CSE 422 HW5

Rohan Mukherjee

February 11, 2025

1. **Image Compression**

(a) Here is the picture of the rank-k approximation of the image for different ks:

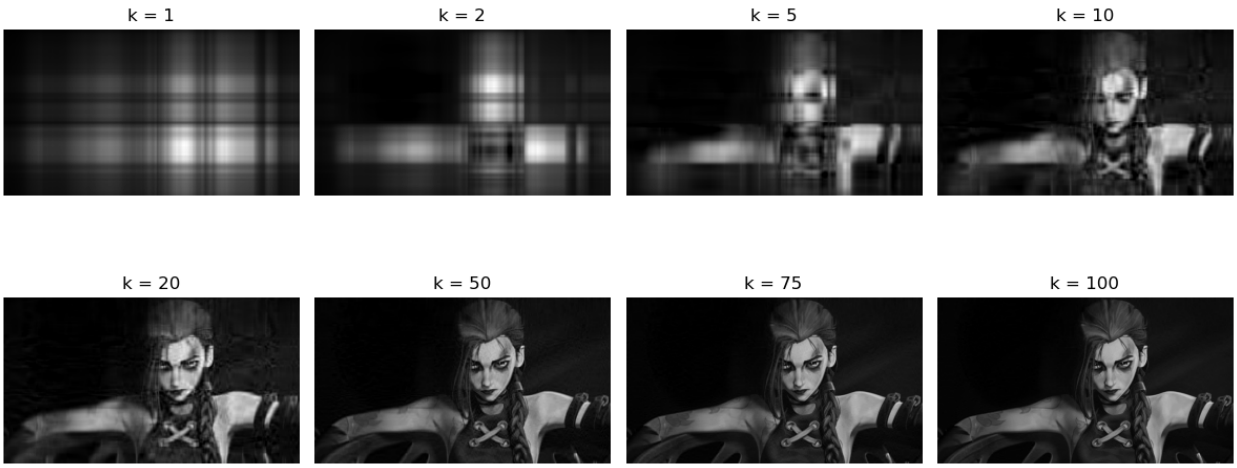


Figure 1: Rank-k approximation of the image

(b) Since the image is a 563×1000 matrix, it has rank at most 563. So there are at very most 563 non-zero singular values, so a rank 563 SVD is prescisely the image.

(c) Here is a picture of the first left/right singular vectors, plotted against the column/row average:

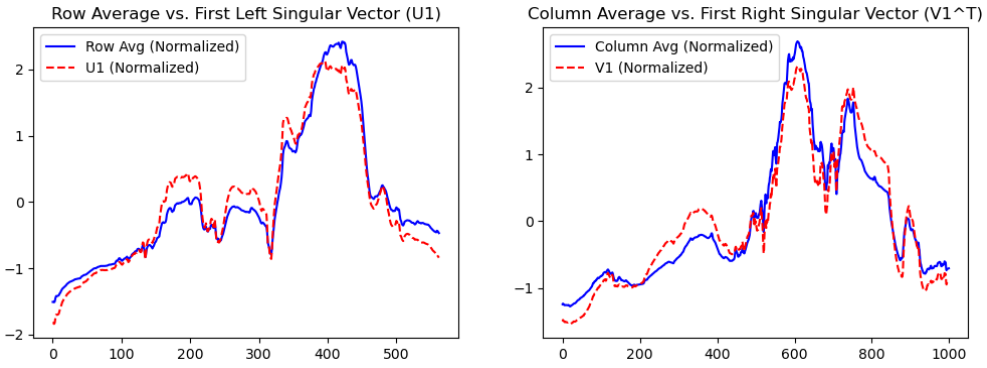


Figure 2: First left/right singular vectors plotted against the column/row average

As you can see, these are almost the same thing. The first left/right singular vectors very closely approximate the column/row average of the image. This is of course what I would've expected since it would make sense that the best rank-1 approximation should be some sort of average.

(d) Recall that an SVD of $M \in \mathbb{R}^{m \times n}$ is $M = U\Sigma V^T$, where U is $m \times m$, Σ is $m \times n$ and V is $n \times n$. So if we had a rank-k approximation, we would need to store the k singular values, the $m \times k$ matrix U and the $k \times n$ matrix V. This takes a total of $m \times k + k + k \times n = mk + kn + k$ bytes, which is $O(k(m+n))$. In the above image, at least for me, the $k = 50$ approximation looks almost indistinguishable from the real image. A $k = 50$ approximation thus takes approximately $50 \times (563 + 1000) = 50 \times 1563 = 78150$ bytes, which is a 7x savings from $563 \times 1000 = 563000$ bytes needed to store the original image.

