INFO-F-422: Statistical foundations of machine learning

Linear regression

Gianluca Bontempi

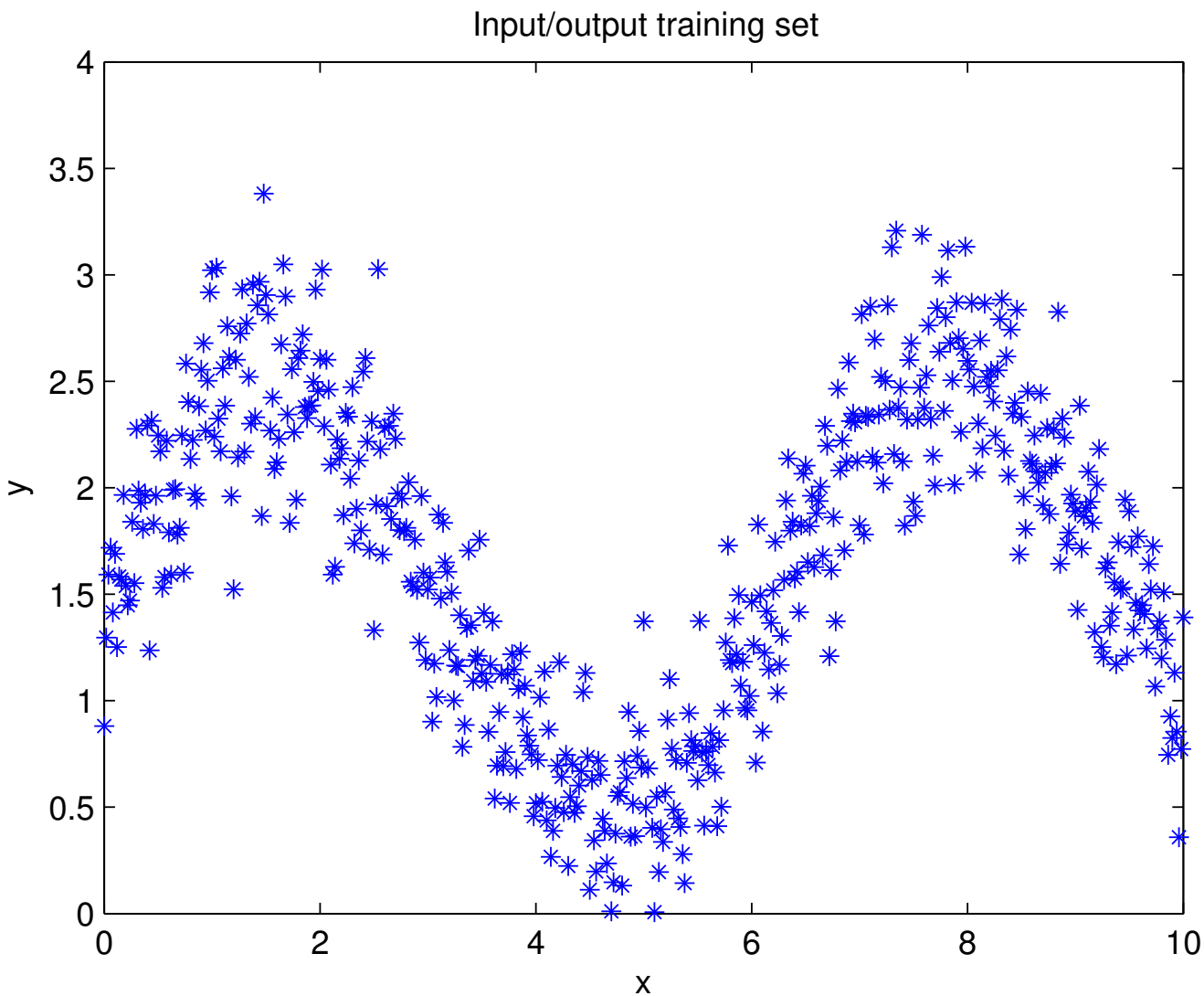
Machine Learning Group
Computer Science Department
mlg.ulb.ac.be

