Linear algebra in your daily (digital) life

Andrew Schultz

Wellesley College

December 11, 2017

Outline of the talk

- Who cares about eigenvalues?

Outline of the talk

- Who cares about eigenvalues?
    - Google's PageRank algorithm

Outline of the talk

- Who cares about eigenvalues?
    - Google's PageRank algorithm
- Who cares about orthonormality?

Outline of the talk

- Who cares about eigenvalues?
  - Google's PageRank algorithm
- Who cares about orthonormality?
  - Filtering noise
  - Image compression

Outline of the talk

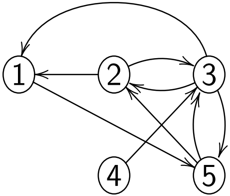
- Who cares about eigenvalues?
  - Google's PageRank algorithm
- Who cares about orthonormality?
  - Filtering noise
  - Image compression
- Who cares about change of basis?

Outline of the talk

- Who cares about eigenvalues?
  - Google's PageRank algorithm
- Who cares about orthonormality?
  - Filtering noise
  - Image compression
- Who cares about change of basis?
  - Image compression
  - Sound compression

The general constraints

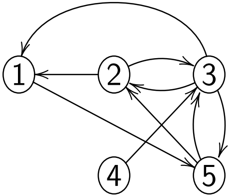
Suppose that G is a directed graph



The general constraints

Suppose that G is a directed graph

- V is the vertex set for G
- $A \subseteq V \times V$ is the set of arcs of G

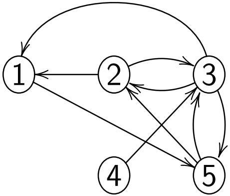


The general constraints

Suppose that G is a directed graph

- V is the vertex set for G
- $A \subseteq V \times V$ is the set of arcs of G

Want to rank vertices (using $R : V \to \mathbb{R}$)



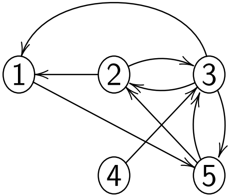
The general constraints

Suppose that G is a directed graph

- V is the vertex set for G
- $A \subseteq V \times V$ is the set of arcs of G

Want to rank vertices (using $R : V \to \mathbb{R}$)

- should depend only on structure of G



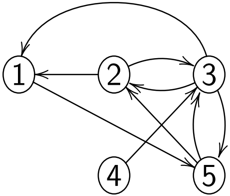
The general constraints

Suppose that G is a directed graph

- V is the vertex set for G
- $A \subseteq V \times V$ is the set of arcs of G

Want to rank vertices (using $R : V \to \mathbb{R}$)

- should depend only on structure of G
- shouldn't produce many ties



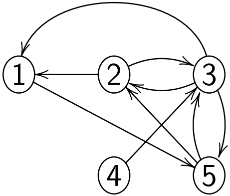
The general constraints

Suppose that G is a directed graph

- V is the vertex set for G
- $A \subseteq V \times V$ is the set of arcs of G

Want to rank vertices (using $R : V \to \mathbb{R}$)

- should depend only on structure of G
- shouldn't produce many ties
- should be equitable



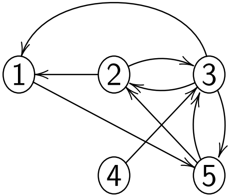
The general constraints

Suppose that G is a directed graph

- V is the vertex set for G
- $A \subseteq V \times V$ is the set of arcs of G

Want to rank vertices (using $R : V \to \mathbb{R}$)

- should depend only on structure of G
- shouldn't produce many ties
- should be equitable
- should be stable under "attack"



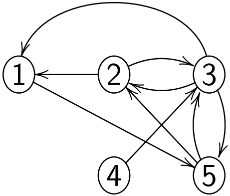
The general constraints

Suppose that G is a directed graph

- V is the vertex set for G
- $A \subseteq V \times V$ is the set of arcs of G

Want to rank vertices (using $R : V \to \mathbb{R}$)

- should depend only on structure of G
- shouldn't produce many ties
- should be equitable
- should be stable under "attack"
- should be computable



The first approach

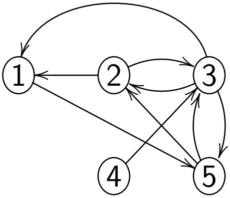
$(j) \longrightarrow (i)$ is a vote for (i) from (j)

Form matrix L so that

$$L_{i,j} = \begin{cases} 1 & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$

The first approach

$(j) \longrightarrow (i)$ is a vote for (i) from (j)



Form matrix L so that

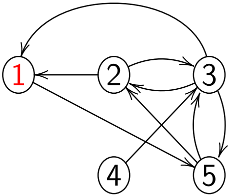
$$L_{i,j} = \begin{cases} 1 & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$

The first approach

$(j) \longrightarrow (i)$ is a vote for (i) from (j)

Form matrix L so that

$$L_{i,j} = \begin{cases} 1 & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



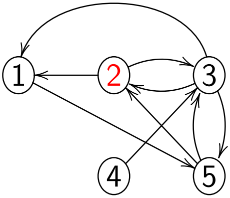
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The first approach

$(j) \longrightarrow (i)$ is a vote for (i) from (j)

Form matrix L so that

$$L_{i,j} = \begin{cases} 1 & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



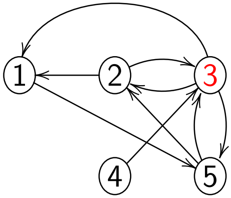
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The first approach

$(j) \longrightarrow (i)$ is a vote for (i) from (j)

Form matrix L so that

$$L_{i,j} = \begin{cases} 1 & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



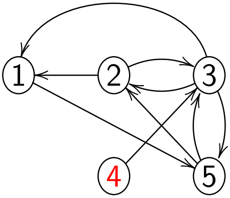
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The first approach

$(j) \longrightarrow (i)$ is a vote for (i) from (j)

Form matrix L so that

$$L_{i,j} = \begin{cases} 1 & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



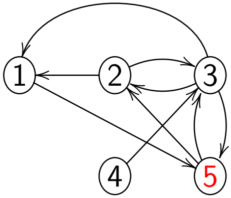
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The first approach

$(j) \rightarrow (i)$ is a vote for (i) from (j)

Form matrix L so that

$$L_{i,j} = \begin{cases} 1 & , \text{ if } (j) \rightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

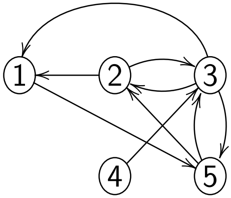
The first approach

$(j) \longrightarrow (i)$ is a vote for (i) from (j)

Form matrix L so that

$$L_{i,j} = \begin{cases} 1 & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$

