National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics

A Deep Equilibrium Approach for Trainable Nonnegative Matrix Factorization

DIPLOMA THESIS

of

Foivos Ntoulas-Panagiotopoulos

Supervisor: Athanasios Rontogiannis
Associate Professor NTUA

Computer Vision, Speech Communication and Signal Processing Group
Athens, July 2025

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Signals, Control and Robotics
Computer Vision, Speech Communication and Signal Processing Group

A Deep Equilibrium Approach for Trainable Nonnegative Matrix Factorization

DIPLOMA THESIS

of

Foivos Ntoulas-Panagiotopoulos

Supervisor: Athanasios Rontogiannis
Associate Professor NTUA

Approved by the Examining Committee on 18th July, 2025.

.....	.....	.....
Athanasios Rontogiannis	Charalampos Psillakis	Ioannis Kordonis
Associate Professor NTUA	Assistant Professor NTUA	Assistant Professor NTUA

Athens, July 2025

.....
Foivos Ntoulas-Panagiotopoulos
Graduate of Electrical and Computer Engineering NTUA

Abstract

This thesis explores the evolving field of Non-negative Matrix Factorization (NMF). NMF is a powerful technique widely employed in fields such as signal processing and image analysis due to its interpretability and effectiveness in uncovering hidden structures in data. At its core, NMF seeks to approximate a non-negative data matrix as the product of two lower-rank non-negative matrices, thereby enabling a parts-based, interpretable representation of the original data. Typically, one of the factorized matrices acts as a dictionary of basis elements, while the other encodes how these elements are combined to reconstruct the original data. This factorization not only reduces dimensionality but also reveals latent features that can be crucial for tasks such as classification, clustering, and source separation. Recent advancements in NMF have introduced various regularization schemes and deep architectures, further enhancing its performance and adaptability across diverse domains.

However, traditional NMF algorithms encounter significant challenges, including slow convergence rates and limited reconstruction accuracy when dealing with complex, high-dimensional datasets. Algorithm Unrolling is an innovative approach that transforms iterative optimization algorithms into structured deep neural networks by mapping each iteration to a corresponding network layer. This framework has recently been applied to NMF, leading to the development of Deep NMF (DNMF) by unrolling the classical multiplicative update (MU) algorithm into a trainable, layer-wise architecture. The idea behind this method is introducing training to these otherwise "blind" optimization algorithms. This approach significantly improves computational efficiency and maintains interpretability by clearly linking deep architectures to their iterative algorithm counterparts.

In parallel with unrolled networks, Deep Equilibrium Models (DEQs) have emerged as a complementary approach in deep learning, offering a fundamentally different perspective on network design. DEQs represent a recent paradigm shift, where instead of stacking a finite number of layers, the model employs a single implicit layer that computes its output as the fixed point of a parameterized nonlinear transformation. These models leverage weight tying, reusing the same set of parameters across all iterations of the transformation, which enables compact representations and can improve generalization. By directly solving for this equilibrium point, DEQs connect ideas from classical fixed-point iteration with modern deep learning, and support efficient gradient-based training via implicit differentiation.

This thesis introduces both a classical DEQ-NMF approach, a novel equilibrium approximation scheme-based method, and modified versions of these for the supervised NMF framework. The DEQ-NMF models utilize fixed-point solvers for the forward pass and implicit differentiation for the backward pass, enabling constant memory usage regardless of the number of forward iterations. The equilibrium approximation scheme further improves efficiency by estimating the equilibrium in a single step, which allows for explicit gradient computation and leverages modern optimizers, such as Adam, during training. The results demonstrate that the proposed algorithms consistently outperform both conventional NMF methods and existing deep learning approaches within the supervised NMF framework, yielding higher reconstruction accuracy and faster convergence. These findings highlight the strengths of equilibrium-based modeling in supervised NMF, and underscore the potential of these approaches for broader applications across scientific and practical domains.

Keywords. Non-negative Matrix Factorization, Algorithm Unrolling, Weight-Tied Networks, Fixed-Point, Deep Equilibrium Models.

Περίληψη

Η παρούσα διατριβή εξερευνά τον ταχέως εξελισσόμενο τομέα της Μη Αρνητικής Παραγοντοποίησης Μητρώου (Non-negative Matrix Factorization, NMF). Η NMF αποτελεί μια ισχυρή τεχνική που χρησιμοποιείται ευρέως σε πεδία όπως η επεξεργασία σήματος και η ανάλυση εικόνας, χάρη στην ερμηνευσιμότητά της και στην ικανότητά της να αποκαλύπτει κρυμμένες δομές στα δεδομένα. Στον πυρήνα της, η NMF στοχεύει στην προσέγγιση ενός μη αρνητικού μητρώου δεδομένων ως το γινόμενο δύο χαμηλότερης τάξης μη αρνητικών μητρώων, επιτρέποντας έτσι μια αναπαράσταση της αρχικής πληροφορίας βασισμένη σε μέρη, η οποία είναι εύκολα ερμηνεύσιμη. Συνήθως, το ένα από τα παραγοντοποιημένα μητρώα λειτουργεί ως «λεξικό» βασικών στοιχείων, ενώ το άλλο κωδικοποιεί τον τρόπο με τον οποίο αυτά τα στοιχεία συνδυάζονται για την ανακατασκευή των αρχικών δεδομένων. Πρόσφατες εξελίξεις στην NMF έχουν εισαγάγει διάφορα σχήματα κανονικοποίησης και βαθιές αρχιτεκτονικές, βελτιώνοντας περαιτέρω την απόδοσή της και την προσαρμοστικότητά της σε ποικίλα πεδία εφαρμογής.

Ωστόσο, οι παραδοσιακοί αλγόριθμοι NMF αντιμετωπίζουν σημαντικές προκλήσεις, όπως αργούς ρυθμούς σύγκλισης και περιορισμένη ακρίβεια ανακατασκευής όταν εφαρμόζονται σε πολύπλοκα και υψηλών διαστάσεων δεδομένα. Η μέθοδος του Ξετυλίγματος Αλγορίθμου (Algorithm Unrolling) αποτελεί μια καινοτόμο μεθοδολογία που μετασχηματίζει επαναληπτικούς αλγορίθμους βελτιστοποίησης σε βαθιά νευρωνικά δίκτυα, αντιστοιχίζοντας κάθε επανάληψη σε ένα αντίστοιχο επίπεδο (layer) του δικτύου. Το πλαίσιο αυτό έχει πρόσφατα εφαρμοστεί στην NMF, οδηγώντας στην ανάπτυξη της Deep NMF (DNMF), μέσω του ξετυλίγματος του κλασικού αλγορίθμου πολλαπλασιαστικής ενημέρωσης (Multiplicative Update, MU) σε μια αρχιτεκτονική εκπαιδεύσιμου δικτύου. Η βασική ιδέα πίσω από αυτή τη μέθοδο είναι η εισαγωγή εκπαίδευσης σε αλγορίθμους βελτιστοποίησης που κατά τα άλλα είναι «τυφλοί». Αυτή η προσέγγιση βελτιώνει σημαντικά την υπολογιστική απόδοση και διατηρεί την ερμηνευσιμότητα, καθώς η δομή του βαθιού δικτύου συσχετίζεται άμεσα με τον αντίστοιχο αλγόριθμο.

Παράλληλα με τα unrolled δίκτυα, τα Μοντέλα Βαθιάς Ισορροπίας (Deep Equilibrium Models, DEQs) έχουν αναδειχθεί ως μια συμπληρωματική προσέγγιση στο πεδίο της βαθιάς μάθησης, προσφέροντας μια θεμελιωδώς διαφορετική οπτική στο σχεδιασμό των νευρωνικών δικτύων. Τα DEQs αντιπροσωπεύουν μια νέα τεχνική, όπου αντί της στοίβαξης πεπερασμένου αριθμού επιπέδων, το μοντέλο χρησιμοποιεί ένα μόνο implicit επίπεδο που υπολογίζει την έξοδό του ως το Σταθερό Σημείο (Fixed-Point) μιας παραμετροποιημένης μη γραμμικής συνάρτησης. Αυτά τα μοντέλα αξιοποιούν τα Δίκτυα με Κοινά Βάρη (Weight-Tied Networks), επαναχρησιμοποιώντας το ίδιο σύνολο παραμέτρων σε όλες τις επαναλήψεις του μετασχηματισμού, γεγονός που επιτρέπει συμπαγείς αναπαραστάσεις και μπορεί να βελτιώσει τη γενίκευση των μοντέλων. Επιλύοντας για να βρουν άμεσα αυτό το Σταθερό Σημείο, τα DEQs συνδέουν ιδέες από τον κλασικό fixed-point iteration με το σύγχρονο deep learning και υποστηρίζουν αποδοτική εκπαίδευση μέσω του implicit differentiation.

Στο πλαίσιο αυτής της διατριβής παρουσιάζονται τόσο μια κλασική προσέγγιση DEQ-NMF, όσο και μια νέα μέθοδο προσέγγισης του equilibrium, καθώς και τροποποιημένες εκδοχές αυτών για το supervised NMF framework. Τα DEQ-NMF μοντέλα χρησιμοποιούν fixed-point solvers για το forward pass και implicit differentiation για το backward pass, επιτρέποντας σταθερή χρήση μνήμης ανεξαρτήτως του αριθμού επαναλήψεων προς το σημείο ισορροπίας. Τα αποτελέσματα δείχνουν ότι οι προτεινόμενοι αλγόριθμοι υπερτερούν σταθερά τόσο έναντι των κλασικών μεθόδων NMF όσο και των υπαρχόντων δικτύων βαθειάς μάθησης στην επιβλεπόμενη NMF framework, προσφέροντας υψηλότερη ακρίβεια ανακατασκευής και ταχύτερη σύγκλιση.

Λέξεις Κλειδιά. Μη Αρνητική Παραγοντοποίηση Μητρώου (Non-negative Matrix Factorization, NMF), Ξετύλιγμα Αλγορίθμου (Algorithm Unrolling), Δίκτυα με Κοινά Βάρη (Weight-Tied Networks), Σταθερό Σημείο (Fixed-Point), Μοντέλα Βαθιάς Ισορροπίας (Deep Equilibrium Models, DEQs)

Acknowledgments

I am deeply grateful to Professor Athanasios Rontogiannis for his expert guidance, invaluable insights, and unwavering support, which have been instrumental in bringing this thesis to fruition. My heartfelt thanks go to my family for their endless love, patience, and encouragement throughout this academic journey. I also extend my sincere appreciation to my friends, whose intellectual stimulation and steadfast camaraderie have sustained and motivated me along the way.

Foivos Ntoulas–Panagiotopoulos
July 2025

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον καθηγητή μου, κ. Αθανάσιο Ροντογιάννη, για την πολύτιμη καθοδήγηση, τις ανεκτίμητες γνώσεις και την αδιάλειπτη υποστήριξή του, που υπήρξαν καθοριστικές για την ολοκλήρωση αυτής της διπλωματικής εργασίας. Ευχαριστώ επίσης την οικογένειά μου για την απεριόριστη αγάπη, την αμέριστη κατανόηση και την αδιάκοπη ενθάρρυνσή της καθ' όλη τη διάρκεια αυτής της ακαδημαϊκής διαδρομής. Τέλος, εκφράζω την ειλικρινή ευγνωμοσύνη μου στους φίλους μου, οι οποίοι με διανοητική και συναισθηματική υποστήριξη έκαναν αυτή την εμπειρία πραγματικά αξέχαστη.

Φοίβος Ντούλας–Παναγιωτόπουλος
Ιούλιος 2025

Contents

List of Figures

List of Tables

Chapter 1

Εκτεταμένη Περίληψη στα Ελληνικά

1.1 Θεωρητικό Υπόβαθρο

1.1.1 Συμβολισμοί

Χρησιμοποιούμε έντονα κεφαλαία γράμματα $\mathbf{X}$ για να αναπαριστούμε μητρώα και έντονα πεζά γράμματα $\mathbf{x}$ για διανύσματα-στήλες. Συμβολίζουμε με x_i το i-οστό στοιχείο του διανύσματος $\mathbf{x}$, με x_{ij} το στοιχείο στη θέση (i, j) του μητρώου $\mathbf{X}$ και με $\mathbf{x}_i$ τη i-οστή στήλη του $\mathbf{X}$, αντίστοιχα, εκτός από περιπτώσεις όπου διατηρούμε τον αρχικό συμβολισμό από τις αρχικές δημοσιεύσεις και ορίζουμε ρητά τη σημασία κάθε συμβόλου. Τα σύμβολα $\odot$ και $\oslash$ χρησιμοποιούνται για το γινόμενο ανά στοιχείο και τη διαίρεση ανά στοιχείο, αντίστοιχα. $\mathbf{I}_n$ είναι η μοναδιαία μήτρα διαστάσεων $n \times n$ και $\mathbf{1}_{n\times m}$ η μήτρα όλων των μονάδων διαστάσεων $n \times m$. Τέλος, $||\mathbf{X}||_{\mathrm{F}}$ δηλώνει τη Frobenius νόρμα του $\mathbf{X}$.

1.1.2 Εισαγωγή στην NMF

Η Μη Αρνητική Παραγοντοποίηση Μητρώου (Non-negative Matrix Factorization, NMF) αναζητά μια αθροιστική, βασισμένη σε μέρη παραγοντοποίηση ενός μη αρνητικού μητρώου δεδομένων $\mathbf{V} \in \mathbb{R}^{n\times m}_{\geq 0}$ σε δύο χαμηλής τάξης παράγοντες:

$$\mathbf{V} \approx \mathbf{W}\,\mathbf{H}, \mathbf{W} \in \mathbb{R}^{n\times r}_{\geq 0}, \quad \mathbf{H} \in \mathbb{R}^{r\times m}_{\geq 0}, \quad r \ll \min(n, m).$$

Με την επιβολή των μη αρνητικών περιορισμών $\mathbf{W}, \mathbf{H} \geq 0$, η NMF παράγει μια παραγοντοποίηση όπου οι στήλες του $\mathbf{W}$ αντιπροσωπεύουν μη αρνητικά «δομικά στοιχεία» και οι αντίστοιχες στήλες του $\mathbf{H}$ κωδικοποιούν μη αρνητικούς συνδυαστικούς συντελεστές. Σε αντίθεση με άλλες τεχνικές γραμμικής μείωσης διαστάσεων, όπως η Principal Component Analysis (PCA) [1] ή η Singular Value Decomposition (SVD) [2], που επιτρέπουν θετικές και αρνητικές τιμές στα στοιχεία των πινάκων παραγόντων, η NMF επιβάλλει καθαρά αθροιστικό μοντέλο, οδηγώντας συχνά σε πιο ερμηνεύσιμες και αραιές (sparse) αναπαραστάσεις.

Διατύπωση του Προβλήματος NMF

Τυπικά, δεδομένου ενός μη αρνητικού μητρώου $\mathbf{V} \in \mathbb{R}^{n\times m}_{\geq 0}$, ενός επιθυμητού βαθμού παραγοντοποίησης r και ενός μέτρου απόστασης $D(\cdot, \cdot)$ μεταξύ μητρών, η NMF αναζητά δύο μη αρνητικούς παράγοντες $\mathbf{W} \in \mathbb{R}^{n\times r}_{\geq 0}$ και $\mathbf{H} \in \mathbb{R}^{r\times m}_{\geq 0}$ ώστε:

$$\min_{\mathbf{W}\in\mathbb{R}^{n\times r}_{\geq 0},\, \mathbf{H}\in\mathbb{R}^{r\times m}_{\geq 0}} D\big(\mathbf{V},\, \mathbf{W}\,\mathbf{H}\big). \tag{1.1.1}$$

Επιλογή του Βαθμού Παραγοντοποίησης r

Ο βαθμός παραγοντοποίησης r παίζει κεντρικό ρόλο στη NMF, καθώς ελέγχει την ικανότητα του μοντέλου να αποτυπώνει λανθάνουσες δομές. Επιλογή πολύ μικρού r οδηγεί σε υποπλήρες μοντέλο, όπου η περιορισμένη βάση αναγκάζει το μοντέλο να συμπιέσει ποικίλα μοτίβα, προκαλώντας υψηλό σφάλμα ανακατασκευής και υποπροσαρμογή (underfitting). Αντίθετα, επιλογή πολύ μεγάλου r οδηγεί σε υπερπλήρες μοντέλο που ανακατασκευάζει σχεδόν τέλεια τα δεδομένα, αλλά σε βάρος της μοναδικότητας και της ευρωστίας, αυξάνοντας τον κίνδυνο υπερπροσαρμογής (overfitting) και μειωμένης ερμηνευσιμότητας. Δεν υπάρχει ένας καθολικά βέλτιστος βαθμός r που να αποδίδει καλύτερα σε όλα τα σύνολα δεδομένων ή εργασίες. Στην πράξη, το r επιλέγεται μέσω διαδικασιών επιλογής μοντέλου, όπως cross-validation, μετρικών απόδοσης προσαρμοσμένων στην εφαρμογή, ή domain-specific ευρημάτων που εξισορροπούν την ποιότητα ανακατασκευής με την απλότητα του μοντέλου.

Μη-μοναδικότητα και η ανάγκη για κανονικοποίηση

Το πρόβλημα βελτιστοποίησης που ορίζεται στο (1.1.1) είναι *ταυτόχρονα μη-κυρτό* στις μεταβλητές $(\mathbf{W}, \mathbf{H})$ και έχει άπειρες λύσεις. Συγκεκριμένα, για κάθε αντιστρέψιμη μήτρα $\mathbf{Q} \geq 0$ με $\mathbf{Q}^{-1} \geq 0$, το μετασχηματισμένο ζεύγος

$$\mathbf{W}' = \mathbf{W}\,\mathbf{Q}, \qquad \mathbf{H}' = \mathbf{Q}^{-1}\,\mathbf{H}$$

ικανοποιεί

$$\mathbf{W}'\mathbf{H}' \;=\; \mathbf{W}\,\mathbf{Q}\,\mathbf{Q}^{-1}\,\mathbf{H} \;=\; \mathbf{W}\,\mathbf{H}.$$

Η έμφυτη αυτή αμφισημία περιορίζει την ερμηνευσιμότητα των λύσεων της NMF, ιδιαιτέρως σε υπερπλήρεις ρυθμίσεις.

Για να αντιμετωπιστεί αυτό, συνήθως το πρόβλημα της NMF εμπλουτίζεται με επιπλέον όρους κανονικοποίησης ή περιορισμούς που επιβάλλουν επιθυμητή δομή στους παράγοντες [3]. Παραδείγματα:

- **Περιορισμοί αραιότητας** στο $\mathbf{H}$, για την ενίσχυση αναπαραστάσεων βασισμένων σε μέρη.
- **Περιορισμοί ορθογωνιότητας** στο $\mathbf{W}$, για την προώθηση ποικιλίας στη βάση.
- **Κανονικοποίηση βάσει ομαλότητας ή γράφων**, για τη μοντελοποίηση σχέσεων μεταξύ χαρακτηριστικών ή δειγμάτων.
- **Περιορισμοί όγκου ή simplex**, για την επιβολή γεωμετρικής ή πιθανοτικής ερμηνευσιμότητας.

Τέτοιοι όροι κανονικοποίησης βοηθούν στο σπάσιμο της συμμετρίας του χώρου λύσεων και κατευθύνουν τη βελτιστοποίηση προς ουσιαστικές και μοναδικές παραγοντοποιήσεις [4].

1.1.3 Αλγόριθμος Πολλαπλασιαστικών Ανανεώσεων για την NMF

Πολλοί αλγόριθμοι έχουν προταθεί για την επίλυση του προβλήματος NMF, μεταξύ των οποίων τα Εναλλασσόμενα Ελάχιστα Τετράγωνα (Alternating Least Squares, ALS) [5], οι Projected Gradient Methods [6], η Μέθοδος Κατάβασης Συντεταγμένων (Coordinate Descent) [7] και η Ιεραρχική Εναλλασσόμενη Βελτιστοποίηση [8]. Μεταξύ αυτών, ο αλγόριθμος Πολλαπλασιαστικών Ενημερώσεων (Multiplicative Update, MU) που εισήγαγαν οι Lee και Seung [9] είναι ιδιαίτερα δημοφιλής χάρη στην απλότητα, την ευκολία υλοποίησης και την εγγενή διατήρηση της μη-αρνητικότητας των στοιχείων των παραγόντων. Στην παρούσα εργασία εστιάζουμε στον αλγόριθμο MU, καθώς αποτελεί τη βάση για τις επόμενες επεκτάσεις που θα ακολουθήσουν.

Δεδομένης της αντικειμενικής συνάρτησης που οριζεται με βάση τη Frobenius νόρμα:

$$\min_{\mathbf{W},\mathbf{H}\geq 0} \quad \frac{1}{2}\|\mathbf{V}-\mathbf{W}\mathbf{H}\|_F^2,$$

οι κανόνες MU ενημερώνουν επαναληπτικά τα στοιχεία των μητρώων $\mathbf{W}$ και $\mathbf{H}$ ως εξής:

1. Αρχικοποίηση των $\mathbf{W}$ και $\mathbf{H}$ με τυχαίες μη-αρνητικές τιμές.
2. Επανάληψη έως ότου επιτευχθεί σύγκλιση:

$$h_{ij} \leftarrow h_{ij}\frac{(\mathbf{W}^\top\mathbf{V})_{ij}}{(\mathbf{W}^\top\mathbf{W}\,\mathbf{H})_{ij}+\epsilon}, \quad \forall\, i,j,$$

$$w_{ij} \leftarrow w_{ij}\frac{(\mathbf{V}\,\mathbf{H}^\top)_{ij}}{(\mathbf{W}\,\mathbf{H}\,\mathbf{H}^\top)_{ij}+\epsilon}, \quad \forall\, i,j,$$

όπου ϵ είναι ένας μικρός σταθερός παράγοντας για αριθμητική σταθερότητα και αποφυγή διαίρεσης με το μηδέν. Κάθε ενημέρωση διατηρεί τη μη-αρνητικότητα λόγω της πολλαπλασιαστικής φύσης των κανόνων.

1.1.4 Εφαρμογές της NMF

Η Μη Αρνητική Παραγοντοποίηση Μητρώου (NMF) βρίσκει ποικίλες εφαρμογές σε πολλούς τομείς, μεταξύ των οποίων:

- **Επεξεργασία Εικόνας – Εξαγωγή Χαρακτηριστικών Προσώπου.** Μια κλασική εφαρμογή της NMF είναι η εξαγωγή χαρακτηριστικών προσώπου, όπως πρωτοπαρουσιάστηκε από τους Lee και Seung [10]. Σε αυτή την περίπτωση, έχουμε ένα μητρώο δεδομένων $\mathbf{X}\in\mathbb{R}_{\geq 0}^{n\times m}$ που αναπαριστά διάνυσμα γκρι-επίπεδων εικόνων ανθρώπινων προσώπων. Η τιμή του στοιχείου x_{ij} αντιστοιχεί στην ένταση του i-οστού πίξελ στην j-οστή εικόνα. Με την NMF, το $\mathbf{V}$ παραγοντοποιείται σε μια βάση $\mathbf{W}\in\mathbb{R}_{\geq 0}^{n\times r}$ και έναν πίνακα συντελεστών $\mathbf{H}\in\mathbb{R}_{\geq 0}^{r\times m}$. Τα εξαγόμενα στοιχεία του πίνακα βάσης αντιστοιχούν σε τμήματα του προσώπου, όπως μάτια, χείλη, μουστάκια και μύτες, και λόγω της μη-αρνητικότητας, οι εικόνες βάσης αθροίζονται μόνο προσθετικά για την ανασύνθεση της αρχικής εικόνας. Όπως φαίνεται στο Σχήμα 1.1.1:

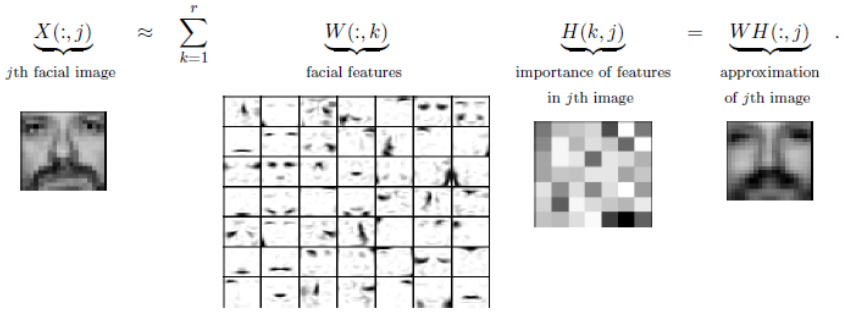


Figure 1.1.1: NMF Facial Feature Extraction. From [11].

- **Εξόρυξη Κειμένου – Ανάκτηση Θεμάτων και Ταξινόμηση Εγγράφων.** Η χρήση της NMF για ομαδοποίηση εγγράφων και ανάλυση θεμάτων εισήχθη από τους Xu et al. [12]. Εδώ, το μητρώο $\mathbf{X} \in \mathbb{R}_{\geq 0}^{n \times m}$ έχει στήλες που αντιστοιχούν σε έγγραφα και γραμμές σε λέξεις, όπου $\mathbf{X}(i,j)$ είναι το πλήθος εμφάνισης της j-οστής λέξης στο i-οστό έγγραφο. Λόγω της μη-αρνητικότητας, κάθε στήλη της $\mathbf{W}$ ερμηνεύεται ως διανυσματική αναπαράσταση ενός θέματος, και η $\mathbf{H}$ καθορίζει την κατανομή των θεμάτων ανά έγγραφο. Αυτό απεικονίζεται στο Σχήμα 1.1.2:

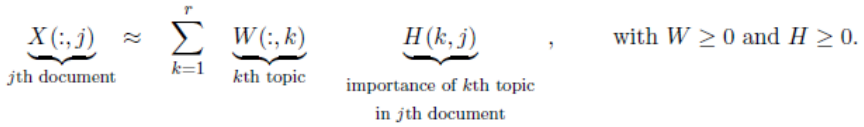


Figure 1.1.2: NMF for Topic Modeling. From [11].

- **Υπερφασματικός Διαχωρισμός – Ανίχνευση Endmembers και Ταξινόμηση Πίξελ.** Η NMF προτάθηκε για πρώτη φορά στη θεωρία του υπερφασματικού διαχωρισμού από τους Nascimento και Dias [13]. Σε μια υπερφασματική εικόνα, κάθε πίξελ περιέχει φασματικές υπογραφές πολλαπλών μηκών κύματος. Στόχος είναι η αναγνώριση των μοναδικών υπογραφών κάθε υλικού (endmembers) και ο υπολογισμός της ατομικής συμβολής τους σε κάθε πίξελ. Το μητρώο $\mathbf{X}$ παραγοντοποιείται ως $\mathbf{W}$ (στήλες = endmembers) και $\mathbf{H}$ (abundance matrix), όπως φαίνεται στο Σχήμα 1.1.3:

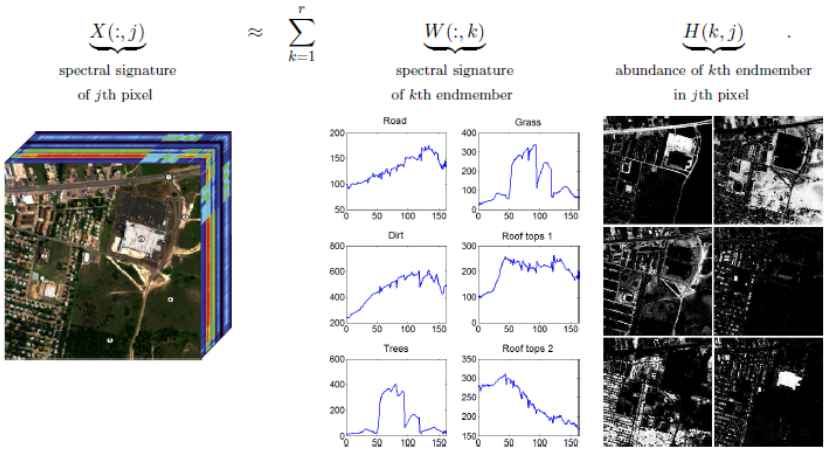


Figure 1.1.3: NMF For Spectral Unmixing. From [11].

- **Ιατρικές Εφαρμογές – Γονιδιώματα Καρκίνου.** Η NMF έχει σημαντική εφαρμογή στην ανάλυση γονιδιωμάτων καρκίνου. Στην εμβληματική μελέτη των Alexandrov et al. [14], εφαρμόστηκε NMF σε δεκάδες χιλιάδες αλληλουχίες γονιδιωμάτων για την εξαγωγή διακριτών «υπογραφών μεταλλάξεων» και την εκτίμηση της έκθεσής τους. Εδώ, το $\mathbf{V}$ είναι πίνακας μετρήσεων μεταλλάξεων, η $\mathbf{W}$ οι υπογραφές και η $\mathbf{H}$ οι εκθέσεις, αποκαλύπτοντας μηχανισμούς βλάβης και επιδιόρθωσης DNA στον καρκίνο.

1.2 Algorithm Unrolling

Το Algorithm Unrolling είναι μία τεχνική που μετασχηματίζει επαναληπτικούς αλγορίθμους σε βαθιά νευρωνικά δίκτυα, αντιστοιχίζοντας κάθε επανάληψη του αλγορίθμου σε ένα επίπεδο (layer) του δικτύου. Ουσιαστικά, κάθε βήμα μιας επαναληπτικής διαδικασίας γίνεται ένα ερμηνεύσιμο επίπεδο, του οποίου οι παράμετροι—προηγουμένως καθοριζόμενες χειροκίνητα ή με σταθερούς κανόνες—τώρα μπορούν να μαθευτούν από τα δεδομένα. Με τη στοίβαξη αυτών των επιπέδων, η προκύπτουσα βαθιά αρχιτεκτονική διατηρεί την ερμηνευσιμότητα των παραδοσιακών αλγορίθμων, ενώ αποκτά την προσαρμοστικότητα και τα οφέλη απόδοσης της βαθιάς μάθησης. Η δομή ενός τέτοιου unrolled δικτύου φαίνεται στο Σχήμα 1.2.1.

Το παλαιότερο έργο στο Algorithm Unrolling παρουσιάστηκε από τους Gregor et al. [15], με στόχο τη βελτίωση της υπολογιστικής αποδοτικότητας αλγορίθμων αραιής κωδικοποίησης μέσω end-to-end εκπαίδευσης. Στο έργο τους, ξεδίπλωσαν τον Iterative Shrinkage and Thresholding Algorithm (ISTA), ο οποίος επιλύει το πρόβλημα της αραιής κωδικοποίησης [16], και πρότειναν το αντίστοιχο unrolled δίκτυο, αργότερα ονομαζόμενο learned ISTA (LISTA). Στη συνέχεια, αναλύουμε αυτό το παράδειγμα ξετυλίγματος λεπτομερώς, εξετάζοντας πώς κάθε επανάληψη του ISTA αντιστοιχίζεται σε επίπεδο δικτύου.

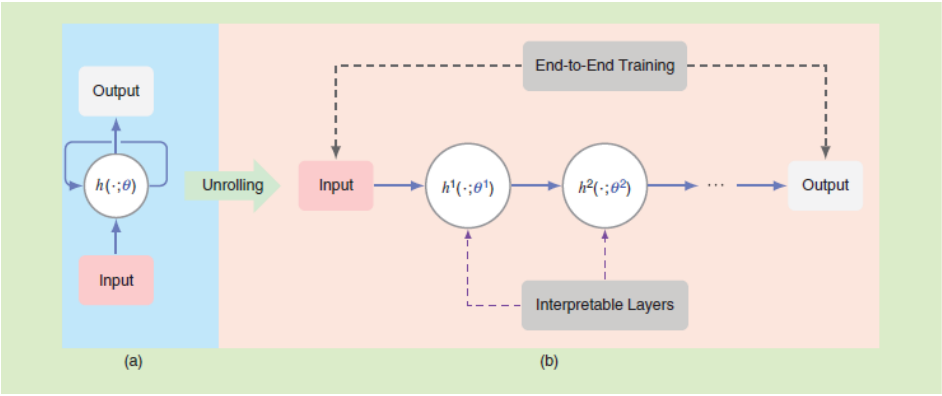


Figure 1.2.1: Απεικόνιση unrolled δικτύου από [17].

Συγκεκριμένα, δεδομένου ενός διανύσματος εισόδου $\mathbf{y} \in \mathbb{R}^n$ και ενός υπερπλήρους λεξικού $\mathbf{W} \in \mathbb{R}^{n\times m}$ με $m > n$, η αραιή κωδικοποίηση αναφέρεται στην εύρεση μιας αραιής αναπαράστασης του $\mathbf{y}$ χρησιμοποιώντας το $\mathbf{W}$. Με άλλα λόγια, επιδιώκουμε να βρούμε έναν αραιό κώδικα $\mathbf{x} \in \mathbb{R}^m$ τέτοιο ώστε $\mathbf{y} \approx \mathbf{W}\mathbf{x}$, ενώ "σπρώχνουμε" όσο το δυνατόν περισσότερους συντελεστές του $\mathbf{x}$ στο μηδέν. Μια συνήθης προσέγγιση είναι η επίλυση του ακόλουθου κυρτού προβλήματος βελτιστοποίησης:

$$\min_{\mathbf{x}\in\mathbb{R}^m} \ \frac{1}{2}\|\mathbf{y} - \mathbf{W}\mathbf{x}\|_2^2 \ + \ \lambda\|\mathbf{x}\|_1, \tag{1.2.1}$$

όπου $\lambda > 0$ είναι παράμετρος κανονικοποίησης που ελέγχει την αραιότητα της λύσης. Συγκεκριμένα, ο ISTA βελτιώνει την εκτίμησή του σε κάθε επανάληψη ως εξής:

$$\mathbf{x}^{t+1} \ = \ S_{\lambda/\mu}\Big(\big(\mathbf{I} - \tfrac{1}{\mu}\mathbf{W}^\top\mathbf{W}\big)\,\mathbf{x}^t + \tfrac{1}{\mu}\mathbf{W}^\top\mathbf{y}\Big), \quad t = 0, 1, \ldots \tag{1.2.2}$$

όπου $\mathbf{I} \in \mathbb{R}^{m\times m}$ η μοναδιαία μήτρα, $\mu > 0$ ο ρυθμός βήματος και $S_\theta(\cdot)$ ο στοιχειώδης soft-thresholding τελεστής $S_\theta(\mathbf{x}) = \text{sign}(\mathbf{x})\max\{|\mathbf{x}| - \theta, 0\}$.

