446 Section 09

TA: Varun Ananth

Plans for today!

1. This
2. Reminders
3. SVD
4. PCA
5. CNNs

Reminders

- HW4 due March 12th!
- Final is Wednesday March 19th!
  - < 2 weeks from today

SVD

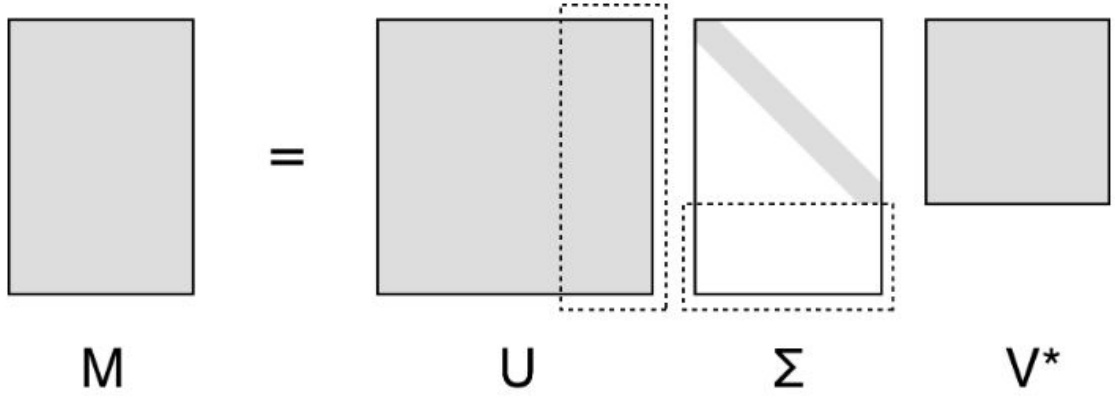
Singular **V**alue **D**ecomposition

Fact: Any matrix **M** can be decomposed into three separate matrices.

- I will not be discussing how to *solve* SVD, rather what its solution *means.*

(BTW: This is equivalent to thinking of **M** as a rotation, scaling, and another rotation).

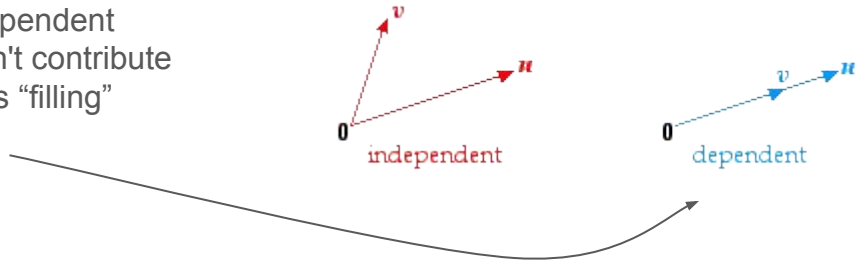
Say **M** is rank **r**



Alternative explanation:
Rank is how much space the matrix “fills”.

Linearly dependent vectors don't contribute more to this “filling”

How I like to think of rank: How much space is **actually** taken up by the vectors in the matrix?



Rank-r approximations

Say that **M** is “bloated”. It has many linearly dependent vectors which increase its size by a lot for no reason.

The information of **M** is confined to the subspace spanned by its linearly independent vectors

Basically minimizing MSE for the matrix reconstruction

$$\text{minimize}_{L \in \mathbb{R}^{m \times n}} \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{i,j} - L_{i,j})^2$$