Beyond parameter estimation

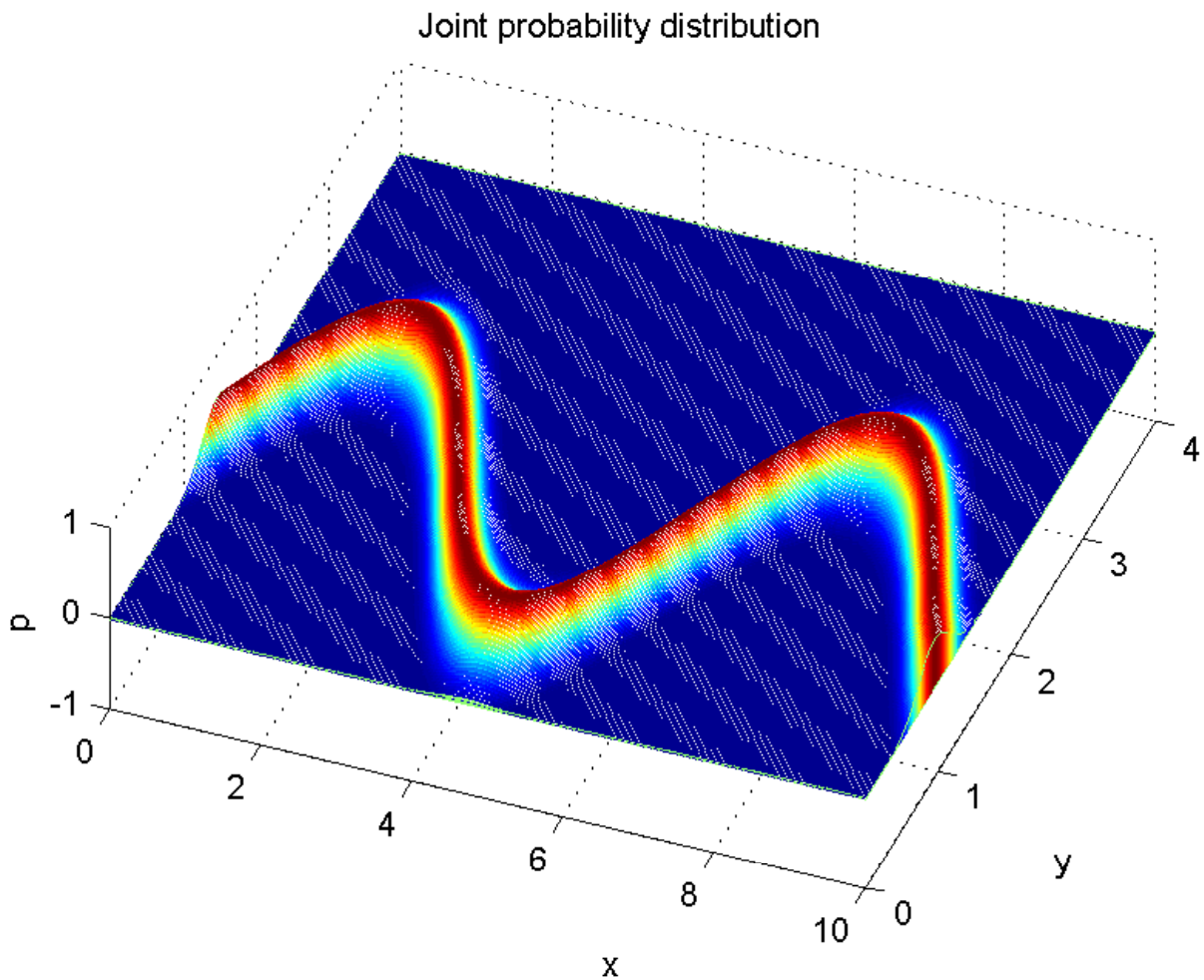
So far simple task: estimation of parameters of univariate distributions.
More complex estimation tasks:

- **Parameters of multivariate distributions:** consider for example multivariate gaussians.
- **More complex functionals.**
- **Discrete conditional distributions:** pattern recognition or pattern classification.
- **Continuous conditional distributions**: regression.

Bivariate continuous random scatterplot



Bivariate density distribution



Prediction problems

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack, on the basis of demographic, diet and clinical measurements.
- Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data.
- Identify the risk factors for breast cancer, based on clinical, demographic and genetic variables.
- Classify the category of a text email (spam or not) on the basis of its text content.
- Characterize the mechanical property of a steel plate on the basis of its physical and chemical composition.