Μπορούμε να ερμηνεύσουμε τις L επαναλήψεις του ISTA ως επίπεδα δικτύου, σχηματίζοντας ένα βαθύ δίκτυο L-επιπέδων όπως απεικονίζεται στο Σχήμα 3.3.2. Στο unrolled δίκτυο λαμβάνουμε υπόψη τις παραμέτρους προς

εκμάθηση:

$$\mathbf{W}_t = \mathbf{I} - \tfrac{1}{\mu}\mathbf{W}^\top\mathbf{W}, \quad \mathbf{W}_e = \tfrac{1}{\mu}\mathbf{W}^\top.$$

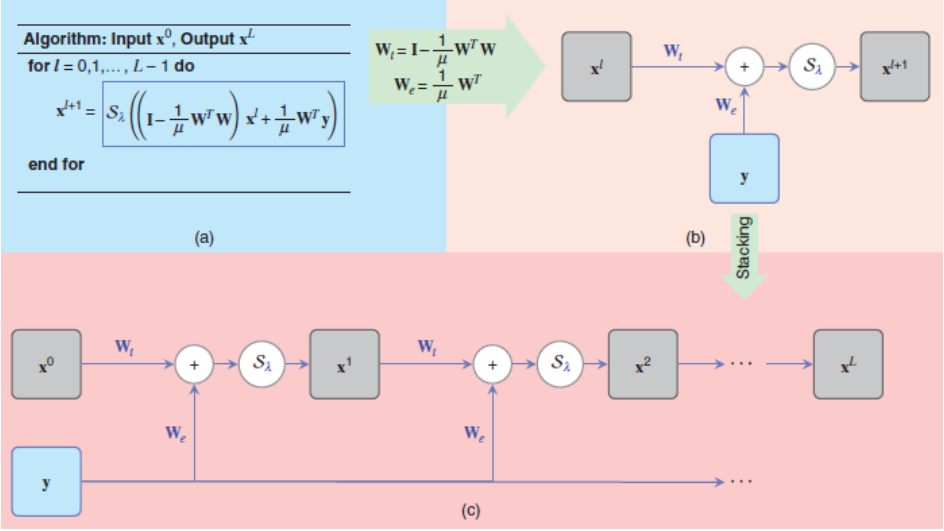


Figure 1.2.2: Ξεδίπλωμα του ISTA σε LISTA από [17].

Το unrolled δίκτυο εκπαιδεύεται με πραγματικά δεδομένα για τη βελτιστοποίηση των παραμέτρων $\mathbf{W}_t$, $\mathbf{W}_e$ και λ. Η εκπαίδευση του LISTA γίνεται πάνω σε ακολουθία διανυσμάτων $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^N \in \mathbb{R}^n$ και των αντίστοιχων "ground-truth" σπάνιων κωδικών $\mathbf{x}^{*1}, \mathbf{x}^{*2}, \ldots, \mathbf{x}^{*N}$. Οι παράμετροι του δικτύου βελτιστοποιούνται με δημοφιλείς gradient-based μεθόδους, όπως το Stochastic Gradient Descent [18]. Πειραματικά παρατηρήθηκε ότι ο αριθμός των επιπέδων του LISTA είναι σημαντικά μικρότερος από τον αριθμό επαναλήψεων που απαιτεί ο ISTA για την επίτευξη των ίδιων αποτελεσμάτων.

1.3 Μοντέλα Βαθιάς Ισορροπίας

Σε αυτή την ενότητα συγκεντρώνουμε τις βασικές ιδέες που αποτελούν τη βάση των Μοντέλων Βαθιάς Ισορροπίας (Deep Equilibrium Models, DEQs). Ξεκινάμε αναπαριστώντας ένα βαθύ δίκτυο ως επανάληψη σταθερού σημείου (fixed-point iteration), και στη συνέχεια δείχνουμε πώς το ξετύλιγμα μιας τέτοιας επανάληψης σε ρητά ορισμένα «επίπεδα» και η κοινή δέσμευση των βαρών τους οδηγεί σε μια ισχυρή επαναλαμβανόμενη αρχιτεκτονική. Βάσει αυτού, εισάγουμε τον όρο των implicit layers, των οποίων οι έξοδοι ορίζονται όχι από μια πεπερασμένη αλυσίδα μετασχηματισμών αλλά από την επίλυση μιας εξίσωσης ισορροπίας, και δείχνουμε πώς η εύρεση ρίζας και η χρήση implicit differentiation συνδυάζονται για να σχηματίσουν τα DEQs.

1.3.1 Implicit Layers

Όπως αναφέρθηκε προηγουμένως, ένα βαθύ νευρωνικό δίκτυο αποτελείται από μεγάλο αριθμό συνδεδεμένων επιπέδων. Στη σύγχρονη βαθιά μάθηση, η συντριπτική πλειοψηφία αυτών των επιπέδων είναι ρητά (explicit) — ορισμένα δηλαδή από μια σταθερή σειρά λειτουργιών που μετασχηματίζουν την είσοδο σε έξοδο. Αντίθετα, ένα implicit επίπεδο ορίζει τις συνθήκες που πρέπει να ικανοποιεί η έξοδός του, χωρίς να προσδιορίζει τον ακριβή υπολογισμό για την απόκτησή της. Η λογική ενός implicit επιπέδου και η διαφορά του με ένα ρητά ορισμένο επίπεδο παρουσιάζεται στο Σχήμα 1.3.1.

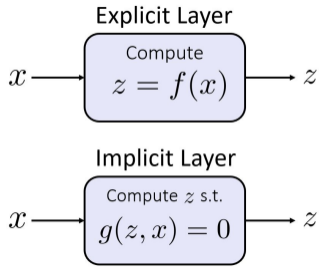
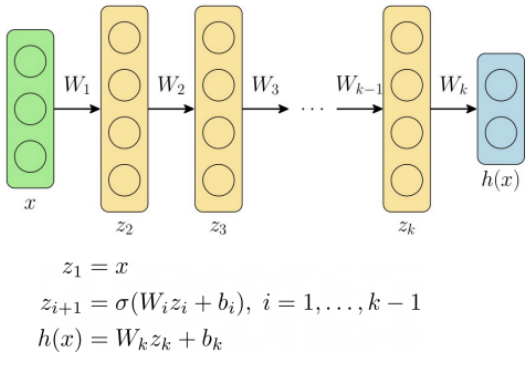


Figure 1.3.1: Απεικόνιση Implicit Επιπέδου [19]

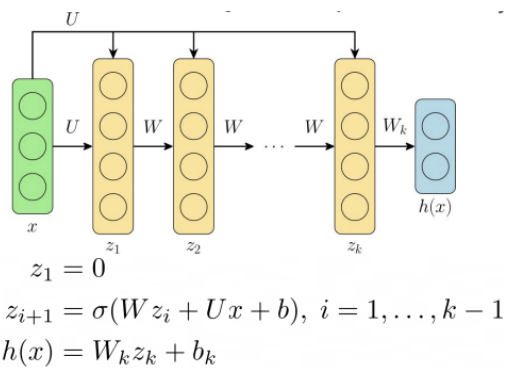
1.3.2 Δίκτυα με Κοινά Βάρη

Τα δίκτυα με κοινά βάρη (Weight-Tied Networks) αποτελούν μια κατηγορία νευρωνικών δικτύων όπου το ίδιο σύνολο βαρών επαναχρησιμοποιείται σε πολλαπλά επίπεδα. Αντί να μαθαίνουν ξεχωριστές παραμέτρους για κάθε επίπεδο, τα δίκτυα με κοινά βάρη επιβάλλουν κοινή χρήση παραμέτρων, δηλαδή κάθε μετασχηματισμός που ακολουθεί στη σειρά εφαρμόζει την ίδια μήτρα βαρών. Σχηματικά, για μια είσοδο $\mathbf{x}$, το δίκτυο υπολογίζει τις κρυφές καταστάσεις μέσω της επανάληψης $\mathbf{z}_{i+1} = \sigma\big(\mathbf{W}\,\mathbf{z}_i + \mathbf{U}\,\mathbf{x} + \mathbf{b}\big)$, όπου οι μήτρες $\mathbf{W}$ και $\mathbf{U}$ είναι κοινές σε όλα τα επίπεδα. Η ιδέα της κοινής χρήσης embeddings εισόδου-εξόδου, που αντικατοπτρίζει την αρχή των δικτύων με κοινά βάρη, πρωτοτυπικά θεμελιώθηκε από τους Press και Wolf [20] και έχει γίνει θεμελιώδης στην μοντελοποίηση ακολουθιών. Η δέσμευση βαρών είναι ιδιαίτερα χρήσιμη σε μοντέλα για επεξεργασία ακολουθιακών δεδομένων, όπως στα Επαναληπτικά Νευρωνικά Δίκτυα (Recurrent Neural Networks, RNNs), και λειτουργεί ως εννοιολογικό βήμα προς τα DEQs. Σημειώνουμε ότι οποιοδήποτε βαθύ δίκτυο μπορεί να δομηθεί εκ νέου ως δίκτυο με κοινά βάρη με το ίδιο βάθος και με μέγεθος μονάδας ίσο με το άθροισμα των μεγεθών των κρυφών μονάδων του αρχικού δικτύου—η απόδειξη αναφέρεται στο παράρτημα του [19].

Αυτή η δομική διαφορά απεικονίζεται στο Σχήμα 1.3.2, όπου το δίκτυο με κοινά βάρη επαναχρησιμοποιεί την ίδια μήτρα μετασχηματισμού σε όλα τα επίπεδα, σε αντίθεση με ένα κλασικό βαθύ δίκτυο, που μαθαίνει ξεχωριστή μήτρα ανά επίπεδο.



(a) Κλασικό Βαθύ Δίκτυο



(b) Δίκτυο με Κοινά Βάρη

Figure 1.3.2: Σύγκριση αρχιτεκτονικών κλασικού βαθιού δικτύου και δικτύου με κοινά βάρη.

1.3.3 Σταθερά Σημεία

Η έννοια του σταθερού σημείου (fixed point) αποτελεί θεμελιώδες μαθηματικό εργαλείο που θα χρειαστούμε σύντομα όταν συζητήσουμε μια ευρεία κατηγορία μοντέλων ορισμένων μέσω recurrent ή equilibrium διατυπώσεων. Εστω $f : \mathcal{X} \to \mathcal{X}$ μια συνάρτηση ορισμένη σε σύνολο $\mathcal{X}$. Ένα σημείο $x^\star \in \mathcal{X}$ ονομάζεται **σταθερό σημείο** της f αν

$$f(x^\star) = x^\star.$$

Δηλαδή, αν επιλέξουμε ένα $x \in \mathcal{X}$ και εφαρμόσουμε ξανά και ξανά τη συνάρτηση f, τελικά θα καταλήξουμε σε ένα σημείο $x^\star$ —το σταθερό σημείο ή σημείο ισορροπίας— που δεν μεταβάλλεται περαιτέρω όταν του εφαρμοστεί η f. Τέτοια σημεία συχνά αντιπροσωπεύουν καταστάσεις ισορροπίας ή σύγκλισης σε δυναμικά συστήματα και επαναληπτικές διαδικασίες. Στην επόμενη ενότητα, θα εξετάσουμε μοντέλα που αξιοποιούν διατυπώσεις σταθερών σημείων για να ορίσουν τη συμπεριφορά τους.

1.3.4 Εισαγωγή στα Μοντέλα Βαθιάς Ισορροπίας

Τα Μοντέλα Βαθιάς Ισορροπίας (Deep Equilibrium Models, DEQs) είναι μοντέλα θεμελιωδώς βασισμένα στην έννοια των επαναληπτικών σταθερών σημείων. Σε παραδοσιακά δίκτυα με κοινά βάρη, η ανανέωση κάθε επιπέδου γράφεται ως:

$$\mathbf{z}_{1:T}^{[i+1]} = f_\theta^{[i]}\big(\mathbf{z}_{1:T}^{[i]}\ ;\ \mathbf{x}_{1:T}\big) \quad \text{για } i = 0, 1, 2, \ldots, L-1, \tag{1.3.1}$$

όπου i είναι ο δείκτης του επιπέδου, $\mathbf{z}_{1:T}^{[i]}$ η κρυφή ακολουθία μήκους T στο επίπεδο i, L ο συνολικός αριθμός επιπέδων, $\mathbf{x}_{1:T}$ η ακολουθία εισόδου, και $f_\theta^{[i]}$ κάποιος μη γραμμικός μετασχηματισμός που συνήθως επιβάλλει αιτιότητα.

Στην πράξη, έχει παρατηρηθεί ότι εάν η συνάρτηση f_θ πληροί ορισμένες προϋποθέσεις και εφαρμοστεί επαναληπτικά, το δίκτυο συγκλίνει σε ένα σημείο ισορροπίας $\mathbf{z}_{1:T}^\star$. Αυτή η σύγκλιση αποτυπώνεται ως:

$$\lim_{i\to\infty} \mathbf{z}_{1:T}^{[i]} \ = \ \lim_{i\to\infty} f_\theta\big(\mathbf{z}_{1:T}^{[i]};\ \mathbf{x}_{1:T}\big) \ \equiv \ f_\theta\big(\mathbf{z}_{1:T}^\star;\ \mathbf{x}_{1:T}\big) = \mathbf{z}_{1:T}^\star. \tag{1.3.2}$$

Στην πράξη έχει δειχτεί ότι καθώς αυξάνεται το βάθος ενός δικτύου με κοινά βάρη, επιτυγχάνεται βελτιωμένη απόδοση. Ωστόσο, στη συμβατική βαθιά μάθηση, για να πραγματοποιηθεί backpropagation σε ολόκληρο το δίκτυο, απαιτείται ο υπολογισμός των παραγώγων του συνόλου των παραμέτρων σε κάθε επίπεδο, γεγονός που περιορίζει το βάθος στο οποίο μπορούμε να εκπαιδεύσουμε τα μοντέλα, ανάλογα με τη διαθέσιμη μνήμη. Τα DEQs, αντίθετα, χρησιμοποιούν ένα ενιαίο implicit επίπεδο, το οποίο ορίζεται έμμεσα από την εξίσωση σταθερού σημείου $\mathbf{z}^* = f_{\boldsymbol{\theta}}\big(\mathbf{z}^*, \mathbf{x}\big)$. Με άλλα λόγια, αντί να στοιβάζουμε L ρητά επίπεδα, ορίζουμε μέσω ενός implicit επιπέδου τη λύση $\mathbf{z}^*$ αυτού του προβλήματος εύρεσης ρίζας. Στη συνέχεια, αξιοποιούμε την implicit παραγώγιση και εκτελούμε ένα μόνο βήμα αναδρομής, αντί να κάνουμε backpropagate σε κάθε επανάληψη του αριθμητικού επιλυτή εξισώσεως που έχουμε διαλέξει. Αυτές οι ιδέες αναπτύσσονται λεπτομερώς παρακάτω.

Forward Pass Η εκπαίδευση των DEQs περιλαμβάνει διαφορετικές διαδικασίες forward και backward σε σύγκριση με τα συμβατικά βαθιά δίκτυα. Στο forward pass, τα DEQs βρίσκουν την κατάσταση ισορροπίας τους επιλύοντας μια εξίσωση σταθερού σημείου, αντί να υπολογίζουν διαδοχικά τις ενεργοποιήσεις επίπεδο προς επίπεδο. Ορίζουμε:

$$g_\theta(\mathbf{z}_{1:T}^\star; \mathbf{x}_{1:T}) = f_\theta(\mathbf{z}_{1:T}^\star; \mathbf{x}_{1:T}) - \mathbf{z}_{1:T}^\star \tag{1.3.3}$$

Στη συνέχεια, χρησιμοποιούμε αριθμητικές τεχνικές εύρεσης ρίζας για την επίλυση της Εξίσωσης 1.3.3. Οι μέθοδοι αυτές βελτιώνουν επαναληπτικά μια αρχική εικασία έως ότου επιτευχθεί ένα σταθερό σημείο ισορροπίας. Στην πράξη, οι συνηθέστερες μέθοδοι εύρεσης ρίζας για επαναληπτικές διαδικασίες σταθερού σημείου περιλαμβάνουν την επιτάχυνση Anderson [21], [22], τη μέθοδο Broyden [23] και τη μέθοδο Newton [24], [25]. Το Σχήμα 1.3.3 απεικονίζει το forward pass ενός μοντέλου DEQ ως επαναληπτική διαδικασία που συγκλίνει στο σημείο ισορροπίας $\mathbf{z}^*$, παρουσιάζοντας τη διαφορά σε σχέση με τη συνηθισμένη υπολογιστική διαδικασία σε δίκτυα με κοινά βάρη.

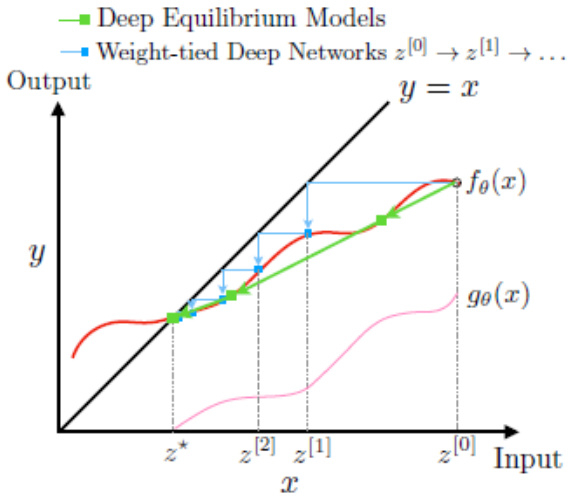


Figure 1.3.3: DEQ vs weight-tied βαθιά δίκτυα λύνοντας για το σημείο ισορροπίας στις 2 διαστάσεις. Από [19].

Backward Pass Ένα ζήτημα που προκύπτει από τη χρήση ενός "black-box" αλγορίθμου εύρεσης ρίζας είναι ότι δεν μπορούμε πλέον να εκτελέσουμε backpropagation μέσω των λειτουργιών στο forward pass , όπως συμβαίνει στη συμβατική βαθιά μάθηση. Αντίθετα, το backward pass στα DEQs βασίζεται στο Θεώρημα Implicit Συναρτήσεων [26]. Πιο γενικά, το πλαίσιο των DEQs επεκτείνει αυτή την προσέγγιση εφαρμόζοντας απευθείας το θεώρημα στην εξίσωση σταθερού σημείου, επιτρέποντας end-to-end παραγώγιση χωρίς ξετύλιγμα των επαναλήψεων εύρεσης ρίζας. Έχει δειχτεί ότι το backward gradient μέσω του δικτύου με "άπειρο" βάθος μπορεί να υπολογιστεί με ένα μόνο πολλαπλασιασμό μητρώων, ο οποίος περιλαμβάνει την Ιακωβιανή υπολογισμένη στο σημείο ισορροπίας $\mathbf{z}^\star_{1:T}$. Συγκεκριμένα, η παράγωγος της συνάρτησης απώλειας ℓ ως προς μια παράμετρο $(\cdot)$ υπολογίζεται όπως στην Εξίσωση (1.3.4), όπου το h συμβολίζει οποιαδήποτε παραγωγίσιμη βαθμωτή συνάρτηση του σημείου ισορροπίας $\mathbf{z}^\star_{1:T}$ και χρησιμοποιείται για να εφαρμόσουμε τον κανόνα αλυσίδας:

$$\frac{\partial \ell}{\partial (\cdot)} = -\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial (\cdot)} = -\frac{\partial \ell}{\partial h} \frac{\partial h}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial (\cdot)}. \tag{1.3.4}$$

Οποιοσδήποτε συνήθης κανόνας ενημέρωσης (π.χ. gradient descent ή Adam) μπορεί να χρησιμοποιηθεί για την ενημέρωση των παραμέτρων θ. Για παράδειγμα, ένα βήμα ενημέρωσης με βάση τον αλγόριθμο Στοχαστικής Κατάβασης Δυναμικού (SGD) θα είχε τη μορφή:

$$\theta^+ = \theta - \alpha \frac{\partial \ell}{\partial \theta} = \theta + \alpha \frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \right) \Big|_{\mathbf{z}^\star_{1:T}} \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial \theta}. \tag{1.3.5}$$

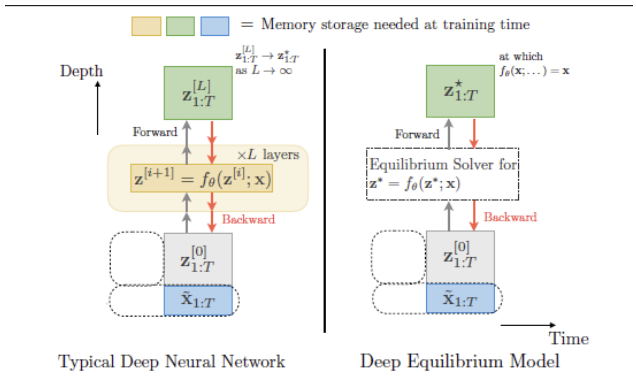


Figure 1.3.4: Σύγκριση χρήσης μνήμης κατά την εκπαίδευση: ένα κλασικό βαθύ δίκτυο με T επίπεδα αποθηκεύει όλες τις ενδιάμεσες ενεργοποιήσεις, ενώ ένα DEQ μοντέλο διατηρεί μόνο το σημείο ισορροπίας, επιτυγχάνοντας σταθερή χρήση μνήμης. Από [19].

Συνήθως, σε ένα κλασικό βαθύ δίκτυο, πρέπει να αποθηκεύσουμε τις ενδιάμεσες τιμές $\mathbf{z}_{1:T}^{[1]}, \ldots, \mathbf{z}_{1:T}^{[L]}$ και να κάνουμε backpropagation μέσω όλων των επιπέδων για την ενημέρωση των παραμέτρων. Εκμεταλλευόμενα το Θεώρημα των Implicit Συναρτήσεων, τα DEQs αποθηκεύουν μόνο το σημείο ισορροπίας $\mathbf{z}_{1:T}^{\star}$, χρησιμοποιώντας έτσι σταθερή μνήμη. Η διαφορά αυτή απεικονίζεται στο Σχήμα 1.3.4. Στην πράξη, η implicit παραγώγιση που απαιτείται κατά το backward pass μπορεί να υλοποιηθεί αποδοτικά με σύγχρονες βιβλιοθήκες αυτόματης παραγώγισης όπως το `PyTorch` [27] ή το `JAX` [28], οι οποίες υποστηρίζουν γινόμενα διανύσματος-Ιακωβιανής και Ιακωβιανής-διανύσματος χωρίς ρητή κατασκευή ολόκληρου του Ιακωβιανού μητρώου.

1.4 NMF με Κανονικοποίηση ℓ_1–ℓ_2

Προηγουμένως παρουσιάσαμε τα θεμελιώδη στοιχεία της NMF και ορίσαμε στους τυπικούς κανόνες MU που εγγυώνται τη σύγκλιση για το μη-κανονικοποιημένο πρόβλημα. Παρότι αυτές οι ενημερώσεις αποτελούν τον κορμό πολλών πρακτικών αλγορίθμων NMF, σε πραγματικά σύνολα δεδομένων συχνά ωφελούνται από πρόσθετους περιορισμούς ή priors. Σε αυτήν την ενότητα, θα ενσωματώσουμε ρητούς όρους ποινής στο πλαίσιο των MU κανόνων. Θα περιγράψουμε ένα υβριδικό σχήμα με όρους ℓ_1 και ℓ_2 κανονικοποίησης, με στόχο η NMF να οδηγήσει σε παράγοντες που είναι ερμηνεύσιμοι, σταθεροί και καλά προσαρμοσμένοι σε θορυβώδη ή ανομοιογενή δεδομένα. Μια κανονικοποιημένη παραλλαγή της NMF, που εφαρμόζει ποινές ℓ_1 και ℓ_2 ανά στοιχείο στον πίνακα συντελεστών $\mathbf{H}$, προτάθηκε από τον Nasser *et al.* στο [29]. Στην παραλλαγή αυτή οδηγούμαστε στο ακόλουθο πρόβλημα ελαχιστοποίησης:

$$\min_{\mathbf{H},\mathbf{W}\geq 0} \left\{ \frac{1}{2} \left\| \mathbf{V} - \mathbf{W}\,\mathbf{H} \right\|_F^2 \;+\; \lambda_1 \left\| \mathbf{H} \right\|_1 \;+\; \frac{\lambda_2}{2} \left\| \mathbf{H} \right\|_F^2 \right\}. \tag{1.4.1}$$

Εμπνευσμένες από τον κλασικό αλγόριθμο MU, προτείνονται οι εξής ανανεώσεις:

$$\mathbf{H} \;\leftarrow\; \mathbf{H} \;\odot\; \frac{\mathbf{W}^\top \mathbf{V}}{\mathbf{W}^\top \mathbf{W}\,\mathbf{H} \;+\; \lambda_1\, \mathbf{1}_{p\times m} \;+\; \lambda_2\, \mathbf{H}}, \tag{1.4.2}$$

$$\mathbf{W} \;\leftarrow\; \mathbf{W} \;\odot\; \frac{\mathbf{V}\,\mathbf{H}^\top}{\mathbf{W}\,\mathbf{H}\,\mathbf{H}^\top}, \tag{1.4.3}$$

όπου το "$\odot$" δηλώνει το ανα στοιχείο (Hadamard) γινόμενο και οι διαιρέσεις γίνονται επίσης ανά στοιχείο. Έχει αποδειχθεί ότι η αντικειμενική συνάρτηση δεν αυξάνεται υπό αυτές τις ανανεώσεις. Ο όρος ℓ_1 επιβάλλει αραιότητα σε κάθε στοιχείο του $\mathbf{H}$, ενώ ο όρος ℓ_2 εισάγει shrinkage που ομαλοποιεί τα στοιχεία του πίνακα συντελεστών και συμβάλλει στη σταθεροποίηση της σύγκλισης.

1.5 Deep Unrolling της Κανονικοποιημένης MU–NMF

Προηγουμένως συζητήσαμε την έννοια του Algorithm Unrolling και πώς μπορούμε να «μετασχηματίσουμε» έναν επαναληπτικό "τυφλό" αλγόριθμο σε ένα εκπαιδεύσιμο δίκτυο. Στη συνέχεια, θα δείξουμε πώς, στο έργο των Nasser *et al.* [29], εφαρμόζεται αυτή η ιδέα στην κανονικοποιημένη εκδοχή του αλγορίθμου MU–NMF.

Όπως αναφέρθηκε προηγουμένως, οι κλασικές πολλαπλασιαστικές ανανεώσεις για το κανονικοποιημένο πρόβλημα MU–NMF που προτείνονται από τους Nasser *et al.* [29] έχουν τη μορφή:

$$\mathbf{H} \;\leftarrow\; \mathbf{H} \;\odot\; \frac{\mathbf{W}^\top \mathbf{V}}{\mathbf{W}^\top \mathbf{W}\,\mathbf{H} \;+\; \lambda_1\, \mathbf{1}_{p\times m} \;+\; \lambda_2\, \mathbf{H}} \quad , \quad \mathbf{W} \;\leftarrow\; \mathbf{W} \;\odot\; \frac{\mathbf{V}\,\mathbf{H}^\top}{\mathbf{W}\,\mathbf{H}\,\mathbf{H}^\top}. \tag{1.5.1}$$

Εποπτευόμενη Deep NMF (DNMF). Βασιζόμενοι στον παραπάνω κανόνα MU, περιγράφουμε εδώ την εποπτευόμενη βαθιά unrolled παραλλαγή του (DNMF). Στην προσέγγιση αυτή υποθέτουμε ότι διαθέτουμε την πραγματική μήτρα συντελεστών $\mathbf{H}$ και την πραγματική μήτρα δεδομένων $\mathbf{V}$. Τώρα θα παρουσιάσουμε τη δομή και τις ιδιότητες του εποπτευόμενου δικτύου DNMF.

Έστω $\mathbf{V} \in \mathbb{R}^{n\times m}$ η μήτρα δεδομένων και $\mathbf{H}^{(l)} \in \mathbb{R}_{\geq 0}^{p\times m}$ οι εκτιμήσεις των συντελεστών στο επίπεδο l. Για κάθε $l = 0, 1, \ldots, L-1$ θεωρούμε ως παραμέτρους του δικτύου τις εξής εκπαιδεύσιμες μη-αρνητικές μήτρες:

- $\mathbf{A}^{(l)} \;=\; \left(\mathbf{W}^{(l)}\right)^\top \;\in\; \mathbb{R}_{\geq 0}^{p\times n}$,

- $\mathbf{B}^{(l)} = \left(\mathbf{W}^\top \mathbf{W}\right)^{(l)} \in \mathbb{R}_{\geq 0}^{p \times p}$,
- καθώς και τους κοινούς για όλα τα επίπεδα βαθμωτούς συντελεστές κανονικοποίησης λ_1 και λ_2.

Η μοντελοποίηση αυτή αγνοεί την εξάρτηση μεταξύ των όρων $\mathbf{W}^\top$ και $\mathbf{W}^\top \mathbf{W}$, αντιμετωπίζοντάς τους ως ανεξάρτητες παραμέτρους του δικτύου. Για να απεικονίσουμε την ανανέωση ανα επίπεδο που συνιστά το forward pass του εποπτευόμενου DNMF, είναι χρήσιμο να εστιάσουμε σε μία μόνο στήλη δείγματος $\mathbf{v} \in \mathbb{R}_{\geq 0}^{n}$ και στην αντίστοιχη στήλη συντελεστών $\mathbf{h}^{(l)} \in \mathbb{R}_{\geq 0}^{p}$. Σε κάθε επίπεδο l, το forward pass εφαρμόζει την ανά στήλη συνάρτηση ενεργοποίησης f:

$$\mathbf{h}^{(l+1)} = f\left(\mathbf{h}^{(l)}, \mathbf{v}\right) = \mathbf{h}^{(l)} \odot \frac{\mathbf{A}^{(l)}\,\mathbf{v}}{\mathbf{B}^{(l)}\,\mathbf{h}^{(l)} + \lambda_1\,\mathbf{1}_p + \lambda_2\,\mathbf{h}^{(l)}}. \tag{1.5.2}$$

Η δομή ενός μόνο επιπέδου του εποπτευόμενου DNMF φαίνεται στο Σχήμα 1.5.1. Αφού ορίσουμε τη συμπεριφορά μίας στρώσης, συνθέτουμε πλήρες unrolled δίκτυο συνδέοντας ℓ στρώσεις. Η είσοδος κάθε στρώσης l είναι το διάνυσμα $\mathbf{v}$ και η έξοδος της προηγούμενης στρώσης $\mathbf{h}^{(l-1)}$. Το πλήρες unrolled δίκτυο με ℓ στρώσεις απεικονίζεται στο Σχήμα 1.5.2.

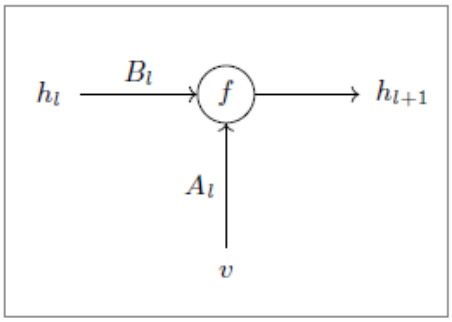


Figure 1.5.1: Ένα επίπεδο DNMF που λειτουργεί στις $\mathbf{h}^{(l)}$ και $\mathbf{v}$.

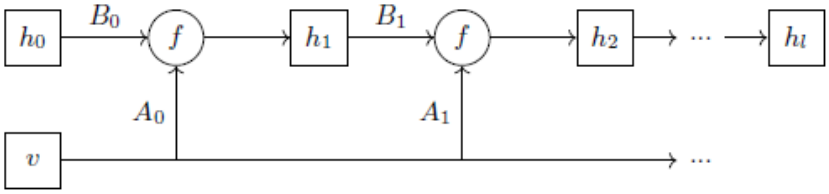


Figure 1.5.2: Unrolled δίκτυο DNMF με ℓ στρώσεις.

Επειδή ο παραπάνω πολλαπλασιαστικός κανόνας εφαρμόζεται ανεξάρτητα σε κάθε στήλη $\mathbf{h}$, δρα φυσικά ως μία ενιαία λειτουργία πάνω σε ολόκληρη τη μήτρα συντελεστών $\mathbf{H}$, ενημερώνοντας όλες τις στήλες παράλληλα:

$$\mathbf{H}^{(l+1)} = \mathbf{H}^{(l)} \odot \frac{\mathbf{A}^{(l)}\,\mathbf{V}}{\mathbf{B}^{(l)}\,\mathbf{H}^{(l)} + \lambda_1\,\mathbf{1}_{p \times m} + \lambda_2\,\mathbf{H}^{(l)}}. \tag{1.5.3}$$

Αρχικοποιούμε τα στοιχεία του πίνακα συντελεστών $\mathbf{H}^{(0)}$ σε μια θετική σταθερά (π.χ. όλες οι καταχωρίσεις ίσες με 1) και επιβάλλουμε τη μη-αρνητικότητα των $\mathbf{A}^{(l)}$ και $\mathbf{B}^{(l)}$ μέσω προβολής στον μη-αρνητικό κώνο μετά από κάθε βήμα ανανέωσης τους.

Εκπαίδευση του Εποπτευόμενου DNMF. Για να εξηγήσουμε τη διαδικασία εκπαίδευσης του DNMF, θα χρησιμοποιήσουμε ξανά προσέγγιση μίας μόνο στήλης. Κατά την εκπαίδευση, στοχεύουμε στη βελτιστοποίηση της ακόλουθης συνάρτησης μέσου τετραγωνικού σφάλματος:

$$\mathcal{L} = \left\| \mathbf{h}^{(L)} - \mathbf{h}' \right\|_2^2,$$

όπου L είναι ο αριθμός των επιπέδων στο δίκτυό μας, $\mathbf{h}^{(L)}$ η έξοδος του τελικού επιπέδου και $\mathbf{h}'$ η πραγματική στήλη συντελεστών. Σε περίπτωση που θέλουμε να εκτιμήσουμε ολόκληρη τη μήτρα συντελεστών, το μοντέλο μας μπορεί να εκπαιδευτεί για να προβλέψει όλες τις m στήλες ταυτόχρονα. Γενικεύουμε τη συνάρτηση σφάλματος στη Frobenius νόρμα τετραγώνου:

$$\mathcal{L} = \left\| \mathbf{H}^{(L)} - \mathbf{H}' \right\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} \left(h_{ij}^{(L)} - h'_{ij} \right)^2 = \operatorname{tr}\left[(\mathbf{H}^{(L)} - \mathbf{H}')^\top (\mathbf{H}^{(L)} - \mathbf{H}') \right]. \tag{1.5.4}$$

Ισοδύναμα, πρόκειται για άθροισμα των στήλη-προς-στήλη των απωλειών ℓ_2:

$$\mathcal{L} = \sum_{i=1}^{m} \left\| \mathbf{h}_i^{(L)} - \mathbf{h}'_i \right\|_2^2. \tag{1.5.5}$$

Η διατύπωση αυτή επιτρέπει στο δίκτυο να μάθει από κοινού όλους τους χάρτες συντελεστών, διατηρώντας μια απλή τετραγωνική ποινή.