Scheme 1: $R\left((i)\right) = \sum_{j=1}^{n} L_{i,j}$



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

The first approach

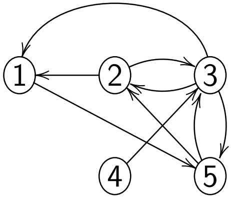
$\textcircled{j} \longrightarrow \textcircled{i}$ is a vote for $\textcircled{i}$ from $\textcircled{j}$

Form matrix L so that

$$L_{i,j} = \begin{cases} 1 & , \text{ if } \textcircled{j} \longrightarrow \textcircled{i} \\ 0 & , \text{ else} \end{cases}$$

Scheme 1: $R\left(\textcircled{i}\right) = \sum_{j=1}^{n} L_{i,j}$

Page 3 wins (score 3), Pages 1,2,5 tie for second (score 2), Page 4 loses



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Problems with first approach

- Potential for lots of ties

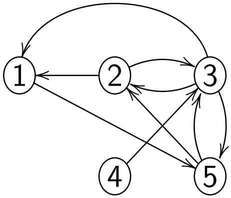
Problems with first approach

- Potential for lots of ties

$$\begin{pmatrix} 0 & \color{red}{1} & \color{red}{1} & 0 & 0 \\ 0 & 0 & \color{red}{1} & 0 & \color{red}{1} \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ \color{red}{1} & 0 & \color{red}{1} & 0 & 0 \end{pmatrix}$$

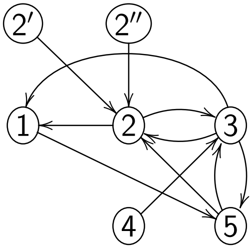
Problems with first approach

- Potential for lots of ties
- Evil users can manipulate results



Problems with first approach

- Potential for lots of ties
- Evil users can manipulate results



Problems with first approach

- Potential for lots of ties
- Evil users can manipulate results
- Not equitable

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Problems with first approach

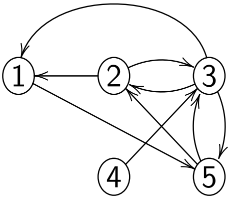
- Potential for lots of ties
- Evil users can manipulate results
- Not equitable

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ \color{red}{1} & 0 & \color{red}{1} & 0 & 0 \end{pmatrix}$$

Updating our approach

Each page gets a total of 1 vote:

$$\ell(j) = \sum_i L_{i,j}$$



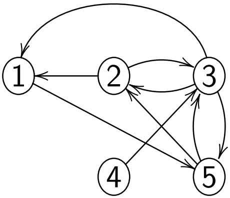
Updating our approach

Each page gets a total of 1 vote:

$$\ell(j) = \sum_i L_{i,j}$$

Form matrix W so that

$$W_{i,j} = \begin{cases} \frac{1}{\ell(j)} & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



$$\begin{pmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0.5 \\ 0 & 0.5 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.33 & 0 & 0 \end{pmatrix}$$

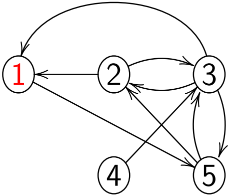
Updating our approach

Each page gets a total of 1 vote:

$$\ell(j) = \sum_i L_{i,j}$$

Form matrix W so that

$$W_{i,j} = \begin{cases} \frac{1}{\ell(j)} & \text{, if } (j) \longrightarrow (i) \\ 0 & \text{, else} \end{cases}$$



$$\begin{pmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0.5 \\ 0 & 0.5 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.33 & 0 & 0 \end{pmatrix}$$

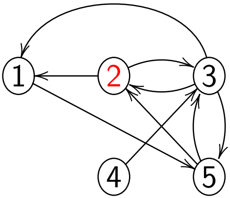
Updating our approach

Each page gets a total of 1 vote:

$$\ell(j) = \sum_i L_{i,j}$$

Form matrix W so that

$$W_{i,j} = \begin{cases} \frac{1}{\ell(j)} & \text{, if } (j) \longrightarrow (i) \\ 0 & \text{, else} \end{cases}$$



$$\begin{pmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0.5 \\ 0 & 0.5 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.33 & 0 & 0 \end{pmatrix}$$

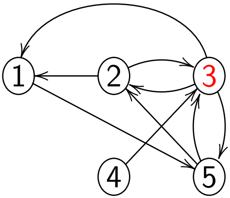
Updating our approach

Each page gets a total of 1 vote:

$$\ell(j) = \sum_i L_{i,j}$$

Form matrix W so that

$$W_{i,j} = \begin{cases} \frac{1}{\ell(j)} & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



$$\begin{pmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0.5 \\ 0 & 0.5 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.33 & 0 & 0 \end{pmatrix}$$

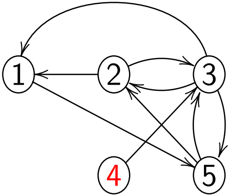
Updating our approach

Each page gets a total of 1 vote:

$$\ell(j) = \sum_i L_{i,j}$$

Form matrix W so that

$$W_{i,j} = \begin{cases} \frac{1}{\ell(j)} & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



$$\begin{pmatrix} 0 & 0.5 & 0.33 & \color{red}{0} & 0 \\ 0 & 0 & 0.33 & \color{red}{0} & 0.5 \\ 0 & 0.5 & 0 & \color{red}{1} & 0.5 \\ 0 & 0 & 0 & \color{red}{0} & 0 \\ 1 & 0 & 0.33 & \color{red}{0} & 0 \end{pmatrix}$$

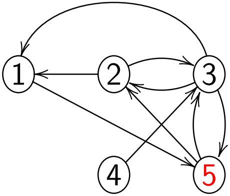
Updating our approach

Each page gets a total of 1 vote:

$$\ell(j) = \sum_i L_{i,j}$$

Form matrix W so that

$$W_{i,j} = \begin{cases} \frac{1}{\ell(j)} & , \text{ if } (j) \longrightarrow (i) \\ 0 & , \text{ else} \end{cases}$$



$$\begin{pmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0.5 \\ 0 & 0.5 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.33 & 0 & 0 \end{pmatrix}$$

Updating our approach

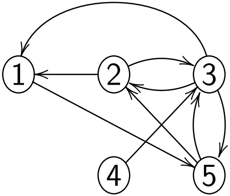
Each page gets a total of 1 vote:

$$\ell(j) = \sum_i L_{i,j}$$

Form matrix W so that

$$W_{i,j} = \begin{cases} \frac{1}{\ell(j)} & \text{, if } (j) \longrightarrow (i) \\ 0 & \text{, else} \end{cases}$$

Scheme 2: $R\left(\textcircled{i}\right) = \sum_{j=1}^{n} W_{i,j}$



$$\begin{pmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0.5 \\ 0 & 0.5 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.33 & 0 & 0 \end{pmatrix}$$

Problems with second approach

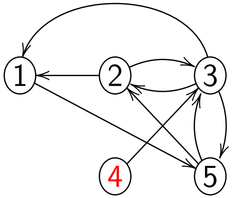
- Evil users can really manipulate results

Problems with second approach

- Evil users can really manipulate results
- Gives importance to links from unimportant pages

Problems with second approach

- Evil users can really manipulate results
- Gives importance to links from unimportant pages



$$\begin{pmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0.5 \\ 0 & 0.5 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.33 & 0 & 0 \end{pmatrix}$$

Weighting votes

Importance of $(j) \longrightarrow (i)$ should depend on $R((j))$.

