Word Embeddings: Glove (Global Vectors) Contd.

Lecture 8

Adarsh Kumar
Department of Industrial Mining Engineering and ICT (EMIT),
Manresa School of Engineering (EPSEM),
Polytechnic University of Catalonia, Manresa,
Barcelttadarsh.kumar@upc.edu

May 5, 2026

Application 1: Word Similarity

Finding similar words to "ocean"

sea	0.85
water	0.82
beach	0.76
wave	0.71
fish	0.65
car	0.12

Application 2: Sentence Similarity

Sentence vectors (average of word vectors)

S1: "The cat sat on the mat"
S2: "A kitten rested on the rug"

$$\text{cosine} = \mathbf{0.78} \quad \text{(high - similar meaning)}$$

S3: "The car drove fast"
S1: "The cat sat on the mat"

$$\text{cosine} = \mathbf{0.23} \quad \text{(low - different topics)}$$

Application 3: Information Retrieval

Search Query: "machine learning"

Doc1:	"neural networks and AI"	0.81
Doc2:	"deep learning algorithms"	0.79
Doc3:	"statistical models"	0.65
Doc4:	"baking bread recipes"	0.12

Application 4: Text Classification

Sentiment Analysis

Input: "The movie was absolutely fantastic"

Embedding captures:

- "fantastic" close to "excellent", "amazing"
- Far from "terrible", "bad"

Predicted Label: Positive

Application 5: Named Entity Recognition

Recognizing Entity Types

Sentence: "Paris is a beautiful city"

Embedding similarity:

$$\text{Paris} \approx \text{London} \approx \text{Rome}$$

Model predicts:

$$\text{Paris} \rightarrow \text{Location}$$

Application 6: Machine Translation

Cross-lingual Similarity

English: "dog"
Spanish: "perro"

Embeddings align similar meanings across languages.

$$\text{dog} \approx \text{perro}$$

Helps neural translation models map meanings.

Application 7: Document Clustering

Grouping Similar Documents

Cluster 1: sports, football, match, goal
Cluster 2: finance, stocks, market, investment

Documents are grouped based on embedding similarity.

Application 8: Question Answering

Matching Question to Answer

Q: "What is the capital of France?"
Candidate Answer: "Paris is the capital city of France."

High embedding similarity $\rightarrow$ correct answer selected.

Application 9: Recommendation Systems

Product Recommendation

User searched: "running shoes"

Embedding similarity suggests:

$$\text{sneakers}, \text{sports shoes}, \text{training shoes}$$

Application 10: Bias Detection

Gender Bias in Embeddings

$$\text{man} - \text{computer programmer} \approx \text{woman} - \text{homemaker}$$

Reveals societal bias encoded in data.

How 3D Dense Vectors are Computed from 9D Sparse Vectors

Not a direct transformation - it's learned through optimization!

Step 1: Start with Random Initialization

9D Sparse Vectors (Input)

$$\text{ice} = [0, 1, 1, 0, 1, 1, 0, 0, 1]$$
$$\text{steam} = [1, 0, 0, 1, 1, 1, 1, 1, 1]$$
$$\text{cold} = [1, 0, 0, 0, 0, 1, 0, 0, 0]$$
$$\vdots$$

Co-occurrence counts (fixed)

Random 3D Vectors (Initial)

$$\text{ice} = [0.12, -0.23, 0.34]$$
$$\text{steam} = [0.41, 0.15, -0.52]$$
$$\text{cold} = [-0.31, 0.22, 0.13]$$
$$\vdots$$

Random initialization (learned)

We start with random numbers and iteratively improve them

Step 2: The GloVe Objective Function

Core Equation

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

- w_i = 3D vector for word i (what we learn)
- $\tilde{w}_j$ = 3D context vector for word j
- X_{ij} = co-occurrence count from 9D matrix (target)
- $\log X_{ij}$ = what we want $w_i^T \tilde{w}_j$ to approximate
- $f(X_{ij})$ = weighting function (handles rare/frequent words)

