Machine Learning

Linear Regression

Matteo Papini

Credits to Alberto Maria Metelli and Francesco Trovò

Politecnico di Milano

Outline

1. Administrivia
2. Regression Problem
3. Practical Example

Outline

1. Administrivia
2. Regression Problem
3. Practical Example

Administrivia

Matteo Papini
Assistant Professor

- Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)
  - bulding 21
  - first floor
  - office 19
- Email: `matteo.papini@polimi.it`

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep
  `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep
  `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep
  `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep
  `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep
  `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep
  `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Course Material

- Course materials of lectures and exercise sessions uploaded to WeBeep
  `https://webeep.polimi.it/course/view.php?id=12810`
- Material of exercise sessions
  - **Recorded exercise sessions**
  - **Slides** for lectures and exercise sessions **(some topics are only covered in exercise sessions)**
  - **Pdf of exercises**: solutions uploaded after the exercise session
  - **Colab notebooks (we require you to understand the code at the exam**, e.g., find and rectify mistakes).
- **Review** (on your own):
  - Linear Algebra and Statistics Recap (pdf)
  - Introduction to Python (Colab notebook)

Suggested Literature

- Bishop, C.M., “Pattern recognition and machine learning”, 2006, Springer
  https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf
- James, G., Witten, D., Hastie, T., Tibshirani, R., “An introduction to statistical learning”, 2013, Springer
  https://www.statlearning.com/

Outline

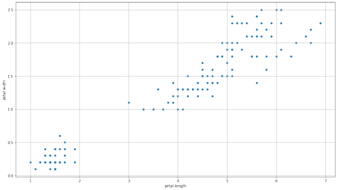
1. Administrivia
2. Regression Problem
3. Practical Example

Where Everything Starts

Consider the **Iris Dataset**
(`https://en.wikipedia.org/wiki/Iris_flower_data_set`):

- Sepal length
- Sepal width
- Petal length
- Petal width
- Species (Iris setosa, Iris virginica e Iris versicolor)

$N = 150$ total samples (50 per species)



Example of Dataset

Sepal length	Sepal width	Petal length	Petal width	Class
5.1000	3.5000	1.4000	0.2000	Iris-setosa
4.9000	3.0000	1.4000	0.2000	Iris-setosa
4.7000	3.2000	1.3000	0.2000	Iris-setosa
4.6000	3.1000	1.5000	0.2000	Iris-setosa
5.0000	3.6000	1.4000	0.2000	Iris-setosa
5.4000	3.9000	1.7000	0.4000	Iris-setosa
4.6000	3.4000	1.4000	0.3000	Iris-setosa
5.0000	3.4000	1.5000	0.2000	Iris-setosa
4.4000	2.9000	1.4000	0.2000	Iris-setosa
4.9000	3.1000	1.5000	0.1000	Iris-setosa
5.4000	3.7000	1.5000	0.2000	Iris-setosa

Scientific Questions

- Can we extract some information from the data?
- What can we infer from them?
- Can we provide predictions on some of the quantities on newly seen data?
- Can we predict the **petal width** of a specific kind of *Iris setosa* by using the **petal length**?

Scientific Questions

- Can we extract some information from the data?
- What can we infer from them?
- Can we provide predictions on some of the quantities on newly seen data?
- Can we predict the **petal width** of a specific kind of *Iris setosa* by using the **petal length**?

Scientific Questions

- Can we extract some information from the data?
- What can we infer from them?
- Can we provide predictions on some of the quantities on newly seen data?
- Can we predict the **petal width** of a specific kind of *Iris setosa* by using the **petal length**?

Scientific Questions

- Can we extract some information from the data?
- What can we infer from them?
- Can we provide predictions on some of the quantities on newly seen data?
- Can we predict the **petal width** of a specific kind of *Iris setosa* by using the **petal length**?

Scientific Questions

- Can we extract some information from the data?
- What can we infer from them?
- Can we provide predictions on some of the quantities on newly seen data?
- Can we predict the **petal width** of a specific kind of *Iris setosa* by using the **petal length**?

