Homework 5:
Low rank approximation

Problem 1: Image compression

Download the image `jinx.png` from the website. It is a 563×1000 greyscale image which you can think of as a 563×1000 matrix M of integer pixel values between 0 and 255.

(a) [6 points] Run SVD and recover the rank-k approximation of M, for $k \in \{1, 2, 5, 10, 20, 50, 75, 100, 563\}$. You should find that the resulting matrices cannot always be immediately interpreted as images themselves. Explain why this is the case, and perform some simple postprocessing to obtain pictures from these low rank approximations. In your assignment, include your recovered image for $k = 1, 10, 50, 75$.

(b) [3 points] Why do we stop at $k = 563$?

(c) [3 points] Can you propose an interpretation for the first (left and right) singular vectors?

(d) [3 points] How much memory is required to efficiently store the rank k approximation? Give an expression for all k. Assume that each floating point number takes one unit of memory. How much better is this than naively saving the image as a matrix of pixel values?

(e) [3 points] What are some kinds of artifacts do you notice on the compressed images, especially for small values of k? Can you give a reasonable explanation for why you might expect to see these types of artifacts?

Problem 2: Word embeddings

A word embedding is a mapping from words to vector representations of the words. Ideally, the geometry of the vectors will capture the semantic and syntactic meaning of the words. For example, words similar in meaning should have representations that are close to each other in the vector space. A good word embedding provides a means for mapping text into vectors, from which one can then apply all the usual learning algorithms that take, as input, a set of vectors.

Word embeddings have taken natural language processing (NLP) by storm in the last few years and have become the backbone for numerous NLP tasks such as question answering and machine translation. There are neural-network approaches to learning word embeddings, but in this question, we will study a simple SVD-based scheme that does a surprisingly good job at learning word embeddings.

Your input data is a word co-occurrence matrix M of the 10,000 most frequent words from a Wikipedia corpus with 1.5 billion words. Entry M_{ij} of the matrix denotes the number of times in the corpus that the i-th and j-th words occur within 5 words of each other. The file `co_occur.csv.gz` contains the symmetric co-occurrence matrix M. The file `dictionary.txt` contains the dictionary for interpreting this matrix: The i-th row of the dictionary is the word corresponding to the i-th row and column of M. The dictionary is sorted according to the word frequencies. Therefore, the first word in the dictionary `the` is the most common word in the corpus, and the first row and column of M contain the co-occurrence counts of `the` with every other word in the dictionary.

Before you start, make sure you can import the dataset and do a few trials to ensure you can interpret the entries of M using the dictionary.

(a) [3 points] For the rest of the problem, let us use $\hat{M}$ to denote the normalized matrix defined by

$$\hat{M}_{ij} = \ln(1 + M_{ij})$$

Compute the rank-100 approximation to $\hat{M}$ according to the singular value decomposition:

$$\hat{M} = UDV^T$$

and plot the top 100 singular values of $\hat{M}$. Does $\hat{M}$ look close to being low rank?

Note: Computing the full SVD will be computationally intensive; it is better to compute the top 100 singular values and singular vectors and keep those around as you will be using them later.

(b) [5 points] Since $\hat{M}$ is symmetric, the left and right singular vectors are the same, up to flipping signs of some columns. You will now interpret the singular vectors (columns of U or V). For any i, denote by v_i the singular vector corresponding to the i-th largest singular value. The coordinates of this vector correspond to the 10,000 words in our dictionary.

Find six interesting/interpretable singular vectors, and describe what semantic or syntactic structures they capture. For each of the six vectors you choose, provide a list of the 10 words corresponding to the coordinates with the largest values and the 10 words corresponding to the coordinates with the smallest values. Not all singular vectors have easy-to-interpret semantics; why would you expect this to be the case?

(c) [5 points] Normalize the rows of U so that each one has unit ℓ_2 norm. The i-th row now represents an embedding of the i-th word into $\mathbb{R}^{100}$. We will explore a curious property of these word embeddings: certain directions in the embedded space correspond to specific syntactic or semantic concepts.

Let u_1 be the word embedding for `woman` and let u_2 be the word embedding for `man`. Define the vector:

$$u = u_1 - u_2$$

Project the embeddings of the following words onto u: `boy, girl, brother, sister, king, queen, he, she, john, mary, all, tree`. Present a plot of these projections and discuss the results.

(d) [5 points] Present a similar plot for the words `math, matrix, history, nurse, doctor, pilot, teacher, engineer, science, arts, literature, bob, alice`. Discuss the implications of these projections. Why might this be problematic? Consider how word embeddings might impact applications such as job candidate searches on LinkedIn.

(e) [14 points] Now you will explore in more depth the property that directions in the embedded space correspond to semantic or syntactic concepts.

1. [4 points] Define the similarity between two words i and j as the cosine similarity of their embeddings:

   $$\text{similarity}(w_i, w_j) = \langle w_i, w_j \rangle$$

   First, explain why for these vectors, this coincides with the usual notion of cosine similarity we've seen in the past. Then, using this definition, find the most similar words to `washington`.

2. [10 points] Word embeddings can solve analogy tasks, such as:

   boat is to plane as captain is to _

   by finding the word whose embedding is closest to the vector:

   $$w_{\text{plane}} - w_{\text{boat}} + w_{\text{captain}}$$

   Using the dataset in `analogy_task.txt`, compute the accuracy of the word embedding approach for solving analogies. Discuss the results and identify difficult analogies for this approach.