(e) For very small values of k, we can see a lot of blurring. In computer vision, I learned that the blurring filter is really just a moving average. From what we discussed above, the first singular vectors are column and row averages. So we will see a lot of blurring since that's just what averages look like. The next few singular vectors are probably some kind of averages too, but when you put a lot of them together it looks great.

2. **Word Embeddings**

(a) Here is a plot of the eigenvalues of $\hat{M}$ in decreasing order:

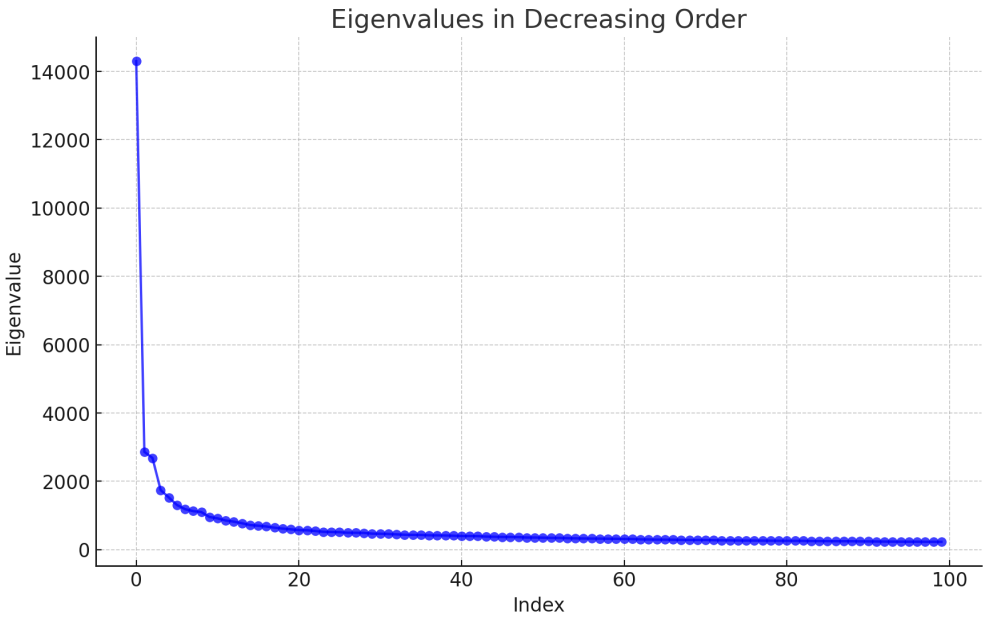


Figure 3: Eigenvalues of $\hat{M}$ in decreasing order

Except for the extremely large outlier of 14000 at the top, the eigenvalues exhibit gradual decay, so I would say that $\hat{M}$ is not close to being low rank.

(b) Here are 6 interesting singular vectors and their corresponding strongest words:

```
#1 singular vector most influential words:
['the', 'and', 'of', 'in', 'to', 'for', 'as', 'is', 'with',
   'was']
```

It looks like the first eigenvector measures if you are an article.

```
#3 singular vector most influential words:
['born', 'john', 'james', 'david', 'robert', 'william', 'jr
   ', 'george', 'thomas', 'michael']
```

Outside of the weird 'born', it seems like the third eigenvector measures if you are a (male) name.

```
#6 singular vector most influential words:
['troops', 'digital', 'science', 'software', 'technology', '
   engineering', 'him', 'computer', 'research', 'online']
```

These seem to measure if you are a technology word (it is interesting that 'him' shows up in the list of the 10 biggest words, I am wondering if this is related to how the distribution of tech workers is skewed towards men).

```
#7 singular vector most influential words:
['ancient', 'medieval', 'century', 'greek', 'culture', '
   latin', 'poetry', 'literature', 'folk', 'contemporary']
```

These look like words that describe history.

```
#10 singular vector most influential words:
['ii', 'commander', 'german', 'de', 'iii', 'squadron', 'mm',
    'aircraft', 'iv', 'community']
```

These are words that describe WWII.