Weighting votes

Importance of $\textcircled{j} \longrightarrow \textcircled{i}$ should depend on $R(\textcircled{j})$. Want

$$R\left(\textcircled{i}\right) = \sum_j R(\textcircled{j}) W_{i,j}$$

Weighting votes

Importance of $\textcircled{j} \longrightarrow \textcircled{i}$ should depend on $R(\textcircled{j})$. Want

$$R\left(\textcircled{i}\right) = \sum_j R(\textcircled{j}) W_{i,j}$$

$$W \begin{pmatrix} R(\textcircled{1}) \\ \vdots \\ R(\textcircled{n}) \end{pmatrix} = \begin{pmatrix} R(\textcircled{1}) \\ \vdots \\ R(\textcircled{n}) \end{pmatrix}$$

Weighting votes

Importance of $(j) \longrightarrow (i)$ should depend on $R((j))$. Want

$$R\left((i)\right) = \sum_j R((j)) W_{i,j}$$

$$W \begin{pmatrix} R((1)) \\ \vdots \\ R((n)) \end{pmatrix} = \begin{pmatrix} R((1)) \\ \vdots \\ R((n)) \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0 & 0 & 0.33 & 0 & 0.5 \\ 0 & 0.5 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0.33 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0.41 \\ 0.47 \\ 0.52 \\ 0 \\ 0.58 \end{pmatrix}$$ is 1-eigenvector

Some slight modifications

This system satisfies most of the properties that we're interested in.

Some slight modifications

This system satisfies most of the properties that we're interested in.

One small problem: what if $\dim(E_1) > 1$?

Some slight modifications

This system satisfies most of the properties that we're interested in.

One small problem: what if $\dim(E_1) > 1$?

Theorem

If an $n \times n$ matrix has positive entries and columns sum to 1, then $\dim(E_1) = 1$.

Some slight modifications

This system satisfies most of the properties that we're interested in.

One small problem: what if $\dim(E_1) > 1$?

Theorem

If an $n \times n$ matrix has positive entries and columns sum to 1, then $\dim(E_1) = 1$.

Let $PR = dW + (1-d)\begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & \ddots & \vdots \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}$

Computability

There are lots of webpages! How can we feasibly compute an eigenvector for $10^{13} \times 10^{13}$ matrix?

Computability

There are lots of webpages! How can we feasibly compute an eigenvector for $10^{13} \times 10^{13}$ matrix?

Row reduction is a bad idea:

- would take about $(10^{13})^3 = 10^{39}$ computations.

Computability

There are lots of webpages! How can we feasibly compute an eigenvector for $10^{13} \times 10^{13}$ matrix?

Row reduction is a bad idea:

- would take about $(10^{13})^3 = 10^{39}$ computations.
- the fastest super computer runs about 2.5×10^{15} computations per second.

Computability

There are lots of webpages! How can we feasibly compute an eigenvector for $10^{13} \times 10^{13}$ matrix?

Row reduction is a bad idea:

- would take about $(10^{13})^3 = 10^{39}$ computations.
- the fastest super computer runs about 2.5×10^{15} computations per second.
- this row reduction would take about 10^{17} years.

Computability

There are lots of webpages! How can we feasibly compute an eigenvector for $10^{13} \times 10^{13}$ matrix?

Row reduction is a bad idea:

- would take about $(10^{13})^3 = 10^{39}$ computations.
- the fastest super computer runs about 2.5×10^{15} computations per second.
- this row reduction would take about 10^{17} years.
  - Current age of the universe is about 10^{10} years.

Computability

There are lots of webpages! How can we feasibly compute an eigenvector for $10^{13} \times 10^{13}$ matrix?

Row reduction is a bad idea:

- would take about $(10^{13})^3 = 10^{39}$ computations.
- the fastest super computer runs about 2.5×10^{15} computations per second.
- this row reduction would take about 10^{17} years.
  - Current age of the universe is about 10^{10} years.
  - If a supercomputer started at the dawn of the universe, then today it would be 0.0000001% done.

Approximation is our friend

Suppose that we have an eigenbasis $\{\overrightarrow{v_1}, \cdots, \overrightarrow{v_{10^{13}}}\}$ for PR.

Approximation is our friend

Suppose that we have an eigenbasis $\{\overrightarrow{v_1}, \cdots, \overrightarrow{v_{10^{13}}}\}$ for PR. Then a random vector $\overrightarrow{v}$ can be expressed in the form

$$\overrightarrow{v} = c_1 \overrightarrow{v_1} + \cdots + c_{10^{13}} \overrightarrow{v_{10^{13}}}.$$

Approximation is our friend

Suppose that we have an eigenbasis $\{\overrightarrow{v_1}, \cdots, \overrightarrow{v_{10^{13}}}\}$ for PR. Then a random vector $\overrightarrow{v}$ can be expressed in the form

$$\overrightarrow{v} = c_1\overrightarrow{v_1} + \cdots + c_{10^{13}}\overrightarrow{v_{10^{13}}}.$$

Then

$$PR^k\overrightarrow{v} = c_1\lambda_1^k\overrightarrow{v_1} + \cdots + c_{10^{13}}\lambda_{10^{13}}^k\overrightarrow{v_{10^{13}}}.$$

Approximation is our friend

Suppose that we have an eigenbasis $\{\vec{v_1}, \cdots, \overrightarrow{v_{10^{13}}}\}$ for PR. Then a random vector $\vec{v}$ can be expressed in the form

$$\vec{v} = c_1 \vec{v_1} + \cdots + c_{10^{13}} \overrightarrow{v_{10^{13}}}.$$

Then

$$PR^k \vec{v} = c_1 \lambda_1^k \vec{v_1} + \cdots + c_{10^{13}} \lambda_{10^{13}}^k \overrightarrow{v_{10^{13}}}.$$

Fact: If an $n \times n$ matrix has positive entries and columns sum to 1, then 1 is the largest eigenvalue (in absolute value).