$$\text{subject to } \text{rank}(L) = r$$

- The optimal rank-r approximation is $L = U_{1:r} S_{1:r,1:r} V_{1:r}^T$

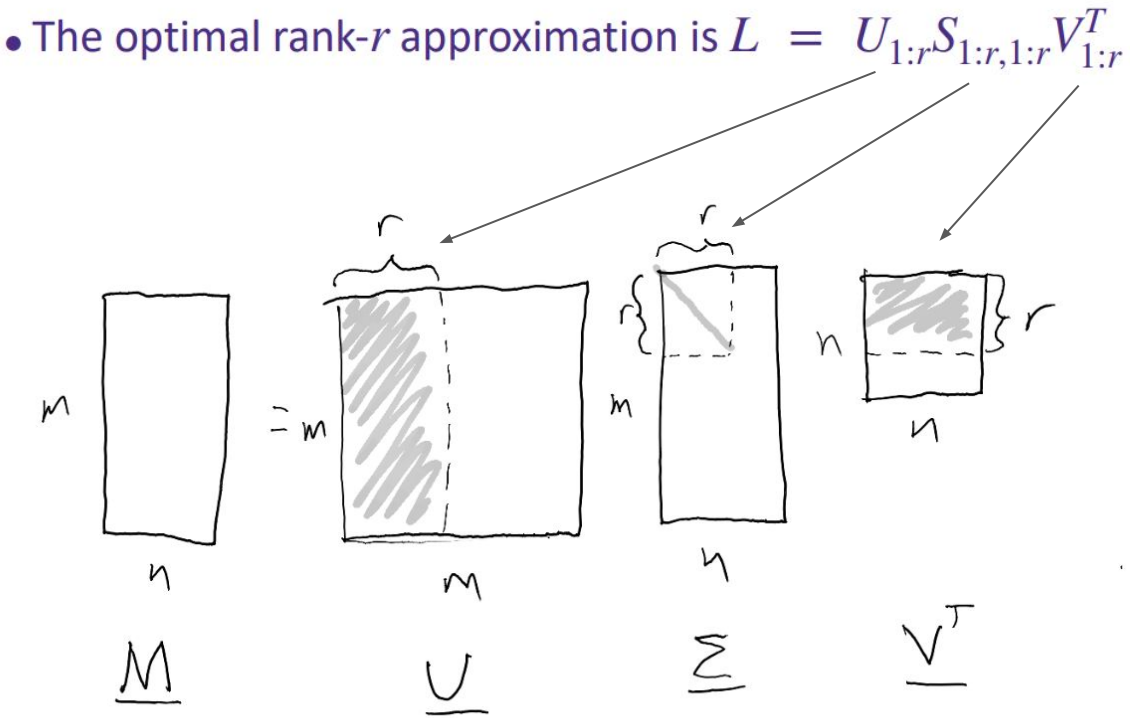
TL;DR: SVD is useful because it lets us approximately describe **M** with much less data (especially if **M** is “bloated”)

What does this look like in terms of **U, S, and V?**

Rank-r approximations

Shaded areas are what is "enough" to reconstruct **M**.

For compression using SVD, we pick **r** to be small to save us lots of space!

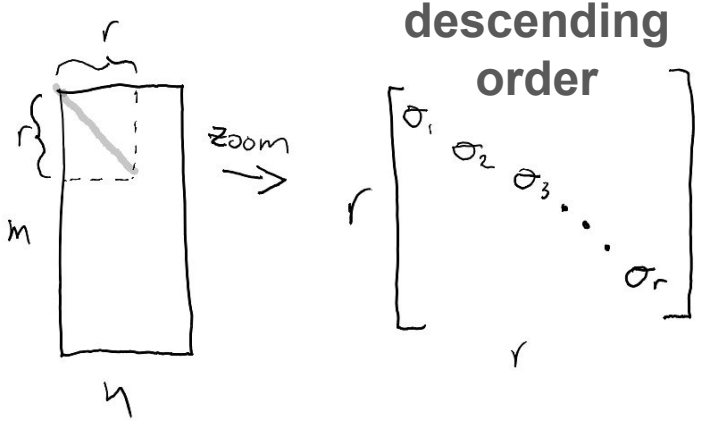


Understanding the decomposition

Consider what actually happens during the matrix multiplication.

Give names to the diagonal values of **Σ**.

Write the multiplication as a summation.