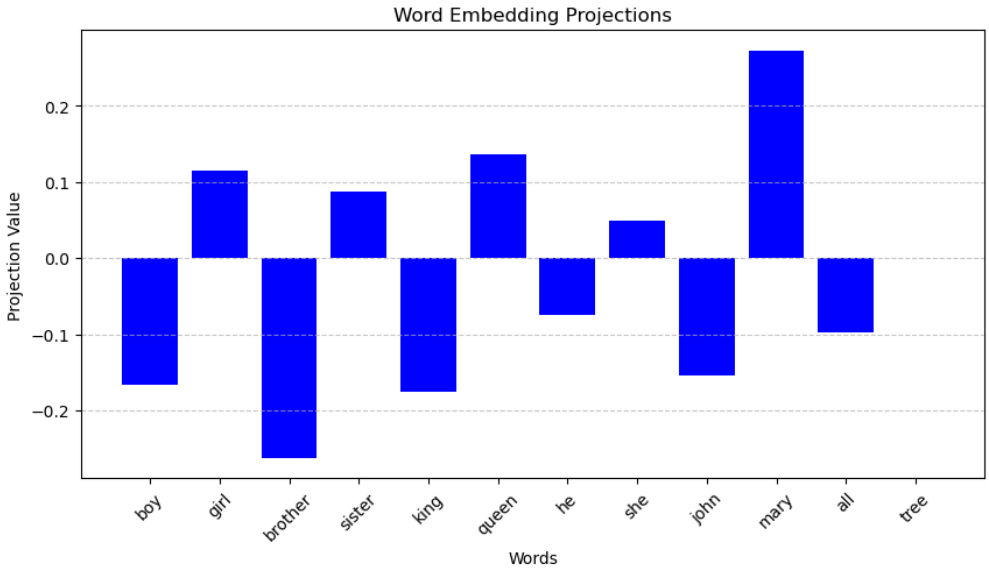
```
#17 singular vector most influential words:
['republican', 'democratic', 'conservative', 'at', 'her', '
   anti', 'election', 'paris', 'she', 'maria']
```

These are political words (it is again interesting that 'her' and 'she' show up here).

PCA measures the direction of greatest variance. With our starting assumption that a word is described by its context, we would expect the most variance to be in the direction of the semantics that humans usually use. For example, articles are used in an extremely different context, than names. Names come at the start of a sentence usually and articles are words that go in between two other words. So it makes sense that the eigenvectors would have some measure of semantics. There are of course eigenvectors which aren't interporable, such as:

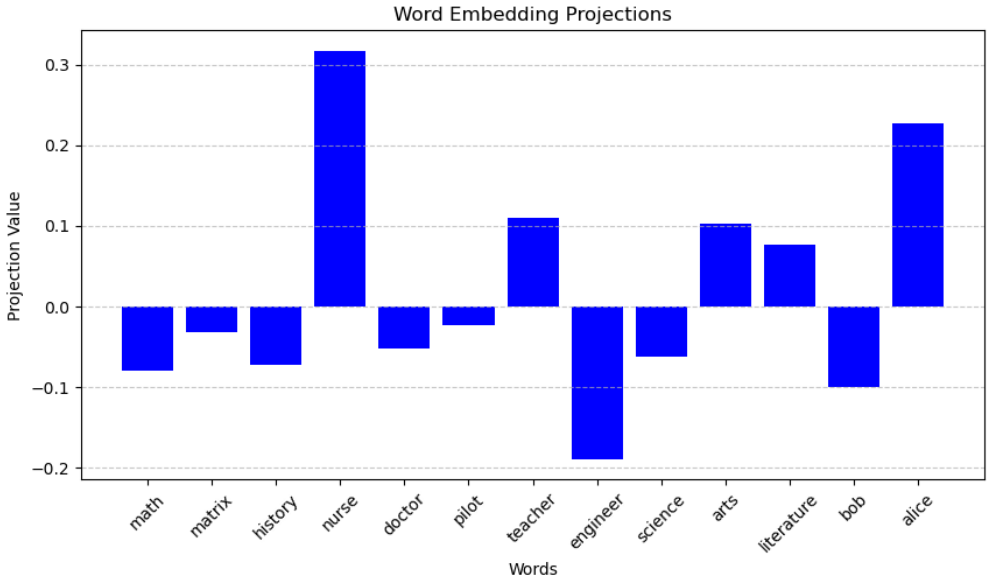
```
#4 singular vector most influential words:
['district', 'you', 'county', 'album', 'council', 'love', '
   university', 'me', 'national', 'my']
```

(c) Here is a picture of the projections of the words onto $u = u_1 - u_2$ where u_1 is th embedding of woman and u_2 is the embedding of man:



Who would've thought. Tree is orthogonal to what makes a man into a woman. This is a telling picture: you would expect brother = sister + (man - woman), which is why it has a really strong negative inner product with woman - man. It is also able to tell "gender" of the name Mary and John using this same logic, since John = Mary + (brother - sister), and Mary = John + (sister - brother). Similarly, we would expect that king = queen + (brother - sister) and queen = king + (sister - brother), and we do indeed get those inner products.

(d) Here is a similar plot for those job words:



The implication is that 'nurse' is a more woman job and 'engineer' is a more man job, based off its correlation with woman-man. Although weaker, it thinks literature, arts, teacher and

teacher are more woman jobs, and math, matrix, history, doctor, pilot, and science are more man jobs. It also thinks that bob is a more masculine name and alice is a more feminine one.

