Linear regression is a solution to the problem of predicting a real valued variable, called the response, from a set of inputs, called covariates.

In some communities, like ORFE, this is taught in depth. In others, like CS, you are meant to pick it up if needed.

The goal is to predict $y$ from $x$ with a linear function:

$$F(x) = \beta_1 x + \beta_0$$

**Example:**
- Given the stock price today, what is the price tomorrow?
- Given today's precipitation, what is the precipitation in a week?
- Given my mother's height, what is my shoe size?
- Others? Where have you seen linear regression?
Usually, we have multiple covariates, each representing a different feature of the data that might be predictive of the response.

\[ X_n = \langle x_{n,1}, x_{n,2}, x_{n,3}, \ldots, x_{n,p} \rangle \]

We will fit a linear function of these inputs

\[ f(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i \]

In general, \( \beta^T x = 0 \) is a hyperplane.

This set-up is less limiting than you might imagine:

- Inputs can be
  - any feature of the data
  - transformations of original features, e.g.,
    \[ x_2 = \log x_1 \text{ or } x_2 = \sqrt{x_1} \]
  - a basis expansion, e.g.,
    \[ x_2 = x_1^2, x_3 = x_1^3 \]
  - indicator functions of qualitative inputs, e.g.,
    \[ 1[ \text{is the subject brown-haired} ] \]
  - interactions b/w inputs, e.g.,
    \[ x_3 = x_1 \cdot x_2 \]

Its simplicity and flexibility make linear regression one of the most important and widely used statistical prediction methods.
Fitting a regression

Given data \( \Omega = \{ (x_n, y_n) \}_{n=1}^{N} \), find the coefficient \( \beta \) that can predict \( y_{\text{new}} \) from \( x_{\text{new}} \).

Simplification: \( \beta \) intercept, i.e., \( \beta_0 = 0 \)

one input (\( p = 1 \))

Reasonable approach: minimize the sum of the squared Euclidean distance between each fitted response \( \beta x_n \) and true response \( y_n \).

\[
\text{RSS} = \sum_{n=1}^{N} (y_n - \beta x_n)^2
\]

\( \text{RSS} \): "residual sum of squares"

\[
\text{RSS} (\beta) = \frac{1}{2} \sum_{n=1}^{N} (y_n - \beta x_n)^2 \quad \text{Objective as a function of} \ \beta.
\]

\[
\frac{d}{d\beta} \text{RSS} (\beta) = -\sum_{n=1}^{N} (y_n - \beta x_n) x_n \quad \text{Derivative}
\]

\[
\hat{\beta} = \frac{\sum_{n=1}^{N} y_n x_n}{\sum_{n=1}^{N} x_n^2}
\]

\( \hat{\beta} \): Optimal value

\[
\hat{y}_{\text{new}} = \hat{\beta} x_{\text{new}}
\]

Note: \( x_{\text{new}} \) is always assumed given.
In general we have an intercept and multiple inputs:

\[ y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i \]

To simplify, set \( \beta_{p+1} = \beta_0 \) and \( x_{p+1} = 1 \).

\[ y = \beta^T x \]

and

\[ \text{RSS}(\beta) = \frac{1}{2} \sum_{n=1}^{N} (y_n - \beta^T x_n)^2 \]

The derivative with respect to one component of \( \beta \) is

\[ \frac{\partial \text{RSS}(\beta)}{\partial \beta_i} = -\sum_{n=1}^{N} (y_n - \beta^T x_n) x_{n,i} \]

The gradient is

\[ \nabla_{\beta} \text{RSS}(\beta) = -\sum_{n=1}^{N} (y_n - \beta^T x_n) x_n \]

[Here you might stop and plug this into a blackbox optimizer.]

But, in this case we can solve exactly...
The design matrix is defined:

\[ X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix} \]

Each row is an input vector from our data.

The response vector is

\[ y = \langle y_1, y_2, \ldots, y_N \rangle \]

Each component is a response from our data.

