From Prompts to Vectors:
A Linear Algebraic and Geometric View of Large Language Models

Special Topics Lecture
Mathematical Physics

Abstract

Large Language Models (LLMs) do not operate directly on words or symbolic meaning. Instead, they convert text into vectors in high-dimensional real vector spaces and then apply learned linear operators and nonlinearities. This document begins with an intuitive overview of how prompts become vectors, then develops a rigorous mathematical framework (embeddings, positional encodings, attention, and training) suitable for advanced students. Concrete toy examples are included to make the ideas explicit and computational.

Contents

1 The Big Picture: What Happens When You Type a Prompt? 3

2 Tokenization 3

3 From Token IDs to Vectors 3

4 Vocabulary Space and Canonical Basis 4

5 Embedding as a Linear Map 4

6 Geometry of the Embedding Space 5

7 A Concrete Toy Example: Small Vocabulary, Small Embedding Dimension 5
- 7.1 One-hot vectors in $\mathbb{R}^6$ . . . 5
- 7.2 Choose a small embedding dimension . . . 6
- 7.3 Define an explicit embedding matrix . . . 6
- 7.4 Embed a prompt . . . 6
- 7.5 Geometry now encodes similarity . . . 7
- 7.6 A linear-combination step (attention-like behavior) . . . 7

8 Positional Encoding 7

9 Scaled Dot-Product Attention 8

10 Attention as a Kernel Operator 8

11 High-Dimensional Geometry 8

12 Nonlinear Layers 9

13 Variational Interpretation **9**

14 Spectral Structure **9**

15 Conceptual Synthesis **9**

16 Exercises **10**

17 References **10**

1 The Big Picture: What Happens When You Type a Prompt?

Consider the prompt:

> *Explain the ethical risks of autonomous weapons.*

The model does not "read" words symbolically. Instead, it performs:

$$\text{Text} \to \text{Tokens} \to \text{Token IDs} \to \text{Vectors},$$

and from that point forward the model performs linear algebra and nonlinear transformations on vectors.

> An LLM can be viewed as a dynamical system operating in high-dimensional vector space.

Understanding how text becomes vectors is therefore foundational.

2 Tokenization

LLMs do not operate on full words. They break text into *tokens*, which may be:

- whole words,
- subword fragments,
- punctuation,
- parts of words.

For example, a phrase like "autonomous weapons" might be split into fragments such as `["auto", "nomous", " weapons"]`. Each token is assigned an integer ID from a vocabulary of size V. Thus text becomes a sequence of integers.

3 From Token IDs to Vectors

Each token ID is mapped to a vector in $\mathbb{R}^d$, where typical embedding dimensions satisfy

$$d \in \{768, 1024, 4096, \dots\}.$$

A prompt of length n becomes a matrix

$$X \in \mathbb{R}^{n \times d},$$

and all subsequent computation occurs in $\mathbb{R}^d$.

4 Vocabulary Space and Canonical Basis

Let the vocabulary size be V. Each token corresponds to an index $i \in \{1, \dots, V\}$. A common representation of token i is the i^{th} canonical basis vector in $\mathbb{R}^V$:

$$e_i = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \qquad \langle e_i, e_j \rangle = \delta_{ij}.$$

A tokenized prompt can therefore be represented as a sequence of canonical basis vectors.

5 Embedding as a Linear Map

Define the embedding matrix

$$E \in \mathbb{R}^{V \times d}.$$

The embedding operation maps a canonical basis vector $e_i \in \mathbb{R}^V$ to an embedding vector $x_i \in \mathbb{R}^d$ via

$$x_i = E^T e_i.$$

Thus the embedding layer implements a learned linear transformation

$$\mathcal{E} : \mathbb{R}^V \to \mathbb{R}^d.$$

One-hot vectors (ML jargon) and what changes after embedding

A *one-hot vector* is simply a canonical basis vector. Concretely, $e_i \in \mathbb{R}^V$ is defined componentwise by

$$(e_i)_j = \begin{cases} 1 & j = i, \\ 0 & j \neq i. \end{cases}$$

The collection $\{e_1, \dots, e_V\}$ is the canonical (orthonormal) basis of $\mathbb{R}^V$.

Although these vectors live in $\mathbb{R}^V$, the model's *input* is restricted to the finite set $\{e_1, \dots, e_V\}$: it never begins by taking arbitrary linear combinations of token basis vectors. In particular, for $i \neq j$,

$$\langle e_i, e_j \rangle = 0, \qquad \|e_i - e_j\| = \sqrt{2}.$$

So in the one-hot representation, all distinct tokens are equally far apart; there is no notion of "cat is closer to dog than to runs."