Input/output problems

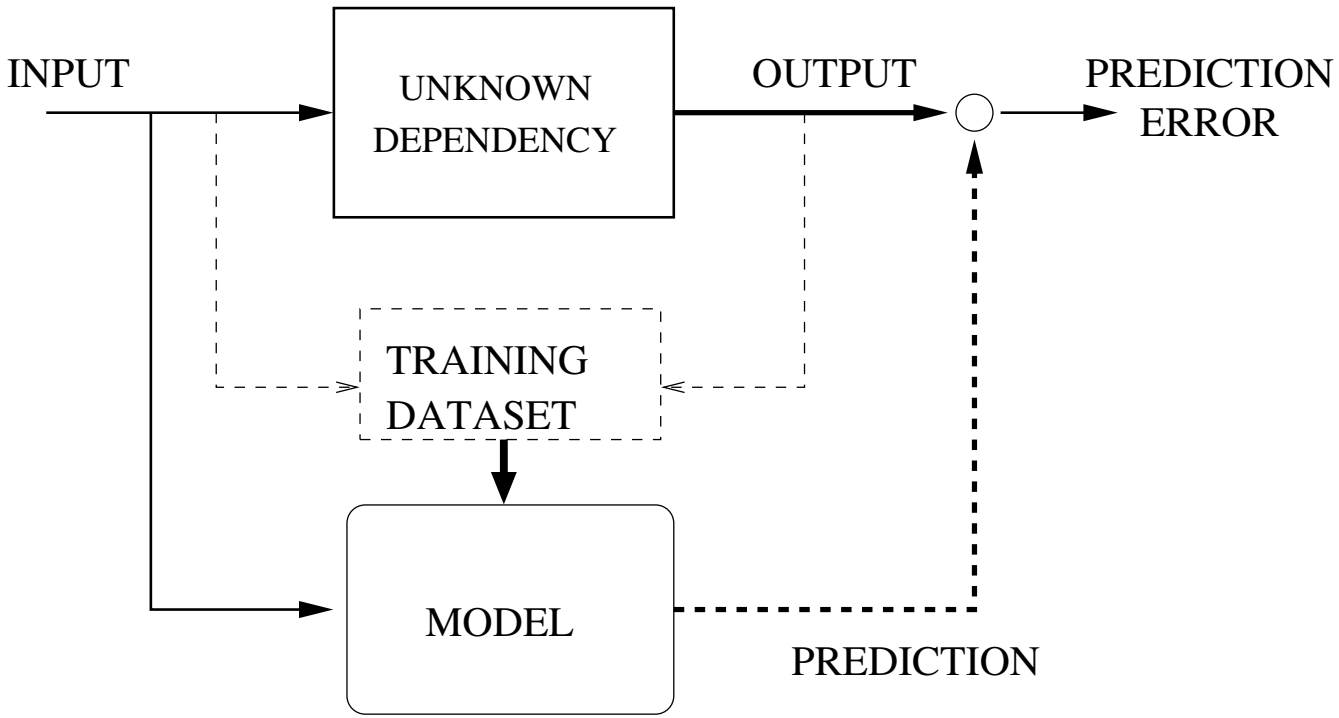
All the previous examples are characterized by

1. An outcome measurement, also called **output**, usually quantitative (like a stock price) or categorical (like heart attack/no heart attack).
2. a set of **features** or **inputs**, also quantitative or categorical, that we wish to use to predict the output.

Assumption: input variables provide some explanation for the variability of the output.

We collected a set of input/output data (**training set**), we use statistical methods to build a **prediction model** (**learner**) to predict the outcome for **new unseen objects**.

Supervised learning



Supervised learning because of the presence of the outcome variable which guides the learning process.
Collecting a set of training data is like having a teacher suggesting the correct answer for each input.

Regression and classification

According to the type of output, two prediction tasks:

- **Regression**: *quantitative* outputs, e.g. real or integer numbers
- **Classification (or pattern recognition)**: *qualitative* or *categorical* outputs which take values in a finite set of classes (e.g. black, white and red) where there is no explicit ordering. Qualitative variables are also referred to as **factors**.

Simple linear model

The simplest regression model is is the linear model

$$\mathbf{y} = \beta_0 + \beta_1 x + \mathbf{w}$$

where

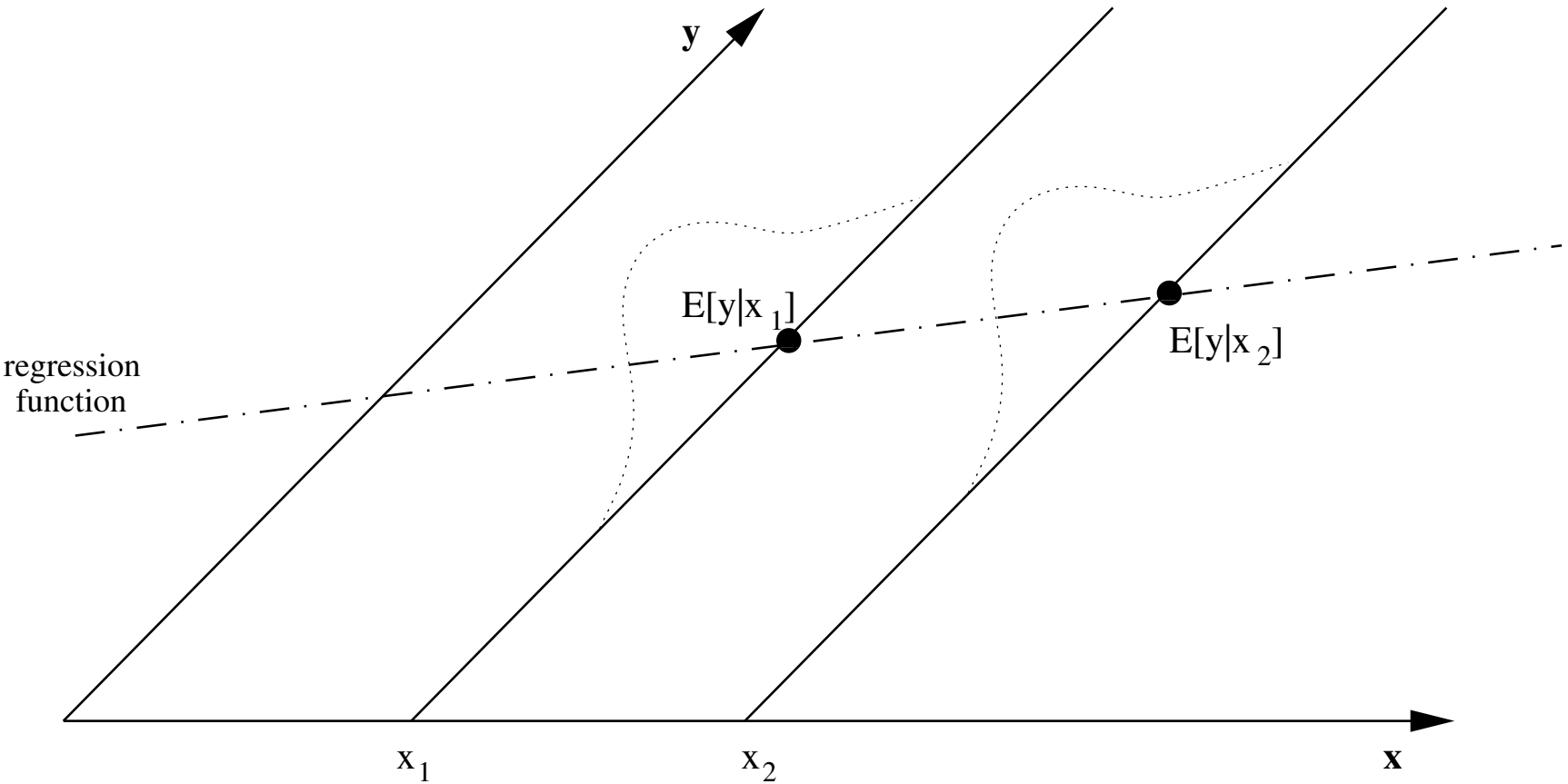
- $x \in \mathbb{R}$ is the regressor (or independent) variable,
- $\mathbf{y} \in \mathbb{R}$ is the measured response (or dependent) variable,
- β_0 is the intercept, β_1 is the slope
- $E[\mathbf{w}] = 0$ where $\mathbf{w}$ is the *model error*

