# Introduction: The Embedding Evolution

## The Problem: Representing Knowledge

How do we represent knowledge in a form that machines can learn from? This question has driven decades of research in artificial intelligence, natural language processing, and knowledge representation.

Traditional approaches have fundamental limitations:

- **Symbolic logic** (1950s-1980s): Precise but brittle, requires hand-crafted rules
- **Vector embeddings** (1990s-2010s): Points in space capture similarity but cannot naturally represent containment
- **Deep embeddings** (2010s-present): Learned points (Word2Vec, BERT, GPT) but still limited to distance-based relationships

The challenge is particularly acute for **hierarchical relationships**: How do we encode that “dog” is contained within “mammal”, which is contained within “animal”? Distance between points cannot capture this containment structure.

## The Evolution: From Points to Regions

The history of embeddings reveals a progression toward richer geometric structures:

**1950s-1980s: Symbolic Representations**

- Knowledge encoded as logical rules and symbols
- Precise but requires manual engineering
- Cannot learn from data

**1990s-2000s: Vector Embeddings**

- Entities as points in Euclidean space
- Distance measures similarity (Word2Vec, GloVe)
- Learned from data but limited to distance-based relationships

**2010s: Deep Embeddings**

- Neural networks learn point embeddings (BERT, GPT, TransE)
- More expressive but still fundamentally point-based
- Cannot naturally represent containment or hierarchy

**2018: Box Embeddings Emerge**

- **Vilnis et al. (2018)**: “Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures”
- First to represent entities as **regions** (boxes) rather than points
- Containment becomes geometric: if box A contains box B, then A subsumes B
- Enables natural representation of hierarchical relationships

**2020: Gumbel Boxes**

- **Dasgupta et al. (2020)**: “Improving Local Identifiability in Probabilistic Box Embeddings” (NeurIPS)
- Probabilistic boundaries using Gumbel distributions
- Solves the local identifiability problem (zero gradients in hard boxes)
- Enables effective gradient-based learning
- Achieves approximately 6 F1 score improvement over smoothed boxes on WordNet hypernym prediction
- The choice of Gumbel distributions is theoretically motivated: they are the only max-stable family with infinite support, providing both smoothness and algebraic closure

**2023-2025: Modern Applications and Extensions**
- **Concept2Box (2023)**: Joint learning of concepts (boxes) and entities (vectors)
- **TransBox (2024)**: OWL ontology embeddings with logical closure
- **RegD (2025)**: Hyperbolic-like expressiveness with Euclidean regions
- **Octagon Embeddings (2024)**: Region-based embeddings with explicit rule representation
- **ExpressivE (2022)**: Hyper-parallelogram embeddings for complex relational patterns
- **Geometric Algebra Embeddings (2020-2024)**: Clifford algebra framework extending quaternions
- **Annular Sector Embeddings (2025)**: Novel geometric structures for relational modeling

## Why Box Embeddings?

Box embeddings solve a fundamental limitation of point-based representations: **points cannot naturally encode containment relationships**.

Consider the hierarchy "dog $\subseteq$ mammal $\subseteq$ animal":
- **Vector embeddings**: Three points in space. Distance measures similarity, but there's no natural way to encode containment.
- **Box embeddings**: Three boxes where "dog" is contained in "mammal", which is contained in "animal". Containment is geometrically explicit and transitive.

The key insight is that **containment is transitive**: if A contains B and B contains C, then A contains C. This matches logical subsumption perfectly: if dog is a mammal and mammal is an animal, then dog is an animal. Box embeddings preserve this transitivity through geometric containment, ensuring that learned hierarchies are consistent and interpretable.

**Lattice-theoretic foundation:** Box embeddings are fundamentally grounded in **lattice theory** and **order theory**. They form a proper **box lattice** under the reverse product order, where the join operation (union) produces the smallest enclosing box and the meet operation (intersection) produces the box intersection. This lattice structure is strictly more general than order embedding lattices, providing the mathematical foundation for understanding box embeddings' expressiveness and their relationship to Boolean algebras and distributive lattices. The **intersectional closure** property—that the intersection of two boxes is itself a box—ensures the lattice remains closed under the meet operation, essential for modeling concept hierarchies.

## Geometric Intuition

Points vs. Regions:

Vector embeddings represent entities as points:
- "dog" = point at (0.3, 0.5, 0.2)
- "mammal" = point at (0.4, 0.6, 0.3)
- "animal" = point at (0.5, 0.7, 0.4)

Distance measures similarity, but there's no natural way to encode "dog $\subseteq$ mammal $\subseteq$ animal".

