Applied Mathematics 120:
Applied Linear Algebra and Big Data

Detailed syllabus (this document, most recent version available here)

FAS course web page: https://canvas.harvard.edu/courses/174896 *(Spring 2027)*

Last updated: Monday 8th June, 2026, 18:09.

1 Administrative

Instructor: Eli Tziperman (eli@seas.harvard.edu); Head TF: Elle Weeks. **TFs:** please see Canvas. Feel free to email or visit us with any questions.

Day, time & location: Tue, Thu 1:30–2:45, Jefferson 250.

Office hours: Each of the teaching staff will hold weekly office hours; see course web page for times & place. Eli's office: 24 Oxford, museum building, 4th floor, room 456.

Course resources:

1. *Course notes*: see the Dropbox link and password on the Canvas course page.
2. *Supplementary materials*, Matlab/Python demos, HW starter codes: (a) Dropbox link on Canvas; **or** (b) Harvard link, (requires VPN outside campus)
3. The detailed syllabus (this document) lists material used for each lecture.
4. FAQ: advice and answers to questions from previous years.
5. Review problems w/detailed solutions, distinct from past quizzes and final.
6. Practice quizzes and finals from previous years on Canvas/Files.

Course materials are the property of the instructors or other copyright holders, are provided for your personal use, and may not be distributed or posted on any websites.

Prerequisites: Applied/Math 21a and 21b or equivalent; CS50, APM10, or equivalent.

Computer Skills: Programming experience is expected, and homework assignments involve significant code writing; you may use either Matlab or Python.
Using Matlab: install from FAS downloads.
Using Python: Install anaconda for Python 3.12. Course demos have been tested by running them from a terminal ($ python demo.py). Python is a great way to go and is fully supported in APM120, yet some of its packages are a bit more complicated to install and require some Googling and a hacker spirit... :-)

Sections/ weekly HW help sessions: Monday 5–7 pm, or as advertised, in the EPS faculty lounge on the 4th floor of Hoffman, 20 Oxford St. Please come to work on the homework assignments, ask questions, and offer help to others.

Homework assigned every Tuesday (Canvas/Files/), due the following Tuesday, 1 pm. Practicing the lecture material via the weekly HW assignments is the only way to become comfortable with the subjects covered. Solutions are released weekly.
• **Course forum:** Please post questions regarding HW/quizzes/lectures to the course *Ed* forums (edstem.org), rather than emailing the teaching staff. You are very welcome to respond to other students' questions.
• Electronic homework submission via www.gradescope.com: Your submission,

including code and figures, should not exceed 20 Mb or 30 pages. It may be typeset or carefully scanned but must be clear, easily legible, and correctly rotated. Upload a single PDF and tag which pages correspond to which question; see tutorial video. Unacceptable scans could be rejected or result in a 15% penalty. Late policy: Each student is allowed two up to 48-hour late submissions in addition to the dropped HW implied by the grading scheme below. A reduction of 2% per minute after the due time will be applied for other submissions or beyond the 48 hours.

Quizzes: Two evening quizzes,

1. TBA
2. TBA

Contact Eli when these are announced—about ten days before these dates—if you have any conflicts.

In-class mini-quizzes: Answer multiple-choice in-class questions at pollev.com/apm120, after a discussion among the students, with repeated opportunities to vote. For credit, answer correctly 75% of the mini-quizzes for each of the periods before and after spring break; there are 1–3 mini-quizzes per class. You need to let us know during the week of a given class of any medically or otherwise justified absence.

Regrading: Homework and quiz grades are posted to Canvas; please come to Eli's office hours within 7 days of the release of grades if you see a problem. Please approach (visit) Eli rather than the TFs with any grading issues.

Grading:

$$\begin{aligned}\text{HW} &= \min\Big(100,\ \text{mean}\{\text{homework assignments, ignoring lowest grade}\}\Big)\\ \text{mini-quizzes} &= 50 \text{ if answered correctly 75\% of in-class mini-quizzes until spring break, 0 otherwise; 50 more for classes after spring break}\\ \text{Course grade} &= \min\Big(100, 0.3\times\text{HW}+0.15\times\min(100,\text{quiz1})+0.15\times\min(100,\text{quiz2})\\ &\quad + 0.30\times\min(100,\text{final})+0.1\times\text{mini-quizzes}\Big)\end{aligned}$$

The total maximum # of points is 100. Then, A: ≥ 94, A−: ≥ 89, B+: ≥ 83, B: $\geq 75\ldots$ The course may be taken pass/fail only in unusual circumstances and with instructor approval during the first week of classes.

