Linear Regression

Christos Dimitrakakis

October 14, 2025

Outline

Simple linear regression

Input and output

- Data pairs (x_t, y_t), $t = 1, \ldots, T$.
- Input $x_t \in \mathbb{R}$
- Output $y_t \in \mathbb{R}$.

Modelling the conditional expectation $\mathbb{E}[y_t | x_t]$

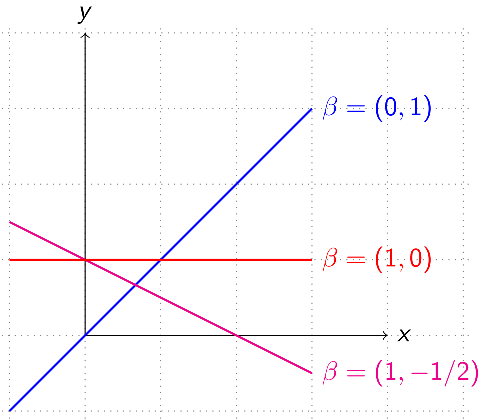
- Parameters $\beta_0, \beta_1 \in \mathbb{R}$
- Function $\pi_\beta : \mathbb{R} \to \mathbb{R}$, defined as

$$\pi_\beta(x_t) = \beta_0 + \beta_1 x_t$$

Probabilistic predictions: Modelling the conditional probability $\mathbb{P}[y_t | x_t]$

- $y_t = \mathbb{E}[y_t | x_t] + \epsilon_t$, with $\epsilon_t \in \mathbb{R}$ being zero-mean noise.
- Simplest model: variance $\sigma = \mathbb{V}(\epsilon) = \mathbb{E}[\epsilon_t^2]$

Linear models



$$\pi_\beta(x) = \beta_0 + \beta_1 x = [\beta_0, \beta_1] \begin{bmatrix} 1 \\ x \end{bmatrix}$$

Two views of the problem

Learning as optimisation

- Each value $\pi_\beta(x)$ is a prediction about the value of y

Learning as inference

Two views of the problem

Learning as optimisation

- Each value $\pi_\beta(x)$ is a prediction about the value of y
- We suffer a loss $\ell(y, \pi_\beta(x))$ for every example (x, y) that we see.

Learning as inference

Two views of the problem

Learning as optimisation

- Each value $\pi_\beta(x)$ is a prediction about the value of y
- We suffer a loss $\ell(y, \pi_\beta(x))$ for every example (x, y) that we see.
- We can minimise the average loss over the data

Learning as inference

Two views of the problem

Learning as optimisation

- Each value $\pi_\beta(x)$ is a prediction about the value of y
- We suffer a loss $\ell(y, \pi_\beta(x))$ for every example (x, y) that we see.
- We can minimise the average loss over the data
- Ideally, we want to minimise the expected loss, with respect to the unknown distribution P.

Learning as inference

Two views of the problem

Learning as optimisation

- Each value $\pi_\beta(x)$ is a prediction about the value of y
- We suffer a loss $\ell(y, \pi_\beta(x))$ for every example (x, y) that we see.
- We can minimise the average loss over the data
- Ideally, we want to minimise the expected loss, with respect to the unknown distribution P.

Learning as inference

- The parameters β define a probabilistic model $P_\beta(y|x)$ for every value of y.

Two views of the problem

Learning as optimisation

- Each value $\pi_\beta(x)$ is a prediction about the value of y
- We suffer a loss $\ell(y, \pi_\beta(x))$ for every example (x, y) that we see.
- We can minimise the average loss over the data
- Ideally, we want to minimise the expected loss, with respect to the unknown distribution P.

Learning as inference

- The parameters β define a probabilistic model $P_\beta(y|x)$ for every value of y.
- We want to find the parameters giving the highest probability on the observed data.

Two views of the problem

Learning as optimisation

- Each value $\pi_\beta(x)$ is a prediction about the value of y
- We suffer a loss $\ell(y, \pi_\beta(x))$ for every example (x, y) that we see.
- We can minimise the average loss over the data
- Ideally, we want to minimise the expected loss, with respect to the unknown distribution P.

