# 2. Linear Regression

# Outline

## Problem setup

- example: predicting house prices from features
- *features*: size, number of bedrooms, etc.
- *response*: price
- goal: learn a model that predicts price from features
- *training data*: pairs of features and prices

| Living area ($\mathrm{ft}^2$) | # Bedrooms | Price ($1000s) |
|---|---|---|
| 2104 | 3 | 400 |
| 1600 | 3 | 330 |
| 2400 | 3 | 369 |
| 1416 | 2 | 232 |
| 3000 | 4 | 540 |
| ⋮ | ⋮ | ⋮ |

## Problem setup

- data:
$$\{(x^{(i)}, y^{(i)}) \mid i = 1, \cdots, n\},$$
- $x^{(i)} \in \mathbf{R}^d$ is a *feature* vector
- $y^{(i)} \in \mathbf{R}$ is the *response* variable
- *linear* model (*hypothesis*):
$$h(x;\theta) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d = \theta^T x$$
- $\theta = (\theta_0, \theta_1, \cdots, \theta_d) \in \mathbf{R}^{d+1}$ is the *parameter* vector.
- $x = (1, x_1, \cdots, x_d) \in \mathbf{R}^{d+1}$ is the *augmented* feature vector.
- objective: learning $\theta^*$ from the data so that $h(x;\theta^*)$ predicts $y$ *well*.

# Loss function

- squared loss:

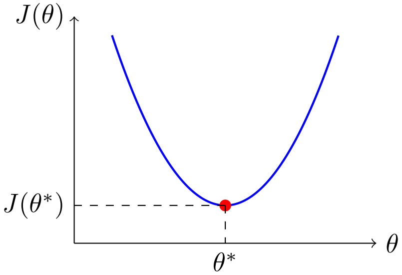
$$J(\theta) = \frac{1}{2n} \sum_{i=1}^{n} \left( h(x^{(i)}, \theta) - y^{(i)} \right)^2$$

- learn $\theta^*$ by minimizing $J(\theta)$.
- this is called *ordinary least squares* (OLS) regression model.

# Outline

# Solve the OLS problem

- solve for $\theta^*$ that minimizes $J(\theta)$.




- $J(\theta)$ is a *convex* function of $\theta$.
- $J(\theta)$ has a shape like a *bowl* with a single *minimum point*.
- analytical solution
- gradient descent

## Analytical solution

- express $J(\theta)$ using matrix notation:

$$J(\theta) = \frac{1}{2n}(X\theta - y)^T(X\theta - y),$$

  where $X$ is the *design matrix* and $y$ is the *response vector*.

- differentiate $J(\theta)$ with respect to $\theta$ and set to zero:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{n}X^T(X\theta - y).$$

- the **normal equation**

$$X^T X\theta = X^T y,$$

- solve for $\theta^*$

$$\theta^* = (X^T X)^{-1}X^T y.$$

## Computing the analytical solution

- it is attempting to directly use $\theta^* = (X^T X)^{-1} X^T y$ to compute $\theta^*$
- not recommended because it involves inverting a matrix
- in practice, direct matrix inversion is rarely used
- alternative methods that are numerically more stable are used
- they often involve matrix factorization techniques
- *QR decomposition*, *SVD*, *Cholesky decomposition*

# Computing the analytical solution

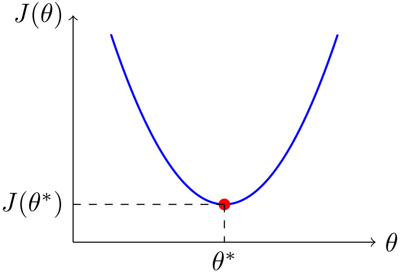
- *QR decomposition*: $X = QR$
- $Q$ is an orthonormal matrix, i.e., $Q^T Q = I$
- $R$ is an upper triangular matrix
- $\theta^* = R^{-1} Q^T y \iff R\theta^* = Q^T y$
- ```
  Q, R = jnp.linalg.qr(x)
  theta_qr = jax.scipy.linalg.solve_triangular(R, Q.T @ y)
  ```
  ```

# Gradient descent

- update rule:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j},$$

  for all $j = 0, 1, \cdots, d$.
- $\alpha$ is the *learning rate*.
- repeat until convergence.



# Outline

# Probabilistic interpretation

- the loss function $J(\theta)$ can be derived from the a probabilistic model
- assume the observed response $y^{(i)}$ is generated by

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)},$$

  where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is the *noise* term
- the likelihood of $y^{(i)}$ given $x^{(i)}$:

$$p(y^{(i)}|x^{(i)};\theta) = \mathcal{N}(y^{(i)}|\theta^T x^{(i)}, \sigma^2)$$

- the *maximum likelihood estimation* (MLE) of $\theta$ is equivalent to minimizing $J(\theta)$

# Maximum likelihood estimation

- data: $\{(x^{(i)}, y^{(i)}) \mid i = 1, \cdots, n\}$,
- likelihood for $y^{(i)}$

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}\left(y^{(i)} - \theta^T x^{(i)}\right)^2\right)$$

- *likelihood function* for all data

$$L(\theta) = \prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}\left(y^{(i)} - \theta^T x^{(i)}\right)^2\right)$$

## Maximum likelihood estimation

- log-likelihood function

$$\ell(\theta) = \log L(\theta) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

- MLE of $\theta$: $\theta^*$ that maximizes $\ell(\theta)$

$$\theta^* = \arg\max_{\theta} \ell(\theta)$$

- maximizing $\ell(\theta)$ is equivalent to minimizing $J(\theta)$

# Tutorial

Linear regression tutorial