Now, \( \nabla_\beta \text{RSS}(\beta) = -X^T(y - X\beta) \) \( (p \times N) (N \times 2) \rightarrow p \times 2 \)

Setting to the 0-vector and solving gives

\[ X^Ty - X^TX\hat{\beta} = 0 \]

\[ X^TX\hat{\beta} = X^Ty \]

\[ \hat{\beta} = (X^TX)^{-1}X^Ty \left\downarrow \text{normal equations} \right. \]

Note: \( X^TX \) must be invertible, i.e., \( X \) must be full rank.
Probabilistic interpretation

\[ Y_n \sim N(\beta^T X_n, \sigma^2) \]

Like putting a Gaussian "bump" around the mean, which is a linear function of the inputs.

Note: Conditional model - inputs are not modeled.

Conditional ML:

\[
\mathcal{L}(\beta; X_{1:N}, Y_{1:N}) = \log \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left\{ -\frac{(y_n - \beta^T x_n)^2}{2\sigma^2} \right\} \\
= \sum_{n=1}^{N} -\frac{1}{2} \log 2\pi \sigma^2 - \frac{1}{2} \frac{(y_n - \beta^T x_n)^2}{\sigma^2}
\]

Optimize w.r.t. \( \beta \) gives

\[
\hat{\beta} = \arg \max \frac{1}{2} \sum_{n=1}^{N} (y_n - \beta^T x_n)^2 / \sigma^2
\]

This is equivalent to minimizing the RSS.

Prediction comes from expectation:

\[
\mathbb{E}[Y_{\text{new}} | X_{\text{new}}] = \hat{\beta}^T X_{\text{new}}
\]

Note: \( \sigma^2 \) does not play a role.
Important aside: Bias-Variance trade off

Consider a random data set drawn from a LR model

\[ Y \mid x_n, \beta \sim N(\beta^* x_n, \sigma^2) \]

We can contemplate the MLE \( \hat{\beta} \) as a random variable governed by the distribution of this data set.

Consider a new input \( x \). How good is the predicted response mean compared to the true one?

\[
\text{MSE} (\hat{\beta} x) = \mathbb{E}_{\sigma} \left[ (\hat{\beta} x - \beta x)^2 \right]
\]

\[ \uparrow\quad \uparrow \quad \text{random}\quad \text{not random} \]

\[ = \mathbb{E}(\hat{\beta}^2 x^2) - 2 \mathbb{E}[\hat{\beta} x] \beta x + (\beta x)^2 \]

\[ = \mathbb{E}(\hat{\beta}^2 x^2) - 2 \mathbb{E}[\hat{\beta} x] \beta x + (\beta x)^2 + \mathbb{E}[(\hat{\beta} x)^2] - \mathbb{E}[\hat{\beta} x]^2 \]

\[ = (\mathbb{E}(\hat{\beta}^2 x^2) - \mathbb{E}[\hat{\beta} x]^2) + (\mathbb{E}[\hat{\beta} x] - \beta x)^2 \]

\[
\begin{align*}
\text{Variance} & \quad \text{Squared bias} \\
\text{reflects how sensitive the estimator is to the randomness inherent in the data.} & \quad \text{\( \mathbb{E}[\hat{\beta} x] - \beta x = 0 \) means \( \hat{\beta} \) is unbiased.}
\end{align*}
\]

- Classical stats focuses on unbiased estimators
- Modern stats explores the trade off.
Gauss–Markov theorem
The MLE / least squares estimate is the unbiased estimate with the smallest variance.

In regression, we trade-off bias for variance through regularization – placing constraints on $\beta$.

If the true MLE $\hat{\beta}$ lives outside the space then our estimator must be biased.

- Constraining $\Rightarrow$ lower variance
- G.M. $\Rightarrow$ biasedness.

- Regularization encourages smaller and simpler models.
- Intuitively, simpler models are more robust to overfitting, generalizing poorly but of close match to training data.
- Simpler models are also more interpretable, often a goal of regression.