Approximation is our friend

Suppose that we have an eigenbasis $\{\overrightarrow{v_1}, \cdots, \overrightarrow{v_{10^{13}}}\}$ for PR. Then a random vector $\overrightarrow{v}$ can be expressed in the form

$$\overrightarrow{v} = c_1\overrightarrow{v_1} + \cdots + c_{10^{13}}\overrightarrow{v_{10^{13}}}.$$

Then

$$PR^k\overrightarrow{v} = c_1\lambda_1^k\overrightarrow{v_1} + \cdots + c_{10^{13}}\lambda_{10^{13}}^k\overrightarrow{v_{10^{13}}}.$$

Fact: If an $n \times n$ matrix has positive entries and columns sum to 1, then 1 is the largest eigenvalue (in absolute value).

Hence $\lim_{k\to\infty} PR^k\overrightarrow{v}$ is an eigenvalue with eigenvalue 1.

The Singular Value Decomposition

An often overlooked gem in linear algebra is

The Singular Value Decomposition

For any $r \times c$ matrix A with real entries, there exist

The Singular Value Decomposition

An often overlooked gem in linear algebra is

The Singular Value Decomposition

For any $r \times c$ matrix A with real entries, there exist orthonormal bases $\{\vec{v}_1, \cdots, \vec{v}_r\} \subseteq \mathbb{R}^r$ and $\{\vec{w}_1, \cdots, \vec{w}_c\} \subseteq \mathbb{R}^c$

The Singular Value Decomposition

An often overlooked gem in linear algebra is

The Singular Value Decomposition

For any $r \times c$ matrix A with real entries, there exist orthonormal bases $\{\vec{v}_1, \cdots, \vec{v}_r\} \subseteq \mathbb{R}^r$ and $\{\vec{w}_1, \cdots, \vec{w}_c\} \subseteq \mathbb{R}^c$, and scalars $\sigma_1 \geq \cdots \geq \sigma_\ell \geq 0$ such that

The Singular Value Decomposition

An often overlooked gem in linear algebra is

The Singular Value Decomposition

For any $r \times c$ matrix A with real entries, there exist orthonormal bases $\{\vec{v}_1, \cdots, \vec{v}_r\} \subseteq \mathbb{R}^r$ and $\{\vec{w}_1, \cdots, \vec{w}_c\} \subseteq \mathbb{R}^c$, and scalars $\sigma_1 \geq \cdots \geq \sigma_\ell \geq 0$ such that

$$A = \left(\begin{array}{c|c|c} \vec{v}_1 & \cdots & \vec{v}_r \end{array} \right) \begin{pmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \end{pmatrix} \begin{pmatrix} \vec{w}_1 \\ \hline \vdots \\ \hline \vec{w}_c \end{pmatrix}.$$

What SVD captures

This decomposition encodes loads of info for A

What SVD captures

This decomposition encodes loads of info for A

- rank(A) is number of non-zero σ_i's

What SVD captures

This decomposition encodes loads of info for A

- rank(A) is number of non-zero σ_i's
- Orthonormal bases for $\mathrm{im}(A)$, $\ker(A)$, $\mathrm{im}(A^T)$, $\ker(A^T)$

What SVD captures

This decomposition encodes loads of info for A

- rank(A) is number of non-zero σ_i's
- Orthonormal bases for $\mathrm{im}(A)$, $\ker(A)$, $\mathrm{im}(A^T)$, $\ker(A^T)$
- if A is square, $|\det(A)| = \prod \sigma_i$

What SVD captures

This decomposition encodes loads of info for A

- rank(A) is number of non-zero σ_i's
- Orthonormal bases for $\mathrm{im}(A)$, $\ker(A)$, $\mathrm{im}(A^T)$, $\ker(A^T)$
- if A is square, $|\det(A)| = \prod \sigma_i$
- simple expression for PsuedoInverse

What SVD captures

This decomposition encodes loads of info for A

- rank(A) is number of non-zero σ_i's
- Orthonormal bases for $\mathrm{im}(A)$, $\ker(A)$, $\mathrm{im}(A^T)$, $\ker(A^T)$
- if A is square, $|\det(A)| = \prod \sigma_i$
- simple expression for PsuedoInverse
- Quick test for numerical stability of matrix

SVD for approximating with rank 1 matrices

SVD also gives us a method for writing A as sum of rank 1 matrices:

SVD for approximating with rank 1 matrices

SVD also gives us a method for writing A as sum of rank 1 matrices:

$$A = \sum_{i=1}^{\mathrm{rk}(A)} \underbrace{\sigma_i \left(\begin{array}{c|c|c} \vec{v}_1 & \cdots & \vec{v}_r \end{array} \right) E(i,i) \left(\begin{array}{c} \vec{w}_1 \\ \hline \vdots \\ \hline \vec{w}_c \end{array} \right)}_{A_i}.$$

SVD for approximating with rank 1 matrices

SVD also gives us a method for writing A as sum of rank 1 matrices:

$$A = \sum_{i=1}^{\text{rk}(A)} \underbrace{\sigma_i \left(\begin{array}{c|c|c} \vec{v}_1 & \cdots & \vec{v}_r \end{array} \right) E(i,i) \left(\begin{array}{c} \vec{w}_1 \\ \hline \vdots \\ \hline \vec{w}_c \end{array} \right)}_{A_i}.$$

Since $\sigma_1 \geq \cdots \geq \sigma_\ell \geq 0$, A_{i+1} is "less significant" than A_i

SVD for approximating with rank 1 matrices

SVD also gives us a method for writing A as sum of rank 1 matrices:

$$A = \sum_{i=1}^{\mathrm{rk}(A)} \underbrace{\sigma_i \left(\vec{v}_1 \Big| \cdots \Big| \vec{v}_r \right) E(i,i) \begin{pmatrix} \vec{w}_1 \\ \hline \vdots \\ \hline \vec{w}_c \end{pmatrix}}_{A_i}.$$

Since $\sigma_1 \geq \cdots \geq \sigma_\ell \geq 0$, A_{i+1} is "less significant" than A_i

If $\sum_{i>s} \sigma_i$ is "insignificant," then we have $A \approx \sum_{i=1}^{s} A_i$

Noise Filtering

Matrix representations of images

You can think of an image as a matrix.

- Each pixel contains a gray value
- Gray values range from 0 to 255

Matrix representations of images

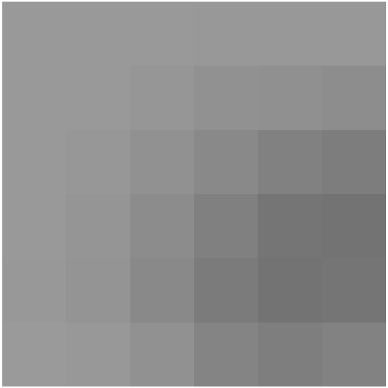
You can think of an image as a matrix.

