Statistical Optimization: Lecture 6

Ordinary Least Squares and Gradient Descent

Zijian Guo

Zhejiang University
Center for Data Science

March 30, 2026

Least-squares setup

Given data $\{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, consider the linear predictor

$$f_\theta(x) = x^\top \theta.$$

Then the empirical least-squares objective is

$$\widehat{R}(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^\top \theta)^2.$$

Let

$$y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n, \qquad X = \begin{pmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{pmatrix} \in \mathbb{R}^{n \times d}.$$

Then

$$\widehat{R}(\theta) = \frac{1}{n} \|y - X\theta\|_2^2.$$

Closed-form solution: normal equations

Expand the objective:

$$\widehat{R}(\theta) = \frac{1}{n}\left(\|y\|_2^2 - 2\theta^\top X^\top y + \theta^\top X^\top X\theta\right).$$

Hence

$$\nabla\widehat{R}(\theta) = \frac{2}{n}(X^\top X\theta - X^\top y).$$

Setting the gradient to zero gives the normal equations

$$X^\top X\widehat{\theta} = X^\top y.$$

If $X^\top X$ is invertible, we have

$$\widehat{\theta} = \left(\frac{1}{n}X^\top X\right)^{-1}\left(\frac{1}{n}X^\top y\right).$$

Orthogonal projection

Let $V \subset \mathbb{R}^n$ be a linear subspace, and let $y \in \mathbb{R}^n$.

A vector $\bar{y} \in V$ is called the orthogonal projection of y onto V if

$$y - \bar{y} \perp V.$$

That is,

$$(y - \bar{y})^\top v = 0, \qquad \forall v \in V.$$

Interpretation:

- $\bar{y}$ lies in the subspace V,
- the error $y - \bar{y}$ is orthogonal to the whole subspace,
- so $\bar{y}$ is the point in V that best matches y.

Geometric interpretation of OLS (1/2)

Proposition. The vector of predictions

$$X\hat{\theta} = X(X^\top X)^{-1}X^\top y$$

is the orthogonal projection of $y \in \mathbb{R}^n$ onto $\mathrm{im}(X) \subset \mathbb{R}^n$, the column space of X. Here $\mathrm{im}(X) := \{Xa : \ a \in \mathbb{R}^d\}$.

Proof.
Let$p := X(X^\top X)^{-1}X^\top y$. Then clearly $p \in \mathrm{im}(X)$. For any $a \in \mathbb{R}^d$, we have

$$a^\top X^\top (y - p) = a^\top X^\top \Big(y - X(X^\top X)^{-1}X^\top y\Big) = 0.$$

Hence

$$(Xa)^\top (y - p) = 0, \qquad \forall a \in \mathbb{R}^d.$$

So $y - p$ is orthogonal to every vector in $\mathrm{im}(X)$.

Therefore $p = X\hat{\theta}$ is the orthogonal projection of y onto $\mathrm{im}(X)$. □

Geometric interpretation of OLS (2/2)

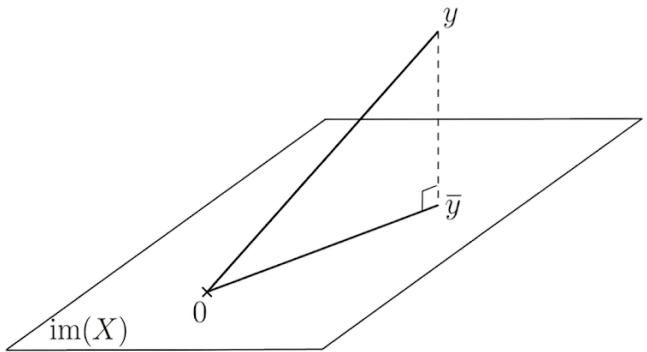


Figure: A geometric interpretation of OLS

Extension to feature maps

We can extend linear regression to nonlinear features by introducing a feature map

$$\phi : \mathbb{R}^d \to \mathbb{R}^p.$$

Then we still use a linear model in the transformed feature space:

$$f_\theta(x) = \phi(x)^\top \theta.$$

Define the feature matrix

$$\Phi = \begin{pmatrix} \phi(x_1)^\top \\ \vdots \\ \phi(x_n)^\top \end{pmatrix} \in \mathbb{R}^{n \times p}.$$