Academic Integrity and Collaboration Policy: We strongly encourage you to discuss and work on homework problems with other students and with the teaching staff. However, after discussions with peers or consulting AI tools, you need to work through the problems yourself, and any answers you submit for evaluation should be the result of your own efforts, reflect your understanding, and be written in your words. In the case of assignments requiring programming, you need to write and use your code; code sharing is not allowed. You must appropriately cite any books, articles, websites, lectures, AI tools, etc. used and explain how and why you used each such source.

Optional reading: [**Str:**] Strang, G, *Linear Algebra and its Applications, 4th ed., 2006* (see also this&this); [**MMD:**] Leskovec, Rajaraman and Ullman, "*Mining of Massive Datasets*"; [**Nielsen:**] Michael Nielsen, "*Neural networks and deep learning*".

Contents

2 Outline

Topics in linear algebra that frequently arise in applications, especially in the analysis of large data sets: linear equations, eigenvalue problems, linear differential equations, principal component analysis, singular value decomposition; data mining and machine learning methods: clustering (unsupervised learning) and classification (supervised) using neural networks and random forests. Examples from physical sciences, biology, climate, commerce, the internet, image processing, and more will be given. The approach is application-motivated, focusing on an intuitive understanding of the algorithms behind these methods obtained by analyzing small data sets. This course may be used to satisfy the QRD requirement.

Please see here for a presentation with a review of example applications.

3 Syllabus

Follow links to see the source material and Matlab/Python demo programs used for each lecture under the appropriate section of the course downloads web page. Grayed subjects below are not covered this year.

1. Introduction, overview. Sources.
   We'll discuss some logistics, the course requirements, provide an overview of the course, what to expect and what not to expect (presentation).

2. Linear equations. Sources. Notes: chapter 2.

   (a) Notation

(b) Motivation: Matrices and linear equations arise in the analysis of CT scans, electrical networks, chemical reactions, large ones arise in network analysis, Leontief economic models, numerical finite-difference solution of PDEs, and more.

(c) Example applications of linear equations: medical tomography, which may lead to either under- or over-determined systems (notes §2.1.1); economic input-output Leontief models (notes §2.1.2). Climate effects on forest fires (multiple linear regression, notes §2.1.3).

(d) Row and column geometric interpretations for linear equations $\mathbf{Ax} = \mathbf{b}$ (notes §2.2).

(e) Solution of large linear systems via direct vs. iterative techniques.

i. Direct method: LU factorization (notes §2.3).

ii. Using LU to efficiently solve for many RHS.

iii. Why pivoting is critical (notes §2.3.4).

iv. Computational cost of Gaussian elimination (notes §2.3.3)

v. Why the LU decomposition algorithm works (notes §2.3.5).

vi. Iterative projection methods: the Kaczmarz method, aka algebraic reconstruction technique (notes §2.6.2).

vii. Bad news: LU factorization of a sparse matrix is dense, so it might be best to use an iterative method to solve the corresponding linear system of equations. This may be alleviated, to some degree, using complete (full) pivoting.

viii. Iterative relaxation methods: (notes §2.6.1) Jacobi, Gauss-Seidel.

ix. Another relaxation method: SOR. Generally, iterative methods converge best for matrices that are either diagonally dominant, or symmetric positive definite, or both.

(f) Does a solution exist, and is it sensitive to noise/round-off error? (notes §2.4)

(g) Dealing with huge systems:

i. Sparse matrices (notes §2.5.1).

ii. Google's MapReduce (Hadoop) algorithm: general idea and word-count example (notes §2.5.3). Additional Examples: (i) finding mutual friends on Facebook; (ii) calculating mean daily flight delays; (iii) matrix-matrix multiplication using one MapReduce step. The more efficient two-step approach to matrix multiplication (**MMD**§2.3.9). The story behind the development of MapReduce: here.

3. Eigenvalues, eigenvectors. Sources. Notes: chapter 3.

(a) Motivation: Google's PageRank, partitioning (clustering) of graphs/networks, ordinary differential equations, and explosive development of weather systems.

