Homework 4

posted on Brightspace
due TODAY at 11:59pm

`Homework4.pdf` (written description)
`Homework4.ipynb` (notebook to use for your solution)
`difdata.csv`

download from Brightspace

```
VectorsMatricesLinearAlgebra.ipynb
SVD.ipynb
ControlFlow.ipynb
Functions.ipynb
```

Homework5

also posted on Brightspace
due next Wed (Oct 12) in class

`Homework5.pdf` (written description)
`Homework5.ipynb`
`BoysBW.jpg`

Homework5

Q1. program (using a function*) matrix multiplication using for loops and compare to built-in matrix multiplication
* we'll talk about creating functions soon

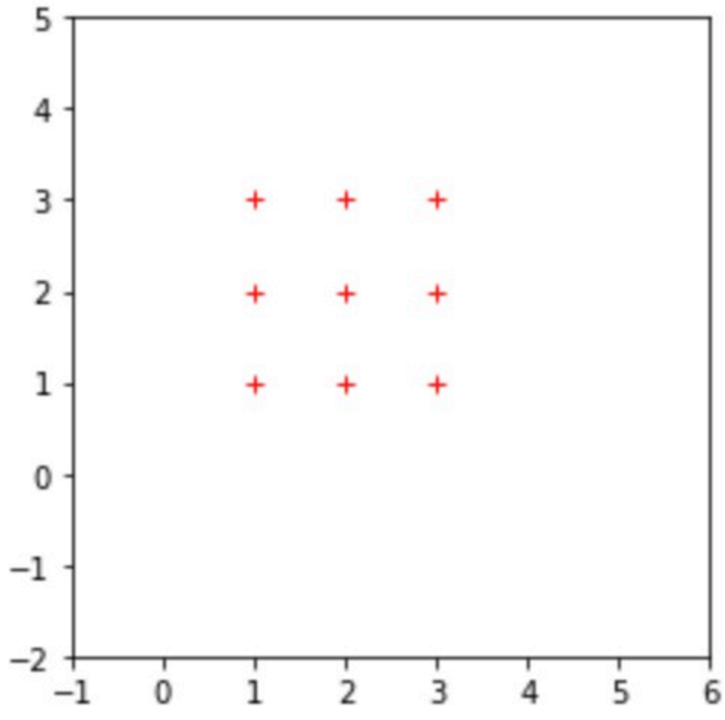
Q2. explore commutativity and associativity of elementwise multiplication and matrix multiplication

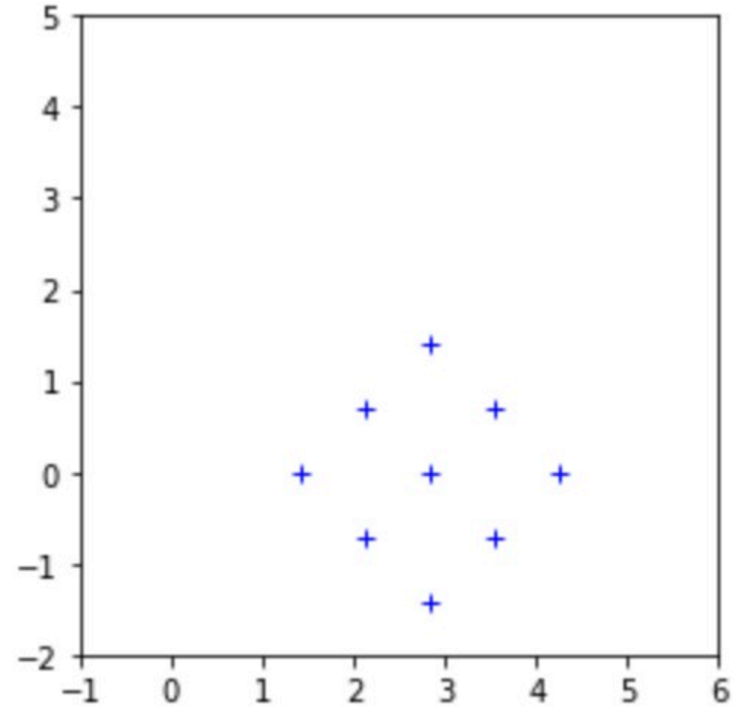
Q3. compare the inverse of a matrix with the elementwise matrix of inverses (1/x)

matrices as transformations

```
theta = 45
R = np.array(
       [[m.cos(m.radians(theta)), -m.sin(m.radians(theta))],
        [m.sin(m.radians(theta)),  m.cos(m.radians(theta))]])

B = A @ R
plt.plot(A[:,0],A[:,1],'r+')
plt.plot(B[:,0],B[:,1],'b+')
```

