Senior Thesis in Mathematics

Data Representation with Low Rank, Real Valued Matrices

Author:
Ziv Epstein
ziv.epstein@pomona.edu

Advisor:
Dr. Blake Hunter

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

December 7, 2016

Contents

Abstract

foo bar

Chapter 1

Introduction

human perception [8, 13, 18, 10, 16]

Chapter 2

Optimization

Chapter 3

Techniques

3.1 Singular Value Decomposition

The first technique to represent a matrix X is to factorize it using singular value decomposition.

Theorem 1. Suppose $X \in \mathbb{R}^{m\times n}$, then there exists a factorization, called the singular value decomposition of X, of the form

$$X = U\Sigma V^T$$

where $U \in \mathbb{R}^{m\times m}$ is unitary, $\Sigma \in \mathbb{R}^{m\times n}$ with non-negative real numbers on the diagonal and $V^T \in \mathbb{R}^{n\times n}$ is unitary.

The diagonal entries σ_i of Σ are called the **singular values** of X.

Proof. (Bast) Observe that $X^TX \in \mathbb{R}^{n\times n}$ is symmetric. Thus, there exist n eigenvectors $v_1, \cdots, v_r$ with corresponding eigenvalues $\lambda_1, \cdots, \lambda_r$ that are pairwise orthogonal (i.e. $v_i^T v_j = 0\ \forall i \neq j$) with $r = \text{rank}(X^TX) = \text{rank}(X)$. Furthermore, we have that $X^TX = VDV^T$, where $V = [v_1, \cdots, v_n] \in \mathbb{R}^{n\times n}$ and

$$d = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}.$$

Since v_i is an eigenvector for X^TX, observe that $XX^TXv_i = X\lambda_i v_i = \lambda_i Xv_i$. Which is to say that if v_i is an eigenvector for X^TX, then Xv_i is an eigenvector for XX^T with the same eigenvalue λ_i. Now we can compute the squared norm of XV_i as follows

$$||XV_i||^2 = (Xv_i)^T(Xv_i) = v_i^TX^TXv_i = v_i^T\lambda_i v_i = \lambda_i v_i^T v_i$$

So the norm of $XV_i = \sqrt{\lambda_i} = \sigma_i$. Now let $u_i = Xv_i\sigma_i$. Then we have

$$u_i^TXv_j = (Xv_i/\sigma_i)^TXv_j = \frac{v_i^TX^TXv_j}{\sigma_i} = v_i^Tv_j \times \frac{\lambda_i}{\sigma_i}$$

which is to say $u_i^T X v_j = 0$ if $i \neq j$ and $u_i^T X v_j = \sigma_i$ if $i = j$. Now writing these equations for $i = 1, \cdots, r$ yields

$$\begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_r^T \end{bmatrix} \cdot X \cdot [v_1 \cdots v_r] = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_r \end{bmatrix}$$

Since $u_i \in \mathbb{R}^m$, the lefthand matrix we have is a $r \times m$ matrix. Thus we generate u_i for $i = r+1, \cdots, m$ to expand the matrix to be $m \times m$ by arbitrarily picking unit vectors that are pairwise orthogonal and orthogonal to $u_1, \cdots, u_r$. Similarly for the v_i's, we generate v_i for $i = r+1, \cdots, n$ to expand the matrix to be $n \times n$ by arbitrarily picking unit vectors that are pairwise orthogonal and orthogonal to $v_1, \cdots, v_r$. This extension gives us in matrix form

$$\begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_m^T \end{bmatrix} \cdot X \cdot [v_1 \cdots v_n] = \Sigma$$

Now let $U = [u_1 u_2 \cdots u_m]$ and $V = [v_1 v_2 \cdots v_n]$ to write the above equation in a more clear way: $U^T X V = \Sigma$. Since U and V are unitary, we have $V^T = V^{-1}$ and $U^T = U^{-1}$. Thus we can solve the equation for X:

$$X = U \Sigma V^T$$

□

An extension of the SVD can be used to create a low-rank approximation for the data. For a low rank $r < \min(m, n)$, we can find an optimal approximation using the following theorem.