Sorted in descending order

Σ

Indexes refer to columns

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \sum_{i=1}^{r} u_i \sigma_i v_i^\top$$

Understanding the decomposition

Make sure the shapes check out in your head:

$$\mathbf{M} \in \mathbb{R}^{m \times n}$$

$$u_i \in \mathbb{R}^m;\ \sigma_i \in \mathbb{R};\ v_i \in \mathbb{R}^n \therefore$$

$$u_i \sigma_i v_i^\top \in \mathbb{R}^{m \times n}$$

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \sum_{i=1}^{r} u_i \sigma_i v_i^\top$$

Rank-r approximation

Rank-2 approximation

Rank-1 approximation

$$\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = u_1 \sigma_1 v_1^\top + u_2 \sigma_2 v_2^\top + \ldots + u_r \sigma_r v_r^\top$$

Understanding the decomposition

Make sure the shapes check out in your head:

$$\mathbf{M} \in \mathbb{R}^{m \times n}$$

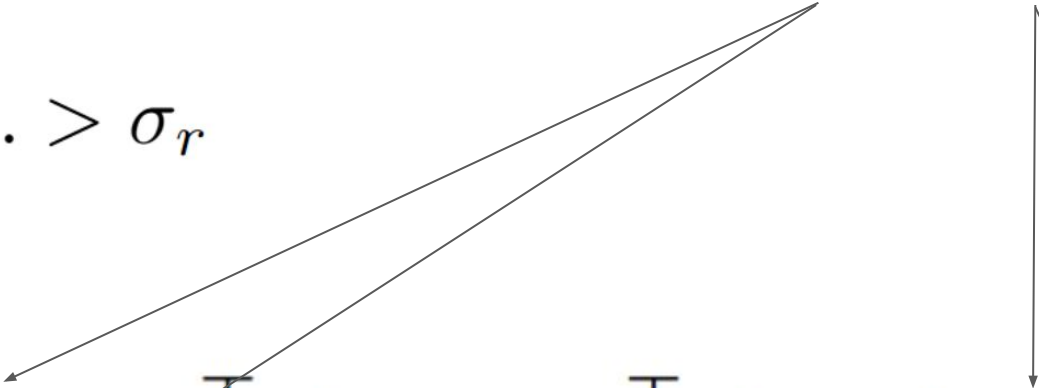
$$u_i \in \mathbb{R}^m;\ \sigma_i \in \mathbb{R};\ v_i \in \mathbb{R}^n \therefore$$

$$u_i \sigma_i v_i^\top \in \mathbb{R}^{m \times n}$$

Note: The u_i and v_i vectors are called left and right singular vectors respectively.

Q: Which singular vectors affect the reconstruction of **M** the most and least. Why?

A: $\sigma_1 > \sigma_2 > \ldots > \sigma_r$



$$\mathbf{U\Sigma V}^\top = u_1\sigma_1 v_1^\top + u_2\sigma_2 v_2^\top + \ldots + u_r\sigma_r v_r^\top$$

PCA

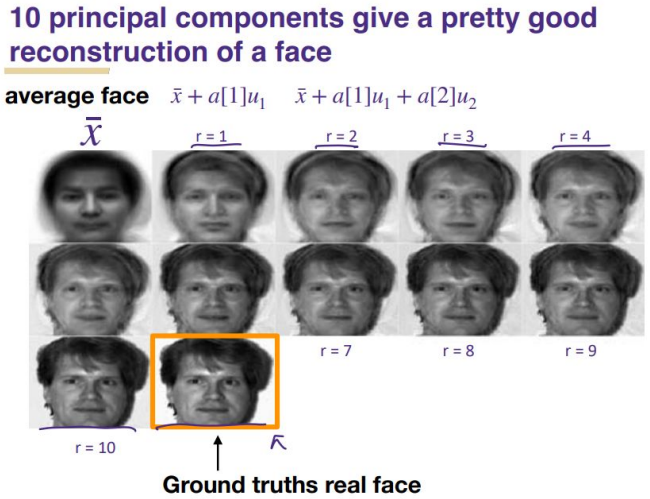
PCA: Dimensionality reduction

Dealing with high-dimensional data can lead to problems (curse of dimensionality)

Visualization is very hard for $d > 2$

Goal: Find a lower-dimensional representation of your data **X** with the least loss of information.