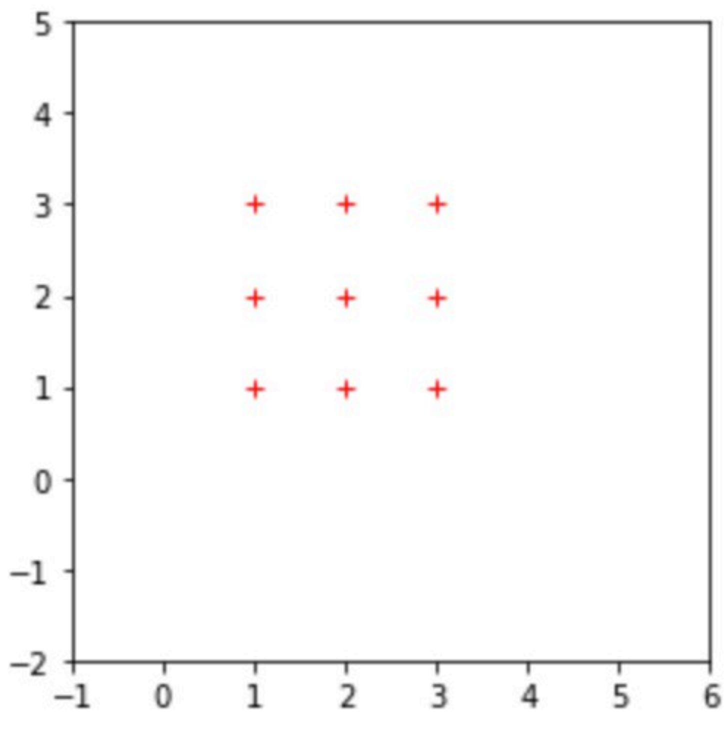


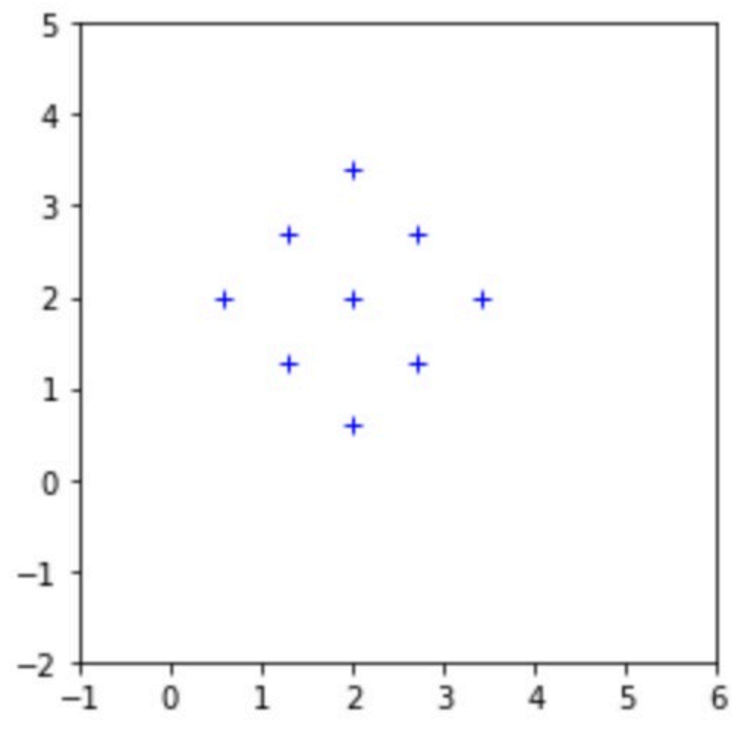


matrices as transformations

```
theta = 45
R = np.array(
       [[m.cos(m.radians(theta)), -m.sin(m.radians(theta))],
        [m.sin(m.radians(theta)),  m.cos(m.radians(theta))]])

C = np.mean(A, axis=0);  B = (A-C) @ R + C
plt.plot(A[:,0],A[:,1],'r+')
plt.plot(B[:,0],B[:,1],'b+')
```