(b) Reminder: Eigenvalue problems $\mathsf{A}\mathbf{x} = \lambda\mathbf{x}$, finding eigenvalues through $\det(\mathsf{A} - \lambda\mathsf{I}) = 0$, then finding eigenvectors by solving $(\mathsf{A} - \lambda_i\mathsf{I})\mathbf{e}_i = 0$ (**Str**§5.1). Similarity transformation $\mathsf{S}^{-1}\mathsf{A}\mathsf{S}$ and diagonalization of matrices with a full set of eigenvectors (**Str**§5.2) and of symmetric matrices (**Str**§5.6, 5S).

(c) Google's PageRank algorithm (notes §3.2). First, Google vs. BMW: here.

(d) Power methods (notes §3.3)

i. Calculating the largest eigenvalue/vector using the power method (notes §3.3.2).

ii. Calculating the largest p eigenvalues/vectors of a normal matrix with orthogonal eigenvectors using the block power method (notes §3.3.3).

iii. Inverse power method: calculating the smallest eigenvalue/eigenvector (notes §3.3.4).

iv. The shifted inverse power method (notes §3.3.5)

v. Iterative projection methods, AKA Algebraic reconstruction technique or the Kaczmarz method (notes §2.6.2).

vi. Deflation methods for calculating the next eigenvalues/vectors (notes §3.3.6): Hotelling deflation for orthogonal eigenvectors; Wielandt deflation for the more general case (eigenvalues only). Extracting eigenvectors as well, and the actual deflation of the matrix.

(e) Spectral clustering (partitioning) of networks via eigenvectors of corresponding Laplacian matrices (notes §3.4).

(f) Generalized eigenvalue problems $\mathsf{A}\mathbf{x} = \lambda\mathsf{B}\mathbf{x}$ (notes §3.7).

(g) Linear constant coefficient ordinary differential equations and matrix exponentiation (notes §3.5). Motivation: **Strogatz**'s Romeo and Juliette and dynamic Leontief input-output economic models (notes §3.5.1). Higher-order constant coefficient linear ODEs (notes §3.5.3). Eigenvalues and stability (notes §3.5.5), phase space plots.

(h) Dramatic surprises on the path to tranquility: Non-normal matrices, transient amplification, and optimal initial conditions (notes §3.6).

(i) Jordan form and generalized eigenvectors: when a straightforward diagonalization using standard eigenvectors doesn't work because they are not independent (notes §3.8)

i. A simple example. Definition and statement of the ability to always transform to a Jordan normal form.

ii. An example second-order ODE equivalent to a first-order set in Jordan form that leads to a resonant solution, see notes.

iii. How to find the Jordan form using the matrix of generalized eigenvalues

iv. Extreme sensitivity of the Jordan form to round-off error.

v. (Time permitting:) Proof by recursion that a Jordan form can always be found is also in **Str** Appendix B.

4. Principal component analysis, Singular Value Decomposition. Sources. Notes: chapter 4.

   (a) Motivation: PCA is used to understand patterns in large data sets such as stocks or El Niño. There are numerous applications of SVD, from image compression and face recognition to comparing the structure of folded proteins and solving linear equations with more unknowns than equations.

   (b) Principal Component Analysis (PCA; also known as Factor Analysis or Empirical Orthogonal Functions): calculation from covariance matrix (notes §4.1).

   (c) Singular Value Decomposition (SVD): statement, examples, and a guideline for calculating the SVD decomposition, $\mathsf{X} = \mathsf{U}\Sigma\mathsf{V}^T$ (notes §4.2.1).

   (d) Geometric interpretation of SVD for the particular case of a real square matrix with a positive determinant (notes §4.2.2; see animation and caption from Wikipedia by Kieff, with some more details here.)

   (e) SVD applications

      i. Image compression, low-rank approximation, variance explained. (Notes §4.3).

      ii. Numerical rank of a matrix (notes §4.5.1).

      iii. Polar decomposition (notes §4.4).

         A. The geometric interpretation of polar decomposition.

         B. Computer animation (notes §4.4.1)

         C. The polar-decomposition-based Kabsch Algorithm for comparing protein structures using the root-mean-square deviation method (notes §4.4.2, (Kavraki, 2007, p 1-5)).

