EECS 445 - Introduction to Machine Learning

Lecture 2: Linear Algebra and Optimization

Date: September 12, 2016

Instructor: Jacob Abernethy and Jia Deng

In [1]:
```
from IPython.core.display import HTML, Image
from IPython.display import YouTubeVideo
from sympy import init_printing, Matrix, symbols, Rational
import sympy as sym
from warnings import filterwarnings
init_printing(use_latex = 'mathjax')
filterwarnings('ignore')

%pylab inline

import numpy as np
```

```
Populating the interactive namespace from numpy and matplotlib
```

TODAY: Fast overview of linear algebra + convexity

we're going to do:

- Vectors and norms
- Matrices
- Positive definite matrices
- Eigendecomposition
- Singular Value Decomposition

In [2]:
```
a11, a12, a13, a21, a22, a23, a31, a32, a33, b11, b12, b13, b21, b22, b2
3, b31, b32, b33 = symbols('a11 a12 a13 a21 a22 a23 a31 a32 a33 b11 b12
 b13 b21 b22 b23 b31 b32 b33')
```

Basic matrix multiplication

In [3]:
```
A = Matrix([[a11, a12, a13], [a21, a22, a23]])
B = Matrix([[b11, b12], [b21, b22], [b31, b32]])
A, B
```

Out[3]:

$$\left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}, \quad \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \right)$$

In [4]:
```
C = A * B
C
```

Out[4]:

$$\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{bmatrix}$$

In [5]:
```
A = Matrix([[3, 1], [1, 3]])
B = Matrix ([[1, 2], [1,4]])
C1 = A * B
C2 = B * A
C1, C2
```

Out[5]:

$$\left(\begin{bmatrix} 4 & 10 \\ 4 & 14 \end{bmatrix}, \quad \begin{bmatrix} 5 & 7 \\ 7 & 13 \end{bmatrix} \right)$$

Matrix Transpose

- The transpose A^T of a matrix A is what you get from "swapping" rows and columns

$$A \in \mathbb{R}^{n \times m} \implies A^T \in \mathbb{R}^{m \times n}$$

$$(A^T)_{i,j} := A_{j,i}$$

In [6]:
```
A = Matrix ([[1, 2], [3,4], [5,6]])
B = Matrix ([[1, 2], [3,4]])
A, A.transpose(), B, B.transpose()
```

Out[6]:

$$\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \quad \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \right)$$

- A matrix A is *symmetric* if we have $A^\top = A$

In [7]:
```
A = Matrix ([[1, 2], [2,1]])
A, A.transpose()
```

Out[7]:

$$\left(\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \right)$$

- Some easy ways to get a symmetric matrix:

$$A + A^\top, \quad AA^\top, \quad A^\top A$$

Transpose properties

- Obvious properties of the transpose:
  - $(A+B)^T = A^T + B^T$
  - $(AB)^T = A^T B^T$ (......right?)

- No! Careful!
  - $(AB)^T = B^T A^T$

Rank of a matrix

- Linear independent vectors: no vector can be represented as a linear combination of other vectors.
- rank(A) (the rank of a m-by-n matrix A) is
  - The maximal number of linearly independent columns = The maximal number of linearly independent rows
- col(A), the column space of a m-by-n matrix A, is the set of all possible linear combinations of its column vectors.
- row(A), the row space of a m-by-n matrix A, is the set of all possible linear combinations of its row vectors.
- rank(A) = dimension of col(A) = dimension of row(A)

We can still talk about Rank for non-square matrices

- If A is n by m, then
  - rank(A) ≤ min(m,n)
  - If rank(A) = n, then A has full row rank
  - If rank(A) = m, then A has full column rank

Vector Norms

- A norm measures the "length" of a vector
- We usually use notation $\|x\|$ to denote the norm of x
- A norm is a function $f : \mathbb{R}^n \to \mathbb{R}$ such that:
  - $f(x) \geq 0$ for all x
  - $f(x) = 0 \iff x = 0$
  - $f(tx) = |t| f(x)$ for all x
  - $f(x+y) \leq f(x) + f(y)$ for all x and y (Triangle Inequality)