- Each pixel contains a gray value
- Gray values range from 0 to 255

Matrix representations of images

You can think of an image as a matrix.

- Each pixel contains a gray value
- Gray values range from 0 to 255



Matrix representations of images

You can think of an image as a matrix.

- Each pixel contains a gray value
- Gray values range from 0 to 255

153	153	153	152	152	152
153	153	150	145	144	141
153	151	145	137	129	125
153	149	140	128	117	115
152	148	137	123	115	117
154	152	145	132	126	130

Keeping only "significant" terms

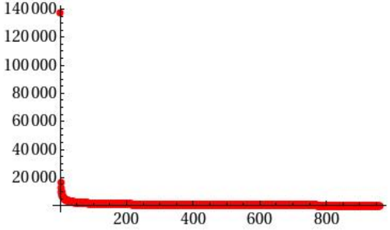
According to our theory, if there are s-many significant singular values, then

$$M \approx \sum_{i=1}^{s} M_i$$

Keeping only "significant" terms

According to our theory, if there are s-many significant singular values, then

$$M \approx \sum_{i=1}^{s} M_i$$

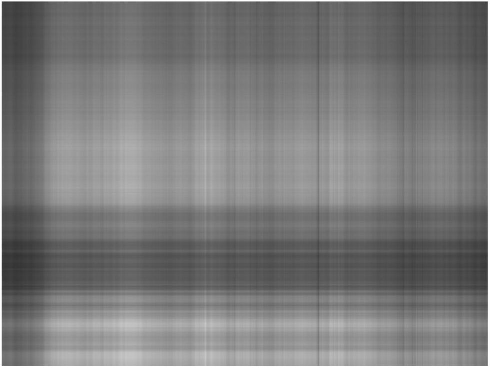


$$\sigma_1 \approx 138,000$$
$$\sigma_2 \approx 17,000$$
$$\sigma_{50} \approx 2,200$$
$$\sigma_{200} \approx 900$$

Some approximations

Let's see what our truncated matrix "looks like"

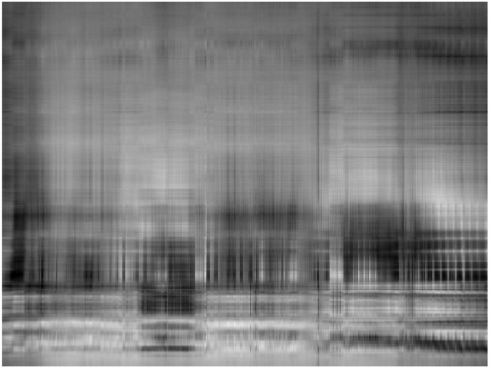
- s=1
- Compression: 0.18%



Some approximations

Let's see what our truncated matrix "looks like"

- s=5
- Compression: 0.9%



Some approximations

Let's see what our truncated matrix "looks like"

- s=10
- Compression: 1.8%

Some approximations

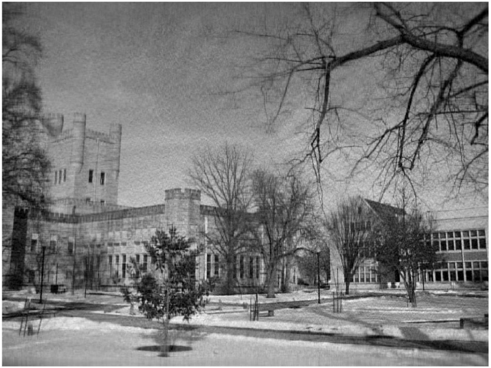
Let's see what our truncated matrix "looks like"

- s=25
- Compression: 4.5%

Some approximations

Let's see what our truncated matrix "looks like"

- s=100
- Compression: 18%



Problems in this approach

This technique has some problems

Problems in this approach

This technique has some problems

- ad hoc method for determining when we're done

Problems in this approach

This technique has some problems

- ad hoc method for determining when we're done
- requires we keep track of singular values and basis elements ($1 + r + c$ pieces of data for each singular value!)

Problems in this approach

This technique has some problems

- ad hoc method for determining when we're done
- requires we keep track of singular values and basis elements ($1 + r + c$ pieces of data for each singular value!)

Would be nice to find something more systematic

Problems in this approach

This technique has some problems

- ad hoc method for determining when we're done
- requires we keep track of singular values and basis elements ($1 + r + c$ pieces of data for each singular value!)

Would be nice to find something more systematic

- controlling quality of compressed image

Problems in this approach

This technique has some problems

- ad hoc method for determining when we're done
- requires we keep track of singular values and basis elements ($1 + r + c$ pieces of data for each singular value!)

Would be nice to find something more systematic

- controlling quality of compressed image
- doesn't require us to keep track of a basis

Problems in this approach

This technique has some problems

- ad hoc method for determining when we're done
- requires we keep track of singular values and basis elements ($1 + r + c$ pieces of data for each singular value!)

Would be nice to find something more systematic

- controlling quality of compressed image
- doesn't require us to keep track of a basis
- takes advantage of properties of images

The “usual” way of thinking about a matrix

Typically we think of a matrix in terms of its entries.

$$A = \sum_{i,j} a_{ij} E(i,j)$$

where $E(i,j)$ is the matrix with a 1 in the ith row, jth column.

The "usual" way of thinking about a matrix

Typically we think of a matrix in terms of its entries.

$$A = \sum_{i,j} a_{ij} E(i,j)$$

where $E(i,j)$ is the matrix with a 1 in the ith row, jth column.

- Each $E(i,j)$ solely responsible for its local behavior.

The "usual" way of thinking about a matrix

Typically we think of a matrix in terms of its entries.

$$A = \sum_{i,j} a_{ij} E(i,j)$$

where $E(i,j)$ is the matrix with a 1 in the ith row, jth column.

- Each $E(i,j)$ solely responsible for its local behavior.
- Deleting an a_{ij} completely wipes out pixel info.