         D. Proof that polar decomposition of the correlation matrix between molecule coordinates is indeed the optimal rotation matrix (notes §4.4.2.)

      iv. **Linear equations again:** when the number of unknowns is different from the number of equations:

         A. Overdetermined systems: more equations than unknown and least squares (notes §4.5.2).

         B. Using QR decomposition for an efficient solution of least-square problems (notes §4.5.3)

         C. Iterative solution of overdetermined equations (notes §4.5.4).

         D. Under-determined systems, more unknowns than equations: pseudo-inverse solution using SVD (notes §4.5.5).

         E. Rank-deficient over-determined or under-determined systems: $r < \min(N, M)$ (notes §4.5.6).

         F. A review of all types of linear equations.

v. PCA using SVD: (notes §4.6.1).

vi. Multivariate Principal Component Analysis (notes §4.6.2) and Maximum Covariance Analysis (MCA, notes § 4.6.3): analysis of two co-varying data sets. E.g., M stocks from NY and L stocks from Tokyo, both given for N days.

vii. The Netflix challenge part I: latent factor models and SVD. First, highlighted text and Figs. 1 and 2 on pp 43-44 of Koren et al. (2009); then, notes §4.7; finally, example code, `SVD_applications_Netflix.m/py`. [Optional: information on the fuller procedure in highlighted parts of section 6.1 of Vozalis and Margaritis (2006) available here; Instead of Eqn (4) in Vozalis, let the predicted rating of movie a by user j be $pr_{aj} = \sum_{i=1}^{n} sim_{ji}(rr_{ai} + \bar{r}_a)/(\sum_{i=1}^{n} |sim_{ji}|)$, where sim_{ji} is the similarity between the ratings of movies i and j by all users, and the sum is over movies].

5. Similar items and frequent patterns. Sources. Notes: chapter 5.

(a) Motivation for similar items: face recognition, fingerprint recognition, comparing texts to find plagiarism, Netflix movie ratings. (**MMD**§3)

(b) Similar items:

i. Jaccard Similarity index (**MMD**§3.1.1 p 74; demo, `Jaccard_examples.m/py`, for logical variables, numbers, text files).

ii. Converting text data to numbers: Shingles, k-shingles, hashing, sets of hashes (**MMD**§3.2 p 77-80; section 1 of notes and corresponding Matlab/Python demo of an oversimplified hash function; another demo, `crc32_demo.m/py`, for the Cyclic Redundancy Check (crc32) hash function)

iii. Matrix representation of sets (**MMD**§3.3.1 p 81)

iv. (Time permitting:) MinHash algorithm for comparing sets (**MMD**§3.3 p 80-86, and section 2 of notes with summary of MinHash steps)

A. Minhashing: creating a similarity-conserving signature matrix that is much smaller than the original data matrix, and that allows for efficient comparison of sets. A signature matrix is based on a set of random permutations of the rows of the data matrix (**MMD**§3.3.2,§3.3.4 p 81-83)

B. "Proof" that the probability of having similar MinHash signatures of two sets is equal to the Jaccard similarity of the two sets (**MMD**§3.3.3 p 82-83)

C. MinHash signature estimated using a set of random hash functions acting on the data matrix (**MMD**§3.3.5 p 83-86)

D. Additional resources: code, `MinHash_and_signature_matrix_example.m/py`, for calculating signature matrix and using it to estimate Jaccard similarity; A more elaborate example Python code by Chris McCormick, run using `spyder`)

E. Locality-Sensitive Hashing (LSH, **MMD**§3.4-3.8)

(c) (Time permitting:) Motivation for frequent patterns: market basket analysis: hot dogs and mustard, diapers and beer; frequently combined Internet searches: Brad and Angelina; medical diagnosis: biomarkers in blood samples and diseases; detecting plagiarism. (**MMD**§6)

(d) Frequent patterns and association rules.

i. Mining frequent patterns (and association rule learning): support for set I (number of baskets for which I is a subset); I is frequent if its support is larger than some threshold support s; (**MMD**§6.1 p 201-206)

ii. Association rules $I \to j$ between a set I and an item j; confidence (fraction of baskets with I that also contain j) and interest (difference between confidence in $I \to j$ and fraction of baskets that contain j); (**MMD**§6.1.3-6.1.4)

iii. Apriori algorithm: (**MMD**§6.2, highlighted parts on p 209-217)

A. Baskets as sets of numbers (**MMD**§6.2.1 p 209)

B. Monotonicity of itemsets (**MMD**§6.2.3 p 212)

C. A-priory first pass; renumbering of relevant itemsets between passes; and second pass to identify frequent pairs (**MMD**§6.2.5; a simple code, `apriori_example.m/py`)

D. Beyond frequent pairs: larger frequent itemsets (**MMD**§6.2.6)

E. Example of finding association rules via A-priori algorithm, Matlab code by Narine Manukyan, run using `demoAssociationAnalysis;`

6. Unsupervised learning: cluster analysis. Sources. Notes: chapter 6.

(a) Motivation: Archaeology/Anthropology; Genetics; TV marketing; Criminology; Medical imaging; Internet/social networks; Internet search results; Weather and climate.