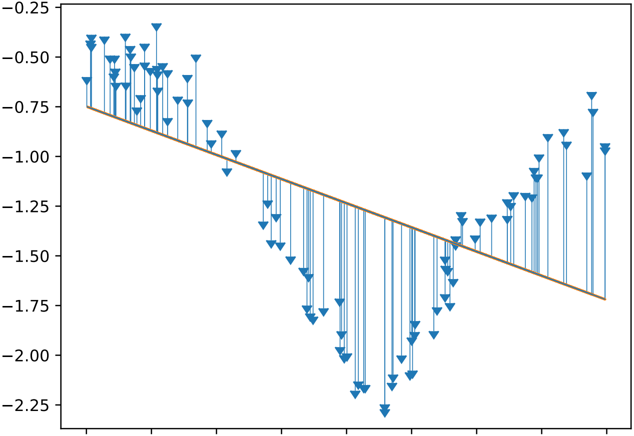
Learning as inference

- The parameters β define a probabilistic model $P_\beta(y|x)$ for every value of y.
- We want to find the parameters giving the highest probability on the observed data.
- Ideally, we want to find the true conditional distribution $P(y|x)$.

Learning as Optimisation

Find the parameters β minimising squared error

$$\min_{\beta} \frac{1}{T} \sum_{t=1}^{T} \Big[\underbrace{y_t - \pi_\beta(x_t)}_{\text{residual}} \Big]^2$$

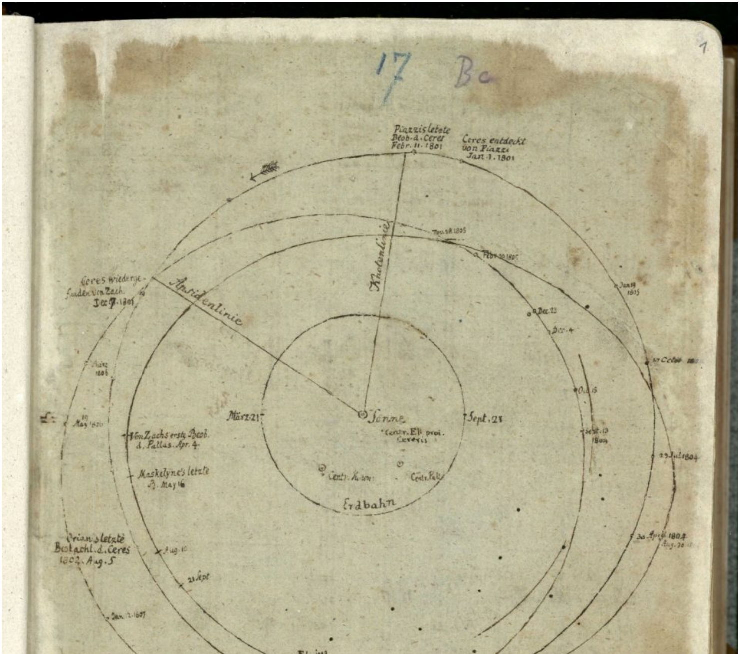


Origins

Figure: Gauss: originator

Figure: Legendre: first publication

The orbit of Ceres



Maximum likelihood inference

Gaussian noise model:

$$y_t = f(x_t) + \epsilon_t, \qquad \epsilon_t \sim \mathrm{Normal}(0, \sigma)$$

With conditional density

$$p_\beta(y_t | x_t) \propto \exp(-[y_t - \pi_\beta(x_t)]^2 / 2\sigma^2)$$

Maximum likelihood inference

Gaussian noise model:

$$y_t = f(x_t) + \epsilon_t, \qquad \epsilon_t \sim \mathrm{Normal}(0, \sigma)$$

With conditional density

$$p_\beta(y_t | x_t) \propto \exp(-[y_t - \pi_\beta(x_t)]^2 / 2\sigma^2)$$

Maximum likelihood inference

Idea: For data D, find parameters maximising $P_\beta(D)$

Maximum likelihood inference

Gaussian noise model:

$$y_t = f(x_t) + \epsilon_t, \qquad \epsilon_t \sim \mathrm{Normal}(0, \sigma)$$

With conditional density

$$p_\beta(y_t | x_t) \propto \exp(-[y_t - \pi_\beta(x_t)]^2 / 2\sigma^2)$$

Maximum likelihood inference

Idea: For data D, find parameters maximising $P_\beta(D)$

$$\begin{aligned}
\arg\max_\beta P_\beta(D) &= \arg\max_\beta p_\beta(y_1, \ldots, y_t | x_1, \ldots, x_T) = \arg\max_\beta \ln \prod_t p_\beta(y_t | x_t) \\
&= \arg\max_\beta \sum \ln p_\beta(y_t | x_t) \\
&= \arg\max_\beta \sum_t \ln \left\{ \exp\left(-[y_t - \pi_\beta(x_t)]^2 / 2\sigma^2 \right) \right\} \\
&= \arg\max_\beta \sum_t -[y_t - \pi_\beta(x_t)]^2 / 2\sigma^2 = \arg\min_\beta \sum_t |y_t - \pi_\beta(x_t)|^2
\end{aligned}$$

Coding break

- Show implementation
- Fit and residuals
- Multiple draws from the distribution
- Fit on non-linear data?