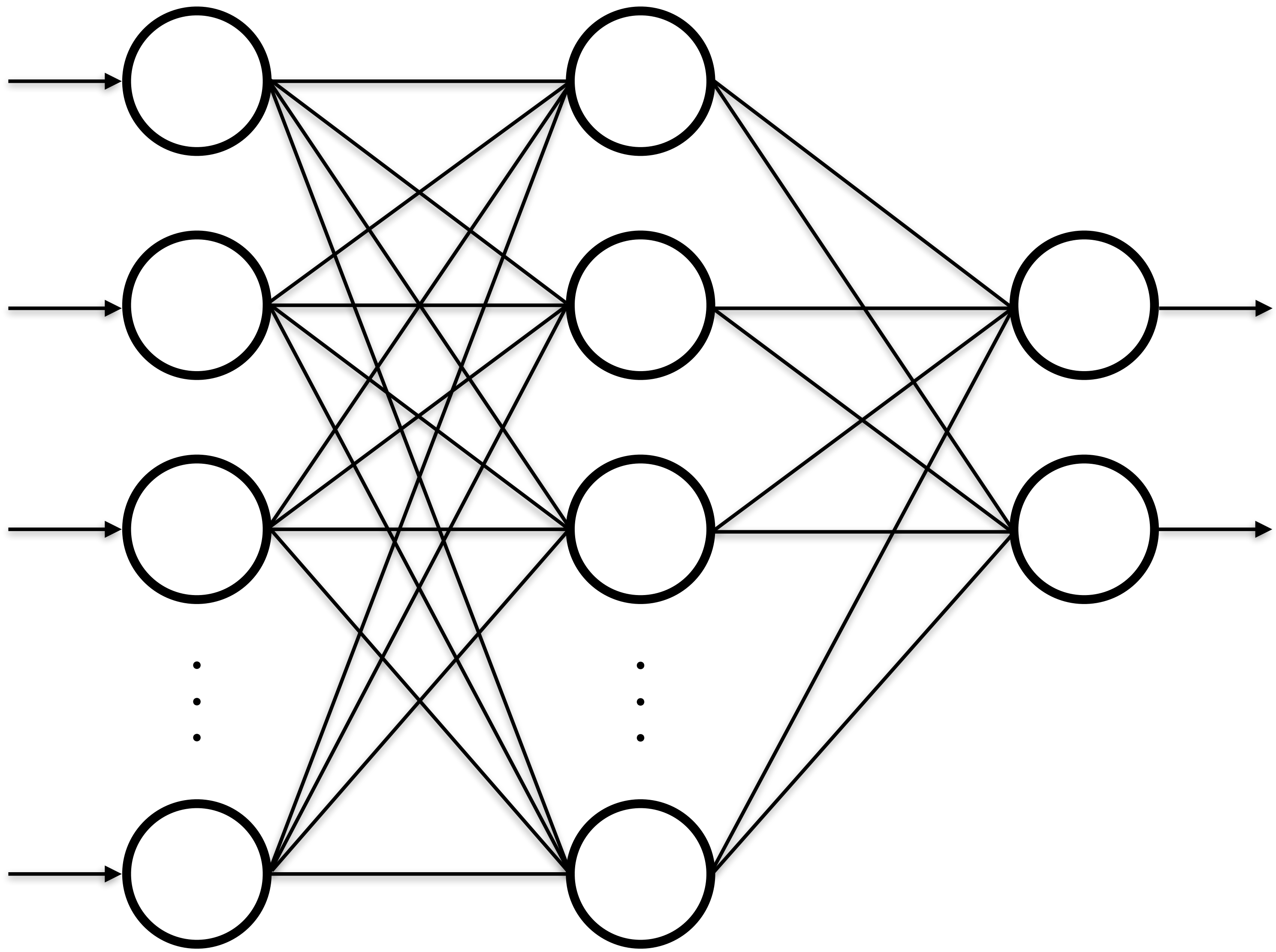




matrices as transformations

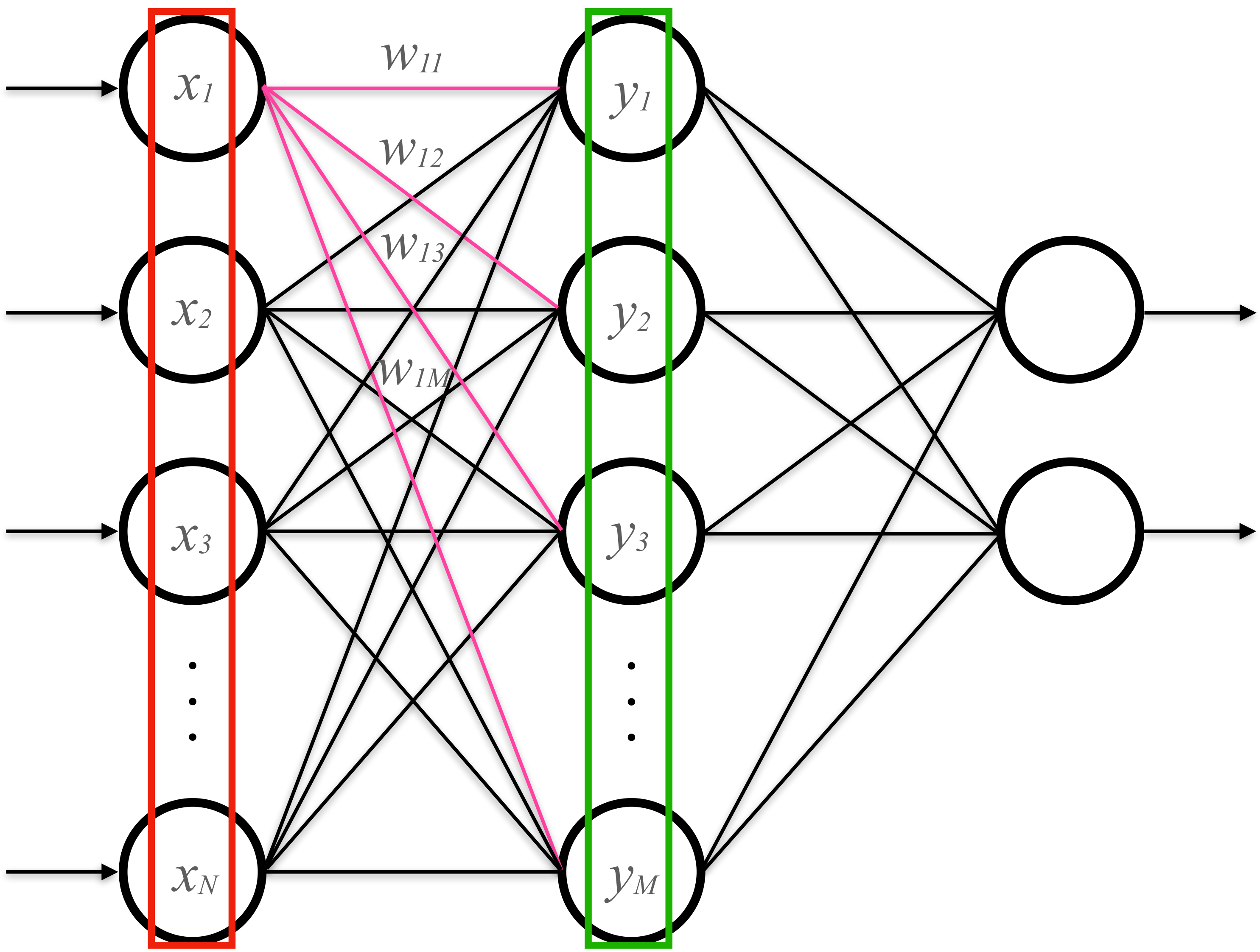
Linear transformations and matrices
Chapter 3 Essence of linear algebra (3Blue1Brown)
https://www.youtube.com/watch?v=kYB8IZa5AuE

