# STATS 230: Computational Statistics
# Numerical linear algebra

Babak Shahbaba

Department of Statistics, UCI

# Overview

- We are interested in solving equations $Ax = b$
- First, I will go over some basic concepts in linear algebra
- Next, I will talk about numerical linear algebra for developing fast computational methods
- Finally, I will discuss these methods in the context of linear regression models
- The review of linear algebra, algorithms, and most of examples presented here are mainly based on Strang (2012)
- For more details, refer to Strang (2012), Boyd and Vandenberghe (2004), and Thisted (1988)

# Some important concepts in linear algebra

# Four fundamental spaces

- Conceptually, $Ax = b$ should be interpreted as "$A$ acts on $x$ to produce $b$".
- Further, we can think of $Ax$ as a linear combinations of the columns of $A$: $x_1 a_1 + x_2 a_2, \dots x_n a_n$, where $a_1, a_2, \dots, a_n$ are the columns of $A$.
- All possible combinations of the columns form the columns space $C(A)$
- $Ax = b$ is solvable if $b \in C(A)$
- The null space $N(A)$ on the other hand includes all vectors $x$ such that $Ax = 0$
- For full column rank matrices, $N(A)$ contains only zero: $x = (0, \dots, 0)$.
- When $A$ is $m$ by $n$, then $C(A)$ is a subspace of $R^m$ and $N(A)$ is a subspace of $R^n$.

# Four fundamental spaces

- We can also talk about two other subspaces: $C(A^\top)$, which is also called the row space, and $N(A^\top)$.
- Together, $C(A)$, $N(A)$, $C(A^\top)$, and $N(A^\top)$ create four fundamental subspaces, which are very important in linear algebra.
- Recall that the dimension, $r$, of a space is the number of independent vectors.
- We can show that
    - The column space $C(A)$ in $R^m$ and the row space $C(A^\top)$ in $R^n$ have the same dimension $r$.
    - The null spaces $N(A)$ and $N(A^\top)$ have dimensions $n - r$ and $m - r$ respectively.

# Four fundamental spaces

- Recall that if $V$ is a subspace of $R^n$, its orthogonal complement is

$$V^{\perp} = \{x | z^{\top} x = 0, \forall z \in V\}$$

then, we can write each vector in $R^n$ as a sum of two vectors from $V$ and $V^{\perp}$

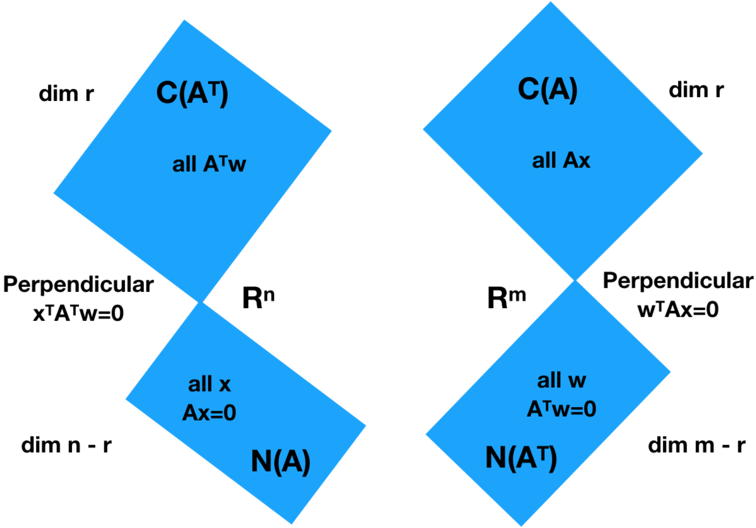
$$R^n = V \overset{\perp}{\bigoplus} V^{\perp}$$

- Given $A_{m \times n}$, for the four fundamental subspaces mentioned above we have

$$\begin{aligned} N(A^{\top}) \quad &\overset{\perp}{\bigoplus} \quad C(A) = R^m \\ N(A) \quad &\overset{\perp}{\bigoplus} \quad C(A^{\top}) = R^n \end{aligned}$$

# Four fundamental spaces

- Schematically, the four subspaces can be presented as follows (Strang, 2012).



