Linear Algebra/Matrix Calculus Recitation

Suraj Nair

Caltech CS155

Winter 2017

Linear Algebra Outline

We will review some basic linear algebra concepts and cover how they can be applied using numpy and python. We will skip some concepts which are not relevant such as Dependence /Independence, Rank, and Linear Maps

- Basic Linear Algebra + Numpy Tools
  - Linear Space
  - Matrix and Vectors
  - Matrix Multiplication
  - Operators and Properties
  - Special Matrices
  - Vector Norms
  - Inverse of Matrix

Linear Space

- A vector space over a field F is a set V with operations addition, subtraction, and scalar multiplication.
- Also satisfies number of axioms such as zero vector, additive inverse, associative law, commutative law, identity element, distributivity.

Matrix and Vectors

- A matrix A^{mxn} is a rectangular array of numbers, a column vector $x_C \in \mathbb{R}^m$, and a row vector $x_R \in \mathbb{R}^n$

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad x_C = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \quad x_R = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$$

- In Numpy:
  Matrix:

```
A = np.array([[1, 2], [3, 4]])
```

  Vector:

```
>>> np.array([1,2,3]).reshape((3,1)) # Column Vector
array([[1],
   [2],
   [3]])
>>> np.array([1,2,3]).reshape((1,3)) # Row Vector
array([[1, 2, 3]])
```

Matrix Multiplication

- With A^{mxn}, B^{nxp}, then $C^{mxp} = AB$ with

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

Properties:

- Associativity:$(AB)C = A(BC)$
- Distributive: $A(B + C) = AB + AC$
- Non-commutative generally $AB \neq BA$

Matrix Multiplication (Code)

For matrix multiplication np.dot is the best option. Note, np.multiply is not matrix multiplication it is pairwise. np.matrix does not work as well for > 2 dimensions, so we recommend using np.array.

```
>>> A = np.array([[1,2,3],[4,5,6]])
>>> A.shape
(2, 3)
>>> B = np.array([[1,2,3,4,5],[6,7,8,9,10],
[11,12,13,14,15]])
>>> B.shape
(3, 5)
>>> np.dot(A,B).shape
(2, 5)
>>> np.dot(A,B)
array([[ 46,  52,  58,  64,  70],
[100, 115, 130, 145, 160]])
```

Operators and Properties

- Transpose: if $A \in \mathbb{R}^{m \times n}$, then $A^T \in \mathbb{R}^{n \times m} : (A^T)_{ij} = A_{ji}$
  - $(A^T)^T = A$
  - $(AB)^T = B^T A^T$
  - $(A + B)^T = A^T + B^T$
  - In Numpy:

```
>>> A = np.array([[1,2,3],[4,5,6]])
>>> A
array([[1, 2, 3],
[4, 5, 6]])
>>> A.T
array([[1, 4],
[2, 5],
[3, 6]])
```

- Trace - sum along diagonal.

```
np.trace()
```

Special Matrices

- Identity Matrix: $I = I_n \in \mathbb{R}^{n \times n}$, $\forall A \in \mathbb{R}^{n \times n} : AI_n = I_n A = A$

  `np.identity(n)`
- Diagonal Matrix: Only nonzero along diagonal
- Symmetric Matrix: Square matrix where $A = A^T$
- Orthogonal Matrix: Square matrix where $AA^T = A^T A = I$

Vector Norm

A norm of a vector space V is a function $||.|| : V \rightarrow R^+$ such that

- $||x|| = 0$ iff $x = 0$
- $||\alpha x|| = |\alpha| * ||x||$
- $||x + y|| \leq ||x|| + ||y||$
- In Numpy:

  `np.linalg.norm(x, ord=None)`

  ord parameter allows you to select what type of norm (l_1, l_2, etc), default Frobenius

Norm of a vector is a measure of its magnitude. Formally:

- l_p norm: $||x||_p = (\sum_{i=1}^{n} x_i^p)^{1/p}$

Most common are the l_2 and l_1 norms, used in Ridge and Lasso regularization respectively.