We need to find a subspace of the original data span to project down to. The **Principal Components** of our data can help us do this

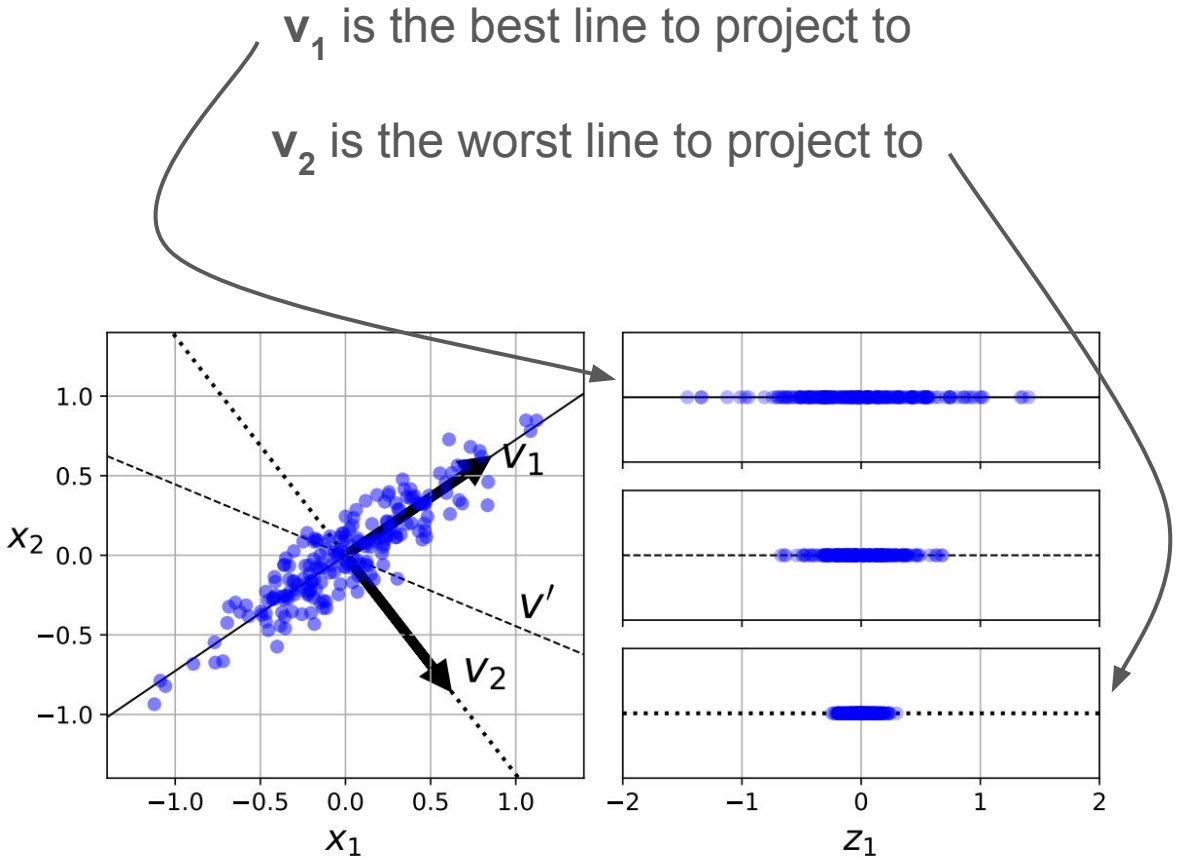


Variance maximization

One way to think about “information” in data is its spread, or variance.

Projection data down to one single point = loss of all information.

So what is the opposite?



Our goal is to find a projection matrix $\mathbf{V}_q \in \mathbb{R}^{d \times q}$ that minimizes the reconstruction error of our demeaned data. Mathematically, we want to find:

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^\top (x_i - \bar{x})||_2^2$$

Turns out, the $\mathbf{V}_q$ that minimizes this equation is **the first q eigenvectors of $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$**. There is a proof in the lecture notes, but here is the inuition:

- Eigenvalues/eigenvectors tell you the "direction and magnitude of greatest stretch" when a matrix $\mathbf{M}$ is applied as a transformation.
- If $\mathbf{X}$ is our data, make $\tilde{\mathbf{X}}$ the demeaned data. This makes $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ proportional to the **sample covariance matrix** of $\mathbf{X}$. Essentially, a matrix which holds information about the variance of $\mathbf{X}$.
- The eigenvectors of $\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}$ will tell us the directions of greatest variance. Intuitively, finding the eigenvectors of this should give us **principal components** to project onto!