Αφού υπολογίσουμε τη συνάρτηση σφάλματος, λαμβάνουμε τις παραγώγους για την ενημέρωση των εκπαιδεύσιμων παραμέτρων του δικτύου

$$\left\{ \mathbf{A}^{(\ell)},\, \mathbf{B}^{(\ell)},\, \lambda_1,\, \lambda_2 \right\}_{\ell=0}^{L-1}$$

μέσω backpropagation με τον βελτιστοποιητή ADAM.

Εφόσον οι λ_1 και λ_2 περιλαμβάνονται ως εκπαιδεύσιμες μεταβλητές, το δίκτυο «μαθαίνει» αυτόματα τον βέλτιστο βαθμό κανονικοποίησης από τα δεδομένα—εξαλείφοντας την ανάγκη για εξωτερικό βρόχο αναζήτησης υπερπαραμέτρων. Κάθε βήμα του gradient ενημερώνει τόσο τους γραμμικούς τελεστές $\{\mathbf{A}^{(\ell)}, \mathbf{B}^{(\ell)}\}$ όσο και τα βάρη κανονικοποίησης λ_1, λ_2, παράγοντας κανόνες ανανέωσης προσαρμοσμένους στα δεδομένα, οι οποίοι συνδυάζουν πιστότητα ανακατασκευής (μέσω MSE) και εκμάθηση των συντελεστών κανονικοποίησης σε μία ενιαία end-to-end διαδικασία.

1.6 Περιγραφή Δεδομένων

Στην παρούσα ενότητα σκοπεύουμε να περιγράψουμε τη φύση των δεδομένων στα οποία θα διεξάγουμε αργότερα τα πειράματά μας. Χρησιμοποιήσαμε προσομοιωμένα σύνολα δεδομένων μεταλλάξεων, όπου η μήτρα δεδομένων $\mathbf{V}$ προκύπτει από τη δράση ορισμένων μεταλλακτικών μηχανισμών των οποίων οι υπογραφές δίνονται από το λεξικό $\mathbf{W}$ και οι εκθέσεις από τη μήτρα συντελεστών $\mathbf{H}$. Για να κατανοήσουμε τη δομή αυτών των συνόλων δεδομένων, πρέπει πρώτα να θέσουμε τις μοριακές βάσεις των δεδομένων μεταλλάξεων. Συγκεκριμένα, αρχίζουμε με ανασκόπηση της δομής της διπλής έλικας του DNA [30] και της αρχής της συμπληρωματικής ζεύξης βάσεων, που μαζί εξηγούν πώς προκύπτουν απλές υποκαταστάσεις βάσεων και πώς καταγράφονται για τη διαμόρφωση του πίνακά μας.

Συμπληρωματική Ζεύξη Βάσεων Το DNA είναι ένα διπλόκλωνο πολυμερές όπου κάθε νουκλεοτίδιο σε μία αλυσίδα συνδέεται με υδρογονικό δεσμό με το συμπλήρωμά του στην αντίθετη αλυσίδα. Αυτό απεικονίζεται στο Σχήμα 1.6.1. Υπάρχουν τέσσερις νουκλεοτιδικές βάσεις συνολικά: αδενίνη (A), γουανίνη (G), κυτοσίνη (C) και θυμίνη (T). Η αδενίνη ζευγαρώνει με τη θυμίνη και η γουανίνη με την κυτοσίνη, δηλαδή οι μοναδικές δυνατές ζεύξεις είναι

$$\mathrm{A} \longleftrightarrow \mathrm{T} \quad \text{και} \quad \mathrm{G} \longleftrightarrow \mathrm{C}.$$

Οι μεταλλάξεις μπορεί να προκύψουν μέσω διαφόρων μηχανισμών—απλές υποκαταστάσεις βάσης (SBS), μικρές εισαγωγές ή μικρές διαγραφές—καθένας από τους οποίους μεταβάλλει διαφορετικά την αλληλουχία του DNA. Εδώ θα εστιάσουμε στις SBS, όπου μια νουκλεοτιδική βάση αντικαθίσταται από μια άλλη χωρίς να αλλάξει το συνολικό μήκος της αλυσίδας. Όταν συμβαίνει μια απλή υποκατάσταση βάσης (π.χ. A→G) στην «εμπρόσθια»

αλυσίδα, οι μηχανισμοί αντιγραφής ή επιδιόρθωσης θα παράγουν τελικά την συμπληρωματική αλλαγή (T→C) στην αντίθετη αλυσίδα. Δεδομένου ότι αυτά τα δύο γεγονότα είναι βιολογικά ισοδύναμα, συνήθως εκφράζουμε όλες τις μεταβολές που αφορούν πουρίνες (A→X ή G→X) ως τις αντίστοιχες συμπληρωματικές αλλαγές σε πυριμιδίνες (T→Y ή C→Y, όπου Y είναι το συμπλήρωμα του X).

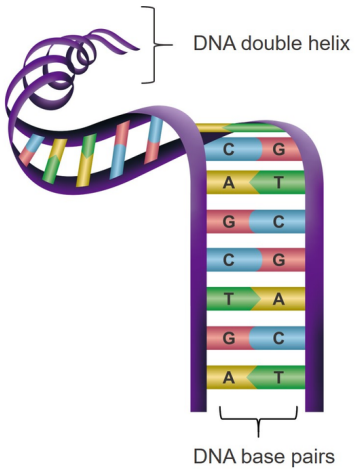


Figure 1.6.1: Διπλή έλικα DNA και απεικόνιση δεσμών νουκλεοτιδίων. Από [31].

Έπειτα, μπορούμε να καταγράψουμε κάθε απλή υποκατάσταση βάσης, φτάνοντας σε 12 διακριτές μεταβολές:

$$A \to C, \quad A \to G, \quad A \to T, \tag{1.6.1}$$
$$C \to A, \quad C \to G, \quad C \to T, \tag{1.6.2}$$
$$G \to A, \quad G \to C, \quad G \to T, \tag{1.6.3}$$
$$T \to A, \quad T \to C, \quad T \to G. \tag{1.6.4}$$

Όπως εξηγήσαμε, η γνώση της βάσης σε μία αλυσίδα του DNA καθορίζει μοναδικά τη συμπλήρωσή της στην αντίθετη αλυσίδα, αφού κάθε βάση ζευγαρώνει μόνο με το συγκεκριμένο συμπλήρωμά της. Επομένως, κατά την κατηγοριοποίηση των υποκαταστάσεων, καταλήγουμε στις ακόλουθες έξι κανονικές μεταλλάξεις:

$$\{\,\mathrm{C}\to\mathrm{A},\ \mathrm{C}\to\mathrm{G},\ \mathrm{C}\to\mathrm{T},\ \mathrm{T}\to\mathrm{A},\ \mathrm{T}\to\mathrm{C},\ \mathrm{T}\to\mathrm{G}\}. \tag{1.6.5}$$

Καταστάσεις Τρινουκλεοτιδίων Για να αποτυπώσουμε τοπικές επιδράσεις της αλληλουχίας βάσεων, καταγράφουμε τις βάσεις που προηγούνται και ακολουθούν αμέσως τη μεσαία SBS, όπως προτάθηκε από τους Nik-Zainal *et al.* [31]. Με τέσσερις δυνατές βάσεις πριν και τέσσερις μετά, υπάρχουν

$$4 \times 4 = 16$$

διατεταγμένα ζεύγη πλαισίων (flank-pairs). Συνδυάζοντάς τα με τις έξι κεντρικές SBS προκύπτουν

$$6 \times 16 = 96$$

διακριτές καταστάσεις τρινουκλεοτιδίων. Παράδειγμα SBS ενώ λαμβάνουμε υπόψιν το πλάισιο:

$$\mathrm{A}[\,\mathrm{C} \to \mathrm{T}\,]\mathrm{G},$$

όπου η αλλαγή C→T συμβαίνει με A πριν και G μετά.

Μήτρα Δεδομένων V Η μήτρα δεδομένων που χρησιμοποιείται στα πειράματά μας —και σε προγενέστερες μελέτες— αποτελείται από m δείγματα, το καθένα αναπαριστώμενο από ένα προφίλ καταμετρήσεων των 96 μεταλλάξεων:

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{961} & v_{962} & \cdots & v_{96m} \end{bmatrix} \in \mathbb{R}^{96\times m},$$

όπου v_{ij} είναι ο αριθμός εμφανίσεων της i-στης μετάλλαξης στο δείγμα j και m ο συνολικός αριθμός δειγμάτων.

Μήτρες Υπογραφών και Έκθεσης Στην εποπτευόμενη εκδοχή του DNMF, τα πειράματα διεξήχθησαν σε προσομοιωμένες μήτρες $\mathbf{W}$ και $\mathbf{H}$. Αυτές οι μήτρες παραγόντες και ο βαθμός παραγοντοποίησης p ελήφθησαν μέσω NMF στη μελέτη των Alexandrov *et al.* [14]. Έτσι, έχουμε την κλασική παραγοντοποίηση:

$$\mathbf{V} \approx \mathbf{W}\,\mathbf{H},$$

με

$$\mathbf{W} \in \mathbb{R}_{\geq 0}^{96\times p}, \quad \mathbf{H} \in \mathbb{R}_{\geq 0}^{p\times m}.$$

Η μήτρα βάσεων $\mathbf{W}$ ονομάζεται σε αυτή την εφαρμογή *μήτρα υπογραφών* (signature matrix). Ορίζουμε μια μεταλλακτική υπογραφή ως διακριτή κατανομή πιθανότητας πάνω στις 96 πιθανές μεταλλάξεις. Ο βαθμός p που προέκυψε από τις προσομοιώσεις αντιστοιχεί στον συνολικό αριθμό υπογραφών εξαγόμενων από τα δεδομένα μας. Η δομή της μήτρας υπογραφών $\mathbf{W}$ γράφεται ως:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1p} \\ w_{21} & w_{22} & \cdots & w_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ w_{961} & w_{962} & \cdots & w_{96p} \end{bmatrix} \in \mathbb{R}^{96\times p}.$$

Κάθε υπογραφή μπορεί να θεωρηθεί ως διακριτή κατανομή πιθανότητας στις 96 καταστάσεις τρινουκλεοτιδίων, οπότε το στοιχείο w_{ij} αντιστοιχεί στην πιθανότητα η j-οστή υπογραφή να περιλαμβάνει την i-στή μετάλλαξη. Κατά συνέπεια, για κάθε στήλη j πρέπει:

$$\sum_{i=1}^{96} w_{ij} = 1.$$

Η μήτρα *εκθέσεων* $\mathbf{H}$ κωδικοποιεί την έκθεση (exposure) κάθε υπογραφής σε κάθε δείγμα, δηλαδή τον αριθμό (ή το ποσοστό) των μεταλλάξεων ανά υπογραφή. Συγκεκριμένα:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1m} \\ h_{21} & h_{22} & \cdots & h_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ h_{p1} & h_{p2} & \cdots & h_{pm} \end{bmatrix} \in \mathbb{R}^{p\times m}.$$

Στη μήτρα εκθέσεων, κάθε καταχώριση h_{ij} δηλώνει τον αριθμό μεταλλάξεων στο δείγμα j που αποδίδονται στην i-οστή υπογραφή.

1.7 Προτεινόμενες Μέθοδοι

1.7.1 DEQ-NMF

Στην παρούσα υποενότητα παρουσιάζουμε την υλοποίηση ενός μοντέλου DEQ για την NMF, το οποίο ονομάζεται DEQ-NMF. Σε αυτό το πλαίσιο, διατυπώνουμε τη μήτρα συντελεστών $\mathbf{H}$ ως το σταθερό σημείο ενός μη γραμμικού χάρτη ανανέωσης με εκπαιδεύσιμες παραμέτρους. Χρησιμοποιώντας τον κλασικό πολλαπλασιαστικό κανόνα ανανέωσης για την $\mathbf{H}$ και διατηρώντας τις παραμέτρους δικτύου που προτάθηκαν στο DNMF, ορίζουμε την εξίσωση ισορροπίας:

$$\mathbf{H}^\star = f(\mathbf{H}^\star) = \mathbf{H}^\star \odot \frac{\mathbf{A}\,\mathbf{V}}{(\mathbf{B} + \lambda_2 \mathbf{I})\,\mathbf{H}^\star + \lambda_1}.$$

Forward Pass. Όπως υποδεικνύει η αρχή των DEQs, χρησιμοποιούμε έναν αριθμητικό επιλυτή για να εντοπίσουμε το σταθερό σημείο $\mathbf{H}^\star$ που ικανοποιεί $f(\mathbf{H}^\star) = \mathbf{H}^\star$. Συγκεκριμένα, δεδομένης της εισόδου $\mathbf{V}$ και των τρεχουσών παραμέτρων, η μήτρα ισορροπίας $\mathbf{H}^\star$ υπολογίζεται με επαναληπτική εφαρμογή του χάρτη f:

$$\mathbf{H}^{(t+1)} = f\big(\mathbf{H}^{(t)}\big),$$

ξεκινώντας από αρχικοποίηση $\mathbf{H}^{(0)}$. Η επανάληψη εκτελείται για σταθερό πλήθος βημάτων (π.χ. $T = 50$) ή έως ότου ικανοποιηθούν κριτήρια σύγκλισης, ώστε να προσεγγιστεί το σταθερό σημείο:

$$\mathbf{H}^\star \approx \mathbf{H}^{(T)}.$$

Η επιλογή του επιλυτή στο forward pass δεν επηρεάζει την παραγώγιση και το backpropagation στο πλαίσιο των DEQ.

Backward Pass. Αφού αποκτήσουμε το $\mathbf{H}^\star$ στο forward pass, υλοποιούμε έναν προσαρμοσμένο υπολογισμό των παραγώγων για το DEQ layer. Αντί να χρειαζόμαστε τις παραγώγους για όλες τις επαναλήψεις εύρεσης του σταθερού σημείου έτσι ώστε να κάνουμε backpropagate, αξιοποιούμε το Θεώρημα Implicit Συναρτήσεων για να υπολογίσουμε τις παραγώγους απευθείας στο σημείο ισορροπίας.

Δεδομένης της συνάρτησης απώλειας $\mathcal{L}$, η παράγωγος ως προς τις παραμέτρους του μοντέλου θ (π.χ. $\mathbf{A}, \mathbf{B}, \lambda_1, \lambda_2$) υπολογίζεται μέσω του κανόνα αλυσίδας:

$$\frac{d\mathcal{L}}{d\theta} = \frac{\partial \mathcal{L}}{\partial \mathbf{H}^\star} \cdot \frac{d\mathbf{H}^\star}{d\theta},$$

όπου $\mathbf{H}^\star$ ικανοποιεί την εξίσωση σταθερού σημείου $f(\mathbf{H}^\star; \theta) = \mathbf{H}^\star$.

Εφαρμόζοντας το Θεώρημα Implicit Συναρτήσεων καταλήγουμε στο:

$$\frac{d\mathbf{H}^\star}{d\theta} = \Big[I \; - \; \frac{\partial f}{\partial \mathbf{H}}\big|_{\mathbf{H}^\star} \Big]^{-1} \frac{\partial f}{\partial \theta}\big|_{\mathbf{H}^\star}.$$

Αντί να αντιστρέφουμε απευθείας αυτήν την (πιθανώς μεγάλη) Ιακωβιανή, ορίζουμε την *ενδιάμεση μεταβλητή* $\mathbf{u}$ ως τη λύση του επαναληπτικού συστήματος

$$\mathbf{u} = \frac{\partial \mathcal{L}}{\partial \mathbf{H}^\star} + \mathbf{u} \; \cdot \; \frac{\partial f}{\partial \mathbf{H}}\big|_{\mathbf{H}^\star},$$

εκτελώντας λίγα βήματα σταθερού σημείου (τυπικά 10). Αυτό ισοδυναμεί με την επίλυση της γραμμικής εξίσωσης

$$\Big[I \; - \; \frac{\partial f}{\partial \mathbf{H}} \Big]^\top \mathbf{u} = \frac{\partial \mathcal{L}}{\partial \mathbf{H}^\star}$$

για την $\mathbf{u}$, την οποία στη συνέχεια χρησιμοποιούμε για τον υπολογισμό των παραγώγων των μεταβλητών του δικτύου:

$$\frac{d\mathcal{L}}{d\theta} = \mathbf{u}^\top \; \cdot \; \frac{\partial f}{\partial \theta}\big|_{\mathbf{H}^\star}.$$

Στον κώδικά μας, η διαδικασία αυτή υλοποιείται αποδοτικά με την αυτόματη παραγώγιση του JAX και τη λειτουργία `custom_vjp`, επιτρέποντάς μας να καθορίσουμε την παραπάνω implicit λογική backpropagation. Τα γινόμενα διάνυσματος–Ιακωβιανής υπολογίζονται χωρίς ρητή κατασκευή των μεγάλων Ιακωβιανών, επιτυγχάνοντας κλιμακώσιμη και αποδοτική σε μνήμη εκπαίδευση.

DEQ-NMF με Κοινή Μήτρα Βάσης και Παραμέτρους Κανονικοποίησης. Όπως αναφέρθηκε προηγουμένως, το δίκτυο DNMF αγνοεί την εξάρτηση μεταξύ των όρων $\mathbf{W}^\top$ και $\mathbf{W}^\top\mathbf{W}$, αντιμετωπίζοντάς τους ως ανεξάρτητες εκπαιδεύσιμες παραμέτρους $\mathbf{A}$ και $\mathbf{B}$. Στο πρόβλημα NMF: $\mathbf{V} \approx \mathbf{W}\,\mathbf{H}$, επιδιώκουμε να βρούμε τις πραγματικές μήτρες $\mathbf{W}$ και $\mathbf{H}$. Ωστόσο, κατά τα πειράματά μας διαπιστώσαμε ότι τόσο το DNMF όσο και η προτεινόμενη μέθοδός μας απέτυχαν να εκτιμήσουν με ακρίβεια τη μήτρα βάσης $\mathbf{W}$, δηλαδή την παράμετρο μοντέλου $\mathbf{A}^\top$. Συνεπώς, παρότι τα δίκτυα αυτά παρείχαν ικανοποιητικές εκτιμήσεις της μήτρας συντελεστών $\mathbf{H}$, το συνολικό σφάλμα ανακατασκευής

$$\big\|\mathbf{V}_{\text{true}} - \mathbf{A}^\top \mathbf{H}_{\text{pred}}\big\|_F^2$$

παρέμενε υψηλό. Φυσικά αναρωτηθήκαμε ποια θα ήταν η συμπεριφορά του μοντέλου μας εάν χρησιμοποιούσαμε μόνο μία εκπαιδεύσιμη μήτρα. Γι' αυτό, εισάγουμε μια απλοποιημένη παραλλαγή, ονόματι DEQ-NMF-A, όπου ο

χάρτης ενημέρωσης NMF παραμετροποιείται αποκλειστικά από μία μήτρα βάσης $\mathbf{A}$ και τους βαθμωτούς συντελεστές κανονικοποίησης λ_1, λ_2. Σε αυτή τη θεώρηση, η εξίσωση σταθερού σημείου απλοποιείται σε:

$$\mathbf{H}^\star = \mathbf{H}^\star \odot \frac{\mathbf{A}\,\mathbf{V}}{\left(\mathbf{A}\,\mathbf{A}^\top\right)\mathbf{H}^\star + \lambda_2\,\mathbf{H}^\star + \lambda_1},$$

Οι διαδικασίες εκπαίδευσης και παραγώγισης είναι πλήρως ανάλογες με τις περιγραφείσες παραπάνω, με τη μοναδική διαφορά ότι η $\mathbf{B}$ δένεται ως $\mathbf{A}^\top\mathbf{A}$ και μόνο οι $\mathbf{A}$, λ_1 και λ_2 εκπαιδεύονται. Αυτή η παραλλαγή μειώνει τον αριθμό των παραμέτρων και επιβάλλει κλασική δομή NMF, ενώ εξακολουθεί να επιτρέπει end-to-end εκπαίδευση με implicit παραγώγιση μέσω του σταθερού σημείου. Τελικά, τα πειράματά μας επιβεβαίωσαν ότι αυτή η στρατηγική κοινής χρήσης παραμέτρων οδήγησε σε υψηλότερη ακρίβεια τόσο στην εκτίμηση της $\mathbf{H}_{\text{true}}$ όσο και στην ανακατασκευή της $\mathbf{V}_{\text{true}}$, σε σύγκριση με την μη-περιορισμένη προσέγγιση.

1.7.2 Σχήματα Προσέγγισης του Σημείου Ισορροπίας

Στην υποενότητα αυτή εισάγουμε τρία μοντέλα προσέγγισης του σημείου ισορροπίας (equilibrium-approximation), βασισμένα στη DEQ θεώρηση της εποπτευόμενης NMF. Κάθε ένα από τα προτεινόμενα μοντέλα αντικαθιστά την τυπική επαναληπτική πολυ-επίπεδη διαδικασία με μία μόνο κλειστής έκφρασης ReLU εκτίμηση του σημείου ισορροπίας, επιτρέποντας ρητή παραγώγιση στο backward pass. Στη συνέχεια διαφοροποιούμε τις παραλλαγές ανάλογα με τα εξειδικευμένα δίκτυα παραμέτρων και τις προσαρμοσμένες συναρτήσεις κόστους. Όλες οι παραλλαγές μειώνουν τη χρήση μνήμης και παρέχουν σημαντικές βελτιώσεις στην υπολογιστική αποδοτικότητα και την ακρίβεια εκτίμησης σε σύγκριση με τις κλασικές μεθόδους deep unrolling και MU-based NMF.

Ξεκινάμε με το A-DEQ-NMF [32], ένα μοντέλο που διατηρεί τις αρχικές εκπαιδεύσιμες παραμέτρους δικτύου του DNMF, και παρουσιάζουμε λεπτομερώς τη διαδικασία εκπαίδευσης που στοχεύει στην εκμάθηση των πραγματικών $\mathbf{H}_{\text{true}}$ και $\mathbf{W}_{\text{true}}$.

Forward Pass. Στο forward pass επιδιώκουμε να λύσουμε την εξίσωση για το σημείο ισορροπίας του βαθιού δικτύου. Προηγουμένως ορίσαμε τη διαδικασία με την οποία τα DEQ βρίσκουν το σταθερό σημείο της συνάρτησης f, προσομοιώνοντας ένα δίκτυο με άπειρα επίπεδα, μέσω αριθμητικού επιλυτή. Στην περίπτωσή μας, χρησιμοποιώντας την προσέγγιση στήλης, η συνάρτηση ενεργοποίησης που εφαρμόζεται σε κάθε επίπεδο είναι:

$$f(\mathbf{h};\mathbf{v}) = \mathbf{h} \odot \frac{\mathbf{A}\,\mathbf{v}}{\mathbf{B}\,\mathbf{h} \;+\; \lambda_1\,\mathbf{1} \;+\; \lambda_2\,\mathbf{h}}. \tag{1.7.1}$$

Βάσει των ιδεών των DEQs, αντικαθιστούμε στη (1.7.1) το σημείο ισορροπίας $\mathbf{h}^*$ και προσπαθούμε να το ορίσουμε ρητώς:

$$f(\mathbf{h}^*) = \mathbf{h}^* \;\longrightarrow\; \mathbf{h}^* = \mathbf{h}^* \odot \frac{\mathbf{A}\,\mathbf{v}}{\mathbf{B}\,\mathbf{h}^* \;+\; \lambda_1\,\mathbf{1} \;+\; \lambda_2\,\mathbf{h}^*}. \tag{1.7.2}$$

Λόγω της φύσης της NMF, έχουμε ήδη υποθέσει ότι όλα τα στοιχεία της $\mathbf{h}^*$ είναι μη-αρνητικά. Δυστυχώς, δεν μπορούμε να προχωρήσουμε στην εξαγωγή κλειστής έκφρασης για το σημείο ισορροπίας χωρίς να δεχτούμε ότι το σταθερό σημείο $\mathbf{h}^*$ δεν περιέχει μηδενικά στοιχεία. Επομένως, υιοθετώντας αυτήν την υπόθεση, η Εξ. (1.7.2) μπορεί να γραφεί εκ νέου ως:

$$\mathbf{B}\,\mathbf{h}^* + \lambda_1\,\mathbf{1} + \lambda_2\,\mathbf{h}^* = \mathbf{A}\,\mathbf{v}, \quad \mathbf{h}^* > 0, \tag{1.7.3}$$

ή ισοδύναμα

$$(\mathbf{B} + \lambda_2\,\mathbf{I}_p)\,\mathbf{h}^* = \mathbf{A}\,\mathbf{v} - \lambda_1\,\mathbf{1}, \quad \mathbf{h}^* > 0. \tag{1.7.4}$$

Λόγω των θετικών περιορισμών $\mathbf{h}^* > 0$, δεν είναι εφικτό να αποκτήσουμε αναλυτική έκφραση της λύσης. Για να παρακάμψουμε την ανάγκη επαναληπτικών διαδρομών και να διατηρήσουμε τη μη-αρνητικότητα, εφαρμόζουμε ReLU στο τέλος του forward pass και προτείνουμε την ακόλουθη κλειστή εκτίμηση τύπου ReLU, που προκύπτει άμεσα από την (1.7.4):

$$\hat{\mathbf{h}}^* = \max\Big[(\mathbf{B} + \lambda_2\,\mathbf{I}_p)^{-1}\big(\mathbf{A}\,\mathbf{v} - \lambda_1\,\mathbf{1}\big),\, 0\Big], \tag{1.7.5}$$

ή σε μορφή μήτρας:

$$\hat{\mathbf{H}}^* = \max\Big[(\mathbf{B} + \lambda_2 \mathbf{I}_p)^{-1}\big(\mathbf{A}\,\mathbf{V} - \lambda_1 \mathbf{1}_{p\times m}\big),\, 0\Big], \tag{1.7.6}$$

όπου το μέγιστο λαμβάνεται ανά στοιχείο. Σημειώστε ότι αυτή η προσέγγιση λύνει το γραμμικό σύστημα σε ένα μόνο βήμα, σε αντίθεση με προηγούμενα forward pass, όπου ο αριθμός των πράξεων εξαρτάται από τον αριθμό επιπέδων του δικτύου.

Backward Pass. Προηγουμένως εξηγήσαμε το backward pass στα κλασικά DEQ δίκτυα. Θυμίζουμε ότι, επειδή το σημείο ισορροπίας προκύπτει μέσω "black-box" επιλυτή ριζών, δεν διαθέτουμε αναλυτική έκφρασή του και πρέπει να χρησιμοποιούμε implicit τεχνικές παραγώγισης. Ωστόσο, στην προσέγγισή μας καταλήξαμε σε κλειστή έκφραση για το $\mathbf{h}^*$ και επομένως μπορούμε να εφαρμόσουμε τυπικές μεθόδους ρητής παραγώγισης. Στο backward step χρησιμοποιούμε κλασσικό backpropagation με ADAM για την ενημέρωση των παραμέτρων $\mathbf{A}, \mathbf{B}, \lambda_1, \lambda_2$, διατηρώντας τη συνάρτηση απώλειας που ορίστηκε στο DNMF:

$$\mathcal{L} = \mathrm{MSE}\big(\mathbf{h}^*,\, \mathbf{h}'\big) = \frac{1}{n}\sum_{i=1}^{n}\big(h_i^* - h_i'\big)^2,$$

όπου $\mathbf{h}'$ είναι στήλη της πραγματικής μήτρας συντελεστών. Η απώλεια μπορεί επίσης να γραφεί σε μορφή μητρώων.

Στην πράξη, βασιζόμαστε σε βελτιστοποιημένες βιβλιοθήκες Python για τον υπολογισμό των παραγώγων και τις ανανεώσεις των παραμέτρων. Παρ' όλα αυτά, παρέχουμε αναλυτικές εκφράσεις για τις ανανεώσεις παραμέτρων χρησιμοποιώντας τις ρητές παραγώγους της λύσης ισορροπίας. Παρακάτω δίνουμε τις ανανεώσεις των παραμέτρων με βάση τον αλγόριθμο κατάβασης δυναμικού σε συμπαγή μητρική μορφή. Η βήμα-βήμα απόδειξη από την προσέγγιση μίας μόνο στήλης παρέχεται στο Παράρτημα (Appendix).

$$\mathbf{A}^+ = \mathbf{A} - 2\,\alpha\,\Big[\mathbf{M}^{-1}\,(\mathbf{H}^* - \mathbf{H}_{\text{true}})\Big)\,\mathbf{V}^\top \tag{1.7.7}$$

$$\mathbf{B}^+ = \mathbf{B} + 2\,\alpha\,\Big[\mathbf{M}^{-1}(\mathbf{H}^* - \mathbf{H}_{\text{true}})\Big)\,\big(\mathbf{M}^{-1}\mathbf{Y}\big)^\top \tag{1.7.8}$$

$$\lambda_1^+ = \lambda_1 + 2\,\alpha\,\mathrm{tr}\Big[(\mathbf{H}^* - \mathbf{H}_{\text{true}})^\top(\mathbf{M}^{-1}\mathbf{1}_p\,\mathbf{1}_m^\top)\Big] \tag{1.7.9}$$

$$\lambda_2^+ = \lambda_2 + 2\,\alpha\,\mathrm{tr}\Big[(\mathbf{H}^* - \mathbf{H}_{\text{true}})^\top\mathbf{M}^{-2}\,\mathbf{Y}\Big] \tag{1.7.10}$$

όπου

$$\mathbf{M} = \mathbf{B} + \lambda_2\,\mathbf{I}_p \;\in\; \mathbb{R}^{p\times p}, \quad \mathbf{Y} = \mathbf{A}\,\mathbf{V} + \lambda_1\,\mathbf{1}_p\,\mathbf{1}_m^\top \;\in\; \mathbb{R}^{p\times m}.$$

Κατά τα πειράματά μας υλοποιήσαμε αυτή τη μέθοδο και την συγκρίναμε με άλλα μοντέλα. Δυστυχώς, διαπιστώσαμε ότι η χρήση του αλγορίθμου της κατάβασης δυναμικού δεν αρκεί για ανταγωνιστική απόδοση, σε σύγκριση με πιο προηγμένους βελτιστοποιητές όπως ο ADAM.

Στην προσπάθειά μας να ενσωματώσουμε την ιδέα ενός implicit επιπέδου από τα DEQs στο δίκτυο DNMF, καταλήξαμε σε μια κλειστή έκφραση που προσεγγίζει τη λύση ισορροπίας. Όπως μπορεί να παρατηρήσει κανείς, το μοντέλο εποπτευόμενης NMF που παρουσιάσαμε δεν είναι παρά μια επαναληπτική διαδικασία, που αξιοποιεί όμως τον optimizer ADAM αντί των κλασικών MU ενημερώσεων για την $\mathbf{W}$. Παρ' όλα αυτά, η μέθοδος αυτή επιτυγχάνει σημαντική μείωση στον αριθμό των παραμέτρων σε σχέση με το αρχικό DNMF και, όπως δείχνουν τα πειράματά μας, προσφέρει υψηλότερη ακρίβεια συγκριτικά με το DNMF και τον κλασικό αλγόριθμο MU.

A-DEQ-NMF με μόνο τη παράμετρο A. Ακολουθώντας την ίδια ιδέα που οδήγησε στην υλοποίηση του DEQ-NMF-A, παρουσιάζουμε μια τροποποιημένη έκδοση, ονόματι A-DEQ-NMF-A, όπου θεωρούμε την $\mathbf{A}$ ως μοναδική εκπαιδεύσιμη μήτρα παραμέτρων.

Forward Pass Στο forward pass, η μόνη αλλαγή είναι ότι υπολογίζουμε το $\mathbf{W}^\top\mathbf{W}$ μέσω της $\mathbf{A}$. Συγκεκριμένα:

$$\hat{\mathbf{H}}^* = \max\Big[(\mathbf{A}\mathbf{A}^\top + \lambda_2\,\mathbf{I}_p)^{-1}(\mathbf{A}\,\mathbf{V} - \lambda_1\,\mathbf{1}_{p\times m}),\ 0\Big]. \tag{1.7.11}$$

Backward Pass. Αντίστοιχα, στο Backward Pass δεν υπολογίζουμε πλέον τις παραγώγους ως προς την $\mathbf{B}$, αλλά χρειαζόμαστε την νέα πιο πολύπλοκη παράγωγο της $\hat{\mathbf{H}}^*$ ως προς την $\mathbf{A}$.

Στη συνέχεια, ενημερώνουμε τις παραμέτρους $\mathbf{A}, \lambda_1, \lambda_2$ με την ίδια διαδικασία που περιγράφηκε παραπάνω. Στα πειράματά μας μετρήσαμε το NSE μεταξύ των πραγματικών και των προβλεπόμενων μητρώων συντελεστών, επιβεβαιώνοντας τη βελτίωση της ακρίβειας των μοντέλων μας όταν δεν παραβλέπουμε την εξάρτηση μεταξύ $\mathbf{A}$ και $\mathbf{B}$.

A-DEQ-NMF-A με όρο απώλειας ανακατασκευής. Για να τιμωρήσουμε απευθείας σφάλματα στην ανακατασκευή της $\mathbf{V}$, επεκτείνουμε το προηγούμενο μοντέλο με έναν ρητό όρο απώλειας ανακατασκευής, και ονομάζουμε την παραλλαγή A-DEQ-NMF-A1. Πιο συγκεκριμένα, μαζί με την απώλεια εκτίμησης ισορροπίας, εισάγουμε τη συνάρτηση απωλειών:

$$\mathcal{L}_{\text{rec}} = \big\|\mathbf{V}_{\text{true}} - \mathbf{A}^\top\,\hat{\mathbf{H}}^*\big\|_F^2,$$

και βελτιστοποιούμε τη συνδυαστική αντικειμενική συνάρτηση:

$$\mathcal{L} = \big\|\hat{\mathbf{H}}^* - \mathbf{H}_{\text{true}}\big\|_F^2 \;+\; \alpha\,\mathcal{L}_{\text{rec}},$$

όπου ο παράγοντας $\alpha > 0$ ισορροπεί τους όρους απώλειας συντελεστών και ανακατασκευής.