Rewriting the matrix

What if we chose a different basis for $n \times n$ matrices?

$$A = \sum_{i,j} c_{ij} B(i,j)$$

Rewriting the matrix

What if we chose a different basis for $n \times n$ matrices?

$$A = \sum_{i,j} c_{ij} B(i,j)$$

Would be nice if

Rewriting the matrix

What if we chose a different basis for $n \times n$ matrices?

$$A = \sum_{i,j} c_{ij} B(i,j)$$

Would be nice if

- basis weren't so "local"

Rewriting the matrix

What if we chose a different basis for $n \times n$ matrices?

$$A = \sum_{i,j} c_{ij} B(i,j)$$

Would be nice if

- basis weren't so "local"
- deleting $c_{i,j}$ has gradual (though global) effect

A potential basis

We'll choose an basis of $\mathbb{R}^n$ from Fourier series

$$\vec{f}_i = \alpha_i \left\{ \cos\left[\frac{\pi}{n}\left(\frac{2j+1}{2}\right) i\right] \right\}_{j=0}^{n-1}$$

A potential basis

We'll choose an basis of $\mathbb{R}^n$ from Fourier series

$$\vec{f}_i = \alpha_i \left\{ \cos\left[\frac{\pi}{n} \left(\frac{2j+1}{2} \right) i \right] \right\}_{j=0}^{n-1}$$

We'll simply change basis to $\mathcal{B} = \{\vec{f}_0, \cdots, \vec{f}_{n-1}\}$

A potential basis

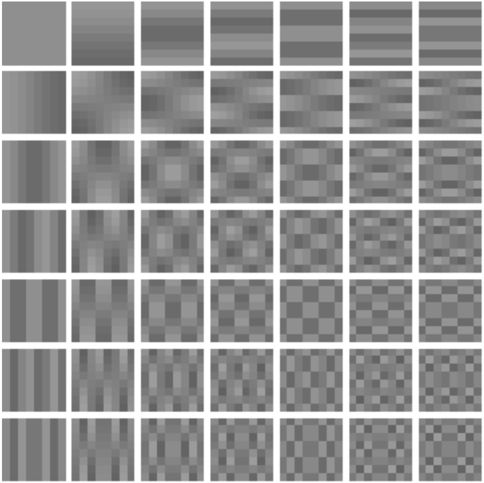
We'll choose an basis of $\mathbb{R}^n$ from Fourier series

$$\vec{f}_i = \alpha_i \left\{ \cos\left[\frac{\pi}{n}\left(\frac{2j+1}{2}\right) i\right]\right\}_{j=0}^{n-1}$$

We'll simply change basis to $\mathcal{B} = \{\vec{f}_0, \cdots, \vec{f}_{n-1}\}$

$$B(i,j) = \begin{pmatrix} \vec{f}_0 \\ \hline \vdots \\ \hline \vec{f}_{n-1} \end{pmatrix} E(i,j) \left(\vec{f}_0 \,\middle|\, \cdots \,\middle|\, \vec{f}_{n-1} \right)$$

Seeing the new basis ($n = 8$)



Computing the $\mathcal{B}$-matrix

Compute coefficients $c_{i,j}$ such that $A = \sum_{i,j} c_{i,j} B(i,j)$ by

Computing the $\mathcal{B}$-matrix

Compute coefficients $c_{i,j}$ such that $A = \sum_{i,j} c_{i,j} B(i,j)$ by

$$\left(\begin{array}{c} \overrightarrow{f}_0 \\ \hline \vdots \\ \hline \overrightarrow{f}_{n-1} \end{array}\right) A \left(\begin{array}{c|c|c} \overrightarrow{f}_0 & \cdots & \overrightarrow{f}_{n-1} \end{array}\right)$$

Why we chose this basis

This is a good basis because

Why we chose this basis

This is a good basis because

- Human eye only sees in "steps"; if distinguishable step size for $B(i,j)$ is $q_{i,j}$, then

$$\sum_{i,j} c_{i,j} B(i,j) \approx \sum_{i,j} q_{i,j} \left[\frac{c_{i,j}}{q_{i,j}} \right] B(i,j)$$

Why we chose this basis

This is a good basis because

- Human eye only sees in "steps"; if distinguishable step size for $B(i,j)$ is $q_{i,j}$, then

$$\sum_{i,j} c_{i,j} B(i,j) \approx \sum_{i,j} q_{i,j} \left[\frac{c_{i,j}}{q_{i,j}} \right] B(i,j)$$

- Images are "smooth" (small "high frequency" components)

Why we chose this basis

This is a good basis because

- Human eye only sees in "steps"; if distinguishable step size for $B(i,j)$ is $q_{i,j}$, then

$$\sum_{i,j} c_{i,j} B(i,j) \approx \sum_{i,j} q_{i,j} \left[\frac{c_{i,j}}{q_{i,j}} \right] B(i,j)$$

- Images are "smooth" (small "high frequency" components)

$$\sum_{i,j} c_{i,j} B(i,j) \approx \sum_{\text{small } i,j} q_{i,j} \left[\frac{c_{i,j}}{q_{i,j}} \right] B(i,j)$$

How JPEG compression works

Here's (roughly) how JPEG compression uses this idea:

How JPEG compression works

Here's (roughly) how JPEG compression uses this idea:

- Split image into 8×8 blocks

How JPEG compression works

Here's (roughly) how JPEG compression uses this idea:

- Split image into 8×8 blocks
- Change coordinates for each 8×8 submatrix

How JPEG compression works

Here's (roughly) how JPEG compression uses this idea:

- Split image into 8×8 blocks
- Change coordinates for each 8×8 submatrix
- Quantize

How JPEG compression works

Here's (roughly) how JPEG compression uses this idea:

- Split image into 8×8 blocks
- Change coordinates for each 8×8 submatrix
- Quantize

Then decompression is

How JPEG compression works

Here's (roughly) how JPEG compression uses this idea:

- Split image into 8×8 blocks
- Change coordinates for each 8×8 submatrix
- Quantize

Then decompression is

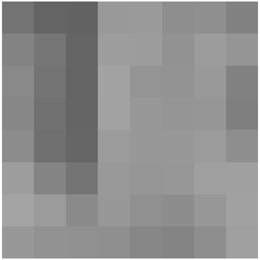
- De-quantize
- Change back to standard coordinates
- Reassemble the 8×8 blocks

Working through an example

Extract an 8×8 block

Working through an example

Extract an 8×8 block



$$\longrightarrow \begin{pmatrix} 115 & 100 & 98 & 153 & 154 & 142 & 143 & 130 \\ 131 & 118 & 101 & 157 & 156 & 146 & 156 & 149 \\ 137 & 115 & 100 & 163 & 148 & 147 & 153 & 130 \\ 135 & 113 & 101 & 163 & 152 & 149 & 150 & 127 \\ 140 & 111 & 102 & 156 & 152 & 152 & 155 & 142 \\ 157 & 132 & 116 & 153 & 150 & 151 & 159 & 160 \\ 164 & 155 & 138 & 152 & 144 & 141 & 151 & 161 \\ 152 & 146 & 145 & 143 & 135 & 132 & 142 & 159 \end{pmatrix}$$