# Basis

- A full set of independent vectors form a basis for a space
- Each vector in the space can be presented as a unique combination of these basis vectors
- Possible choices:
  - Standard basis: columns of the identify matrix
  - General basis: columns of any invertible matrix
  - Orthogonal basis: columns of any orthogonal matrix

# Orthogonal matrices

- Note that for a matrix $Q$ with orthonormal columns, $q_1, \ldots, q_n$, we have

$$q_i^\top q_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \qquad Q^\top Q = I$$

- If $Q$ is a square matrix, it is called an *orthogonal matrix* and $Q^\top = Q^{-1}$

- Multiplying a vector by $Q$ doesn't change its length:

$$\|Qx\|^2 = x^\top Q^\top Q x = x^\top x = \|x\|^2$$

# Eigenvalues and eigenvectors

- When $A$ acts on $x$ (i.e., $Ax$), it almost always changes the direction of $x$
- For some special vectors, $Ax = \lambda x$ so $x$ either stretches, shrinks, reverses directions, or stays unchanged
- Then, we say $x$ is an *eigenvector* for $A$ and $\lambda$ is its corresponding *eigenvalue*
- One possible way (good for low-dimensional problems) to find $x$ and $\lambda$ is through solving

$$(A - \lambda I)x = 0$$

  since $(A - \lambda I)$ needs to be singular so its null space includes $x \neq 0$, we have

$$\det(A - \lambda I) = 0$$

- This is called the characteristic equation, which involves $\lambda$ only (not $x$).

# Eigenvalues and eigenvectors

- After we find $\lambda$'s, we can find the corresponding eigenvectors. We can also calculate

$$\det(A) = \prod_{i=1}^{n} \lambda_i \qquad \mathrm{trace}(A) = \sum_{i=1}^{n} \lambda_i$$

- If $A$ is triangular, then its eigenvalues are given by its diagonal elements

- For square matrices, the eigenvalues of $A^2$ are $\lambda_1^2, \ldots, \lambda_n^2$

$$Ax = \lambda x \Rightarrow A^2 x = \lambda A x = \lambda^2 x; \quad \text{In general, } A^k x = \lambda^k x$$

  and the eigenvalues of $A^{-1}$ are $1/\lambda_1, \ldots, 1/\lambda_n$

$$Ax = \lambda x; A^{-1}Ax = \lambda A^{-1}x; A^{-1}x = \frac{1}{\lambda}x$$

- Note that the eigenvectors remain the same

# Diagonalization

- Suppose $A_{n\times n}$ has $n$ independent eigenvectors, $x_1, \ldots, x_n$, which are the columns of an eigenvector matrix $S$
- The corresponding eigenvalues form a diagonal matrix $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$
- Then, we can write them in a matrix form $AS = S\Lambda$, from which we get

$$\begin{aligned} S^{-1}AS &= \Lambda \\ A &= S\Lambda S^{-1} \end{aligned}$$

- Symmetric matrices have real eigenvalues and orthogonal eigenvector matrix $Q$ where,

$$q_i^\top q_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

therefore, $Q^\top Q = I; Q^\top = Q^{-1}$, and $A = Q\Lambda Q^\top$

# Positive definiteness

- A square matrix, $S$, is positive definite if $u^\top S u > 0, \ \forall u \neq 0$
- You can think of $\frac{1}{2} u^\top S u$ as the *energy* function of a system. We can find its minimum by setting the gradient (with respect to $u$) to zero: $Su = 0$, and show that this point is in fact the minimum if the Hessian (second derivatives) $S$ is positive definite.
- For positive definite matrices, all eigenvalues are positive
- $S = A^\top A$ is symmetric and positive definite (when $A$ has independent columns so the only solution to $Ax = 0$ is the zero vector) or at least semidefinite (when $Ax = 0$ has nonzero solutions)
- If $S_1$ and $S_2$ are positive definite, then $S_1 + S_2$ is also positive definite