Examples of norms

- Perhaps the most common norm is the Euclidean norm

$$\|x\|_2 := \sqrt{x_1^2 + x_2^2 + \ldots x_n^2}$$

- This is a special case of the p-norm:

$$\|x\|_p := (|x_1|^p + \ldots + |x_n|^p)^{1/p}$$

- There's also the so-called **infinity norm**

$$\|x\|_\infty := \max_{i=1,\ldots,n} |x_i|$$

- A vector x is said to be *normalized* if $\|x\| = 1$

Matrix inversion

- The inverse A^{-1} of a *square matrix* A is the unique matrix such that $AA^{-1} = A^{-1}A = I$
- The inverse doesn't always exist! (For example, when A not full-rank)
- If A and B are invertible, then AB is invertible and $(AB)^{-1} = B^{-1}A^{-1}$
- If A is invertible, then A^T is invertible and $(A^T)^{-1} = (A^{-1})^T$

In [8]:
```
X = np.array([[3, 1], [1, 3]])
Matrix(X) # putting Matrix() around X is just for pretty printing
```

Out[8]:

$$\begin{bmatrix} 3.0 & 1.0 \\ 1.0 & 3.0 \end{bmatrix}$$

In [9]:
```
Xinv = np.linalg.inv(X)
Matrix(Xinv), Matrix(Xinv.dot(X)), Matrix(X.dot(Xinv)) # Should give the
 identity matrix
```

Out[9]:

$$\left(\begin{bmatrix} 0.375 & -0.125 \\ -0.125 & 0.375 \end{bmatrix}, \quad \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}, \quad \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix} \right)$$

Determinant + Trace

- The determinant of a **square** matrix A, denoted $|A|$, has the following recursive structure: $$\begin{vmatrix} a & b & c & d\e & f & g & h\i & j & k & l\m & n & o & p \end{vmatrix}=a\begin{vmatrix} f & g & h\j & k & l\n & o & p \end{vmatrix}
- b

  $$|e \quad g \quad h\backslash\mathrm{i} \quad k \quad l\backslash\mathrm{m} \quad o \quad p|$$

  \ +c

  $$|e \quad f \quad h\backslash\mathrm{i} \quad j \quad l\backslash\mathrm{m} \quad n \quad p|$$

  -d

  $$|e \quad f \quad g\backslash\mathrm{i} \quad j \quad k\backslash\mathrm{m} \quad n \quad o|$$

  . $$
- $|A| \neq 0$ if and only if A is invertible (non-singular).
- The trace of a matrix, denoted tr(A), is defined as the sum of the diagonal elements of A

Orthogonal + Normalized = Orthonormal

- Two vectors x, y are *orthogonal* if $x^T y = 0$
- A square matrix $U \in \mathbb{R}^{n \times n}$ is *orthogonal* if all columns $U_1, \ldots, U_n$ are orthogonal to each other (i.e. $U_i^\top U_j = 0$ for $i \neq j$)
- U is *orthonormal* if it is orthogonal **and** the columns are normalized, i.e. $\|U_i\|_2 = 1$ for every i.
- If U is orthonormal, then $U^T U = I$, that is, $U^{-1} = U^T$.

Positive Definiteness

- We say a symmetric matrix A is positive definite if

$$x^\top Ax > 0 \text{ for all } x \neq 0$$

- We say a matrix is positive semi-definite (PSD) if

$$x^\top Ax \geq 0 \text{ for all } x$$

- A matrix that is positive definite gives us a *norm*. Let

$$\|x\|_A := \sqrt{x^\top Ax}$$

Eigenvalues and Eigenvectors

What are eigenvectors?

- A Matrix is a mathematical object that acts on a (column) vector, resulting in a new vector, i.e. A**x**=**b**
- An eigenvector is *non-zero* vector such that the resulting vector is parallel to **x** (some multiple of **x**)

$$A\underline{x} = \lambda\underline{x}$$

- λ is called an eigenvalue.
- The eigenvectors with an eigenvalue of zero are the vectors in the nullspace of A.
- If A is singular (takes some non-zero vector into 0) then zero is an eigenvalue.

In [10]:
```
lamda = symbols('lamda') # Note that lambda is a reserved word in pytho
n, so we use lamda (without the b)
```

How to solve Ax=λx

$$A\underline{x} = \lambda\underline{x}$$
$$(A - \lambda I)\,\underline{x} = \underline{0}$$

- The only solution to this equation is for A-λI to be singular and therefor have a determinant of zero

$$|A - \lambda I| = 0$$

- $|A - \lambda I|$ is a polynomial of the variable λ and is called the characteristic polynomial of A.
- The eigenvalues are the roots of the equation of $|A - \lambda I| = 0$. They may be complex numbers.
- There will be *n* λ's for an $n \times n$ matrix (some of which may be of equal value)
- Given an eigenvalue λ, its eigenvectors are the null space of $A - \lambda I$.