Working through an example

Change to $\mathcal{B}$-version

Working through an example

Change to $\mathcal{B}$-version

$$\begin{pmatrix} 115 & 100 & 98 & 153 & 154 & 142 & 143 & 130 \\ 131 & 118 & 101 & 157 & 156 & 146 & 156 & 149 \\ 137 & 115 & 100 & 163 & 148 & 147 & 153 & 130 \\ 135 & 113 & 101 & 163 & 152 & 149 & 150 & 127 \\ 140 & 111 & 102 & 156 & 152 & 152 & 155 & 142 \\ 157 & 132 & 116 & 153 & 150 & 151 & 159 & 160 \\ 164 & 155 & 138 & 152 & 144 & 141 & 151 & 161 \\ 152 & 146 & 145 & 143 & 135 & 132 & 142 & 159 \end{pmatrix} \longrightarrow$$

Working through an example

Change to $\mathcal{B}$-version

$$\begin{pmatrix} 115 & 100 & 98 & 153 & 154 & 142 & 143 & 130 \\ 131 & 118 & 101 & 157 & 156 & 146 & 156 & 149 \\ 137 & 115 & 100 & 163 & 148 & 147 & 153 & 130 \\ 135 & 113 & 101 & 163 & 152 & 149 & 150 & 127 \\ 140 & 111 & 102 & 156 & 152 & 152 & 155 & 142 \\ 157 & 132 & 116 & 153 & 150 & 151 & 159 & 160 \\ 164 & 155 & 138 & 152 & 144 & 141 & 151 & 161 \\ 152 & 146 & 145 & 143 & 135 & 132 & 142 & 159 \end{pmatrix} \longrightarrow \begin{pmatrix} 1112 & -61 & -14 & 44 & 57 & 34 & -32 & -26 \\ -43 & -36 & -43 & 25 & 13 & 12 & -15 & -8 \\ 2 & 12 & 12 & -26 & -8 & -16 & 7 & 10 \\ 2 & -14 & 1 & 7 & 6 & -3 & 1 & 2 \\ -25 & -4 & -16 & 0 & -1 & 2 & 5 & 3 \\ -2 & 12 & -6 & 1 & -3 & 2 & -1 & -2 \\ -9 & -1 & -2 & 3 & 0 & 5 & 2 & 0 \\ -4 & 2 & -2 & 1 & -1 & 3 & 1 & -1 \end{pmatrix}$$

Working through an example

Quantize

Working through an example

Quantize

$$\left[\frac{\begin{pmatrix} 1112 & -61 & -14 & 44 & 57 & 34 & -32 & -26 \\ -43 & -36 & -43 & 25 & 13 & 12 & -15 & -8 \\ 2 & 12 & 12 & -26 & -8 & -16 & 7 & 10 \\ 2 & -14 & 1 & 7 & 6 & -3 & 1 & 2 \\ -25 & -4 & -16 & 0 & -1 & 2 & 5 & 3 \\ -2 & 12 & -6 & 1 & -3 & 2 & -1 & -2 \\ -9 & -1 & -2 & 3 & 0 & 5 & 2 & 0 \\ -4 & 2 & -2 & 1 & -1 & 3 & 1 & -1 \end{pmatrix}}{\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 71 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}}\right] \longrightarrow$$

Working through an example

Quantize

$$
\left[\frac{\begin{pmatrix} 1112 & -61 & -14 & 44 & 57 & 34 & -32 & -26 \\ -43 & -36 & -43 & 25 & 13 & 12 & -15 & -8 \\ 2 & 12 & 12 & -26 & -8 & -16 & 7 & 10 \\ 2 & -14 & 1 & 7 & 6 & -3 & 1 & 2 \\ -25 & -4 & -16 & 0 & -1 & 2 & 5 & 3 \\ -2 & 12 & -6 & 1 & -3 & 2 & -1 & -2 \\ -9 & -1 & -2 & 3 & 0 & 5 & 2 & 0 \\ -4 & 2 & -2 & 1 & -1 & 3 & 1 & -1 \end{pmatrix}}{\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 71 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}}\right] \longrightarrow \begin{pmatrix} 70 & -6 & -1 & 3 & 2 & 1 & -1 & 0 \\ -4 & -3 & -3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
$$

Working through an example

Quantize

$$\left[\frac{\begin{pmatrix} 1112 & -61 & -14 & 44 & 57 & 34 & -32 & -26 \\ -43 & -36 & -43 & 25 & 13 & 12 & -15 & -8 \\ 2 & 12 & 12 & -26 & -8 & -16 & 7 & 10 \\ 2 & -14 & 1 & 7 & 6 & -3 & 1 & 2 \\ -25 & -4 & -16 & 0 & -1 & 2 & 5 & 3 \\ -2 & 12 & -6 & 1 & -3 & 2 & -1 & -2 \\ -9 & -1 & -2 & 3 & 0 & 5 & 2 & 0 \\ -4 & 2 & -2 & 1 & -1 & 3 & 1 & -1 \end{pmatrix}}{\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 71 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}}\right] \longrightarrow \begin{pmatrix} 70 & -6 & -1 & 3 & 2 & 1 & -1 & 0 \\ -4 & -3 & -3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

We're down to 16 pieces of information!

Reconstituting our image

Here's the result of reversing this process:

Reconstituting our image

Here's the result of reversing this process:

Original

$$\begin{pmatrix} 115 & 100 & 98 & 153 & 154 & 142 & 143 & 130 \\ 131 & 118 & 101 & 157 & 156 & 146 & 156 & 149 \\ 137 & 115 & 100 & 163 & 148 & 147 & 153 & 130 \\ 135 & 113 & 101 & 163 & 152 & 149 & 150 & 127 \\ 140 & 111 & 102 & 156 & 152 & 152 & 155 & 142 \\ 157 & 132 & 116 & 153 & 150 & 151 & 159 & 160 \\ 164 & 155 & 138 & 152 & 144 & 141 & 151 & 161 \\ 152 & 146 & 145 & 143 & 135 & 132 & 142 & 159 \end{pmatrix}$$

Compressed

$$\begin{pmatrix} 114 & 103 & 101 & 144 & 149 & 136 & 154 & 135 \\ 125 & 111 & 106 & 150 & 156 & 142 & 57 & 134 \\ 133 & 115 & 107 & 151 & 160 & 146 & 157 & 130 \\ 134 & 114 & 104 & 147 & 158 & 146 & 157 & 129 \\ 139 & 118 & 106 & 148 & 156 & 146 & 163 & 139 \\ 151 & 132 & 118 & 153 & 155 & 145 & 169 & 153 \\ 162 & 144 & 129 & 155 & 147 & 135 & 165 & 158 \\ 166 & 150 & 132 & 152 & 137 & 122 & 157 & 154 \end{pmatrix}$$