Forward Pass. Όπως πριν, υπολογίζουμε:

$$\hat{\mathbf{H}}^* = \max\Big[(\mathbf{A}\mathbf{A}^\top + \lambda_2\,\mathbf{I}_p)^{-1}(\mathbf{A}\,\mathbf{V} - \lambda_1\,\mathbf{1}_{p\times m}),\ 0\Big],$$

και στη συνέχεια:

$$\widehat{\mathbf{V}} = \mathbf{A}^\top\,\hat{\mathbf{H}}^*.$$

Backward Pass και Ανανεώσεις Παραμέτρων. Η συνεισφορά παραγώγων ως προς τη $\mathbf{H}^*$ προκύπτει από δύο όρους:

$$\underbrace{\nabla_{\mathbf{H}^*}\big\|\mathbf{H}^* - \mathbf{H}_{\text{true}}\big\|_F^2}_{\text{σφάλμα μήτρας συντελεστών}} = 2\big(\mathbf{H}^* - \mathbf{H}_{\text{true}}\big),$$

$$\underbrace{\nabla_{\mathbf{H}^*}\big\|\mathbf{V} - \mathbf{A}^\top\mathbf{H}^*\big\|_F^2}_{\text{σφάλμα ανακατασκευής}} = -2\,\mathbf{A}\,\big(\mathbf{V} - \mathbf{A}^\top\mathbf{H}^*\big).$$

Επομένως η συνολική παράγωγος είναι:

$$\nabla_{\mathbf{H}^*}\,\mathcal{L} = 2\big[\big(\mathbf{H}^* - \mathbf{H}_{\text{true}}\big) \;-\; \alpha\,\mathbf{A}\,\big(\mathbf{V} - \mathbf{A}^\top\mathbf{H}^*\big)\big].$$

Στα πειράματά μας, αυτή η υβριδική συνάρτηση απώλειας οδήγησε σε ακόμη χαμηλότερο σφάλμα $\|\mathbf{V} - \mathbf{A}^\top\hat{\mathbf{H}}^*\|_F^2$ και βελτίωσε την ακρίβεια της εκμάθησης της $\mathbf{A}^\top$ (μήτρας βάσης).

1.8 Αξιολόγηση Απόδοσης των Προτεινόμενων Αλγορίθμων

Σε αυτή την ενότητα παρουσιάζουμε μια σειρά πειραμάτων που πραγματοποιήθηκαν τόσο σε προσομοιωμένα δεδομένα μεταλλάξεων όσο και σε συνθετικά δεδομένα με θόρυβο, προκειμένου να επιδείξουμε τη βελτιωμένη απόδοση των προτεινόμενων μοντέλων μας. Στόχος μας είναι να δείξουμε ότι όχι μόνο εκτιμούν με μεγαλύτερη

ακρίβεια τη μήτρα συντελεστών $\mathbf{H}_{\text{true}}$, αλλά και επιτυγχάνουν πολύ χαμηλό σφάλμα ανακατασκευής στη $\mathbf{V}_{\text{true}}$. Η μετρική που χρησιμοποιούμε είναι το *κανονικοποιημένο τετραγωνικό σφάλμα* (NSE) σε dB, ορισμένο ως

$$\text{NSE}(\mathbf{X}_{\text{true}}, \mathbf{X}_{\text{pred}}) = 10 \log_{10} \left[\frac{\|\mathbf{X}_{\text{true}} - \mathbf{X}_{\text{pred}}\|_F^2}{\|\mathbf{X}_{\text{true}}\|_F^2} \right]. \tag{1.8.1}$$

Όλα τα πειράματα ακολουθούν πρωτόκολλο cross-validation με 5 folds και με αναλογία εκπαίδευσης–δοκιμής (train-test) 80:20 σε κάθε fold. Τα αποτελέσματα που παρουσιάζουμε στους πίνακες και τις εικόνες είναι οι μέσες τιμές από τα 5 folds. Εκπαιδεύουμε κάθε μοντέλο για 800 εποχές με ρυθμό μάθησης 0.01. Για δίκαιη σύγκριση με τον επαναληπτικό αλγόριθμο MU, ακολουθούμε τη λογική που προϋπάρχει στο [29]: εκτελούμε 800 ανανεώσεις για τη $\mathbf{W}_{\text{MU}}$ χρησιμοποιώντας τον πραγματικό πίνακα $\mathbf{H}_{\text{true}}$ και εν συνεχεία 10 ανανεώσεις για το $\mathbf{H}_{\text{MU}}$ με τη «μαθημένη» $\mathbf{W}_{\text{MU}}$. Αρχικοποιούμε επίσης τις παραμέτρους κανονικοποίησης λ_1, λ_2 όλες στο 1. Στους πίνακες που ακολουθούν χρησιμοποιούμε έντονη γραφή για το μικρότερο σφάλμα και υπογράμμιση για το δεύτερο μικρότερο όπου απαιτείται.

Πειράματα με Προσομοιωμένα Δεδομένα Μεταλλάξεων. Αρχικά, διεξάγουμε πειράματα σε 12 προσομοιωμένα σύνολα μεταλλάξεων από το [14], τα οποία περιγράψαμε προηγουμένως. Συγκρίνουμε τις μεθόδους μας με τον κλασικό επαναληπτικό αλγόριθμο MU και το DNMF, μετρώντας τόσο την ακρίβεια εκτίμησης της $\mathbf{H}_{\text{true}}$ όσο και το σφάλμα ανακατασκευής της $\mathbf{V}_{\text{true}}$ στα σύνολα εκπαίδευσης και δοκιμής. Όπως φαίνεται στον Πίνακα 1.1, το A-DEQ-NMF υπερέχει σαφώς τόσο έναντι του MU όσο και του DNMF, επιτυγχάνοντας το χαμηλότερο NSE σε κάθε split εκπαίδευσης, αποδεικνύοντας την ικανότητα του να μάθει τη δομή των δεδομένων.

Δεδομένα	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
1	-0.38	-21.40	-2.08	-5.15	**-32.98**	-1.46	-1.56
2	-1.16	-21.53	-7.36	-6.80	**-25.10**	-3.21	-3.60
3	-0.20	-22.26	-8.08	-9.30	**-28.54**	-2.02	-1.11
4	-0.94	-25.21	-3.01	-4.37	**-25.28**	-11.57	-11.91
5	-0.26	-21.28	-7.80	-6.00	**-31.69**	-2.43	-1.61
6	-0.49	-22.13	-4.07	-5.52	**-28.87**	-3.07	-3.51
7	-0.23	-26.82	-16.11	-9.63	**-51.87**	-1.92	-1.01
8	-0.24	-32.63	-32.27	-17.17	**-50.17**	-1.97	-2.13
9	-1.15	-28.02	-13.57	-12.28	**-35.75**	-14.65	-13.63
10	-0.81	-29.08	-10.13	-12.11	**-35.84**	-14.54	-15.78
11	-1.15	-28.53	-24.31	-19.69	**-30.92**	-18.64	-23.03
12	-1.14	-32.74	-23.66	-19.54	**-41.84**	-21.77	-24.14

Table 1.1: Μέσο NSE($\mathbf{H}_{\text{true}}, \mathbf{H}_{\text{pred}}$) στα σύνολα εκπαίδευσης — Προσομοιωμένα Δεδομένα Μεταλλάξεων.

Αφού διαπιστώσαμε ότι το A-DEQ-NMF επιτυγχάνει το χαμηλότερο NSE στα σύνολα εκπαίδευσης, αξιολογούμε στη συνέχεια την ικανότητά του να γενικεύει στα σύνολα δοκιμής. Σε κάθε εποχή υπολογίζουμε την εκτίμηση $\mathbf{H}_{\text{pred}}$ στα δεδομένα δοκιμής και παρουσιάζουμε τα αποτελέσματα μετά την τελευταία εποχή στον Πίνακα 1.2. Και πάλι, το A-DEQ-NMF υπερέχει τόσο του MU όσο και του DNMF σε κάθε σύνολο, αποδεικνύοντας τη σημαντική του ικανότητα γενίκευσης.

Δεδομένα	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
1	-0.39	-20.28	-2.38	-5.13	**-31.46**	-1.49	-1.53
2	-1.16	-20.56	-7.17	-6.60	**-24.79**	-3.12	-3.56
3	-0.42	-10.85	0.13	-2.60	**-20.57**	-1.71	-1.22
4	-0.96	-20.28	-2.20	-3.20	**-21.84**	-8.87	-9.17
5	-0.25	-20.70	-7.73	-5.51	**-29.73**	-2.38	-1.49
6	-0.50	-21.18	-3.83	-5.58	**-28.55**	-3.09	-3.67
7	-0.23	-26.72	-16.59	-9.95	**-51.74**	-1.91	-1.00
8	-0.24	-32.22	-32.23	-17.32	**-50.09**	-1.99	-2.28
9	-1.15	-26.03	-10.61	-11.08	**-34.40**	-13.62	-13.49
10	-0.83	-23.71	-8.84	-11.50	**-30.24**	-12.89	-14.41
11	-1.17	-27.10	-22.43	-18.10	**-30.49**	-17.27	-22.02
12	-1.15	-29.95	-22.60	-15.97	**-39.89**	-20.62	-23.09

Table 1.2: Μέσο NSE($\mathbf{H}_{\text{true}}, \mathbf{H}_{\text{pred}}$) στα σύνολα δοκιμής — Προσομοιωμένα Δεδομένα Μεταλλάξεων.

Στη συνέχεια αξιολογούμε το σφάλμα ανακατασκευής μεταξύ της προβλεπόμενης μήτρας δεδομένων και της πραγματικής. Για τον MU ορίζουμε $\mathbf{V}_{\text{pred}} = \mathbf{W}_{\text{MU}}\,\mathbf{H}_{\text{MU}}$, ενώ για τα υπόλοιπα μοντέλα χρησιμοποιούμε την εκτιμημένη παράμετρο $\mathbf{A}$ (αντιστοιχεί σε $\mathbf{W}^\top$) και υπολογίζουμε το $\mathbf{V}_{\text{pred}} = \mathbf{A}^\top\,\mathbf{H}_{\text{pred}}$. Παρουσιάζουμε τα αποτελέσματα στα σύνολα εκπαίδευσης στον Πίνακα 1.3 και παρατηρούμε ότι το A-DEQ-NMF-A1 επιτυγχάνει το χαμηλότερο σφάλμα ανακατασκευής σε σχεδόν κάθε σύνολο, επιβεβαιώνοντας την ιδέα της προσθήκης όρου σφάλματος ανακατασκευής στην συνάρτηση απώλειας. Το δεύτερο καλύτερο μοντέλο, A-DEQ-NMF-A, υπερτερεί σημαντικά των εναλλακτικών, υπογραμμίζοντας ότι η μη-παράβλεψη της εξάρτησης μεταξύ $\mathbf{W}^\top$ και $\mathbf{W}^\top\mathbf{W}$ οδηγεί σε πιο ακριβή ανακατασκευή της μήτρας δεδομένων.

Δεδομένα	**MU**	**DNMF**	**DEQ-NMF**	**DEQ-NMF-A**	**A-DEQ-NMF**	**A-DEQ-NMF-A**	**A-DEQ-NMF-A1**
1	-1.64	35.88	34.32	-0.92	36.08	-6.13	**-8.44**
2	-2.04	34.48	25.70	-3.38	35.45	-9.25	**-11.95**
3	-1.85	37.44	36.18	-1.83	36.73	-8.52	**-9.37**
4	-2.17	28.39	25.92	-3.34	29.08	-17.98	**-20.61**
5	-1.04	37.32	33.32	-0.90	37.22	-8.16	**-9.48**
6	-1.85	37.22	29.93	-1.46	37.69	-11.91	**-12.22**
7	-0.33	39.23	34.74	-0.17	39.30	-8.27	**-9.77**
8	-0.32	38.33	33.40	-0.30	38.40	**-8.03**	-6.61
9	-3.00	24.04	20.37	-9.79	24.85	-15.81	**-24.47**
10	-4.76	24.86	20.09	-8.46	25.59	-15.45	**-31.39**
11	-2.98	25.54	21.17	-6.27	25.33	-12.64	**-27.35**
12	-2.88	23.95	16.92	-4.90	24.30	-8.99	**-37.13**

Table 1.3: Μέσο $\text{NSE}(\mathbf{V}_{\text{true}}, \mathbf{V}_{\text{pred}})$ στα σύνολα εκπαίδευσης — Προσομοιωμένα Δεδομένα Μεταλλάξεων.

Με παρόμοιο τρόπο παρουσιάζουμε τα αποτελέσματα στα σύνολα δοκιμής στον Πίνακα 1.4. Όπως μπορούμε να παρατηρήσουμε, το A-DEQ-NMF-A1 συνεχίζει να υπερέχει σημαντικά όλων των άλλων μεθόδων σε κάθε σύνολο, με το A-DEQ-NMF-A να έπεται ως δεύτερο καλύτερο μοντέλο.

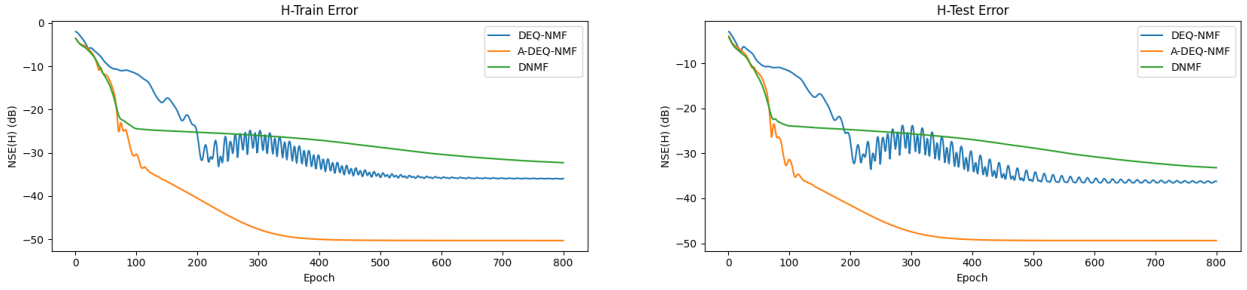
Δεδομένα	**MU**	**DNMF**	**DEQ-NMF**	**DEQ-NMF-A**	**A-DEQ-NMF**	**A-DEQ-NMF-A**	**A-DEQ-NMF-A1**
1	-1.64	35.85	34.46	-0.93	36.07	-6.23	**-8.51**
2	-2.04	34.47	25.79	-3.46	35.46	-9.32	**-11.88**
3	-2.10	33.19	32.39	-0.91	34.80	-6.98	**-7.92**
4	-2.20	28.91	23.74	-4.91	30.06	-16.81	**-19.67**
5	-1.06	37.53	33.57	-0.79	37.41	-8.65	**-9.54**
6	-1.85	37.20	29.75	-1.47	37.68	-11.92	**-12.11**
7	-0.33	39.23	34.76	-0.17	39.30	-8.27	**-9.75**
8	-0.32	38.33	33.35	-0.31	38.40	**-7.89**	-6.34
9	-2.97	24.62	20.47	-9.45	25.02	-15.78	**-22.98**
10	-4.72	25.42	20.00	-7.88	26.05	-14.93	**-29.34**
11	-2.97	26.19	22.64	-6.06	25.94	-12.48	**-26.14**
12	-2.88	24.56	17.02	-4.84	24.76	-8.92	**-36.04**

Table 1.4: Μέσο $\text{NSE}(\mathbf{V}_{\text{true}}, \mathbf{V}_{\text{pred}})$ στα σύνολα δοκιμής — Προσομοιωμένα Δεδομένα Μεταλλάξεων.

Επιπλέον, στο τρέχον πλαίσιο πειραμάτων παρατηρούμε ότι το A-DEQ-NMF δεν επιτυγχάνει μόνο το χαμηλότερο NSE για τις μήτρες συντελεστών, αλλά εισέρχεται σε καθεστώς ταχείας μείωσης του σφάλματος πολύ νωρίς κατά την εκπαίδευση. Αυτό αναδεικνύεται στην Εικ. 1.8.1, όπου παρουσιάζουμε το NSE σε κάθε εποχή για το μεταλλακτικό σύνολο 8.

Πειράματα με Συνθετικά Δεδομένα. Για να αναδείξουμε περαιτέρω την αποτελεσματικότητα των μοντέλων μας, διεξάγουμε πειράματα σε συνθετικά δεδομένα. Κατασκευάζουμε τις μήτρες $\mathbf{W}_{\text{true}} \in \mathbb{R}^{100\times 10}$ και $\mathbf{H}_{\text{true}} \in \mathbb{R}^{10\times 1000}$ με τυχαίες τιμές από ομοιόμορφη κατανομή στο διάστημα $[0, 1]$. Στη συνέχεια, παράγουμε τα συνθετικά δεδομένα μέσω του μοντέλου:

$$\mathbf{V}_{\text{true}} = \mathbf{W}_{\text{true}}\,\mathbf{H}_{\text{true}} + \mathbf{E},$$



(a) Σφάλμα εκπαίδευσης

(b) Σφάλμα δοκιμής

Figure 1.8.1: Κανονικοποιημένο Τετραγωνικό Σφάλμα (NSE) της $\mathbf{H}$ σε βάθος εποχών για το 8ο Προσομοιωμένο Σύνολο Δεδομένων.

όπου τα στοιχεία της $\mathbf{E}$ προέρχονται επίσης από ομοιόμορφη κατανομή στο $[0, c]$, ώστε να επιτυγχάνεται η επιθυμητή αναλογία σήματος προς θόρυβο (SNR). Στον Πίνακα 1.5 συνοψίζεται η απόδοση κάθε μοντέλου για διάφορες τιμές SNR. Παρατηρούμε ότι όλα τα μοντέλα προσέγγισης ισορροπίας επιτυγχάνουν πολύ χαμηλό NSE στα σύνολα εκπαίδευσης για μικρές τιμές SNR, ενώ η ακρίβεια των A-DEQ-NMF-A και A-DEQ-NMF-A1 βελτιώνεται περαιτέρω σε υψηλότερα SNR.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	-7.24	-6.39	-0.71	-5.46	-8.29	**-8.36**	-6.19
5	-7.57	-6.49	-1.40	-6.45	-9.67	**-10.22**	-9.22
10	-8.07	-6.58	-2.17	-9.07	-11.80	**-13.56**	-12.95
15	-8.49	-6.64	-2.92	-13.09	-14.23	**-18.01**	-17.53
20	-8.82	-6.78	-3.38	-14.91	-15.88	**-22.17**	-21.96

Table 1.5: Μέσο NSE($\mathbf{H}_{\text{true}}, \mathbf{H}_{\text{pred}}$) στα σύνολα εκπαίδευσης — Συνθετικά Δεδομένα.

Στον Πίνακα 1.6 παρουσιάζεται η συμπεριφορά των μοντέλων στα σύνολα δοκιμής. Όλες οι μέθοδοι διατηρούν παρόμοιες επιδόσεις, επιβεβαιώνοντας την ικανότητά τους να γενικεύουν σε νέα, άγνωστα δεδομένα.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	-7.00	-6.36	-0.64	-5.43	**-7.55**	-7.26	-6.45
5	-7.44	-6.48	-1.46	-6.31	**-9.16**	-9.12	-9.15
10	-8.02	-6.57	-2.16	-8.92	-11.52	-12.52	**-12.81**
15	-8.47	-6.62	-2.94	-12.90	-14.04	-16.90	**-17.40**
20	-8.81	-6.76	-3.39	-14.84	-15.75	-21.20	**-21.85**

Table 1.6: Μέσο NSE($\mathbf{H}_{\text{true}}, \mathbf{H}_{\text{pred}}$) στα σύνολα δοκιμής — Συνθετικά Δεδομένα.

Όπως και με τα benchmarks σφάλματος ανακατασκευής που λάβαμε στα προσομοιωμένα σύνολα δεδομένων, παρατηρούμε ξανά στους Πίνακες 1.7 και 1.8 ότι το A-DEQ-NMF-A1 επιδεικνύει ανώτερη απόδοση, επιτυγχάνοντας πολύ χαμηλότερο σφάλμα από τα υπόλοιπα βαθιά δίκτυα.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	-11.66	-0.84	-0.09	-11.01	-0.19	-11.08	**-12.26**
5	-14.35	-1.06	-0.14	-12.47	-0.32	-12.39	**-15.08**
10	-17.37	-1.26	-0.20	-13.31	-0.44	-12.62	**-18.64**
15	-20.05	-1.33	-0.25	-12.83	-0.58	-13.55	**-22.77**
20	-21.97	-1.78	-0.27	-10.38	-0.65	-11.35	**-27.21**

Table 1.7: Μέσο NSE($\mathbf{V}_{\text{true}}, \mathbf{V}_{\text{pred}}$) στα σύνολα εκπαίδευσης – Συνθετικά Δεδομένα.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	-11.62	-0.84	-0.09	-11.00	-0.19	-11.08	**-12.01**
5	-14.31	-1.06	-0.14	-12.48	-0.32	-12.39	**-14.91**
10	-17.33	-1.26	-0.20	-13.32	-0.44	-12.62	**-18.52**
15	-20.02	-1.33	-0.25	-12.82	-0.58	-13.55	**-22.65**
20	-21.96	-1.78	-0.27	-10.38	-0.65	-11.35	**-27.09**

Table 1.8: Μέσο NSE($\mathbf{V}_{\text{true}}, \mathbf{V}_{\text{pred}}$) στα σύνολα δοκιμής – Συνθετικά Δεδομένα.

Για τα συνθετικά δεδομένα έχουμε πλήρη γνώση της πραγματικής μήτρας $\mathbf{W}_{\text{true}}$ και μπορούμε να μετρήσουμε το σφάλμα μεταξύ αυτής και της παραμέτρου δικτύου $\mathbf{A}^\top$ για κάθε μοντέλο. Στην περίπτωση του MU χρησιμοποιούμε τη $\mathbf{W}_{\text{MU}}$ που προκύπτει από 800 επαναληπτικές MU ανανεώσεις. Στον Πίνακα 1.9 διαπιστώνουμε ότι ο MU επιτυγχάνει το χαμηλότερο NSE σε όλες τις περιπτώσεις, αφού χρησιμοποιεί άμεσες επαναληπτικές ανανεώσεις, ενώ το A-DEQ-NMF-A1 ακολουθεί στενά τη χαμηλότερη τιμή σφάλματος για ελαφρώς υψηλότερα SNR.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	**-1.98**	-1.23	-0.14	0.89	-0.38	-0.90	1.74
5	**-6.92**	-1.23	-0.17	-2.27	-0.53	-4.10	-3.95
10	**-11.92**	-1.22	-0.19	-5.45	-0.63	-5.76	-10.00
15	**-16.99**	-1.14	-0.22	-9.17	-0.75	-6.66	-15.71
20	**-21.87**	-1.41	-0.24	-8.67	-0.80	-6.55	-19.05

Table 1.9: Μέσο NSE($\mathbf{W}_{\text{true}}, \mathbf{W}_{\text{pred}}$) – Συνθετικά Δεδομένα.

Για λόγους πληρότητας, παρουσιάζουμε στην Εικ. 1.8.2 τις τιμές NSE σε dB για τη μήτρα συντελεστών και τη μήτρα δεδομένων στα σύνολα εκπαίδευσης και δοκιμής κατά τη συνολική διάρκεια της εκπαίδευσης για όλα τα μοντέλα.

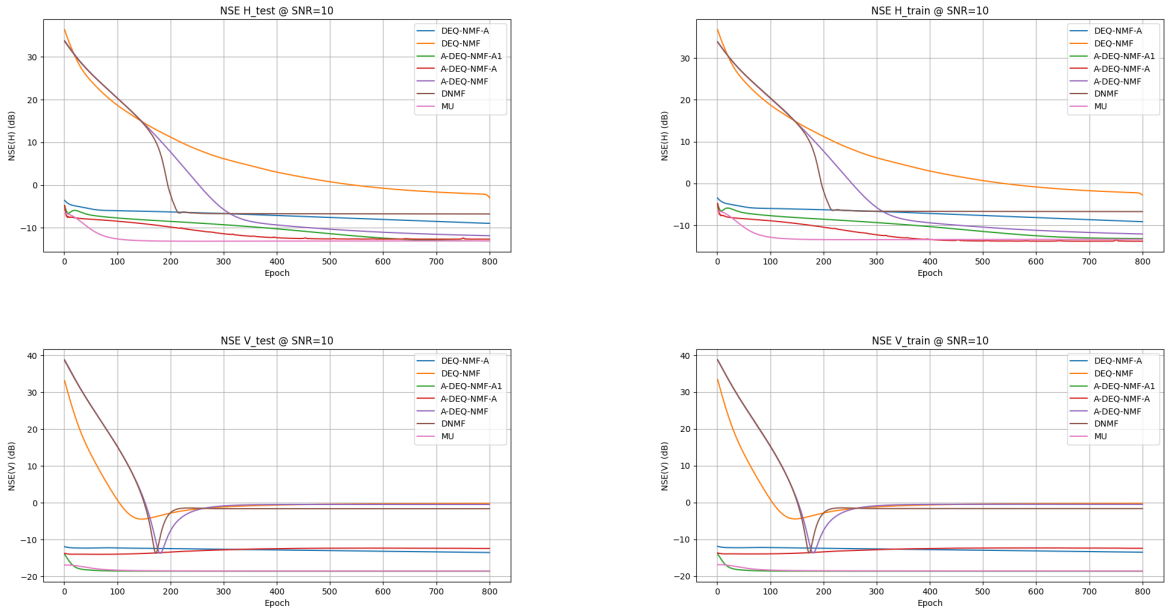


Figure 1.8.2: Τιμές NSE κατά την εκπαίδευση σε συνθετικά δεδομένα (SNR=10 dB).

1.9 Συμπεράσματα και Μελλοντικές Επεκτάσεις

1.9.1 Συμπεράσματα

Στην παρούσα διατριβή αντιμετωπίσαμε την πρόκληση της υλοποίησης των βασικών ιδεών των DEQs σε ένα βαθύ δίκτυο που επιλύει το εποπτευόμενο πρόβλημα NMF. Εισαγάγαμε ένα μοντέλο βασισμένο σε DEQ για την επίλυση του εποπτευόμενου προβλήματος NMF, αξιοποιώντας τεχνικές implicit παραγώγισης για εξοικονόμηση μνήμης. Κατόπιν ενισχύσαμε αυτό το μοντέλο τροποποιώντας τη δομή των παραμέτρων του δικτύου, επιτυγχάνοντας καλύτερη επίδοση ως προς κάποιες μετρικές.

Στη συνέχεια προτείναμε ένα σχήμα προσέγγισης ισορροπίας που υπολογίζει το σημείο ισορροπίας σε ένα μόνο βήμα. Όπως και στο DEQ-NMF, βελτιστοποιήσαμε τις επιδόσεις της μεθόδου προσέγγισης ισορροπίας με αλλαγές στις παραμετρους και στη συνάρτηση απώλειας. Τέλος, πραγματοποιήσαμε εκτενείς πειραματικές συγκρίσεις, όπου τα προτεινόμενα μοντέλα υπερέβησαν τις υπάρχουσες μεθόδους, εμφανίζοντας εξαιρετικά χαμηλά σφάλματα NSE τόσο σε συνθετικά όσο και σε προσομοιωμένα δεδομένα.

1.9.2 Μελλοντικές Επεκτάσεις

Στην τρέχουσα έρευνα μας σκοπεύουμε να εξελίξουμε περαιτέρω τόσο το μοντέλο DEQ-NMF όσο και το σχήμα προσέγγισης ισορροπίας για την επίτευξη ανώτερης απόδοσης. Συγκεκριμένα, οι μελλοντικοί μας άξονες εργασίας είναι οι εξής:

- **DEQ-NMF:** Στην τρέχουσα υλοποίηση χρησιμοποιήσαμε έναν απλό solver αντί για πιο προηγμένες μεθόδους όπως το Anderson acceleration, κυρίως λόγω περιορισμών στις διαθέσιμες βιβλιοθήκες. Στο μέλλον προγραμματίζουμε να διερευνήσουμε και να ενσωματώσουμε προηγμένους solver, ώστε να ενισχύσουμε τη σύγκλιση και την ακρίβεια του DEQ-NMF, καθώς και να πειραματιστούμε συστηματικά με fine-tuning των παραμέτρων του δικτύου και του solver.
- **A-DEQ-NMF:** Τα αποτελέσματά μας υποδεικνύουν ότι οι τυπικές αρχιτεκτονικές δικτύων δυσκολεύονται να μάθουν με ακρίβεια τη μητρώα βάσης όταν αυτή αντιμετωπίζεται ως παραμετρικό δικτύου, ενώ οι προτεινόμενες τροποποιήσεις βελτίωσαν ελαφρώς την απόδοση. Ωστόσο, στα μεταλλακτικά δεδομένα η εκτίμηση παρέμεινε μη ικανοποιητική. Σκοπεύουμε να διεξάγουμε εις βάθος ανάλυση των αιτίων αυτού του περιορισμού και να αναπτύξουμε στοχευμένες τροποποιήσεις για την άρση του.
- **Γενίκευση και Εκτεταμένη Αξιολόγηση:** Για τα προτεινόμενα μοντέλα βαθιάς ισορροπίας και προσέγγισης ισορροπίας, σχεδιάζουμε να αναπτύξουμε εναλλακτικές εκδόσεις των αρχιτεκτονικών μας προσαρμοσμένες για την περίπτωση του μη-επιβλεπόμενου NMF, όπου διαθέτουμε μόνο τη μήτρα δεδομένων. Επιπλέον, δεδομένου ότι οι μέθοδοι μας υπερέβησαν σταθερά τα baseline, θα επικυρώσουμε την αποτελεσματικότητά τους σε ευρύτερο φάσμα συνόλων δεδομένων και εφαρμογών, ώστε να εξετάσουμε εις βάθος τη γενικευσιμότητα και τα πλεονεκτήματά τους.

Chapter 2

Introduction

2.1 Motivation

Non-negative Matrix Factorization (NMF) has gained significant attention due to its ability to decompose complex data into interpretable, additive components under non-negativity constraints. This makes it especially useful in real-world applications where the data naturally consists of non-negative quantities, such as pixel intensities, word frequencies, or chemical concentrations. The factorized matrices often reveal meaningful latent structures, making NMF a powerful tool for dimensionality reduction, source separation, and feature extraction. Its simplicity, interpretability, and wide applicability across domains—ranging from hyperspectral imaging and audio processing to mutational signature analysis and document analysis—make it a compelling subject of study both from theoretical and applied perspectives.

In the recent years, a new technique named Algorithm Unrolling has opened ways to explore how classic iterative machine learning algorithms can be translated to deep neural networks. In most cases, the classic algorithms are blind, meaning their parameters are initialized empirically and are kept constant through the iterations. With algorithm unrolling, if we have knowledge over some dataset, we can train our network and find the ideal parameters through more advanced algorithms like gradient descent or ADAM. In that way we can make more practical use of our knowledge of a problem.

Concurrently, driven by the need to reduce both computational and memory demands of traditional deep architectures, Deep Equilibrium Models (DEQs) recast an arbitrarily deep, weight-tied network as a single-layer fixed-point problem. By solving for an equilibrium state rather than unrolling multiple layers, DEQs achieve the expressive power of an "infinite" network while incurring only constant memory cost during both training and inference.

Recognizing the practical benefits of such an approach, it is natural to aim to integrate DEQ's core principles into unrolled networks to enhance both memory efficiency and computational performance. Accordingly, in this thesis we extend those ideas to the deep network created by unrolling the iterative Multiplicative Update (MU)–NMF algorithm.

2.2 Contributions

To address the challenge of solving the Supervised NMF problem with minimal memory footprint, we first provide a DEQ-NMF model that is based on the unrolled version of the iterative Multiplicative Update (MU) algorithm. In contrast to previous implementations, our method allows increasing the number of iterations in the forward pass while maintaining constant memory throughout backpropagation in the backward step. This becomes possible by utilizing implicit differentiation, as proposed in [19]. In this work, we not only present an analytical derivation of the required gradients for training our model within a gradient descent framework, but also demonstrate the practical use of advanced optimization algorithms such as ADAM during training. We then proceed to propose a modified version of DEQ-NMF, where we alter the structure of our network aiming to increase its effectiveness on existing and other metrics/parameters.

After implementing the aforementioned methods, we introduce a novel one-step equilibrium approximation scheme designed to encapsulate the core fixed-point computation within a single update. This work has been accepted for presentation at the 25th International Conference on Digital Signal Processing (DSP 2025), to be held in Costa Navarino, Messinia, Greece, in June 2025 [32]. We further refine this scheme through targeted architectural enhancements aimed at improving its overall effectiveness. To the best of our knowledge, this equilibrium approximation approach constitutes a new contribution that has not been previously explored in the literature. Empirical evaluations on both synthetic and simulated mutational datasets demonstrate that our proposed methods not only match, but in several key metrics, surpass the accuracy of classical MU algorithms and existing deep NMF techniques on the datasets considered.

2.3 Thesis Outline

- **Chapter 2** introduces the basic concepts in the fields of non-negative matrix factorization and deep learning and provides an overview of new techniques employed in those fields the recent years.

- **Chapter 3** goes over the regularized NMF framework, that was the base of the unrolled network for NMF. This chapter aims to establish a solid background for the following developments and methods later presented in this thesis.
- **Chapter 4** introduces a DEQ-inspired deep network for supervised NMF that achieves constant memory cost, together with a one-step fixed-point approximation approach. These architectures are then refined through targeted structural enhancements. We evaluate the proposed models by conducting extensive experiments on both synthetic and simulated mutational datasets, demonstrating superior performance compared to classical MU algorithms and prior unrolled approaches.
- **Chapter 5** serves as the concluding chapter of this thesis, providing a summary of the main findings and presenting comprehensive conclusions based on the research conducted. It also outlines potential directions for future work, highlighting opportunities for further advancement in the field.

2.4 Notation

We use bold uppercase letters $\mathbf{X}$ to represent matrices and bold lowercase letters $\mathbf{x}$ for column vectors. We denote x_i as the ith element of the column vector $\mathbf{x}$, x_{ij} as the ijth entry and $\mathbf{x}_i$ as the ith column of matrix $\mathbf{X}$, respectively, except in cases where we retain the original notation from referenced works, in which case the meaning of each symbol is explicitly defined. Symbols $\odot$, $\oslash$, $\mathbf{I}_n$, and $\mathbf{1}_{n\times m}$ are used for elementwise multiplication and division, the $n \times n$ identity matrix, and the $n \times m$ all-ones matrix, respectively. Finally $||\mathbf{X}||_{\mathrm{F}}$ denotes the Frobenius norm of $\mathbf{X}$.