Then the least-squares problem becomes

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \|y - \Phi\theta\|_2^2.$$

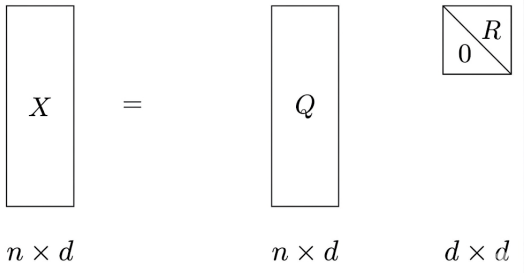
Outline

Least Squares and QR decomposition

Least Squares via First Order Method

Least Squares via Second Order Method

QR decomposition: Full Column Rank



We conduct the QR decomposition as

$$X = QR,$$

where $R \in \mathbb{R}^{d \times d}$ is upper triangular and invertible and

$$Q \in \mathbb{R}^{n \times d}, \qquad Q^\top Q = I_d.$$

Applying QR to least squares

Then

$$X^\top X = (QR)^\top (QR) = R^\top Q^\top QR = R^\top R,$$

and

$$X^\top y = (QR)^\top y = R^\top Q^\top y.$$

Recall the normal equation

$$X^\top X \widehat{\theta} = X^\top y.$$

So the normal equation becomes

$$R^\top R \widehat{\theta} = R^\top Q^\top y.$$

Since R is invertible, we obtain

$$R\widehat{\theta} = Q^\top y.$$

So least squares is reduced to solving an upper-triangular system.

Solving OLS with QR in Python

```
import numpy as np
# Input:
# X : (n, d) design matrix
# y : (n,) response vector

# Step 1: QR factorization
Q, R = np.linalg.qr(X, mode='reduced') #thin QR

# Step 2: project y onto the orthonormal basis
b = Q.T @ y

# Step 3: solve the upper-triangular system R theta = b
theta = np.linalg.solve(R, b)
```

This returns the least-squares solution $\widehat{\theta}$.

Outline

Least Squares and QR decomposition

Least Squares via First Order Method

Least Squares via Second Order Method

Gradient Descent

We consider the optimization problem

$$\min_{\theta \in \mathbb{R}^d} f(\theta).$$

For a small step $v \in \mathbb{R}^d$, the first-order Taylor expansion gives

$$f(\theta + v) \approx f(\theta) + \nabla f(\theta)^\top v.$$

So to decrease f, we want to choose v such that

$$\nabla f(\theta)^\top v < 0.$$

And the steepest descent direction is $-\nabla f(\theta)$.

Gradient descent (GD) starts from an initial point $\theta_0 \in \mathbb{R}^d$, and updates by

$$\theta_{t+1} = \theta_t - \gamma_t \nabla f(\theta_t), \qquad t \geq 0,$$

where $(\gamma_t)_{t \geq 0}$ is a sequence of step sizes.

Gradient Descent for OLS

For OLS, the objective function for $X \in \mathbb{R}^{n \times d}, y \in \mathbb{R}^n$ is

$$f(\theta) = \frac{1}{2n} \|X\theta - y\|_2^2,$$

Its gradient is

$$\nabla f(\theta) = \frac{1}{n} X^\top (X\theta - y) = \frac{1}{n} X^\top X \theta - \frac{1}{n} X^\top y = H\theta - \frac{1}{n} X^\top y \quad \text{with} \quad H = \frac{1}{n} X^\top X.$$

Now apply gradient descent with a fixed step size $\gamma_t = \gamma$:

$$\theta_{t+1} = \theta_t - \gamma \nabla f(\theta_t) = \theta_t - \gamma \left(H\theta_t - \frac{1}{n} X^\top y \right).$$