# Singular vs. nonsingular

- As mentioned above, we are primarily interested in solving equations $Ax = b$
- If possible, we could solve the above equation as $x = A^{-1}b$
- We can express whether this is possible or not in different ways

| **Nonsingular** | **Singular** |
|---|---|
| $A$ is invertible | $A$ is not invertible |
| $Ax = b$ has one solution: $A^{-1}b$ | $Ax = b$ has no solution or infinitely many solution |
| $Ax = 0$ has one solution: $x = 0$ | $Ax = 0$ has many solutions |
| The columns are independent | The columns are dependent |
| The row are independent | The row are dependent |
| The columns space is $R^n$ | The column space has dim $r < n$ |
| The row space is $R^n$ | The row space has dim $r < n$ |
| $A$ has full rank | $A$ has rank $r < n$ |
| $A$ has $n$ positive singular values | $A$ has $r < n$ singular values |
| $A^\top A$ is symmetric positive definite | $A^\top A$ is only semidefinite |
| All eigenvalus of $A$ are nonzero | Zero is an eigenvalue of $A$ |
| The determinant is nonzero | The determinant is zero |

# Numerical linear algebra

# Flops

- In general, solving $Ax = b$ is difficult for big matrices
- The computational cost is lower when working with structured matrices: symmetric, triangular, orthonormal, sparse, diagonal
- The computational cost of algorithms in numerical linear algebra is commonly measured by the total number of floating-point operations (flops)
- A flop is one addition, subtraction, multiplication, or division of two floating-point numbers
- To evaluate the computational cost of an algorithm, we count the total number of flops as a function of the dimensions of matrices and vectors
- This is usually a polynomial function, and we typically focus on the dominant (higher order) terms by using the big-O notation: $\mathcal{O}$.

# Flops

| Operation | Cost |
|---|---|
| Inner product | $\mathcal{O}(n)$ |
| $A_{m\times n}x$ | $\mathcal{O}(mn)$ |
| $A_{m\times n}N_{n\times p}$ | $\mathcal{O}(mnp)$ |
| Solving $Ax = b$; $A$ is dense | $\mathcal{O}(n^3)$ |
| Solving $Ax = b$; $A$ is orthogonal | $\mathcal{O}(n^2)$ |
| Solving $Ax = b$; $A$ is triangular | $\mathcal{O}(n^2)$ |
| Solving $Ax = b$; $A$ is banded with bandwidth $k$ | $\mathcal{O}(k^2n)$ |
| Solving $Ax = b$; $A$ is diagonal | $\mathcal{O}(n)$ |

# Factorization

- As mentioned earlier, it is easier and more computationally efficient to work with structured matrices.
- We now discuss three types of factorizations to generate such matrices
  - $A = LU =$ lower triangular $\times$ upper triangular
  - $A = QR =$ Orthonormal columns $\times$ upper triangular
  - $A = U\Sigma V^\top$, where $U$ and $V$ are orthonormal and $\Sigma$ is a diagonal matrix

# $LU$ Factorization

- To solve $Ax = b$, we could use a set of forward elimination operations to change $A$ to an upper triangular matrix $U$,

$$U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & u_{nn} \end{pmatrix}$$

- Changing the problem to $Ux = c$, we solve the system by backward substitution starting from the last equation.

- It turns out a lower triangular matrix, $L$, can take $U$ back to $A$ such that

$$A = LU$$

- This provides a factorization for $A$

# $LU$ Factorization

- Sometimes, we need to apply some row permutations to $A$ before factorizing it,

$$PA = LU$$

  note that permutations cost zero flops.