Chapter 3

Theoretical Background

3.1 Non-negative Matrix Factorization

In this chapter we lay the mathematical and algorithmic foundations that our later work will build upon. We begin by reviewing each key concept in turn so that the reader can follow the development of our proposed approach in the chapters that follow. By working through these ideas step by step, we aim to clarify how they interrelate and to prepare the ground for the contributions presented in Chapter 4. Specifically, this chapter reviews four key pillars of our approach: Nonnegative Matrix Factorization (NMF), core principles of deep learning, the algorithm-unrolling paradigm, and deep equilibrium models (DEQs). Together, these concepts form the theoretical toolkit that underpins the novel methods developed later in our thesis.

3.1.1 Introduction to NMF

Nonnegative Matrix Factorization (NMF) seeks an additive, parts-based decomposition of a nonnegative data matrix $\mathbf{V} \in \mathbb{R}_{\geq 0}^{n \times m}$ into two low-rank factors, i.e.,

$$\mathbf{V} \;\approx\; \mathbf{W}\,\mathbf{H}, \quad \mathbf{W} \in \mathbb{R}_{\geq 0}^{n \times r},\ \mathbf{H} \in \mathbb{R}_{\geq 0}^{r \times m},\ r \ll \min(n, m).$$

By enforcing the nonnegativity constraints $\mathbf{W}, \mathbf{H} \geq 0$, NMF produces a decomposition where the columns of $\mathbf{W}$ represent nonnegative "building blocks" and the corresponding columns of $\mathbf{H}$ encode nonnegative combination coefficients. Unlike other linear dimensionality reduction (LDR) techniques such as Principal Component Analysis (PCA) [1] or the Singular Value Decomposition (SVD) [2], which allow both positive and negative entries due to their orthogonality properties, NMF enforces a strictly additive model. This lack of subtraction often results in more interpretable and sparse representations.

NMF Problem Statement

Formally, given a nonnegative data matrix $\mathbf{V} \in \mathbb{R}_{\geq 0}^{n \times m}$, a desired factorization rank r, and a distance measure $D(\cdot, \cdot)$ between matrices, NMF seeks two nonnegative factors $\mathbf{W} \in \mathbb{R}_{\geq 0}^{n \times r}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{r \times m}$ such that $D(\mathbf{V}, \mathbf{WH})$ is minimized:

$$\min_{\mathbf{W} \in \mathbb{R}_{\geq 0}^{n \times r},\, \mathbf{H} \in \mathbb{R}_{\geq 0}^{r \times m}} \quad D(\mathbf{V}, \mathbf{W}\,\mathbf{H}). \tag{3.1.1}$$

Choice of Factorization Rank r

The factorization rank r plays a central role in Non-negative Matrix Factorization (NMF), as it controls the model's capacity to capture latent structure. Choosing r too small results in an undercomplete representation, where the limited basis forces the model to compress diverse patterns, often causing large reconstruction errors and underfitting. In contrast, selecting r too large leads to an overcomplete representation that may reconstruct the data nearly exactly, but at the cost of uniqueness and robustness, increasing the risk of overfitting and reduced interpretability. There is no universally optimal rank r that performs best across all datasets or tasks. In practice, r must be selected through model selection procedures such as cross-validation, performance metrics tailored to the application, or domain-specific heuristics that balance reconstruction quality with model simplicity.

Non-Uniqueness and the Need for Regularization

The optimization problem posed in (3.1.1) is *jointly non-convex* in the variables $(\mathbf{W}, \mathbf{H})$, and admits infinitely many solutions. In particular, for any invertible matrix $\mathbf{Q} \geq 0$ with $\mathbf{Q}^{-1} \geq 0$, the transformed pair

$$\mathbf{W}' = \mathbf{WQ}, \qquad \mathbf{H}' = \mathbf{Q}^{-1}\mathbf{H}$$

satisfies

$$\mathbf{W}'\mathbf{H}' = \mathbf{WQ}\,\mathbf{Q}^{-1}\mathbf{H} = \mathbf{WH}.$$

This inherent ambiguity limits the interpretability of NMF solutions, especially in overcomplete settings.

To address this, one often augments the NMF objective with additional regularization terms or constraints that induce desirable structure in the factors [3]. Examples include:

- **Sparsity** constraints on $\mathbf{H}$, to promote part-based representations.
- **Orthogonality** constraints on $\mathbf{W}$, to encourage diversity in the basis.
- **Smoothness or graph-based** regularization, to encode relationships between features or samples.
- **Volume or simplex constraints**, to enforce geometric or probabilistic interpretability.

Such regularizers help break the symmetry of the solution space and guide the optimization toward meaningful and unique factorizations [4].

3.1.2 Multiplicative Update Algorithm for NMF

Numerous algorithms have been proposed to solve the NMF optimization problem, including alternating least squares (ALS) [5], projected gradient methods [6], coordinate descent [7], and hierarchical alternating optimization [8]. Among these, the Multiplicative Update (MU) algorithm introduced by Lee and Seung [9] is particularly popular due to its simplicity, ease of implementation, and ability to preserve non-negativity inherently. In this work, we focus on MU as it forms the basis for subsequent developments presented in later sections. Given the objective function defined by the Frobenius norm:

$$\min_{\mathbf{W},\mathbf{H}\geq 0} \quad \frac{1}{2}\|\mathbf{V}-\mathbf{W}\mathbf{H}\|_F^2,$$

the Multiplicative Update rules iteratively update matrices $\mathbf{W}$ and $\mathbf{H}$ according to the following equations:

1. Initialize $\mathbf{W}$ and $\mathbf{H}$ with non-negative random values.
2. Iterate until convergence:

$$h_{ij} \leftarrow h_{ij}\frac{(\mathbf{W}^\top\mathbf{V})_{ij}}{(\mathbf{W}^\top\mathbf{W}\mathbf{H})_{ij}+\epsilon}, \quad \text{for all } i,j$$

$$w_{ij} \leftarrow w_{ij}\frac{(\mathbf{V}\mathbf{H}^\top)_{ij}}{(\mathbf{W}\mathbf{H}\mathbf{H}^\top)_{ij}+\epsilon}, \quad \text{for all } i,j$$

where ϵ is a small constant added for numerical stability to avoid division by zero. Each update preserves the non-negativity constraints due to the multiplicative nature of the update rules.

3.1.3 NMF Applications

Non-negative Matrix Factorization has diverse applications across many fields, including:

- **Image Processing – Facial Feature Extraction.** A classic and foundational application of NMF is facial feature extraction, as first demonstrated by Lee and Seung [10]. In this application, we have a data matrix $\mathbf{X} \in \mathbb{R}_{\geq 0}^{n\times m}$ representing a vectorized gray-level image of a human face. The value of the element x_{ij} is the intensity of the ith pixel in the corresponding jth face. With NMF, $\mathbf{V}$ is factorized to a basis matrix $\mathbf{W} \in \mathbb{R}_{\geq 0}^{n\times r}$ that consists of basis elements and a matrix $\mathbf{H} \in \mathbb{R}_{\geq 0}^{r\times m}$ that contains the combination of basis elements that construct each face. A fact that reinforces the interpretability of NMF, is that the basis elements extracted are parts of the human face such as eyes, lips, mustaches and noses. Moreover, because of the non-negativity constraints, the basis images can only be summed up to obtain each original face image. This can be seen in 3.1.1:
- **Text Mining – Topic Recovery and Document Classification.** The use of NMF in document clustering and topic modeling was introduced by Xu et al. [12]. NMF was formulated as follows: The data matrix $\mathbf{X} \in \mathbb{R}_{\geq 0}^{n\times m}$ has columns which correspond to a document and rows that correspond to a certain word. In this way the $\mathbf{X}(i,j)$ can contain the number of times the jth word appears in the ith document. Because of the non-negativity, each of the columns of $\mathbf{W}$ can be interpreted as a set of words. Since the number of documents m is usually quite bigger than the number of basis elements r, the columns of $\mathbf{W}$ should be sets of words found in several documents. Therefore we can interpret the basis elements as topics, to which we assign each document using the matrix $\mathbf{H}$. All in all, NMF

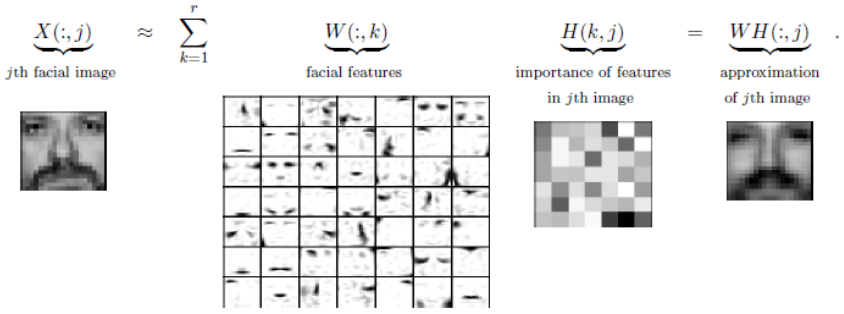


Figure 3.1.1: NMF Facial Feature Extraction. From [11].

manages to identify topics and classify the documents among those topics at the same time. This is illustrated in 3.1.2.

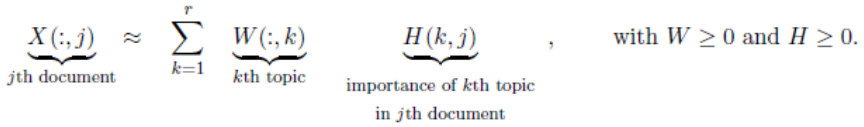


Figure 3.1.2: NMF for Topic Modeling. From [11]

- **Hyperspectral Unmixing – Identify Endmembers and Classify Pixels.** NMF was first proposed for hyperspectral unmixing by Nascimento and Dias [13]. In hyperspectral imaging, each pixel holds information of the reflected light among many wavelengths, consisting the unique spectral signature of the pixel. Hyperspectral unmixing aims to identify every material's unique spectral signature, called endmember, and then proceeds to estimate what combination of spectral signatures each pixel holds. Typically, this involves the use of the linear mixing model, which assumes that each pixel's spectral signature contains a linear combination of endmembers. NMF is formulated as following: The data matrix **X** has columns which contain the spectral signatures of pixels. We then factorize it to a basis matrix **W**, which has columns that represent the endmembers and an abundance matrix **H** which indicates how much each endmember is present in each pixel. In Fig. 3.1.3, we see how a hyperspectral cube is decomposed in a set of endmembers and their corresponding abundance maps.

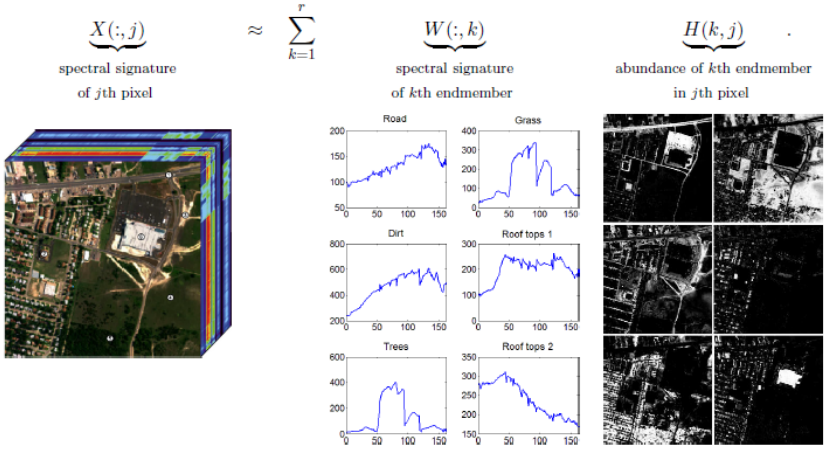


Figure 3.1.3: NMF For Spectral Unmixing. From [11].

- **Medical Applications – Cancer Genomics.** NMF has proven especially powerful in cancer genomics for uncovering mutational processes hidden within tumor genomes. In a landmark study, Alexandrov et al. [14] applied NMF to tens of thousands of cancer genome and exome sequences, decomposing complex mutational catalogs into distinct mutational signatures and estimating their contributions across cancer types. In this setting, the matrix $\mathbf{V}$ represents a mutation count matrix per mutation type per sample, $\mathbf{W}$ captures the mutational signatures, and $\mathbf{H}$ their exposures. This decomposition has provided fundamental insights into DNA damage and repair mechanisms in cancer.

3.2 Introduction to Neural Networks and Deep Learning

This section serves to clarify some key concepts of deep learning, offering a brief historical overview of the field and introducing the main architectures and algorithms behind neural networks. Deep learning is a concept widely discussed in this thesis and used to enhance traditional NMF algorithms.

3.2.1 Historical Review

Neural networks are a class of machine learning algorithms inspired by the structure and functioning of biological neural networks found in the human brain. These models comprise interconnected computational units, often referred to as "neurons", which collaboratively process data, identify patterns, and generate predictions. A key distinguishing feature of neural networks is their ability to learn from examples and generalize this knowledge to previously unseen data. This learning process is enabled through backpropagation, an algorithm that minimizes prediction errors by iteratively updating the model's internal parameters. Over time, this iterative refinement enhances the network's predictive accuracy, making neural networks powerful tools for a wide range of applications.

The origins of neural networks can be traced back to the 1940s, with foundational work by Walter Pitts and Warren McCulloch, who first introduced the concept of a "neural network" [33]. A significant milestone followed in 1957 with Frank Rosenblatt's development of the perceptron [34], an early model that laid the groundwork for future advancements. However, the practical realization of neural networks' full potential was initially hindered by limited computational resources and the scarcity of large datasets.
A resurgence of interest occurred in the 1980s and intensified in the 2000s, driven by breakthroughs in computational hardware and the growing availability of extensive datasets. These developments enabled the training of more sophisticated and deeper network architectures. The emergence of deep neural networks—networks composed of multiple layers—has since revolutionized the field of artificial intelligence. Notable innovations such as convolutional neural networks (CNNs) [35], [36] for image processing and recurrent neural networks (RNNs) [37], [38] for sequential data analysis have profoundly impacted areas including computer vision, natural language processing (NLP), and speech recognition.

The continued evolution of deep learning has been supported by advances in hardware, particularly the use of Graphics Processing Units (GPUs) for parallel computation and the development of specialized hardware such as Tensor Processing Units (TPUs). These technologies have significantly accelerated both the training and deployment of large-scale neural models, enabling breakthroughs across complex machine learning tasks. Most recently, the advent of Large Language Models (LLMs) represents a transformative development in NLP and AI. LLMs demonstrate an exceptional capacity to understand and generate human-like language. Leveraging vast computational resources and advanced training algorithms, these models excel at tasks such as translation, summarization, question answering, and text generation. Beyond traditional NLP applications, LLMs are also influencing domains such as virtual assistants, automated content creation, and sentiment analysis, marking a significant milestone in the advancement of artificial intelligence.

In recent years, neural networks and deep learning have been increasingly incorporated into model-based approaches for NMF, enhancing their expressiveness and adaptability. This thesis builds upon one such idea, drawing inspiration from these developments and further exploring its implications and extensions within a broader deep learning framework.

3.2.2 Basic Structure of Neural Networks

Within the field of artificial intelligence (AI), the neuron serves as the fundamental unit of artificial neural networks, modeled after the biological neurons found in the human brain. In artificial systems, a neuron is a basic computational element responsible for processing and transmitting information throughout the network. Similar to its biological analogue, an artificial neuron takes input signals, applies a mathematical transformation, and generates an output. These neurons are interconnected via weighted connections, often referred to as "synapses," allowing them to collectively analyze and interpret data, as illustrated in Fig. 3.2.1. Their behavior is governed by a combination of weighted summation and activation functions, enabling the network to model complex patterns and dependencies within the data. The primary components of a neural network include:

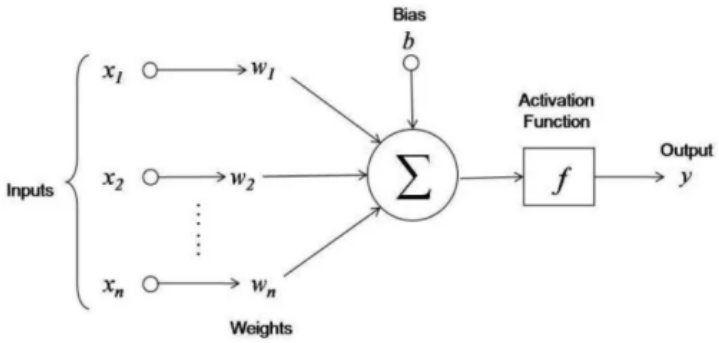


Figure 3.2.1: Structure of a classic neural network. From [39].

- **Input Layer:** Provides the initial feature set or input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ to the neuron, where each x_i represents a distinct input feature.
- **Weights:** The parameters $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ represent the strength of the connections between input features and the neuron, influencing the neuron's response.
- **Summation Function:** Calculates the weighted sum z of inputs and weights, with an added bias term b: $z = \sum_{i=1}^{n}(x_i \cdot w_i) + b$.
- **Activation Function:** Applies a non-linear transformation to z, enabling the neuron to model complex patterns.
- **Output:** The neuron's output y, produced after activation, serves as its response for tasks such as classification or prediction.

Activation functions are essential components of artificial neurons, introducing non-linearity that allows neural networks to model complex patterns in data. Without them, the network would be limited to linear transformations, reducing its expressive power. By enabling varied responses to inputs, activation functions enhance a network's ability to learn intricate relationships, which is crucial in domains like image recognition and natural language processing. They also support function approximation, helping networks generalize and extract meaningful features.

3.2.3 Training a Neural Network

In this subsection, we present some key concepts of training a neural network. To make accurate predictions, neural networks require a training procedure that adjusts their internal parameters. Depending on the architecture and application, the number of these parameters can range from thousands to billions. The objective of training is to align the network's predictions with known targets by minimizing a predefined loss function using gradient-based optimization techniques.

Backpropagation is the central algorithm used to update the network's weights [40]. It consists of two main phases:

- **Forward Pass:** In this phase, the input data is passed through the network layer by layer. Each layer applies transformations and activation functions, and the final layer produces the output. The forward pass is responsible for generating predictions based on the current state of the network.
- **Backward Pass:** Also known as backpropagation, this phase follows the forward pass. It calculates the error between the predicted output and the actual target values, then propagates this error backward through the network. Gradients of the loss with respect to each parameter are computed and used to update the weights, typically via gradient descent. This iterative process enables the network to refine its parameters and improve prediction accuracy over time.

Once the per-parameter gradients have been computed by backpropagation, they must be used to acquire our new weight parameters. In this work we consider two such first-order update rules: **Gradient Descent** and the **Adam** optimizer (Algorithm 1).

Gradient Descent is an iterative optimization algorithm commonly used to minimize a differentiable loss function [41]. At each step, the algorithm updates the parameters in the direction of the negative gradient of the loss, which corresponds to the steepest descent. Formally, the update rule from step k to $k+1$ is given by:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla L(\mathbf{w}_k)$$

Here, $\mathbf{w}_k$ represents the parameters at step k, $L(\mathbf{w})$ is the loss function, $\nabla L(\mathbf{w}_k)$ is its gradient, and η is the learning rate (or step size), which controls how large each update is. A smaller learning rate may slow convergence or get stuck in local minima, while a larger one may overshoot and destabilize training. Choosing an appropriate learning rate is crucial for effective and stable training [42].

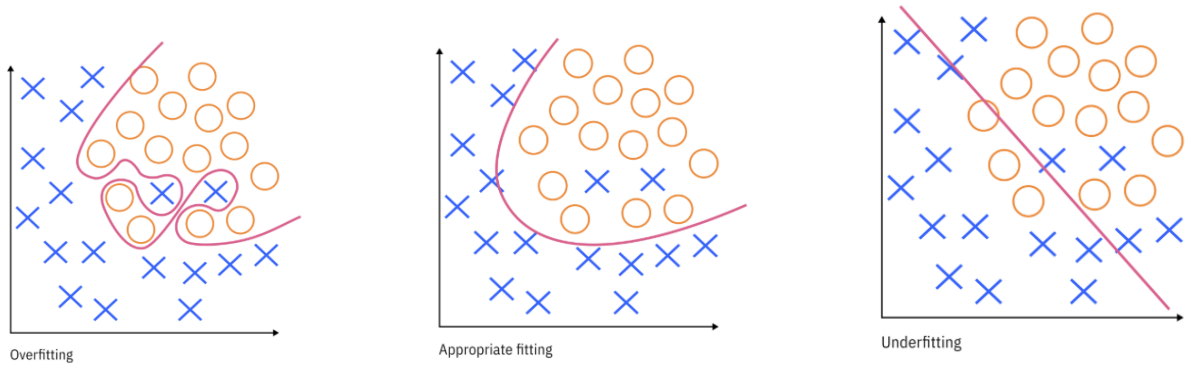
While standard gradient descent provides a foundational method for optimizing differentiable objectives, it can struggle with issues such as slow convergence, noisy gradients, or poorly scaled parameters. To address these limitations, a variety of adaptive optimization algorithms have been proposed. One widely used and effective method is Adam (Adaptive Moment Estimation), which combines ideas from both momentum based methods [43], [44] and RMSProp [45]. Adam maintains exponentially decaying averages of past gradients (first moment) and squared gradients (second moment) to adaptively adjust the learning rate for each parameter. This makes it particularly well-suited for problems with sparse gradients or non-stationary objectives. The algorithm has become a default choice for training deep learning models due to its robustness, efficiency, and minimal need for manual tuning. Below, we present the pseudocode for the Adam optimizer as introduced in [41]. We provide this formal description to clarify its internal mechanism, as the algorithm will be used in our own experiments and model training processes later in this work.

Algorithm 1. Adam [41]

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1]$: Decay rates
Require: $f(\theta)$: Stochastic objective
Require: θ_0: Initial parameters
$m_0 \leftarrow 0$ (Initialize 1^{st} moment vector) $v_0 \leftarrow 0$ (Initialize 2^{nd} moment vector) $t \leftarrow 0$ (Initialize time step)
while *not converged* **do**
  $t \leftarrow t + 1$ $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t) $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ (Update biased first moment estimate) $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ (Update biased second raw moment estimate) $\hat{m}_t \leftarrow \dfrac{m_t}{1 - \beta_1^t}$ (Compute bias-corrected first moment estimate) $\hat{v}_t \leftarrow \dfrac{v_t}{1 - \beta_2^t}$ (Compute bias-corrected second raw moment estimate) $\theta_t \leftarrow \theta_{t-1} - \alpha \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$ (Update parameters)
return θ_t (Resulting parameters)

Overfitting and Underfitting: Two important concepts in neural network training are overfitting/underfitting, which describe how well a model generalizes to new data and regularization. Understanding and addressing these phenomena is essential for developing models that perform reliably on both training and unseen data. Overfitting occurs when a model captures noise or irrelevant patterns in the training data,

performing well on the training set but failing to generalize to new, unseen data. This typically results from excessive model complexity or overly long training. In contrast, underfitting happens when a model is too simple or insufficiently trained, failing to capture the underlying structure of the data. It performs poorly on both training and test sets. Achieving a good balance between overfitting and underfitting involves selecting appropriate model complexity and employing techniques like regularization, cross-validation, and careful tuning of hyperparameters. An illustration of these concepts in a binary classification task is shown in Fig. 3.2.2.



(a) Overfitting (b) Appropriate fitting (c) Underfitting

Figure 3.2.2: Model fitting scenarios in binary classification. From [46].

Regularization: Regularization introduces a penalty term to the loss function to discourage overly complex models and reduce overfitting. Two common types are ℓ_1 (Lasso) and ℓ_2 (Ridge) regularization. Lasso promotes sparsity by driving some weights to zero, while ridge penalizes large weights, keeping them small. In neural networks, ridge regularization is often used to stabilize weight updates and improve generalization. Mathematically, these penalties are expressed as:

$$L_1(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} |w_i|, \tag{3.2.1}$$

$$L_2(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} w_i^2. \tag{3.2.2}$$

These regularization techniques are employed later in this thesis to improve model generalization and prevent overfitting during training.

3.3 Algorithm Unrolling

Algorithm unrolling, also known as unfolding, is an emerging technique that maps each iteration of a classical signal-processing algorithm onto a layer of a deep neural network with trainable parameters. By doing so, it forges a direct, systematic link between well-understood optimization routines and modern learning architectures, effectively imbuing networks with the interpretability, convergence behavior, and efficiency of their algorithmic counterparts. Originally introduced to accelerate sparse-coding solvers, unrolled models have since gained widespread attention, finding applications across imaging, vision, and speech processing. Their popularity stems from the ability to leverage domain knowledge to build high-performance models that require far less data and offer clearer theoretical insights than conventional "black-box" deep networks.

3.3.1 Deep Neural Networks and Interpretability

Deep neural networks (DNNs) have revolutionized many fields, especially signal and image processing, by delivering unprecedented performance improvements. However, their widespread adoption in practical scenarios

is significantly impeded by their opaque, "black-box" nature. Due to their complex, multilayered structures involving millions of learned parameters, it is notoriously difficult to interpret how and why specific decisions are made within the network. Consequently, this lack of interpretability complicates validation, debugging, and trust-building, making it challenging to gain insights into what information or domain-specific knowledge has been implicitly captured by the model.

3.3.2 Traditional Iterative Algorithms

On the other hand, traditional iterative algorithms offer high interpretability by explicitly modeling physical processes and embedding prior domain knowledge directly into their structure. However, these methods suffer from the critical drawback of relying on empirically chosen parameters, which are typically fixed without systematic optimization based on available real-world datasets. This manual selection limits the algorithm's adaptability, potentially causing suboptimal performance, especially when comprehensive datasets exist that could otherwise inform and refine these parameters. Thus, iterative algorithms often cannot fully leverage the extensive knowledge present in modern datasets, restricting their capacity to achieve optimal, data-informed solutions.

3.3.3 Algorithm Unrolling Emerges

Algorithm unrolling (or unfolding) is a technique that transforms iterative algorithms into deep neural networks by mapping each iteration of the algorithm into one layer of the network. Essentially, each step of an iterative procedure, becomes an interpretable layer whose parameters, previously determined manually or through fixed rules, become learnable from data. By stacking these layers, the resulting deep architecture retains the interpretability of traditional algorithms while gaining the adaptivity and performance benefits of deep learning. The structure of such an unrolled network can be seen in Fig. 3.3.1.

The earliest work in algorithm unrolling was presented by Gregor et al. [15], aiming to improve the computational efficiency of sparse coding algorithms through end-to-end training. In their work, they unfolded the Iterative Shrinkage and Thresholding Algorithm (ISTA), an algorithm used to solve the sparse coding problem [16] and proposed its unrolled counterpart, later called learned ISTA (LISTA). In what follows, we analyze this unfolding example in detail, examining how each ISTA iteration maps to a corresponding network layer.

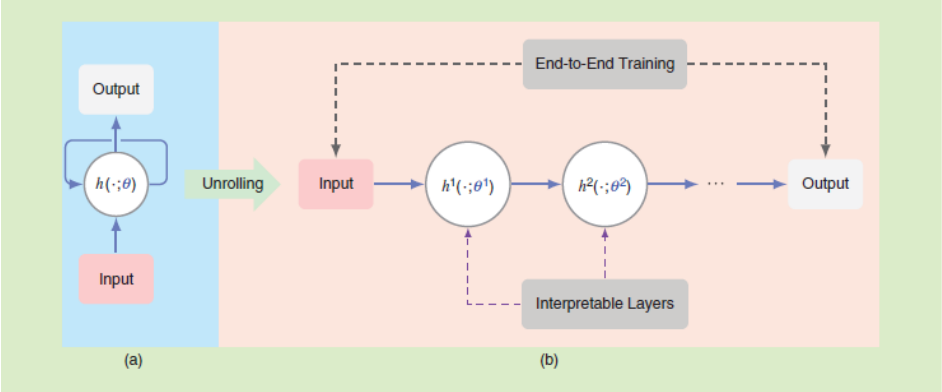


Figure 3.3.1: Unrolled Network illustration from [17].

Concretely, given an input vector $\mathbf{y} \in \mathbb{R}^n$ and an overcomplete dictionary $\mathbf{W} \in \mathbb{R}^{n \times m}$ with $m > n$, sparse coding refers to finding a sparse representation of $\mathbf{y}$ using $\mathbf{W}$. In other words, we aim to find a sparse code $\mathbf{x} \in \mathbb{R}^m$ such that $\mathbf{y} \approx \mathbf{W}\mathbf{x}$ while "pushing" as many coefficients as possible in $\mathbf{x}$ to be zero. A common approach to determine $\mathbf{x}$ is to solve the following convex optimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^m} \frac{1}{2}\|\mathbf{y} - \mathbf{W}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1, \tag{3.3.1}$$

where $\lambda > 0$ is a regularization parameter that controls the sparsity of the solution. Specifically, ISTA refines its estimate at each iteration as follows:

$$\mathbf{x}^{t+1} = S_{\lambda/\mu}\Big(\big(\mathbf{I} - \tfrac{1}{\mu}\mathbf{W}^\top\mathbf{W}\big)\,\mathbf{x}^t + \tfrac{1}{\mu}\mathbf{W}^\top\mathbf{y}\Big), \quad t = 0, 1, \ldots \tag{3.3.2}$$

where $\mathbf{I} \in \mathbb{R}^{m\times m}$ is the identity matrix, $\mu > 0$ is a step-size parameter, and $S_\theta(\cdot)$ is the element-wise soft-thresholding operator defined as $S_\theta(\mathbf{x}) = \operatorname{sign}(\mathbf{x}) \cdot \max\big\{|\mathbf{x}| - \theta,\, 0\big\}$.

We can interpret L of ISTA as cascading L layers together, which essentially form a L-layer deep network as illustrated in Fig. 3.3.2. In the unrolled network, we now consider the learnable parameters:

$$\mathbf{W}_t = \mathbf{I} - \tfrac{1}{\mu}\mathbf{W}^\top\mathbf{W}, \quad \mathbf{W}_e = \tfrac{1}{\mu}\mathbf{W}^\top.$$

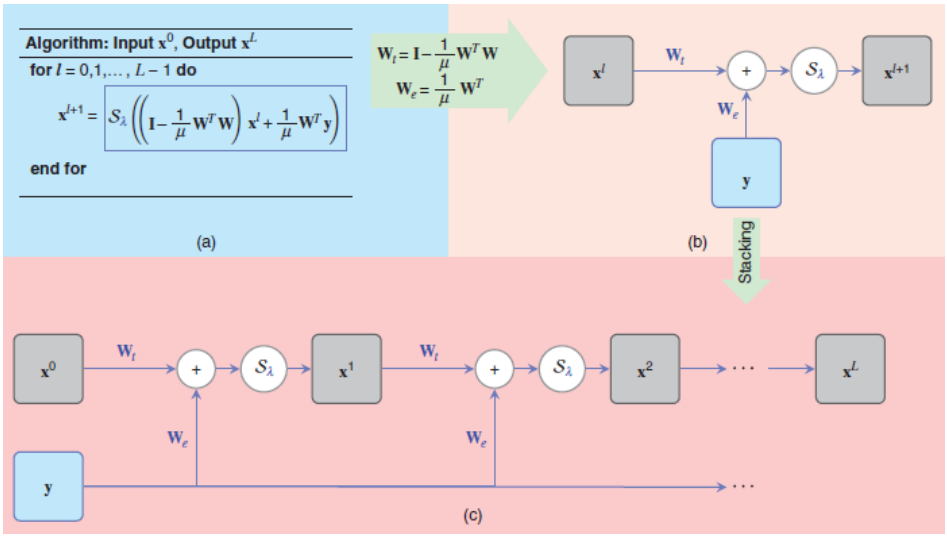


Figure 3.3.2: Unfolding of ISTA to LISTA from [17].

The unrolled network is trained using real datasets to optimize the parameters $\mathbf{W}_t$, $\mathbf{W}_e$, and λ. The training of LISTA is performed through a sequence of vectors $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^N \in \mathbb{R}^n$ and their corresponding ground-truth sparse codes $\mathbf{x}^{*1}, \mathbf{x}^{*2}, \ldots, \mathbf{x}^{*N}$. The network's parameters are then optimized using popular gradient-based learning techniques, such as stochastic gradient descent [18]. After conducting experiments, it was observed that the number of layers in LISTA is quite smaller than the number of iterations that ISTA would need to achieve the same results.

3.4 Deep Equilibrium Models

In this section, we'll assemble the key ideas that consist the base of Deep Equilibrium Models (DEQs). We begin by casting a deep network as a fixed-point iteration, then show how unrolling such an iteration into explicit "layers" and tying the same weights across them produces a powerful, recurrent-style architecture. Building on that, we introduce implicit layers, whose outputs are defined not by a finite chain of transformations but by the solution of an equilibrium equation, and demonstrate how root-finding and implicit differentiation come together to form the Deep Equilibrium Model. These concepts—fixed-point views, algorithm unrolling, weight tying, and implicit layers—will underpin our entire treatment of DEQs.

3.4.1 Implicit Layers

As we discussed previously, a deep neural network is comprised of a large number of connected layers. In modern deep learning, the vast majority of these layers are explicit—that is, defined by a fixed sequence of operations that transform the input into the output. In contrast, an implicit layer defines the conditions

that the layer's output must satisfy, rather than specifying the exact computation to obtain it. A prominent early example of this idea is OptNet [47], which proposed defining a layer's output as the solution to a quadratic program (QP), effectively making it a constrained optimization-based implicit layer. The principal advantage of implicit layers lies in the clear separation between the definition of the layer (e.g., the QP in OptNet) and the algorithm used to solve it. By decoupling these two components, implicit layers gain significant modularity, allowing practitioners to leverage specialized, highly efficient solvers tailored to their application. The logic behind an implicit layer and the distinction with a traditional explicit layer is shown in Fig. 3.4.1.