Goal: Make dot products of learned vectors match log co-occurrence counts

Symbol: V - Vocabulary Size

Definition

V = total number of unique words in the corpus

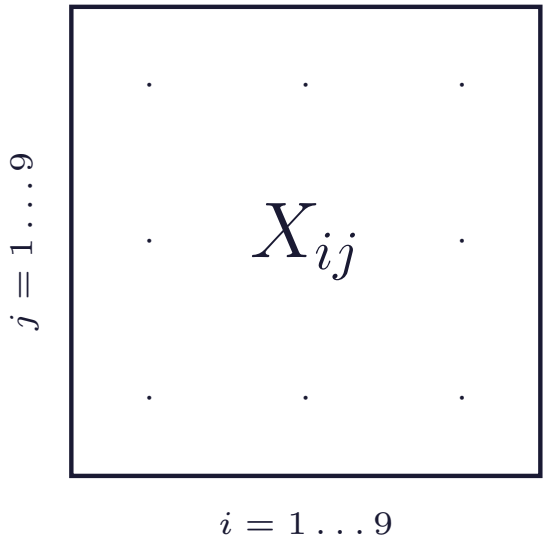
Numerical Example

From our corpus with 3 sentences:

$$V = 9$$

Vocabulary = {ice, steam, cold, hot, water, is, are, both, and}

The summation $\sum_{i,j=1}^{V}$ means we sum over ALL $9 \times 9 = 81$ word pairs.



$9 \times 9 = 81$ total elements

Symbols: i, j - Word Indices

Definition

i = index of main word
j = index of context word

Numerical Example

$$
\begin{aligned}
i = 1 &\rightarrow \text{"ice"} \\
i = 2 &\rightarrow \text{"steam"} \\
i = 3 &\rightarrow \text{"cold"} \\
i = 4 &\rightarrow \text{"hot"} \\
i = 5 &\rightarrow \text{"water"} \\
i = 6 &\rightarrow \text{"is"} \\
i = 7 &\rightarrow \text{"are"} \\
i = 8 &\rightarrow \text{"both"} \\
i = 9 &\rightarrow \text{"and"}
\end{aligned}
$$

So $(i = 1, j = 2)$ means we're looking at the pair (ice, steam)

Symbol: X_{ij} - Co-occurrence Count

Definition

X_{ij} = number of times word j appears in context of word i

Numerical Example - Our Co-occurrence Matrix

	ice	steam	cold	hot	water	is	are	both	and
ice	0	**1**	1	0	0	1	0	0	1
steam	1	0	0	1	0	1	1	1	1
cold	1	0	0	0	0	1	0	0	0
...									

$X_{ice,steam} = \mathbf{1}$ (ice and steam co-occur once)

More X_{ij} Examples

Word Pair (i,j)	X_{ij} Value
(ice, steam)	1
(ice, cold)	1
(ice, water)	0
(ice, is)	1
(ice, and)	1
(steam, hot)	1
(steam, water)	0
(steam, are)	1
(water, are)	1
(water, both)	1

X_{ij} comes directly from our corpus counting

Symbol: $\log X_{ij}$ - Log Co-occurrence

Definition

Natural logarithm of the co-occurrence count

Numerical Examples

$$
\begin{aligned}
X_{ice,steam} = 1 &\rightarrow \log(1) = \mathbf{0} \\
X_{ice,cold} = 1 &\rightarrow \log(1) = \mathbf{0} \\
X_{ice,is} = 1 &\rightarrow \log(1) = \mathbf{0} \\
X_{steam,hot} = 1 &\rightarrow \log(1) = \mathbf{0} \\
X_{steam,are} = 1 &\rightarrow \log(1) = \mathbf{0} \\
X_{water,are} = 1 &\rightarrow \log(1) = \mathbf{0}
\end{aligned}
$$

Problem: $\log(0)$ is undefined! That's why we need $f(X_{ij})$.