Example eigenvalue problem

In [11]:
```
A = Matrix([[3, 1], [1, 3]])
I = sym.eye(2)
A, I # Printing A and the 2-by-2 identity matrix to the screen
```

Out[11]:

$$\left(\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$$

In [12]:
```
(A - lamda * I) # Printing A minus lambda times the identity matrix to t
he screen
```

Out[12]:

$$\begin{bmatrix} -\lambda + 3 & 1 \\ 1 & -\lambda + 3 \end{bmatrix}$$

- This will have the following (symbolic) determinant polynomial

In [13]:
```
(A - lamda * I).det()
```

Out[13]:

$$\lambda^2 - 6\lambda + 8$$

Eigenvalue example

- We can solve the polynomial $\lambda^2 - 6\lambda + 8$ with python:

In [14]:
```
((A - lamda * I).det()).factor()
```

Out[14]:

$$(\lambda - 4)(\lambda - 2)$$

- I now have two eigenvalues of 2 and 4
- I can get the two eigenvectors, x_1 and x_2, by solving

$$(A - 2I)x_1 = 0 \text{ and } (A - 4I)x_2 = 0$$

- I need to find a vector in the *null space* of $A - 2I$ and $A - 4I$.

Getting eigenvalues/vectors using numpy

In [15]:
```
X = np.array([[3, 1], [1, 3]])
Matrix(X)
```

Out[15]:

$$\begin{bmatrix} 3.0 & 1.0 \\ 1.0 & 3.0 \end{bmatrix}$$

In [16]:
```
eigenvals, eigenvecs = np.linalg.eig(X)
Matrix(eigenvecs)
```

Out[16]:

$$\begin{bmatrix} 0.707106781186547 & -0.707106781186547 \\ 0.707106781186547 & 0.707106781186547 \end{bmatrix}$$

In [17]:
```
Matrix(eigenvals)
```

Out[17]:

$$\begin{bmatrix} 4.0 \\ 2.0 \end{bmatrix}$$

Trace related to eigenvals

- Let A be a matrix whose eigenvalues are $\lambda_1, \ldots, \lambda_n$
- Then we have the trace of satisfying

$$\text{tr}(A) = \sum_{i=1}^{n} \lambda_i$$

In [18]:
```
X = np.random.randn(5,10)
A = X.dot(X.T) # For fun, let's look at A = X * X^T
eigenvals, eigvecs = np.linalg.eig(A) # Compute eigenvalues of A
sum_of_eigs = sum(eigenvals) # Sum the eigenvalues
trace_of_A = A.trace() # Look at the trace
(sum_of_eigs, trace_of_A) # Are they the same?
```

Out[18]:

$$(56.1848049439, \quad 56.1848049439)$$

Determinant related to eigenvals

- Let A be a matrix whose eigenvalues are $\lambda_1, \ldots, \lambda_n$
- Then we have the trace of satisfying

$$|A| = \prod_{i=1}^{n} \lambda_i$$

In [19]:
```
# We'll use the same matrix A as before
prod_of_eigs = np.prod(eigenvals) # Sum the eigenvalues
determinant = np.linalg.det(A) # Look at the trace
(prod_of_eigs, determinant) # Are they the same?
```

Out[19]:

$$(21358.046809, \quad 21358.046809)$$

Singular Value Decomposition

- Any matrix (symmetric, non-symmetric, etc.) $A \in \mathbb{R}^{n \times m}$ admits a *singular value decomposition* (SVD)
- The decomposition has three factors, $U \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times m}$, and $V \in \mathbb{R}^{m \times m}$

$$A = U \Sigma V^\top$$

- U and V are both orthonormal matrices, and Σ is diagonal

SVD Example

In [20]:
```
A = np.array([[4, 4], [-3, 3]])
Matrix(A)
```

Out[20]:

$$\begin{bmatrix} 4.0 & 4.0 \\ -3.0 & 3.0 \end{bmatrix}$$

- Let's show Sigma from the SVD output

In [21]:
```
U, Sigma_diags, V = np.linalg.svd(A)
Matrix(np.diag(Sigma_diags)) # Numpy's SVD only returns diagonals, here
 I'm showing full Sigma
```

Out[21]:

$$\begin{bmatrix} 5.65685424949238 & 0.0 \\ 0.0 & 4.24264068711928 \end{bmatrix}$$