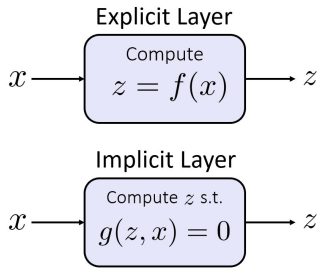
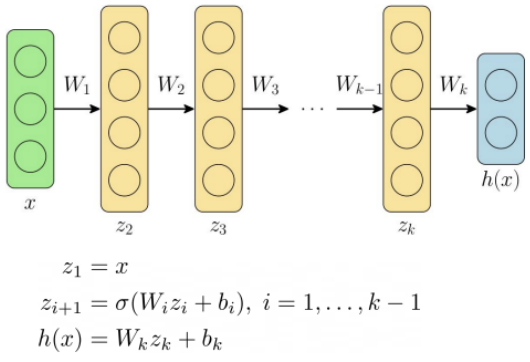


Figure 3.4.1: Illustration of an Implicit Layer [19]

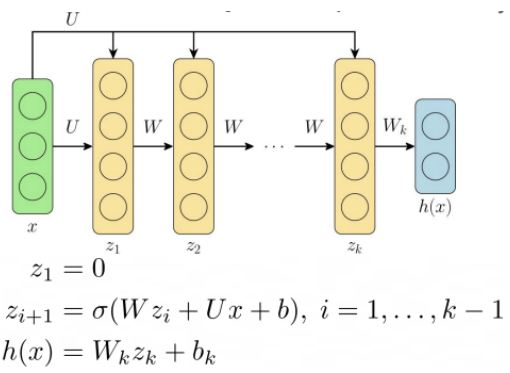
3.4.2 Weight-Tied Neural Networks

Weight-tied networks are a class of neural networks in which the same set of weights is reused across multiple layers. Rather than learning a separate set of parameters for each layer, a weight-tied network enforces parameter sharing, meaning each transformation in the sequence applies the same weight matrix. Formally, given an input $\mathbf{x}$, the network computes hidden states using the recurrence $\mathbf{z}_{i+1} = \sigma(\mathbf{W}\mathbf{z}_i + \mathbf{U}\mathbf{x} + \mathbf{b})$, where $\mathbf{W}$ and $\mathbf{U}$ are shared across all layers. The idea of tying input and output embeddings, which reflects the weight-tied principle, was first formalized by Press and Wolf [20], and has since become foundational in sequence modeling. Weight tying is particularly useful in models designed to process sequential data, such as Recurrent Neural Networks (RNNs), and serves as a conceptual stepping stone to Deep Equilibrium Models (DEQs). It is worth noting that any deep network can be restructured as a weight-tied network with the same depth and with a unit size equal to the sum of the individual hidden unit sizes in the original network. The proof of this can be found in the appendix of [19].

This structural difference is visually illustrated in Figure 3.4.2b, where the weight-tied network reuses the same transformation matrix across layers, in contrast to the classic deep network, shown in Fig. 3.4.2a, that learns a distinct matrix per layer.



(a) Classic Deep Network



(b) Weight-tied Network

Figure 3.4.2: Comparison of classic deep network and weight-tied network architectures.

3.4.3 Fixed Points

The concept of a fixed point is a fundamental mathematical tool that we will need shortly when discussing a broad class of models defined through iterative or equilibrium-based formulations. Formally, let $f : \mathcal{X} \to \mathcal{X}$ be a function defined on a set $\mathcal{X}$. A point $x^\star \in \mathcal{X}$ is called a **fixed point** of f if

$$f(x^\star) = x^\star.$$

That is, if we pick a point $x \in \mathcal{X}$ and we apply f an infinite amount of times, we will eventually get to a point x^* called fixed point or equilibrium point, that will no longer change when f is applied to it. Such points often represent equilibrium or convergence states in dynamical systems and iterative processes. In the subsequent section, we will explore models that leverage fixed-point formulations to define their behavior.

3.4.4 Introduction to Deep Equilibrium Models

Deep Equilibrium Models (DEQs) are models fundamentally built upon the mathematical concept of fixed-point iterations. In standard weight-tied models the update of each layer can be written as:

$$\mathbf{z}_{1:T}^{[i+1]} = f_\theta^{[i]}\left(\mathbf{z}_{1:T}^{[i]}; \mathbf{x}_{1:T}\right) \quad \text{for } i = 0, 1, 2, \ldots, L-1, \tag{3.4.1}$$

where i is the layer index; $\mathbf{z}_{1:T}^{[i]}$ is the hidden sequence of length T at layer i; L is the number of layers; $\mathbf{x}_{1:T}$ is the input sequence and $f_\theta^{[i]}$ is some nonlinear transformation which typically enforces causality. In practice it has been observed that if f_θ meets certain conditions and we apply it repeatedly, the network reaches an equilibrium point $\mathbf{z}_{1:T}^\star$. This convergence can be expressed formally as:

$$\lim_{i\to\infty} \mathbf{z}_{1:T}^{[i]} = \lim_{i\to\infty} f_\theta\left(\mathbf{z}_{1:T}^{[i]}; \mathbf{x}_{1:T}\right) \equiv f_\theta\left(\mathbf{z}_{1:T}^\star; \mathbf{x}_{1:T}\right) = \mathbf{z}_{1:T}^\star \tag{3.4.2}$$

It has been shown that as the depth of a weight-tied deep network increases, we achieve improved perfomance. However, in traditional deep learning, in order to backpropagate through the whole network, we have to compute the gradients for each layer's parameters. Thus our models cannot be trained beyond a certain depth that depends on the available memory. DEQs, however, use a single implicit layer, which is defined implicitly by the fixed-point equation $\mathbf{z}^* = f_{\boldsymbol{\theta}}(\mathbf{z}^*, \mathbf{x})$. In other words, instead of stacking L explicit layers, the layer is defined to be the solution $\mathbf{z}^*$ of this root-finding problem. We can then leverage implicit differentiation and do a single backwards step, rather than backpropagating through all the solver iterations. We present these ideas in detail below.

3.4.5 Forward Pass

Training DEQs involves distinct forward and backward computational processes compared to standard deep networks. In the forward pass, DEQs find their equilibrium state by solving a fixed-point equation rather than sequentially computing layer-by-layer activations. We define:

$$g_\theta(\mathbf{z}_{1:T}^\star; \mathbf{x}_{1:T}) = f_\theta(\mathbf{z}_{1:T}^\star; \mathbf{x}_{1:T}) - \mathbf{z}_{1:T}^\star \tag{3.4.3}$$

Then numerical root-finding techniques are used to find the solution of Eq. 3.4.3. These methods iteratively refine an initial guess until a stable equilibrium point is reached. In practice, the most common root-finding methods for fixed-point iterations include Anderson acceleration [21], [22], Broyden's method [23], and Newton's method [24], [25]. Fig. 3.4.3 visualizes the forward pass of a DEQ model as an iterative process converging to the equilibrium point z^* illustrating the contrast with conventional weight-tied layer-wise computation.

3.4.6 Backward Pass

An issue that arises from the use of a black-box root-finding algorithm, is that we can no longer backpropagate through the operations of the forward-pass, like we do in traditional deep learning. Instead, the backward pass in DEQs is based on the Implicit Function Theorem [26]. More generally, the DEQ framework extends this

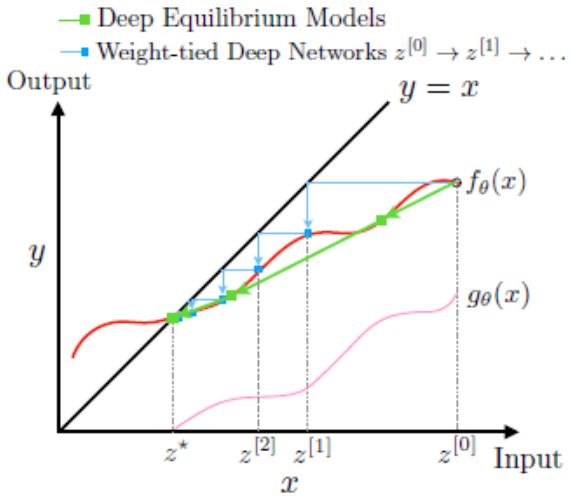


Figure 3.4.3: DEQ vs weight-tied DN solving for the equilibrium point in 2 dimensions. From [19].

approach by applying the implicit function theorem directly to the fixed-point equation, enabling end-to-end differentiation without unrolling. It is shown that the backward gradient through the "infinite" depth network can be computed with a single matrix multiplication that involves the Jacobian evaluated at equilibrium point $\mathbf{z}^*_{\mathbf{1:T}}$. In other words, one can directly compute the gradient of a network parameter $(\cdot)$ w.r.t a loss function ℓ as shown in Eq. 3.4.4, where h denotes any differentiable scalar function of the equilibrium point $\mathbf{z}^\star_{1:T}$ used to apply the chain rule:

$$\frac{\partial \ell}{\partial (\cdot)} = -\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial (\cdot)} = -\frac{\partial \ell}{\partial h} \frac{\partial h}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial (\cdot)}. \tag{3.4.4}$$

Any standard update rule (e.g., gradient descent, Adam) can be used to update the network parameters θ. For example, an SGD update step to update the model's parameters would be:

$$\theta^+ = \theta - \alpha \cdot \frac{\partial \ell}{\partial \theta} = \theta + \alpha \frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \Big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial \theta}. \tag{3.4.5}$$

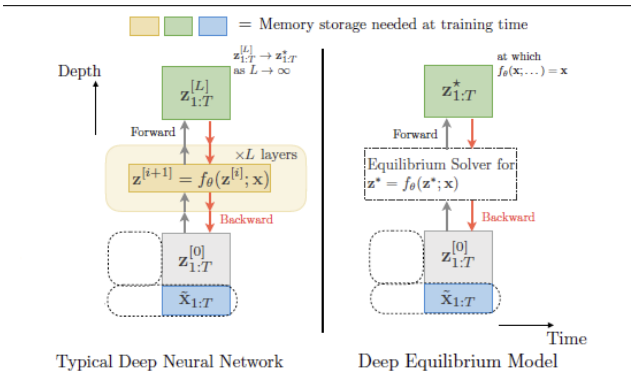


Figure 3.4.4: Training memory comparison: a standard DN with T layers stores all intermediate layer activations, whereas a DEQ model only retains the equilibrium state, yielding constant memory usage. From [19].

Normally, in a classic deep network, we would need to store the intermediate values $\mathbf{z}^1_{1:T}, ..., \mathbf{z}^L_{1:T}$ that are computed between each layer and backpropagate through our network to update its parameters. Taking advantage of the Implicit Function Theorem, DEQs only store the equilibrium point $\mathbf{z}^*_{1:T}$, thus requiring constant memory. This distinction is illustrated in Figure 3.4.4. In practice, the implicit differentiation required by the backward pass can be efficiently implemented using modern automatic differentiation libraries such as `PyTorch` [27] or `JAX` [28], which support vector-Jacobian and Jacobian-vector products without explicitly constructing the full Jacobian matrix.

Chapter 4

Regularized and Unfolded NMF Algorithms

4.1 Regularized NMF

In the previous chapter, we introduced the fundamentals of NMF and derived the standard MU rules that guarantee convergence for the unregularized problem. While these updates form the backbone of many practical NMF algorithms, real-world datasets often benefit from additional constraints or priors. Indeed, we briefly remarked on how regularization is used in neural networks to prevent overfitting and promote a desirable solution structure. In this chapter, we carry that idea into the NMF setting by incorporating explicit penalty terms into the MU framework. First, we present an ℓ_1-penalized NMF, which enforces sparsity in the factor matrices. Next, we introduce an $\ell_{2,1}$-norm penalty to promote group-sparsity and enhance robustness against noise. Finally, we outline a hybrid $\ell_1/\ell_{2,1}$ scheme that leverages both individual-entry and group-level regularization, yielding factors that are interpretable, stable, and well-suited to noisy or heterogeneous data.

NMF with Sparseness Constraints. Sparse coding [48], is a scheme in which each data vector is expressed using only a small subset of an (often overcomplete) basis—ensuring that most basis functions remain inactive for any given input—thereby yielding efficient, robust, and easily interpretable representations. In effect, this implies that the most entries of a sparse vector take values close to zero while only few take significantly non-zero values. It has been observed that many datasets found in nature tend to follow some degree of sparsity. This natural tendency toward sparse structure aligns perfectly with NMF's inherent parts-based factorization, motivating the incorporation of explicit sparsity constraints into the NMF framework to extract even more localized, meaningful, and interpretable components.

Hoyer [49] was among the first to introduce sparsity directly in NMF by using a sparseness term in an iterative algorithm to update $\mathbf{W}$ and $\mathbf{H}$. He proposes the following sparsity measure of a vector $\mathbf{x}$:

$$S(\mathbf{x}) \;=\; \frac{\sqrt{n} \;-\; \dfrac{\|\mathbf{x}\|_1}{\|\mathbf{x}\|_2}}{\sqrt{n}-1}, \tag{4.1.1}$$

where n is the dimensionality of the vector $\mathbf{x}$. This definition of sparseness makes it so that $S(\mathbf{x}) = 1$ if $\mathbf{x}$ has exactly one nonzero entry (maximally sparse) and $S(\mathbf{x}) = 0$ if all entries of $\mathbf{x}$ are equal (maximally dense). This definition of sparsity leverages a combination of ℓ_1 and ℓ_2 regularization.

The formal statement of the NMF problem with sparseness constraints can be shown in Eq. 4.1.2:

Given a non-negative data matrix $\mathbf{V} \in \mathbb{R}^{n\times m}_{\geq 0}$, find non-negative matrices $\mathbf{W} \in \mathbb{R}^{n\times p}_{\geq 0}$ and $\mathbf{H} \in \mathbb{R}^{p\times m}_{\geq 0}$ such that

$$E(\mathbf{W},\mathbf{H}) \;=\; \big\|\mathbf{V}-\mathbf{W}\,\mathbf{H}\big\|_F^2 \tag{4.1.2}$$

is minimized, under the optional constraints

$$\text{sparseness}(\mathbf{w}_i) \;=\; S_w, \quad \forall\, i, \qquad \text{sparseness}(\mathbf{h}_i) \;=\; S_h, \quad \forall\, i,$$

where $\mathbf{w}_i$ denotes the ith column of $\mathbf{W}$ and $\mathbf{h}_i$ denotes the ith row of $\mathbf{H}$. Here p is the number of components, and S_w, S_h are the user-specified sparseness levels for $\mathbf{W}$ and $\mathbf{H}$, respectively.

To optimize the objective in (4.1.2) while enforcing the sparseness levels S_w and S_h, Hoyer proposes an iterative projected-gradient scheme. Starting from random non-negative initializations of $\mathbf{W}$ and $\mathbf{H}$, each iteration alternates between:

1. a small gradient step on the reconstruction error,
2. a projection that (i) zeroes out negative entries, (ii) rescales each vector to unit ℓ_2 norm, and (iii) adjusts the ℓ_1 norm so that the normalized sparseness of each column of $\mathbf{W}$ and each row of $\mathbf{H}$ exactly matches S_w and S_h, respectively.

The complete procedure is summarized in Algorithm 2.

The parameters $S_w, S_h \in [0,1]$ directly control the fraction of zero (or near-zero) entries in each column of $\mathbf{W}$ and row of $\mathbf{H}$. Values close to 1 enforce extreme sparsity (one nonzero entry), whereas values near 0 produce

Algorithm 2. NMF with Sparseness Constraints

1: Initialize $\mathbf{W} \in \mathbb{R}_{\geq 0}^{N \times M}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{M \times T}$ to random positive matrices.
2: If sparseness constraints on $\mathbf{W}$ apply:
3: Set $\mathbf{W} \leftarrow \mathbf{W} - \mu_W (\mathbf{W}\mathbf{H} - \mathbf{V})\mathbf{H}^T$
4: Project each column of $\mathbf{W}$ to be non-negative, keep its $\|\cdot\|_2$ unchanged, set its $\|\cdot\|_1$ to achieve the desired S_w.
5: Else:
6: Set $\mathbf{W} \leftarrow \mathbf{W} \odot (\mathbf{V}\mathbf{H}^T) \oslash (\mathbf{W}\mathbf{H}\mathbf{H}^T)$.
7: If sparseness constraints on $\mathbf{H}$ apply:
8: Set $\mathbf{H} \leftarrow \mathbf{H} - \mu_H \mathbf{W}^T(\mathbf{W}\mathbf{H} - \mathbf{V})$
9: Project each row of $\mathbf{H}$ to be non-negative, set its $\|\cdot\|_2 = 1$, and its $\|\cdot\|_1$ to achieve the desired S_h.
10: Else:
11: Set $\mathbf{H} \leftarrow \mathbf{H} \odot (\mathbf{W}^T\mathbf{V}) \oslash (\mathbf{W}^T\mathbf{W}\mathbf{H})$.
12: Repeat the above steps until convergence.

nearly dense factors. In practice, one selects (S_w, S_h) by cross–validation on held-out reconstruction error, by monitoring Hoyer's own sparseness measure on a validation set, or by leveraging domain knowledge (e.g. expected number of active sources per signal).

Robust NMF– $\ell_{2,1}$ via Split-Row Updates In their work Wang *et al.* [50], proposed a framework where NMF is regularized using the $\ell_{2,1}$ norm, that has been previously used as a variant of ℓ_1 norm in principal component analysis (PCA) for robust subspace factorization [51]. The $\ell_{2,1}$ norm of a matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ is defined as follows:

$$\|\mathbf{M}\|_{2,1} = \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{m} m_{ij}^2} = \sum_{i=1}^{n} \|\mathbf{m}_i\|_2 \tag{4.1.3}$$

where $\mathbf{m}_i$ denotes the ith column of $\mathbf{M}$.

The NMF-$\ell_{2,1}$ optimization problem is formulated as:

$$\min_{\mathbf{W},\mathbf{H} \geq 0} \left\|\mathbf{V} - \mathbf{W}\mathbf{H}\right\|_{2,1} + \lambda \left\|\mathbf{H}\right\|_{2,1}, \tag{4.1.4}$$

where λ is a scalar that controls the degree of regularization. In this scheme, the regularization term is designed to diminish the impact of noise or outliers contained in the original data.

4.2 Combined $\ell_1 - \ell_2$ Regularized NMF

A regularized variant of NMF implementing entry-wise ℓ_1 and ℓ_2 penalties on the coefficient matrix $\mathbf{H}$ was devised by Nasser *et al.* in [29]. In this formulation we are leaded to the following minimization objective:

$$\min_{\mathbf{H},\mathbf{W} \geq 0} \left\{ \frac{1}{2} \left\|\mathbf{V} - \mathbf{W}\mathbf{H}\right\|_F^2 + \lambda_1 \|\mathbf{H}\|_1 + \frac{\lambda_2}{2} \|\mathbf{H}\|_F^2 \right\}. \tag{4.2.1}$$

Drawing inspiration from the classical Multiplicative Update algorithm, Nasser *et al.* devise the following multiplicative updates which correspond to the objective in Eq. 4.2.1:

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^\top \mathbf{V}}{\mathbf{W}^\top \mathbf{W}\mathbf{H} + \lambda_1 \mathbf{1}_{p \times m} + \lambda_2 \mathbf{H}} \tag{4.2.2}$$

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\mathbf{V}\mathbf{H}^\top}{\mathbf{W}\mathbf{H}\mathbf{H}^\top} \tag{4.2.3}$$

where "$\odot$" denotes the Hadamard (element-wise) product, and divisions are taken element-wise.

It is proven that the minimization objective is non-increasing under these updates. The ℓ_1 term enforces fine-grained sparsity on each entry of $\mathbf{H}$, while the λ_2 term adds a shrinkage that smooths the coefficients and helps stabilize convergence.

4.3 Deep Unfolding of Regularized MU-NMF

In the previous chapter, we discussed the concept of Algorithm Unrolling and how we can "transform" an iterative blind algorithm into a trainable feedforward network by mapping each iteration to one layer and replacing certain fixed operations with learnable parameters. In this section, we will show how in their work [29], Nasser *et al.* apply this idea to the regularized version of the MU-NMF algorithm for both a supervised and an unsupervised framework.

As previously noted, the classical multiplicative updates for the regularized MU–NMF problem proposed by Nasser *et al.* in [29], take the form

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^\top \mathbf{V}}{\mathbf{W}^\top \mathbf{W}\,\mathbf{H} \;+\; \lambda_1\,\mathbf{1}_{p\times m} \;+\; \lambda_2\,\mathbf{H}} \quad , \quad \mathbf{W} \leftarrow \mathbf{W} \odot \frac{\mathbf{V}\,\mathbf{H}^\top}{\mathbf{W}\,\mathbf{H}\,\mathbf{H}^\top} \tag{4.3.1}$$

Supervised Deep NMF (DNMF). Building on the previously defined MU rule, we now describe its supervised deep-unfolded variant. In this approach, we consider to have knowledge of the ground truth coefficient matrix $\mathbf{H}$ and the ground truth data matrix $\mathbf{V}$. We will now present the structure and the properties of the supervised DNMF network.

Let $\mathbf{V} \in \mathbb{R}^{n\times m}$ be the data matrix and $\mathbf{H}^{(l)} \in \mathbb{R}^{p\times m}_{\geq 0}$ the coefficient estimates at layer l. For each layer $l = 0, \ldots, L-1$, we consider the following trainable, nonnegative matrices as network parameters:

- $\mathbf{A}^{(l)} = \mathbf{W}^{(l)T} \in \mathbb{R}^{p\times n}_{\geq 0}$,
- $\mathbf{B}^{(l)} = \left(\mathbf{W}^\top \mathbf{W}\right)^{(l)} \in \mathbb{R}^{p\times p}_{\geq 0}$,
- We also consider the global trainable scalar regularization parameters λ_1, λ_2.

This formulation ignores the depedency between the $\mathbf{W}^\top$ and $\mathbf{W}^\top\mathbf{W}$ terms, treating them as independent network parameters. To illustrate the layer-wise update that consists the forward pass of the supervised DNMF network, it is helpful to focus on a single sample column $\mathbf{v} \in \mathbb{R}^n_{\geq 0}$ and the corresponding coefficient column $\mathbf{h}^{(l)} \in \mathbb{R}^p_{\geq 0}$. At each layer l, the forward step in our network consists of applying the column-wise activation function f:

$$\mathbf{h}^{(l+1)} = f\big(\mathbf{h}^{(l)}, \mathbf{v}\big) \;=\; \mathbf{h}^{(l)} \;\odot\; \frac{\mathbf{A}^{(l)}\,\mathbf{v}}{\mathbf{B}^{(l)}\,\mathbf{h}^{(l)} \;+\; \lambda_1 \mathbf{1}_p \;+\; \lambda_2\,\mathbf{h}^{(l)}}\,. \tag{4.3.2}$$

The structure of a single layer of the supervised DNMF can be visualised in Fig 4.3.1. Having defined the properties of a single layer, we can now form a full unrolled network, by connecting ℓ layers. The input of each layer ℓ is the data $\mathbf{v}$ and the output of the previous layer $\mathbf{h}^{(l-1)}$. The full unrolled network with ℓ layers can be seen in Fig 4.3.2.

Because this multiplicative rule applies independently to every column $\mathbf{h}$, it admits a completely natural transition to a single, batched operation over the full data matrix—updating all columns of $\mathbf{H}$ in parallel, so we can express the update in the following matrix form:

$$\mathbf{H}^{(l+1)} = \mathbf{H}^{(l)} \;\odot\; \frac{\mathbf{A}^{(l)}\,\mathbf{V}}{\mathbf{B}^{(l)}\,\mathbf{H}^{(l)} \;+\; \lambda_1\,\mathbf{1}_{p\times m} \;+\; \lambda_2\,\mathbf{H}^{(l)}}. \tag{4.3.3}$$

We initialize $\mathbf{H}^{(0)}$ with a positive constant (e.g. all ones) and enforce nonnegativity of $\mathbf{A}^{(l)}, \mathbf{B}^{(l)}$ by projection on the nonnegative orthant after each gradient step.

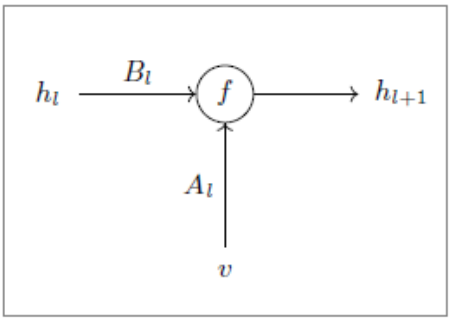


Figure 4.3.1: A single layer acting on $h^{(l)}$ and v.

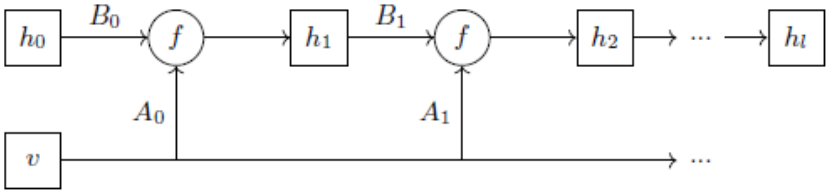


Figure 4.3.2: Unrolled network with ℓ layers.

Training the Supervised DNMF. To explain the training process of DNMF, we will once again use a single column approach. In training, we aim to optimize the following mean squared error loss:

$$\mathcal{L} \;=\; \left\| \mathbf{h}^{(L)} \;-\; \mathbf{h}' \right\|_2^2,$$

where L is the number of layers in our network, $\mathbf{h}^{(L)}$ is the output of the final layer and $\mathbf{h}'$ is the ground-truth coefficient column. In the case where we want to estimate the whole coefficient matrix, our model can be trained to predict all m coefficient columns at once. We generalize the loss to the squared Frobenius norm:

$$\mathcal{L} = \left\|\mathbf{H}^{(L)} - \mathbf{H}'\right\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(h_{ij}^{(L)} - h'_{ij}\right)^2 = \mathrm{tr}\left[(\mathbf{H}^{(L)} - \mathbf{H}')^\top(\mathbf{H}^{(L)} - \mathbf{H}')\right]. \tag{4.3.4}$$

Equivalently, this is the sum of the column-wise ℓ_2 losses:

$$\mathcal{L} = \sum_{i=1}^{m}\left\|\mathbf{h}_i^{(L)} - \mathbf{h}'_i\right\|_2^2. \tag{4.3.5}$$

This formulation allows the network to learn all coefficient maps jointly while maintaining a simple quadratic penalty.

After we compute the loss, we can obtain the gradients needed to update our network's trainable parameters:

$$\left\{\, \mathbf{A}^{(\ell)},\, \mathbf{B}^{(\ell)},\, \lambda_1,\, \lambda_2 \right\}_{\ell=0}^{L-1}$$

using backpropagation provided by the ADAM optimizer.

Because λ_1 and λ_2 are included as trainable variables, our network automatically "learns" the optimal regularization strength from data—eliminating the need for an outer loop of hyperparameter-grid search. In effect, every gradient step updates not only the forward/backward linear operators $\{\mathbf{A}^{(\ell)}, \mathbf{B}^{(\ell)}\}$ but also the regularization weights λ_1, λ_2. This yields data-adapted update rules that encode both reconstruction fidelity

(via the MSE term) and learned regularization, all in a single end-to-end training procedure. The steps of training the supervised DNMF network are summarized in Algorithm 3.

Algorithm 3. Supervised DNMF (Single Epoch)

Require: $\mathbf{H}_{\text{true}}$, $\mathbf{V}_{\text{true}}$, depth ℓ, regularizers λ_1, λ_2
Initialize $\lambda_1 \leftarrow 1,\ \lambda_2 \leftarrow 1$
for $\ell = 0, \ldots, L-1$ **do**
$\mathbf{A}^{(\ell)} \leftarrow \mathbf{1}_{p\times n}, \quad \mathbf{B}^{(\ell)} \leftarrow \mathbf{1}_{p\times p}$
end for
Forward:
$\mathbf{H} \leftarrow \mathbf{H}^{(0)}$
for $\ell = 0, \ldots, L-1$ **do**

$$\mathbf{H}^{(\mathbf{l}+\mathbf{1})} \leftarrow \mathbf{H}^{(\mathbf{l})} \odot \frac{\mathbf{A}^{(\ell)}\,\mathbf{V}}{\mathbf{B}^{(\ell)}\,\mathbf{H}^{(l)} + \lambda_2\,\mathbf{H}^{(l)} + \lambda_1\,\mathbf{1} + \varepsilon}$$

end for
$\mathbf{H}_{\text{pred}} \leftarrow \mathbf{H}^{(L)}$
Compute loss:
$\mathcal{L} = \frac{1}{2}\left\|\mathbf{H}_{\text{pred}} - \mathbf{H}_{\text{true}}\right\|_F^2$
Backward:
Compute $\nabla_{\mathbf{A}^{(\ell)}}\mathcal{L},\ \nabla_{\mathbf{B}^{(\ell)}}\mathcal{L},\ \frac{\partial\mathcal{L}}{\partial\lambda_1},\ \frac{\partial\mathcal{L}}{\partial\lambda_2}$
Update $\mathbf{A}^{(\ell)}, \mathbf{B}^{(\ell)}, \lambda_1, \lambda_2$ via Adam Optimizer
Enforce nonnegativity:
- $\mathbf{A}^{(\ell)} \leftarrow \max\big(\mathbf{A}^{(\ell)}, 0\big)$
- $\mathbf{B}^{(\ell)} \leftarrow \max\big(\mathbf{B}^{(\ell)}, 0\big)$
- $\lambda_1 \leftarrow \max\big(\lambda_1, 0\big)$
- $\lambda_2 \leftarrow \max\big(\lambda_2, 0\big)$

Unsupervised DNMF. We will now go on and describe the unsupervised version of DNMF in which neither $\mathbf{W}$ nor $\mathbf{H}$ is given, we only know the data matrix $\mathbf{V}$. The goal is to jointly estimate both the basis matrix $\mathbf{W}$ and $\mathbf{H}$ using a deep-unfolded framework. We outline the main steps below:

The forward process of the unsupervised DNMF model is similar to the supervised version, that is, we go through ℓ layers feeding a sample column $\mathbf{v} \in \mathbb{R}^n$ and obtain a coefficient estimate $\mathbf{h}^{(L)}$. Stacking these yields the entire coefficient matrix:

$$\widehat{\mathbf{H}} \;=\; \big[\,\mathbf{h}_1^{(L)},\,\mathbf{h}_2^{(L)},\,\ldots,\,\mathbf{h}_m^{(L)}\big] \;\in\; \mathbb{R}_{\geq 0}^{p\times m}.$$

The main difference between these aforementioned approaches lies in the backward step. In contrast to the supervised variant, we do not know the ground-truth matrix $\mathbf{H}$, thus we can no longer use it in the loss function to update our network parameters via backpropagation. Therefore, we now implement a different loss function, i.e.,

$$\mathcal{L} \;=\; \frac{1}{2}\left\|\,\mathbf{V} \;-\; \widehat{\mathbf{W}}\,\widehat{\mathbf{H}}\right\|_F^2 \;+\; \lambda\left\|\,\widehat{\mathbf{H}}\right\|_1 \;+\; \tfrac{\lambda}{2}\left\|\,\widehat{\mathbf{H}}\right\|_F^2. \tag{4.3.6}$$

As someone can notice, the loss function uses $\widehat{\mathbf{W}}$, which is a prediction of the basis matrix. In each epoch of the training process, once $\widehat{\mathbf{H}}$ is obtained from the forward pass, we solve the following nonnegative least-squares problem to estimate $\mathbf{W}$:

$$\widehat{\mathbf{W}} \;=\; \arg\min_{\mathbf{W}\,\geq\,0}\left\|\,\mathbf{V} \;-\; \mathbf{W}\,\widehat{\mathbf{H}}\right\|_F^2. \tag{4.3.7}$$

Since $\widehat{\mathbf{H}}$ is fixed in this step, (4.3.7) decomposes into independent NNLS subproblems (one per row of $\mathbf{W}$). The NNLS subproblems are solved by either the classic Lawson–Hanson active-set method [52] or by the faster algorithm of Bro and de Jong [53].

Once we compute $\widehat{\mathbf{W}}$, we can train our model based on the reconstruction error, having knowledge over only the data matrix $\mathbf{V}$. This structure of the unsupervised DNMF network can be depicted for better clarity in Fig. 4.3.3. We also provide the steps to train this network for a single epoch in Algorithm 4.

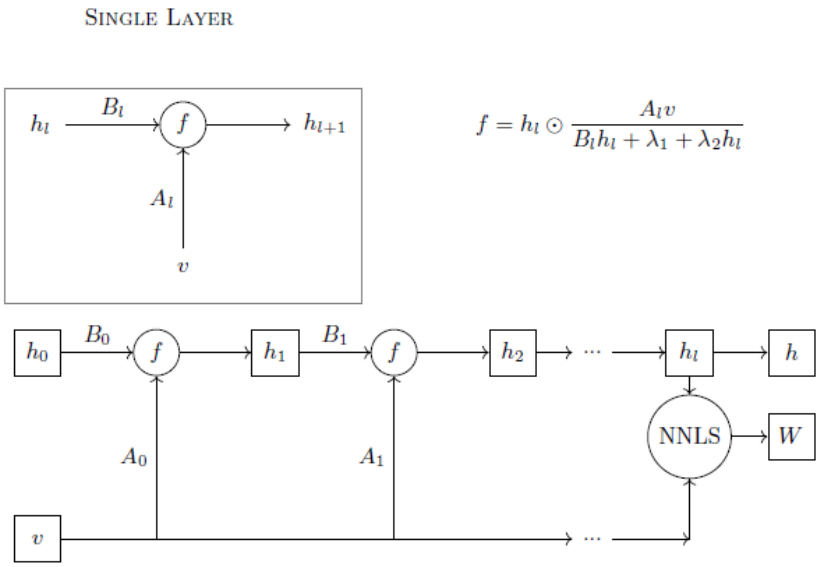


Figure 4.3.3: An illustration of the unrolled deep network for the unsupervised DNMF version. From [29].