Gradient Descent for OLS: error recursion

Note that the minimizer η^* satisfies

$$\nabla f(\eta^*) = H\eta^* - \frac{1}{n}X^\top y = 0 \quad \Longrightarrow \quad H\eta^* = \frac{1}{n}X^\top y.$$

which implies

$$\nabla f(\theta) = H(\theta - \eta^*).$$

The gradient descent can be expressed as

$$\theta_{t+1} = \theta_t - \gamma H(\theta_t - \eta^*).$$

Subtracting η^* from both sides yields

$$\theta_{t+1} - \eta^* = \theta_t - \eta^* - \gamma H(\theta_t - \eta^*) = (I - \gamma H)(\theta_t - \eta^*).$$

Iterating this recursion gives

$$\theta_t - \eta^* = (I - \gamma H)^t(\theta_0 - \eta^*), \qquad t \geq 0.$$

Gradient Descent for OLS: variable distance (1/5)

Substituting the explicit form of the iteration gives

$$\|\theta_t - \eta^*\|_2^2 = \big((I - \gamma H)^t(\theta_0 - \eta^*)\big)^\top \big((I - \gamma H)^t(\theta_0 - \eta^*)\big).$$

Since H is symmetric, $I - \gamma H$ is also symmetric, so

$$\|\theta_t - \eta^*\|_2^2 = (\theta_0 - \eta^*)^\top (I - \gamma H)^{2t}(\theta_0 - \eta^*).$$

Lemma. Assume $0 < \mu = \lambda_{\min}(H) \le \lambda_{\max}(H) = L$. Then for every $x \in \mathbb{R}^d$,

$$\|(I - \gamma H)^t x\|_2^2 = x^\top (I - \gamma H)^{2t} x \le \left(\max_{\lambda \in [\mu, L]} |1 - \gamma\lambda|^{2t}\right) \|x\|_2^2.$$

Gradient Descent for OLS: variable distance (2/5)

Proof. Since H is symmetric, it is orthogonally diagonalizable:

$$H = U\Lambda U^\top,$$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$ with

$$\lambda_i \in [\mu, L].$$

Hence

$$I - \gamma H = U(I - \gamma\Lambda)U^\top,$$

and therefore

$$(I - \gamma H)^{2t} = U(I - \gamma\Lambda)^{2t}U^\top.$$

Gradient Descent for OLS: variable distance (3/5)

So the eigenvalues of $(I - \gamma H)^{2t}$ are

$$(1 - \gamma\lambda_i)^{2t}, \qquad i = 1, \ldots, d.$$

Thus

$$\lambda_{\max}\big((I - \gamma H)^{2t}\big) = \max_{\lambda \in [\mu, L]} |1 - \gamma\lambda|^{2t}.$$

Since $(I - \gamma H)^{2t}$ is symmetric,

$$x^\top (I - \gamma H)^{2t} x \leq \lambda_{\max}\big((I - \gamma H)^{2t}\big) \|x\|_2^2.$$

This proves the result. □

Gradient Descent for OLS: variable distance (4/5)

Theorem. Choose $\gamma = \frac{1}{L}$ and denote $\kappa = \frac{L}{\mu}$. Then for gradient descent, we have

$$\|\theta_t - \eta^*\|_2^2 \leq \left(1 - \frac{1}{\kappa}\right)^{2t} \|\theta_0 - \eta^*\|_2^2 \leq e^{-2t/\kappa} \|\theta_0 - \eta^*\|_2^2.$$

Proof. From the error recursion,

$$\theta_t - \eta^* = (I - \gamma H)^t (\theta_0 - \eta^*).$$

Applying the lemma with $x = \theta_0 - \eta^*$ gives

$$\|\theta_t - \eta^*\|_2^2 \leq \left(\max_{\lambda \in [\mu, L]} |1 - \gamma\lambda|^{2t}\right) \|\theta_0 - \eta^*\|_2^2.$$

Now set $\gamma = 1/L$. For $\lambda \in [\mu, L]$,

$$0 \leq 1 - \frac{\lambda}{L} \leq 1 - \frac{\mu}{L} = 1 - \frac{1}{\kappa}.$$

Gradient Descent for OLS: variable distance (5/5)

Therefore,

$$\|\theta_t - \eta^*\|_2^2 \leq \left(1 - \frac{1}{\kappa}\right)^{2t} \|\theta_0 - \eta^*\|_2^2.$$

