Machine Learning for Genomics

Mathematical Foundations: Linear Algebra for Genomics Applications

Sarwan Ali

Department of Computer Science
Georgia State University

Today's Learning Journey

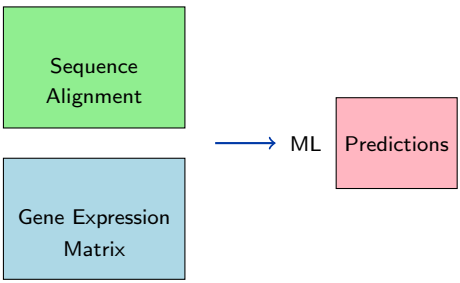
Why Linear Algebra in Genomics?

Genomics generates massive data:

- DNA sequences: 3.2 billion base pairs in human genome
- Gene expression: 20,000+ genes across tissues
- Protein interactions: Millions of potential pairs
- Population genetics: Thousands of individuals

Linear algebra provides:

- Efficient computation on large matrices
- Dimensionality reduction techniques
- Pattern recognition in high-dimensional data



Vectors in Genomics

Gene Expression Vector:

$$\mathbf{g} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix}$$

where g_i represents expression level of gene i

DNA Sequence as Vector:

$$\text{ATCG} \rightarrow \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad (1)$$

One-hot encoding: A=1000, T=0100, C=0010, G=0001

k-mer Frequency Vector:

$$\mathbf{f} = \begin{pmatrix} f_{AA} \\ f_{AT} \\ f_{AC} \\ \vdots \\ f_{GG} \end{pmatrix}$$

Frequency of each k-mer in sequence

Vector Operations in Genomics

Similarity between gene expression profiles:

$$\text{Cosine Similarity} = \frac{\mathbf{g_1} \cdot \mathbf{g_2}}{||\mathbf{g_1}|| \cdot ||\mathbf{g_2}||} = \frac{\sum_{i=1}^{n} g_{1i} g_{2i}}{\sqrt{\sum_{i=1}^{n} g_{1i}^2} \sqrt{\sum_{i=1}^{n} g_{2i}^2}}$$

Euclidean Distance:

$$d(\mathbf{g_1}, \mathbf{g_2}) = ||\mathbf{g_1} - \mathbf{g_2}|| = \sqrt{\sum_{i=1}^{n} (g_{1i} - g_{2i})^2}$$

Application

Used in clustering similar cell types, identifying co-expressed genes, and measuring evolutionary distance between species.

Gene Expression Matrix

Standard Form:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

- **Rows**: n samples (patients, conditions, time points)
- **Columns**: p genes/features
- **Entry** x_{ij}: Expression level of gene j in sample i

Typical dimensions:

- Microarray: $n \approx 100$, $p \approx 20,000$
- RNA-seq: $n \approx 1,000$, $p \approx 60,000$
- Single-cell RNA-seq: $n \approx 10,000$, $p \approx 30,000$

Sequence Alignment Matrices

Substitution Matrix (e.g., BLOSUM62):

$$\mathbf{S} = \begin{pmatrix} s_{AA} & s_{AR} & \cdots & s_{AY} \\ s_{RA} & s_{RR} & \cdots & s_{RY} \\ \vdots & \vdots & \ddots & \vdots \\ s_{YA} & s_{YR} & \cdots & s_{YY} \end{pmatrix}$$

Dynamic Programming Matrix:

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) & \text{(match/mismatch)} \\ F(i-1,j) + d & \text{(deletion)} \\ F(i,j-1) + d & \text{(insertion)} \end{cases} \tag{2}$$

where $s(x_i, y_j)$ is the substitution score and d is the gap penalty.

