1 Singularity and Small Sample Problems

1.1 Singularity Example Setup

Flashcard 1.0: Singularity Example

Front: **Consider binary classification with 4 features (m=4) but only 3 total samples (n=3). What's the problem?**

Back: Example: Class 1: 2 samples, Class 2: 1 sample

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix} \quad (2 \times 4)$$

$$\mathbf{X}_2 = \begin{bmatrix} 9 & 10 & 11 & 12 \end{bmatrix} \quad (1 \times 4)$$

Problem: We have m=4 features but n =2, n =1 samples.

- Maximum possible rank of $\mathbf{S}_1 = \min(n-1, m) = \min(1, 4) = 1$
- Maximum possible rank of $\mathbf{S}_2 = \min(n-1, m) = \min(0, 4) = 0$
- So $\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$ has max rank 1

But needs rank 4 to be invertible!

1.2 Why S_W Becomes Singular

Flashcard 1.0: Rank Deficiency

Front: **Why does S_W become singular when n $<$ m?**

Back: $\mathbf{S}_k = \bar{\mathbf{X}}_k^T \bar{\mathbf{X}}_k$ where $\bar{\mathbf{X}}_k$ is $(n_k \times m)$

- Rank of $\bar{\mathbf{X}}_k$ $\min(n_k, m)$
- After centering, rank $n_k - 1$ (loses 1 degree of freedom)
- So rank$(\mathbf{S}_k)$ $n_k - 1$
- Therefore rank$(\mathbf{S}_W)$ $(n-1) + (n-1) = n - 2$

Condition for invertibility: $n - 2 \geq m$
If $n - 2 < m$, $\mathbf{S}_W$ is rank-deficient and singular.

2 Solutions to Singularity

2.1 Regularization Solution

Flashcard 2.0: Regularization

Front: **How does regularization $\mathbf{S}_W + \lambda\mathbf{I}$ solve the singularity problem?**

Back: Instead of $\mathbf{w} = \mathbf{S}_W^{-1}\mathbf{d}$, use:

$$\mathbf{w} = (\mathbf{S}_W + \lambda\mathbf{I})^{-1}\mathbf{d}$$

where $\lambda > 0$ is a small constant.
Why it works:

1. $\mathbf{S}_W$ is positive semi-definite (may have zero eigenvalues)
2. Adding $\lambda\mathbf{I}$ adds λ to all eigenvalues
3. New matrix has minimum eigenvalue $= \lambda > 0$
4. Now invertible regardless of n and m

Typical λ values: 0.001 to 0.1, or use cross-validation.

2.2 PCA Preprocessing Solution

Flashcard 2.0: PCA Preprocessing

Front: **How does PCA preprocessing help with the n $<$ m problem?**

Back:

1. Apply PCA to the combined data $\mathbf{X} = [\mathbf{X}_1; \mathbf{X}_2]$ (size n×m)
2. Keep only the top k principal components where k n - 2
3. Project all data to k dimensions: $\mathbf{X}' = \mathbf{X}\mathbf{V}_k$ where $\mathbf{V}_k$ is (m×k)
4. Apply LDA to $\mathbf{X}'$ which is (n×k) with k n - 2

Advantage: Reduces dimensionality while preserving variance.
Disadvantage: PCA is unsupervised - may discard discriminative information.

3 Worked Examples and Alternatives

3.1 Complete Worked Example: n < m Case

Flashcard 3.0: Worked Example

Front: **Solve LDA for data with n=3, m=4 using regularization.**

Back: Data: $\mathbf{X}_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \end{bmatrix}$, $\mathbf{X}_2 = \begin{bmatrix} 9 & 8 & 7 & 6 \end{bmatrix}$

1. Means: $\mu_1 = [1.5, 2.5, 3.5, 4.5]$, $\mu_2 = [9, 8, 7, 6]$
2. Center: $\bar{\mathbf{X}}_1 = \begin{bmatrix} -0.5 & -0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$, $\bar{\mathbf{X}}_2 = [0, 0, 0, 0]$
3. $\mathbf{S}_1 = \bar{\mathbf{X}}_1^T \bar{\mathbf{X}}_1 = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$, $\mathbf{S}_2 = \mathbf{0}$
4. $\mathbf{S}_W = \mathbf{S}_1$ **(rank 1, singular!)**
5. Regularize: $\mathbf{S}_W^{reg} = \mathbf{S}_W + 0.1\mathbf{I}_4$
6. $\mathbf{d} = \mu_1 - \mu_2 = [-7.5, -5.5, -3.5, -1.5]^T$
7. $\mathbf{w} = (\mathbf{S}_W^{reg})^{-1}\mathbf{d}$ **(now computable)**

3.2 Alternative: Pseudoinverse Solution

Flashcard 3.0: Pseudoinverse

Front: **What is the pseudoinverse solution for singular $\mathbf{S}_W$?**

Back: Use Moore-Penrose pseudoinverse:

$$\mathbf{w} = \mathbf{S}_W^+ \mathbf{d}$$

where $\mathbf{S}_W^+ = \mathbf{V}\boldsymbol{\Sigma}^+\mathbf{U}^T$ from SVD:

$$\mathbf{S}_W = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

How it works:

1. Compute SVD of $\mathbf{S}_W$
2. $\boldsymbol{\Sigma}^+$ replaces non-zero with $1/$, zero remain 0
3. Gives minimum norm solution

Equivalent to: Regularization with $\to 0$

Numerically more stable than explicit regularization for very small .

4 Practical Guidelines

4.1 Practical Rule of Thumb

Flashcard 4.0: Stability Rule

Front: What's the practical rule for ensuring stable LDA?

Back: **Golden Rule:** Need $n \geq m + 2$ for stable $\mathbf{S}_W^{-1}$

- $n \geq m$ and $n \geq m$ ideally
- At minimum: $n + n \geq m + 2$

If rule violated:

1. **n slightly < m:** Use regularization (= 0.01 to 0.1)
2. **n m:** Use PCA first to reduce to k n-2 dimensions
3. **Features correlated:** Even with n > m, $\mathbf{S}_W$ can be near-singular if features are linearly dependent

Check condition number: $\kappa(\mathbf{S}_W) = \frac{\sigma_{max}}{\sigma_{min}}$. If > 10 , use regularization.

4.2 Implementation Check

Flashcard 4.0: Numerical Checks

Front: What numerical checks should you implement for LDA?

Back:

```
# Pseudo-code for robust LDA
def robust_LDA(X1, X2, lambda_reg=0.01):
    n1, m = X1.shape
    n2, _ = X2.shape

    # Check dimensions
    if n1 + n2 < m + 2:
        print("Warning: n < m+2. Using regularization.")

    # Compute S_W
    S1 = (X1 - X1.mean(axis=0)).T @ (X1 - X1.mean(axis=0))
    S2 = (X2 - X2.mean(axis=0)).T @ (X2 - X2.mean(axis=0))
    S_W = S1 + S2

    # Check condition number
    cond_num = np.linalg.cond(S_W)
    if cond_num > 1e6:
        print(f"Condition number {cond_num:.1e} too high. Regularizing.")
        S_W_reg = S_W + lambda_reg * np.eye(m)
        w = np.linalg.solve(S_W_reg, mu1 - mu2)
    else:
        w = np.linalg.solve(S_W, mu1 - mu2)

    return w / np.linalg.norm(w)
```