Finally, since $1 - x \leq e^{-x}$,

$$\left(1 - \frac{1}{\kappa}\right)^{2t} \leq e^{-2t/\kappa}.$$

We have

$$\|\theta_t - \eta^*\|_2^2 \leq \left(1 - \frac{1}{\kappa}\right)^{2t} \|\theta_0 - \eta^*\|_2^2 \leq e^{-2t/\kappa} \|\theta_0 - \eta^*\|_2^2.$$

□

Gradient Descent for OLS: function value (1/5)

Recall the OLS objective

$$f(\theta) = \frac{1}{2n}\|X\theta - y\|_2^2,$$

Hence

$$f(\theta) - f(\eta^*) = \frac{1}{2n}\Big(\|X\theta - y\|_2^2 - \|X\eta^* - y\|_2^2\Big).$$

Write

$$X\theta - y = X(\theta - \eta^*) + (X\eta^* - y).$$

Then

$$\|X\theta - y\|_2^2 = \|X(\theta - \eta^*)\|_2^2 + 2(\theta - \eta^*)^\top X^\top (X\eta^* - y) + \|X\eta^* - y\|_2^2.$$

Since the minimizer η^* satisfies

$$X^\top (X\eta^* - y) = 0,$$

the cross term vanishes. Therefore,

$$f(\theta) - f(\eta^*) = \frac{1}{2n}\|X(\theta - \eta^*)\|_2^2 = \frac{1}{2}(\theta - \eta^*)^\top H(\theta - \eta^*).$$

Gradient Descent for OLS: function value (2/5)

Replace θ by θ_t,

$$f(\theta_t) - f(\eta^*) = \frac{1}{2}(\theta_t - \eta^*)^\top H(\theta_t - \eta^*).$$

Substituting again,

$$f(\theta_t) - f(\eta^*) = \frac{1}{2}\big((I - \gamma H)^t(\theta_0 - \eta^*)\big)^\top H\big((I - \gamma H)^t(\theta_0 - \eta^*)\big).$$

Using symmetry and the fact that H commutes with $I - \gamma H$, we get

$$f(\theta_t) - f(\eta^*) = \frac{1}{2}(\theta_0 - \eta^*)^\top (I - \gamma H)^{2t} H(\theta_0 - \eta^*).$$

Gradient Descent for OLS: function value (3/5)

Theorem. Assume $0 < \gamma \leq \frac{1}{L}$. Then for gradient descent applied to OLS,

$$f(\theta_t) - f(\eta^*) \leq \frac{1}{8t\gamma}\|\theta_0 - \eta^*\|_2^2, \qquad t \geq 1.$$

In particular, with $\gamma = \frac{1}{L}$, we have

$$f(\theta_t) - f(\eta^*) \leq \frac{L}{8t}\|\theta_0 - \eta^*\|_2^2.$$

Remark. This result does not require the minimizer to be unique; it still holds even when $\mu = 0$.

Gradient Descent for OLS: function value (4/5)

Proof. Recall that for OLS,

$$f(\theta_t) - f(\eta^*) = \frac{1}{2}(\theta_0 - \eta^*)^\top (I - \gamma H)^{2t} H (\theta_0 - \eta^*).$$

Like in the proof of convergence in distance, it is enough to bound

$$g(\lambda) := \lambda(1 - \gamma\lambda)^{2t}, \qquad \lambda \in [0, L].$$

We differentiate $g(\lambda)$:

$$g'(\lambda) = (1 - \gamma\lambda)^{2t} - 2t\gamma\lambda(1 - \gamma\lambda)^{2t-1}.$$

Factoring out $(1 - \gamma\lambda)^{2t-1}$, we get

$$g'(\lambda) = (1 - \gamma\lambda)^{2t-1}\big(1 - (2t+1)\gamma\lambda\big).$$

Recall that $0 < \gamma \leq \frac{1}{L}$. Hence the only critical point is

$$\lambda^* = \frac{1}{(2t+1)\gamma}.$$

Gradient Descent for OLS: function value (5/5)

