$, $\lambda = 0$) solves the MSE optimization task.
- However, we have to consider its steady state performance for $\beta \neq 1$. Even for the stationary case, $\beta \neq 1$ results in an excess MSE. To this end, we adopt the same setting as that which is followed in Chapter 5.
- We adopt the following model for generating the data,

$$y_n = \boldsymbol{\theta}_{o,n}^T \mathbf{x}_n + \eta_n,$$

and

$$\boldsymbol{\theta}_{\mathbf{o,n}} = \boldsymbol{\theta}_{\mathbf{o,n-1}} + \boldsymbol{\omega}_{\mathbf{n}}, \text{ with } \mathbb{E}[\boldsymbol{\omega}_{\mathbf{n}} \boldsymbol{\omega}_{\mathbf{n}}^{\mathbf{T}}] = \Sigma_{\omega}.$$

where η_n are the noise samples, i.i.d drawn, of variance σ_η^2.

- A performance index is related to the excess MSE with respect to the optimal MSE estimator, for each time instant, n. Since the MSE estimator is the optimal one, it results in the minimum MSE error, J_{min}. The estimator associated with the RLS, minimizing the EWLS, will result in higher MSE, by an amount J_{exc}.

Steady State Performance of the RLS

- Compared to the stochastic gradient techniques, we do not have to worry whether RLS converges and where it converges. The RLS computes the exact solution of the EWLS minimization task in an iterative way. Asymptotically and for $\beta = 1$, $\lambda = 0$) solves the MSE optimization task.
- However, we have to consider its steady state performance for $\beta \neq 1$. Even for the stationary case, $\beta \neq 1$ results in an excess MSE. To this end, we adopt the same setting as that which is followed in Chapter 5.
- We adopt the following model for generating the data,

$$y_n = \boldsymbol{\theta}_{o,n}^T \mathbf{x}_n + \eta_n,$$

and

$$\boldsymbol{\theta}_{\mathbf{o,n}} = \boldsymbol{\theta}_{\mathbf{o,n-1}} + \boldsymbol{\omega}_{\mathbf{n}}, \text{ with } \mathbb{E}[\boldsymbol{\omega}_{\mathbf{n}} \boldsymbol{\omega}_{\mathbf{n}}^{\mathbf{T}}] = \Sigma_{\omega}.$$

where η_n are the noise samples, i.i.d drawn, of variance σ_η^2.

- A performance index is related to the excess MSE with respect to the optimal MSE estimator, for each time instant, n. Since the MSE estimator is the optimal one, it results in the minimum MSE error, J_{min}. The estimator associated with the RLS, minimizing the EWLS, will result in higher MSE, by an amount J_{exc}.

Steady State Performance of the RLS

- The resulting excess MSE, J_{exc}, for the RLS is given below together with that of the LMS, for the sake of comparison.

Algorithm	Excess MSE, J_{exc}, at Steady-state.
LMS	$\frac{1}{2}\mu\sigma_\eta^2 \text{Trace}\{\Sigma_x\} + \frac{1}{2}\mu^{-1}\text{Trace}\{\Sigma_\omega\}$
RLS	$\frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1}\text{Trace}\{\Sigma_\omega \Sigma_x\}$

Table: The Steady State Excess MSE, for small values of μ and β. For stationary environments, Σ_ω is set equal to zero.

- According to the table, the following remarks are in order:
  - For stationary environments, the performance of the RLS is independent of input data covariance matrix, Σ_x. Of course, if one knows that the environment is stationary then ideally $\beta = 1$ should be the choice. Yet, for $\beta = 1$, the algorithm has stability problems.
  - Note that for small μ and $\beta \simeq 1$, there is an "equivalence" of $\mu \simeq 1 - \beta$, for the two parameters in the LMS and RLS. That is, larger values of μ are beneficial to the tracking performance of LMS, while smaller values of β need for faster tracking of the RLS; this is expected since the algorithm forgets the past.

Steady State Performance of the RLS

- The resulting excess MSE, J_{exc}, for the RLS is given below together with that of the LMS, for the sake of comparison.

Algorithm	Excess MSE, J_{exc}, at Steady-state.
LMS	$\frac{1}{2}\mu\sigma_\eta^2 \text{Trace}\{\Sigma_x\} + \frac{1}{2}\mu^{-1}\text{Trace}\{\Sigma_\omega\}$
RLS	$\frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1}\text{Trace}\{\Sigma_\omega \Sigma_x\}$

Table: The Steady State Excess MSE, for small values of μ and β. For stationary environments, Σ_ω is set equal to zero.

- According to the table, the following remarks are in order:
  - For stationary environments, the performance of the RLS is independent of input data covariance matrix, Σ_x. Of course, if one knows that the environment is stationary then ideally $\beta = 1$ should be the choice. Yet, for $\beta = 1$, the algorithm has stability problems.
  - Note that for small μ and $\beta \simeq 1$, there is an "equivalence" of $\mu \simeq 1 - \beta$, for the two parameters in the LMS and RLS. That is, larger values of μ are beneficial to the tracking performance of LMS, while smaller values of β need for faster tracking of the RLS; this is expected since the algorithm forgets the past.

Steady State Performance of the RLS

- The resulting excess MSE, J_{exc}, for the RLS is given below together with that of the LMS, for the sake of comparison.

Algorithm	Excess MSE, J_{exc}, at Steady-state.
LMS	$\frac{1}{2}\mu\sigma_\eta^2 \text{Trace}\{\Sigma_x\} + \frac{1}{2}\mu^{-1}\text{Trace}\{\Sigma_\omega\}$
RLS	$\frac{1}{2}(1-\beta)\sigma_\eta^2 l + \frac{1}{2}(1-\beta)^{-1}\text{Trace}\{\Sigma_\omega \Sigma_x\}$

Table: The Steady State Excess MSE, for small values of μ and β. For stationary environments, Σ_ω is set equal to zero.

- According to the table, the following remarks are in order:
  - For stationary environments, the performance of the RLS is independent of input data covariance matrix, Σ_x. Of course, if one knows that the environment is stationary then ideally $\beta = 1$ should be the choice. Yet, for $\beta = 1$, the algorithm has stability problems.
  - Note that for small μ and $\beta \simeq 1$, there is an "equivalence" of $\mu \simeq 1 - \beta$, for the two parameters in the LMS and RLS. That is, larger values of μ are beneficial to the tracking performance of LMS, while smaller values of β need for faster tracking of the RLS; this is expected since the algorithm forgets the past.

Steady State Performance of the RLS

- The minimum values for the excess MSE, corresponding to the optimal set up of the parameters, μ and β, for the LMS and RLS, respectively, is easily shown to obey the following,

$$\frac{J_{\min}^{\text{LMS}}}{J_{\min}^{\text{RLS}}} = \sqrt{\frac{\text{Trace}\{\Sigma_x\}\text{Trace}\{\Sigma_\omega\}}{l\,\text{Trace}\{\Sigma_\omega\Sigma_x\}}}.$$

This ratio depends on Σ_ω and Σ_x. Sometimes LMS tracks better, yet in other problems RLS is the winner. Having said that, it must be pointed out that the RLS always converges to the steady-state faster and the difference in the rate, compared to the LMS, increases with the condition number of the input covariance matrix. Recall that, the steady-state of an online algorithm has been reached if the parameter error covariance matrix does not change with time.

- Dealing with the analysis of online algorithms is a mathematically tough task, and the previous reported results have to be considered as a first and a rough justification of what is experimentally observed in practice; they are results obtained under a set of strong, and sometimes unrealistic, assumptions. Some further discussion on these issues is provided in the text.

Steady State Performance of the RLS

- The minimum values for the excess MSE, corresponding to the optimal set up of the parameters, μ and β, for the LMS and RLS, respectively, is easily shown to obey the following,

$$\frac{J_{\min}^{\text{LMS}}}{J_{\min}^{\text{RLS}}} = \sqrt{\frac{\text{Trace}\{\Sigma_x\}\text{Trace}\{\Sigma_\omega\}}{l\,\text{Trace}\{\Sigma_\omega\Sigma_x\}}}.$$

  This ratio depends on Σ_ω and Σ_x. Sometimes LMS tracks better, yet in other problems RLS is the winner. Having said that, it must be pointed out that the RLS always converges to the steady-state faster and the difference in the rate, compared to the LMS, increases with the condition number of the input covariance matrix. Recall that, the steady-state of an online algorithm has been reached if the parameter error covariance matrix does not change with time.

- Dealing with the analysis of online algorithms is a mathematically tough task, and the previous reported results have to be considered as a first and a rough justification of what is experimentally observed in practice; they are results obtained under a set of strong, and sometimes unrealistic, assumptions. Some further discussion on these issues is provided in the text.