Relating PCA & SVD

From lecture notes:

Theorem [SVD]: Let $A \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $A = USV^T$ where $S \in \mathbb{R}^{r \times r}$ is diagonal with non-negative entries, $U^T U = I$, and $V^T V = I$.

$\mathbf{V}$ are the first r eigenvectors of $\mathbf{A}^T\mathbf{A}$ with eigenvalues $\text{diag}(\mathbf{S}^2)$
$\mathbf{U}$ are the first r eigenvectors of $\mathbf{A}\mathbf{A}^T$ with eigenvalues $\text{diag}(\mathbf{S}^2)$

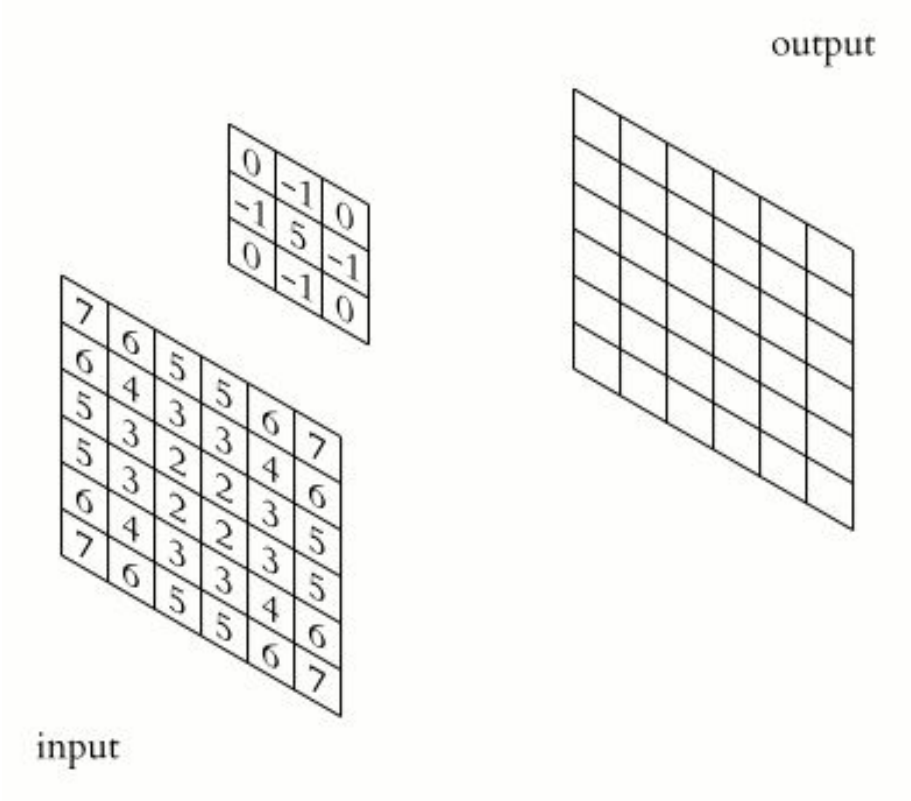
So how do we find the eigenvectors of $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$?

$\text{SVD}(\tilde{\mathbf{X}}) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \therefore$ *(definition of SVD)*
$\mathbf{V}$ contains the first $\mathbf{r}$ eigenvectors of $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}, \therefore$ *(fact from lecture notes)*
the principal components of $\mathbf{X}$ exist in $\mathbf{V}$ from the SVD of $\tilde{\mathbf{X}}$ *(PCs of* $\mathbf{X}$ *= eigenvectors of* $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$

Remember this?: $\sigma_1 > \sigma_2 > \ldots > \sigma_r$

Selecting the first *q* vectors from **V** gives us the directions with the GREATEST variance… **principal components!**

CNNs



This video shows a convolutional pass being done. The kernel they use here is a “sharpening kernel”

