Linear Algebra Part 1

Arthur J. Redfern
axr180074@utdallas.edu
Aug 22, 2018

---

0 Outline

Previous
1. Introduction

Current
1. High level view
2. Vector spaces
3. Linear feature extraction and prediction
4. References

Next
1. Linear algebra part 2

1 High Level View

CNNs are compositions of nonlinear functions

$$\mathbf{y} = f_{D-1}(\dots (f_2(f_1(f_0(\mathbf{x}, h_0), h_1), h_2), \dots), h_{D-1})$$

- Transform from data space to feature space to information space
- The key part of the nonlinear functions are actually linear transformations
- Understanding linear transformations is a key to the design and implementation of xNNs

Goal is to be comfortable with all 3 of the following
- Theory
- Mechanics of the operations
- Intuition of what the operations are doing

Presentation

- The book is a more comprehensive in its presentation of linear algebra topics
  - The provided references are even more comprehensive
- This lecture will bias the coverage of linear algebra topics to emphasize connections to xNNs
  - Elements and vector spaces
  - Linear feature extraction and prediction
    - Matrix vector multiplication

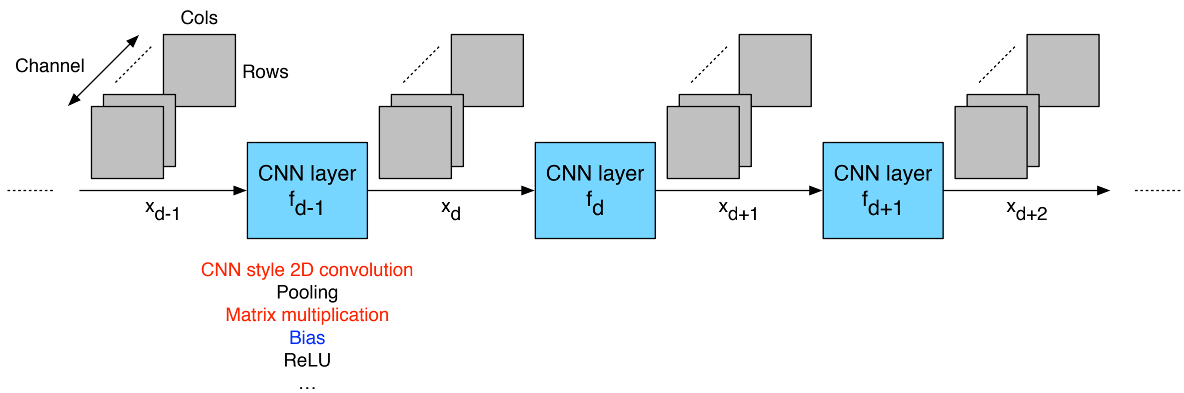

Figure: CNNs are compositions of nonlinear functions; however, the most important building blocks are linear transformations (linear transformations in red, affine transformation in blue)

2 Vector Spaces

Notation: scalars are not bold, vectors are bold lower case, matrices are bold upper case

- Indices start at 0 and go from 0, …, size – 1

Set

- A collection of distinct objects

Field

- A set with well defined addition and multiplication operations
  - Associativity: a + (b + c) = (a + b) + c and a (b c) = (a b) c
  - Commutativity: a + b = b + a and a b = b a
  - Additive identity: a + 0 = 0
  - Additive inverse: a + (-a) = 0
  - Multiplicative identity: 1 a = a
  - Multiplicative inverse: a a^{-1} = 1
  - Distributivity: a (b + c) = (a b) + (b c)

- Elements of fields are generally referred to as scalars
- Examples: R (real scalars), C (complex scalars)

Vector
- K tuple of scalars, always columns
- F^K
- Examples: R^K and C^K

Matrix
- M x K tuple of scalars
- Collection of K vectors of size M x 1 arranged in columns
  - Leads to column space and right null space
    - What can matrix vector multiplication reach and what can it not
  - Visualize using outer product of matrix vector multiplication
- Collection of M vectors of size K x 1 transposed and arranged as rows
  - Leads to row space and left null space
    - What can vector matrix multiplication reach and what can it not
  - Visualize using outer product of vector matrix multiplication

Tensor
- K_0 x … x K_{D-1} array of scalars
- Ordering
  - Last dimension is contiguous in memory
  - Working from right to left goes from closest to farthest spacing in memory
  - Batch x channel x row x column
  - Motivation: efficiency in hardware implementation when reading from memory