Solution: Linear Regression

We need to specify:

- Hypothesis space: **linear models**

$$\hat{t} = y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_j = \mathbf{w}^\top \mathbf{x}$$

where $\mathbf{w} = (w_0, w_1, \ldots, w_{M-1})^\top$ and $\mathbf{x} = (1, x_1, \ldots, x_{M-1})^\top$

- Loss function: **residual sum of squares** over the N samples $\{(\mathbf{x}_n, t_n)\}_{n=1}^N$:

$$\mathrm{RSS}(\mathbf{w}) = \sum_{n=1}^{N} (y(\mathbf{x}_n, \mathbf{w}) - t_n)^2$$

- Optimization method: closed form, gradient descent, ...

Solution: Linear Regression

We need to specify:

- Hypothesis space: **linear models**

$$\hat{t} = y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_j = \mathbf{w}^\top \mathbf{x}$$

  where $\mathbf{w} = (w_0, w_1, \ldots, w_{M-1})^\top$ and $\mathbf{x} = (1, x_1, \ldots, x_{M-1})^\top$

- Loss function: **residual sum of squares** over the N samples $\{(\mathbf{x}_n, t_n)\}_{n=1}^{N}$:

$$\mathrm{RSS}(\mathbf{w}) = \sum_{n=1}^{N} (y(\mathbf{x}_n, \mathbf{w}) - t_n)^2$$

- Optimization method: closed form, gradient descent, ...

Solution: Linear Regression

We need to specify:

- Hypothesis space: **linear models**

$$\hat{t} = y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_j = \mathbf{w}^\top \mathbf{x}$$

  where $\mathbf{w} = (w_0, w_1, \dots, w_{M-1})^\top$ and $\mathbf{x} = (1, x_1, \dots, x_{M-1})^\top$

- Loss function: **residual sum of squares** over the N samples $\{(\mathbf{x}_n, t_n)\}_{n=1}^N$:

$$\mathrm{RSS}(\mathbf{w}) = \sum_{n=1}^{N} (y(\mathbf{x}_n, \mathbf{w}) - t_n)^2$$

- Optimization method: closed form, gradient descent, ...

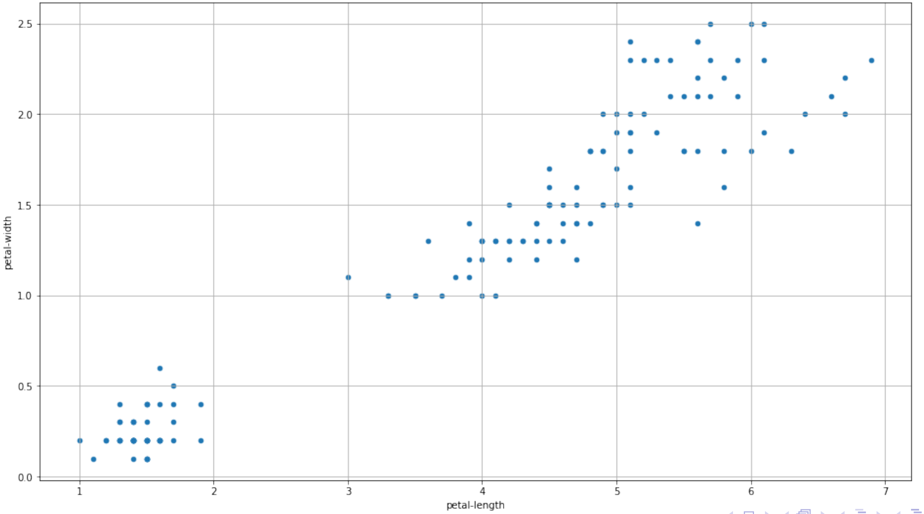
Example of Dataset

In our Iris dataset example: $\mathbf{x} = (1, \text{Petal length})^\top$ $(M = 2)$ and $t = \text{Petal width}$

rows, instances (N)