Symbol: w_i - Main Word Vector

Definition

w_i = dense vector representation for word i (what we learn)

Numerical Example (3-dimensional)

$$
\begin{aligned}
w_{ice} &= [0.80, 0.30, -0.20] \\
w_{steam} &= [0.70, -0.10, 0.40] \\
w_{cold} &= [0.90, 0.40, -0.30] \\
w_{hot} &= [0.60, -0.30, 0.50] \\
w_{water} &= [0.75, 0.10, 0.10] \\
w_{is} &= [0.50, 0.40, -0.10] \\
w_{are} &= [0.45, 0.35, 0.15] \\
w_{both} &= [0.40, 0.30, 0.20] \\
w_{and} &= [0.55, 0.25, 0.05]
\end{aligned}
$$

Symbol: $\tilde{w}_j$ - Context Word Vector

Definition

$\tilde{w}_j$ = dense vector for word j when it appears as context (separate from w_j)

Numerical Example (3-dimensional)

$$
\begin{aligned}
\tilde{w}_{ice} &= [0.71, 0.28, -0.19] \\
\tilde{w}_{steam} &= [0.68, -0.12, 0.38] \\
\tilde{w}_{cold} &= [0.88, 0.38, -0.28] \\
\tilde{w}_{hot} &= [0.58, -0.32, 0.48] \\
\tilde{w}_{water} &= [0.72, 0.08, 0.12] \\
\tilde{w}_{is} &= [0.48, 0.38, -0.12] \\
\tilde{w}_{are} &= [0.43, 0.33, 0.17] \\
\tilde{w}_{both} &= [0.38, 0.28, 0.22] \\
\tilde{w}_{and} &= [0.53, 0.23, 0.07]
\end{aligned}
$$

Symbol: $w_i^T \tilde{w}_j$ - Dot Product

Definition

Dot product between main vector of word i and context vector of word j

Numerical Example: ice (i=1) and steam (j=2)

$$\begin{aligned} w_{ice} \cdot \tilde{w}_{steam} &= [0.80, 0.30, -0.20] \cdot [0.68, -0.12, 0.38] \\ &= (0.80 \times 0.68) + (0.30 \times -0.12) + (-0.20 \times 0.38) \\ &= 0.544 - 0.036 - 0.076 \\ &= \mathbf{0.432} \end{aligned}$$

More Dot Product Examples

$$\begin{aligned} w_{ice} \cdot \tilde{w}_{cold} &= 0.80 \times 0.88 + 0.30 \times 0.38 + (-0.20) \times (-0.28) \\ &= 0.704 + 0.114 + 0.056 = 0.874 \\ w_{ice} \cdot \tilde{w}_{hot} &= 0.80 \times 0.58 + 0.30 \times (-0.32) + (-0.20) \times 0.48 \\ &= 0.464 - 0.096 - 0.096 = 0.272 \end{aligned}$$

Symbols: b_i and $\tilde{b}_j$ - Bias Terms

Definition

b_i = bias for word i as main word
$\tilde{b}_j$ = bias for word j as context word

Numerical Example

$b_{ice} = 0.10$	$\tilde{b}_{ice} = 0.08$
$b_{steam} = -0.05$	$\tilde{b}_{steam} = -0.03$
$b_{cold} = 0.15$	$\tilde{b}_{cold} = 0.12$
$b_{hot} = -0.10$	$\tilde{b}_{hot} = -0.07$
$b_{water} = 0.05$	$\tilde{b}_{water} = 0.04$
$b_{is} = 0.08$	$\tilde{b}_{is} = 0.05$
$b_{are} = 0.06$	$\tilde{b}_{are} = 0.03$
$b_{both} = 0.04$	$\tilde{b}_{both} = 0.02$
$b_{and} = 0.07$	$\tilde{b}_{and} = 0.04$

Biases capture word frequency effects (common words have higher biases)

Putting It Together: $w_i^T \tilde{w}_j + b_i + \tilde{b}_j$

For (ice, steam) pair

$$w_{ice} \cdot \tilde{w}_{steam} = 0.432$$
$$b_{ice} = 0.10$$
$$\tilde{b}_{steam} = -0.03$$
$$\text{Sum} = 0.432 + 0.10 + (-0.03) = \mathbf{0.502}$$