matrices in neural networks



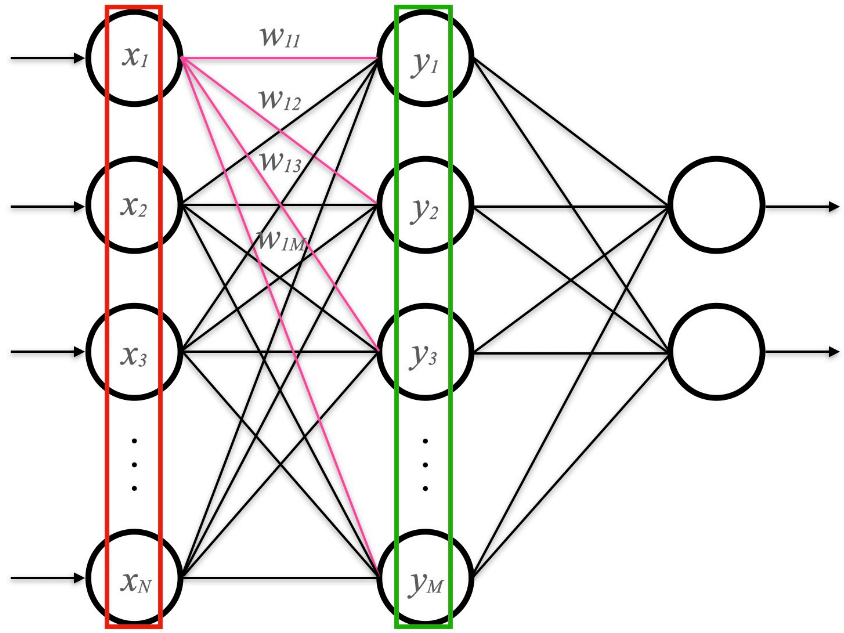
"Tensor" in Tensorflow is a multidimensional matrix

matrices in neural networks



"Tensor" in Tensorflow is a multidimensional matrix

matrices in neural networks



$$[x_1 \quad x_2 \quad \cdots \quad x_N] \times \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NM} \end{bmatrix} = [y_1 \quad y_2 \quad \cdots \quad y_M]$$

for a linear network

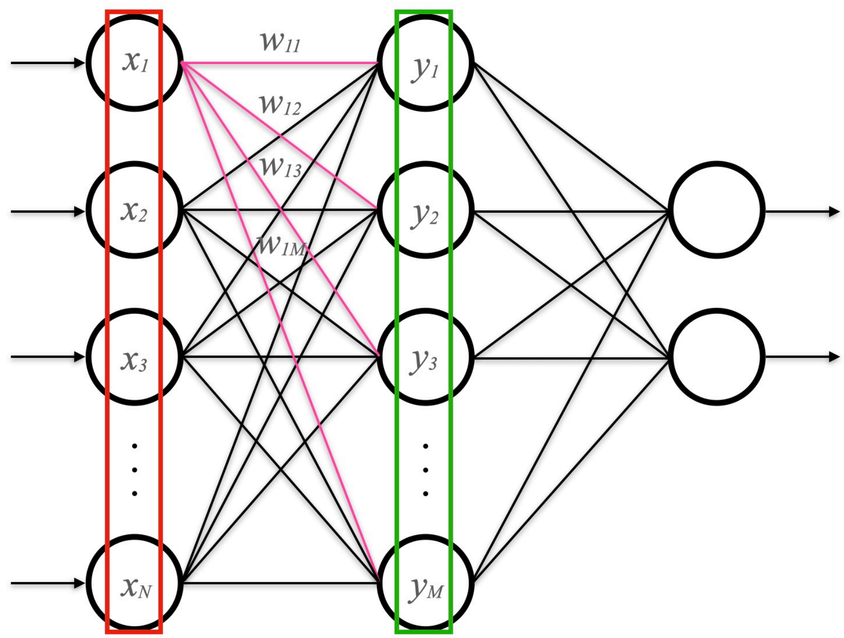
1 X N matrix

N X M matrix

1 X M matrix

"Tensor" in Tensorflow is a multidimensional matrix

matrices in neural networks



$$f\left([x_1 \quad x_2 \quad \cdots \quad x_N] \times \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1M} \\ w_{21} & w_{22} & \cdots & w_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & w_{NM} \end{bmatrix} \right) = [y_1 \quad y_2 \quad \cdots \quad y_M]$$