This implies that

$$E[\mathbf{y}|x] = f(x) = \beta_0 + \beta_1 x$$
$$\mathrm{Var}\,[\mathbf{y}|x] = \mathrm{Var}\,[\mathbf{w}]$$

Linear regression function

The function $f(x) = E[\mathbf{y}|x]$ is also known as *regression function*.



What does "linear" mean?

Linear model is each input/output relationship which is *linear in the model parameters* and not necessarily in the dependent variables.

This means that

1. any value of the response variable y is described by a linear combination of a series of parameters (regression slopes, intercept)
2. no parameter appears as an exponent or is multiplied or divided by another parameter.

Example of linear models

According to our definition of linear model, then

- $y = \beta_0 + \beta_1 x$ is a linear model
- $y = \beta_0 + \beta_1 x^2$ is again a linear model. Simply by making the transformation $X = x^2$, the model can be put in in the linear form $y = \beta_0 + \beta_1 X$
- $y = \sum_{m=1}^{M} \beta_m \phi_m(x)$ is again a linear model and $\phi_m(\cdot)$ are called *basis functions*
- $y = B_0 x^{\beta_1}$ can be studied as a linear model between $Y = \log(y)$, $X = \log(x)$ and $\beta_0 = \log(B_0)$ thanks to the equality

$$\log(y) = \log(B_0) + \beta_1 \log(x) \Leftrightarrow Y = \beta_0 + \beta_1 X$$

- the relationship $y = \beta_0 + \beta_1 \beta_2^x$ cannot be linearized.

Model estimation

- N i.id. pairs of observations $D_N = \{\langle x_i, y_i \rangle\}$, $i = 1, \ldots, N$
- Data generated by the stochastic process

$$y_i = \beta_0 + \beta_1 x_i + w_i, \qquad i = 1, \ldots, N$$

where

1. $\mathbf{w}_i$ are iid realizations of the r.v. $\mathbf{w}$ having mean zero and constant variance $\sigma^2_{\mathbf{w}}$ (homoscedasticity),
2. x_i are non random and observed with negligible error

- Then the unknown parameters (also known as *regression coefficients*) β_0 and β_1 can be estimated by the *least-squares method*.

Least squares formulation

The method of least squares is designed to provide

1. estimations $\hat{\beta}_0$ and $\hat{\beta}_1$ of β_0 and β_1
2. the fitted values of the response y

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i, \quad i = 1, \ldots, N$$

so that the **residual sum of squares**

$$\text{SSE}_{\text{emp}}(b_0, b_1) = \sum_{i=1}^{N} (y_i - b_0 - b_1 x_i)^2$$

is minimized.

See Shiny dashboard `leastsquares.R`.