- And we can show the orthonormal bases U and V

In [22]:
```
U,V = np.round(U,decimals=5), np.round(V,decimals=5)
```

In [23]:
```
Matrix(U), Matrix(V) # I rounded the values for clarity
```

Out[23]:

$$\left(\begin{bmatrix} -1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}, \begin{bmatrix} -0.70711 & -0.70711 \\ -0.70711 & 0.70711 \end{bmatrix} \right)$$

Properties of the SVD

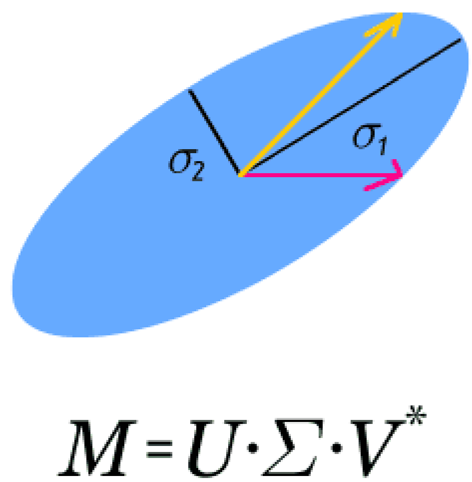
$$\text{SVD:} \qquad A = U\Sigma V^\top$$

- The *singular values* of A are the diagonal elements of Σ
- The singular values of A are the *square roots of the eigenvalues* of both $A^\top A$ and $AA^\top$
- The *left-singular vectors* of A, i.e. the columns of U, are the *eigenvectors* of $AA^\top$
- The *right-singular vectors* of A, i.e. the columns of V, are the *eigenvectors* of $A^\top A$

$$M = U\Sigma V^T$$

In [24]:
```
Image(url='https://upload.wikimedia.org/wikipedia/commons/e/e9/Singular_value_decomposition.gif')
```

Out[24]:



- Wikipedia: Visualization of the SVD of a 2d matrix M. First, we see the unit disc in blue together with the two canonical unit vectors. We then see the action of M, which distorts the disk to an ellipse. The SVD decomposes M into three simple transformations: an initial rotation V∗, a scaling Σ along the coordinate axes, and a final rotation U. The lengths σ1 and σ2 are singular values of M.

Functions and Convexity

- Let f be a function mapping $\mathbb{R}^n \to \mathbb{R}$, and assume f is twice differentiable.
- The *gradient* and *hessian* of f, denoted $\nabla f(x)$ and $\nabla^2 f(x)$, are the vector an matrix functions:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- Note: the hessian is always symmetric!

Gradients of three simple functions

- Let b be some vector, and A be some matrix
- $f(x) = b^\top x \implies \nabla_x f(x) = b$
- $f(x) = x^\top A x \implies \nabla_x f(x) = 2Ax$
- $f(x) = x^\top A x \implies \nabla_x^2 f(x) = 2A$

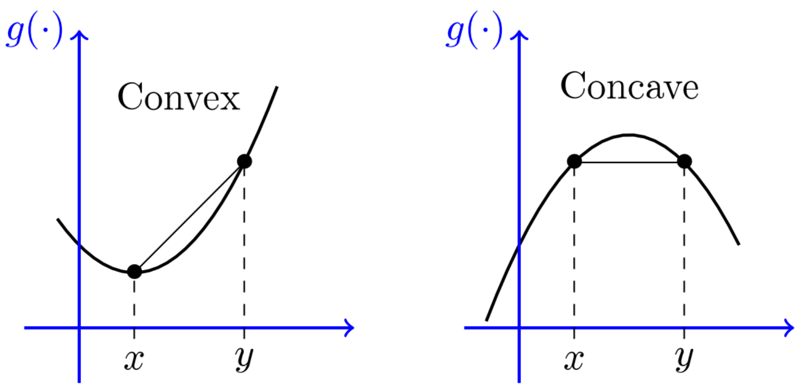
Convex functions

- We say that a function f is *convex* if, for any distinct pair of points x, y we have

$$f\left(\frac{x+y}{2}\right) \leq \frac{f(x)}{2} + \frac{f(y)}{2}$$

In [25]:
```
Image(url='http://www.probabilitycourse.com/images/chapter6/Convex_b.png', width=400)
```

Out[25]:



Fun facts about convex functions

- If f is differentiable, then f is convex iff f "lies above its linear approximation", i.e.:

$$f(x+y) \geq f(x) + \nabla_x f(x) \cdot y \quad \text{for every } x, y$$

- If f is twice-differentiable, then the hessian is always positive semi-definite!

See you all on Wednesday!