Algorithm 4. Unsupervised DNMF

Require: $\mathbf{V}$, ℓ, k, η, λ_1, λ_2, T

Initialize:

For each $\ell = 0, \ldots, L-1$, set $\mathbf{A}^{(\ell)} \leftarrow \mathbf{1}_{k \times f}$, $\mathbf{B}^{(\ell)} \leftarrow \mathbf{1}_{k \times k}$ Set $\mathbf{H}^{(0)} \leftarrow \mathbf{1}_{k \times n}$ Initialize Adam on parameters $\{\mathbf{A}^{(\ell)}, \mathbf{B}^{(\ell)}, \lambda_1, \lambda_2\}$ with rate η **for** *epochs* $t = 1, \ldots, T$ **do**

Forward:

$\mathbf{H} \leftarrow \mathbf{H}^{(0)}$ **for** $\ell = 0, \ldots, L-1$ **do**

$$\mathbf{H}^{(\ell+1)} \leftarrow \mathbf{H}^{(\ell)} \odot \frac{\mathbf{A}^{(\ell)}\mathbf{V}}{\mathbf{B}^{(\ell)}\mathbf{H}^{(\ell)} + \lambda_2 \mathbf{H}^{(\ell)} + \lambda_1 \mathbf{1} + \varepsilon}$$

$\mathbf{H}_{\text{pred}} \leftarrow \mathbf{H}^{(L)}$

Estimate W (NNLS): Solve $\widehat{\mathbf{W}} = \arg\min_{\mathbf{W} \geq 0} \left\| \mathbf{V}_{\mathbf{true}} - \mathbf{W}\,\mathbf{H}_{\text{pred}} \right\|_F^2$

Compute loss:

$$\mathcal{L} = \frac{1}{2} \left\| \mathbf{V} - \widehat{\mathbf{W}}\,\mathbf{H}_{\text{pred}} \right\|_F^2 + \lambda_1 \left\| \mathbf{H}_{\text{pred}} \right\|_1 + \frac{\lambda_2}{2} \left\| \mathbf{H}_{\text{pred}} \right\|_F^2$$

Backward:

Compute gradients $\nabla_{\mathbf{A}^{(\ell)}}\mathcal{L}$, $\nabla_{\mathbf{B}^{(\ell)}}\mathcal{L}$, $\frac{\partial \mathcal{L}}{\partial \lambda_1}$, $\frac{\partial \mathcal{L}}{\partial \lambda_2}$ Update each $\mathbf{A}^{(\ell)}$, $\mathbf{B}^{(\ell)}$, λ_1, λ_2 via Adam and then project to nonnegative:

$$\mathbf{A}^{(\ell)} \leftarrow \max\left(\mathbf{A}^{(\ell)}_{\text{new}}, 0\right), \quad \mathbf{B}^{(\ell)} \leftarrow \max\left(\mathbf{B}^{(\ell)}_{\text{new}}, 0\right), \quad \lambda_1 \leftarrow \max\left(\lambda_{1,\text{new}}, 0\right), \quad \lambda_2 \leftarrow \max\left(\lambda_{2,\text{new}}, 0\right).$$

return $\mathbf{H}^{\text{pred}}, \widehat{\mathbf{W}}, \{\mathbf{A}^{(\ell)}, \mathbf{B}^{(\ell)}\}_{\ell=0}^{L-1}, \lambda_1, \lambda_2$

4.4 Data Description

In this subsection we aim to explain the nature of the data we will be conducting our experiments later on. We used simulated mutation data sets, in which the data matrix $\mathbf{V}$ is assumed to be the result of the activity of certain mutational processes whose signatures are given by the dictionary $\mathbf{W}$ and whose exposures

are given by the coefficient matrix **H**. To understand the structure of our datasets we must first establish the molecular foundation underlying the mutation data. In particular, we begin by reviewing the structure of the DNA double helix [30] and the principle of complementary base pairing, which together explain how single-base substitutions arise and how they are catalogued in order to form out data.

Complementary Base-Pairing. DNA is a double-stranded polymer in which each nucleotide base on one strand is hydrogen-bonded to its complement on the opposite strand. This is visualized in Fig. 4.4.1. There are 4 nucleotide bases in total, adenine (A), guanine (G), cytosine (C), and thymine (T). Adenine bonds with thymine and guanine bonds with cytosine meaning we can only have connections of the form:

$$\mathrm{A} \longleftrightarrow \mathrm{T} \quad \text{and} \quad \mathrm{G} \longleftrightarrow \mathrm{C}.$$

Mutations can arise through several different mechanisms—single-base substitutions (SBS), small insertions, or small deletions—each of which alters the DNA sequence in a distinct way. For now, we will focus on SBS, in which one nucleotide is replaced by another without changing the overall length of the DNA strand. When a single-base substitution occurs (for example A→G) on the "forward" strand, replication or repair will eventually produce the complementary change (T→C) on the opposite strand. Because these two events are biologically equivalent, we conventionally re-express all purine-centered changes (A→X or G→X) as their pyrimidine-centered reverse complements (T→Y or C→Y, where Y is the complement of X).

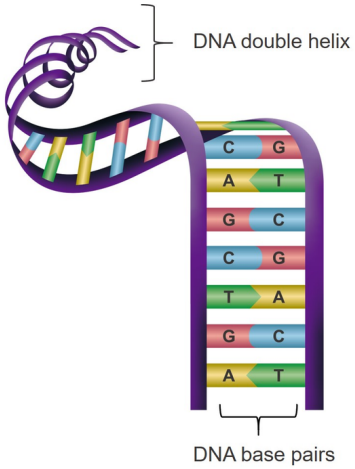


Figure 4.4.1: DNA double helix and nucleotide bonds illustration. From [31].

Next, we can write down each single substitution mutation and count 12 distinct base-substitutions in total:

$$A \to C,\ A \to G,\ A \to T, \tag{4.4.1}$$
$$C \to A,\ C \to G,\ C \to T, \tag{4.4.2}$$
$$G \to A,\ G \to C,\ G \to T, \tag{4.4.3}$$
$$T \to A,\ T \to C,\ T \to G \tag{4.4.4}$$

As we explained earlier, knowing the base on one strand of the DNA double helix uniquely determines its partner on the opposite strand, since each base pairs only with its specific complement. Therefore, when classifying substitution mutations, we conclude on the following six canonical mutations:

$$\{\,\mathrm{C}\to\mathrm{A},\ \mathrm{C}\to\mathrm{G},\ \mathrm{C}\to\mathrm{T},\ \mathrm{T}\to\mathrm{A},\ \mathrm{T}\to\mathrm{C},\ \mathrm{T}\to\mathrm{G}\}. \tag{4.4.5}$$

Trinucleotide Contexts. To capture local sequence effects, we record the identity of the bases immediately preceding and following the central substitution as proposed by [31]. With four possible preceding bases and four possible following bases, there are

$$4 \times 4 \;=\; 16$$

ordered flank-pairs. Combining these with the six central substitutions yields

$$6 \times 16 \;=\; 96$$

distinct trinucleotide contexts. An example substitution within context would be A[C→T]G, where the C→T change occurs with an A immediately before and a G immediately after.

Data Matrix V. The data matrix used in our experiments—and in prior studies—comprises m samples, each represented by a 96-entry mutation-count profile.

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{961} & v_{962} & \cdots & v_{96m} \end{bmatrix} \in \mathbb{R}^{96\times m}.$$

where $\mathbf{v}_{ij}$ is the number of times that mutation i was observed in sample j and m is the number of samples.

Signature and Exposure Matrices. In the supervised version of DNMF, experiments were conducted on simulated $\mathbf{W}$ and $\mathbf{H}$ matrices. Those factor-matrices and the factorization rank p were obtained using NMF, in the work of Alexandrov *et al.* [14]. Hence, we have a classic factorization scheme:

$$\mathbf{V} \;\approx\; \mathbf{W}\,\mathbf{H},$$

with

$$\mathbf{W} \in \mathbb{R}^{96\times p}_{\geq 0}, \quad \mathbf{H} \in \mathbb{R}^{p\times m}_{\geq 0}.$$

The basis matrix $\mathbf{W}$ is also called the signature matrix in this application. We define a mutational signature as a combination of some of the 96 mutations. Notice that the rank p that was obtained through simulations, resembles the total number of signatures extracted from our data. Below we can see the structure of the signature matrix $\mathbf{W}$:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1p} \\ w_{21} & w_{22} & \cdots & w_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ w_{961} & w_{962} & \cdots & w_{96p} \end{bmatrix} \in \mathbb{R}^{96\times p}.$$

Each signature can be viewed as a discrete probability distribution over the 96 possible mutation contexts, so the element where $\mathbf{w}_{ij}$ can be viewed as the probability of the jth signature containing the ith mutation. Following this model, its natural that the sum of the elements of each column to be equal to one. So we have:

The exposure matrix $\mathbf{H}$ encodes the exposures (or activities) of each signature in each sample or in other words, how many mutations (or what fraction of all mutations) are attributed to each signature. Therefore we have:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1m} \\ h_{21} & h_{22} & \cdots & h_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ h_{p1} & h_{p2} & \cdots & h_{pm} \end{bmatrix} \in \mathbb{R}^{p\times m}.$$

In the exposure matrix, each entry $\mathbf{h}_{ij}$ denotes the number of mutations in sample j that were associated with the ith signature.

Chapter 5

Trainable DEQ-NMF Algorithms

5.1 Deep Equilibrium Algorithms

In this chapter, we present a unified framework for supervised NMF built on DEQ principles. We introduce two DEQ models: a baseline equilibrium solver with implicit differentiation, and a modified variant that extends the first by altering its parameterization. We then propose three equilibrium-approximation schemes, each replacing iterative layers with a single closed-form ReLU estimate of the fixed point and incorporating specific modifications to their parameter networks and loss functions. For all five models, we detail the forward pass, the backward pass and the training process. Finally, we present extensive experiments on both simulated cancer mutational data and noisy synthetic data with various SNRs, to validate the performance of the proposed algorithms, demonstrating their advantages in both estimation accuracy and computational efficiency, as compared to previous related schemes.

5.1.1 Deep Equilibrium NMF

In this subsection, we present the implementation of a DEQ model for NMF named DEQ-NMF. In this framework, we formulate the NMF coefficient matrix $\mathbf{H}$ as the fixed point of a nonlinear update map with trainable parameters. Using the classic multiplicative update for $\mathbf{H}$ and retaining the network parameters proposed in DNMF we define the equilibrium equation:

$$\mathbf{H}^\star = f(\mathbf{H}^\star) = \mathbf{H}^\star \odot \frac{\mathbf{AV}}{(\mathbf{B} + \lambda_2 \mathbf{I})\mathbf{H}^\star + \lambda_1}.$$

Forward Pass. As the DEQ principle indicates, we use a numerical solver to find the fixed point $\mathbf{H}^*$ satisfying $f(\mathbf{H}^*) = \mathbf{H}^*$. Concretely, given the input matrix $\mathbf{V}$ and current parameters, the equilibrium coefficient matrix $\mathbf{H}^\star$ is found by iteratively applying the update map f:

$$\mathbf{H}^{(t+1)} = f(\mathbf{H}^{(t)})$$

starting from an initialization $\mathbf{H}^{(0)}$. The iteration proceeds for a fixed number of steps (e.g., $T = 50$) or until convergence criteria are met, approximating the fixed point:

$$\mathbf{H}^\star \approx \mathbf{H}^{(T)}.$$

The choice of the forward solver does not affect the backward differentiation in the DEQ setting.

Backward Pass. After obtaining the equilibrium $\mathbf{H}^\star$ in the forward pass, we implement a custom backward (gradient) computation for the DEQ layer. Rather than differentiating through all the fixed-point iterations, we leverage the implicit function theorem to compute gradients directly at the equilibrium. Considering the fixed point solution:

$$\mathbf{H}^* = f(\mathbf{H}^*, \mathbf{V}).$$

Our goal will be to compute the vector-Jacobian product

$$\left(\frac{\partial \mathbf{H}^*}{\partial \theta}\right)^\top \mathbf{y}$$

for some vector $\mathbf{y}$, where θ here is a stand-in for any quantity we want to differentiate the fixed point with respect to. Since this vector-Jacobian product is the key aspect to integrating these DEQ layers within back-propagation, such a routine allows us to integrate the DEQ layer within standard automatic differentiation tools.

By differentiating both sides of the fixed point solution w.r.t θ, we have

$$\frac{\partial \mathbf{H}^*}{\partial \theta} = \frac{\partial f(\mathbf{H}^*, \mathbf{V})}{\partial \theta} + \frac{\partial f(\mathbf{H}^*, \mathbf{V})}{\partial \mathbf{H}^*} \frac{\partial \mathbf{H}^*}{\partial \theta},$$

where θ represents any network parameter with respect to which we differentiate the equilibrium point. Then, rearranging terms, we get an explicit expression for the Jacobian:

$$\frac{\partial \mathbf{H}^*}{\partial \theta} = \left(I - \frac{\partial f(\mathbf{H}^*, \mathbf{V})}{\partial \mathbf{H}^*}\right)^{-1} \frac{\partial f(\mathbf{H}^*, \mathbf{V})}{\partial \theta}.$$

Finally, to compute the vector-Jacobian product, we have that:

$$\left(\frac{\partial \mathbf{H}^*}{\partial \theta}\right)^\top \mathbf{y} = \left(\frac{\partial f(\mathbf{H}^*, \mathbf{V})}{\partial \theta}\right)^\top \left(I - \frac{\partial f(\mathbf{H}^*, \mathbf{V})}{\partial \mathbf{H}^*}\right)^{-\top} \mathbf{y}.$$

Here, $\mathbf{y}$ is an arbitrary vector of the same dimension as the equilibrium variable $\mathbf{H}^*$; in the context of backpropagation, it is typically taken as the gradient of the loss with respect to $\mathbf{H}^*$, i.e., $\mathbf{y} = \dfrac{\partial \mathcal{L}}{\partial \mathbf{H}^*}$.

Let's consider how we compute this quantity in practice. The key term of interest here is the solution $\mathbf{g}$ of the linear system:

$$\mathbf{g} = \left(I - \frac{\partial f(\mathbf{H}^*, \mathbf{V})}{\partial \mathbf{H}^*}\right)^{-\top} \mathbf{y}$$

By solving for $\mathbf{g}$ either by direct inversion or some iterative solver we can finally compute the final Jacobian vector product as:

$$\mathbf{g} = \left(\frac{\partial f(\mathbf{H}^*, \mathbf{V})}{\partial \mathbf{H}^*}\right)^\top \mathbf{g} + \mathbf{y}.$$

In our code, this is efficiently implemented using JAX's automatic differentiation and `custom_vjp` functionality, which allows us to specify this implicit backward logic. The vector–Jacobian products are computed without explicitly forming large Jacobian matrices, enabling scalable and memory-efficient training.

Training. The DEQ-NMF layer is trained end-to-end by embedding the equilibrium solver as a differentiable implicit layer within a standard optimizer loop. All parameters are projected to the nonnegative orthant via a ReLU transformation at each iteration. The whole training process is summarized in Algorithm 5.

Algorithm 5. End-to-End Training of DEQ-NMF with Learnable $\mathbf{A}$, $\mathbf{B}$, λ_1, λ_2

Require: Training data $(\mathbf{V}, \mathbf{H}_{\text{true}})$, unconstrained parameters $\theta = (\mathbf{A}_{\text{raw}}, \mathbf{B}_{\text{raw}}, \lambda_{1,\text{raw}}, \lambda_{2,\text{raw}})$, learning rate η

Initialize Adam optimizer state
for epoch $= 1, 2, \ldots, N_{\text{epochs}}$ **do**
  # **Positivity transform**: ensure all parameters are nonnegative
  $\mathbf{A} \leftarrow \max(\mathbf{A}, 0)$
  $\mathbf{B} \leftarrow \max(\mathbf{B}, 0)$
  $\lambda_1 \leftarrow \max(\lambda_1, 0)$
  $\lambda_1 \leftarrow \max(\lambda_2, 0)$
  # **Equilibrium solve (forward pass)**
  Initialize $\mathbf{H}^{(0)}$ (e.g., all entries 0.1)
  for $t = 0, 1, \ldots, T-1$ **do**
    $\mathbf{H}^{(t+1)} \leftarrow f(\mathbf{H}^{(t)};\ \mathbf{A}, \mathbf{B}, \lambda_1, \lambda_2, \mathbf{V})$
    if converged **then break**
  end for
  $\mathbf{H}^\star \leftarrow \mathbf{H}^{(t+1)}$ # Fixed point
  # **Compute loss (e.g., MSE)**
  $\ell \leftarrow \text{MSE}(\mathbf{H}^\star,\ \mathbf{H}_{\text{true}})$
  # **Implicit backward pass (custom VJP)**
  Solve for $\mathbf{u}$ (adjoint variable) in:
    $\left[I - \frac{\partial f}{\partial \mathbf{H}}\big|_{\mathbf{H}^\star}\right]^\top \mathbf{u} = \nabla_{\mathbf{H}^\star} \ell$
    (Use K fixed-point iterations; e.g., $\mathbf{u}^{(k+1)} \leftarrow [\frac{\partial f}{\partial \mathbf{H}}\big|_{\mathbf{H}^\star}]^\top \mathbf{u}^{(k)} + \nabla_{\mathbf{H}^\star} \ell$)
  Compute gradients:
    $\nabla_{\mathbf{A}} \ell,\ \nabla_{\mathbf{B}} \ell,\ \nabla_{\lambda_1} \ell,\ \nabla_{\lambda_2} \ell$ using $\mathbf{u}$
  # **Adam optimizer update**
  Update unconstrained parameters $(\mathbf{A}_{\text{raw}}, \mathbf{B}_{\text{raw}}, \lambda_{1,\text{raw}}, \lambda_{2,\text{raw}})$ using Adam and learning rate η
end for

This approach enables memory-efficient, scalable, and fully end-to-end training of NMF within the DEQ paradigm.

DEQ-NMF with Shared Basis Matrix and Regularization Parameters. As mentioned previously, the DNMF network chooses to ignore the dependency of the $\mathbf{W}^\top$ and $\mathbf{W}^\top\mathbf{W}$ terms, treating them as independent trainable parameters $\mathbf{A}$ and $\mathbf{B}$. In the NMF problem: $\mathbf{V} \approx \mathbf{WH}$, we aim to find both the ground-truth matrices $\mathbf{W}$ and $\mathbf{H}$. During our experiments however, we found out that both DNMF and our proposed approach failed to estimate the basis matrix $\mathbf{W}$ accurately, i.e., the model parameter $\mathbf{A}^\top$. Therefore, even though those networks manage to provide accurate estimations of the coefficient matrix $\mathbf{H}$, the overall reconstruction error $\|\mathbf{V}_{\text{true}} - \mathbf{A}^\top\mathbf{H}_{\text{pred}}\|_F^2$ remained high. We naturally wondered, what would be the behavior of our model if we use only one learnable matrix parameter. Hence, we also consider a streamlined variant called DEQ-NMF-A where the NMF update map is parameterized by a single trainable basis matrix $\mathbf{A}$ and scalar regularization parameters λ_1 and λ_2. In this setting, the fixed-point equation reduces to:

$$\mathbf{H}^\star = \mathbf{H}^\star \odot \frac{\mathbf{AV}}{(\mathbf{AA}^\top)\mathbf{H}^\star + \lambda_2\mathbf{H}^\star + \lambda_1}$$

where all operations are elementwise. The training and differentiation procedures are fully analogous to those described above, with the only difference being that $\mathbf{B}$ is tied to $\mathbf{A}^\top\mathbf{A}$ and only $\mathbf{A}$, λ_1, and λ_2 are learned. This variant reduces the parameter count and enforces a classical NMF structure while still enabling end-to-end training with implicit differentiation through the fixed point. Ultimately, our experiments confirmed that this parameter sharing strategy resulted in increased accuracy for both $\mathbf{H}_{\text{true}}$ estimation and $\mathbf{V}_{\text{true}}$ reconstruction compared to the unconstrained approach.

5.1.2 Equilibrium Approximation Schemes

In this subsection, we introduce three equilibrium-approximation models that build on our DEQ reformulation of supervised NMF. Each of the proposed models replaces the usual multi-layer iteration with a single closed-form ReLU estimate of the equilibrium and enables explicit differentiation in the backward pass. We then differentiate the variants by their specialized parameter networks and tailored loss functions, all of which reduce memory usage and deliver significant improvements in both computational efficiency and estimation accuracy over standard deep unrolling and MU-based NMF approaches.

We begin by introducing A-DEQ-NMF [32], a model that preserves the original trainable parameters of DNMF and show in detail the training process that aims to learn the ground truth matrices $\mathbf{H}_{\text{true}}$ and $\mathbf{W}_{\text{true}}$.

Forward Pass.

In our forward step, we aim to solve for the equilibrium point of our deep network. We previously defined how DEQs try to find the equilibrium point of a function f, simulating an infinite layer deep network, using a numerical solver. In our case, using the column wise approach, the activation function that is applied in each layer is:

$$f(\mathbf{h};\mathbf{v}) = \mathbf{h} \odot \frac{\mathbf{Av}}{\mathbf{Bh} + \lambda_1\mathbf{1} + \lambda_2\mathbf{h}}. \tag{5.1.1}$$

Using ideas inspired by DEQs, we substitute the equilibrium point $\mathbf{h}^*$ in Eq. (5.1.1) and try to define it explicitly:

$$f(\mathbf{h}^*) = \mathbf{h}^* \longrightarrow \mathbf{h}^* = \mathbf{h}^* \odot \frac{\mathbf{Av}}{\mathbf{Bh}^* + \lambda_1\mathbf{1} + \lambda_2\mathbf{h}^*}. \tag{5.1.2}$$

Due to the nature of NMF we have already assumed that the elements of $\mathbf{h}^*$ are nonnegative. Unfortunately, we can't continue solving for a closed expression of the equilibrium point without assuming that the fixed

point $\mathbf{h}^*$ has no zero elements. Therefore, adopting this assumption, Eq. (5.1.2) can be rewritten as:

$$\mathbf{B}\mathbf{h}^* + \lambda_1 \mathbf{1} + \lambda_2 \mathbf{h}^* = \mathbf{A}\mathbf{v}, \;\; \mathbf{h}^* > 0, \tag{5.1.3}$$

or equivalently:

$$(\mathbf{B} + \lambda_2 \mathbf{I}_p)\mathbf{h}^* = \mathbf{A}\mathbf{v} - \lambda_1 \mathbf{1}, \;\; \mathbf{h}^* > 0. \tag{5.1.4}$$

Due to the positivity constraints $\mathbf{h}^* > 0$, obtaining an analytical expression for the equilibrium from (5.1.4) is not feasible. To circumvent the need for an iterative scheme while maintaining the nonnegativity constraints of classical NMF, we enforce nonnegativity at the end of our forward step by applying a ReLU and propose the following closed-form ReLU-type approximation of the equilibrium, which can be readily derived from (5.1.4):

$$\hat{\mathbf{h}}^* = \max\left[(\mathbf{B} + \lambda_2 \mathbf{I}_p)^{-1}(\mathbf{A}\mathbf{v} - \lambda_1 \mathbf{1}_{p\times 1}), 0\right], \tag{5.1.5}$$

or in matrix form

$$\hat{\mathbf{H}}^* = \max\left[(\mathbf{B} + \lambda_2 \mathbf{I}_p)^{-1}(\mathbf{A}\mathbf{V} - \lambda_1 \mathbf{1}_{p\times m}, 0\right], \tag{5.1.6}$$

where the maximum is taken elementwise. Notice that this approximation involves solving this linear system in a single step, in contrast to classic forward steps, in which the number of operations performed heavily depend on the chosen number of layers of each network.

Backward Pass. In Chapter 3, we explained the backward process of classic DEQ networks. Recall that since the equilibrium point is obtained through a black box root finder, we do not have an analytical expression for it. Therefore, we are not able to differentiate it explicitly and we must leverage implicit differentiation techniques. However, in our approach, we concluded to a closed expression for the equilibrium point $\mathbf{h}^*$ and we can use standard explicit differentiation methods. In our backward step, we choose to deploy standard ADAM-based backpropagation to update our network parameters $\mathbf{A}, \mathbf{B}, \lambda_1, \lambda_2$, retaining the original loss function used in DNMF, that is:

$$\mathcal{L} = \text{MSE}(\mathbf{h}^*, \mathbf{h}') = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{h}_i^* - \mathbf{h}_i'\right)^2$$

where h' is a column of the ground-truth coefficient matrix. The loss can be naturally expressed in matrix form.

In practice, we rely on existing optimized Python libraries for gradient computations and updates. However we provide analytical expressions for the parameter updates using the explicit derivatives of the equilibrium point. Below we present the gradient descent updates of our network parameters in a compact matrix form. The step to step derivation of those updates from a single column approach can be found in the Appendix.

$$\mathbf{A}^+ \;=\; \mathbf{A} \;-\; 2\,\alpha\,\left[\mathbf{M}^{-1}\left(\mathbf{H}^* - \mathbf{H}_{\text{true}}\right)\right)\,\mathbf{V}^\top] \tag{5.1.7}$$

$$\mathbf{B}^+ = \mathbf{B} \;+\; 2\,\alpha\,\left[\mathbf{M}^{-1}(\mathbf{H}^* - \mathbf{H}_{\text{true}})\,\left(\mathbf{M}^{-1}\mathbf{Y}\right)^\top\right] \tag{5.1.8}$$

$$\lambda_1^+ = \lambda_1 \;+\; 2\,\alpha\;\text{tr}\left[(\mathbf{H}^* - \mathbf{H}_{\text{true}})^\top(\mathbf{M}^{-1}\mathbf{1}_p\,\mathbf{1}_m^\top)\right] \tag{5.1.9}$$

$$\lambda_2^+ = \lambda_2 \;+\; 2\,\alpha\;\text{tr}\left[(\mathbf{H}^* - \mathbf{H}_{\text{true}})^\top\,\mathbf{M}^{-2}\,\mathbf{Y}\right] \tag{5.1.10}$$

where $\mathbf{M} = \mathbf{B} \;+\; \lambda_2\,\mathbf{I}_p \in \mathbb{R}^{p\times p}, \mathbf{Y} = \mathbf{A}\,\mathbf{V} \;+\; \lambda_1\,\mathbf{1}_p\,\mathbf{1}_m^\top \in \mathbb{R}^{p\times m}$.

During our experiments we implemented this method, benchmarked it against other models, and summarized the steps in our code in Algorithm 6. Unfortunately, we found that relying solely on plain gradient descent

was insufficient to achieve competitive performance, especially compared to more advanced optimizers such as ADAM.

Algorithm 6. Proposed Training Procedure Using Gradient Descent

Require: Input data matrix $\mathbf{V} \in \mathbb{R}^{n \times m}$, ground-truth coefficients $\mathbf{H}_{\text{true}} \in \mathbb{R}^{p \times m}$
Initialize $\mathbf{A} \leftarrow \mathbf{1}_{p \times n},\ \mathbf{B} \leftarrow \mathbf{1}_{p \times p},\ \lambda_1 \leftarrow 1,\ \lambda_2 \leftarrow 1$

Forward Step:
Compute $\mathbf{M} \leftarrow \mathbf{B} + \lambda_2 \mathbf{I}_p,\ \mathbf{Y} \leftarrow \mathbf{A}\,\mathbf{V} \ - \ \lambda_1\,\mathbf{1}_p\,\mathbf{1}_m^\top$
Compute equilibrium approximation: $\hat{\mathbf{H}}^* \leftarrow \max\big[\mathbf{M}^{-1}\,\mathbf{Y},\, 0\big]$

Backward Step:
$\mathbf{A} \ \leftarrow \ \mathbf{A} \ - \ 2\,\alpha\,\mathbf{M}^{-1}(\mathbf{H}^* - \mathbf{H}_{\text{true}})\,\mathbf{V}^\top$
$\mathbf{B} \ \leftarrow \ \mathbf{B} \ + \ 2\,\alpha\,\mathbf{M}^{-1}(\mathbf{H}^* - \mathbf{H}_{\text{true}})\,\big(\mathbf{M}^{-1}\mathbf{Y}\big)^\top$
$\lambda_1 \ \leftarrow \ \lambda_1 \ + \ 2\,\alpha\ \text{tr}\Big[\mathbf{M}^{-1}(\mathbf{H}^* - \mathbf{H}_{\text{true}})(\mathbf{M}^{-1}\mathbf{1}_p\,\mathbf{1}_m^\top)\Big]$
$\lambda_2 \ \leftarrow \ \lambda_2 \ + \ 2\,\alpha\ \text{tr}\Big[(\mathbf{H}^* - \mathbf{H}_{\text{true}})\mathbf{M}^{-2}\,\mathbf{Y}\Big]$

While attempting to integrate the idea of a single implicit layer from DEQs into the DNMF network, we ultimately derived a closed-form expression that approximates the equilibrium solution. As someone may notice, this model for supervised NMF that we just presented, is nothing more than an iterative algorithm itself, utilizing the power of ADAM optimizer instead of doing the classic MU updates for $\mathbf{W}$. However, this method yields a substantial reduction in parameters relative to the original DNMF, and later shows superior accuracy in the conducted experiments, compared to DNMF and the classic MU algorithm.

A-DEQ-NMF using only parameter A. Following the same idea that led us to implement DEQ-NMF-A, we present a modified version of DEQ-NMF named A-DEQ-NMF-A, where we consider only $\mathbf{A}$ as a matrix network parameter for our model.

Forward Pass.

Throughout the forward pass, the only thing that differs is that we now compute the term $\mathbf{W}^\top\mathbf{W}$ using $\mathbf{A}$ as shown in Eq. (5.1.11):

$$\hat{\mathbf{H}}^* = \max\left[(\mathbf{A}\mathbf{A}^\top + \lambda_2\mathbf{I}_p)^{-1}(\mathbf{A}\mathbf{V} - \lambda_1\mathbf{1}_{p\times m}), 0\right], \tag{5.1.11}$$

Backward Pass. In a similar manner to the forward pass, this network modification causes very little change to the backward pass. Essentially, now we do not compute the gradient of the approximation of the equilibrium point w.r.t to $\mathbf{B}$, but we need to compute the more complex gradient of $\mathbf{H}^*$ w.r.t to $\mathbf{A}$.

The next step after computing this new gradient is updating our network parameters $\mathbf{A}, \lambda_1, \lambda_2$ using the same training procedure as we did previously. In our experiments, we measured the the NSE between the ground-truth matrices and the matrices predicted by our models, verifying our intuition about improved accuracy when we choose to not ignore the dependency between the parameters A and B in the original MU equation for $\mathbf{H}$.

A-DEQ-NMF-A incorporating the reconstruction loss. To directly penalize errors in the data matrix reconstruction, we augment the previous model with an explicit reconstruction loss term and introduce a variant named A-DEQ-NMF-A1. Concretely, alongside the loss regarding the equilibrium-based estimation, we introduce the reconstruction loss:

$$\mathcal{L}_{\text{rec}} = \left\|\mathbf{V}_{\text{true}} \ - \ \mathbf{A}^\top\,\hat{\mathbf{H}}^*\right\|_F^2$$

and optimize the joint objective

$$\mathcal{L} = \left\|\hat{\mathbf{H}}^* \ - \ \mathbf{H}_{\text{true}}\right\|_F^2 \ + \ \alpha\,\mathcal{L}_{\text{rec}},$$

where the scalar $\alpha > 0$ balances coefficient and reconstruction-error terms.

Forward Pass.

Like before, we compute the equilibrium estimate $\hat{\mathbf{H}}^*$ via

$$\hat{\mathbf{H}}^* = \max\Big[(\mathbf{A}\mathbf{A}^\top + \lambda_2 \mathbf{I}_p)^{-1}(\mathbf{A}\mathbf{V} \; - \; \lambda_1 \mathbf{1}_{p\times m}),\, 0\Big]. \tag{5.1.12}$$

We then form the reconstruction

$$\widehat{\mathbf{V}} = \mathbf{A}^\top \hat{\mathbf{H}}^*.$$

Backward Pass and Parameter Updates. Backpropagation now must account for two gradient contributions w.r.t. $\mathbf{H}^*$, namely

$$\underbrace{\nabla_{\mathbf{H}^*} \left\|\mathbf{H}^* - \mathbf{H}_{\text{true}}\right\|_{\text{F}}^2}_{\text{coefficient-error}} = 2\big(\mathbf{H}^* - \mathbf{H}_{\text{true}}\big),$$

$$\underbrace{\nabla_{\mathbf{H}^*} \left\|\mathbf{V} - \mathbf{A}^\top\mathbf{H}^*\right\|_{\text{F}}^2}_{\text{reconstruction-error}} = -2\,\mathbf{A}\,\big(\mathbf{V} - \mathbf{A}^\top\mathbf{H}^*\big).$$

Hence the total gradient is

$$\nabla_{\mathbf{H}^*}\mathcal{L} = 2\big[\big(\mathbf{H}^* - \mathbf{H}_{\text{true}}\big) \; - \; \alpha\,\mathbf{A}\,\big(\mathbf{V} - \mathbf{A}^\top\mathbf{H}^*\big)\big].$$

In our experiments, this hybrid loss yielded even lower reconstruction error $\|\mathbf{V} - \mathbf{A}^\top\hat{\mathbf{H}}^*\|_F^2$ and improved the accuracy of the learned network parameter $\mathbf{A}^\top$ which corresponds to the basis matrix.

5.2 Performance Evaluation of the Proposed Algorithms

In this section, we present a series of experiments conducted on both simulated mutational and synthetic noisy data to demonstrate the enhanced performance of our proposed models. Our goal is to show that they not only learn the coefficient matrix $\mathbf{H}_{\text{true}}$ more accurately, but also achieve very low reconstruction error on $\mathbf{V}_{\text{true}}$.

The metric used in our experiments is the normalized squared error (NSE) in dB, defined as:

$$\text{NSE}(\mathbf{X}_{\text{true}}, \mathbf{X}_{\text{pred}}) = 10\log_{10}\left[\frac{\|\mathbf{X}_{\text{true}} - \mathbf{X}_{\text{pred}}\|_{\text{F}}^2}{\|\mathbf{X}_{\text{true}}\|_{\text{F}}^2}\right]. \tag{5.2.1}$$

All experiments employ a 5-fold cross-validation protocol with an 80:20 train–test split in each fold; the results shown in the tables and figures we present are the mean performance over the five splits. We train each model for 800 epochs setting the learning rate to 0.01. In order to do a fair comparison with the iterative MU algorithm, we adopt the logic of [29] and do 800 updates for $\mathbf{W}_{\text{mu}}$ using the ground-truth matrix $\mathbf{H}_{\text{true}}$ followed by 10 updates for $\mathbf{H}_{\text{mu}}$ with the "learned" basis matrix $\mathbf{W}_{\text{mu}}$. We also set the initial value of the regularization parameters λ_1, λ_2 to 1 in all cases. In the tables below, we use bold notation for the smallest error between the models and underline the second lowest error in specific cases. For clarity, Table 5.1 lists the models and their descriptions, and indicates how each model name will appear in the experimental results tables.

Table 5.1: Summary of models and their descriptions.

Model Name	**Description**
DEQ-NMF	Deep Equilibrium NMF Model
DEQ-NMF-A	Deep Equilibrium NMF Model, only A as learnable matrix parameter
A-DEQ-NMF	Approximation scheme
A-DEQ-NMF-A	Approximation scheme, only A as learnable matrix parameter
A-DEQ-NMF-A1	Approximation scheme, only A as learnable matrix parameter + reconstruction term in loss

Experiments with Simulated Mutational Data. First, we conduct experiments over 12 simulated mutational datasets provided in [14], which we previously described in Chapter 4. We benchmark our methods against the classic iterative MU update algorithm and DNMF, measuring accuracy in estimating $\mathbf{H}_{\text{true}}$ and reconstructing $\mathbf{V}_{\text{true}}$ on both the training and test splits. As shown in Table 5.2, A-DEQ-NMF consistently outperforms both MU and DNMF by a wide margin, achieving the lowest NSE on every training split, demonstrating its superior ability to learn the underlying structure of the training data.