Ordinary least-squares solution

- Error function $\mathrm{SSE}_{\mathrm{emp}}(b_0, b_1)$ is a quadratic function of the coefficients b_0 and b_1
- Minimization of the error function has a **unique** solution which can be found in **closed form**.
- This is called the *ordinary least-squares solution*

$$\{\hat{\beta}_0, \hat{\beta}_1\} = \arg \min_{\{b_0, b_1\}} \mathrm{SSE}_{\mathrm{emp}}(b_0, b_1) =$$

$$= \arg \min_{\{b_0, b_1\}} \sum_{i=1}^{N} (y_i - b_0 - b_1 x_i)^2$$

Empirical risk

From $\mathrm{SSE}_{\mathrm{emp}}$ we can define the term

$$\widehat{\mathrm{MISE}}_{\mathrm{emp}} = \min_{\{b_0, b_1\}} \frac{\mathrm{SSE}_{\mathrm{emp}}(b_0, b_1)}{N} =$$

$$= \frac{\mathrm{SSE}_{\mathrm{emp}}(\hat{\beta}_0, \hat{\beta}_1)}{N} = \frac{\sum_{i=1}^{N} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2}{N}$$

is called the **empirical risk** or **training error**.

Note that the term $\mathrm{SSE}_{\mathrm{emp}}$ is a function of the training set and as such it can be considered as a realization of a random variable.

Multiple linear dependency

- Linear relationship between an independent variable $x \in \mathcal{X} \subset \mathbb{R}^n$ and a dependent random variable $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}$

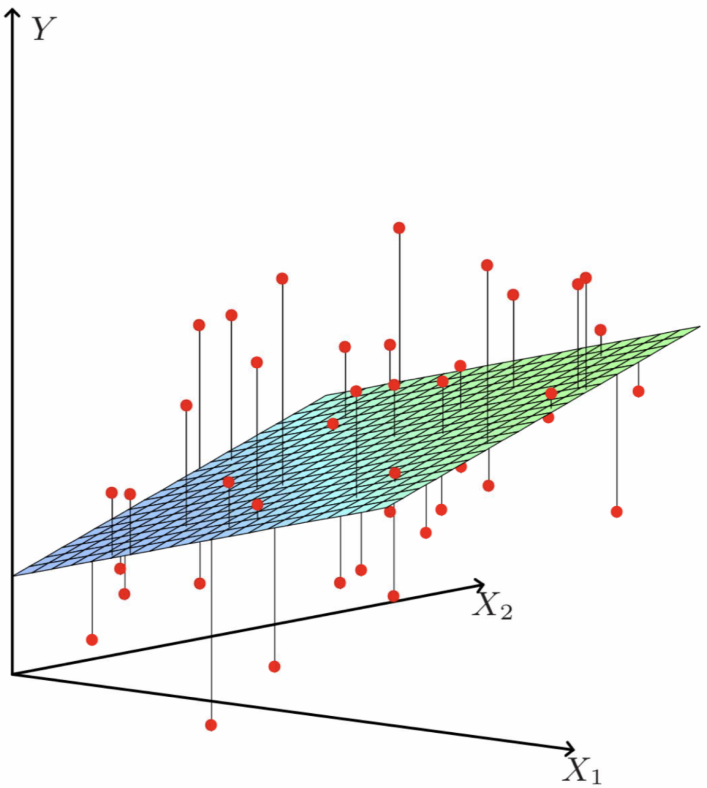
$$\mathbf{y} = \beta_0 + \beta_1 x_{\cdot 1} + \beta_2 x_{\cdot 2} + \cdots + \beta_n x_{\cdot n} + \mathbf{w}$$

  where $\mathbf{w}$ represents a random variable with mean zero and constant variance $\sigma^2_{\mathbf{w}}$.
- In matrix notation as:

$$\mathbf{y} = x^T \beta + \mathbf{w}$$

  where x stands for the $[p \times 1]$ vector $x = [1, x_{\cdot 1}, x_{\cdot 2}, \ldots, x_{\cdot n}]^T$, $\beta = [\beta_0, \ldots, \beta_n]^T$ is the vector of parameters and $p = n + 1$ is the total number of model parameters.
- NB: in the following $x_{\cdot j}$ (and $\mathbf{x}_j$) will denote the jth $(j = 1, \ldots, n)$ variable of the vector x, while x_i $(i = 1, \ldots, N)$ will denote the ith observation of the vector x.

Multiple linear regression with $n = 2$



(excerpt from "The Elements of Statistical Learning " book)

The multiple linear regression model

Consider N observations $D_N = \{\langle x_i, y_i \rangle : i = 1, \ldots, N\}$, where $x_i = (1, x_{i1}, \ldots, x_{in})$ and y_i are generated according to the linear model. We can write

$$Y = X\beta + W$$

where Y is the $[N \times 1]$ response vector, X is the $[N \times p]$ *data matrix*, whose $(j+1)^{\text{th}}$ column of X contains readings on the j^{th} regressor, β is the $[p \times 1]$ vector of parameters

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_N \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1n} \\ 1 & x_{21} & x_{22} & \ldots & x_{2n} \\ . & . & . & & . \\ . & . & . & & . \\ . & . & . & & . \\ 1 & x_{N1} & x_{N2} & \ldots & x_{Nn} \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ . \\ . \\ . \\ x_N^T \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ . \\ . \\ . \\ \beta_n \end{bmatrix} \quad W = \begin{bmatrix} w_1 \\ w_2 \\ . \\ . \\ . \\ w_N \end{bmatrix}$$

where w_i are assumed uncorrelated, with mean zero and constant variance $\sigma^2_{\mathbf{w}}$ (homogeneous variance). Then
$\text{Var}[\mathbf{w}_1, \ldots, \mathbf{w}_N] = \sigma^2_{\mathbf{w}} I_N$.