Covariance and Correlation Matrices

Sample Covariance Matrix:

$$\mathbf{C} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x_i} - \bar{\mathbf{x}})(\mathbf{x_i} - \bar{\mathbf{x}})^T$$

Correlation Matrix:

$$\mathbf{R}_{jk} = \frac{\sum_{i=1}^{n} (x_{ij} - \bar{x_j})(x_{ik} - \bar{x_k})}{\sqrt{\sum_{i=1}^{n} (x_{ij} - \bar{x_j})^2} \sqrt{\sum_{i=1}^{n} (x_{ik} - \bar{x_k})^2}}$$

Applications:

- Gene co-expression networks
- Linkage disequilibrium
- Pathway analysis

Properties:

- Symmetric: $\mathbf{R} = \mathbf{R}^T$
- Diagonal $= 1$
- Positive semi-definite

Eigendecomposition Fundamentals

Definition: For square matrix $\mathbf{A}$, if $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ for non-zero $\mathbf{v}$:

- λ is an eigenvalue
- $\mathbf{v}$ is the corresponding eigenvector

Characteristic Equation:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

Eigendecomposition:

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$$

where $\mathbf{Q}$ contains eigenvectors and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$

Key Property

For symmetric matrices: $\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ (orthogonal eigenvectors)

Principal Component Analysis (PCA)

Goal: Find directions of maximum variance in gene expression data

Steps:

1. Center the data: $\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}\bar{\mathbf{x}}^T$
2. Compute covariance: $\mathbf{C} = \frac{1}{n-1}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$
3. Eigendecomposition: $\mathbf{C} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$
4. Project data: $\mathbf{Y} = \tilde{\mathbf{X}}\mathbf{Q}$

Principal Components:

$$PC_k = \sum_{j=1}^{p} q_{jk} x_j$$

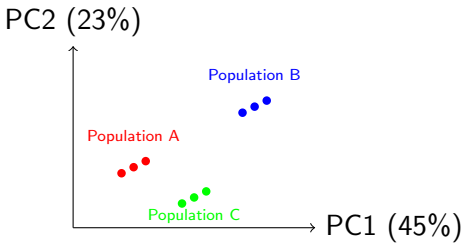
Variance Explained:

$$\text{Proportion of variance by } \text{PC}_k = \frac{\lambda_k}{\sum_{i=1}^{p} \lambda_i}$$

PCA in Genomics Applications

Population Structure:

- PC1, PC2 often correlate with geographic origin
- Identify population stratification
- Control for ancestry in GWAS



Gene Expression:

- Reduce 20K genes to top PCs
- Identify expression modules
- Batch effect detection

Interpretation

First few PCs capture major biological signals; later PCs often represent noise or technical artifacts.

SVD: The Swiss Army Knife

Any matrix $\mathbf{X}_{n \times p}$ can be decomposed as:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where:

- $\mathbf{U}_{n \times n}$: Left singular vectors (orthogonal)
- $\mathbf{\Sigma}_{n \times p}$: Diagonal matrix of singular values
- $\mathbf{V}_{p \times p}$: Right singular vectors (orthogonal)

Relationship to Eigendecomposition:

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T \tag{3}$$

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^T\mathbf{U}^T \tag{4}$$

Singular values: $\sigma_i = \sqrt{\lambda_i}$ where λ_i are eigenvalues of $\mathbf{X}^T\mathbf{X}$

SVD Applications in Genomics

1. **Dimensionality Reduction:**

$$\mathbf{X}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$$

Keep only top k singular values

2. **Missing Value Imputation:**
   - Iterative SVD for incomplete gene expression matrices
   - Reconstruct missing entries using low-rank approximation

3. **Batch Effect Removal:**
   - Identify technical variation in top singular vectors
   - Remove batch-associated components

4. **Phylogenetic Analysis:**
   - SVD of sequence alignment matrices
   - Identify evolutionary relationships

Non-negative Matrix Factorization (NMF)

Problem: Decompose $\mathbf{X} \approx \mathbf{WH}$ where $\mathbf{W}, \mathbf{H} \geq 0$

$$\mathbf{X}_{n \times p} \approx \mathbf{W}_{n \times k} \mathbf{H}_{k \times p}$$