- The factorization usually cost $(2/3)n^3$ flops, but it could be much lower for sparse matrices

- We can then solve the equation $PAx = LUx = Pb$ as follows
  - 1) $z_1 = Pb$; zero flops
  - 2) $Lz_2 = z_1$; $n^2$ flops
  - 3) $Ux = z_2$; $n^2$ flops

# Cholesky Factorization

- For symmetric matrices, we obtain symmetric factorizations

$$A = LDL^\top$$

where $D$ is a diagonal matrix so

$$A^\top = (L^\top)^\top DL^\top = LDL^\top$$

- Alternatively, we can write this as

$$A = L\sqrt{D}\sqrt{D}L^\top = L^*L^{*\top}$$

- This factorization costs $(1/3)n^3$ flops

# Orthogonalization: $QR$ factorization

- A common factorization is $A = QR$, where $R$ is an upper triangular matrix and $Q$ has orthonormal columns, $Q^\top Q = I$
- Then, instead of columns $a_1, \ldots, a_n$ as basis, we would use $q_1, \ldots, q_n$
- Two common methods for $QR$ factorization are Gram-Schmidt and Householder (discussed later)
- Using this factorization, operations involving $A^\top A$ simplify to $Q^\top Q = I$
- Additionally, because of $Q$'s *stability*, we can avoid overflow and underflow
- Finally, since multiplying by $Q$ does not change the size, small numerical errors $\Delta b$ will not generate very large errors in $x$:

$$Q(\Delta x) = \Delta b \Rightarrow \|\Delta x\| = \|\Delta b\|$$

# Gram-Schmidt

- We start with setting $q_1 = a_1/\|a_1\|$ and since $a_1 = r_{11}q_1$, we have $r_{11} = \|a_1\|$
- To find $q_2$, we subtract from $a_2$ its component in $q_1$ direction and then normalize:

$$w_2 = a_2 - (q_1^\top a_2)q_1; \quad q_2 = w_2/\|w_2\|$$

- At step $k$, we subtract from $a_k$ its projection on $q_1, \ldots, q_{k-1}$ and then normalize
- Finally, we will have $A_{m\times n} = Q_{m\times n} \times R_{n\times n}$
- In contrast, the Householder algorithm (discussed later) creates $A_{m\times n} = Q_{m\times m} \times R_{m\times n}$

# Gram-Schmidt

Initialize $Q_{m\times n} = 0$, $R_{n\times n} = 0$, and $v = A[, 1]$

$R[1, 1] = \mathrm{norm}(v)$

$Q[, 1] = v/R[1, 1]$

**for** $j = 2$ to $n$ **do**

  v = A[, j]

  **for** $i = 1$ to $j - 1$ **do**

    $R[i, j] = Q[, i]^\top A[, j]$

    $v = v - R[i, j]Q[, i]$

  **end for**

  $R[j, j] = \mathrm{norm}(v)$

  $Q[, j] = v/R[j, j]$

**end for**

# Singular value decomposition (SVD)

- We would like to work with diagonal matrices, but $A = S\Lambda S^{-1}$ doesn't produce orthogonal $S$ in general
- We instead use the following factorization $A = U\Sigma V^\top$, where $U$ and $V$ are orthonormal and $\Sigma$ is a diagonal matrix
- This is similar to $Q\Lambda Q^\top$, but the left and right orthogonal matrices are not the same.
- Then,

| | |
|---|---|
| Instead of eigenvalues $\Lambda$, | we have singular values $\Sigma$ |
| Instead of eigenvectors $S$, | we have left and right singular vectors $U$ and $V$ |
| Instead of $Ax = \lambda x$, | we have $Av = \sigma u$ |
| Instead of $AS = S\Lambda$, | we have $AV = U\Sigma$ |