The least-squares solution

The **least-squares estimator** $\hat{\beta}$ is

$$\hat{\beta} = \arg\min_b \sum_{i=1}^{N} (y_i - x_i^T b)^2 = \arg\min_b \Big((Y - Xb)^T (Y - Xb) \Big)$$

The vector $\hat{\beta}$ must satisfy

$$\frac{\partial}{\partial b} [(Y - Xb)^T (Y - Xb)] \Big|_{b=\hat{\beta}} = 0 \Leftrightarrow -2X^T (Y - X\hat{\beta}) = 0$$

Normal equations

We obtain the *least-squares normal equations*

$$(X^T X)\hat{\beta} = X^T Y$$

As a result, assuming X is of full column rank

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

where the $X^T X$ matrix is a positive definite symmetric $[p \times p]$ matrix.

In Python notation:

```
import numpy as np
betahat = np.linalg.inv(X.T @ X) @ (X.T @ Y)
```

Python function OLS

```
import numpy as np
import statsmodels.api as sm
N = 50
n = 2
# Generate a N x n array of random normals
X = np.random.normal(loc=0.0, scale=1.0, size=(N, n))
# Generate y=x_1+w
Y = X[:, 0] + np.random.normal(loc=0.0, scale=0.1, size=N)
# Add an intercept term to X for the regression
X_design = sm.add_constant(X)

# Perform linear regression using Ordinary Least Squares (OLS)
model = sm.OLS(Y, X_design)
results = model.fit()
# Print the summary of the regression model
print(results.summary())
```

OLS Regression Results

Dep. Variable:	y	R-squared:	0.988
Model:	OLS	Adj. R-squared:	0.987
Method:	Least Squares	F-statistic:	1860.
Date:	Fri, 21 Feb 2025	Prob (F-statistic):	1.81e-45
Time:	13:11:47	Log-Likelihood:	40.209
No. Observations:	50	AIC:	-74.42
Df Residuals:	47	BIC:	-68.68
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0001	0.016	0.008	0.994	-0.032	0.032
x1	0.9952	0.016	60.891	0.000	0.962	1.028
x2	7.201e-05	0.016	0.004	0.997	-0.033	0.033

Empirical risk

Given $\hat{\beta}$ we obtain

$$\text{SSE}_{\text{emp}} = \left((Y - X\hat{\beta})^T (Y - X\hat{\beta}) \right) = e^T e$$

where SSE_{emp} represents the **residual sum of squares** for linear models and e is the $[N \times 1]$ vector of residuals.

The **empirical risk or training error** is

$$\widehat{\text{MISE}}_{\text{emp}} = \frac{\sum_{i=1}^{N} (y_i - x_i^T \hat{\beta})^2}{N} = \frac{\text{SSE}_{\text{emp}}}{N}$$

Analysis of the LS estimate

If the linear dependency assumption holds

- If $E[\mathbf{w}] = 0$ then $\hat{\boldsymbol{\beta}}$ is an unbiased estimator of β.
- The *residual mean square* estimator

$$\hat{\boldsymbol{\sigma}}^2 = \frac{(Y - X\hat{\boldsymbol{\beta}})^T (Y - X\hat{\boldsymbol{\beta}})}{N - p}$$

  is an unbiased estimator of the error variance $\sigma^2_{\mathbf{w}}$.
- If the $\mathbf{w}_i$ are uncorrelated and have common variance, the variance-covariance matrix of $\hat{\boldsymbol{\beta}}$ is given by

$$\mathrm{Var}[\hat{\boldsymbol{\beta}}] = \sigma^2_{\mathbf{w}} (X^T X)^{-1}$$

- Python script `Linear/bv_mult.py`.