Optimization:

$$\min_{\mathbf{W},\mathbf{H} \geq 0} ||\mathbf{X} - \mathbf{WH}||_F^2$$

Genomics Applications:

- **Mutational signatures:** $\mathbf{W}$ = signature strengths, $\mathbf{H}$ = signature patterns
- **Cell type deconvolution:** $\mathbf{W}$ = cell type proportions, $\mathbf{H}$ = cell type profiles
- **Gene modules:** $\mathbf{W}$ = module activities, $\mathbf{H}$ = gene loadings

Advantage

Non-negativity constraint leads to parts-based, interpretable decomposition

Independent Component Analysis (ICA)

Goal: Find statistically independent components

$$\mathbf{X} = \mathbf{AS}$$

where $\mathbf{S}$ contains independent components and $\mathbf{A}$ is the mixing matrix.

ICA vs PCA:

- **PCA:** Maximizes variance (uncorrelated components)
- **ICA:** Maximizes statistical independence

Genomics Applications:

- Separate overlapping biological processes
- Identify regulatory modules
- Remove technical artifacts
- Discover hidden factors in expression data

Solving Linear Systems

General form: $\mathbf{Ax} = \mathbf{b}$

Methods:

1. **Direct:** $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ (when $\mathbf{A}$ is invertible)
2. **Least squares:** $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ (overdetermined)
3. **Regularized:** $\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{b}$

Genomics Examples:

- **Gene regulatory networks:** $\dot{\mathbf{x}} = \mathbf{Ax}$
- **Linkage analysis:** Solve for recombination frequencies
- **Quantitative genetics:** $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$

Regularization in High-Dimensional Genomics

The Curse of Dimensionality: $p >> n$ (more genes than samples)

Ridge Regression (L2):

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$$

LASSO (L1):

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1$$

Elastic Net:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda_1||\boldsymbol{\beta}||_1 + \lambda_2||\boldsymbol{\beta}||_2^2$$

Applications

Gene selection, SNP association, expression quantitative trait loci (eQTL) mapping

Big Data Challenges

Scale of Genomic Data:

- Whole genome sequencing: 200 GB per individual
- Population studies: 100,000+ individuals
- Single-cell RNA-seq: 1 million+ cells

Computational Strategies:

1. **Sparse matrices:** Most genomic matrices are sparse
2. **Randomized algorithms:** Approximate SVD, random projection
3. **Iterative methods:** Conjugate gradient, power iteration
4. **Parallel computing:** GPU acceleration, distributed computing

Memory-Efficient Approaches:

- Block-wise processing
- Out-of-core algorithms
- Compressed representations

Numerical Stability

Common Issues:

- **Ill-conditioned matrices:** High condition number $\kappa(\mathbf{A}) = \frac{\sigma_{\max}}{\sigma_{\min}}$
- **Near-singular matrices:** Multicollinearity in genomic data
- **Floating-point precision:** Accumulated errors in iterative algorithms

Solutions:

- Use SVD instead of eigendecomposition when possible
- Regularization to improve conditioning
- Pivoting in matrix factorizations
- Monitor convergence in iterative methods

Best Practice

Always check the condition number and rank of your matrices before applying linear algebra operations.

Case Study 1: Gene Expression Analysis

Dataset: RNA-seq from 500 cancer patients, 20,000 genes

Analysis Pipeline:

1. **Normalization:** Log-transform and center data
2. **PCA:** Reduce to top 50 components (explaining 80% variance)
3. **Clustering:** K-means on PC space to identify subtypes
4. **Feature selection:** LASSO to identify prognostic genes

Key Linear Algebra Operations:

- Covariance matrix computation: $O(p^2 n)$
- Eigendecomposition: $O(p^3)$
- LASSO optimization: Iterative coordinate descent

Results: Identified 3 cancer subtypes with distinct survival patterns using 47 marker genes.