for a nonlinear network

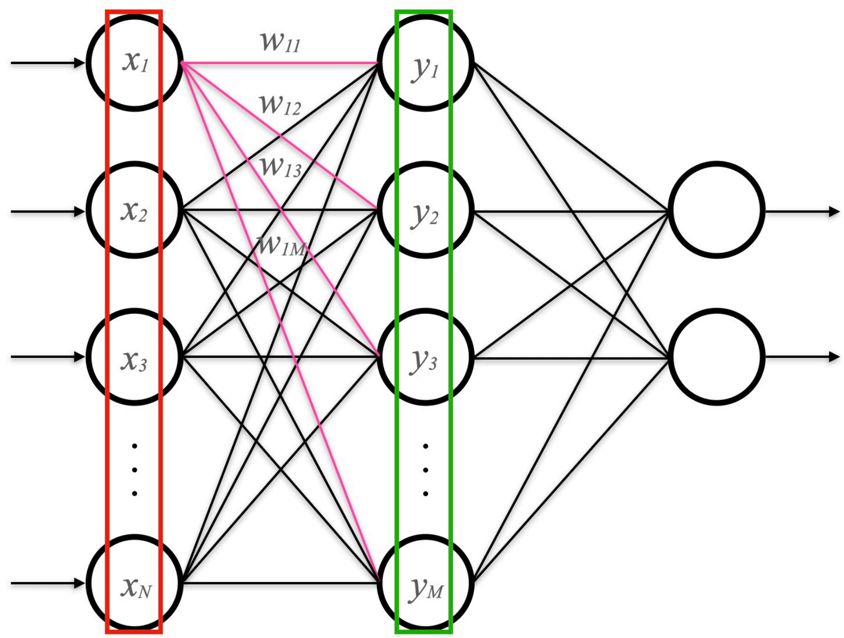
1 X N matrix

N X M matrix

1 X M matrix

"Tensor" in Tensorflow is a multidimensional matrix

matrices in neural networks



matrices also transform between dimensions

e.g., from N-dimensional to M-dimensional

e.g., from M-dimensional to 2-dimensional

brains (neural networks) perform dimensionality reduction, e.g., from 100-million-dimensional retinal image to a lower-dimensional perceptual/psychological representation

Matrix Decomposition

Matrix decomposition (or matrix factorization) turns a matrix into the product of other matrices. Linear algebra defines numerous matrix decomposition methods (with numerous uses): e.g., LU Decomposition, Cholesky Factorization, QR Decomposition, Eigendecomposition, Singular Value Decomposition (SVD), and more

They can be used to understand what transformation a matrix performs (e.g., a complex transformation is decomposed into its parts), define numerically stable operations (e.g., the inverse of a decomposed matrix is more numerically stable), perform dimensionality reduction if the matrix contains data (e.g., PCA)

Singular Value Decomposition (SVD)

we'll just introduce SVD now in enough detail to complete **Homework 5** - we might discuss SVD more (what it does, what it means) again later in the semester

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$$

matrix you
are given

matrices discovered by SVD that
when multiplied together equal $\mathbf{A}$
(we will not discuss how)

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$$

n x m   n x n   n x m   m x m

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\underbrace{\mathbf{V}^{\top}}$$

this is defined as V^{T}
for conceptual reasons

could have SVD defined as
$$\mathbf{X} = \mathbf{A}\ \mathbf{B}\ \mathbf{C}$$

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\top}$$

if $n = m$

$$\underbrace{\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nn} \end{bmatrix}}_{\mathbf{U}} \times \underbrace{\begin{bmatrix} s_{11} & 0 & \cdots & 0 \\ 0 & s_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_{nn} \end{bmatrix}}_{\boldsymbol{\Sigma}} \times \underbrace{\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix}}_{\mathbf{V}^{\top}}$$

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$$

if $n > m$

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ x_{31} & x_{32} & \cdots & x_{3m} \\ x_{41} & x_{42} & \cdots & x_{4m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & \cdots & u_{1n} \\ u_{21} & u_{22} & u_{23} & u_{24} & \cdots & u_{2n} \\ u_{31} & u_{42} & u_{33} & u_{34} & \cdots & u_{3n} \\ u_{31} & u_{42} & u_{43} & u_{44} & \cdots & u_{4n} \\ \vdots & \vdots & & & \ddots & \vdots \\ u_{n1} & u_{n2} & u_{n3} & u_{n4} & \cdots & u_{nn} \end{bmatrix} \times \begin{bmatrix} s_{11} & 0 & \cdots & 0 \\ 0 & s_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_{mm} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{m1} & v_{m2} & \cdots & v_{mm} \end{bmatrix}$$

$$\mathbf{X} \qquad \mathbf{U} \qquad \mathbf{\Sigma} \qquad \mathbf{V}^{\top}$$

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

if $n < m$

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & x_{24} & \cdots & x_{2m} \\ \vdots & \vdots & & & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & x_{n4} & \cdots & x_{nm} \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nn} \end{bmatrix} \times \begin{bmatrix} s_{11} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & s_{22} & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & \cdots & 0 \\ 0 & 0 & \cdots & s_{nn} & 0 & \cdots & 0 \end{bmatrix} \times \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & v_{23} & v_{12} & \cdots & v_{2m} \\ v_{31} & v_{32} & v_{33} & v_{12} & \cdots & v_{3m} \\ v_{41} & v_{42} & v_{43} & v_{12} & \cdots & v_{4m} \\ \vdots & \vdots & & & \ddots & \vdots \\ v_{m1} & v_{m2} & v_{m3} & v_{12} & \cdots & v_{mm} \end{bmatrix}$$

$$\mathbf{X} \qquad \mathbf{U} \qquad \boldsymbol{\Sigma} \qquad \mathbf{V}^\top$$

