Basics of Linear Algebra for Machine Learning

Code at https://github.com/MaxwellJChen/Basics_for_Linear_Algebra_for_Machine_Learning

Part II: Foundations

1: Introduction to Linear Algebra

- Mathematics of data
- Numerical linear algebra (with computers)
  - GPU = good bc it can do linear algebra calcs fast
  - Implemented by many open source numerical linear algebra libraries (FORTRAN, LAPACK, etc.)
- Important in statistics
- Numerous practical mathematical applications (computer graphics, Fourier series, statistics – LSRL, etc.)

2: Linear Algebra and Machine Learning

- Don't learn LA if you're a beginner
- Reasons to learn:
  - Notation = important to read, write, and execute efficiently in code
  - Necessary for statistics
  - Must know how to factorize matrices
  - Learn least squares regressions (plays a wider role in ML besides LSRL)

3: Examples of LA in ML

- Datasets and data files are matrices (X) and vectors (y)
- Images/photos are matrices
- One-hot encoding
- Linear regression based on matrix factorization
- Regularization (reduce magnitude of model coefficients) minimize vector norm
- Principal component analysis: common way to identify most relevant features in data
- Singular value decomposition based in LA, used in feature selection, visualization, noise reduction, etc.
- Latent semantic analysis: frequency distribution of words, matrix factorization (SVD) to reduce
- Recommender systems use LA (e.g. calculate similarity between predictions and user activity)
- Deep learning: representing algs, made up of LA calcs

Part III: NumPy

4: Introduction to NumPy

5: Index, Slice, and Reshape NumPy Arrays

6: NumPy Array Broadcasting

Part IV: Matrices

7: Vectors and Vector Arithmetic

- Vectors: tuple of one or more values (scalars)
- Direction and magnitude
- Can represent direction in space, but best as a floating point for high dim ML applications
- Dot product: sum of products of paired terms in vectors (a.dot(b)) where a and b are equal length NumPy vectors

8: Vector Norms

- Norms calculate magnitude/length of vector; always positive except for vector of zeros
- L^1 norm: $||v||_1 = |a_1| + |a_2| + |a_3| + \ldots$, sum of abs values of scalars (taxicab or Manhattan norm)
- L^2 norm: $||v||_2 = \sqrt{}(a_1^2 + a_2^2 + a_3^2 + \ldots)$, (Euclidean norm bc distance from origin to vector coord)
    - Most commonly used norm for machine learning applications by far
- L^1 and L^2 used to minimize size of model coefficients for regularization
- Max norm L^{inf} or L^{∞}: $||v||_{inf} = \max |a_1|, |a_2|, |a_3|, \ldots$, scalar with largest magnitude in vector

9: Matrices and Matrix Arithmetic

- 2D array of scalars represented by uppercase letter
    - Index with $a_{i,j}$ where i = row # and j = column #
- Hadamard product (element-wise matrix multiplication): $C = A \circ B$
- Matrix-matrix multiplication (matrix dot product)
    - # of columns in first matrix A = # of rows in second matrix B
    - C(m,k) = A(m,n) * B(n,k)
    - Calculate dot product between each row in A with each column in B
- Matrix-vector multiplication

10: Types of Matrices

- Square: order of matrix = # of rows/columns
- Symmetric: symmetric across main diagonal from top left to bottom right
- Triangular: values only in upper right or lower left (the main diagonal is also nonzero)

- Diagonal: only main diagonal is nonzero
- Identity (I^n): 1s along main diagonal
- Orthogonal (Q): a square matrix whose rows are mutually orthonormal and whose columns are mutually orthonormal (dot products of 0 and lengths of 1)
    - $Q^T * Q = Q * Q^T = I$
    - $Q^T = Q^{-1}$

11: Matrix Operations

- Transpose: flipping across the main diagonal (inverting row and columns)
- Inverse: finds a matrix that when multiplied with given matrix produces I^n
    - Invertible: has an inverse
    - Singular: square matrix that is not invertible
    - Solve systems of matrix equations (e.g., linear regression)
- Trace: sum of values on main diagonal
- Determinant (det(A) or |A|): describes relative geometry of vectors in the matrix, the volume of the box with sides given by rows of A
    - If 0, matrix cannot be inverted
- Rank: estimate of number of linearly independent rows or columns in matrix
    - Number of dimensions spanned by all vectors within a matrix