Inverse of Matrix

A matrix $A \in \mathbb{R}^{n \times n}$ is invertible $\exists B \in \mathbb{R}^{n \times n}$ such that $AB = I = BA$. If invertible: B is the inverse of A, written as $B = A^{-1}$

- $(A^{-1})^{-1} = \text{A}$
- $(AB)^{-1} = B^{-1}A^{-1}$
- $(A^{-1})^T = (A^T)^{-1}$

Easiest way to check if matrix is invertible is to see if determinant is not 0, where determinant is defined as:

$$det(A) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} a_{i,\sigma_i}$$

There are other ways (Look up Invertible Matrix Theorem), but we will not cover them here.

Matrix Calculus Outline

Useful Resources

- All of the matrix calculus rules you need to know for this class can be found in "The Matrix Cookbook" (matrixcookbook.com).
- Also the Wikipedia to matrix calculus is a very good introduction to this material.

The Gradient

Consider a differentiable real function $f : \mathbb{R}^{mxn} \to \mathbb{R}$
Then, for $x \in \mathbb{R}^{mxn}$, the gradient $\nabla : \mathbb{R}^{mxn} \to \mathbb{R}^{mxn}$ of f is by definition

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_{11}} & \frac{\partial f(x)}{\partial x_{12}} & \cdots & \frac{\partial f(x)}{\partial x_{1n}} \\ \frac{\partial f(x)}{\partial x_{21}} & \frac{\partial f(x)}{\partial x_{22}} & \cdots & \frac{\partial f(x)}{\partial x_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(x)}{\partial x_{m1}} & \frac{\partial f(x)}{\partial x_{m2}} & \cdots & \frac{\partial f(x)}{\partial x_{mn}} \end{bmatrix}$$

In most cases in this course, x will be a vector $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$. In that case the gradient $\nabla : \mathbb{R}^n \to \mathbb{R}^n$ of f is

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

The Hessian

Let x be a vector $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ Then the Hessian $\nabla^2 : \mathbb{R}^n \to \mathbb{R}^n$ of f is

$$\nabla_x^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

Gradient for Vector-Valued Functions

For $x \in \mathbb{R}$ and $v : \mathbb{R} \to \mathbb{R}^N$ Then the gradient $\nabla : \mathbb{R} \to \mathbb{R}^N$ of v is

$$\nabla_x v(x) = \begin{bmatrix} \frac{\partial v_1(x)}{\partial x} & \frac{\partial v_2(x)}{\partial x} & \cdots & \frac{\partial v_N(x)}{\partial x} \end{bmatrix}$$

For $x \in \mathbb{R}^K$ and $h : \mathbb{R}^K \to \mathbb{R}^N$ Then the gradient $\nabla : \mathbb{R}^K \to \mathbb{R}^{KxN}$ of h is

$$\nabla_x h(x) = \begin{bmatrix} \frac{\partial h_1(x)}{\partial x_1} & \frac{\partial h_2(x)}{\partial x_1} & \cdots & \frac{\partial h_N(x)}{\partial x_1} \\ \frac{\partial h_1(x)}{\partial x_2} & \frac{\partial h_2(x)}{\partial x_2} & \cdots & \frac{\partial h_N(x)}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_1(x)}{\partial x_K} & \frac{\partial h_2(x)}{\partial x_K} & \cdots & \frac{\partial h_N(x)}{\partial x_K} \end{bmatrix}$$

Note About Layouts

There are two main notational conventions for matrix calculus
For $x \in \mathbb{R}^K$ and $h : \mathbb{R}^K \to \mathbb{R}^N$ Then the gradient of h is
Numerator Layout : $\nabla : \mathbb{R}^K \to \mathbb{R}^{NxK}$

$$\nabla_x h(x) = \begin{bmatrix} \frac{\partial h_1(x)}{\partial x_1} & \frac{\partial h_1(x)}{\partial x_2} & \cdots & \frac{\partial h_1(x)}{\partial x_K} \\ \frac{\partial h_2(x)}{\partial x_1} & \frac{\partial h_2(x)}{\partial x_2} & \cdots & \frac{\partial h_2(x)}{\partial x_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_N(x)}{\partial x_1} & \frac{\partial h_N(x)}{\partial x_2} & \cdots & \frac{\partial h_N(x)}{\partial x_K} \end{bmatrix}$$