Case Study 2: Phylogenetic Reconstruction

Problem: Construct evolutionary tree from DNA sequences

Linear Algebra Approach:

1. Create distance matrix $\mathbf{D}$ between all sequence pairs
2. Apply PCA to visualize relationships
3. Use SVD for dimensionality reduction
4. Construct neighbor-joining tree

Distance Calculation:

$$d_{ij} = -\frac{3}{4} \ln \left(1 - \frac{4}{3} p_{ij}\right)$$

where p_{ij} is the proportion of differing sites (Jukes-Cantor model)

Matrix Properties:

- Symmetric: $\mathbf{D} = \mathbf{D}^T$
- Zero diagonal: $d_{ii} = 0$
- Satisfies triangle inequality (approximately)

Case Study 3: GWAS and Population Structure

Challenge: Population stratification can cause false positives

Solution using PCA:

1. Compute genetic relationship matrix: $\mathbf{G} = \frac{\mathbf{X}\mathbf{X}^T}{p}$
2. Eigendecomposition: $\mathbf{G} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$
3. Use top PCs as covariates in association testing

Linear Mixed Model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\epsilon}$$

where $\mathbf{u} \sim N(0, \sigma_g^2 \mathbf{K})$ and $\mathbf{K}$ is the kinship matrix derived from PCs.

Benefits:

- Controls for population structure
- Reduces genomic inflation factor λ
- Increases power to detect true associations

Result: Identified 12 genome-wide significant loci after controlling for the top 10 PCs.

Tensor Decomposition in Multi-way Genomics

Beyond Matrices: Multi-dimensional genomic data
3-way Tensor: Genes $\times$ Samples $\times$ Conditions

$$\mathcal{X} \in \mathbb{R}^{I \times J \times K}$$

CANDECOMP/PARAFAC (CP) Decomposition:

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

Applications:

- Time-course gene expression across multiple conditions
- Multi-tissue eQTL analysis
- 3D genome organization (Hi-C across cell types)
- Pharmacogenomics: Drug $\times$ Gene $\times$ Patient interactions

Advantage

Captures multi-way interactions that matrix methods cannot detect

Graph Laplacian in Network Biology

Gene Regulatory Networks as Graphs:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

where $\mathbf{D}$ is degree matrix and $\mathbf{A}$ is adjacency matrix.

Normalized Laplacian:

$$\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

Spectral Properties:

- Eigenvalues: $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$
- Number of connected components = multiplicity of eigenvalue 0
- λ_2 (Fiedler value) measures connectivity

Applications:

- Community detection in protein interaction networks
- Gene module identification
- Network-based classification

Random Matrix Theory in Genomics

Null Models for High-Dimensional Data:
For random matrix $\mathbf{X}_{n \times p}$ with i.i.d. entries:
Marchenko-Pastur Law: As $n, p \to \infty$ with $p/n \to \gamma$:

$$\rho(\lambda) = \frac{1}{2\pi\gamma\sigma^2} \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{\lambda}$$

where $\lambda_\pm = \sigma^2(1 \pm \sqrt{\gamma})^2$

Tracy-Widom Distribution: Largest eigenvalue distribution

Genomics Applications:

- Determine number of significant PCs in expression data
- Test for population structure in genetic data
- Identify batch effects vs. true biological signals
- Null hypothesis testing in high-dimensional settings

Python Implementation Example

PCA for Gene Expression Data:

```
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
# Load gene expression data (genes x samples)
X = np.loadtxt('expression_data.csv', delimiter=',')
X = X.T  # Transpose to samples x genes
# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Apply PCA
pca = PCA()
X_pca = pca.fit_transform(X_scaled)
# Plot explained variance
plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
plt.plot(range(1, 21), pca.explained_variance_ratio_[:20], 'o-')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance Ratio')
plt.title('Scree Plot')
# Plot first two PCs
plt.subplot(1, 2, 2)
plt.scatter(X_pca[:, 0], X_pca[:, 1])
plt.xlabel(f'PC1 ({pca.explained_variance_ratio_[0]:.1%})')
plt.ylabel(f'PC2 ({pca.explained_variance_ratio_[1]:.1%})')
plt.title('PCA Projection')
plt.show()
```