After embedding, token i is represented by $x_i \in \mathbb{R}^d$ with real-valued coordinates. Subsequent layers (especially attention) form *linear combinations* of such vectors, producing states that can lie anywhere in $\mathbb{R}^d$. This is the key conceptual shift: the model moves from a finite set of basis vectors to a real vector space where geometry (angles, distances, directions) can encode structure.

6 Geometry of the Embedding Space

The embedding space $\mathbb{R}^d$ is equipped with the standard inner product

$$\langle x, y \rangle = x^T y.$$

Similarity is often measured by cosine similarity:

$$\cos\theta = \frac{x^T y}{\|x\| \, \|y\|}.$$

Empirically, tokens that appear in similar contexts tend to have embedding vectors with larger cosine similarity.

7 A Concrete Toy Example: Small Vocabulary, Small Embedding Dimension

To make the embedding map explicit, consider a tiny vocabulary

$$\mathcal{V} = \{\text{cat},\ \text{dog},\ \text{fish},\ \text{runs},\ \text{eats},\ \text{fast}\}, \qquad V = 6,$$

with the index assignment in Table 1.

Token	Index
cat	1
dog	2
fish	3
runs	4
eats	5
fast	6

Table 1: A tiny vocabulary and an indexing scheme.

7.1 One-hot vectors in $\mathbb{R}^6$

Each token corresponds to a canonical basis vector in $\mathbb{R}^6$. For example,

$$e_{\text{cat}} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad e_{\text{dog}} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

All distinct basis vectors are orthogonal, and any two distinct tokens are the same distance apart:

$$\|e_i - e_j\| = \sqrt{2} \quad (i \neq j).$$

So in one-hot space there is no graded notion of similarity.

7.2 Choose a small embedding dimension

Let the embedding dimension be

$$d = 2,$$

so the embedding layer maps $\mathbb{R}^6 \to \mathbb{R}^2$.

7.3 Define an explicit embedding matrix

Define

$$E \in \mathbb{R}^{6\times 2}$$

by

$$E = \begin{pmatrix} 1.0 & 1.0 \\ 1.2 & 1.1 \\ 0.9 & 1.3 \\ -1.0 & -1.1 \\ -1.2 & -1.0 \\ -0.8 & -1.3 \end{pmatrix}.$$

Interpreting rows as token embeddings, we have the mapping in Table 2.

Token	Index	Embedding vector in $\mathbb{R}^2$
cat	1	$(1.0,\ 1.0)$
dog	2	$(1.2,\ 1.1)$
fish	3	$(0.9,\ 1.3)$
runs	4	$(-1.0,\ -1.1)$
eats	5	$(-1.2,\ -1.0)$
fast	6	$(-0.8,\ -1.3)$

Table 2: A hand-chosen embedding matrix E (for illustration). In real LLMs, E is learned.

Here we have intentionally placed "animals" near $(1, 1)$ and "verbs/adverbs" near $(-1, -1)$ to create a simple geometry.

7.4 Embed a prompt

Consider the prompt:

```
dog eats fish fast
```

This corresponds to the token sequence (dog, eats, fish, fast). After embedding, the prompt becomes the matrix

$$X = \begin{pmatrix} 1.2 & 1.1 \\ -1.2 & -1.0 \\ 0.9 & 1.3 \\ -0.8 & -1.3 \end{pmatrix} \in \mathbb{R}^{4\times 2}.$$

7.5 Geometry now encodes similarity

Compute a dot product between two "animal" embeddings:

$$\langle (1.2, 1.1), (0.9, 1.3) \rangle = 1.2(0.9) + 1.1(1.3) = 1.08 + 1.43 = 2.51.$$

Compute a dot product between an "animal" and a "verb" embedding:

$$\langle (1.2, 1.1), (-1.2, -1.0) \rangle = -1.44 - 1.1 = -2.54.$$

In this toy construction, similar categories have positive alignment and dissimilar categories have negative alignment. This is precisely the kind of geometry that training encourages in real models (though in vastly higher dimension).

7.6 A linear-combination step (attention-like behavior)

A crucial point is that later layers form linear combinations of token representations. For example, suppose a layer computes a weighted combination

$$x_{\text{new}} = \sum_{j=1}^{4} \alpha_j v_j, \qquad \alpha = (0.4,\ 0.3,\ 0.2,\ 0.1),$$

where v_j are the four embedded token vectors in the prompt (rows of X). Then

$$x_{\text{new}} = 0.4(1.2, 1.1) + 0.3(-1.2, -1.0) + 0.2(0.9, 1.3) + 0.1(-0.8, -1.3).$$

Compute coordinates:

$$x_{\text{new},1} = 0.48 - 0.36 + 0.18 - 0.08 = 0.22, \qquad x_{\text{new},2} = 0.44 - 0.30 + 0.26 - 0.13 = 0.27.$$

Thus

$$x_{\text{new}} = (0.22,\ 0.27),$$

which is *not* equal to any single token embedding. This illustrates how the model's internal state can move throughout $\mathbb{R}^d$.