Theorem 2 (Eckart-Young)**.** Suppose $X \in \mathbb{R}^{m \times n}$ has a singular value decomposition of $X = U \Sigma V^*$ with

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ are the singular values of A and where $U \in \mathbb{R}^{m \times m}$ and $V^* \in \mathbb{R}^{n \times n}$ are unitary. Then the matrix $X_r = U \Sigma_r V^*$, where

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & & 0 \\ 0 & 0 & \cdots & \sigma_r & \cdots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 \end{bmatrix}$$

is a global minimizer of the problem

$$\min_{\hat{X}, rank(\hat{X}) \leq r} \frac{1}{2}||X - \hat{X}||_f^2$$

and its error is

$$\frac{1}{2}||X - \hat{X}||_f^2 = \frac{1}{2}\sum_{i=r+1}^{n} \sigma_i$$

Proof. content... □

As we saw in Section 2, optimization can be algorithmically challenging, and is often solved using local techniques, such as gradient descent. The fact that the Eckart-Young low-rank approximation guarantees a global minimum is a very nice result.

3.2 Non negative matrix factorization

Non-negative matrix factorization is a popular method used to represent the original with a low-rank approximation. NMF was first employed by Paatero and Tapper [12] but was made popular by Lee and Seung [8].

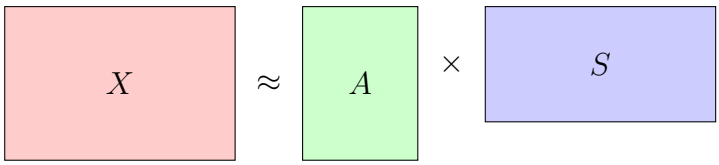

Figure 3.1: A visual representation of the non-negative matrix factorization

The problem of finding such a factorization can be formulated as finding a non-negative A and S that minimize the error

$$F = ||X - AS||^2 \tag{3.1}$$

While this optimization problem is not convex in both A and S, it is convex in one of them. So for a given, fixed S, we can find the optimal A by setting the gradient equal to zero. Since $||X - AS||^2 = \langle X - AS, X - AS\rangle = X^TX - 2X^TAS + (AS)^T(AS)$ we have

$$\frac{\partial F}{\partial A}(X^TX - 2X^TAS + (AS)^T(AS)) = 0$$
$$\text{implies } S^TAS = 2X^TS$$

which is to say $\frac{X^TS}{S^TAS} = 1$ at the optimal A. This equality gives us the below multiplicative

update algorithm.

Input: k=0; Initialize A^0, S^0

repeat

$$A^{k+1} = A^k \circ \frac{XS^k}{A^k(S^k)^T A^k}$$

$$S^{k+1} = S^k \circ \frac{XA^{k+1}}{S^k(A^{k+1})^T A^{k+1}}$$

$$k = k + 1$$

until *Stopping condition*;

Algorithm 1: Multiplicative Update

This optimization scheme naturally leads to a convex optimization function, so the above algorithm can simply be iteratively applied (until for given T we have $k > T$ or for a given $\epsilon > 0$ we have $||X - AS||^2 \leq \epsilon$).

Theorem 3. The Euclidean distance $||X - AS||^2$ is non-increasing under the updating rules of Algorithm 1.

3.3 Neural Network Representation

A data matrix $X \in \mathbb{R}^{m \times n}$ can also be considered as vectorized input to a neural network. Since there has been a growing interest in the representations that neural networks produce, we will explore some examples here. A **neural network** is a network of **neurons**, which