Function
- Mapping f: $X \rightarrow Y$ from domain to co domain
- Injective: one to one; each y produced by at most 1 x
- Surjective: onto; each y produced by at least 1 x
- Bijective: one to one and onto
  - Bijective functions are invertible
  - Motivation: allow a modification to the ReLU operation to reduce memory required during xNN training (re generation through ReLU vs storing input during back propagation)
- An infinite set is
  - Countably infinite if there's a bijection between the natural numbers and elements of the set
  - Uncountably infinite if there's not

Vector space
- Set of vectors and linear combinations of those vectors

- Satisfy associativity, commutativity, additive identity, additive inverse, multiplicative compatibility, multiplicative identity and distributivity
  - Associativity: $\mathbf{x} + (\mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y}) + \mathbf{z}$
  - Commutativity: $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$
  - Additive identity: $\mathbf{x} + \mathbf{0} = \mathbf{0}$
  - Additive inverse: $\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$
  - Multiplicative compatibility: $a (b\ \mathbf{x}) = b (a\ \mathbf{x})$
  - Multiplicative identity: $1\ \mathbf{x} = \mathbf{x}$
  - Distributivity: $(a + b)(\mathbf{x} + \mathbf{y}) = a\ \mathbf{x} + a\ \mathbf{y} + b\ \mathbf{x} + b\ \mathbf{y}$
- Examples: R^K, C^K, $R^{K\_0 \times \ldots \times K\_D-1}$

Normed vector space

- A vector space with a notion of distance
- A norm maps an element of the vector space to a scalar
- Satisfies non negativity, absolute scalability and the triangle inequality
  - Non negativity: $\|\mathbf{x}\| \geq 0$ and $\|\mathbf{x}\| = 0$ iff $\mathbf{x} = \mathbf{0}$
  - Absolute scalability: $\|a\ \mathbf{x}\| = |a|\ \|\mathbf{x}\|$
  - Triangle inequality: $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$
- Example: l_p norm (common p = 1, 2 and ∞)

$$\|\mathbf{x}\|_p = (\Sigma_n(|x(n)|^p))^{1/p},\ p \geq 1$$

Inner product space

- A vector space with a notion of distance and angle
- An inner product maps 2 elements of a vector space to a scalar
- Satisfies positive definiteness, conjugate symmetry, linearity
  - Positive definiteness: $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$ and $\langle \mathbf{x}, \mathbf{x} \rangle = 0$ iff $\mathbf{x} = \mathbf{0}$
  - Conjugate symmetry: $\langle \mathbf{x}, \mathbf{y} \rangle = \text{conj}(\langle \mathbf{y}, \mathbf{x} \rangle)$
  - Linearity: $\langle a\ \mathbf{x}, \mathbf{y} \rangle = a \langle \mathbf{x}, \mathbf{y} \rangle$ and $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$
- Inner products induce norms on a vector space
  - But not all norms have associated inner products (e.g., l_∞)
- Example: dot product

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x} \cdot \mathbf{y} = \mathbf{x}^H\ \mathbf{y} = \Sigma_n\ (\text{conj}(x(n))\ y(n)) = \|\mathbf{x}\|_2\ \|\mathbf{y}\|_2 \cos(\theta)$$

- Motivation: understanding linear feature extraction and prediction

3 Linear Feature Extraction And Prediction

More specifically, linear transformations that will be used for feature extraction and prediction

3.1 Matrix Vector Multiplication

Notation: M (output dimension), K (input dimension)

- Setting up for BLAS notation

$$\begin{bmatrix} y(0) \\ \vdots \\ y(M-1) \end{bmatrix} = \begin{bmatrix} A(0,0) & \cdots & A(0,K-1) \\ \vdots & & \vdots \\ A(M-1,0) & \cdots & A(M-1,K-1) \end{bmatrix} \begin{bmatrix} x(0) \\ \vdots \\ x(K-1) \end{bmatrix}$$

Mechanics

- Inner product of matrix row and vector input to produce each output

Motivation: traditional neural network is composed of fully connected layers

- Output vector = pointwise nonlinearity (matrix * input vector + bias vector)
- Repeat

Matrix vector multiplication is a linear transformation

- Every linear map can be represented as a matrix and every matrix represents a linear map
  - So matrix vector multiplication in a neural network is doing linear transformations
- Multiple linear transformations can be composed into a single linear transformation

$$\mathbf{y} = \mathbf{A}_{D-1} \ldots \mathbf{A}_1 \mathbf{A}_0 \mathbf{x} = \mathbf{A}\, \mathbf{x}$$