Steady State Performance of the RLS

- The minimum values for the excess MSE, corresponding to the optimal set up of the parameters, μ and β, for the LMS and RLS, respectively, is easily shown to obey the following,

$$\frac{J_{\min}^{\text{LMS}}}{J_{\min}^{\text{RLS}}} = \sqrt{\frac{\text{Trace}\{\Sigma_x\}\text{Trace}\{\Sigma_\omega\}}{l\text{Trace}\{\Sigma_\omega\Sigma_x\}}}.$$

  This ratio depends on Σ_ω and Σ_x. Sometimes LMS tracks better, yet in other problems RLS is the winner. Having said that, it must be pointed out that the RLS always converges to the steady-state faster and the difference in the rate, compared to the LMS, increases with the condition number of the input covariance matrix. Recall that, the steady-state of an online algorithm has been reached if the parameter error covariance matrix does not change with time.
- Dealing with the analysis of online algorithms is a mathematically tough task, and the previous reported results have to be considered as a first and a rough justification of what is experimentally observed in practice; they are results obtained under a set of strong, and sometimes unrealistic, assumptions. Some further discussion on these issues is provided in the text.

Comparative Performance of the RLS: Some Simulation Examples

- Stationary Environment: The focus of this example is to demonstrate the comparative performance, with respect to the convergence rate of the RLS, the NLMS and the APA algorithms, which are discussed in Chapter 5. To this end, data were generated according to the regression model

$$y_n = \boldsymbol{\theta}_o^T \boldsymbol{x}_n + \eta_n,$$

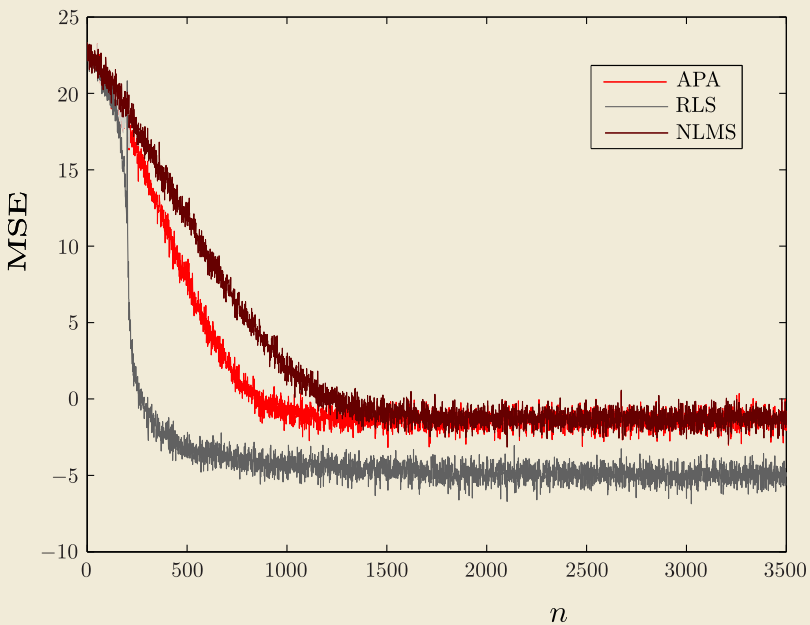
where $\boldsymbol{\theta}_o \in \mathbb{R}^{200}$. Its elements are generated randomly according the normalized Gaussian. The noise samples are i.i.d generated via the zero mean Gaussian with variance equal to $\sigma_\eta^2 = 0.01$. The elements of the input vector are also i.i.d. generated via the normalized Gaussian. Using the generated samples $(y_n, \boldsymbol{x}_n),\ n = 0, 1, \ldots$, as the training sequence, the convergence curves of the figure shown below are obtained.

Comparative Performance of the RLS: Some Simulation Examples

- Stationary Environment: The focus of this example is to demonstrate the comparative performance, with respect to the convergence rate of the RLS, the NLMS and the APA algorithms, which are discussed in Chapter 5. To this end, data were generated according to the regression model

$$y_n = \boldsymbol{\theta}_o^T \boldsymbol{x}_n + \eta_n,$$

where $\boldsymbol{\theta}_o \in \mathbb{R}^{200}$. Its elements are generated randomly according the normalized Gaussian. The noise samples are i.i.d generated via the zero mean Gaussian with variance equal to $\sigma_\eta^2 = 0.01$. The elements of the input vector are also i.i.d. generated via the normalized Gaussian. Using the generated samples $(y_n, \boldsymbol{x}_n)$, $n = 0, 1, \ldots$, as the training sequence, the convergence curves of the figure shown below are obtained.



The curves show the average mean square in dBs ($10\log_{10}(e_n^2)$), averaged over 100 different realizations of the experiments, as a function of the time index n. The parameters used for the involved algorithms are: a) For he NLMS, we used $\mu = 1.2$ and $\delta = 0.001$, b) for the APA, we used $\mu = 0.2$, $\delta = 0.001$ and $q = 30$ and c) for the RLS $\beta = 1$ and $\lambda = 0.1$. The parameters for the NLMS and the APA were chosen so that both algorithms to converge to the same error floor.
Observe that the the RLS converges faster and at lower error floor.

Comparative Performance of the RLS: Some Simulation Examples

- Rayleigh fading channels: This example focusses on the comparative tracking performance of the RLS and NLMS. Our goal is to demonstrate some cases, where the RLS fails to do as good as the NLMS. Of course, it has to be kept in mind that, according to the theory, the comparative performance is very much dependent on the specific application.
- For the needs of our example, data were generated according to the model,

$$y_n = \boldsymbol{x}_n^T \boldsymbol{\theta}_{o,n} + \eta_n, \text{ where } \boldsymbol{\theta}_{o,n} = \alpha \boldsymbol{\theta}_{o,n-1} + \boldsymbol{\omega}_n, \ \boldsymbol{\theta}_{o,n} \in \mathbb{R}^5$$

- It turns out that such a time varying model is closely related to what is known in communications as a Rayleigh fading channel, if the parameters comprising $\boldsymbol{\theta}_o$ are thought to represent the impulse response of such a channel. Rayleigh fading channels are very common and can adequately model a number of transmission channels in wireless communications. Playing with the parameters α and the variance of the corresponding noise source, $\boldsymbol{\omega}$, one can achieve fast or slow time varying scenarios. In our case, we chose $\alpha = 0.97$ and the noise followed a Gaussian distribution of zero mean and covariance matrix $\Sigma_\omega = 0.1I$. This choice corresponds to a fast fading channel. The comparative performance curves are shown in the next figure.

Comparative Performance of the RLS: Some Simulation Examples

- Rayleigh fading channels: This example focusses on the comparative tracking performance of the RLS and NLMS. Our goal is to demonstrate some cases, where the RLS fails to do as good as the NLMS. Of course, it has to be kept in mind that, according to the theory, the comparative performance is very much dependent on the specific application.
- For the needs of our example, data were generated according to the model,

$$y_n = \boldsymbol{x}_n^T \boldsymbol{\theta}_{o,n} + \eta_n, \text{ where } \boldsymbol{\theta}_{o,n} = \alpha \boldsymbol{\theta}_{o,n-1} + \boldsymbol{\omega}_n, \ \boldsymbol{\theta}_{o,n} \in \mathbb{R}^5$$

- It turns out that such a time varying model is closely related to what is known in communications as a Rayleigh fading channel, if the parameters comprising θ_o are thought to represent the impulse response of such a channel. Rayleigh fading channels are very common and can adequately model a number of transmission channels in wireless communications. Playing with the parameters α and the variance of the corresponding noise source, $\boldsymbol{\omega}$, one can achieve fast or slow time varying scenarios. In our case, we chose $\alpha = 0.97$ and the noise followed a Gaussian distribution of zero mean and covariance matrix $\Sigma_\omega = 0.1I$. This choice corresponds to a fast fading channel. The comparative performance curves are shown in the next figure.

Comparative Performance of the RLS: Some Simulation Examples

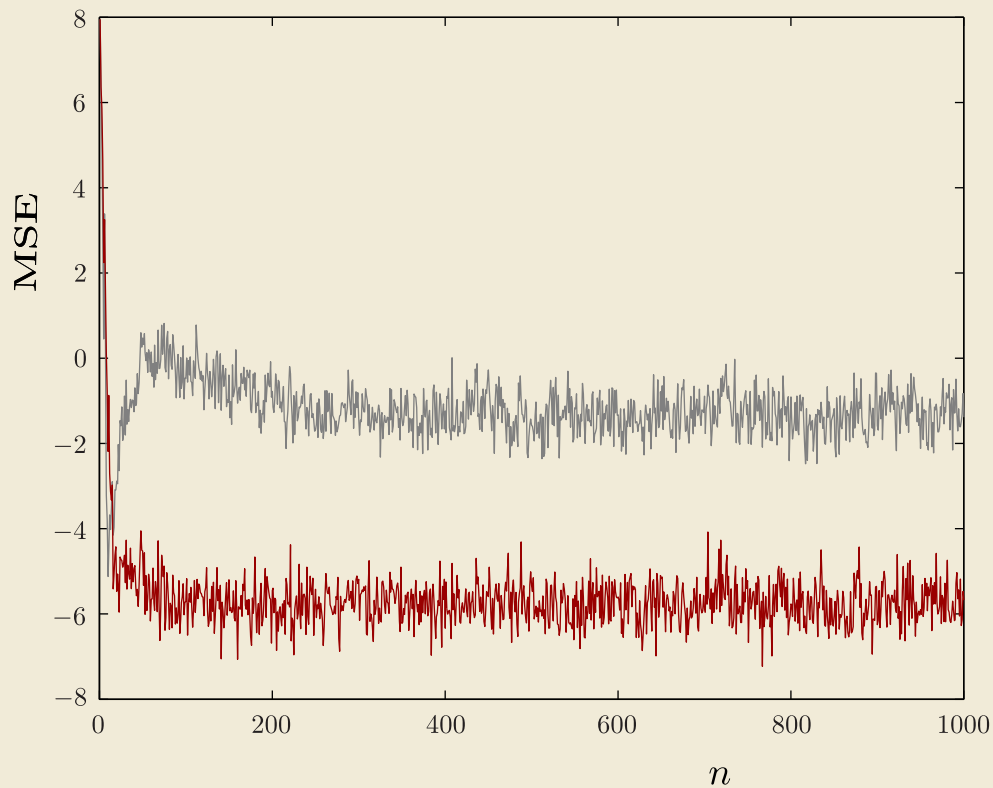
- Rayleigh fading channels: This example focusses on the comparative tracking performance of the RLS and NLMS. Our goal is to demonstrate some cases, where the RLS fails to do as good as the NLMS. Of course, it has to be kept in mind that, according to the theory, the comparative performance is very much dependent on the specific application.
- For the needs of our example, data were generated according to the model,