# Singular value decomposition (SVD)

- When calculating $A^\top A$, we have

$$A^\top A = V\Sigma^\top U^\top U\Sigma V^\top = V\Sigma^\top \Sigma V^\top$$

  which is similar to $Q\Lambda Q^\top$

- The diagonal elements $\sigma_i^2$ are the positive eigenvalues of $A^\top A$
- $V$ contains the orthogonal eigenvectors of $A^\top A$
- $U$ contains the orthogonal eigenvectors of $AA^\top$
- $U$ and $V$ provide perfect bases for column and row space of $A$,

$$A_{m\times n} = U_{m\times r}\Sigma_{r\times r}V^\top{}_{r\times n}$$

# Singular value decomposition (SVD)

- The above presentation is the reduced form.
- By adding any orthonormal basis $v_{r+1}, \ldots, v_n$ from the null space of $A$, and any orthonormal basis $u_{r+1}, \ldots, u_m$ from the null space of $A^\top$, and completing $\Sigma$ to an $m \times n$ matrix by adding zeros, we can write the full SVD as follows:

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^\top{}_{n\times n}$$

- Ordering $\sigma_1 \geq \ldots \geq \sigma_r > 0$, we can write

$$A = u_1\sigma_1 v_1^\top + \ldots + u_r\sigma_r v_r^\top = \sum_{i=1}^{r} \sigma_i u_i v_i^\top$$

- Left and right singular vectors are also known as Karhunen-Loève bases

# Pseudo inverse

- Using SVD, we can now define a more general concept of inverse
- Note that $Av_i = \sigma_i u_i$ can be interpreted as $A$ taking a vector (a basis in this case) from the row space to the column space
- We can define $A^\dagger$ (called pseudo inverse of $A$) that reverses this operation,

$$A^\dagger u_i = v_i/\sigma_i, \; i \leq r; \quad A^\dagger u_i = 0, \; i > r$$

- Singular values of $A^\dagger$ are $\Sigma^\dagger = \mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_r)$,

$$A^\dagger = V\Sigma^\dagger U^\top$$

- If $\mathrm{rank}(A) = n$, then $A^\dagger = (A^\top A)^{-1}A^\top$ and $A^\dagger A = I_n$
- If $\mathrm{rank}(A) = m$, then $A^\dagger = A^\top (AA^\top)^{-1}$ and $AA^\dagger = I_m$
- If $A$ is square and invertible, then $A^\dagger = A^{-1}$

# Condition number

- When solving linear systems, $Ax = b$, we are interested in measuring the sensitivity of the results to small fluctuations in inputs (e.g., round-off error due to floating-point representation)
- That is, we want to measure $\Delta x$ given $\Delta b$
- Suppose $A$ is positive definite

$$\Delta x = A^{-1} \Delta b$$

# Condition number

- Recall that the eigenvalues of $A^{-1}$ are $1/\lambda(A)$
- Therefore, $1/\lambda_{min}(A)$ is the largest eigenvalue, and vectors along with the corresponding eigenvectors have the maximum stretch,

$$\|\Delta x\| \leq \|\Delta b\| / \lambda_{min}(A)$$

- It is better to work with relative errors,

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\lambda_{max}(A)}{\lambda_{min}(A)} \frac{\|\Delta b\|}{\|b\|}$$

- The term $c(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$ is called the condition number

# Condition number

- For non-symmetric matrices, the error might blow up along vectors other than eigenvectors.
- In such case, we use the norm $\|A\|$ instead, which is defined as

$$\max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

- Then, the condition number is then defined as $c(A) = \|A\| \|A^{-1}\|$
- As a rule of thumb, the computer loses $\log c$ decimals to roundoff error