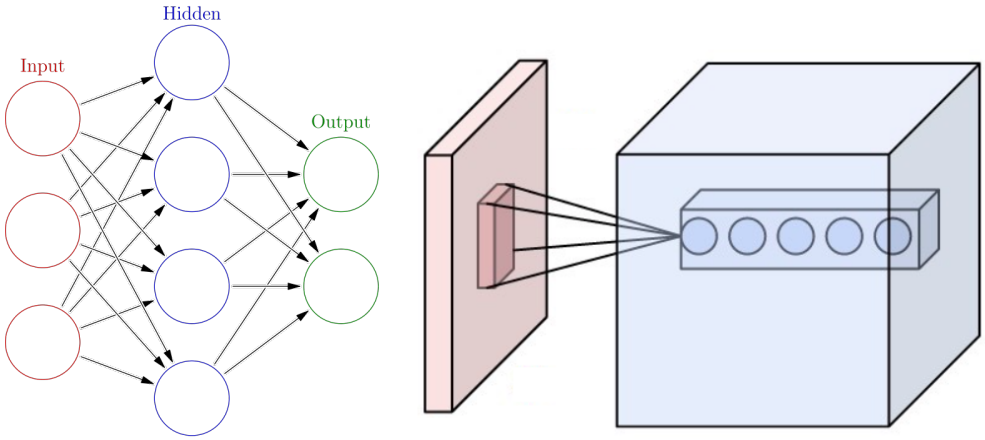

Figure 3.2: Left: a very simple neural network with 3-dimensional input (red), one hidden layer with 4 hidden units (blue), and 2-dimensional output (green). Right: A pictorial representation of a convolutional layer of a neural network.

represent units of computation organized in **layers**. (see Figure 2). For example, say we have input 3-dimensional input $\vec{x} = [1, 2, 3]$. Each of those values would be passed to the input neurons in Figure 1. Then, the first set of black arrows moves the input layer to the hidden

layers by multiplying them by some weights and summin]g them together. This is equivalent to multiplying the 3×1 vector $\vec{x}$ by a 5×3 matrix of weights, W. Then a nonlinear function f is applied elementwise to the 5 dimensional vector $W\vec{x}$. This nonlinear function increases the complexity of the space the neural network can explore, and is usually $f(x) = \tanh(x)$. This process is repeated with the output layer to achieve a 2×1 output vector. In total, the computational process of the neural network can be captured by the equation

$$\vec{output} = f(W^2(f(W^1(\vec{input}))))$$

where $\vec{input}$ is the 3×1 input data, W^1 is the 5×3 weight matrix for the first layer, W^2 is the 2×5 weight matrix for the second layer, f is an elementwise nonlinear function, and $\vec{output}$ is the 2×1 output vector. The **activation of a layer**, F^ℓ is the intermediary result of running a given input through the network. For example, the activation of the first layer F^1 is

$$F^1 = f(W^1(\vec{input}))$$

A **convolutional neural network** is a neural network is at least one convolutional layer. This neural architecture is inspired by the organization of the human visual cortex, which small regions of neurons are aggregated into a single value. A convolutional layer is a layer that takes a small window of the input data and computes a single value for that region. For example, a **max pooling** convolutional layer simply computes the maximum value found for a given window (see Figure 2).

Through many layers and complex nonlinearities, a neural network with only a single hidden layer can approximate any function [**?**] with sufficient weights. But how are these weight matrices W^i constructed? This requires **training** the neural network to configure these weight matrices. In general, this is achieved by feeding it labeled data, (x, y) (with $x_i \in \mathbb{R}^3$ and $y_i \in \mathbb{R}^2$ for our example) and initializing the weights as random. Then, for each observation x_i, a prediction $\hat{y}$ is computed by running x_i through the network:

$$\hat{y} = f(W^2(f(W^1(x_i))))$$

Then the error in the network can be computed as the difference between the prediction and the actual output:

$$E(x_i) = (\hat{y} - \hat{y})^2$$

The weights can then be updated by descending the gradient of the error using **backpropagation**. A review of the backpropagation algorithm is beyond the scope of this paper, but for a full overview, see [14].