Constant	Petal length	Petal width
	variables, features, input, $\mathbf{x}$	target, response, output, t
1.0000	3.5000	0.2000
1.0000	1.4000	0.2000
1.0000	1.3000	0.2000
1.0000	1.5000	0.2000
1.0000	1.4000	0.2000
1.0000	1.7000	0.4000
1.0000	1.4000	0.3000
1.0000	1.5000	0.2000
1.0000	1.4000	0.2000
1.0000	1.5000	0.1000
1.0000	1.5000	0.2000

Example of Dataset



Outline

1. Administrivia
2. Regression Problem
3. Practical Example

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
  - shuffling (`shuffle()`)
  - remove inconsistent data
  - remove outliers
  - normalize or standardize data (`zscore()`)
  - fill missing data (`is_nan()`)

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
  - shuffling (`shuffle()`)
  - remove inconsistent data
  - remove outliers
  - normalize or standardize data (`zscore()`)
  - fill missing data (`is_nan()`)

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
    - shuffling (`shuffle()`)
    - remove inconsistent data
    - remove outliers
    - normalize or standardize data (`zscore()`)
    - fill missing data (`is_nan()`)

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
  - shuffling (`shuffle()`)
  - remove inconsistent data
  - remove outliers
  - normalize or standardize data (`zscore()`)
  - fill missing data (`is_nan()`)

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
    - shuffling (`shuffle()`)
    - remove inconsistent data
    - remove outliers
    - normalize or standardize data (`zscore()`)
    - fill missing data (`is_nan()`)

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
  - shuffling (`shuffle()`)
  - remove inconsistent data
  - remove outliers
  - normalize or standardize data (`zscore()`)
  - fill missing data (`is_nan()`)

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
  - shuffling (`shuffle()`)
  - remove inconsistent data
  - remove outliers
  - normalize or standardize data (`zscore()`)
  - fill missing data (`is_nan()`)

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
  - shuffling (`shuffle()`)
  - remove inconsistent data
  - remove outliers
  - normalize or standardize data (`zscore()`)
  - fill missing data (`is_nan()`)

Preliminary Operations

- Load data
- Inspect data
- Select the interesting data
- Preprocessing
  - shuffling (`shuffle()`)
  - remove inconsistent data
  - remove outliers
  - normalize or standardize data (`zscore()`)
  - fill missing data (`is_nan()`)

Data Normalization

Samples $\{s_1, \ldots, s_N\} \rightarrow$ normalization of sample s:

- z-score

$$\frac{s - \bar{s}}{S}$$

where $\bar{s} = \frac{1}{N}\sum_{n=1}^{N} s_n$ and $S^2 = \frac{1}{N-1}\sum_{n=1}^{N}(s_n - \bar{s})^2$

- Min-max feature scaling

$$\frac{s - s_{\min}}{s_{\max} - s_{\min}}$$

where $s_{\max} = \max_{n \in \{1,\ldots,N\}} s_n$ and $s_{\min} = \min_{n \in \{1,\ldots,N\}} s_n$

Data Normalization

Samples $\{s_1, \ldots, s_N\} \rightarrow$ normalization of sample s:

- z-score

$$\frac{s - \bar{s}}{S}$$

where $\bar{s} = \frac{1}{N}\sum_{n=1}^{N} s_n$ and $S^2 = \frac{1}{N-1}\sum_{n=1}^{N}(s_n - \bar{s})^2$

- Min-max feature scaling

$$\frac{s - s_{\min}}{s_{\max} - s_{\min}}$$

where $s_{\max} = \max_{n \in \{1,\ldots,N\}} s_n$ and $s_{\min} = \min_{n \in \{1,\ldots,N\}} s_n$

Data Normalization

Samples $\{s_1, \dots, s_N\} \rightarrow$ normalization of sample s:

- z-score
$$\frac{s - \bar{s}}{S}$$
where $\bar{s} = \frac{1}{N}\sum_{n=1}^{N} s_n$ and $S^2 = \frac{1}{N-1}\sum_{n=1}^{N}(s_n - \bar{s})^2$
- Min-max feature scaling
$$\frac{s - s_{\min}}{s_{\max} - s_{\min}}$$
where $s_{\max} = \max_{n \in \{1,\dots,N\}} s_n$ and $s_{\min} = \min_{n \in \{1,\dots,N\}} s_n$

Linear Regression in Python

We have several solutions from different libraries:

- `sklearn` with `LinearRegression()`
- `statsmodels` with `OLS`
- by hand implementation $\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{t}$

First the `sklearn` option:

- Create a linear model (`LinearRegression()`)
- Fit the model to the data (`fit()`)
- Analyze the results

Linear Regression in Python

We have several solutions from different libraries:

- `sklearn` with `LinearRegression()`
- `statsmodels` with `OLS`
- by hand implementation $\mathbf{w}^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{t}$

First the `sklearn` option:

- Create a linear model (`LinearRegression()`)
- Fit the model to the data (`fit()`)
- Analyze the results

Linear Regression in Python

We have several solutions from different libraries:

- `sklearn` with `LinearRegression()`
- `statsmodels` with `OLS`
- by hand implementation $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$

First the `sklearn` option:

- Create a linear model (`LinearRegression()`)
- Fit the model to the data (`fit()`)
- Analyze the results

Linear Regression in Python

We have several solutions from different libraries:

- `sklearn` with `LinearRegression()`
- `statsmodels` with `OLS`
- by hand implementation $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{t}$

First the `sklearn` option:

- Create a linear model (`LinearRegression()`)
- Fit the model to the data (`fit()`)
- Analyze the results

Linear Regression in Python

We have several solutions from different libraries:

- `sklearn` with `LinearRegression()`
- `statsmodels` with `OLS`
- by hand implementation $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$

First the `sklearn` option:

- Create a linear model (`LinearRegression()`)
- Fit the model to the data (`fit()`)
- Analyze the results

Linear Regression in Python

We have several solutions from different libraries:

- `sklearn` with `LinearRegression()`
- `statsmodels` with `OLS`
- by hand implementation $\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{t}$

First the `sklearn` option:

- Create a linear model (`LinearRegression()`)
- Fit the model to the data (`fit()`)
- Analyze the results

Example

Evaluating the Results

- Residual Sum of Squares (RSS), Sum Of Squared Errors (SSE):

$$\mathrm{RSS}(\mathbf{w}) = \sum_{n=1}^{N} (\hat{t}_n - t_n)^2 \qquad \text{where} \qquad \hat{t}_n = y(\mathbf{x}_n, \mathbf{w})$$

- Mean Square Error: $\mathrm{MSE} = \frac{\mathrm{RSS}(\mathbf{w})}{N}$
- Root Mean Square Error: $\mathrm{RMSE} = \sqrt{\frac{\mathrm{RSS}(\mathbf{w})}{N}}$

Evaluating the Results

- Residual Sum of Squares (RSS), Sum Of Squared Errors (SSE):

$$\text{RSS}(\mathbf{w}) = \sum_{n=1}^{N} (\hat{t}_n - t_n)^2 \qquad \text{where} \qquad \hat{t}_n = y(\mathbf{x}_n, \mathbf{w})$$

- Mean Square Error: $\text{MSE} = \dfrac{\text{RSS}(\mathbf{w})}{N}$
- Root Mean Square Error: $\text{RMSE} = \sqrt{\dfrac{\text{RSS}(\mathbf{w})}{N}}$

Evaluating the Results

- Residual Sum of Squares (RSS), Sum Of Squared Errors (SSE):

$$\mathrm{RSS}(\mathbf{w}) = \sum_{n=1}^{N} (\hat{t}_n - t_n)^2 \qquad \text{where} \qquad \hat{t}_n = y(\mathbf{x}_n, \mathbf{w})$$

- Mean Square Error: $\mathrm{MSE} = \dfrac{\mathrm{RSS}(\mathbf{w})}{N}$
- Root Mean Square Error: $\mathrm{RMSE} = \sqrt{\dfrac{\mathrm{RSS}(\mathbf{w})}{N}}$