Takeaway

In this toy example, $V = 6$ and $d = 2$, so the embedding layer compresses a 6-dimensional canonical-basis representation into a 2-dimensional representation with useful geometry. In real models, V may be on the order of 50,000 while d may be on the order of 10^3.

8 Positional Encoding

Embeddings alone do not encode word order. To incorporate position, define a map

$$P : \mathbb{N} \to \mathbb{R}^d.$$

In the original transformer architecture, positional encodings were sinusoidal:

$$P_{2k}(i) = \sin\left(\frac{i}{10000^{2k/d}}\right), \qquad P_{2k+1}(i) = \cos\left(\frac{i}{10000^{2k/d}}\right).$$

The input representation at position i is then

$$Z_i = E(token_i) + P(i).$$

This can be viewed as adding a multi-frequency Fourier-like basis encoding of discrete position.

9 Scaled Dot-Product Attention

Given $X \in \mathbb{R}^{n \times d}$, define learned projections

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V,$$

where

$$W_Q, W_K, W_V \in \mathbb{R}^{d \times d_k}.$$

Then

$$S = \frac{QK^T}{\sqrt{d_k}}, \qquad S_{ij} = \frac{\langle q_i, k_j \rangle}{\sqrt{d_k}}.$$

Thus S is a scaled Gram matrix. Apply softmax row-wise:

$$A_{ij} = \frac{\exp(S_{ij})}{\sum_j \exp(S_{ij})}.$$

The attention output is

$$\text{Attention}(X) = AV.$$

Each token representation becomes a weighted linear combination of value vectors.

10 Attention as a Kernel Operator

Attention resembles a kernel method. One may define a kernel

$$K(x_i, x_j) = \exp\left(\frac{\langle W_Q x_i, W_K x_j \rangle}{\sqrt{d_k}} \right),$$

so that the output at position i is a similarity-weighted aggregation of transformed vectors at other positions. This is one reason attention can be interpreted as a learned similarity-based integral operator acting on a sequence.

11 High-Dimensional Geometry

A central reason embeddings and dot products are effective is high-dimensional geometry. Let $x, y \sim \mathcal{N}(0, I_d)$. Then

$$\mathbb{E}[\langle x, y \rangle] = 0, \qquad \text{Var}(\langle x, y \rangle) = d.$$

Therefore the scaled dot product $\langle x, y \rangle / \sqrt{d}$ has variance $\mathcal{O}(1)$. In high dimension:

- random vectors are nearly orthogonal,
- concentration of measure occurs,
- linear separability becomes easier.

These facts motivate the $\frac{1}{\sqrt{d_k}}$ scaling in attention.

12 Nonlinear Layers

A transformer block typically includes:

1. multi-head attention,
2. a feedforward network
$$\mathrm{FFN}(x) = \sigma(xW_1 + b_1)W_2 + b_2,$$
3. residual connections,
4. layer normalization.

Thus the network is a deep composition of affine maps and nonlinearities.

13 Variational Interpretation

Training minimizes the negative log-likelihood

$$\mathcal{L}(\theta) = -\sum_t \log P_\theta(token_t \mid token_{<t}),$$

where θ denotes all parameters (including E, W_Q, W_K, etc.). Gradient descent updates

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}.$$

This resembles a variational principle: parameters are adjusted to minimize a functional over a high-dimensional parameter space.

14 Spectral Structure

The embedding matrix admits an SVD:

$$E = U\Sigma V^T.$$

A rapidly decaying singular spectrum suggests low intrinsic dimension (effective rank $r \ll d$), connecting to PCA, low-rank approximation, and model compression.

15 Conceptual Synthesis

From a mathematical physics perspective, an LLM can be viewed as:

- a state (matrix) evolving in $\mathbb{R}^d$,
- under learned linear operators (projections and attention),
- interspersed with nonlinear transformations,
- optimized by minimizing a loss functional.

Much of what appears as "semantic behavior" is grounded in geometry and optimization.

16 Exercises

Conceptual

1. Why are high-dimensional random vectors nearly orthogonal?
2. Why is the $1/\sqrt{d_k}$ scaling used in attention?
3. In what sense is attention similar to a kernel method?

Mathematical

1. Compute $\mathrm{Var}(\langle x, y \rangle)$ for $x, y \sim \mathcal{N}(0, I_d)$.
2. Prove that applying softmax row-wise produces a row-stochastic matrix.
3. Determine the computational complexity of attention as a function of n and d.

17 References

1. Vaswani et al., *Attention Is All You Need*, 2017.
2. Goodfellow, Bengio, Courville, *Deep Learning*.
3. Murphy, *Probabilistic Machine Learning*.