For the purposes of image processing, the input data is raw images $x \in \mathbb{R}^{1000 \times 1000 \times 3}$ assuming they are 1000 pixels by 1000 pixels and are colored. The output y used to trained the network is often labels corresponding to what the image is of, such as:

$$y \in \{\text{elephant}, \text{doughut}, \cdots, \text{child}, \text{apple}\}$$

A well documented phenomenon of applying DCNNs to image processing is that lower level layers correspond to pixel or edge level information, and as you look at higher level layers, you get more high order structure. As information passes from the input image pixels to

the output classifier, the "scale" of that projection increases. For example, layers towards the input correspond to local pixelated regions, edges and neighboring contours. Layers towards the output correspond to global structure, groups of aggregated edges, etc. The work presented in this paper takes advantage of this important fact. We will refer to this phenomenon as the visual hierarchy phenomenon (VHP).

Chapter 4

Types of Data

4.1 Image

4.2 Sound

4.3 Graph

4.4 Natural Language

With the vast amount of digital text being generated across the internet, methods for understanding and processing corpora of human language become necessary. Across mathematics and computer science, many techniques have been put forward that allow one to understand a body of text far too large to read herself. A successful method in this domain is *topic modelling*, whereby semantically cohesive subgroups of words can be identified. In particular, let $\mathcal{C} = \{d_1, d_2, \cdots, d_n\}$ be a collection of documents with a vocabulary $\mathcal{V}$. A *topic* t_i is a vector over the words in the vocabulary that represents a coherent high level notion in the corpus:

$$t_i = \{v_1^i, v_2^i, \cdots, v_m^i\}$$

where m is the size of the vocabulary. Topic modelling offers a powerful tool for understanding large amounts of text because they can discover latent semantic structure within text.

There are two primary techniques for learning these topics t_i. The first is LDA, a generative Bayesian statistical model which views each document d_j as a mixture of various topics. The second is non-negative matrix factorization, which aims to factor the document/word matrix into a document/topic and a topic/word matrix [8]. The focus of this thesis will be NNMF, because of its relation to linear algebra, and its deep visual and conceptual intuition.

Given our n documents with vocabulary $\mathcal{V}$ of size m, we construct a matrix $X \in \mathbb{R}^{n \times m}$ where $X_{i,j}$ is the number of occurences of word j in document i. For a given inner dimension k, we seek to factor X into two matrices A and S such that

$$X \approx AS$$

where $A \in \mathbb{R}^{n \times k}$ is the document/topic matrix and $S \in \mathbb{R}^{k \times m}$ is the topic/word matrix. When we impose that A and S must be non-negative, a strong intuition emerges. In particular, the (i, j)th entry of A corresponds to the proportion of topic j in document i and the (i, j)th entry of S corresponds to the relevance of word j in topic i.

Chapter 5

Hierarchical Representation

Many times, the data we wish to represent exhibits hierarchical structure. One such example is with the topics found in text data. Since topics correspond to semantic ideas, it makes sense that they would be organized in hierarchicaly. For example, in the 20 News Group data set, there are two separate topics for the Cincinnati Reds and Toronto Blue Jays, two baseball teams. Thus one could think about a supertopic, baseball, that branches into the two topics the standard NNMF finds.

5.1 Approach

For a given document matrix V, we use the python library `scikitlearn` to decompose V into document/topic matrix W and topic/word matrix H such that

$$V \approx WH.$$

The `scikitlearn` implementation uses alternating gradient descent with the following objective function to generate optimal guesses for W and H.

$$c(H, W) = \tfrac{1}{2}||X - WH||_{fro}^2 + \alpha\lambda||W||_1 + \alpha\lambda||H||_1 + \tfrac{1}{2}\alpha(1-\lambda)||W||_{fro}^2 + \tfrac{1}{2}\alpha(1-\lambda)||H||_{fro}^2$$

where $||\cdot||_{fro}$ is the Frobenius norm, $||\cdot||_1$ is the L1 norm, λ is the L1 ratio and α is a free parameter.