# Least squares estimation

# Least squares estimates

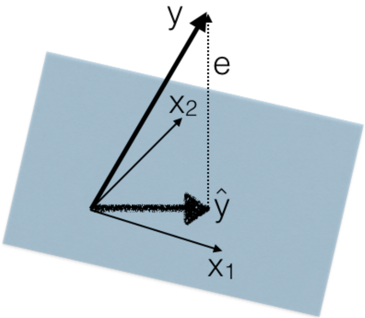
- We now discuss least square estimates for $X\beta = y$, where $X$ is a $n \times p$ $(n > p)$ matrix
- This doesn't have any solution: $X^{-1}$ doesn't exist; the system is overdetermined (too many equations)
- Instead, we find a solution $\hat{\beta}$ such that

$$X\hat{\beta} = \hat{y}; \qquad y = \hat{y} + e$$

- We find the best solution $\hat{\beta}$ by making $e$ small so $y$ and $\hat{y}$ are "close" to each other
- We can minimize $\|e\|^2 = \|y - X\hat{\beta}\|^2 = (y - X\hat{\beta})^\top (y - X\hat{\beta})$

# Least squares estimates

- Geometrically, however, $e$ would be small when it's perpendicular to $\hat{y}$ and the column space of $X$



# Least squares estimates

- Recall that $N(X^\top) = c(X)^\perp$; $e$ is in the null space of $X^\top$

$$\begin{aligned} X^\top e &= 0 \\ X^\top(y - \hat{y}) &= 0 \\ X^\top(y - X\hat{\beta}) &= 0 \end{aligned}$$

- From this, we get the following normal equation,

$$X^\top X\hat{\beta} = X^\top y$$

- Therefore,

$$\begin{aligned} \hat{\beta} &= (X^\top X)^{-1}X^\top y \\ \hat{y} &= X\hat{\beta} = X(X^\top X)^{-1}X^\top y = Hy \end{aligned}$$

# Least squares estimates

- To find $\hat{\beta}$, we can solve the normal equation directly, however, this could create some computational difficulties if the condition number of $(X^\top X)$ (which is the square of the condition number of $X$) could be large

- To avoid this, we could use orthogonalization $X = QR$, then

$$\begin{aligned} X^\top X\hat{\beta} &= X^\top y \\ (QR)^\top QR\hat{\beta} &= (QR)^\top y \\ R^\top R\hat{\beta} &= R^\top Q^\top y \\ R\hat{\beta} &= Q^\top y \end{aligned}$$

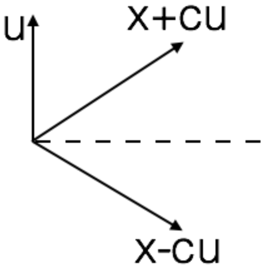
- We find $Q^\top y$, then use back substitution

# The Householder algorithm

- To find $QR$, we could use Gram-Schmidt as discussed before
- Alternatively, we can use the Householder algorithm
- For this, we use reflectors $H = I - 2uu^\top$, where $u$ is a unit length vector
- $H$ is symmetric and orthogonal: $H^\top H = (I - 2uu^\top)(I - 2uu^\top) = I$
- $H$ reflects $u$ to $-u$: $Hu = (I - 2uu^\top)u = u - 2u = -u$
- $H$ doesn't change $x$ vectors perpendicular to $u$: $Hx = x$

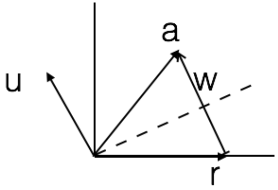
# The Householder algorithm

- In general, $H$ reflects $x + cu$ to $x - cu$; the mirror is perpendicular to $u$




- We can use this fact to create an upper triangular matrix $R$ by taking each column of $X$ and creating zeros below its diagonal

# The Householder algorithm




- Consider $a = \binom{4}{3}$, we want to find $H$ such that $Ha = r$
- Since $H$ is orthogonal, $\|a\| = \|r\|$, which means $r = \binom{5}{0}$
- $w = a - r = \binom{-1}{3}$, and $u = w/\|w\| = \frac{1}{\sqrt{10}}\binom{-1}{3}$
- $H = I - 2uu^\top$, therefore

$$H = \frac{1}{5}\begin{pmatrix} 4 & 3 \\ 3 & -4 \end{pmatrix}$$

# The Householder algorithm

- We continue as above until we find $H_1, \ldots, H_p$
- Applying these reflectors to $X$ creates $R$