Variance of the prediction

- The prediction $\widehat{\mathbf{y}}$ for a generic input value $x = x_0$ is unbiased

$$E[\widehat{\mathbf{y}}|x_0] = x_0^T \beta$$

- The variance of the prediction $\widehat{\mathbf{y}}$ for a generic input value $x = x_0$ is given by

$$\text{Var}[\widehat{\mathbf{y}}|x_0] = \sigma^2_{\mathbf{w}} x_0^T (X^T X)^{-1} x_0$$

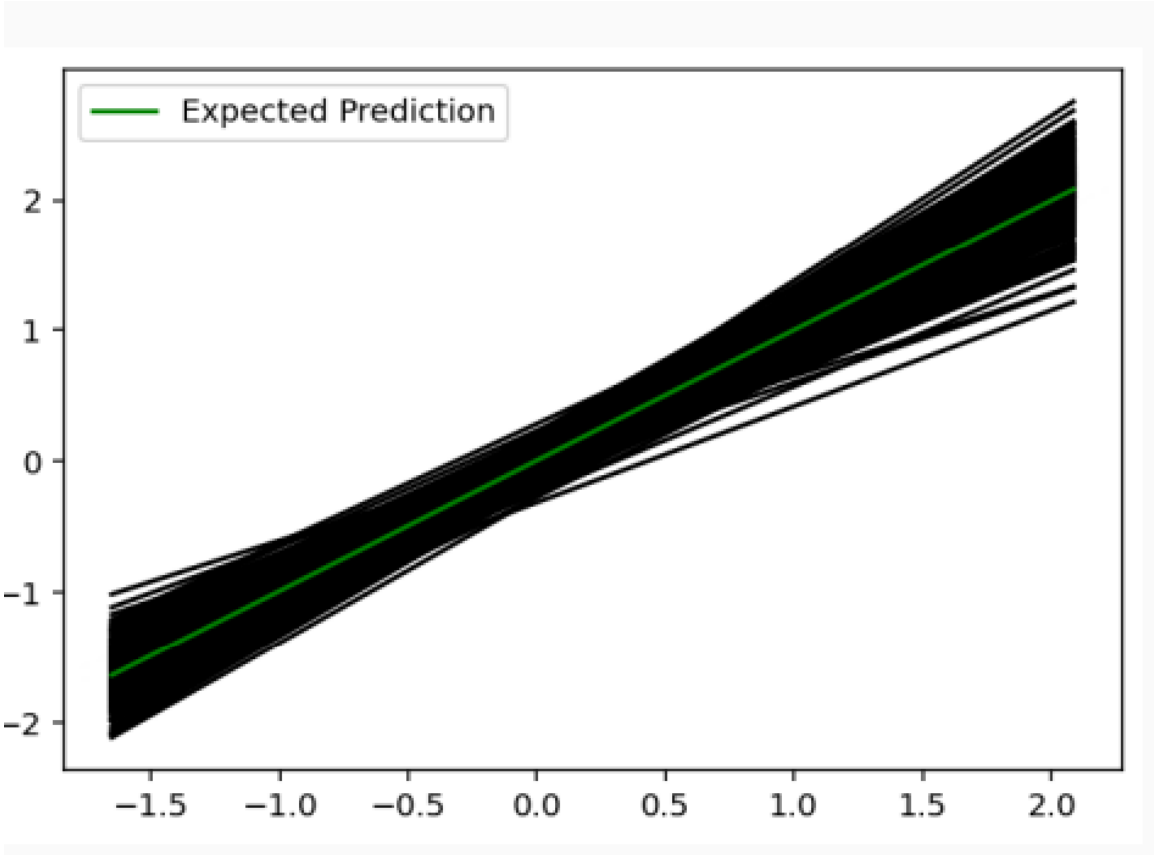
- Assuming a **normal** residual $\mathbf{w}$, the $100(1-\alpha)\%$ confidence bound for the regression value $E[\widehat{\mathbf{y}}|x = x_0]$ is given by

$$\hat{y}(x_0) \pm t_{\alpha/2, N-p} \hat{\sigma}_{\mathbf{w}} \sqrt{x_0^T (X^T X)^{-1} x_0}$$

where $t_{\alpha/2,N-p}$ is the upper $\alpha/2$ percent point of the t-distribution with $N - p$ degrees of freedom and the quantity $\hat{\sigma}_{\mathbf{w}} \sqrt{x_0^T (X^T X)^{-1} x_0}$ is the *standard error of prediction* for multiple regression.

Sampling distribution of the prediction

$\beta_0 = 0, \beta_1 = 1, N = 30, \sigma_{\mathbf{w}} = 0.5$



Code in the script `Linear/bvis.py`.

```
R = 500 ## number of MC trials
N = 30 ## number of observations
beta0 = 0   ## intercept of the linear data generating process
beta1 = 1   ## slope of the linear data generating process
X = np.sort(np.random.normal(0, 1, N))
Yhatr = None
for r in range(R):
    ## dataset generation of size N
    Y = beta0 + beta1 * X + np.random.normal(0, 0.5, N)
    XX = np.column_stack((np.ones(N), X))
    ## Least squares
    betahat = np.linalg.solve(XX.T @ XX, XX.T @ Y)
    ## Prediction for the given dataset
    prediction = betahat[0] + betahat[1] * X
    if Yhatr is None:
        Yhatr = prediction.reshape(-1, 1)
        plt.figure()
    else:
        Yhatr = np.column_stack((prediction, Yhatr))
    plt.plot(X, prediction, color='black')
expected_prediction = np.mean(Yhatr, axis=1)
plt.plot(X, expected_prediction, 'g-', label='Expected Prediction')
```