(b) Overview: Two main approaches to clustering: hierarchical (each point is an initial cluster, then clusters are merged to form larger ones) and point-assignment (starting with points that are cluster representatives, clusteroids, and then adding other points one by one). Other considerations: Euclidean vs. non-Euclidean and large vs. small memory requirements (**MMD**§7.1.2, p 243).

(c) Distances/ metrics (notes §6.2)

i. Requirements from a distance function (notes §6.2; **MMD**§3.5.1, p92-93.

ii. Examples of distance functions (notes §6.2); **MMD**§3.5, p 93–97): Euclidean (L_2 distance), L_r distance, Manhattan (L_1), maximum (L_∞), Hamming distance between two strings of equal length or between vectors of Booleans or other vectors, cosine (angle between vectors), Jaccard distance (one minus Jaccard similarity), and edit. Noting that "average" distance does not necessarily exist in non-Euclidean spaces.

(d) Curse of dimensionality: problems with Euclidean distance measures in high dimensions, where random vectors tend to be far from each other and perpendicular to each other, making clustering difficult (notes §6.3; **MMD**§7.1.3 p 244–245)

(e) Hierarchical clustering (notes §6.4; **MMD**§7.2.1, Figs 7.2, 7.3, 7.4, 7.5, 7.6). Cluster quality, merging and stopping criteria, dendrogram, elbow plot, efficiency.

(f) Hierarchical Clustering in non-Euclidean spaces using clusteroids (notes §6.4.3; **MMD**§7.2.4, pp. 252–253).

(g) K-means algorithm (notes §6.5): a point-assignment/centroid-based clustering method.

(h) Self-organizing maps (AKA "Kohonen maps"): a machine learning approach to clustering (notes §6.6).

(i) Mahalanobis distance: first for stretched data, diagonal covariance matrix, then non-diagonal, stretched and rotated (notes §6.7).

(j) Spectral clustering (notes §6.8). This approach is related to the problem of network partition using the second eigenvector of the Laplacian matrix covered in notes §3.4.

(k) BFR: Clustering large data sets that cannot be fully contained in memory: BFR algorithm and Summarization (notes §6.9; **MMD**§7.3.4 and 7.3.5, p 257 to the middle of 261).

(l) CURE (Clustering Using REpresentatives), for clusters that have complex shapes, such as concentric rings. This is a point-assignment clustering algorithm, like k-means, not relying on centroids but on a set of representative points for each cluster that can span its complex shape (notes §6.10; **MMD**§7.4, p 262–265).

(m) Outlier/anomaly detection: a brief overview only. Motivation: unusual credit activity as an indication of credit card theft. Detection using statistical methods, e.g., assuming Gaussian distribution.

7. Supervised learning: classification. Sources. Notes: chapter 7.
(We stick to Euclidean distances for now; other options were discussed under cluster analysis).

(a) Motivation: Optical character recognition, handwriting recognition, speech recognition, spam filtering, language translation, sentiment analysis of tweets (e.g., angry/sad/happy/voting preference), Amazon book recommendations, the Netflix challenge, ad blocking on Internet sites, credit scores, predicting loan defaulting, predicting the weather, and mastering the game of Go!

(b) Machine Learning Introduction (notes §7)

(c) Perceptrons:

i. Intro (notes §7.2)
ii. Training (notes §7.2.1), example (notes §7.2.2)
iii. Problems (notes §7.2.3)

(d) Support vector machines (notes §7.3; **MMD**§12.3, p 461–469);
i. Introduction, formulation for separated data sets, formulation for overlapping data sets,
ii. Solution via gradient method, and a numerical Example.