Since $g'(\lambda) > 0$ for $\lambda < \lambda^*$ while $g'(\lambda) < 0$ for $\lambda > \lambda^*$, the function g attains its maximum at λ^*. Therefore,

$$g(\lambda) \le g(\lambda^*) = \frac{1}{(2t+1)\gamma}\left(1 - \frac{1}{2t+1}\right)^{2t} = \frac{1}{2t\gamma}\left(1 - \frac{1}{2t+1}\right)^{2t+1}.$$

Now by the well-known inequality

$$\left(1 - \frac{1}{n}\right)^n < e^{-1}, \quad \forall n \in \mathbb{Z}^+$$

we have

$$g(\lambda) \le \frac{1}{2et\gamma} \le \frac{1}{4t\gamma}.$$

Therefore,

$$f(\theta_t) - f(\eta^*) \le \frac{1}{2} \cdot \frac{1}{4t\gamma}\|\theta_0 - \eta^*\|_2^2 = \frac{1}{8t\gamma}\|\theta_0 - \eta^*\|_2^2.$$

□

Solving OLS with Gradient Descent in Python

```
import numpy as np
# X : (n, d) design matrix; y : (n,) response vector
# theta0 : (d,) initial guess; gamma : step size
# tol : tolerance; max_iter : maximum number of iterations
theta = theta0.copy(); n = X.shape[0]; k = 0;
while k < max_iter:
    # Step 1: compute the gradient
    grad = X.T @ (X @ theta - y) / n
    # Step 2: stopping criterion
    if np.linalg.norm(grad) < tol: break
    # Step 3: gradient descent update
    theta = theta - gamma * grad
    k += 1
```

This returns an approximate least-squares solution.

Outline

Least Squares and QR decomposition

Least Squares via First Order Method

Least Squares via Second Order Method

Newton's method

The second-order idea is to approximate the objective near the current iterate by a quadratic model.

For a twice differentiable function,

$$f(\theta) \approx f(\theta_t) + \nabla f(\theta_t)^\top (\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)^\top \nabla^2 f(\theta_t)(\theta - \theta_t).$$

Minimizing this local quadratic approximation gives the Newton update:

$$\theta_{t+1} = \theta_t - [\nabla^2 f(\theta_t)]^{-1} \nabla f(\theta_t).$$

Intuition: Newton uses curvature information to rescale the gradient.

Newton method for least squares

For least squares,

$$f(\theta) = \frac{1}{2n}\|X\theta - y\|_2^2,$$

with

$$\nabla f(\theta) = \frac{1}{n}X^\top(X\theta - y), \qquad \nabla^2 f(\theta) = \frac{1}{n}X^\top X.$$

Since the Hessian is constant, Newton's method gives

$$\theta_{t+1} = \theta_t - \left(\frac{1}{n}X^\top X\right)^{-1}\left(\frac{1}{n}X^\top(X\theta_t - y)\right).$$

After simplification,

$$\theta_{t+1} = (X^\top X)^{-1}X^\top y = \widehat{\theta}.$$

Conclusion: for quadratic least squares, Newton reaches the exact minimizer in one step.

Solving OLS with Newton's Method in Python

```
import numpy as np
# Input:
# X : (n, d) design matrix; y : (n,) response vector
# theta0 : (d,) initial guess
theta = theta0.copy(); n = X.shape[0]
# Step 1: compute the gradient at theta0
grad = X.T @ (X @ theta - y) / n
# Step 2: form the Hessian
H = X.T @ X
# Step 3: solve H p = grad
p = np.linalg.solve(H, grad)
# Step 4: Newton update
theta = theta - p
```

For OLS, Newton's method reaches the solution in one step when $X^\top X$ is invertible.

Main messages

1. **Direct methods:** Least squares admits a closed-form solution:

$$\widehat{\theta} = (X^\top X)^{-1} X^\top y.$$

   and QR decomposition gives a stable numerical implementation.

2. **First-order methods:** computing the gradient. It is iterative, cheap per iteration, and its speed depends on the condition number.

3. **Second-order methods:** computing the gradient and inverse Hessian. For quadratic least squares, it finds the minimizer in one step.