From the N topics t_n for $n \in \{1 \cdots N\}$[1], we populate an adjacency matrix A where

$$A_{i,j} = \frac{T_i \cdot T_j}{||T_i||\ ||T_j||}$$

is the cosine similarity between topics i and j. We then define a *threshold vector* σ by sorting all the elements of A.

$$\sigma = \{\sigma_1, \sigma_2, \cdots \sigma_{N^2} \mid 0 \leq \sigma_i \leq \sigma_j \leq 1 \forall i \leq j \text{ and } \sigma_k \in A\}$$

[1] observe that t_n is simply the nth row of H

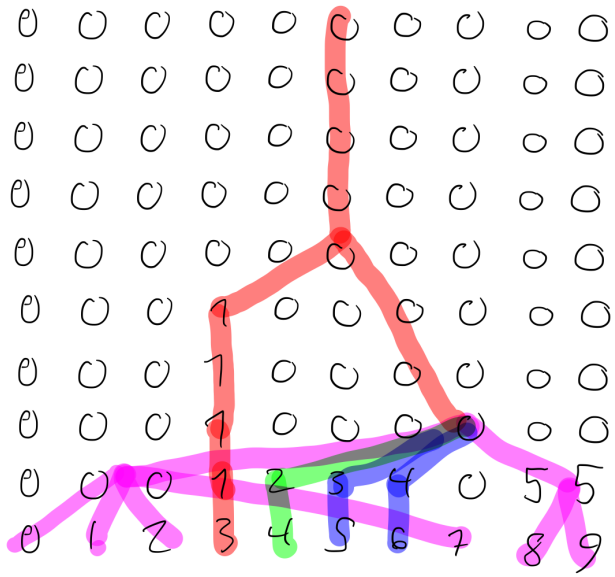

Figure 5.1: How the tree structure is formed for the connected componenet vectors

We then create an array of graphs $A^{(k)}$ thresholded using the values of σ, such that

$$A^{(k)}_{i,j} = \begin{cases} 1 & \text{if } A_{i,j} > \sigma_k \\ 0 & \text{otherwise.} \end{cases}$$

Observe that $A^{(1)}$ is the fully connected graph and $A^{(N^2)}$ is the completely disconnnected graph. By looking at the connected components of a given graph,

$$c(A^{(j)}) = \{c^j_1, c^j_2, \cdots, c^j_i, \cdots, c^j_N\}$$

where $c_i = k$ means that the ith vertex is in the kth order component, we can formulate a tree structure (see Figure 1). For example, say $N = 8$ and we have

$$c(A^{(j)}) = \{0, 0, 0, 0, 1, 1, 1, 1\}$$
$$c(A^{(j+1)}) = \{0, 0, 0, 0, 1, 1, 2, 2\}$$

This means that $A^{(j)}$ has two connected components, ordered 0 (with vertices 1,2,3,4) and 1 (with vertices 5,6,7,8) and that $A^{(j+1)}$ has three connected components, ordered 0 (with vertices 1,2,3,4), 1 (with vertices 5 and 6) and 2 (with vertices 7 and 8). Thus there is a branch from the connect component 1 in $A^{(j)}$ to the connected componenets 1 and 2 in $A^{(j+1)}$. By greedily repeating this iterative algorithm starting with $A^{(1)}$ [2] as the root, we produce the tree of topics. Observe that at this stage, all the leaf nodes correspond to actual topics t_n. We formulate the topic vectors for the parent nodes by additive percolating up the tree. That is, for a given parent topic τ with children $\tau_1, \cdots, \tau_k$ we simply have

$$\tau = \sum_i \tau_i$$

[2] which has by definition only a single connected component and so $c(A^{(1)}) = \{0, \cdots, 0\}$