(e) A brief introduction to Multi-Layer Artificial "feed forward" Neural Networks (notes §7.4):
i. Motivation: these are based on a powerful extension of the perceptron idea and allow computers to perform image/ voice/ handwriting/ face recognition, as well as Mastering the game of Go.
ii. Introduction: perceptron as a neural network with no hidden layers; failure of perceptron for XOR, and success using one hidden layer and a simple nonlinear activation function; a general one-hidden layer formulation (highlighted parts of the introductory notes by Lee Jacobson)
iii. Details: architecture (including the number of layers, number of nodes in each layer, geometry of connections between nodes); example activation functions: tansig, sigmoid, rectified linear, and softplus; selecting output layer activation function based on the need for (1) regression (linear output layer), (2) a yes or no (sigmoid output layer), (3) a discrete set of labels using a softmax output layer plus Matlab's vec2ind (on-line demo), (Goodfellow et al. (2016), §6.2.2, p 181–187; the activation functions are plotted by `neural_networks00_activation_functions_examples.m/py` and in Goodfellow et al. (2016), §3.10, and Figs. 3.3, 3.4, p 69)
iv. Understanding the internals of Matlab/Python's neural networks.
v. Simple example neural network Matlab/Python example codes for classification and regression.
vi. Back-propagation! (notes §7.4.3) Calculating the cost gradient with respect to weights and biases (**Nielsen**)
A. Cost function definition (**Nielsen**§1, Eqn 6)
B. Gradient descent rule (**Nielsen**§1, eqns 16,17, and following two paragraphs).
C. Back-propagation: basically all of **Nielsen**§2.
vii. Ways to recognize problems and choose parameters for neural networks (notes §7.4.4)
A. Overfitting, how to identify it and how to resolve it. (appropriate section of **Nielsen**§3)
B. Learning slow-down and the improved *cross-entropy cost function* that resolves that (appropriate section of **Nielsen**§3, beginning to two demos after eqn 62. See on-line version of the chapter for the nice demos.)

C. Weight initialization to avoid initial saturation and increase initial learning rate (appropriate section of **Nielsen**§3)

D. Choosing network's hyperparameters: learning rate (which may also vary with epochs), regularization constant, a mini-batch size used the average the gradient before applying steepest descent. Find parameters that lead to *any* learning, and improve from there (appropriate section of **Nielsen**§3)

viii. Convolution Neural networks (CNNs) for image processing.

ix. Autoencoders for clustering, dimension reduction, and noise reduction.

x. Unet!

(f) k-nearest neighbors (k-NN) (notes §7.5)

i. Classification: discrete labels (notes §7.5.1; Mitchell (1997) Fig. 8.1, p 233; **MMD**§12.4, p 472–474 including Fig. 12.21)

ii. Locally weighted kernel regression (notes §7.5.2; Mitchell (1997) §8.3.1 p 237–238)

iii. Using PCA for dimensionality reduction to avoid the curse of dimensionality when looking for nearest neighbors in a high-dimensional space. (Section 2 of notes)

iv. k-NN application: the Netflix challenge part II (presentation by Atul S. Kulkarni, remote and local links).

(g) Decision trees (notes §7.6, Sources):

i. Motivation, example: class selection (§7.6).

ii. Discrete (categorical) labels, the Gini index, and the CART algorithm (§7.6.1).

iii. Continuous attributes, discrete labels (§7.6.2).

iv. Regression trees: continuous labels and variance minimization (§7.6.3).

v. Avoiding overfitting by limiting tree growth, by pruning, and by monitoring error on validation data while growing the tree (§7.6.4). Minimal cost-complexity pruning.

vi. Random forests (§7.6.5).

vii. Information entropy, information gain, ID3 algorithm (§7.6.6).

8. Review. Sources. Notes: chapter 8.

References

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Kavraki, L. E. (2007). Molecular distance measures. OpenStax-CNX module: m11608, Version 1.23: Jun 11, 2007, http://cnx.org/content/m11608/1.23/.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge University Press.

Mitchell, T. (1997). *Machine learning*. McGraw-Hill Science/ Engineering/ Math.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Strang, G. (2006). *Linear algebra and its applications. 4th ed.* Cengage Learning.

Strang, G. (2016). *Introduction to linear algebra. 5th ed.* Wellesley-Cambridge Press.

Strang, G. (2019). *Linear Algebra and Learning from Data*. publisher TBD.

Vozalis, M. G. and Margaritis, K. G. (2006). Applying SVD on generalized item-based filtering. *IJCSA*, 3(3):27–51.