Multiple linear regression

Input and output

- Data pairs (x_t, y_t), $t = 1, \dots, T$.
- Input $x_t \in \mathbb{R}^n$
- Output $y_t \in \mathbb{R}^m$.

Multiple linear regression

Input and output

- Data pairs (x_t, y_t), $t = 1, \ldots, T$.
- Input $x_t \in \mathbb{R}^n$
- Output $y_t \in \mathbb{R}^m$.

Point predictions: Modelling the conditional expectation $\mathbb{E}[y_t | x_t]$

- Parameters $\beta \in \mathbb{R}^{n \times m}$
- Function $\pi_\beta : \mathbb{R}^n \to \mathbb{R}^m$, defined as

$$\pi_\beta(x_t) = \beta^\top x_t = \sum_{i=1}^{n} \beta_i x_{t,i}$$

Probabilistic predictions: Modelling the conditional probability $\mathbb{P}[y_t | x_t]$

- $y_t = \mathbb{E}[y_t | x_t] + \epsilon_t$, with $\epsilon_t \in \mathbb{R}^m$ being zero-mean noise.
- Noise covariance matrix $\Sigma = \mathbb{V}(\epsilon) = \mathbb{E}[\epsilon_t \mid \epsilon_t \top]$

Gradient descent algorithm

Minimising a function

$$\min_{\beta} f(\beta) \leq f(\beta')\forall\beta', \qquad \beta^* = \arg\min_{\beta} f(\beta) \Rightarrow f(\beta^*) = \min_{\beta} f(\beta)$$

Gradient descent algorithm

Minimising a function

$$\min_{\beta} f(\beta) \leq f(\beta') \forall \beta', \qquad \beta^* = \arg\min_{\beta} f(\beta) \Rightarrow f(\beta^*) = \min_{\beta} f(\beta)$$

Gradient descent for minimisation

- Input β_0
- For $n = 0, \ldots, N$:
- $\beta_{n+1} = \beta_n - \eta_n \nabla_{\beta} f(\beta_n)$

Gradient descent algorithm

Minimising a function

$$\min_{\beta} f(\beta) \leq f(\beta') \forall \beta', \qquad \beta^* = \arg\min_{\beta} f(\beta) \Rightarrow f(\beta^*) = \min_{\beta} f(\beta)$$

Gradient descent for minimisation

- Input β_0
- For $n = 0, \ldots, N$:
- $\beta_{n+1} = \beta_n - \eta_n \nabla_\beta f(\beta_n)$

Step-size η_n

- η_n fixed: for online learning
- $\eta_n = c/[c + n]$ for asymptotic convergence
- $\eta_n = \arg\min_\eta f(\beta_n + \eta \nabla_\beta)$: Line search.

Gradient descent for squared error

The cost function

$L(\beta, D) = \sum_{t=1}^{T} (y_t - \pi_\beta(x_t))^2 = \sum_{t=1}^{T} \epsilon_t^2$, with $\epsilon_t \triangleq y_t - \pi_\beta(x_t)$.

Gradient descent for squared error

The cost function

$L(\beta, D) = \sum_{t=1}^{T} (y_t - \pi_\beta(x_t))^2 = \sum_{t=1}^{T} \epsilon_t^2$, with $\epsilon_t \triangleq y_t - \pi_\beta(x_t)$.

Cost gradient

Using the chain rule of differentiation, $\nabla_\beta f(\epsilon) = \nabla_\epsilon f(\epsilon) \nabla_\beta \epsilon$.

$$
\begin{aligned}
\nabla_\beta L(\beta, D) &= \nabla_\beta \sum_{t=1}^{T} \epsilon_t^2 = \sum_{t=1}^{T} \nabla_\beta \epsilon_t^2 = \sum_{t=1}^{T} \nabla_{\epsilon_t} \epsilon_t^2 \nabla_\beta \epsilon \\
&= \sum_{t=1}^{T} 2\epsilon_t \nabla_\beta [y_t - \pi_\beta(x_t)] = \sum_{t=1}^{T} 2[y_t - \pi_\beta(x_t)][-\nabla_\beta \pi_\beta(x_t)]
\end{aligned}
$$

Gradient descent for squared error

The cost function

$L(\beta, D) = \sum_{t=1}^{T} (y_t - \pi_\beta(x_t))^2 = \sum_{t=1}^{T} \epsilon_t^2$, with $\epsilon_t \triangleq y_t - \pi_\beta(x_t)$.