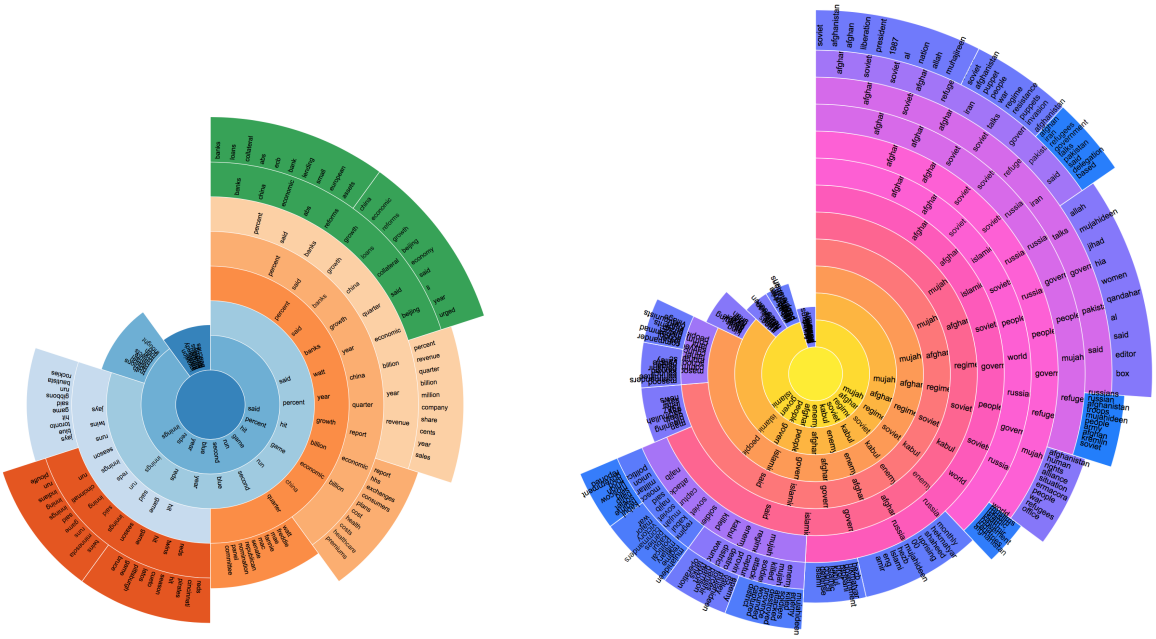

Figure 5.2: Left: Visualization of hierarchical topics in standard NMF; Right: Visualization of hierarchical topics in SSNMF;

5.2 Visualization

I use the d3.js Sunburst implementation to visualize the hierarchical topic model. Arcs at the same level represent discrete topics. A topic on an inner layer that encompasses multiple outer topics represent a super-topic. For example, in Figure 3, the two outer green topics that represent European banking and Chinese banking representatively (with top words

{banks, loans, collateral, abs, ecb bank, lending, small, european, asset}

and

{china, economic, reforms, growth, beijing, economy, said, li, year, urged})

merge into the inner super topic with top words

{banks, china, economic, abs, reforms, growth, loans, collateral, said, beijing}.

The visualization is responsive, dynamic and available at
`http://www1.cmc.edu/pages/faculty/BHunter/ziv.html`

Next, I extended this visualization to the Semi-Supervised domain using the Afghan Dataset. Here each document has associated with it a class $C \in \{1, \cdots, k\}$ which in this case $k = 3$. Recall that the matrix $B \in \mathbb{R}^{k \times r}$ in the semi-supervised NMF model is multiplied by S to obtain an approximation for Y, the label matrix. Thus the i, jth entry of B can be interpreted in the weighted importance of topic i in predicting class j. Thus I sum over the columns of B to capture how important topic i is predicting classes in general. After normalizing, we get a color value c_i for topic i, such that

$$c_i = \sum_j B_{i,j} / \sum_{i,j} B_{i,j}$$

where $c_i = 1$ corresponds to yellow and $c_i = 0$ corresponds to blue (see Figure 3 right)

Chapter 6

Semi-Supervised NNMF

The methods above are unsupervised, which means the algorithms are designed to reveal latent structure within the unlabeled data. Many times however, data is accompanied with labels and the new algorithmic objective is to assign new unobserved data into appropriate labels. In the semi-supervised case, when only some of the data is labeled, the objective is similar to the unsupervised case. However, the known class labels can be incorporated into the model to improve classification performance [9]. In this section, we will explore an extension of NNMF where some of the class labels are known, and this information is used to enhance the performance of the NNMF. This *semi-supervised* NNMF learns a one-versus all separating hyperplane for the observations [9].