$$y_n = \boldsymbol{x}_n^T \boldsymbol{\theta}_{o,n} + \eta_n, \text{ where } \boldsymbol{\theta}_{o,n} = \alpha \boldsymbol{\theta}_{o,n-1} + \boldsymbol{\omega}_n, \ \boldsymbol{\theta}_{o,n} \in \mathbb{R}^5$$

- It turns out that such a time varying model is closely related to what is known in communications as a Rayleigh fading channel, if the parameters comprising $\boldsymbol{\theta}_o$ are thought to represent the impulse response of such a channel. Rayleigh fading channels are very common and can adequately model a number of transmission channels in wireless communications. Playing with the parameters α and the variance of the corresponding noise source, $\boldsymbol{\omega}$, one can achieve fast or slow time varying scenarios. In our case, we chose $\alpha = 0.97$ and the noise followed a Gaussian distribution of zero mean and covariance matrix $\Sigma_\omega = 0.1I$. This choice corresponds to a fast fading channel. The comparative performance curves are shown in the next figure.

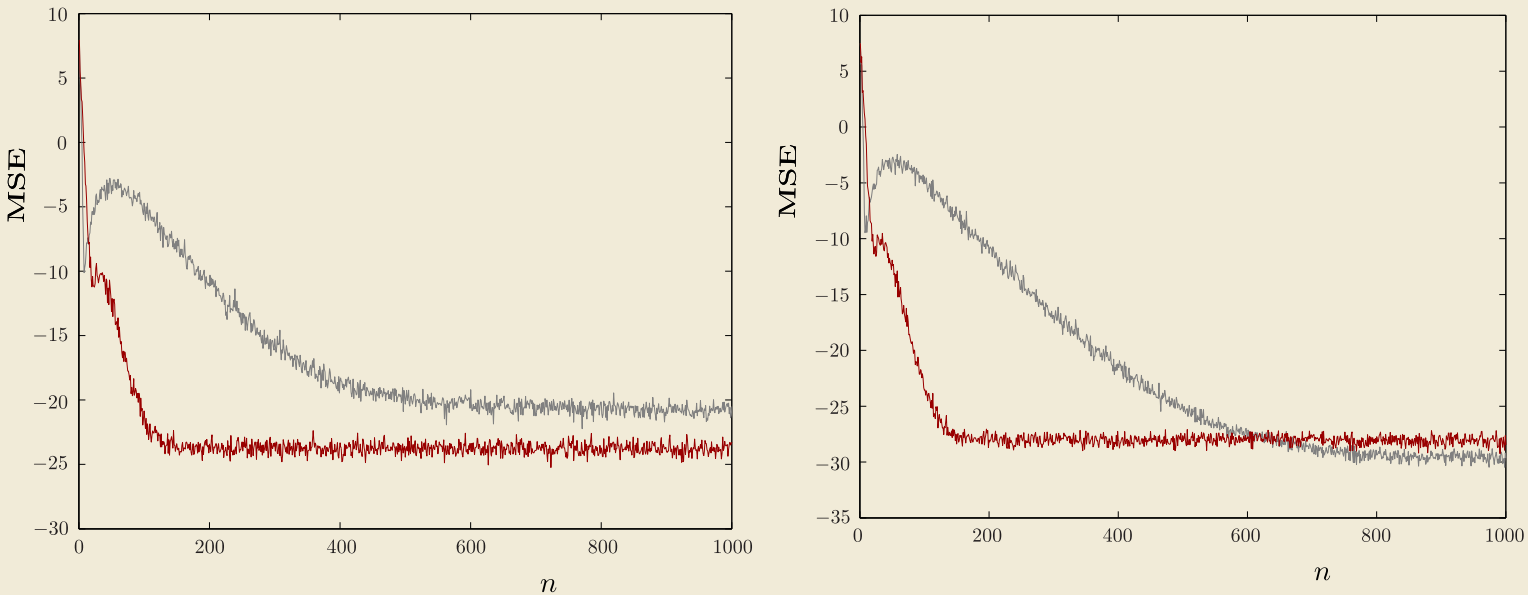
Comparative Performance of the RLS: Some Simulation Examples

- For the RLS (gray), the forgetting factor was set equal to $\beta = 0.995$ and for the NLMS (red), $\mu = 0.5$ and $\delta = 0.001$. Such a choice resulted in the best performance, for both algorithms, after extensive experimentation. The curves are the result of averaging out over 200 independent runs. For this fast fading channel case, the RLS fails to track it, in spite of its very fast initial convergence, compared to the NLMS.



Comparative Performance of the RLS: Some Simulation Examples

- The following figures show the resulting curves for a medium (a) and a slow (b) time varying channels, corresponding to $\Sigma_\omega = 0.01I$ and $\Sigma_\omega = 0.001I$ respectively.



(a) (b)

MSE curves as a function of iteration for a) a medium and b) a slow time varying parameter model. The red curve corresponds to the NLMS and the gray one to the RLS.

Singular Value Decomposition

- Singular Value Decomposition: The Singular Value Decomposition (SVD) of a matrix is one among the most powerful tools in linear algebra. We start by considering the general case.
- Let X be an $m \times l$ matrix and allow its rank, r, not to be necessarily full, i.e., $r \leq \min(m, l)$.
- Then, there exist orthogonal matrices, U and V, of dimensions $m \times m$ and $l \times l$, respectively, so that

$$X = U \begin{bmatrix} D & O \\ O & O \end{bmatrix} V^T$$

where D is an $r \times r$ diagonal matrix with elements $\sigma_i = \sqrt{\lambda_i}$, known as the singular values of X, where $\lambda_i,\ i = 1, 2, \ldots, r$, are the nonzero eigenvalues of XX^T; matrices denoted as O comprise zero elements and are of appropriate dimensions.

- Taking into account the zero elements in the diagonal matrix, the previous matrix factorization can be rewritten as

$$X = U_r D V_r^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T, \tag{7}$$

where

$$U_r := [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{m \times r}, \quad V_r := [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r] \in \mathbb{R}^{l \times r}. \tag{8}$$

Singular Value Decomposition

- Singular Value Decomposition: The Singular Value Decomposition (SVD) of a matrix is one among the most powerful tools in linear algebra. We start by considering the general case.
- Let X be an $m \times l$ matrix and allow its rank, r, not to be necessarily full, i.e., $r \leq \min(m, l)$.
- Then, there exist orthogonal matrices, U and V, of dimensions $m \times m$ and $l \times l$, respectively, so that

$$X = U \begin{bmatrix} D & O \\ O & O \end{bmatrix} V^T$$

where D is an $r \times r$ diagonal matrix with elements $\sigma_i = \sqrt{\lambda_i}$, known as the singular values of X, where $\lambda_i,\ i = 1, 2, \ldots, r$, are the nonzero eigenvalues of XX^T; matrices denoted as O comprise zero elements and are of appropriate dimensions.

- Taking into account the zero elements in the diagonal matrix, the previous matrix factorization can be rewritten as

$$X = U_r D V_r^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T, \tag{7}$$

where

$$U_r := [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{m \times r}, \quad V_r := [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r] \in \mathbb{R}^{l \times r}. \tag{8}$$

Singular Value Decomposition

- Singular Value Decomposition: The Singular Value Decomposition (SVD) of a matrix is one among the most powerful tools in linear algebra. We start by considering the general case.
- Let X be an $m \times l$ matrix and allow its rank, r, not to be necessarily full, i.e., $r \leq \min(m, l)$.
- Then, there exist orthogonal matrices, U and V, of dimensions $m \times m$ and $l \times l$, respectively, so that

$$X = U \begin{bmatrix} D & O \\ O & O \end{bmatrix} V^T$$

where D is an $r \times r$ diagonal matrix with elements $\sigma_i = \sqrt{\lambda_i}$, known as the singular values of X, where $\lambda_i,\ i = 1, 2, \ldots, r$, are the nonzero eigenvalues of XX^T; matrices denoted as O comprise zero elements and are of appropriate dimensions.