Cost gradient

Using the chain rule of differentiation, $\nabla_\beta f(\epsilon) = \nabla_\epsilon f(\epsilon) \nabla_\beta \epsilon$.

$$\begin{aligned}
\nabla_\beta L(\beta, D) &= \nabla_\beta \sum_{t=1}^{T} \epsilon_t^2 = \sum_{t=1}^{T} \nabla_\beta \epsilon_t^2 = \sum_{t=1}^{T} \nabla_{\epsilon_t} \epsilon_t^2 \nabla_\beta \epsilon \\
&= \sum_{t=1}^{T} 2\epsilon_t \nabla_\beta [y_t - \pi_\beta(x_t)] = \sum_{t=1}^{T} 2[y_t - \pi_\beta(x_t)][-\nabla_\beta \pi_\beta(x_t)]
\end{aligned}$$

Parameter gradient for linear regression

Remember $\nabla_\beta f = (\partial / \partial_1 f, \ldots, \partial / \partial_n f)$

$$\frac{\partial}{\partial \beta_j} \pi_\beta(x_t) = \frac{\partial}{\partial \beta_j} \sum_{i=1}^{n} \beta_i x_{t,i} = \sum_{i=1}^{n} \frac{\partial}{\partial \beta_j} \beta_i x_{t,i} = x_{t,j}.$$

Stochastic gradient descent algorithm

When f is an expectation

$$f(\beta) = \int_X dP(x) g(x, \beta).$$

Replacing the expectation with a sample:

$$\begin{aligned} \nabla f(\beta) &= \int_X dP(x) \nabla g(x, \beta) \\ &\approx \frac{1}{K} \sum_{k=1}^{K} \nabla g(x^{(k)}, \beta), \qquad\qquad x^{(k)} \sim P. \end{aligned}$$

Analytical Least-Squares Solution

We need to solve the following equations for A:

$$
\begin{aligned}
y_1 &= x_1^\top \beta \\
\cdots & \quad \cdots \\
y_t &= x_t^\top \beta \\
\cdots & \quad \cdots \\
y_T &= x_T^\top \beta
\end{aligned}
$$

Analytical Least-Squares Solution

We need to solve the following equations for A:

$$\begin{aligned}
y_1 &= x_1^\top \beta \\
\cdots & \quad \cdots \\
y_t &= x_t^\top \beta \\
\cdots & \quad \cdots \\
y_T &= x_T^\top \beta
\end{aligned}$$

We can rewrite it in matrix form:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_t \\ \vdots \\ y_T \end{pmatrix} = \begin{pmatrix} x_1^\top \\ \vdots \\ x_t^\top \\ \vdots \\ x_T^\top \end{pmatrix} \beta$$

Analytical Least-Squares Solution

We need to solve the following equations for A:

$$
\begin{aligned}
y_1 &= x_1^\top \beta \\
\cdots & \quad \cdots \\
y_t &= x_t^\top \beta \\
\cdots & \quad \cdots \\
y_T &= x_T^\top \beta
\end{aligned}
$$

We can rewrite it in matrix form:

$$
\begin{pmatrix} y_1 \\ \vdots \\ y_t \\ \vdots \\ y_T \end{pmatrix} = \begin{pmatrix} x_1^\top \\ \vdots \\ x_t^\top \\ \vdots \\ x_T^\top \end{pmatrix} \beta
$$

Resulting in

$$
\boldsymbol{y} = X\beta.
$$

Analytical Least-Squares Solution

We need to solve the following equations for A:

$$\begin{aligned} y_1 &= x_1^\top \beta \\ \cdots & \quad \cdots \\ y_t &= x_t^\top \beta \\ \cdots & \quad \cdots \\ y_T &= x_T^\top \beta \end{aligned}$$

We can rewrite it in matrix form:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_t \\ \vdots \\ y_T \end{pmatrix} = \begin{pmatrix} x_1^\top \\ \vdots \\ x_t^\top \\ \vdots \\ x_T^\top \end{pmatrix} \beta$$

Resulting in

$$\boldsymbol{y} = X\beta.$$

How can we get β?

Finding the β

We now have a linear equation,

$$\boldsymbol{y} = X\beta.$$

We want to solve for β. If X had an inverse X^{-1}, we could obtain

$$X^{-1}\boldsymbol{y} = X^{-1}X\beta = I\beta = \beta.$$

But X^{-1} does not exist.