Ridge regression

Optimize the RSS subject to a constraint on the sum of squares of the coefficients. This constrains $\beta$ to be in a sphere.

$$\min \sum_{n=1}^{N} (y_n - \beta^T x_n)^2$$

$$\sum_{i=1}^{p} \beta_i^2 \leq s$$

As $s$ increases, bias $\downarrow$, variance $\uparrow$.

- Contours of equal RSS
- Sphere w/ radius $s$
With simple calculus, can express the ridge optimization problem as:

$$ \hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \sum_{n=1}^{N} \frac{1}{2} (y_n - \hat{\beta}^T x_n)^2 + \lambda \sum_{i=1}^{p} \beta_i^2 $$

**Nice:** for fixed $\lambda$, this is convex.
- 1-1 mapping btw $\lambda$ and $\mathbf{s}$
- $\lambda$ is the complexity parameter

Choice of $\lambda$ has a large effect on our fit.

**Q:** What happens if we use training error to fit $\lambda$?
**A:** No regularization!

In practice: use $X$-validation.

- Divide the data into $k$ folds; grid $\lambda$ into candidates.
- For each fold, and candidate $\lambda$:
  - Estimate $\hat{\beta}_{k,\lambda}$ on the out-of-fold samples.
  - Compute error within fold:
    $$ E_{n,\lambda} = \frac{1}{n} \sum_{n=1}^{N} (y_n - \hat{\beta}_{k,\lambda}^T x_n)^2 $$
  - Now, we have an estimate of error for each point, computed from a parameter estimate that did not include that point.
- Finally, choose $\lambda = \arg\min \frac{1}{N} \sum_{n=1}^{N} E_{n,\lambda}$.
Bayesian linear regression

Ridge regression has a close connection to a Bayesian model:

\[ \beta_i \sim N(0, \frac{1}{\lambda}) \]
\[ y_n | x_n, \beta \sim N(\beta^T x_n, \sigma^2) \]

Consider MAP estimation of \( \beta \):

\[ \hat{\beta}_{\text{MAP}} = \arg \max_{\beta} \log p(\beta | x_{1:n}, y_{1:n}, \lambda) \]
\[ = \arg \max_{\beta} \log p(y_{1:n} | x_{1:n}, \beta) \prod_{i=1}^{p} p(\beta_i | \lambda) \]
\[ = \arg \max_{\beta} \log p(y_{1:n} | x_{1:n}, \beta) + \sum_{i=1}^{p} \log p(\beta_i | \lambda) \]

Recall: this = RSS(\( \beta \))

Note: \( p(\beta_i | \lambda) = \left( \frac{1}{\sqrt{2\pi \lambda}} \right) \exp \left( -\frac{1}{2\lambda} \beta_i^2 \right) \)

Same as ridge regression!

As \( \lambda \) increases, we move further from MLE.
As \( \lambda \) decreases, we move closer.
Both data and prior influence the Bayes estimate.
Note: a true blue Bayesian would never set $\lambda$ by X-validation. This uses the data to set the prior!

**Summary of ridge**
- constrain $\hat{\beta}$ to be inside a sphere
- equiv. to minimize RSS + regularization
- also called *shrinkage*, bcl coefficients are pushed towards 0.
- Ridge trades off bias for decrease in var.
- Equivalent to MAP in Bayesian model.

**Briefly: Lasso**

$$\text{minimize } \text{RSS}(\beta) \text{ s.t. } \sum_{i=1}^{p} |\beta_i| < s$$

(replace $L_2$ with $L_1$)

$$\Rightarrow \text{min } \text{RSS}(\beta) + \lambda \sum_{i=1}^{p} |\beta_i|$$

Nice: Still convex.

The contour will first hit the region at a corner. This finds sparse solution $\| \beta \|_1$ coefficients equal to $\lambda$.

Efficient form of feature selection