# MATH 829: Introduction to Data Mining and Analysis
# Linear Regression: old and new (part 2)

Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

February 17, 2017



## The Gauss–Markov theorem

As before, we assume:

$$Y = X_1\beta_1 + \cdots + X_p\beta_p = X^T\beta.$$

We observe $\mathbf{X} \in \mathbb{R}^{n\times p}$, $\mathbf{Y} \in \mathbb{R}^n$. Then

$$\hat\beta_{\rm LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}.$$

Under some natural assumptions, we can show that $\hat\beta_{\rm LS}$ is the *best linear unbiased estimator* for $\beta$.

**Assumptions:** $\mathbf{Y} = \mathbf{X}\beta + \epsilon$, where $\epsilon \in \mathbb{R}^n$ with:

1. $E(\epsilon_i) = 0$.
2. $\mathrm{Var}(\epsilon_i) = \sigma^2 < \infty$.
3. $\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0$ for all $i \neq j$.

Note:

- (3) means that the errors are *uncorrelated*. In particular, (3) holds if the errors are independent.
- The errors need not be normal, nor independent, nor identically distributed.



## Gauss–Markov (cont.)

Remarks: In our model $\mathbf{Y} = \mathbf{X}\beta + \epsilon$,

- $\mathbf{X}$ is fixed.
- $\epsilon$ is random.
- $\mathbf{Y}$ is random.
- $\beta$ is fixed, but unobservable.

We want to estimate $\beta$.

A *linear* estimator of $\beta$, is an estimator of the form $\hat\beta = C\mathbf{Y}$, where $C = (c_{ij}) \in \mathbb{R}^{p\times n}$ is a matrix, and

$$c_{ij} = c_{ij}(\mathbf{X}).$$

Note: $\hat\beta$ is random since $\mathbf{Y}$ is assumed to be random.
In particular, $\hat\beta_{\rm LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ is a linear estimator with $C = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$.

An estimator is *unbiased* if $E(\hat\beta) = \beta$.



## Gauss–Markov (cont.)

Ultimately, we want to use $\hat\beta$ to predict $Y$, i.e.,
$\hat Y_i = X_{i1}\hat\beta_1 + X_{i2}\hat\beta_2 + \cdots + X_{ip}\hat\beta_p$.
We want to control to error of the prediction.

We define the mean squared error (MSE) of a linear combination of the coefficients of $\hat\beta$ by

$$\mathrm{MSE}(a^T\hat\beta) = E\left[\left(\sum_{i=1}^n a_i(\hat\beta_i - \beta_i)\right)^2\right] \qquad (a \in \mathbb{R}^p).$$

**Theorem (Gauss–Markov theorem)**

*Suppose $\mathbf{Y} = \mathbf{X}\beta + \epsilon$ where $\epsilon$ satisfies the previous assumptions. Let $\hat\beta = C\mathbf{Y}$ be a linear unbiased estimator of $\beta$. Then for all $a \in \mathbb{R}^p$,*

$$MSE(a^T\hat\beta_{LS}) \leq MSE(a^T\hat\beta).$$

We say that $\hat\beta_{\rm LS}$ is the **best linear unbiased estimator** (BLUE) of $\beta$.

## Gauss–Markov (cont.)

**The bias-variance tradeoff**
Let $Z = a^T\beta$ and $\hat{Z} = a^T\hat{\beta}$. (Note: $Z$ is non-random). Then

$$
\begin{aligned}
MSE(a^T\hat{\beta}) &= E\left[(a^T(\hat{\beta}-\beta))^2\right] = E\left[(\hat{Z}-Z)^2\right] \\
&= E(Z^2 - 2Z\hat{Z} + \hat{Z}^2) \\
&= E(Z^2) - 2E(Z\hat{Z}) + E(\hat{Z}^2) \\
&= Z^2 - 2ZE(\hat{Z}) + \mathrm{Var}(\hat{Z}) + E(\hat{Z})^2 \\
&= \underbrace{(Z - E(\hat{Z}))^2}_{\text{bias}^2} + \underbrace{\mathrm{Var}(\hat{Z})}_{\text{variance}}.
\end{aligned}
$$

Therefore, $\mathrm{MSE} = \text{Bias-squared} + \text{Variance}$.

As a result, if $\hat{\beta}$ is unbiased, then $\mathrm{MSE}(a^T\beta) = \mathrm{Var}(\hat{Z})$.



## Gauss–Markov (cont.)

We now prove the Gauss–Markov theorem. Using the bias-variance decomposition of MSE, it suffices to show that for every unbiased estimator of $\beta$,

$$\mathrm{Var}(a^T\hat{\beta}_{\mathrm{LS}}) \leq \mathrm{Var}(a^T\hat{\beta}) \qquad \forall a \in \mathbb{R}^p.$$

**Proof.** Let $\hat{\beta} = C\mathbf{Y}$ where $C = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + D$ for some $D \in \mathbb{R}^{p\times n}$. We will compute $E(\hat{\beta})$ and $\mathrm{Var}(a^T\hat{\beta})$.

$$
\begin{aligned}
E(\hat{\beta}) &= E\left[((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + D)\mathbf{Y}\right] \\
&= E\left[((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + D)(\mathbf{X}\beta + \epsilon)\right] \\
&= (I + D\mathbf{X})\beta.
\end{aligned}
$$

In order for $\hat{\beta}$ to be unbiased, we need $D\mathbf{X} = 0$.

We now compute $\mathrm{Var}(a^T\hat{\beta})$.