Finding the β

We now have a linear equation,

$$\boldsymbol{y} = X\beta.$$

We want to solve for β. If X had an inverse X^{-1}, we could obtain

$$X^{-1}\boldsymbol{y} = X^{-1}X\beta = I\beta = \beta.$$

But X^{-1} does not exist.

Least-squares solution

The left-pseudo inverse $\tilde{X}^{-1} \triangleq (X^\top X)^{-1}X^\top$ can be used to obtain

$$\beta = \tilde{X}^{-1}\boldsymbol{y},$$

This follows as:

$$\begin{aligned}
\boldsymbol{y} &= X\beta \\
\tilde{X}^{-1}\boldsymbol{y} &= \tilde{X}^{-1}X\beta \\
\tilde{X}^{-1}\boldsymbol{y} &= \underbrace{(X^\top X)^{-1}}_{A^{-1}}\underbrace{X^\top X}_{A}\beta.
\end{aligned}$$

Some matrix algebra reminders

The identity matrix $I \in \mathbb{R}^{n \times n}$

- For this matrix, $I_{i,i} = 1$ and $I_{i,j} = 0$ when $j \neq i$.
- $Ix = x$ and $IA = A$.

Some matrix algebra reminders

The identity matrix $I \in \mathbb{R}^{n \times n}$

- For this matrix, $I_{i,i} = 1$ and $I_{i,j} = 0$ when $j \neq i$.
- $Ix = x$ and $IA = A$.

The inverse of a matrix $A \in \mathbb{R}^{n \times n}$

A^{-1} is called the inverse of A if

- $AA^{-1} = I$.
- or equivalently $A^{-1}A = I$.

Some matrix algebra reminders

The identity matrix $I \in \mathbb{R}^{n \times n}$

- For this matrix, $I_{i,i} = 1$ and $I_{i,j} = 0$ when $j \neq i$.
- $Ix = x$ and $IA = A$.

The inverse of a matrix $A \in \mathbb{R}^{n \times n}$

A^{-1} is called the inverse of A if

- $AA^{-1} = I$.
- or equivalently $A^{-1}A = I$.

The pseudo-inverse of a matrix $A \in \mathbb{R}^{n \times m}$

- $\tilde{A}^{-1}$ is called the left pseudoinverse of A if $\tilde{A}^{-1}A = I$.
$$\tilde{A}^{-1} = (A^\top A)^{-1} A^\top, \qquad n > m$$
- $\tilde{A}^{-1}$ is called the right pseudoinverse of A if $A\tilde{A}^{-1} = I$.
$$\tilde{A}^{-1} = A^\top (AA^\top)^{-1}, \qquad m > n$$

sklearn

Fitting a model to data

```
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(X, Y)
```

Getting predictions

We can get predictions for all inputs as an array

```
Z = model.predict(X)
```

Statsmodels

Fitting a model to data X, Y

```
import statsmodels.api as sm
Xa = sm.add_constant(X) # adds a constant factor to the data
model = sm.OLS(Y, Xa)
results = model.fit()
```

Getting predictions

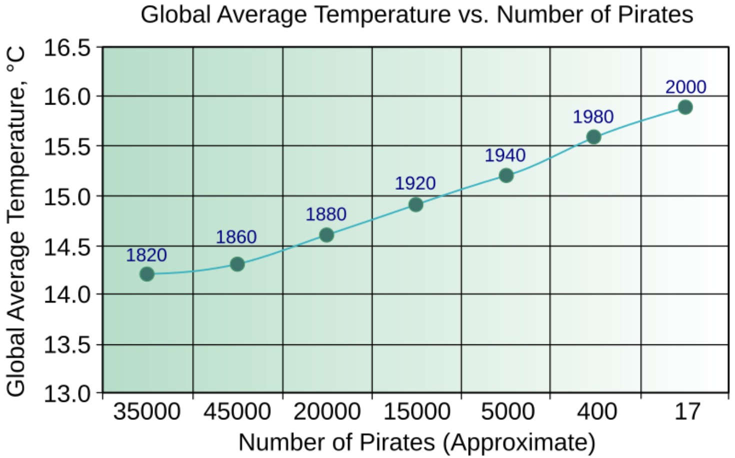
The prediction is not just a point!

```
z = results.get_prediction(Xa[t])
z.predicted_mean # This is E[y|x]
```

Pitfalls

- β_i tells us how much y is correlated with $x_{t,i}$
- However, multiple correlations might be evident.
- Some features may be irrelevant
- The relationship may not be linear
- Correlation is not causation

Correlation is not causation



Linear regression exercises

- Exercises 8, 13 from ISLP
- A variant of Ex. 13 but with Y generated independently of X.