Reconstituting our image

Here's the result of reversing this process:

Original

115	100	98	153	154	142	143	130
131	118	101	157	156	146	156	149
137	115	100	163	148	147	153	130
135	113	101	163	152	149	150	127
140	111	102	156	152	152	155	142
157	132	116	153	150	151	159	160
164	155	138	152	144	141	151	161
152	146	145	143	135	132	142	159

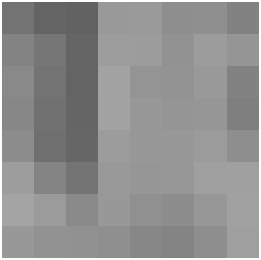
Compressed

114	103	101	144	149	136	154	135
125	111	106	150	156	142	57	134
133	115	107	151	160	146	157	130
134	114	104	147	158	146	157	129
139	118	106	148	156	146	163	139
151	132	118	153	155	145	169	153
162	144	129	155	147	135	165	158
166	150	132	152	137	122	157	154

- Average difference $= 5.73$
- Std Dev $= 4.22$

Seeing is believing

Original



Compressed

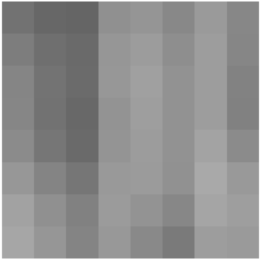


Image Processing

We can use these ideas to do some image processing as well

Image Processing

We can use these ideas to do some image processing as well

- "smooth" part of the image comes from low frequency Fourier coefficients

Image Processing

We can use these ideas to do some image processing as well

- "smooth" part of the image comes from low frequency Fourier coefficients
- "edges" come from the high frequency Fourier coefficients

Image Processing

We can use these ideas to do some image processing as well

- "smooth" part of the image comes from low frequency Fourier coefficients
- "edges" come from the high frequency Fourier coefficients

Note: Here I won't split the image into 8×8 blocks – I want all the information about the image simultaneously

Image Processing

Smooth Part: $B(i,j)$ components for small j, small i

Image Processing

Smooth Part: $B(i,j)$ components for small j, small i

Image Processing

Horizontal Edges: $B(i, j)$ components for small j, large i

Image Processing

Horizontal Edges: $B(i, j)$ components for small j, large i

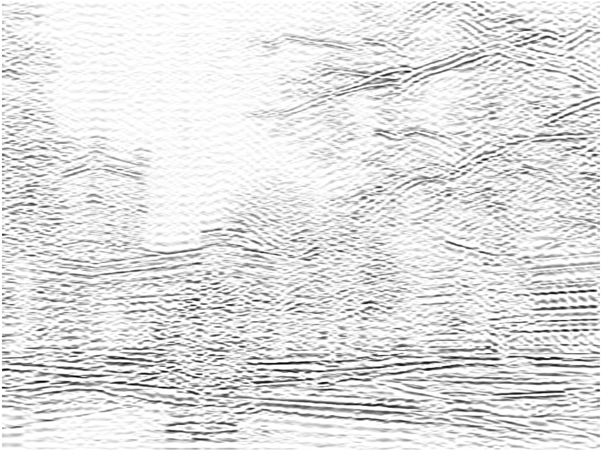


Image Processing

Vertical Edges: $B(i,j)$ components for small i, large j

Image Processing

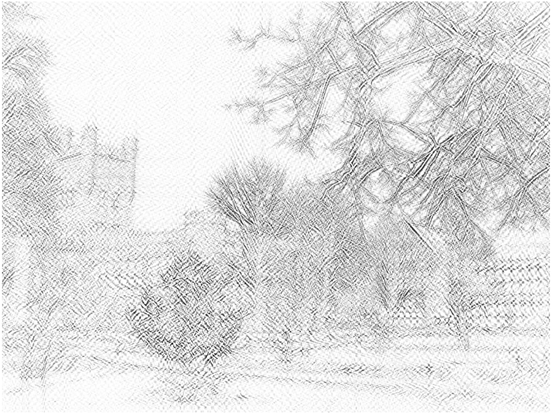
Vertical Edges: $B(i,j)$ components for small i, large j

Image Processing

Scattered Edges: $B(i,j)$ components for large i, large j

Image Processing

Scattered Edges: $B(i,j)$ components for large i, large j



How MP3 compression works

Similar ideas are used to compress music into mp3's

How MP3 compression works

Similar ideas are used to compress music into mp3's

- Sample an audio source

How MP3 compression works

Similar ideas are used to compress music into mp3's

- Sample an audio source
    - Humans hear between 20Hz and 20,000Hz

How MP3 compression works

Similar ideas are used to compress music into mp3's

- Sample an audio source
  - Humans hear between 20Hz and 20,000Hz
  - Sample at 41,000Hz

How MP3 compression works

Similar ideas are used to compress music into mp3's

- Sample an audio source
  - Humans hear between 20Hz and 20,000Hz
  - Sample at 41,000Hz
  - 16bits per sample and 2 channels means 1.3 million bits per second

How MP3 compression works

Similar ideas are used to compress music into mp3's

- Sample an audio source
  - Humans hear between 20Hz and 20,000Hz
  - Sample at 41,000Hz
  - 16bits per sample and 2 channels means 1.3 million bits per second
- Break sample into smaller blocks

How MP3 compression works

Similar ideas are used to compress music into mp3's

- Sample an audio source
    - Humans hear between 20Hz and 20,000Hz
    - Sample at 41,000Hz
    - 16bits per sample and 2 channels means 1.3 million bits per second
- Break sample into smaller blocks
- Express these blocks in terms of $\mathcal{B}$-coordinates

How MP3 compression works

Similar ideas are used to compress music into mp3's

- Sample an audio source
  - Humans hear between 20Hz and 20,000Hz
  - Sample at 41,000Hz
  - 16bits per sample and 2 channels means 1.3 million bits per second
- Break sample into smaller blocks
- Express these blocks in terms of $\mathcal{B}$-coordinates
- Filter out "unnecessary" data using psychoacoustics

Psychoacoustics

- Simultaneous Masking - If two tones with near frequencies are played at the same time, your brain only hears the louder one

Psychoacoustics

- Simultaneous Masking - If two tones with near frequencies are played at the same time, your brain only hears the louder one
- Temporal Masking - Some weak sounds aren't heard if played right after (or right before!) a louder sound

Psychoacoustics

- Simultaneous Masking - If two tones with near frequencies are played at the same time, your brain only hears the louder one
- Temporal Masking - Some weak sounds aren't heard if played right after (or right before!) a louder sound
- Hass effect - If the same tone hits one ear just before another, then your brain perceives it as coming only from the first direction

Thanks!

Thank you!