R Implementation Example

SVD for Missing Value Imputation:

```r
library(bcv)
library(ggplot2)
# Simulate gene expression with missing values
set.seed(123)
n_samples <- 100
n_genes <- 1000
X <- matrix(rnorm(n_samples * n_genes), nrow=n_samples)
# Introduce missing values (10%)
missing_idx <- sample(length(X), 0.1 * length(X))
X_missing <- X
X_missing[missing_idx] <- NA
# Iterative SVD imputation
impute_svd <- function(X, rank=10, max_iter=100, tol=1e-6) {
  # Initialize missing values with column means
  X_imputed <- X
  for(j in 1:ncol(X)) {
    X_imputed[is.na(X[,j]), j] <- mean(X[,j], na.rm=TRUE)}
  for(iter in 1:max_iter) {
    svd_result <- svd(X_imputed)
    X_approx <- svd_result$u[,1:rank] %*%
                diag(svd_result$d[1:rank]) %*%
                t(svd_result$v[,1:rank])
    # Update missing values
    X_new <- X_imputed
    X_new[is.na(X)] <- X_approx[is.na(X)]
    # Check convergence
    if(norm(X_new - X_imputed, 'F') < tol) break
    X_imputed <- X_new}
  return(X_imputed)
}
```

Emerging Trends

1. **Deep Learning Integration:**
   - Variational autoencoders for dimensionality reduction
   - Graph neural networks for biological networks
   - Attention mechanisms for sequence analysis
2. **Multi-omics Integration:**
   - Joint matrix factorization across data types
   - Tensor methods for multi-way omics
   - Network-based integration approaches
3. **Single-cell Technologies:**
   - Handling extreme sparsity and zero-inflation
   - Batch correction in large-scale studies
   - Trajectory inference using manifold learning
4. **Federated Learning:**
   - Privacy-preserving genomic analysis
   - Distributed matrix computations
   - Secure multi-party protocols

Challenges and Opportunities

Current Challenges:

- Scalability: Growing data sizes exceed computational capacity
- Interpretability: Complex models lack biological insight
- Heterogeneity: Batch effects and technical noise
- Integration: Combining diverse data types and scales

Future Opportunities:

- Quantum computing: Exponential speedup for certain linear algebra operations
- Neuromorphic computing: Brain-inspired architectures for genomic analysis
- Edge computing: Real-time genomic analysis in clinical settings
- Explainable AI: Interpretable machine learning for genomics

Key Insight

The future of genomics lies in developing mathematically principled, computationally efficient, and biologically interpretable methods.

Key Takeaways

Linear Algebra is Fundamental:

1. **Vectors and matrices** naturally represent genomic data
2. **Eigendecomposition and SVD** reveal hidden structure
3. **Matrix factorization** enables interpretable analysis
4. **Linear systems** model biological processes

Practical Applications:

- Dimensionality reduction (PCA, SVD)
- Pattern recognition (clustering, classification)
- Network analysis (graph Laplacian)
- Regularized regression (LASSO, Ridge)

Computational Considerations:

- Numerical stability and conditioning
- Scalability for big genomic data
- Memory-efficient algorithms
- Parallel and distributed computing

Next Steps

Recommended Reading:

- Hastie, T. et al. "The Elements of Statistical Learning"
- Gentleman, R. et al. "Bioinformatics and Computational Biology Solutions Using R"
- Ewens, W.J. "Mathematical Population Genetics"

Software Tools:

- **Python:** NumPy, SciPy, scikit-learn, pandas
- **R:** Bioconductor, limma, edgeR, DESeq2
- **Specialized:** PLINK, GCTA, GEMMA for GWAS

Practice Datasets:

- Gene Expression Omnibus (GEO)
- The Cancer Genome Atlas (TCGA)
- 1000 Genomes Project
- UK Biobank

Questions?

 Thank You! 

Contact: sali85@student.gsu.edu