- Taking into account the zero elements in the diagonal matrix, the previous matrix factorization can be rewritten as

$$X = U_r D V_r^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T, \tag{7}$$

where

$$U_r := [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{m \times r}, \quad V_r := [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r] \in \mathbb{R}^{l \times r}. \tag{8}$$

Singular Value Decomposition

- Singular Value Decomposition: The Singular Value Decomposition (SVD) of a matrix is one among the most powerful tools in linear algebra. We start by considering the general case.
- Let X be an $m \times l$ matrix and allow its rank, r, not to be necessarily full, i.e., $r \leq \min(m, l)$.
- Then, there exist orthogonal matrices, U and V, of dimensions $m \times m$ and $l \times l$, respectively, so that

$$X = U \begin{bmatrix} D & O \\ O & O \end{bmatrix} V^T$$

where D is an $r \times r$ diagonal matrix with elements $\sigma_i = \sqrt{\lambda_i}$, known as the singular values of X, where $\lambda_i,\ i = 1, 2, \ldots, r$, are the nonzero eigenvalues of XX^T; matrices denoted as O comprise zero elements and are of appropriate dimensions.

- Taking into account the zero elements in the diagonal matrix, the previous matrix factorization can be rewritten as

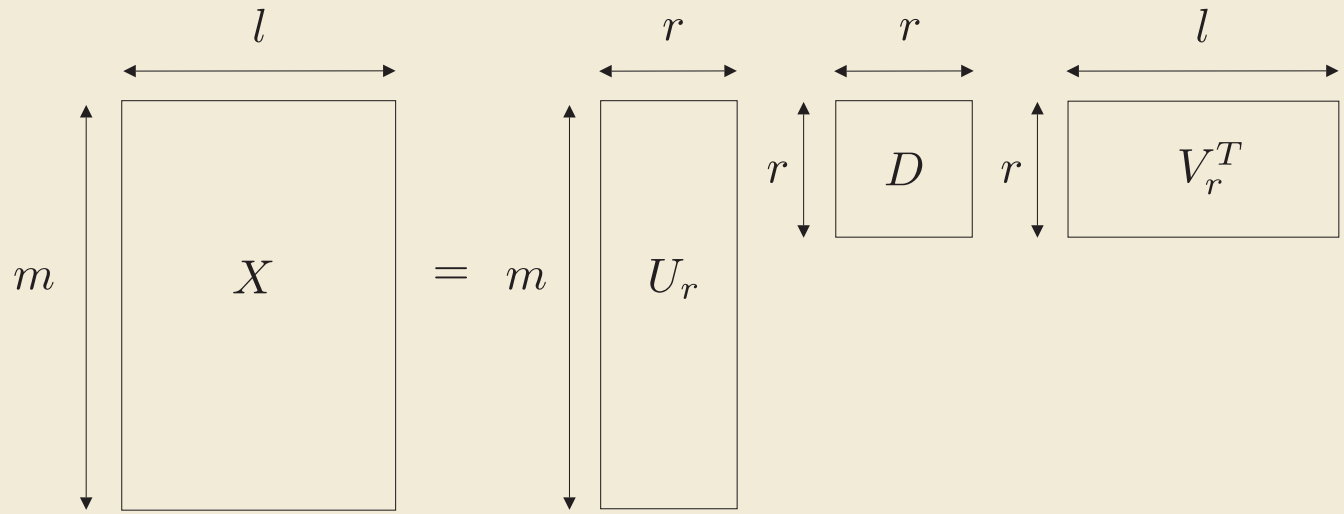
$$X = U_r D V_r^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T, \tag{7}$$

where

$$U_r := [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{m \times r}, \quad V_r := [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r] \in \mathbb{R}^{l \times r}. \tag{8}$$

Singular Value Decomposition

- The figure below offers a schematic illustration of the SVD factorization of an $m \times l$ matrix of rank r.

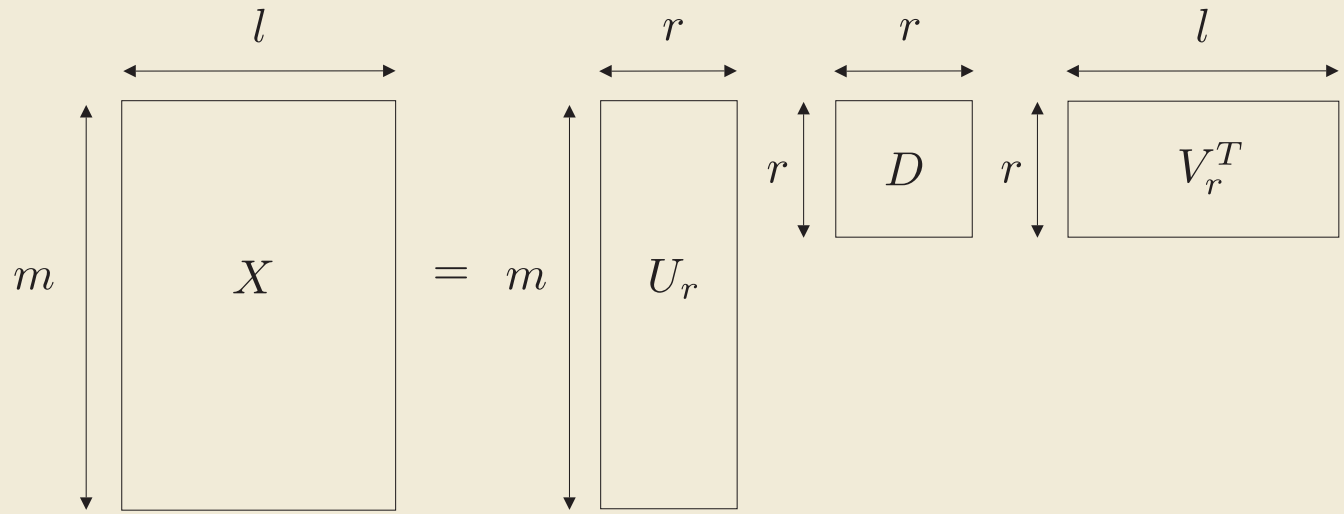


The $m \times l$ matrix X, of rank $r \leq \min(m, l)$, factorizes in terms of the matrices $U_r \in \mathbb{R}^{m \times r}$, $V_r \in \mathbb{R}^{l \times r}$ and the $r \times r$ diagonal matrix D.

- It turns out that, $\boldsymbol{u}_i \in \mathbb{R}^m,\ i = 1, 2, \ldots, r$, known as left singular vectors, are the eigenvectors corresponding to the nonzero eigenvalues of XX^T, and $\boldsymbol{v}_i \in \mathbb{R}^l,\ i = 1, 2, \ldots, r$, are the eigenvectors associated with the nonzero eigenvalues of X^TX and they are known as right singular vectors. Note that both, XX^T and X^TX, share the same eigenvalues.

Singular Value Decomposition

- The figure below offers a schematic illustration of the SVD factorization of an $m \times l$ matrix of rank r.



The $m \times l$ matrix X, of rank $r \leq \min(m, l)$, factorizes in terms of the matrices $U_r \in \mathbb{R}^{m \times r}$, $V_r \in \mathbb{R}^{l \times r}$ and the $r \times r$ diagonal matrix D.

- It turns out that, $\boldsymbol{u}_i \in \mathbb{R}^m,\ i = 1, 2, \ldots, r$, known as left singular vectors, are the eigenvectors corresponding to the nonzero eigenvalues of XX^T, and $\boldsymbol{v}_i \in \mathbb{R}^l,\ i = 1, 2, \ldots, r$, are the eigenvectors associated with the nonzero eigenvalues of X^TX and they are known as right singular vectors. Note that both, XX^T and X^TX, share the same eigenvalues.

Singular Value Decomposition

- **Proof**: By the respective definitions, we have

$$XX^T\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i,\ i = 1, 2, \ldots, r, \tag{9}$$

and

$$X^T X\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i,\ i = 1, 2, \ldots, r. \tag{10}$$

- Moreover, since XX^T and X^TX are symmetric matrices, it is known from linear algebra that their eigenvalues are real and the respective eigenvectors are **orthogonal**, which can then be normalized to unit norm to become **orthonormal**. It is a matter of simple algebra to show from (9) and (10) that,

$$\boldsymbol{u}_i = \frac{1}{\sigma_i} X\boldsymbol{v}_i,\ i = 1, 2, \ldots, r. \tag{11}$$

- Thus, we can write that

$$\sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T = X \sum_{i=1}^{r} \boldsymbol{v}_i \boldsymbol{v}_i^T = X \sum_{i=1}^{l} \boldsymbol{v}_i \boldsymbol{v}_i^T = XVV^T,$$

where we used the fact that for eigenvectors corresponding to $\sigma_i = 0$ ($\lambda_i = 0$), $i = r+1, \ldots, l$, $X\boldsymbol{v}_i = \boldsymbol{0}$. However, due to the orthonormality of $\boldsymbol{v}_i$, $VV^T = I$ and the claim in (7) has been proved.