## Gauss–Markov (cont.)

Recall:

$$\mathrm{Var}(a^T\hat{\beta}) = a^T\Sigma a,$$

where $\Sigma = (\mathrm{Cov}(\hat{\beta}_i, \hat{\beta}_j)) = \mathrm{Var}(\hat{\beta})$. More generally, if $A \in \mathbb{R}^{p\times p}$, then

$$\mathrm{Var}(A\hat{\beta}) = A\,\mathrm{Var}(\hat{\beta})A^T.$$

Using these formulas, we obtain

$$
\begin{aligned}
\mathrm{Var}(\hat{\beta}) &= \mathrm{Var}(C\mathbf{Y}) \\
&= C\,\mathrm{Var}(\mathbf{Y})C^T = \sigma^2 CC^T \\
&= \sigma^2((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + D)((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T + D)^T \\
&= \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1} \\
&\quad + \sigma^2\left[(\mathbf{X}^T\mathbf{X})^{-1}\underbrace{\mathbf{X}^T D^T}_{=(DX)^T=0} + \underbrace{D\mathbf{X}}_{=0}(\mathbf{X}^T\mathbf{X})^{-1} + DD^T\right] \\
&= \sigma^2\left[(X^TX)^{-1} + DD^T\right].
\end{aligned}
$$



## Gauss–Markov

We have shown:

$$\mathrm{Var}(\hat{\beta}) = \sigma^2(X^TX)^{-1} + \sigma^2 DD^T.$$

Note that the matrices $(X^TX)^{-1}$ and $DD^T$ are positive semidefinite.

Therefore,

$$
\begin{aligned}
\mathrm{Var}(a^T\hat{\beta}) = a^T(\sigma^2(X^TX)^{-1} + \sigma^2 DD^T)a &\geq a^T\sigma^2(X^TX)^{-1}a \\
&= \mathrm{Var}(a^T\hat{\beta}_{\mathrm{LS}}).
\end{aligned}
$$

This concludes the proof. $\square$

## Back to bias-variance tradeoff

We saw that

$$\mathrm{MSE}(a^T\hat\beta) = (a^T\beta - E(a^T\hat\beta))^2 + \mathrm{Var}(a^T\hat\beta).$$

Moreover, according to the Gauss–Markov theorem, for every **unbiased estimator** $\hat\beta$,

$$\mathrm{MSE}(a^T\hat\beta_{\mathrm{LS}}) = \mathrm{Var}(a^T\hat\beta_{\mathrm{LS}}) \le \mathrm{MSE}(a^T\hat\beta)$$

**Problems with least squares:**

1. Least squares estimates often have large variance, and can have low prediction accuracy (especially when working with small samples).
2. Generally, all the regression coefficients $\beta_i$ are nonzero, making the model hard to interpret. Often, we want to identify the *relevant* variables to get the "big picture".

We can often increase the prediction accuracy by sacrificing a little bit of bias to reduce the variance of the estimator.
We will later examine some useful alternatives to least squares.



## Training error and test error

A natural way to improve least squares is to force some of the coefficients to be zero.

- Resulting estimator is biased, but can benefit from the bias-variance tradeoff.
- Model is easier to interpret.

Complexity of the model:

- A complex model that fits data very well will often make poor predictions. **Overfitting**.
- On the other hand, a very simple model may not capture the complexity of the data. **Underfitting**.

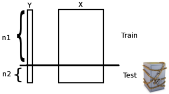
To test the ability of a model to predict new values:

1. We split our data into 2 parts (training data and test data) as uniformly as possible. People often use $75\%$ training, $25\%$ test.
2. We fit our model using the training data only. (This minimizes the **training error**).
3. We use the fitted model to predict values of the test data and compute the **test error**.



## Training error and test error (cont.)

Splitting data into training/test data:




In the case of least squares:

1. $\hat\beta = (X_{\mathrm{train}}^T X_{\mathrm{train}})^{-1} X_{\mathrm{train}}^T Y_{\mathrm{train}}$.
2. $\widehat{Y}_{\mathrm{test}} = X_{\mathrm{test}}\hat\beta$.
3. Test error:

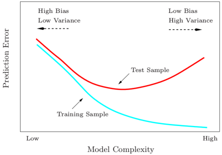
$$\mathrm{MSE}_{\mathrm{test}} = \frac{1}{n_2}\sum_{i=1}^{n_2}(\widehat{Y}_{\mathrm{test},i} - Y_{\mathrm{test},i})^2.$$

We choose a model that minimizes the test error.



## Training error and test error (cont.)

Typical behavior of the test and training error, as model complexity is varied.




ESL, Fig 2.11.

## Training sets and test sets (Python)

Scikit-learn provides a function to split the data automatically for us.

```
from sklearn.cross_validation import train_test_split

# Split data into training and test sets
X_train, X_test, y_train, y_test =
  train_test_split(X, y, test_size=0.25,
  random_state=42)

# Fit model on training data
lin_model = LinearRegression(fit_intercept=True)
lin_model.fit(X_train,y_train)

# Returns the coefficient of determination R^2.
lin_model.score(X_test, y_test)
```



## The coefficient of determination

- Regression models are often ranked using the *coefficient of determination* called "R squared" and denoted $R^2$.

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \overline{y})^2}.$$

- In some sense, the $R^2$ measures "how much better" is the prediction, compared to a constant prediction equal to the average of the $y_i s$.
- The score method in sklearn returns the $R^2$.
- We want a model with a **test** $R^2$ as close to $1$ as possible.