SVD.ipynb

some useful properties of SVD

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

$$\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}$$

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$

$$\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}$$

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top$$

not deriving

$(\mathbf{V}^\top)^\top$

transpose is trivially easy

inverse of diagonal matrix is trivially easy

transpose is trivially easy

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U\Sigma V}^\top$

$$\mathbf{UU}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{VV}^\top = \mathbf{I}$$

$$\mathbf{X} = \mathbf{U\Sigma V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V\Sigma}^{-1}\mathbf{U}^\top$$

$$\mathbf{XX}^{-1} = \mathbf{I}$$

show that it's true

inverse of square matrix $\mathbf{X}$ using SVD

given: $$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

$$\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}$$

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top$$

$$(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top)(\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top) = \mathbf{I}$$

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$

$$\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}$$

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top$$

$$\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top = \mathbf{I}$$

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$

$$\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \boxed{\mathbf{V}\mathbf{V}^\top = \mathbf{I}}$$

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top$$

$$\mathbf{U}\boldsymbol{\Sigma}\boxed{\mathbf{V}^\top\mathbf{V}}\boldsymbol{\Sigma}^{-1}\mathbf{U}^\top = \mathbf{I}$$

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U\Sigma V}^\top$

$$\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}$$

$$\mathbf{X} = \mathbf{U\Sigma V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V\Sigma}^{-1}\mathbf{U}^\top$$

$$\mathbf{U\Sigma I\Sigma}^{-1}\mathbf{U}^\top = \mathbf{I}$$

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U\Sigma V}^\top$

$$\mathbf{UU}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{VV}^\top = \mathbf{I}$$

$$\mathbf{X} = \mathbf{U\Sigma V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V\Sigma}^{-1}\mathbf{U}^\top$$

$$\mathbf{U\Sigma\Sigma}^{-1}\mathbf{U}^\top = \mathbf{I}$$

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\top}$

$$\mathbf{U}\mathbf{U}^{\top} = \mathbf{U}^{\top}\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^{\top}(\mathbf{V}^{\top})^{\top} = \mathbf{V}^{\top}\mathbf{V} = \mathbf{V}\mathbf{V}^{\top} = \mathbf{I}$$

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\top}$$

$$\mathbf{X}^{-1} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{\top}$$

$$\mathbf{U}\boxed{\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}}\mathbf{U}^{\top} = \mathbf{I}$$

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U\Sigma V}^\top$

$$\mathbf{UU}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{VV}^\top = \mathbf{I}$$

$$\mathbf{X} = \mathbf{U\Sigma V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V\Sigma}^{-1}\mathbf{U}^\top$$

$$\mathbf{UIU}^\top = \mathbf{I}$$

inverse of square matrix $\mathbf{X}$ using SVD

given: $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$

$$\boxed{\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}}$$

$$\mathbf{V}^\top(\mathbf{V}^\top)^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{V}\mathbf{V}^\top = \mathbf{I}$$

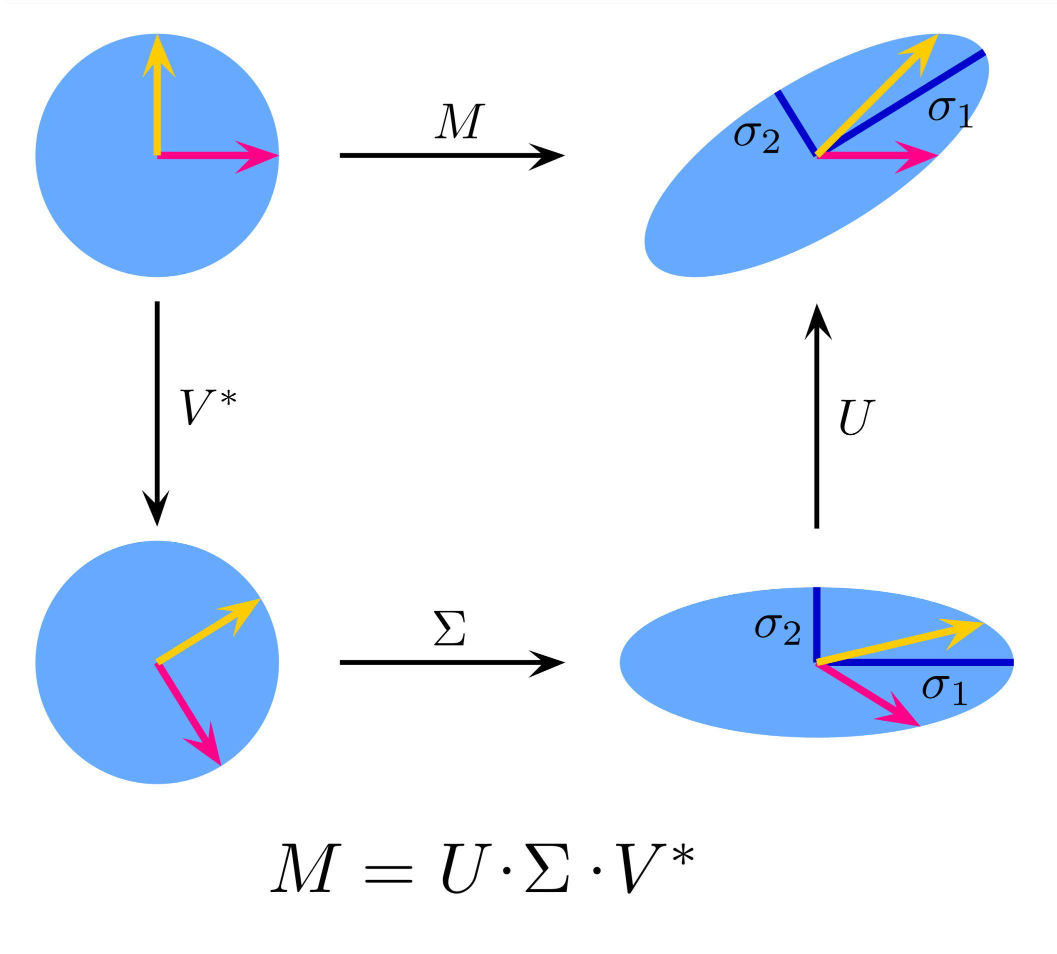
$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