Singular Value Decomposition

- **Proof**: By the respective definitions, we have

$$XX^T\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i,\ i = 1, 2, \ldots, r, \tag{9}$$

and

$$X^T X\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i,\ i = 1, 2, \ldots, r. \tag{10}$$

- Moreover, since XX^T and X^TX are symmetric matrices, it is known from linear algebra that their eigenvalues are real and the respective eigenvectors are **orthogonal**, which can then be normalized to unit norm to become **orthonormal**. It is a matter of simple algebra to show from (9) and (10) that,

$$\boldsymbol{u}_i = \frac{1}{\sigma_i} X\boldsymbol{v}_i,\ i = 1, 2, \ldots, r. \tag{11}$$

- Thus, we can write that

$$\sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T = X \sum_{i=1}^{r} \boldsymbol{v}_i \boldsymbol{v}_i^T = X \sum_{i=1}^{l} \boldsymbol{v}_i \boldsymbol{v}_i^T = XVV^T,$$

where we used the fact that for eigenvectors corresponding to $\sigma_i = 0$ ($\lambda_i = 0$), $i = r+1, \ldots, l$, $X\boldsymbol{v}_i = \boldsymbol{0}$. However, due to the orthonormality of $\boldsymbol{v}_i$, $VV^T = I$ and the claim in (7) has been proved.

Singular Value Decomposition

- **Proof**: By the respective definitions, we have

$$XX^T\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i,\ i = 1, 2, \ldots, r, \tag{9}$$

and

$$X^T X\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i,\ i = 1, 2, \ldots, r. \tag{10}$$

- Moreover, since XX^T and X^TX are symmetric matrices, it is known from linear algebra that their eigenvalues are real and the respective eigenvectors are **orthogonal**, which can then be normalized to unit norm to become **orthonormal**. It is a matter of simple algebra to show from (9) and (10) that,

$$\boldsymbol{u}_i = \frac{1}{\sigma_i} X\boldsymbol{v}_i,\ i = 1, 2, \ldots, r. \tag{11}$$

- Thus, we can write that

$$\sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T = X \sum_{i=1}^{r} \boldsymbol{v}_i \boldsymbol{v}_i^T = X \sum_{i=1}^{l} \boldsymbol{v}_i \boldsymbol{v}_i^T = XVV^T,$$

where we used the fact that for eigenvectors corresponding to $\sigma_i = 0$ ($\lambda_i = 0$), $i = r+1, \ldots, l$, $X\boldsymbol{v}_i = \mathbf{0}$. However, due to the orthonormality of $\boldsymbol{v}_i$, $VV^T = I$ and the claim in (7) has been proved.

Pseudo-inverse Matrix and SVD

- Let us now elaborate on the SVD expansion, in the context of the LS method. By the definition of the pseudoinverse, $X^{\dagger}$, and assuming the $N \times l$ $(N > l)$ data matrix to be full column rank $(r = l)$, then employing the SVD factorization of X, in the respective definition of the pseudoinverse, we get,

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_{LS} = XX^{\dagger}\boldsymbol{y} = X(X^T X)^{-1}X^T\boldsymbol{y} = U_l U_l^T\boldsymbol{y} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l] \begin{bmatrix} \boldsymbol{u}_1^T\boldsymbol{y} \\ \vdots \\ \boldsymbol{u}_l^T\boldsymbol{y} \end{bmatrix},$$

or

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{l} (\boldsymbol{u}_i^T\boldsymbol{y})\boldsymbol{u}_i. \tag{12}$$

This is the projection of y onto the column space of X, i.e., $\text{span}\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$, described via the orthonormal basis, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l\}$.

Pseudo-inverse Matrix and SVD

- Let us now elaborate on the SVD expansion, in the context of the LS method. By the definition of the pseudoinverse, $X^{\dagger}$, and assuming the $N \times l$ $(N > l)$ data matrix to be full column rank $(r = l)$, then employing the SVD factorization of X, in the respective definition of the pseudoinverse, we get,

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_{LS} = XX^{\dagger}\boldsymbol{y} = X(X^T X)^{-1}X^T\boldsymbol{y} = U_l U_l^T\boldsymbol{y} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l] \begin{bmatrix} \boldsymbol{u}_1^T\boldsymbol{y} \\ \vdots \\ \boldsymbol{u}_l^T\boldsymbol{y} \end{bmatrix},$$

or

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{l} (\boldsymbol{u}_i^T\boldsymbol{y})\boldsymbol{u}_i. \tag{12}$$

This is the projection of y onto the column space of X, i.e., $\text{span}\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$, described via the orthonormal basis, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l\}$.

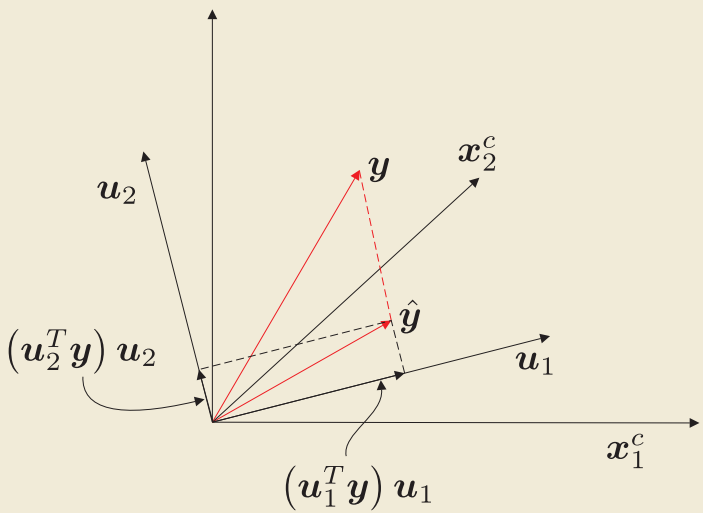
- Let us now elaborate on the SVD expansion, in the context of the LS method. By the definition of the pseudoinverse, $X^\dagger$, and assuming the $N \times l$ $(N > l)$ data matrix to be full column rank $(r = l)$, then employing the SVD factorization of X, in the respective definition of the pseudoinverse, we get,

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_{LS} = XX^\dagger \boldsymbol{y} = X(X^T X)^{-1} X^T \boldsymbol{y} = U_l U_l^T \boldsymbol{y} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l] \begin{bmatrix} \boldsymbol{u}_1^T \boldsymbol{y} \\ \vdots \\ \boldsymbol{u}_l^T \boldsymbol{y} \end{bmatrix},$$

or

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{l} (\boldsymbol{u}_i^T \boldsymbol{y}) \boldsymbol{u}_i. \tag{12}$$

This is the projection of $\boldsymbol{y}$ onto the column space of X, i.e., span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$, described via the orthonormal basis, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l\}$.



The eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2$, form an orthonormal basis, in span$\{\boldsymbol{x}_1^c, \boldsymbol{x}_2^c\}$; that is, the column space of X.

Pseudo-inverse Matrix and SVD

- Moreover, it is easily shown that we can write,

$$X^{\dagger} = (X^T X)^{-1} X^T = V_l D^{-1} U_l^T = \sum_{i=1}^{l} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- As a matter of fact, this is in line with the more general definition of a pseudo-inverse in linear algebra, including matrices which are not full rank (i.e., $X^T X$ is not invertible), namely

$$X^{\dagger} := V_r D^{-1} U_r^T = \sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- Pseudoinverse of a fat matrix: Using the above more general definition, it can be readily shown that the pseudoinverse of a fat matrix, i.e., $N < l$, is given by

$$X^{\dagger} = X^T (X X^T)^{-1}.$$

- The pseudoinverse of a fat matrix has an interesting implication. Given the linear system with less equations than unknowns,

$$X\boldsymbol{\theta} = \boldsymbol{y}, \; X \in \mathbb{R}^{N \times l}, \; \boldsymbol{\theta} \in \mathbb{R}^l, \; \boldsymbol{y} \in \mathbb{R}^N, \; N < l,$$

it is known to have infinite many solutions. However, the solution that has the least Euclidean norm is given by,

$$\hat{\boldsymbol{\theta}} = X^{\dagger} \boldsymbol{y}.$$

Equivalently, $\hat{\theta}$ solves the following constrained optimization task:

$$\text{minimize} \; ||\boldsymbol{\theta}||^2, \; \text{s.t.} \; X\boldsymbol{\theta} = \boldsymbol{y}.$$