Shape of a convolutional layer / maxpooling output: For a $n \times n$ input, $f \times f$ filter, padding p and stride s, the output size is $o \times o$ where:

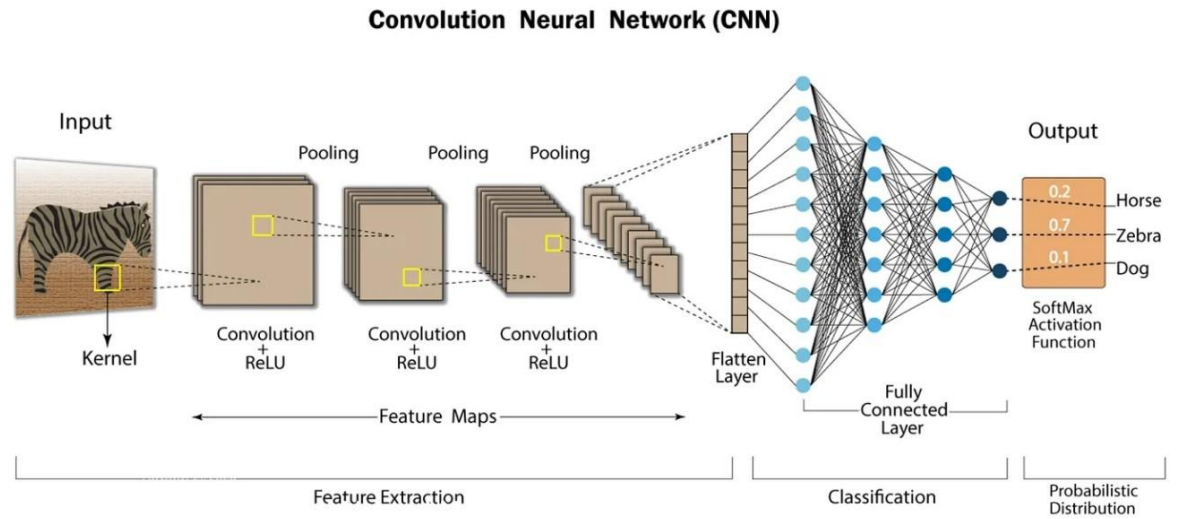
$$o = \frac{n - f + 2p}{s} + 1$$

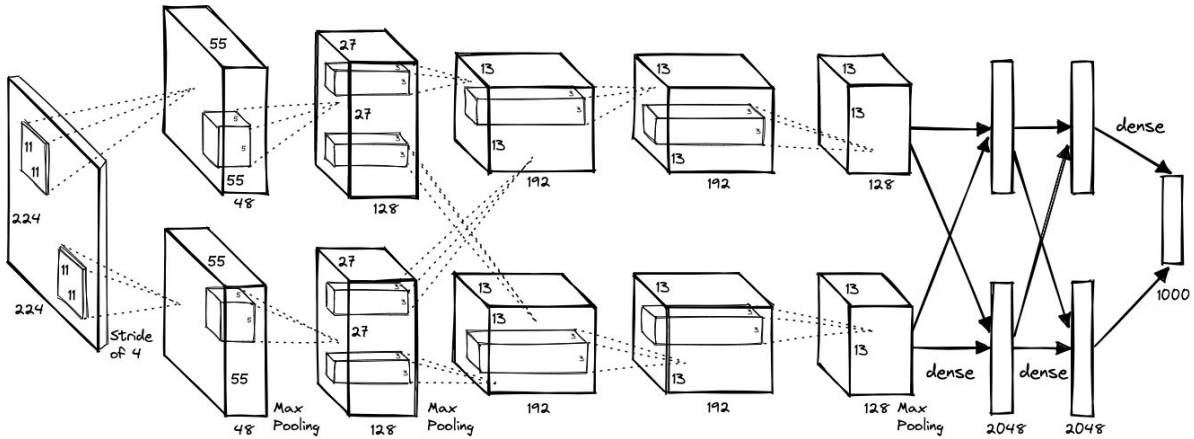
An equation worth memorizing…

Convolutional Neural Networks

I find it most effective to just answer questions about these.

Ask away!





Questions/Chat Time!