$$\mathbf{X}^{-1} = \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top$$

$$\boxed{\mathbf{U}\mathbf{U}^\top} = \mathbf{I}$$

SVD as a series of matrix transformations

`SVD.ipynb`



https://en.wikipedia.org/wiki/Singular_value_decomposition

Homework5

Q4. read in an image (from a file) as a matrix, perform SVD on it and reconstruct the image and show it; I ask you to produce a "reduced" SVD representation using a particular algorithm described in the assignment

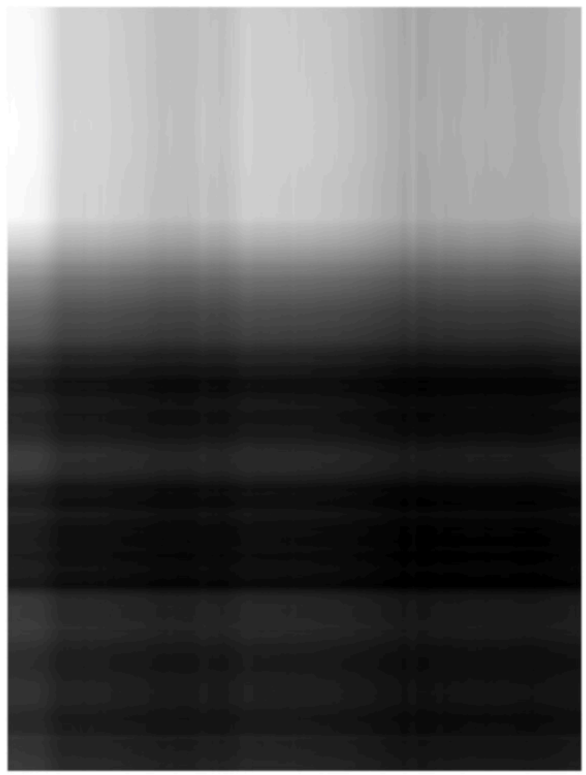
while this question uses an image (and the reduced representation algorithm can be used for image compression), I am asking you to do this to see what SVD is doing and how the order of the rows and columns in the SVD represent the "importance" of information represented in the matrix