Pseudo-inverse Matrix and SVD

- Moreover, it is easily shown that we can write,

$$X^{\dagger} = (X^T X)^{-1} X^T = V_l D^{-1} U_l^T = \sum_{i=1}^{l} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- As a matter of fact, this is in line with the more general definition of a pseudo-inverse in linear algebra, including matrices which are not full rank (i.e., $X^T X$ is not invertible), namely

$$X^{\dagger} := V_r D^{-1} U_r^T = \sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- Pseudoinverse of a fat matrix: Using the above more general definition, it can be readily shown that the pseudoinverse of a fat matrix, i.e., $N < l$, is given by

$$X^{\dagger} = X^T (X X^T)^{-1}.$$

- The pseudoinverse of a fat matrix has an interesting implication. Given the linear system with less equations than unknowns,

$$X\boldsymbol{\theta} = \boldsymbol{y}, \; X \in \mathbb{R}^{N \times l}, \; \boldsymbol{\theta} \in \mathbb{R}^l, \; \boldsymbol{y} \in \mathbb{R}^N, \; N < l,$$

it is known to have infinite many solutions. However, the solution that has the least Euclidean norm is given by,

$$\hat{\boldsymbol{\theta}} = X^{\dagger} \boldsymbol{y}.$$

Equivalently, $\hat{\boldsymbol{\theta}}$ solves the following constrained optimization task:

$$\text{minimize} \; ||\boldsymbol{\theta}||^2, \; \text{s.t.} \; X\boldsymbol{\theta} = \boldsymbol{y}.$$

Pseudo-inverse Matrix and SVD

- Moreover, it is easily shown that we can write,

$$X^{\dagger} = (X^T X)^{-1} X^T = V_l D^{-1} U_l^T = \sum_{i=1}^{l} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- As a matter of fact, this is in line with the more general definition of a pseudo-inverse in linear algebra, including matrices which are not full rank (i.e., $X^T X$ is not invertible), namely

$$X^{\dagger} := V_r D^{-1} U_r^T = \sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- Pseudoinverse of a fat matrix: Using the above more general definition, it can be readily shown that the pseudoinverse of a fat matrix, i.e., $N < l$, is given by

$$X^{\dagger} = X^T (X X^T)^{-1}.$$

- The pseudoinverse of a fat matrix has an interesting implication. Given the linear system with less equations than unknowns,

$$X\boldsymbol{\theta} = \boldsymbol{y}, \; X \in \mathbb{R}^{N \times l}, \; \boldsymbol{\theta} \in \mathbb{R}^l, \; \boldsymbol{y} \in \mathbb{R}^N, \; N < l,$$

it is known to have infinite many solutions. However, the solution that has the least Euclidean norm is given by,

$$\hat{\boldsymbol{\theta}} = X^{\dagger} \boldsymbol{y}.$$

Equivalently, $\hat{\boldsymbol{\theta}}$ solves the following constrained optimization task:

$$\text{minimize} \; ||\boldsymbol{\theta}||^2, \; \text{s.t.} \; X\boldsymbol{\theta} = \boldsymbol{y}.$$

Pseudo-inverse Matrix and SVD

- Moreover, it is easily shown that we can write,

$$X^{\dagger} = (X^T X)^{-1} X^T = V_l D^{-1} U_l^T = \sum_{i=1}^{l} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- As a matter of fact, this is in line with the more general definition of a pseudo-inverse in linear algebra, including matrices which are not full rank (i.e., $X^T X$ is not invertible), namely

$$X^{\dagger} := V_r D^{-1} U_r^T = \sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- Pseudoinverse of a fat matrix: Using the above more general definition, it can be readily shown that the pseudoinverse of a fat matrix, i.e., $N < l$, is given by

$$X^{\dagger} = X^T (X X^T)^{-1}.$$

- The pseudoinverse of a fat matrix has an interesting implication. Given the linear system with less equations than unknowns,

$$X\boldsymbol{\theta} = \boldsymbol{y}, \ X \in \mathbb{R}^{N \times l}, \ \boldsymbol{\theta} \in \mathbb{R}^l, \ \boldsymbol{y} \in \mathbb{R}^N, \ N < l,$$

it is known to have infinite many solutions. However, the solution that has the least Euclidean norm is given by,

$$\hat{\theta} = X^{\dagger} y.$$

Equivalently, $\hat{\theta}$ solves the following constrained optimization task:

$$\text{minimize} \ \ ||\theta||^2, \ \ \text{s.t.} \ X\theta = y.$$

Pseudo-inverse Matrix and SVD

- Moreover, it is easily shown that we can write,

$$X^{\dagger} = (X^T X)^{-1} X^T = V_l D^{-1} U_l^T = \sum_{i=1}^{l} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- As a matter of fact, this is in line with the more general definition of a pseudo-inverse in linear algebra, including matrices which are not full rank (i.e., $X^T X$ is not invertible), namely

$$X^{\dagger} := V_r D^{-1} U_r^T = \sum_{i=1}^{r} \frac{1}{\sigma_i} \boldsymbol{v}_i \boldsymbol{u}_i^T.$$

- Pseudoinverse of a fat matrix: Using the above more general definition, it can be readily shown that the pseudoinverse of a fat matrix, i.e., $N < l$, is given by

$$X^{\dagger} = X^T (X X^T)^{-1}.$$

- The pseudoinverse of a fat matrix has an interesting implication. Given the linear system with less equations than unknowns,

$$X\boldsymbol{\theta} = \boldsymbol{y}, \ X \in \mathbb{R}^{N \times l}, \ \boldsymbol{\theta} \in \mathbb{R}^l, \ \boldsymbol{y} \in \mathbb{R}^N, \ N < l,$$

it is known to have infinite many solutions. However, the solution that has the least Euclidean norm is given by,

$$\hat{\boldsymbol{\theta}} = X^{\dagger} \boldsymbol{y}.$$

Equivalently, $\hat{\boldsymbol{\theta}}$ solves the following constrained optimization task:

$$\text{minimize} \ \ ||\boldsymbol{\theta}||^2, \ \ \text{s.t.} \ X\boldsymbol{\theta} = \boldsymbol{y}.$$

Ridge Regression

- By definition, in ridge regression, the minimizer results as

$$\hat{\boldsymbol{\theta}}_R = \arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - X\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|^2 \right\},$$

  where λ is a user-defined parameter that controls the importance of the regularizing term.

- Taking the gradient w.r. to θ and equating to zero results in

$$\hat{\boldsymbol{\theta}}_R = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}.$$

- Looking at the above equation, we readily observe a) its stabilizing'' effect from the numerical point of view, when $X^T X$ has large condition number and b) its biasing effect on the (unbiased) LS solution. Note that ridge regression provides a solution even if $X^T X$ is not invertible, as it is the case when $N < l$.
- Assuming a full column rank matrix, X, we obtain,

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_R = U_l D (D^2 + \lambda I)^{-1} D U_l^T \boldsymbol{y},$$

  or

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{l} \frac{\sigma_i^2}{\lambda + \sigma_i^2} (\boldsymbol{u}_i^T \boldsymbol{y}) \boldsymbol{u}_i. \tag{13}$$

Ridge Regression

- By definition, in ridge regression, the minimizer results as

$$\hat{\boldsymbol{\theta}}_R = \arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - X\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|^2 \right\},$$

  where λ is a user-defined parameter that controls the importance of the regularizing term.

- Taking the gradient w.r. to $\boldsymbol{\theta}$ and equating to zero results in

$$\hat{\boldsymbol{\theta}}_R = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}.$$

- Looking at the above equation, we readily observe a) its "stabilizing" effect from the numerical point of view, when $X^T X$ has large condition number and b) its biasing effect on the (unbiased) LS solution. Note that ridge regression provides a solution even if $X^T X$ is not invertible, as it is the case when $N < l$.
- Assuming a full column rank matrix, X, we obtain,

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_R = U_l D (D^2 + \lambda I)^{-1} D U_l^T \boldsymbol{y},$$

  or

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{l} \frac{\sigma_i^2}{\lambda + \sigma_i^2} (\boldsymbol{u}_i^T \boldsymbol{y}) \boldsymbol{u}_i. \tag{13}$$

Ridge Regression

- By definition, in ridge regression, the minimizer results as

$$\hat{\boldsymbol{\theta}}_R = \arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - X\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|^2 \right\},$$

  where λ is a user-defined parameter that controls the importance of the regularizing term.

- Taking the gradient w.r. to $\boldsymbol{\theta}$ and equating to zero results in

$$\hat{\boldsymbol{\theta}}_R = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}.$$

- Looking at the above equation, we readily observe a) its stabilizing'' effect from the numerical point of view, when $X^T X$ has large condition number and b) its biasing effect on the (unbiased) LS solution. Note that ridge regression provides a solution even if $X^T X$ is not invertible, as it is the case when $N < l$.

- Assuming a full column rank matrix, X, we obtain,

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_R = U_l D (D^2 + \lambda I)^{-1} D U_l^T \boldsymbol{y},$$

  or

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{l} \frac{\sigma_i^2}{\lambda + \sigma_i^2} (\boldsymbol{u}_i^T \boldsymbol{y}) \boldsymbol{u}_i. \tag{13}$$

Ridge Regression

- By definition, in ridge regression, the minimizer results as

$$\hat{\boldsymbol{\theta}}_R = \arg\min_{\boldsymbol{\theta}} \left\{ \|\boldsymbol{y} - X\boldsymbol{\theta}\|^2 + \lambda \|\boldsymbol{\theta}\|^2 \right\},$$

  where λ is a user-defined parameter that controls the importance of the regularizing term.