$$H_p \ldots H_1 X = R$$

therefore,

$$X = (H_p \ldots H_1)^{-1} R$$

which means

$$Q^{-1} = Q^\top = H_p \ldots H_1$$

- In practice, we don't need to find $Q$, we simply apply the reflectors to $y$,

$$R\hat{\beta} = Q^\top y = H_p \ldots H_1 y$$

and then use back substitution to find $\hat{\beta}$

# The Householder algorithm

Initialize $U_{n\times p} = 0$

**for** $k = 1$ to $p$ **do**

w = X[k:n, k]

$w[1] = w[1] - \mathrm{norm}(w)$

$u = w/\mathrm{norm}(w)$

$U[k:n, k] = u$

$X[k:n, k:p] = X[k:n, k:p] - 2u(u^\top X[k:n, k:p])$

**end for**

Set $R_{p\times p}$ to the upper triangular of $X$

# Least squares using SVD

- Recall the SVD factorization:

$$\begin{aligned} X &= U\Sigma V^\top \\ X^\top X &= V\Sigma^\top \Sigma V^\top \end{aligned}$$

- We could use this factorization to solve the normal equation,

$$\begin{aligned} V\Sigma^\top \Sigma V^\top \hat{\beta} &= V\Sigma^\top U^\top y \\ V^\top \hat{\beta} &= (\Sigma^\top \Sigma)^{-1} \Sigma^\top U^\top y \\ \hat{\beta} &= V\Sigma^\dagger U^\top y \\ &= X^\dagger y \end{aligned}$$

- Here, $\Sigma^\dagger = \mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_n)$ is the pseudoinverse of $\Sigma$ and $X^\dagger$ is the pseudoinverse of $X$

- This is the most compact form for least squares estimates

# Recursive least squares

- Suppose we have obtained $n$ observations, $X_n$ and $y_n$, and found the least squares estimates
$$\hat{\beta}_n = (X_n^\top X_n)^{-1} X_n^\top y_n$$

- Later, we obtain $k$ more observations, which we denote as $X_k$ and $y_k$

- We could of course put all the observations together, $X_{n+k}$ and $y_{n+k}$ to obtain the new estimate of regression parameters, $\hat{\beta}_{n+k}$, from
$$\begin{aligned} X_{n+k}\beta_{n+k} &= y_{n+k} \\ \begin{pmatrix} X_n \\ X_k \end{pmatrix} \beta_{n+k} &= \begin{pmatrix} y_n \\ y_k \end{pmatrix} \end{aligned}$$

- This would be computationally expensive; instead we can obtain the new estimates iteratively based on the old estimates as follows

# Recursive least squares

- We have

$$
\begin{aligned}
X_{n+k}^\top &= (X_n^\top \; X_k^\top) \\
X_{n+k}^\top X_{n+k} &= X_n^\top X_n + X_k^\top X_k
\end{aligned}
$$

- The first term is calculated before; next we have

$$
X_{n+k}^\top y_{n+k} = X_n^\top y_n + X_k^\top y_k
$$

- substituting $X_n^\top X_n$ and multiplying both sides by $(X_{n+k}^\top X_{n+k})^{-1}$, we have

$$
\begin{aligned}
\hat{\beta}_{n+k} &= (X_{n+k}^\top X_{n+k})^{-1}[(X_{n+k}^\top X_{n+k} - X_k^\top X_k)\hat{\beta}_n + X_k^\top y_k] \\
\hat{\beta}_{n+k} &= \hat{\beta}_n + (X_{n+k}^\top X_{n+k})^{-1} X_k^\top (y_k - X_k \hat{\beta}_n)
\end{aligned}
$$

# Recursive least squares

- We can obtain $R_{n+k}^{-1} = (X_{n+k}^{\top} X_{n+k})^{-1}$ using the matrix inversion lemma,