12: Sparse Matrices

- Mostly zero matrices = sparse (vs. dense matrices)
    - sparsity = # of non-zero elements/total elements
- Problems
    - Space: most large matrices are sparse, wasted space from allocating 32-bit or even 64-bit memory to each 0
    - Increased time complexity working with 0s
- Appears often in data, data prep methods (e.g. one-hot encoding, TF-IDF, and count encoding), and fields like NLP, computer vision, and recommender systems
- Dictionary of keys, list of lists, coordinate list, compressed sparse row/column

13: Tensors and Tensor Arithmetic

- Tensor: an array of numbers arranged on a regular grid with a variable number of axes
    - Generalization of vectors and matrices (n-dimensional array)
    - Widely used in physics, engineering, and ML
- Element-wise addition, subtraction, multiplication, and division
- Tensor product (np.tensordot)
    - Order of product is equal to sum of orders of factors

- Represented by ⊗ symbol

Part V: Factorization

14: Matrix Decompositions

- Matrix decomposition: reduce a matrix into constituent parts (also called factorization, like taking the “factors” of a matrix)
- LU decomposition: decompose *square* matrix into L and U components
    - A = L • U where A is the square matrix to decompose and L (lower triangle matrix) and U (upper triangle matrix)
    - Can fail for specific matrices
    - LUP decomposition: more numerically stable version of LU with partial pivoting
        - A = L • U • P
        - A is reordered to simplify decomposition while P specifies how to reorder to obtain original
    - Solve systems of linear equations, linear regression, or calculate determinant + inverse of a matrix
- QR decomposition: applies to non-square matrice A with shape (m, n)
    - A = Q • R, where Q has shape (m, m) and R (upper triangle matrix) has shape (m, n)
    - Can fail for specific matrices
    - Usually for solving systems of linear equations
- Cholesky decomposition: for positive definite matrices (square, symmetric matrices where all values greater than 0)
    - $A = L \cdot L^T$, where L is a lower triangular matrix or $A = U^T \cdot U$ where U is an upper triangular matrix
    - Least squares linear regression, simulation, optimization
    - For symmetric matrices, ~2x as fast as LU

15: Eigendecomposition

- Decompose a square matrix in to eigenvectors and eigenvalues
- A vector is an eigenvector of a matrix if $A \cdot v = \lambda \cdot v$
    - A is the vector being decomposed, v is the eigenvector, λ is the eigenvalue scalar
    - v does not change direction when multiplied by A but may change length by eigenvalue
- Matrix can have one eigenvector + eigenvalue for each dimension, some square matrices cannot be decomposed (may need complex numbers)
- $A = Q \cdot \Lambda \cdot Q^T$
    - Q is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues
- Similar to decomposing an integer into prime factors
- Simplify more complex matrix operations (e.g., compute power of matrix), PCA

- Eigenvector: unit vector or right vector (column vector)
- Eigenvalue: coefficients applied to eigenvector that give them length/magnitude
    - Only positive eigenvalues = positive definite matrix
    - Only negative eigenvalues = negative definite matrix

16: Singular Value Decomposition

- Works with all matrices (but could result in complex numbers)
- Compressing, denoising, data reduction, other matrix calculations (inverse), LSLR
- Matrix decomposition method for reducing a matrix to constituents parts to make certain matrix calculations simpler
- $A = U \bullet \Sigma \bullet V^T$, where A has shape (m, n), U has shape (m, m), ∑ is an (m, n) diagonal matrix, and V^T is (n, n)
- Diagonal values in ∑ are singular values of A, columns of U are left-singular vectors of A, and columns of V are right-singular vectors of A
    - When m ≠ n, then ∑ with have 0s in the lower rows
- Pseudoinverse (Moore-Penrose Inverse): generalized inverse for when matrix is not square
    - $A^+ = V \bullet D^+ \bullet U^T$ where A^+ is the pseudoinverse of A and D^+ is the pseudoinverse of the diagonal matrix ∑
    - V, D^+, and U^T can be obtained from SVD: $A = U \bullet \Sigma \bullet V^T$
    - Solve linear regression equations
- Dimensionality reduction
    - Use SVD and select the most significant (largest) values in ∑ and V^T
    - $B = U \bullet \Sigma_k \bullet V^T{}_k$, where B is an approximation of the original matrix A
    - Used in NLP on matrices of word occurrences – Latent Semantic Analysis/Latent Semantic Indexing
    - $T = U \bullet \Sigma_k$ or $T = A \bullet V^T{}_k$ where T is a dense summary of the matrix