- Taking the gradient w.r. to $\boldsymbol{\theta}$ and equating to zero results in

$$\hat{\boldsymbol{\theta}}_R = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}.$$

- Looking at the above equation, we readily observe a) its stabilizing'' effect from the numerical point of view, when $X^T X$ has large condition number and b) its biasing effect on the (unbiased) LS solution. Note that ridge regression provides a solution even if $X^T X$ is not invertible, as it is the case when $N < l$.
- Assuming a full column rank matrix, X, we obtain,

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\theta}}_R = U_l D (D^2 + \lambda I)^{-1} D U_l^T \boldsymbol{y},$$

  or

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{l} \frac{\sigma_i^2}{\lambda + \sigma_i^2} (\boldsymbol{u}_i^T \boldsymbol{y}) \boldsymbol{u}_i. \tag{13}$$

Ridge Regression

- Comparing (13) and (12), we observe that the components of the projection of $\boldsymbol{y}$ onto the span$\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l\}$ (span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$) are shrunk with respect to their LS counterpart. Moreover, the shrinking level depends on the singular values, σ_i; the smaller the value of σ_i is the higher the shrinkage of the corresponding component becomes. This has the following interesting geometric interpretation.
- Recall that $X^T X$ is a scaled version of the sample covariance matrix for centered regressors. Also, by the definition of the $\boldsymbol{v}_i$'s, we have,

$$(X^T X)\boldsymbol{v}_i = \sigma_i^2 \boldsymbol{v}_i, \quad i = 1, 2, \ldots, l,$$

and in a compact form,

$$\begin{aligned} (X^T X)V_l &= V_l \text{diag}\{\sigma_1^2, \ldots, \sigma_l^2\} \Rightarrow \\ (X^T X) &= V_l D^2 V_l^T = \sum_{i=1}^{l} \sigma_i^2 \boldsymbol{v}_i \boldsymbol{v}_i^T. \end{aligned}$$

- From the last equation, note that, the (scaled) sample covariance matrix is written as a sum of rank one matrices, $\boldsymbol{v}_i \boldsymbol{v}_i^T$, each one weighted by the square of respective singular value, σ_i^2.

Ridge Regression

- Comparing (13) and (12), we observe that the components of the projection of $\boldsymbol{y}$ onto the span$\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l\}$ (span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$) are shrunk with respect to their LS counterpart. Moreover, the shrinking level depends on the singular values, σ_i; the smaller the value of σ_i is the higher the shrinkage of the corresponding component becomes. This has the following interesting geometric interpretation.
- Recall that $X^T X$ is a scaled version of the sample covariance matrix for centered regressors. Also, by the definition of the $\boldsymbol{v}_i$'s, we have,

$$(X^T X)\boldsymbol{v}_i = \sigma_i^2 \boldsymbol{v}_i, \quad i = 1, 2, \ldots, l,$$

and in a compact form,

$$\begin{aligned} (X^T X)V_l &= V_l \text{diag}\{\sigma_1^2, \ldots, \sigma_l^2\} \Rightarrow \\ (X^T X) &= V_l D^2 V_l^T = \sum_{i=1}^{l} \sigma_i^2 \boldsymbol{v}_i \boldsymbol{v}_i^T. \end{aligned}$$

- From the last equation, note that, the (scaled) sample covariance matrix is written as a sum of rank one matrices, $\boldsymbol{v}_i \boldsymbol{v}_i^T$, each one weighted by the square of respective singular value, σ_i^2.

Ridge Regression

- Comparing (13) and (12), we observe that the components of the projection of $\boldsymbol{y}$ onto the span$\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l\}$ (span$\{\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_l^c\}$) are shrunk with respect to their LS counterpart. Moreover, the shrinking level depends on the singular values, σ_i; the smaller the value of σ_i is the higher the shrinkage of the corresponding component becomes. This has the following interesting geometric interpretation.
- Recall that $X^T X$ is a scaled version of the sample covariance matrix for centered regressors. Also, by the definition of the $\boldsymbol{v}_i$'s, we have,

$$(X^T X)\boldsymbol{v}_i = \sigma_i^2 \boldsymbol{v}_i, \quad i = 1, 2, \ldots, l,$$

and in a compact form,

$$\begin{aligned} (X^T X)V_l &= V_l \text{diag}\{\sigma_1^2, \ldots, \sigma_l^2\} \Rightarrow \\ (X^T X) &= V_l D^2 V_l^T = \sum_{i=1}^{l} \sigma_i^2 \boldsymbol{v}_i \boldsymbol{v}_i^T. \end{aligned}$$

- From the last equation, note that, the (scaled) sample covariance matrix is written as a sum of rank one matrices, $\boldsymbol{v}_i \boldsymbol{v}_i^T$, each one weighted by the square of respective singular value, σ_i^2.

Ridge Regression

- Let us now define,

$$\boldsymbol{q}_j := X\boldsymbol{v}_j = \begin{bmatrix} \boldsymbol{x}_1^T\boldsymbol{v}_j \\ \vdots \\ \boldsymbol{x}_N^T\boldsymbol{v}_j \end{bmatrix} \in \mathbb{R}^N, \ j = 1, 2, \ldots, l. \tag{14}$$

- Note that $\boldsymbol{q}_j$ is a vector in the column space of X. Moreover, the respective square norm of $\boldsymbol{q}_j$ is given by,

$$\sum_{n=1}^{N} q_j^2(n) = \boldsymbol{q}_j^T\boldsymbol{q}_j = \boldsymbol{v}_j^T X^T X \boldsymbol{v}_j = \boldsymbol{v}_j^T \left(\sum_{i=1}^{l} \sigma_i^2 \boldsymbol{v}_i \boldsymbol{v}_i^T \right) \boldsymbol{v}_j = \sigma_j^2,$$

  due to the orthonormality of the $\boldsymbol{v}_j$'s.

- That is, σ_j^2 is equal to the (scaled) sample variance of the elements of $\boldsymbol{q}_j$. However, by the definition in (14), this is the sample variance of the projections of the input vectors (regressors), $\boldsymbol{x}_n, \ n = 1, 2, \ldots, N$, along the direction $\boldsymbol{v}_j$. The larger the value of σ_j is the larger the **spread** of the (input) data along the respective direction becomes. This is geometrically shown in the figure of the next slide.

Ridge Regression

- Let us now define,

$$\boldsymbol{q}_j := X\boldsymbol{v}_j = \begin{bmatrix} \boldsymbol{x}_1^T\boldsymbol{v}_j \\ \vdots \\ \boldsymbol{x}_N^T\boldsymbol{v}_j \end{bmatrix} \in \mathbb{R}^N, \; j = 1, 2, \ldots, l. \tag{14}$$

- Note that $\boldsymbol{q}_j$ is a vector in the column space of X. Moreover, the respective square norm of $\boldsymbol{q}_j$ is given by,

$$\sum_{n=1}^{N} q_j^2(n) = \boldsymbol{q}_j^T\boldsymbol{q}_j = \boldsymbol{v}_j^T X^T X \boldsymbol{v}_j = \boldsymbol{v}_j^T \left(\sum_{i=1}^{l} \sigma_i^2 \boldsymbol{v}_i \boldsymbol{v}_i^T \right) \boldsymbol{v}_j = \sigma_j^2,$$

due to the orthonormality of the $\boldsymbol{v}_j$'s.

- That is, σ_j^2 is equal to the (scaled) sample variance of the elements of $\boldsymbol{q}_j$. However, by the definition in (14), this is the sample variance of the projections of the input vectors (regressors), $\boldsymbol{x}_n, \; n = 1, 2, \ldots, N$, along the direction $\boldsymbol{v}_j$. The larger the value of σ_j is the larger the **spread** of the (input) data along the respective direction becomes. This is geometrically shown in the figure of the next slide.

Ridge Regression

- Let us now define,

$$\boldsymbol{q}_j := X\boldsymbol{v}_j = \begin{bmatrix} \boldsymbol{x}_1^T\boldsymbol{v}_j \\ \vdots \\ \boldsymbol{x}_N^T\boldsymbol{v}_j \end{bmatrix} \in \mathbb{R}^N, \ j = 1, 2, \ldots, l. \tag{14}$$

- Note that $\boldsymbol{q}_j$ is a vector in the column space of X. Moreover, the respective square norm of $\boldsymbol{q}_j$ is given by,

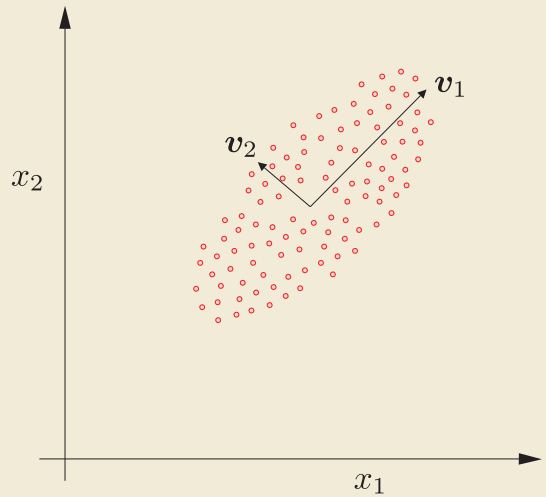
$$\sum_{n=1}^{N} q_j^2(n) = \boldsymbol{q}_j^T\boldsymbol{q}_j = \boldsymbol{v}_j^T X^T X \boldsymbol{v}_j = \boldsymbol{v}_j^T \left(\sum_{i=1}^{l} \sigma_i^2 \boldsymbol{v}_i \boldsymbol{v}_i^T \right) \boldsymbol{v}_j = \sigma_j^2,$$

due to the orthonormality of the $\boldsymbol{v}_j$'s.