Box embeddings represent entities as regions:
- "dog" = box from (0.2, 0.4) to (0.4, 0.6)
- "mammal" = box from (0.1, 0.3) to (0.5, 0.7) $\longleftarrow$ contains "dog"
- "animal" = box from (0.0, 0.0) to (1.0, 1.0) $\longleftarrow$ contains "mammal" (and thus "dog")

Containment is geometrically explicit: you can visually see that "dog" is inside "mammal", which is inside "animal".

Volume as Granularity:

The volume of a box has semantic meaning:
- **Large volume** = general concept (e.g., “animal” covers many species)
- **Small volume** = specific concept (e.g., “dog” is a specific species)
- **Nested volumes** = hierarchical relationships (parent concepts have larger volumes than children)

This is intuitive: more general concepts “cover more ground” in the embedding space.

## The Subsumption Connection

The term “subsume” comes from formal logic and automated reasoning. In logic, one statement **subsumes** another when it is more general and covers all cases that the more specific statement would cover.

This is exactly what box embeddings model:
- **Logical subsumption**: Statement A subsumes statement B if A is more general
- **Geometric subsumption**: Box A subsumes box B if A contains B

The connection is direct:
- If “animal” subsumes “mammal” logically, then the “animal” box should contain the “mammal” box geometrically
- If “mammal” subsumes “dog” logically, then the “mammal” box should contain the “dog” box geometrically
- By transitivity, “animal” subsumes “dog” both logically and geometrically

This makes box embeddings particularly well-suited for:
- **Formal ontologies** (OWL, Description Logic)
- **Knowledge bases** with hierarchical structure
- **Natural language** with hyponym-hypernym relationships
- **Automated reasoning** where subsumption is fundamental

## What Follows

This document series provides the mathematical foundations of box embeddings:

1. **Subsumption**: Geometric containment as logical subsumption
2. **Gumbel Box Volume**: Expected volume computation using Bessel functions
3. **Containment Probability**: First-order approximation for probabilistic containment
4. **Gumbel Max-Stability**: Why intersections preserve the Gumbel family
5. **Log-Sum-Exp and Intersection**: Numerical stability and the Gumbel-Max property
6. **Local Identifiability**: The learning problem and Gumbel solution

Each document follows a textbook-style structure: Definition → Statement → Proof → Example. Together, they provide a complete mathematical foundation for understanding and implementing box embeddings.

## Historical Context: The Path to Gumbel Boxes

The evolution from hard boxes to Gumbel boxes was motivated by a fundamental learning problem:

**Hard Boxes (2018)**: Deterministic intervals with sharp boundaries. Simple and interpretable, but suffer from zero gradients when boxes are disjoint or contained—the optimizer has no signal to guide learning.

**Smooth Boxes (2019)**: Gaussian noise added to boundaries. Solves the gradient problem, but breaks max-stability—intersections of Gaussian boxes are not Gaussian, making volume calculations intractable.

**Gumbel Boxes (2020)**: Probabilistic boundaries using Gumbel distributions. Combines the best of both worlds: smooth gradients (enables learning) and max-stability (preserves mathematical structure). This is why Gumbel distributions, rather than other smooth distributions, are used in box embeddings.

The choice of Gumbel distributions is not arbitrary—they appear naturally in extreme value theory as the limiting distribution of maxima (or minima), making them the "natural" choice for modeling extreme coordinates (minimum and maximum) that define box boundaries. The Fisher-Tippett-Gnedenko theorem (1928-1943) establishes that there are only three possible limiting distributions for normalized maxima: Gumbel (Type I), Fréchet (Type II), and Weibull (Type III). Among these, only the Gumbel distribution is max-stable with scale preservation—the maximum of independent Gumbel random variables remains Gumbel-distributed with the same scale parameter, which is essential for maintaining mathematical consistency in box operations (see the Gumbel Max-Stability document).

## Modern Relevance

Box embeddings are actively used in:

- **Knowledge graph completion**: Predicting missing relationships in knowledge graphs
- **Ontology embedding**: Encoding formal ontologies (OWL, Description Logic) for reasoning
- **Natural language understanding**: Modeling hyponym-hypernym relationships
- **Healthcare and bioinformatics**: Embedding medical ontologies (SNOMED CT, UMLS)
- **Two-view knowledge graphs**: Joint learning of concepts (boxes) and entities (vectors)

Recent work (2023-2025) continues to validate and extend box embeddings, demonstrating their effectiveness and exploring new applications. See the Applications document for details on modern developments.

## Next Steps

To understand the mathematical foundations, proceed to the Subsumption document, which establishes the fundamental connection between geometric containment and logical subsumption. The documents build on each other, so reading in order is recommended.

For a quick reference of formulas, see the Mathematical Foundations guide. For implementation details, see the code documentation.