6.1 Approach

In many cases, in addition to $X \in \mathbb{R}^{n \times m}$ we also have a label matrix $Y \in \mathbb{R}^{n \times k}$, where k is the number of classes and $Y_{i,j}$ is 1 if document i is in class j and 0 otherwise. Given $B \in \mathbb{R}^{k \times r}$, a basis matrix for Y, and $L \in \mathbb{R}^{k \times n}$, a weight matrix to handle missing labels, then the energy function for SSNMF is as follows

$$E - ||(X - AS)||^2 + \lambda ||L \circ (Y - BS)||^2$$

where λ is a tradeoff parameter that governs the importance of the supervised term.

In the same vein of Algorithm 1, this energy function yields a convex optimization prob-

lem solved by the following algorithm.

Input: k=0; Initialize A^0, S^0, B^0

repeat

$$A^{k+1} = A^k \circ \frac{XS^k}{A^k(S^k)^T S^k}$$

$$B^{k+1} = B^k \circ \frac{(L \circ Y)S^k}{(L \circ B(S^k)^T)S^k}$$

$$S^{k+1} = S^k \circ \frac{(A^{k+1})^T X + \lambda B^T (L \circ Y)}{(A^{k+1})^T AS + \lambda B^T (L \circ BS)}$$

$$k = k + 1$$

until *Stopping condition*;

Algorithm 2: Multiplicative Update for Semi-Supervised NNMF

6.2 Visualization

compare different models over classes

Chapter 7

Deep Models

7.1 Approach

Semi-Supervised NNMF as discussed above can be thought of as representing A in a low dimensional representation as S. In this framework, A is the function that maps the low dimensional representation to the original high dimensional representation. However, as the data becomes increasingly complex, it may have many hierarchy of attributes, each of which requires its own mappings. With the motivation in mind, Trigeorgis et al put forward the notion of a Demi-Semi NNMF [15].

$$X \approx A_1 A_2 \cdots A_m S_m$$

This representation of the data can be achieved by recursively factorizing the low-dimensional representation at each level [15].

$$\begin{aligned} S_{m-1} &= A_m S_m \\ &\vdots \\ S_2 &\approx A_3 \cdots A_m S_m \\ S_1 &\approx A_2 \cdots A_m S_m \end{aligned}$$

We then consider the algorithms and notions in the domain of Deep Semi NMF as a model for Hierarchical NMF [2, 15]

7.2 Deep Neural Models

7.3 Visualization

Chapter 8

Discussion

Bibliography

[1] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[2] Jennifer Flenner and Blake Hunter. A deep nonnegative matrix factorization.

[3] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[4] DMBTL Griffiths and MIJJB Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:17, 2004.

[5] Ngoc-Diep Ho. *Nonnegative matrix factorization algorithms and applications*. PhD thesis, ÉCOLE POLYTECHNIQUE, 2008.

[6] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[7] Daniel D Lee and H Sebastian Seung. Unsupervised learning by convex and conic coding. *Advances in neural information processing systems*, pages 515–521, 1997.

[8] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[9] Hyekyoung Lee, Jiho Yoo, and Seungjin Choi. Semi-supervised nonnegative matrix factorization. *IEEE Signal Processing Letters*, 17(1):4–7, 2010.

[10] Nikos K Logothetis and David L Sheinberg. Visual object recognition. *Annual review of neuroscience*, 19(1):577–621, 1996.

[11] Marvin Minsky. A framework for representing knowledge. 1975.

[12] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[13] Stephen E Palmer. Hierarchical structure in perceptual representation. *Cognitive psychology*, 9(4):441–474, 1977.

[14] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[15] George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Bjoern Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pages 1692–1700, 2014.

[16] Shimon Ullman et al. *High-level vision: Object recognition and visual cognition*, volume 2. MIT press Cambridge, MA, 1996.

[17] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[18] E Wachsmuth, MW Oram, and DI Perrett. Recognition of objects and their component parts: responses of single units in the temporal cortex of the macaque. *Cerebral Cortex*, 4(5):509–522, 1994.