CSCI 1952Q Coding 2

Introduction

This paper describes my approach to Nonnegative Matrix Factorization (NMF) for topic modeling. In particular, given a word-word correlation matrix M, where $M_{i,j}$ is proportional to the empirical probability that word i and word j co-occur in a document, our goal is to find nonnegative matrices A and W such that $M \approx AW$.

Initialization

Firstly, we initialize $A \in \mathbb{R}^{m \times r}$ and $W \in \mathbb{R}^{r \times m}$ using Singular Value Decomposition (SVD) on $M \in \mathbb{R}^{m \times m}$, expressed as $M = UDV^T$. Specifically, we initialize A as $UD^{\frac{1}{2}}$, and W as $D^{\frac{1}{2}}V^T$

Model Description

My model alternates between fixing A to optimize W and the opposite each iteration. Additionally, I set the negative values in A or W to 0 each iteration in case the optimization step created negative values. Together these steps insure that I find a factorization of M, and that it is nonnegative.

Objective Function

The objective function used in my model is defined as:

$$f(X, Y) = \sum (M - AW)^2 \tag{1}$$

where M and AW represent the matrices of actual and estimated word-word correlations. This function is derived from the loss function (the squared Frobenius norm of M-AW). While I played around with both L1 and L2 regularization, neither did much to reduce loss.

Training Procedure

The model is trained over 2000 steps. At each step, I compute the gradient $\nabla f(A, W)$ and update A or W accordingly. Rather than using a standard stochastic gradient descent (SGD) method, my updates are based on the Adam optimizer. I also found the Adagrad optimizer to be quite good for this task, although slightly worse than Adam.

Findings

I initially ran into trouble trying to set the negative values of A or W to 0 during my optimization process, so a sort of hacky solution I came up with was to disincentivize negative values by adding their squares to the loss function. This worked fairly well, and I was able to get my loss down to 86.5.

Additionally, I found that with alternating minimization, the order you choose to optimize in seems to affect what the best learning rate is, but not ultimately the loss after adjusting the learning rate.

Performance

Ultimately I was able to get to a loss of 82.75 using Adam, and with Adagrad I was able to get 83.56.