  - Motivation: a reason why nonlinearities are included in xNNs (otherwise there would be no depth)

Intuition of feature extraction and prediction

- Inner product depends on magnitude and angle
  - $y(m) = \mathbf{A}(m, :)\, \mathbf{x}$
  - $y(m)$ is the extracted feature or prediction
  - $\mathbf{A}(m, :)$ is the feature extractor or predictor
  - $\mathbf{x}$ is the input
- How strong or important is a feature extractor? $\|\mathbf{A}(m, :)\|_2$
  - Note that the input magnitude contributes the same to each extracted feature $\|\mathbf{x}\|_2$
  - Here input magnitude only matters relative to bias
  - But input magnitude will also matter for network structures with branches that come together

  - Input magnitude will also matter when the same feature extractor is applied to different inputs
- How aligned is the feature extractor with the input? θ
  - In same direction: positive feature
  - Orthogonal: 0 feature
  - In opposite direction: negative feature
- Note: sometimes linear classification is viewed as template matching where each row is a different template and the predicted class is the maximum output

Intuition of bias

- Affine transformation
- Allows the dividing line to shift
- Implementation of rank 1 outer product
- Will use bias in a constructive variant of the universal approximation proof

Intuition of ReLU

- Removes not positively aligned features or predictions
- Allows depth
  - Subsequent layers combine positively aligned extracted features

Intuition of size of K and M

- Small K to large M
  - Different combinations of a small number of features to predict a large number of classes
- Large K to small M
  - 1 feature or a combination of features to predict a small number of classes is now possible
- Example: ImageNet classification and final fully connected layer size

Arithmetic intensity

• Compute	= MK	(MACs = multiply accumulates)
• Data movement	= K + MK + M	(elements)
• Ratio	= (MK)/(K + MK + M)	
	≈ 1	(memory wall)

If you want to make matrix vector multiplication run fast, you need to build a fast memory subsystem

3.2 Matrix Matrix Multiplication

Notation: M (output dimension), K (input dimension), N (number of inputs and outputs)

- BLAS M, N, K notation for **Y** = **A X**
- Will try and conform to this throughout

- Matrix vector multiplication is a special case with N = 1

$$\begin{bmatrix} Y(0,0) & \cdots & Y(0,N-1) \\ \vdots & & \vdots \\ Y(M-1,0) & \cdots & Y(M-1,N-1) \end{bmatrix} = \begin{bmatrix} A(0,0) & \cdots & A(0,K-1) \\ \vdots & & \vdots \\ A(M-1,0) & \cdots & A(M-1,K-1) \end{bmatrix} \begin{bmatrix} X(0,0) & \cdots & X(0,N-1) \\ \vdots & & \vdots \\ X(K-1,0) & \cdots & X(K-1,N-1) \end{bmatrix}$$

Mechanics

Application of same matrix transformation to multiple input vectors

- Stack all the inputs next to each other
- Get matrix matrix multiplication
- Motivation: cascade approaches with SPP style layers
- Motivation: batching of inputs through fully connected layers

Lots of other matrix operations and decompositions

- Highlight transpose because it will come up later
  - Transpose swaps matrix element indices
  - When applied to products of matrices remember socks then shoes, shoes then socks (or just remember the formula)

$$\mathbf{C}^T = (\mathbf{A}\ \mathbf{B})^T = \mathbf{B}^T\ \mathbf{A}^T$$

- Motivation: important in understanding the multiple variable chain rule when denominator notation is used, this will show up in back propagation for training

Arithmetic intensity

• Compute	= MNK	(MACs)
• Data movement	= KN + MK + MN	(elements)
• Ratio	= (MNK)/(KN + MK + MN)	
	= $N^3/(3*N^2)$	(special case M = N = K)
	= N/3	(ratio maximized with squares)

Why are bubbles spherical? Min surface area per volume enclosed

- Think of surface area as data movement
- Think of volume as MACs

If you want to make matrix matrix multiplication run fast, choose a large matrix size such that you get multiple operations per element of data moved

4 References

Convolutional neural networks: theory, implementation and application

Chapter 3 linear algebra
https://github.com/arthurredfern/UT-Dallas-CS-6301-CNNs/blob/master/References/ConvolutionalNeuralNetworks.pdf

Linear algebra
https://www.math.ucdavis.edu/~linear/linear-guest.pdf

A guide to convolution arithmetic for deep learning
https://arxiv.org/abs/1603.07285