- That is, σ_j^2 is equal to the (scaled) sample variance of the elements of $\boldsymbol{q}_j$. However, by the definition in (14), this is the sample variance of the projections of the input vectors (regressors), $\boldsymbol{x}_n, \ n = 1, 2, \ldots, N$, along the direction $\boldsymbol{v}_j$. The larger the value of σ_j is the larger the **spread** of the (input) data along the respective direction becomes. This is geometrically shown in the figure of the next slide.

Ridge Regression

- The figure illustrates the geometry of the left eigenvectors resulting from the SVD of a matrix.



The singular vector $\boldsymbol{v}_1$, which is associated with the the singular value $\sigma_1 > \sigma_2$, points to the direction where most of the (variance) activity in the data space happens. The variance in the direction of $\boldsymbol{v}_2$ is smaller.

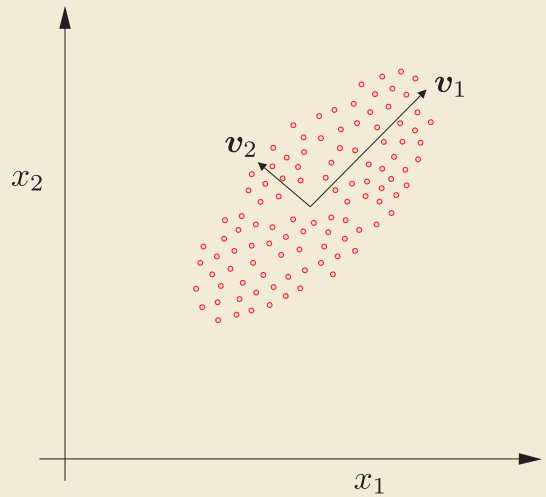
- Moreover, a consequence of (11) ($\boldsymbol{u}_i = \frac{1}{\sigma_i} X \boldsymbol{v}_i$) is that

$$\boldsymbol{q}_j = X \boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j.$$

In other words, $\boldsymbol{u}_j$ points in the direction of $\boldsymbol{q}_j$. Thus, (13) suggests that while projecting $\boldsymbol{y}$ onto the column space of X, the directions, $\boldsymbol{u}_j$, associated with larger values of variance are **weighted more heavily** than the rest. Ridge regression respects and assigns higher weights to the more informative directions, where most of the data "activity" takes place.

Ridge Regression

- The figure illustrates the geometry of the left eigenvectors resulting from the SVD of a matrix.



The singular vector $\boldsymbol{v}_1$, which is associated with the the singular value $\sigma_1 > \sigma_2$, points to the direction where most of the (variance) activity in the data space happens. The variance in the direction of $\boldsymbol{v}_2$ is smaller.

- Moreover, a consequence of (11) ($\boldsymbol{u}_i = \frac{1}{\sigma_i} X \boldsymbol{v}_i$) is that

$$\boldsymbol{q}_j = X \boldsymbol{v}_j = \sigma_j \boldsymbol{u}_j.$$

In other words, $\boldsymbol{u}_j$ points in the direction of $\boldsymbol{q}_j$. Thus, (13) suggests that while projecting $\boldsymbol{y}$ onto the column space of X, the directions, $\boldsymbol{u}_j$, associated with larger values of variance are **weighted more heavily** than the rest. Ridge regression respects and assigns higher weights to the more informative directions, where most of the data "activity" takes place.

Principal Components Regression

- Thus, the effect of the ridge regression is to enforce a shrinking rule, which decreases the contribution of the less important of the components, $\boldsymbol{u}_i$, in the respective summation. This can be considered as a soft shrinkage rule.
- An alternative path is to adopt a hard thresholding rule and keep only the, say, m most significant directions, known as the principal axes or directions, and forget the rest by setting the respective weights equal to zero. Equivalently, we can write

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \hat{\theta}_i \boldsymbol{u}_i, \text{ where } \hat{\theta}_i = \boldsymbol{u}_i^T \boldsymbol{y}, \ i = 1, 2, \ldots, m.$$

- Furthermore, recalling $\boldsymbol{u}_i = \frac{1}{\sigma_i} X \boldsymbol{v}_i$ ((11)) we have that,

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} X \boldsymbol{v}_i,$$

  or equivalently, the weights for the expansion of the solution in terms of the input data can be expressed as

$$\boldsymbol{\theta} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} \boldsymbol{v}_i.$$

- In other words, the prediction $\hat{y}$ is performed in a **subspace** of the column space of X, which is **spanned** by the m principal axes; that is, the subspace where most of the data activity, from the variance point of view, takes place. Such a rationale forms the basis of what is known as dimensionality reduction.

Principal Components Regression

- Thus, the effect of the ridge regression is to enforce a shrinking rule, which decreases the contribution of the less important of the components, $\boldsymbol{u}_i$, in the respective summation. This can be considered as a soft shrinkage rule.
- An alternative path is to adopt a hard thresholding rule and keep only the, say, m most significant directions, known as the principal axes or directions, and forget the rest by setting the respective weights equal to zero. Equivalently, we can write

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \hat{\theta}_i \boldsymbol{u}_i, \text{ where } \hat{\theta}_i = \boldsymbol{u}_i^T \boldsymbol{y}, \ i = 1, 2, \ldots, m.$$

- Furthermore, recalling $\boldsymbol{u}_i = \frac{1}{\sigma_i} X \boldsymbol{v}_i$ ((11)) we have that,

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} X \boldsymbol{v}_i,$$

or equivalently, the weights for the expansion of the solution in terms of the input data can be expressed as

$$\boldsymbol{\theta} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} \boldsymbol{v}_i.$$

- In other words, the prediction $\hat{y}$ is performed in a **subspace** of the column space of X, which is **spanned** by the m principal axes; that is, the subspace where most of the data activity, from the variance point of view, takes place. Such a rationale forms the basis of what is known as dimensionality reduction.

Principal Components Regression

- Thus, the effect of the ridge regression is to enforce a shrinking rule, which decreases the contribution of the less important of the components, $\boldsymbol{u}_i$, in the respective summation. This can be considered as a soft shrinkage rule.
- An alternative path is to adopt a hard thresholding rule and keep only the, say, m most significant directions, known as the principal axes or directions, and forget the rest by setting the respective weights equal to zero. Equivalently, we can write

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \hat{\theta}_i \boldsymbol{u}_i, \text{ where } \hat{\theta}_i = \boldsymbol{u}_i^T \boldsymbol{y}, \ i = 1, 2, \ldots, m.$$

- Furthermore, recalling $\boldsymbol{u}_i = \frac{1}{\sigma_i} X \boldsymbol{v}_i$ ((11)) we have that,

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} X \boldsymbol{v}_i,$$

or equivalently, the weights for the expansion of the solution in terms of the input data can be expressed as

$$\boldsymbol{\theta} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} \boldsymbol{v}_i.$$

- In other words, the prediction $\hat{y}$ is performed in a **subspace** of the column space of X, which is **spanned** by the m principal axes; that is, the subspace where most of the data activity, from the variance point of view, takes place. Such a rationale forms the basis of what is known as dimensionality reduction.

Principal Components Regression

- Thus, the effect of the ridge regression is to enforce a shrinking rule, which decreases the contribution of the less important of the components, $\boldsymbol{u}_i$, in the respective summation. This can be considered as a soft shrinkage rule.
- An alternative path is to adopt a hard thresholding rule and keep only the, say, m most significant directions, known as the principal axes or directions, and forget the rest by setting the respective weights equal to zero. Equivalently, we can write

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \hat{\theta}_i \boldsymbol{u}_i, \text{ where } \hat{\theta}_i = \boldsymbol{u}_i^T \boldsymbol{y}, \ i = 1, 2, \ldots, m.$$

- Furthermore, recalling $\boldsymbol{u}_i = \frac{1}{\sigma_i} X \boldsymbol{v}_i$ ((11)) we have that,

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} X \boldsymbol{v}_i,$$

  or equivalently, the weights for the expansion of the solution in terms of the input data can be expressed as

$$\boldsymbol{\theta} = \sum_{i=1}^{m} \frac{\hat{\theta}_i}{\sigma_i} \boldsymbol{v}_i.$$

- In other words, the prediction $\hat{\boldsymbol{y}}$ is performed in a **subspace** of the column space of X, which is **spanned** by the m principal axes; that is, the subspace where most of the data activity, from the variance point of view, takes place. Such a rationale forms the basis of what is known as dimensionality reduction.