Part VI: Statistics

17: Introduction to Multivariate Statistics

- Expected value: E[x] – the average of a random variable
    - $E[x] = \sum x_1 * p_1 + x_2 * p_2 + x_3 * p_3 + \ldots + x_n * p_n$
- Mean: average value calculated from a sample
    - $\mu = P(x) * \sum x$ where P(x) is the calculated probability for each value
    - x-bar when calculated for a specific variable x
- Variance: how much a random variable X changes from its mean
    - $Var[X] = E[(X - E[X])^2]$ – the expected value/average of the squared difference
    - $Var[X] = \sum(x_1 - E[X])^2 * p_1 + (x_2 - E[X])^2 * p_2 + (x_3 - E[X])^2 * p_3 + \ldots + (x_n - E[X])^2 * p_n$
    - For a single sample, $\sigma^2 = 1/(n-1) * \sum(x_i - \mu)^2$ – divide by n-1 to account for bias

- Standard deviation: $s = \sqrt{(\sigma^2)}$, divide by n-1 as well for unbiased sample estimate
- Covariance: joint probability for two random variables
    - $cov(X, Y) = E[(X - E[X]) * (Y - E[Y])]$
    - $cov(X, Y) = 1/n * \sum(x - E[X]) * (y - E[Y])$
    - Sample covariance: $cov(X, Y) = 1/(n-1) * \sum(x - E[X]) * (y - E[Y])$
    - Magnitude difficult to interpret
    - Positive when samples increase together and negative when samples do not
    - 0 when both variables are completely independent
    - Can be formatted in a square covariance matrix
- Correlation: normalized covariance
    - $r = cov(X, Y)/(s_X * s_Y)$ – r is the correlation coefficient or Pearson correlation coefficient
    - Can be returned in correlation matrix
    - From -1 to 1
- Covariance matrix: square and symmetric matrix describing covariance between ≥2 random variables
    - Diagonal: variances of each random variable
    - $\sum = E[(X - E[X] * (Y - E[Y])]$, $\sum_{i, j} = cov(X_i, X_j)$
    - Key component in principal component analysis (PCA)
- Multivariate analysis: intersection of linear algebra and statistics

18: Principal Component Analysis

- PCA: method for reducing dimensionality, projection where m-columns (features) are projected into a space with ≤m columns while retaining the essence of the original
- Steps (covariance method)
    - Original matrix A
    - Calculate mean of each column – M
    - Center each column at 0 by subtracting the mean column value C = M - A
    - Calculate covariance matrix between columns V = cov(C.T)
    - Find eigenvalues and eigenvector from the covariance matrix
    - Select *k* eigenvectors (principal components)/eigenvalues (singular values) based on which eigenvalues have the largest values
        - Similarly large eigenvalues = data is already reasonably compressed
        - Small eigenvalue = value that can be discorded
    - Project the original, centered data into the subspace – vectors.T • C.T

19: Linear Regression

- Method to model relationship between two scalar values: input x and output y
    - y is a linear fx/weighted sum of x where $y = b_0 + b_1*x_1$

- Multivariate linear regression: $y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots$
- Find coefficients b that minimize the error in the prediction for y
- Matrix formulation: y = X * b
- $$X = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \\ x_{4,1} & x_{4,2} & x_{4,3} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$
  - Columns are features in X and rows are examples
  - Multiple possible values for coefficients b, but always some error
  - Find solution that minimizes the squared error (linear least squares)
- Error: $||X \bullet b - y||^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{i,j} (b_j - y_i)^2$ or $X^T \bullet X \bullet b = X^T \bullet y$
  - $b = (X^T \bullet X)^{-1} \bullet X^T \bullet y$ – the normal equation
- Solving methods
  - Use normal equation and inverses to directly calculate b – computationally expensive and numerically unstable
  - Use QR decomposition with $b = R^{-1} \bullet Q^T \bullet y$ – more computationally efficient and numerically stable, but does not work for all matrices
    - $A = Q \bullet R$
  - SVD and pseudoinverse – more stable (all matrices have SVD decomposition) and favored
    - $b = X^+ \bullet y$ where pseudoinverse $X^+ = U \bullet D^+ \bullet V^T$
    - Standard approach