Generalization error of the linear model

- The linear predictor
$$\hat{y} = x^T \hat{\beta}$$
has been estimated by using the training dataset $D_N = \{\langle x_i, y_i \rangle : i = 1, \ldots, N\}$. Then $\hat{\boldsymbol{\beta}}$ is a r.v.
- Now we want to use it to predict for a test input x the future output $\mathbf{y}(x)$.
- The test output $\mathbf{y}(x)$ is independent of the training set $\mathbf{D}_N$.
- Which precision can we expect from $\hat{y}(x) = x^T \hat{\beta}$ on average?
- A measure of error is the MSE
$$\text{MSE}(x) = E_{\mathbf{D}_N, \mathbf{y}}[(\mathbf{y}(x) - x^T \hat{\boldsymbol{\beta}})^2] = \sigma^2_{\mathbf{w}} + E_{\mathbf{D}_N}[(x^T \beta - x^T \hat{\boldsymbol{\beta}})^2]$$
where $\mathbf{y}(x_i)$ is **independent** of $\mathbf{D}_N$ and the integrated version
$$\text{MISE} = \int_{\mathbf{X}} \text{MSE}(x) p(x) dx$$
- How can we estimate this quantity?

The expected empirical error

- Is the empirical risk a good estimate of the MISE generalization error?
- The expectation of the residual sum of squares can be written as[1]

$$E_{\mathbf{D}_N}\left[\widehat{\mathbf{MISE}}_{\text{emp}}\right] = E_{\mathbf{D}_N}\left[\frac{\sum_{i=1}^{N}(\mathbf{y}_i - x_i^T\hat{\beta})^2}{N}\right] =$$

$$= \frac{N-p}{N}\, E_{\mathbf{D}_N}\left[\frac{\sum_{i=1}^{N}(\mathbf{y}_i - x_i^T\hat{\beta})^2}{N-p}\right] = \frac{N-p}{N}\sigma^2_{\mathbf{w}}$$

- This is the expectation of the error made by a linear model trained on D_N to predict the value of the output in D_N.

[1] derivation in the handbook

MISE error

- Let us compute now the expected prediction error of a linear model trained on D_N when this is used to predict a set of test outputs distributed according to the same linear dependency but **independent** of the training set.
- It can be shown[2] that in case of linear dependency

$$\text{MISE} = E_{\mathbf{D}_N,\mathbf{y}}\left[\frac{\sum_{i=1}^{N}(\mathbf{y} - x_i^T\hat{\boldsymbol{\beta}})^2}{N}\right] = \frac{N+p}{N}\sigma_{\mathbf{w}}^2$$

Note that in the MISE formula the $\mathbf{y}$ distribution is independent of $\mathbf{D}_N$ and then of $\hat{\boldsymbol{\beta}}$.

[2] derivation in the handbook

MISE error

Then it follows that the empirical error returns a biased estimate of MISE, that is

$$E_{\mathbf{D}_N}[\widehat{\mathbf{MISE}}_{\text{emp}}] = \frac{N-p}{N}\sigma^2_{\mathbf{w}} \neq \text{MISE} = \frac{N+p}{N}\sigma^2_{\mathbf{w}}$$

If we replace $\widehat{\text{MISE}}_{\text{emp}}$ with

$$\widehat{\text{MISE}}_{\text{emp}} + 2\frac{p}{N}\sigma^2_{\mathbf{w}}$$

we obtain an unbiased estimator of the quantity MISE (see Python script `Linear/ee.py`).
Nevertheless, this estimator requires an estimate of the noise variance.

The PSE and the FPE

- Given the estimate $\hat{\sigma}^2_{\mathbf{w}}$ we have the **Predicted Squared Error (PSE)** criterion

$$\text{PSE} = \widehat{\text{MISE}}_{\text{emp}} + 2\hat{\sigma}^2_{\mathbf{w}} p/N$$

- Taking as estimate of $\sigma^2_{\mathbf{w}}$

$$\hat{\sigma}^2_{\mathbf{w}} = \frac{1}{N-p}\sum_{i=1}^{N}(y_i - x_i^T\hat{\beta})^2$$

we have the **Final Prediction Error (FPE)**

$$\text{FPE} = \frac{1+p/N}{1-p/N}\widehat{\text{MISE}}_{\text{emp}}$$

- See the Python script `Linear/fpe.py`

Polynomial model selection with PSE

Model candidates:

- degree $n = 0$: $h(x) = \beta_0$
- degree $n = 1$: $h(x) = \beta_0 + \beta_1 x$
- degree $n = 2$: $h(x) = \beta_0 + \beta_1 x + \beta_2 x^2$
- ....
- degree n: $h(x) = \beta_0 + \beta_1 x + \cdots + \beta_n x^n$

$$\tilde{n} = \arg\min_n \left[\widehat{\mathrm{MISE}}_{\mathrm{emp}}(n) + 2\hat{\sigma}^2_{\mathbf{w}} \frac{n+1}{N} \right]$$