Homework5

example using another image

original
450 x 338

r = 1
99.5% compression



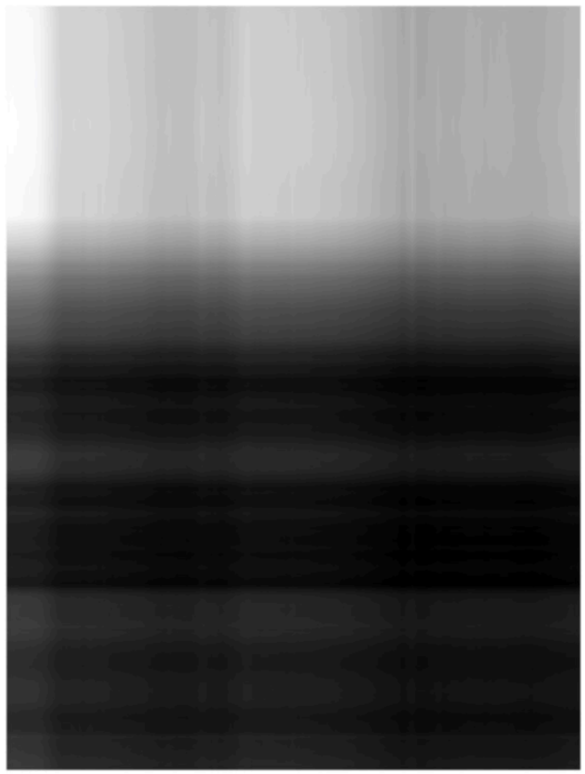
r = 10
94.8% compression

Homework5

example using another image

original
450 x 338

r = 1
99.5% compression



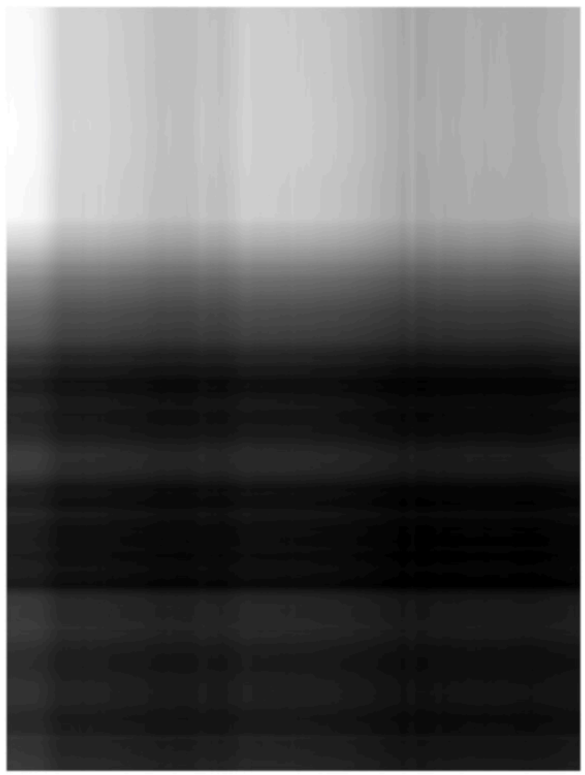
r = 20
89.6% compression

Homework5

example using another image

original
450 x 338

r = 1
99.5% compression



r = 50
74.0% compression

much of this we have seen before, and will be a quick review, but there are a few features we have not touched on yet

Control Flow in Python

`ControlFlow.ipynb`

A Whirlwind Tour of Python, Jake VanderPlas
Chapter 8: Control Flow Statements

https://docs.python.org/3/tutorial/controlflow.html

control flow

Control flow statements in a programming language control whether and how often other statements in that language are executed.

If-Then-Else

For Loops

While Loops

if-then-else statements

controls whether or not statements will be executed based on the results of some logical test

```
if Boolean-condition:
    action 1
    action 2
    action 3
```

`ControlFlow.ipynb`

https://docs.python.org/3/tutorial/controlflow.html#if-statements

if-then-else statements

```
x = 8
if x < 10:
    print('Hit "x < 10" condition')
    x = x - 3
print(x)
```

if-then-else statements

```
x = 12
if x < 10:
    print('Hit "x < 10" condition')
    x = x - 3
else:
    print('Hit "else" condition')
    x = x + 3
print(x)
```

if-then-else statements

```
x = 15
if x < 10:
    print('Hit "x < 10" condition')
    x = x - 10
elif x < 20:
    print('Hit "x < 20" condition')
    x = x - 20
else:
    print('Hit "else" condition')
    x = x + 100
print(x)
```

no switch/case in Python

Matlab

```
switch n
    case -1
        disp('negative one')
    case 0
        disp('zero')
    case 1
        disp('positive one')
    otherwise
        disp('other value')
end
```

C

```
switch(expression) {

   case constant-expression  :
      statement(s);
      break; /* optional */

   case constant-expression  :
      statement(s);
      break; /* optional */

   /* you can have any number of case statements */
   default : /* Optional */
   statement(s);
}
```

https://www.python.org/dev/peps/pep-3103/

one way to implement switch-case in Python

```
if    param == 1:
        param 1 statements
elif  param == 2:
        param 2 statements
elif  param == 3:
        param 3 statements
elif  param == 4:
        param 4 statements
else:
        stuff to do otherwise
```

for loops

iterates over a list, tuple, or other "iterable"

`for` *elem* `in` "*iterable*"`:`

action 1

action 2

action 3

`ControlFlow.ipynb`

https://docs.python.org/3/tutorial/controlflow.html#for-statements

for loops

index iterating

```
data = [2.3, 3.1, 4.5, 1.1, 6.3]

for i in range(len(data)):
    print(i, ' : ', data[i])
```

for loops

direct iterating

```
data = [2.3, 3.1, 4.5, 1.1, 6.3]

for d in data:
    print(d)
```

for loops

do both

```
data = [2.3, 3.1, 4.5, 1.1, 6.3]

for i, d in enumerate(data):
    print(i, ' : ', d)
```

for loops

nested for looping over a multidimensional array

```
data = np.array(
         [[6, 8, 3, 4, 2, 4],
          [1, 4, 1, 5, 6, 6]])

(r, c) = data.shape

for i in range(r):
    for j in range(c):
        print(f'({i},{j}) : {data[i,j]}')
```

for loops

`break` and `else` clauses in for loop

```
for n in range(2, 100):
    for x in range(2, n):
        if n % x == 0:
            break
    else:
        print(f'{n:2d} is a prime number')
```

break out of the inside `for` loop

execute the `else` if the `for` loop completes iterating

for loops

`continue` in for loop

```
data = np.array([324, 143, 434, 241, 104, 341,
                 354, 111, 542, 324])
minRT = 150

tot = 0.;  n = 0
for d in data:
    if d < minRT:
        continue
    tot += d
    n += 1
print(f'{(tot/n):3.0f}')
```

for loops

`zip()`

```
a = [1, 2, 3]
b = ['a', 'b', 'c']
c = [True, False, False]

for A, B, C in zip(a, b, c):
    print(A, B, C)
```

for loops

note difference:

```
a = [1, 2, 3]
b = ['a', 'b', 'c']
c = [True, False, False]

for A in a:
    for B in b:
        for C in c:
            print(A, B, C)
```

while loops

```
while Boolean-condition:
    action 1
    action 2
    action 3
```

`ControlFlow.ipynb`

while loops

How would we replicate a `for` loop with a `while` loop?

```
for i in range(10):
    print(i)
```

while loops

How would we replicate a `for` loop with a `while` loop?

```
for i in range(10):
    print(i)
```

```
i = 0
while (i < N):
    print(i)
    i += 1
```

while loops

```
N = 10;
i = 1;
while i<=N
    i
    i = i + 1;
end
```

A `for` loop loops a set number of times.

A `while` loop can loop until a condition is satisfied.