Dataset	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
1	-0.38	-21.40	-2.08	-5.15	**-32.98**	-1.46	-1.56
2	-1.16	-21.53	-7.36	-6.80	**-25.10**	-3.21	-3.60
3	-0.20	-22.26	-8.08	-9.30	**-28.54**	-2.02	-1.11
4	-0.94	-25.21	-3.01	-4.37	**-25.28**	-11.57	-11.91
5	-0.26	-21.28	-7.80	-6.00	**-31.69**	-2.43	-1.61
6	-0.49	-22.13	-4.07	-5.52	**-28.87**	-3.07	-3.51
7	-0.23	-26.82	-16.11	-9.63	**-51.87**	-1.92	-1.01
8	-0.24	-32.63	-32.27	-17.17	**-50.17**	-1.97	-2.13
9	-1.15	-28.02	-13.57	-12.28	**-35.75**	-14.65	-13.63
10	-0.81	-29.08	-10.13	-12.11	**-35.84**	-14.54	-15.78
11	-1.15	-28.53	-24.31	-19.69	**-30.92**	-18.64	-23.03
12	-1.14	-32.74	-23.66	-19.54	**-41.84**	-21.77	-24.14

Table 5.2: Mean NSE($\mathbf{H}_{\text{true}}, \mathbf{H}_{\text{pred}}$) over the train splits - Simulated Mutational Data.

Having established that A-DEQ-NMF delivers the lowest training NSE, we next evaluate its ability to generalize to the test sets. In each epoch, we do an evaluation of $\mathbf{H}_{\text{pred}}$ on the test data and present the results after the last epoch in Table 5.3. Once again, we can see that A-DEQ-NMF outperforms both MU and DNMF on every dataset, demonstrating its strong generalization capabilities.

Dataset	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
1	-0.39	-20.28	-2.38	-5.13	**-31.46**	-1.49	-1.53
2	-1.16	-20.56	-7.17	-6.60	**-24.79**	-3.12	-3.56
3	-0.42	-10.85	0.13	-2.60	**-20.57**	-1.71	-1.22
4	-0.96	-20.28	-2.20	-3.20	**-21.84**	-8.87	-9.17
5	-0.25	-20.70	-7.73	-5.51	**-29.73**	-2.38	-1.49
6	-0.50	-21.18	-3.83	-5.58	**-28.55**	-3.09	-3.67
7	-0.23	-26.72	-16.59	-9.95	**-51.74**	-1.91	-1.00
8	-0.24	-32.22	-32.23	-17.32	**-50.09**	-1.99	-2.28
9	-1.15	-26.03	-10.61	-11.08	**-34.40**	-13.62	-13.49
10	-0.83	-23.71	-8.84	-11.50	**-30.24**	-12.89	-14.41
11	-1.17	-27.10	-22.43	-18.10	**-30.49**	-17.27	-22.02
12	-1.15	-29.95	-22.60	-15.97	**-39.89**	-20.62	-23.09

Table 5.3: Mean NSE($\mathbf{H}_{\text{true}}, \mathbf{H}_{\text{pred}}$) over the test splits - Simulated Mutational Data.

Next, we evaluate the reconstruction error between the predicted data matrix and the ground-truth one. For MU, we have $\mathbf{V}_{\text{pred}} = \mathbf{W}_{\text{mu}}\mathbf{H}_{\text{mu}}$, while for the other models we use the learned parameter $\mathbf{A}$, which corresponds to $\mathbf{W}^\top$ and compute $\mathbf{V}_{\text{pred}} = \mathbf{A}^\top\mathbf{H}_{\text{pred}}$. We present the results for the train splits in Table 5.4 and observe that A-DEQ-NMF-A1 achieves the lowest reconstruction error in almost every dataset, verifying our idea of using a reconstruction error term in the loss. The second-best model, A-DEQ-NMF-A, substantially outperforms the alternatives, underscoring that not ignoring the dependency between $\mathbf{W}^\top$ and $\mathbf{W}^\top\mathbf{W}$ leads to more accurate reconstruction of the data matrix.

Dataset	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
1	-1.64	35.88	34.32	-0.92	36.08	-6.13	**-8.44**
2	-2.04	34.48	25.70	-3.38	35.45	-9.25	**-11.95**
3	-1.85	37.44	36.18	-1.83	36.73	-8.52	**-9.37**
4	-2.17	28.39	25.92	-3.34	29.08	-17.98	**-20.61**
5	-1.04	37.32	33.32	-0.90	37.22	-8.16	**-9.48**
6	-1.85	37.22	29.93	-1.46	37.69	-11.91	**-12.22**
7	-0.33	39.23	34.74	-0.17	39.30	-8.27	**-9.77**
8	-0.32	38.33	33.40	-0.30	38.40	**-8.03**	-6.61
9	-3.00	24.04	20.37	-9.79	24.85	-15.81	**-24.47**
10	-4.76	24.86	20.09	-8.46	25.59	-15.45	**-31.39**
11	-2.98	25.54	21.17	-6.27	25.33	-12.64	**-27.35**
12	-2.88	23.95	16.92	-4.90	24.30	-8.99	**-37.13**

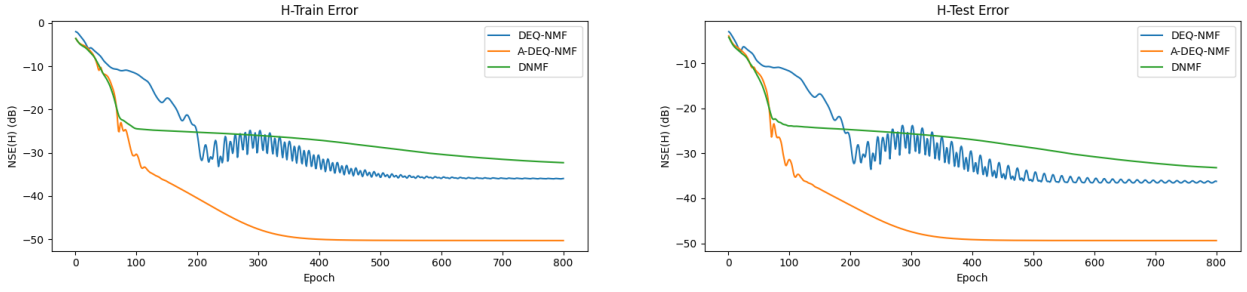
Table 5.4: Mean NSE($\mathbf{V}_{\text{true}}, \mathbf{V}_{\text{pred}}$) over the train splits - Simulated Mutational Data.

In a similar manner, we present the results over the test splits in Table 5.5. As observed on the test splits, A-DEQ-NMF-A1 continues to substantially outperform all other methods across every dataset, with A-DEQ-NMF-A trailing as the second-best model.

Dataset	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
1	-1.64	35.85	34.46	-0.93	36.07	-6.23	**-8.51**
2	-2.04	34.47	25.79	-3.46	35.46	-9.32	**-11.88**
3	-2.10	33.19	32.39	-0.91	34.80	-6.98	**-7.92**
4	-2.20	28.91	23.74	-4.91	30.06	-16.81	**-19.67**
5	-1.06	37.53	33.57	-0.79	37.41	-8.65	**-9.54**
6	-1.85	37.20	29.75	-1.47	37.68	-11.92	**-12.11**
7	-0.33	39.23	34.76	-0.17	39.30	-8.27	**-9.75**
8	-0.32	38.33	33.35	-0.31	38.40	**-7.89**	-6.34
9	-2.97	24.62	20.47	-9.45	25.02	-15.78	**-22.98**
10	-4.72	25.42	20.00	-7.88	26.05	-14.93	**-29.34**
11	-2.97	26.19	22.64	-6.06	25.94	-12.48	**-26.14**
12	-2.88	24.56	17.02	-4.84	24.76	-8.92	**-36.04**

Table 5.5: Mean NSE($\mathbf{V}_{\text{true}}, \mathbf{V}_{\text{pred}}$) over the test splits - Simulated Mutational Data.

We also note, that in the current experiment framework, A-DEQ-NMF not only achieves the lowest NSE error for the coefficient matrices, but also manages to enter a rapid-descent regime quite early in training. This is highlighted in Fig. 5.2.1, where we show the NSE throughout epochs for dataset 8.



(a) (b)

Figure 5.2.1: NSE of $\mathbf{H}$ over epochs for mutational dataset 8.

Experiments with Synthetic Data. To further illustrate the efficacy of our models we also conduct experiments on synthetic data. We construct the matrices $\mathbf{W}_{\text{true}} \in \mathbb{R}^{100\times 10}$ $\mathbf{H}_{\text{true}} \in \mathbb{R}^{10\times 1000}$, whose entries

are drawn independently from a uniform distribution over the interval $[0, 1]$. Then we generate our synthetic data using the following model:

$$\mathbf{V}_{\text{true}} = \mathbf{W}_{\text{true}}\mathbf{H}_{\text{true}} + \mathbf{E}. \tag{5.2.2}$$

To ensure the nonnegativity of the noisy data matrix $\mathbf{V}_{\text{true}}$, the entries of the noise matrix $\mathbf{E}$ are also drawn from a uniform distribution over the interval $[0, c]$, where c is chosen to attain a desired signal-to-noise ratio (SNR). The performance of each model over various SNR ratios is summarized in Table 5.6. We observe that all equilibrium approximation models achieve almost the same very low NSE for small SNR values and the accuracy of A-DEQ-NMF-A and A-DEQ-NMF-A1 improves even more when the noisy data matrix is closer to the ground-truth matrix in higher SNR scenarios.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	-7.24	-6.39	-0.71	-5.46	-8.29	**-8.36**	-6.19
5	-7.57	-6.49	-1.40	-6.45	-9.67	**-10.22**	-9.22
10	-8.07	-6.58	-2.17	-9.07	-11.80	**-13.56**	-12.95
15	-8.49	-6.64	-2.92	-13.09	-14.23	**-18.01**	-17.53
20	-8.82	-6.78	-3.38	-14.91	-15.88	**-22.17**	-21.96

Table 5.6: Mean NSE($\mathbf{H}_{\text{true}}, \mathbf{H}_{\text{pred}}$) over the train splits - Synthetic Data.

We notice in Table 5.7, that the models exhibit consistent behavior on the test sets, confirming their ability to generalize effectively to unseen data.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	-7.00	-6.36	-0.64	-5.43	**-7.55**	-7.26	-6.45
5	-7.44	-6.48	-1.46	-6.31	**-9.16**	-9.12	-9.15
10	-8.02	-6.57	-2.16	-8.92	-11.52	-12.52	**-12.81**
15	-8.47	-6.62	-2.94	-12.90	-14.04	-16.90	**-17.40**
20	-8.81	-6.76	-3.39	-14.84	-15.75	-21.20	**-21.85**

Table 5.7: Mean NSE($\mathbf{H}_{\text{true}}, \mathbf{H}_{\text{pred}}$) over the test splits - Synthetic Data.

In a similar fashion to the reconstruction error benchmarks we got on the simulated mutational datasets, we once again observe in Tables 5.8 and 5.9 that A-DEQ-NMF-A1 shows superior performance, achieving far lower error than its deep network peers.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	-11.66	-0.84	-0.09	-11.01	-0.19	-11.08	**-12.26**
5	-14.35	-1.06	-0.14	-12.47	-0.32	-12.39	**-15.08**
10	-17.37	-1.26	-0.20	-13.31	-0.44	-12.62	**-18.64**
15	-20.05	-1.33	-0.25	-12.83	-0.58	-13.55	**-22.77**
20	-21.97	-1.78	-0.27	-10.38	-0.65	-11.35	**-27.21**

Table 5.8: Mean NSE($\mathbf{V}_{\text{true}}, \mathbf{V}_{\text{pred}}$) over the train splits - Synthetic Data.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	-11.62	-0.84	-0.09	-11.00	-0.19	-11.08	**-12.01**
5	-14.31	-1.06	-0.14	-12.48	-0.32	-12.39	**-14.91**
10	-17.33	-1.26	-0.20	-13.32	-0.44	-12.62	**-18.52**
15	-20.02	-1.33	-0.25	-12.82	-0.58	-13.55	**-22.65**
20	-21.96	-1.78	-0.27	-10.38	-0.65	-11.35	**-27.09**

Table 5.9: Mean NSE($\mathbf{V}_{\text{true}}, \mathbf{V}_{\text{pred}}$) over the test splits - Synthetic Data.

For the synthetic data, we have full knowledge over the ground truth matrix $\mathbf{W}_{\text{true}}$, and we are able to measure the error between it and the network parameter $\mathbf{A}^\top$ for each network. In the case of MU, we use

$\mathbf{W}_{\text{mu}}$ that is learned by doing 800 multiplicative updates. We can see in Table 5.10 that MU achieves the lowest NSE in all cases, since it uses direct iterative updates to learn it. However, we also observe that A-DEQ-NMF-A1 closel trails the lowest error for slightly higher SNR values.

SNR (dB)	MU	DNMF	DEQ-NMF	DEQ-NMF-A	A-DEQ-NMF	A-DEQ-NMF-A	A-DEQ-NMF-A1
0	**-1.98**	-1.23	-0.14	0.89	-0.38	-0.90	1.74
5	**-6.92**	-1.23	-0.17	-2.27	-0.53	-4.10	-3.95
10	**-11.92**	-1.22	-0.19	-5.45	-0.63	-5.76	-10.00
15	**-16.99**	-1.14	-0.22	-9.17	-0.75	-6.66	-15.71
20	**-21.87**	-1.41	-0.24	-8.67	-0.80	-6.55	-19.05

Table 5.10: Mean NSE($\mathbf{W}_{\text{true}}, \mathbf{W}_{\text{pred}}$) - Synthetic Data.

For the sake of completeness, we present in Fig. 5.2.2 the NSE in dB for the coefficient matrix and the data matrix on train and test splits throughout the whole training process for all models.

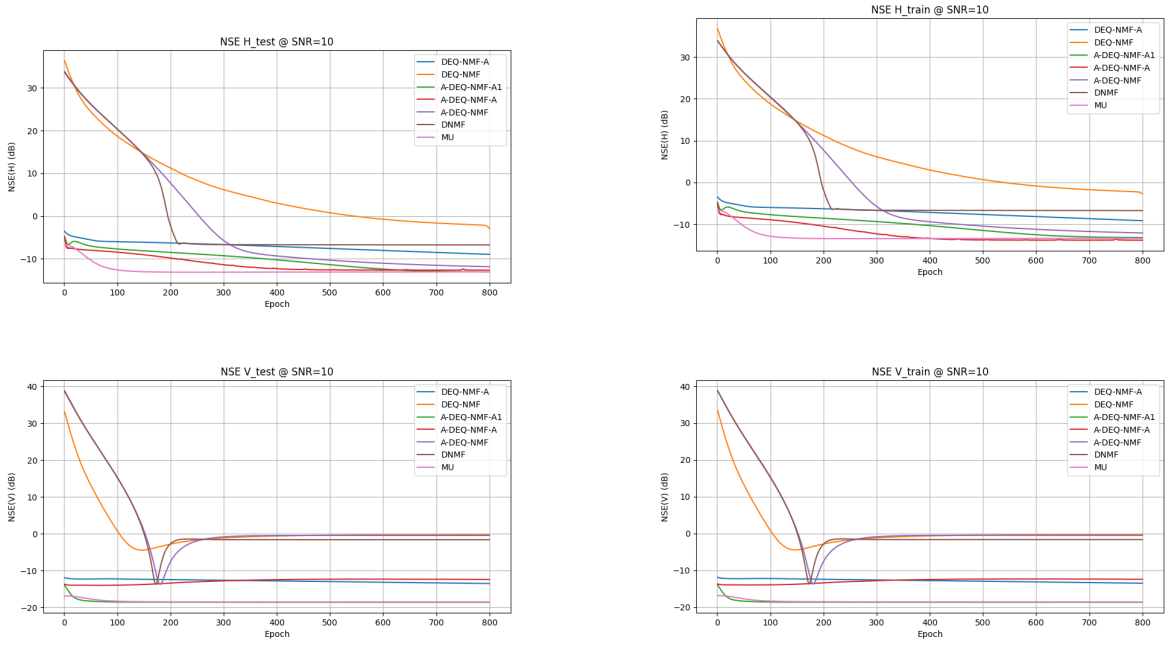


Figure 5.2.2: NSE values throughout training on synthetic data at SNR=10 dB.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we tackled the challenge of implementing the key ideas of DEQs on a deep network that solves the supervised NMF problem. In Chapter 3 we provided a sufficient theoretical background of existing methods in NMF and deep learning, in order to eventually introduce our proposed models and algorithms. Next, in Chapter 4, we first presented the foundational concepts that pave the way for our models and methods, and then gave a detailed description of the simulated data used to evaluate them.

In Chapter 5, we presented a DEQ-based model for solving the supervised NMF problem, that leverages implicit differentiation techniques to maintain memory efficiency. Subsequently, we enhanced this model by altering the structure of our network parameters. We then introduced an approximation scheme, that approximates the equilibrium point in a single step. In a similar manner to the DEQ-NMF model, we enhanced the performance of our equilibrium approximation algorithm by applying some structural changes to its learnable parameters and loss function. Finally, we conducted extensive experiments, in which our proposed models outperform existing approaches showcasing exceptionally low MSE results in both synthetic and simulated data.

6.2 Future Work

In our ongoing research, we aim to further advance both the DEQ-NMF model and the equilibrium approximation scheme to achieve superior performance. Specifically, our future work will focus on the following directions:

- **DEQ-NMF:** In our current implementation, we utilized a basic solver in place of more sophisticated approaches such as Anderson acceleration, primarily due to the practical limitations of available libraries. In the future, we intend to explore and integrate advanced solvers to potentially enhance the convergence and accuracy of the DEQ-NMF framework and we also plan to systematically experiment with fine-tuning both the network and solver parameters.
- **A-DEQ-NMF:** Our results indicate that standard network architectures struggle to accurately learn the basis matrix when treated as a trainable parameter, whereas our proposed modifications offered improvements. However, on mutational datasets, even these modifications did not achieve fully satisfactory estimation. We plan to conduct a thorough investigation into the underlying causes of this limitation and develop targeted modifications to address them.
- **Generalization and Extended Evaluation:** Regarding both the proposed deep equilibrium and equilibrium approximation models, we also aim to develop alternative versions of our architectures tailored for unsupervised NMF, where only the data matrix is available. Moreover, since our proposed methods consistently outperformed existing baselines in our experiments, we plan to validate their effectiveness on a broader range of datasets and applications. This extended evaluation will provide deeper insights into the generalizability and strengths of our approaches.

Chapter 7

Appendix

7.1 Single-column Updates for A-DEQ-NMF: Step-by-Step Derivation

Let $\mathbf{h}^* \in \mathbb{R}^p$ be the predicted vector for a single input $\mathbf{v} \in \mathbb{R}^n$:

$$\begin{aligned}\mathbf{h}^* &= \mathbf{M}^{-1}\mathbf{y} \\ \mathbf{M} &= \mathbf{B} + \lambda_2 \mathbf{I}_p \\ \mathbf{y} &= \mathbf{A}\mathbf{v} - \lambda_1 \mathbf{1}_{p\times 1}\end{aligned}$$

where $\mathbf{I}_p$ is the $p \times p$ identity, $\mathbf{1}_{p\times 1}$ is the p-dimensional vector of ones, $\mathbf{A} \in \mathbb{R}^{p\times n}$, $\mathbf{B} \in \mathbb{R}^{p\times p}$. Let $\mathbf{h}_{\text{true}} \in \mathbb{R}^p$ be the ground truth vector. The loss is $\mathcal{L} = ||\mathbf{h}^* - \mathbf{h}_{\text{true}}||^2$.

7.1.1 Update for a_{jk}

First, compute:

$$\frac{\partial \mathbf{y}}{\partial a_{jk}} = v_k \mathbf{e}_j \implies \frac{\partial \mathbf{h}^*}{\partial a_{jk}} = \mathbf{M}^{-1} v_k \mathbf{e}_j$$

where $\mathbf{e}_j$ is the jth standard basis vector. Gradient of the loss:

$$\frac{\partial \mathcal{L}}{\partial a_{jk}} = 2(\mathbf{h}^* - \mathbf{h}_{\text{true}})^\top (v_k \mathbf{M}^{-1}\mathbf{e}_j)$$

Gradient-descent update:

$$a_{jk}^+ = a_{jk} - \alpha \frac{\partial \mathcal{L}}{\partial a_{jk}}$$

7.1.2 Update for b_{jk}

$\mathbf{M} = \mathbf{B} + \lambda_2 \mathbf{I}_p$ so $\frac{\partial \mathbf{M}}{\partial b_{jk}} = \mathbf{E}_{jk}$, where $\mathbf{E}_{jk}$ is the matrix with 1 at (j, k). Use:

$$\frac{\partial \mathbf{M}^{-1}}{\partial b_{jk}} = -\mathbf{M}^{-1}\mathbf{E}_{jk}\mathbf{M}^{-1}$$

So

$$\frac{\partial \mathbf{h}^*}{\partial b_{jk}} = -\mathbf{M}^{-1}\mathbf{E}_{jk}\mathbf{M}^{-1}\mathbf{y}$$

Gradient:

$$\frac{\partial \mathcal{L}}{\partial b_{jk}} = -2(\mathbf{h}^* - \mathbf{h}_{\text{true}})^\top (\mathbf{M}^{-1}\mathbf{E}_{jk}\mathbf{M}^{-1}\mathbf{y})$$

Update:

$$b_{jk}^+ = b_{jk} + 2\alpha(\mathbf{h}^* - \mathbf{h}_{\text{true}})^\top (\mathbf{M}^{-1}\mathbf{E}_{jk}\mathbf{M}^{-1}\mathbf{y})$$

7.1.3 Update for λ_1

$$\frac{\partial \mathbf{y}}{\partial \lambda_1} = -\mathbf{1}_{p\times 1} \implies \frac{\partial \mathbf{h}^*}{\partial \lambda_1} = -\mathbf{M}^{-1}\mathbf{1}_{p\times 1}$$

So

$$\frac{\partial \mathcal{L}}{\partial \lambda_1} = 2(\mathbf{h}^* - \mathbf{h}_{\text{true}})^\top (-\mathbf{M}^{-1}\mathbf{1}_{p\times 1})$$

Update:

$$\lambda_1^+ = \lambda_1 + 2\alpha(\mathbf{h}^* - \mathbf{h}_{\text{true}})^\top (\mathbf{M}^{-1}\mathbf{1}_{p\times 1})$$

7.1.4 Update for λ_2

$$\frac{\partial \mathbf{M}}{\partial \lambda_2} = \mathbf{I}_p \implies \frac{\partial \mathbf{M}^{-1}}{\partial \lambda_2} = -\mathbf{M}^{-2}$$

Thus,

$$\frac{\partial \mathbf{h}^*}{\partial \lambda_2} = -\mathbf{M}^{-2}\mathbf{y}$$

So,

$$\frac{\partial \mathcal{L}}{\partial \lambda_2} = 2(\mathbf{h}^* - \mathbf{h}_{\text{true}})^\top(-\mathbf{M}^{-2}\mathbf{y})$$

Update:

$$\lambda_2^+ = \lambda_2 + 2\alpha(\mathbf{h}^* - \mathbf{h}_{\text{true}})^\top(\mathbf{M}^{-2}\mathbf{y})$$

7.2 Multi-column (Summation) Updates: Step-by-Step

Suppose we have m columns: $\mathbf{H}^* = [\mathbf{h}_1^* \cdots \mathbf{h}_m^*]$, $\mathbf{H}_{\text{true}} = [\mathbf{h}_{\text{true},1} \cdots \mathbf{h}_{\text{true},m}]$, with corresponding $\mathbf{v}_i$ and $\mathbf{y}_i = \mathbf{A}\mathbf{v}_i - \lambda_1 \mathbf{1}_{p\times 1}$.

The total loss is:

$$\mathcal{L} = \sum_{i=1}^{m} \|\mathbf{h}_i^* - \mathbf{h}_{\text{true},i}\|^2$$

By linearity of differentiation, the parameter updates are:

$$a_{jk}^+ = a_{jk} - 2\alpha \sum_{i=1}^{m} (\mathbf{h}_i^* - \mathbf{h}_{\text{true},i})^\top (v_{k,i}\mathbf{M}^{-1}\mathbf{e}_j)$$

$$b_{jk}^+ = b_{jk} + 2\alpha \sum_{i=1}^{m} (\mathbf{h}_i^* - \mathbf{h}_{\text{true},i})^\top (\mathbf{M}^{-1}\mathbf{E}_{jk}\mathbf{M}^{-1}\mathbf{y}_i)$$

$$\lambda_1^+ = \lambda_1 + 2\alpha \sum_{i=1}^{m} (\mathbf{h}_i^* - \mathbf{h}_{\text{true},i})^\top (\mathbf{M}^{-1}\mathbf{1}_{p\times 1})$$

$$\lambda_2^+ = \lambda_2 + 2\alpha \sum_{i=1}^{m} (\mathbf{h}_i^* - \mathbf{h}_{\text{true},i})^\top (\mathbf{M}^{-2}\mathbf{y}_i)$$

Transition to Compact Matrix Form:

We can rewrite these summations more compactly using matrix multiplication. Let

$$\mathbf{V} = [\mathbf{v}_1 \cdots \mathbf{v}_m] \in \mathbb{R}^{n\times m}, \quad \mathbf{Y} = [\mathbf{y}_1 \cdots \mathbf{y}_m] \in \mathbb{R}^{p\times m}$$

The key observations are:

- The sum over i of terms like $(\mathbf{h}_i^* - \mathbf{h}_{\text{true},i})v_{k,i}$ becomes the matrix product $(\mathbf{H}^* - \mathbf{H}_{\text{true}})\mathbf{V}^\top$.
- The sum of $(\mathbf{h}_i^* - \mathbf{h}_{\text{true},i})$ over all i is $(\mathbf{H}^* - \mathbf{H}_{\text{true}})\mathbf{1}_{m\times 1}$, where $\mathbf{1}_{m\times 1}$ is a column vector of ones.
- The sum of $(\mathbf{h}_i^* - \mathbf{h}_{\text{true},i})^\top(\mathbf{M}^{-2}\mathbf{y}_i)$ over all i is $\text{tr}((\mathbf{H}^* - \mathbf{H}_{\text{true}})^\top\mathbf{M}^{-2}\mathbf{Y})$ by standard trace/cyclic properties.

Compact Matrix Updates:

$$\boxed{\begin{aligned}
\mathbf{A}^{+} &= \mathbf{A} - 2\alpha\, \mathbf{M}^{-1}(\mathbf{H}^{*} - \mathbf{H}_{\text{true}})\, \mathbf{V}^{\top} \\
\mathbf{B}^{+} &= \mathbf{B} + 2\alpha\, \mathbf{M}^{-1}(\mathbf{H}^{*} - \mathbf{H}_{\text{true}})\, (\mathbf{M}^{-1}\mathbf{Y})^{\top} \\
\lambda_1^{+} &= \lambda_1 + 2\alpha\, \mathbf{1}_{1\times p}\, \mathbf{M}^{-1}(\mathbf{H}^{*} - \mathbf{H}_{\text{true}})\, \mathbf{1}_{m\times 1} \\
\lambda_2^{+} &= \lambda_2 + 2\alpha\ \text{tr}\left[(\mathbf{H}^{*} - \mathbf{H}_{\text{true}})^{\top}\, \mathbf{M}^{-2}\mathbf{Y}\right]
\end{aligned}}$$

where $\mathbf{1}_{1\times p}$ is a row vector of ones and $\mathbf{1}_{m\times 1}$ is a column vector of ones.

Bibliography

[1] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, no. 6, pp. 417–441, 1933.

[2] Z. Zhang, "The singular value decomposition, applications and beyond," *CoRR*, vol. abs/1510.08532, 2015. arXiv: 1510.08532. [Online]. Available: http://arxiv.org/abs/1510.08532.

[3] N. Gillis, *Nonnegative Matrix Factorization*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2020. DOI: 10.1137/1.9781611976410. [Online]. Available: https://epubs.siam.org/doi/book/10.1137/1.9781611976410.

[4] X. Fu, K. Huang, N. D. Sidiropoulos, and W.-K. Ma, "Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications," *arXiv preprint arXiv:1803.01257*, 2018.

[5] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[6] C.-J. Lin, "Projected gradient methods for non-negative matrix factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[7] C.-J. Hsieh and I. S. Dhillon, "Fast coordinate descent methods with variable selection for non-negative matrix factorization," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 1064–1072.

[8] A. Cichocki, R. Zdunek, and S.-i. Amari, "Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization," *Independent Component Analysis and Signal Separation*, pp. 169–176, 2007.

[9] D. Lee and H. Seung, "Algorithms for Non-negative Matrix Factorization," in *Advances in Neural Information Processing Systems*, vol. 13, 2001, pp. 556–562.

[10] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[11] N. Gillis, "The why and how of nonnegative matrix factorization," *Computational Statistics & Data Analysis*, vol. 72, pp. 20–40, 2014.

[12] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, 2003, pp. 267–273.

[13] J. C. Nascimento and J. M. P. Dias, "Does independent component analysis play a role in unmixing hyperspectral data?" *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, no. 1, pp. 175–187, 2005.

[14] L. B. Alexandrov, J. Kim, N. J. Haradhvala, *et al.*, "The repertoire of mutational signatures in human cancer," *Nature*, vol. 578, no. 7793, pp. 94–101, Feb. 2020.

[15] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

[16] G. Kutyniok, *Theory and applications of compressed sensing*, 2012. arXiv: 1203.3815 [cs.IT]. [Online]. Available: https://arxiv.org/abs/1203.3815.

[17] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, Mar. 2021.

[18] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K.-.-R. Müller, Eds., vol. 7700, Berlin, Heidelberg: Springer-Verlag, 2012, pp. 9–48.

[19] S. Bai, J. Z. Kolter, and V. Koltun, "Deep Equilibrium Models," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] O. Press and L. Wolf, "Using the output embedding to improve language models," *arXiv preprint arXiv:1608.05859*, 2017.
[21] D. G. Anderson, "Iterative procedures for nonlinear integral equations," *Journal of the ACM*, vol. 12, no. 4, pp. 547–560, 1965.
[22] H. F. Walker and P. Ni, "Anderson acceleration for fixed-point iterations," *SIAM Journal on Numerical Analysis*, vol. 49, no. 4, pp. 1715–1735, 2011. DOI: 10.1137/10078356X.
[23] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of Computation*, vol. 19, no. 92, pp. 577–593, 1965.
[24] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice–Hall, 1983.
[25] P. Deuflhard, *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*. Springer, 2004.
[26] A. L. Dontchev and R. T. Rockafellar, *Implicit Functions and Solution Mappings: A View from Variational Analysis* (Springer Monographs in Mathematics). New York: Springer, 2014, ISBN: 978-1-4939-1036-6. DOI: 10.1007/978-1-4939-1037-3. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4939-1037-3.
[27] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 8024–8035.
[28] JAX Team, *JAX: Composable transformations of Python+NumPy programs*, https://github.com/google/jax, [Online; accessed 05-May-2025], 2018.
[29] R. Nasser, Y. C. Eldar, and R. Sharan, *Deep unfolding for non-negative matrix factorization with application to mutational signature analysis*, 2021. arXiv: 2108.09138 [cs.IT]. [Online]. Available: https://arxiv.org/abs/2108.09138.
[30] J. D. Watson and F. H. C. Crick, "Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid," *Nature*, vol. 171, pp. 737–738, 1953. DOI: 10.1038/171737a0.
[31] S. NikŽainal, D. C. Wedge, L. B. Alexandrov, *et al.*, "Mutational processes molding the genomes of 21 breast cancers," *Cell*, vol. 149, no. 5, pp. 979–993, 2012.
[32] F. Ntoulas-Panagiotopoulos and A. Rontogiannis, "An approximate deep equilibrium scheme for trainable nonnegative matrix factorization," in *Proceedings of the 25th International Conference on Digital Signal Processing (DSP 2025)*, Costa Navarino, Messinia, Greece, 2025. [Online]. Available: https://2025.ic-dsp.org/wp-content/uploads/2025/05/DSP2025-94_Camera_Ready.pdf.
[33] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.
[34] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1957.
[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.
[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[37] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.
[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 3104–3112.
[39] A. Arnx, *First neural network for beginners: Explained with code*, https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf, [Online; accessed 05-May-2025], Aug. 2018.
[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[41] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, 2014.
[42] L. Lessard, B. Recht, and A. Packard, "Analysis and design of optimization algorithms via integral quadratic constraints," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 57–95, 2016.
[43] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[44] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013, pp. 1139–1147.

[45] G. Hinton, *Lecture 6a - overview of mini-batch gradient descent*, https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, Coursera: Neural Networks for Machine Learning, 2012.

[46] IBM, *Overfitting vs. Underfitting*, https://www.ibm.com/think/topics/overfitting-vs-underfitting, [Online; accessed 05-May-2025], 2025.

[47] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 136–145. [Online]. Available: https://proceedings.mlr.press/v70/amos17a.html.

[48] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997. DOI: 10.1016/S0042-6989(97)00169-7.

[49] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004. [Online]. Available: http://jmlr.org/papers/v5/hoyer04a.html.

[50] D. Wang, J.-X. Liu, Y.-L. Gao, J. Yu, C.-H. Zheng, and Y. Xu, "An nmf-$\ell_{2,1}$-norm constraint method for characteristic gene selection," *PLoS ONE*, vol. 11, no. 7, e0158494, 2016. DOI: 10.1371/journal.pone.0158494. [Online]. Available: https://doi.org/10.1371/journal.pone.0158494.

[51] C. H. Q. Ding, D. Zhou, X. He, and H. Zha, "R1-pca: Rotational invariant ℓ_1-norm principal component analysis for robust subspace factorization," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, ACM, 2006, pp. 281–288.

[52] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 1995. DOI: 10.1137/1.9781611971217.

[53] R. Bro and S. de Jong, "A fast non-negativity-constrained least squares algorithm," *Journal of Chemometrics*, vol. 11, no. 5, pp. 393–401, 1997. DOI: 10.1002/(SICI)1099-128X(199709/10)11:5<393::AID-CEM483>3.0.CO;2-L.