Denominator Layout : $\nabla : \mathbb{R}^K \to \mathbb{R}^{KxN}$

$$\nabla_x h(x) = \begin{bmatrix} \frac{\partial h_1(x)}{\partial x_1} & \frac{\partial h_2(x)}{\partial x_1} & \cdots & \frac{\partial h_N(x)}{\partial x_1} \\ \frac{\partial h_1(x)}{\partial x_2} & \frac{\partial h_2(x)}{\partial x_2} & \cdots & \frac{\partial h_N(x)}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_1(x)}{\partial x_K} & \frac{\partial h_2(x)}{\partial x_K} & \cdots & \frac{\partial h_N(x)}{\partial x_K} \end{bmatrix}$$

You can use whichever you prefer, but be consistent.

Handy Matrix Differentiation Rules

- $y = Ax$ where A does not depend on x.

$$\frac{\partial y}{\partial x} = A$$

- $\alpha = y^T Ax$ where A does not depend on x or y.

$$\frac{\partial \alpha}{\partial x} = y^T A, \frac{\partial \alpha}{\partial y} = x^T A^T$$

- $y = A^T xB$ where A and B do not depend on x.

$$\frac{\partial y}{\partial x} = AB^T$$

Examples (Least Squares)

- Real valued function $f : \mathbb{R}^n \to \mathbb{R}$ where for $x \in \mathbb{R}^n$,

$$f(x) = ||Ax - b||_2^2 = (Ax - b)^T(Ax - b)$$

$$= x^T A^T Ax - b^T Ax - x^T A^T b + b^T b$$

- To find the x that minimizes f, we need to compute the gradient with respect to x.

$$\nabla_x f(x) = \nabla_x(x^T A^T Ax - b^T Ax - x^T A^T b + b^T b)$$

$$= \nabla_x(x^T A^T Ax) - \nabla_x(b^T Ax) - \nabla_x(x^T A^T b) + \nabla_x(b^T b)$$

$$= \mathbf{2A^T Ax - 2A^T b}$$

Examples (Least Squares 2)

What exactly is happening here

$$= \nabla_{\boldsymbol{x}}(\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}\boldsymbol{x}) - \nabla_x(b^T Ax) - \nabla_x(x^T A^T b) + \nabla_x(b^T b)$$

$$= 2A^T Ax - 2A^T b$$

$$\nabla_x(x^T A^T Ax) =$$

$$\nabla_x \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

The result of these matrix multiplications is a scalar $c(x)$ of $x = [x_1, x_2, ..., x_n]$. The gradient then is as we saw earlier, $[\frac{\partial c}{\partial x_1}, \frac{\partial c}{\partial x_2}, ..., \frac{\partial c}{\partial x_n}]$

Examples (Vector Valued Functions)

- Consider the vector valued function $v : \mathbb{R} \to \mathbb{R}^K$ where for $x \in \mathbb{R}$,

$$v(x) =< x, x^2, \ldots, x^n >$$

Then,

$$\nabla_x v(x) =< 1, 2x, \ldots, nx^{n-1} >$$

- Consider the vector valued function $h : \mathbb{R}^N \to \mathbb{R}^K$ where for $x \in \mathbb{R}^N$,

$$h(x) = ABx + A^T xB$$

$$\nabla_x h(x) = AB + AB^T$$

Generally, and for most problems in this class, the gradients of matrix products will only require applying basic rules and rules that can be found in any book on Matrix calculus. However, you can always verify them by expanding the matrix product and for each parameter in the output, taking the partials with respect to $x_1, x_2, ..$ as described in the previous slides.

Numpy Examples

- An IPython notebook will be available for download which actually codes up the above examples and more using Numpy
- Will also demonstrate matrix broadcasting, addition, multiplication, and other basic operations.