Evaluating the Results

- Coefficient of determination (R squared):

$$R^2 = 1 - \frac{\text{RSS}(\mathbf{w})}{\text{TSS}}$$

where $\text{TSS} = \sum_{n=1}^{N} (\bar{t} - t_n)^2$ is the Total Sum of Squares and $\bar{t} = \frac{1}{N} \sum_{n=1}^{N} t_n$

- Degrees of Freedom: $\text{dfe} = N - M$
- Adjusted coefficient of determination $R^2_{\text{adj}} = 1 - (1 - R^2)\frac{N-1}{\text{dfe}}$

Evaluating the Results

- Coefficient of determination (R squared):

$$R^2 = 1 - \frac{\mathrm{RSS}(\mathbf{w})}{\mathrm{TSS}}$$

where $\mathrm{TSS} = \sum_{n=1}^{N} (\bar{t} - t_n)^2$ is the Total Sum of Squares and $\bar{t} = \frac{1}{N} \sum_{n=1}^{N} t_n$

- Degrees of Freedom: $\mathrm{dfe} = N - M$
- Adjusted coefficient of determination $R^2_{\mathrm{adj}} = 1 - (1 - R^2)\frac{N-1}{\mathrm{dfe}}$

Evaluating the Results

- Coefficient of determination (R squared):

$$R^2 = 1 - \frac{\mathrm{RSS}(\mathbf{w})}{\mathrm{TSS}}$$

where $\mathrm{TSS} = \sum_{n=1}^{N} (\bar{t} - t_n)^2$ is the Total Sum of Squares and $\bar{t} = \frac{1}{N} \sum_{n=1}^{N} t_n$

- Degrees of Freedom: $\mathrm{dfe} = N - M$
- Adjusted coefficient of determination $R^2_{\mathrm{adj}} = 1 - (1 - R^2)\dfrac{N-1}{\mathrm{dfe}}$

Statistical Tests on Coefficients

- **Assumption**: t_n satisfy $t_n = \mathbf{w}^T \mathbf{x}_n + \epsilon_n$, where ϵ_n is i.i.d. Gaussian white zero-mean noise and variance σ^2
- Exact distribution of the statistic:

$$\frac{\hat{w}_j - w_j}{\hat{\sigma}\sqrt{v_j}} \sim t_{N-M},$$

where w_j is the true parameter, $\hat{w}_j$ the estimated parameter with N samples, v_j is the j-th diagonal element of the matrix $(\mathbf{X}^\top \mathbf{X})^{-1}$, t_{N-M} is the T-student distribution with $\mathrm{dfe} = N - M$ degrees of freedom, and $\hat{\sigma}^2$ is is the unbiased estimated for the target variance:

$$\hat{\sigma}^2 = \frac{\mathrm{RSS}(\hat{\mathbf{w}})}{N - M}.$$

Statistical Tests on Coefficients

- **Assumption**: t_n satisfy $t_n = \mathbf{w}^T \mathbf{x}_n + \epsilon_n$, where ϵ_n is i.i.d. Gaussian white zero-mean noise and variance σ^2
- Exact distribution of the statistic:

$$\frac{\hat{w}_j - w_j}{\hat{\sigma}\sqrt{v_j}} \sim t_{N-M},$$

where w_j is the true parameter, $\hat{w}_j$ the estimated parameter with N samples, v_j is the j-th diagonal element of the matrix $(\mathbf{X}^\top \mathbf{X})^{-1}$, t_{N-M} is the T-student distribution with $\mathrm{dfe} = N - M$ degrees of freedom, and $\hat{\sigma}^2$ is is the unbiased estimated for the target variance:

$$\hat{\sigma}^2 = \frac{\mathrm{RSS}(\hat{\mathbf{w}})}{N - M}.$$

Statistical Tests on Coefficients

- **Test on single coefficients** $j \in \{0, \dots, M-1\}$:

$$H_0 : w_j = 0 \qquad \text{vs.} \qquad H_1 : w_j \neq 0$$

$$t_{\text{stat}} = \frac{\hat{w}_j - w_j}{\hat{\sigma}\sqrt{v_j}} \sim t_{N-M}$$

where t_{N-M} is the T-Student distribution with $N - M$ degrees of freedom

Statistical Tests on Coefficients

- **Test on the overall significance of the model**:

$$H_0 : w_1 = \dots = w_{M-1} = 0 \qquad \text{vs.} \qquad H_1 : \exists j \in \{1, \dots, M-1\} \text{ s.t. } w_j \neq 0$$

$$F_{\text{stat}} = \frac{N-M}{M-1} \cdot \frac{\text{TSS} - \text{RSS}(\hat{\mathbf{w}})}{\text{RSS}(\hat{\mathbf{w}})} \sim F_{M-1,N-M}$$

where $F_{M-1,N-M}$ is the Fisher-Snedecor distribution with parameters $M-1$ and $N-M$

  - We are comparing the full linear model $y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_j$ (with $N-M$ d.o.f.) against the constant model $y(\mathbf{x}, w_0) = w_0$ (with $N-1$ d.o.f.).

Statistical Tests on Coefficients

- **Test on the overall significance of the model**:

$$H_0 : w_1 = \cdots = w_{M-1} = 0 \qquad \text{vs.} \qquad H_1 : \exists j \in \{1, \dots, M-1\} \text{ s.t. } w_j \neq 0$$

$$F_{\text{stat}} = \frac{N-M}{M-1} \cdot \frac{\text{TSS} - \text{RSS}(\hat{\mathbf{w}})}{\text{RSS}(\hat{\mathbf{w}})} \sim F_{M-1,N-M}$$

where $F_{M-1,N-M}$ is the Fisher-Snedecor distribution with parameters $M-1$ and $N-M$

  - We are comparing the full linear model $y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_j$ (with $N-M$ d.o.f.) against the constant model $y(\mathbf{x}, w_0) = w_0$ (with $N-1$ d.o.f.).

Example of `OLS(x, t).fit()` Output

```
Dep. Variable:                  y   R-squared (uncentered):          0.927
Model:                        OLS   Adj. R-squared (uncentered):     0.926
Method:             Least Squares   F-statistic:                     1889.
Date:            Wed, 10 Mar 2021   Prob (F-statistic):           1.56e-86
Time:                    12:28:24   Log-Likelihood:                -16.645
No. Observations:             150   AIC:                             35.29
Df Residuals:                 149   BIC:                             38.30
Df Model:                       1
Covariance Type:        nonrobust
==========================================================================
        coef      std err         t       P>|t|       [0.025      0.975]
--------------------------------------------------------------------------
x1     0.9628      0.022     43.467      0.000        0.919       1.007
==========================================================================
Omnibus:                    2.326   Durbin-Watson:                   1.437
Prob(Omnibus):              0.313   Jarque-Bera (JB):                1.852
Skew:                       0.210   Prob(JB):                        0.396
Kurtosis:                   3.347   Cond. No.                         1.00
```

Different Implementations

`fit()`

- Slow in the execution
- Many checks for consistency
- Recap

By-hand solution

- Very fast
- No checks for consistency
- No tests on the coefficients

Code Exercise

```
x = zscore(dataset['petal-length'].values).reshape(-1, 1)
y = zscore(dataset['petal-width'].values)
Phi = np.ones((len(x), 2))
Phi[:, 1] = x.flatten()
model = inv(Phi.T @ Phi) @ (Phi.T.dot(y))
r_model = linear_model.Ridge(alpha=10)
r_model.fit(x, y)
l_model = linear_model.Lasso(alpha=10)
l_model.fit(x, y)
```

1. Which problem is the snippet above solving?
2. Which algorithms are implemented by the snippet above?
3. Which model would you choose to provide an interpretable solution, e.g., a model whose parameters influence are easy to explain?
4. Which are the pros and the cons of using the algorithm in lines 3-5?