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

- From the previous slide, we have $R_{n+k} = X_{n+k}^{\top} X_{n+k} = (A + BCD)$, where

$$\begin{aligned} A &= X_n^{\top} X_n = R_n \\ B &= X_k^{\top} \\ C &= I \\ D &= X_k \end{aligned}$$

- Therefore,

$$R_{n+k}^{-1} = R_n^{-1} - R_n^{-1} X_k^{\top} (I + X_k R_n^{-1} X_k^{\top})^{-1} X_k R_n^{-1}$$

# Weighted least squares

- Recall that we obtained the least squares estimate, $\hat{\beta}$ by minimizing the length of the residual term $\|e\| = (e^\top e)^{1/2}$, which is the Euclidean ($\ell_2$) norm

- In this case, all $e_i$ elements (all observations) contribute equally

- Sometimes, we want to weight $e_i$ differently and minimize the weighted sum of residuals instead, $(e^\top We)^{1/2}$, where $W$ must be positive definite

$$
\begin{aligned}
X^\top WX\hat{\beta} &= X^\top Wy \\
\hat{\beta} &= (X^\top WX)^{-1}X^\top Wy
\end{aligned}
$$

- For example, when $\mathrm{Cov}(y)=\mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$, we can set

$$
W = \mathrm{diag}(1/\sigma_1^2, \ldots, 1/\sigma_n^2)
$$

# Weighted least squares

- In general, given a positive definite matrix $P$, $\|x\|_P = (x^\top P x)^{1/2}$ is called the quadratic norm
- Note that this is the same as $\|P^{1/2}x\|$, i.e., the Euclidean norm after transformation of $x$

# Iterative methods

- When solving $Ax = b$ involves large but sparse matrices, instead of solving the system directly, we can start with an initial guess $x^{(0)}$, and improve the solution iteratively

- One such approach is called the Jacobi iteration

- For the first equation, we have

$$\begin{aligned} a_{11}x_1 + \ldots + a_{1n}x_n &= b_1 \\ x_1 &= \frac{1}{a_{11}}[-(a_{12}x_2 + \ldots + a_{1n}x_n)] + \frac{1}{a_{11}}b_1 \end{aligned}$$

- In general,

$$x_i = [x_i - \frac{1}{a_{ii}}\sum_{j=1}^{n} a_{ij}x_j] + \frac{1}{a_{ii}}b_i$$

# Iterative methods

- In the vector form,

$$x \quad = \quad [I - P^{-1}A]x + P^{-1}b$$

where $P = diag(a_{11}, \ldots, a_{nn})$ is the diagonal part of $A$

- This gives the recipe, called the Jacobi algorithm, for an iterative approach for finding $x$

$$x^{(k+1)} \quad = \quad [I - P^{-1}A]x^{(k)} + P^{-1}b$$

- The Gauss-Seidel method is similar to the Jacobi method, but uses the components of new $x$ (i.e., $x^{(k+1)}$) as soon as they become available

# Iterative methods

- In general, $P$ is called the preconditioner matrix
- $P$ should be close to $A$ but it should be much simpler to work with
- Then, we can come up with an iterative method as follows:

$$\begin{aligned}
Ax &= b \\
Px &= (P - A)x + b \\
x &= (I - P^{-1}A)x + P^{-1}b \\
x^{(k+1)} &= (I - P^{-1}A)x^{(k)} + P^{-1}b
\end{aligned}$$

# Iterative methods

- Note that, the new error, $e^{(k+1)} = x^{(k+1)} - x$ can be written in terms of the pervious error $e^{(k)} = x^{(k)} - x$,

$$e^{(k+1)} = (I - P^{-1}A)e^{(k)} = Me^{(k)}$$

- To ensure the error is shrinking and we are converging to the true solution, we need $|\lambda(M)| < 1$ for every eigenvalue
- $\max|\lambda(M)|$ is called the spectral radius, which determines the convergence rate