This is problematic because woman - man, has a really strong positive correlation with nurse, and a really strong negative correlation with engineer. If we trained an NLP model on wikipiedia data, and then used it to recommend successful hires, it would recommend more men than woman based off of this data, since it thinks that more men are engineers than woman. Vice versa, it would recommend more woman than men to be nurses.

(e) Because we normalized the rows of U, which is a 10,000 $\times$ 100 matrix, where word i has embedding into $\mathbb{R}^{100}$ given by $U[i]$, the norm of each row is 1. In particular, similarity$(w_i, w_j) = \langle w_i, w_j \rangle / \|w_i\| \|w_j\| = \langle w_i, w_j \rangle$, so this definition corresponds with our usual definition of similarity.

The 10 most similar words to 'washington' I found were:

```
['boston', 'chicago', 'york', 'houston', 'philadelphia', '
   baltimore', 'kansas', 'texas', 'florida', 'michigan']
```

The accuracy of the dataset analogy_tasks.txt I got was 0.549 (I told the analogy finder that it couldn't output a1, a2 or b1 when we are asking for a1:a2::b1:b2, so it might be a little higher). This is extremely impressive, since it has like 10,000 words to choose from and gets the right word, which would randomly happen with probability 10^{-4}, 1/2 of the time. It is more than 3 orderes of magnitude better than random.

This approach has a few challenges. For starters, something like this:

```
#3530 india:indian::ukraine:ukrainian got romanian
#3533 india:indian::brazil:brazilian got mexican
#3535 india:indian::chile:chilean got mexican
#3537 india:indian::croatia:croatian got romanian
#3538 india:indian::denmark:danish got swedish
```

As you can see, the answers it gives are not too wildly off. It gives countries that are reasonably similar, like Ukraine and Romania. I would suspect that this is due to romanian and ukranian having very similar contexts. So it will output something that is reasonable, but might not get the correct answer. We can see this with not just countries:

```
#2291 simple:simpler::fast:faster got slower
```

Again, faster and slower have really similar contexts, so they are very similar words. simple vs simpler captures adjective vs adverb, but it does not capture any semantic knowledge

about being fast or slow. So it will simply pick an adverb that has similar context to faster. Sometimes this is ‘faster’ but in our case this is ‘slower’.

You can see this happening again with:

```
#4294 jumping:jumped::decreasing:decreased got risen
```

However, sometimes it is just plain wrong. No feasible explanation for why. Like this:

```
#5014 dollar:dollars::man:men got woman
#5018 dollar:dollars::woman:women got man
```

This is the most curious result I found.

It can also just get the tense wrong:

```
#5579 write:writes::sing:sings got sang
```

It also really struggles with adverbs. For example:

```
#1535 apparent:apparently::complete:completely got actually
#1536 apparent:apparently::free:freely got actually
#1537 apparent:apparently::immediate:immediately got
   reportedly
#1538 apparent:apparently::most:mostly got quite
#1542 apparent:apparently::precise:precisely got supposedly
#1543 apparent:apparently::professional:professionally got
   already
#1544 apparent:apparently::quick:quickly got surprisingly
#1545 apparent:apparently::rapid:rapidly got initially
#1546 apparent:apparently::rare:rarely got reportedly
#1547 apparent:apparently::safe:safely got supposedly
#1548 apparent:apparently::serious:seriously got reportedly
#1549 apparent:apparently::slow:slowly got too
#1550 apparent:apparently::sudden:suddenly got reportedly
#1551 apparent:apparently::typical:typically got essentially
#1552 apparent:apparently::usual:usually got essentially
#1553 complete:completely::free:freely got totally
#1554 complete:completely::immediate:immediately got totally
#1555 complete:completely::most:mostly got totally
#1556 complete:completely::obvious:obviously got totally
```

```
#1557 complete:completely::occasional:occasionally got
   totally
#1558 complete:completely::possible:possibly got totally
#1559 complete:completely::precise:precisely got totally
#1560 complete:completely::professional:professionally got
   entirely
#1561 complete:completely::quick:quickly got totally
#1562 complete:completely::rapid:rapidly got totally
#1563 complete:completely::rare:rarely got totally
#1564 complete:completely::safe:safely got totally
#1565 complete:completely::serious:seriously got totally
#1567 complete:completely::sudden:suddenly got totally
#1568 complete:completely::typical:typically got entirely
#1569 complete:completely::usual:usually got totally
#1570 complete:completely::apparent:apparently got totally
```

Lots of ‘totally’. This might be related to adverbs having different context to their adjectives. For example, we might say ‘it is apparent that’, to emphasize that something is really true. However, ‘apparently’ has a very different meaning, almost questioning if the result is right. So it gets confused and outputs totally.