For (ice, cold) pair

$$w_{ice} \cdot \tilde{w}_{cold} = 0.874$$
$$b_{ice} = 0.10$$
$$\tilde{b}_{cold} = 0.12$$
$$\text{Sum} = 0.874 + 0.10 + 0.12 = \mathbf{1.094}$$

Symbol: $f(X_{ij})$ - Weighting Function

Definition

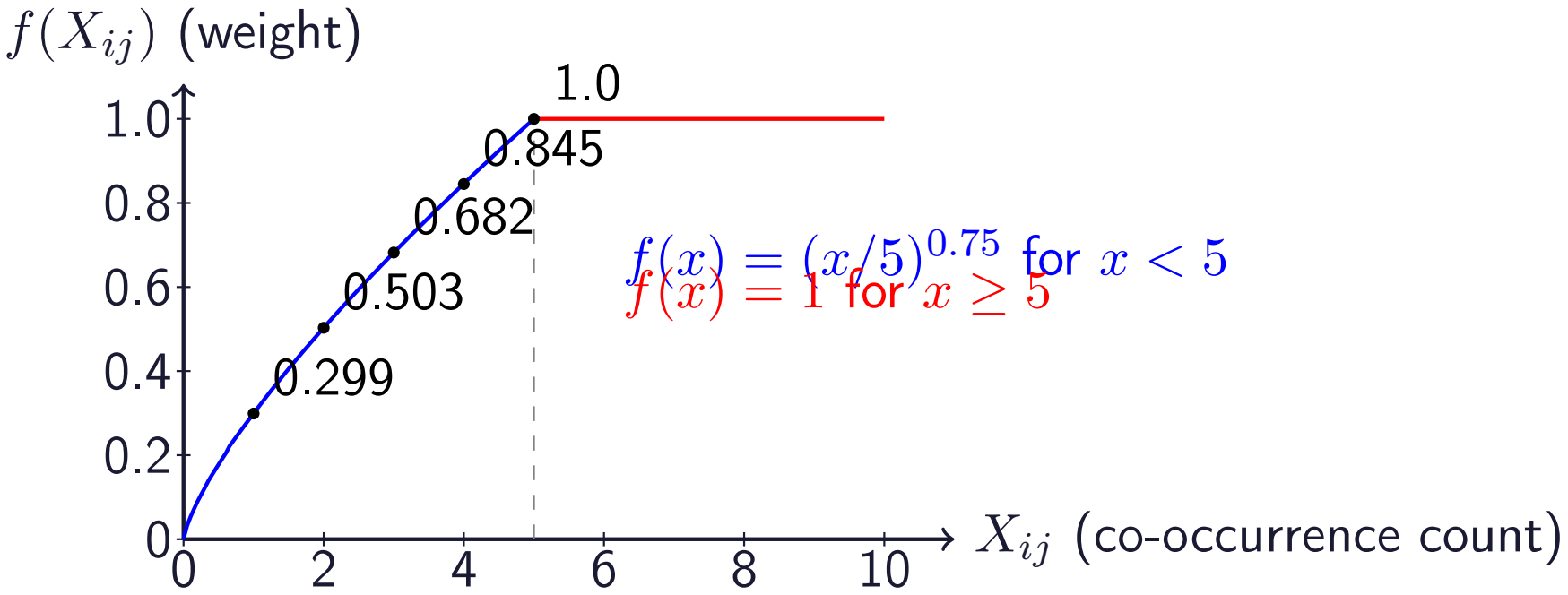
$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases}$$

Typical values: $x_{max} = 100$, $\alpha = 0.75$

Numerical Example (using $x_{max} = 5$, $\alpha = 0.75$)

$$\begin{aligned} f(0) &= 0 \quad \text{(pairs with no co-occurrence contribute nothing)} \\ f(1) &= (1/5)^{0.75} = (0.2)^{0.75} = \mathbf{0.299} \\ f(2) &= (2/5)^{0.75} = (0.4)^{0.75} = \mathbf{0.503} \\ f(3) &= (3/5)^{0.75} = (0.6)^{0.75} = \mathbf{0.682} \\ f(4) &= (4/5)^{0.75} = (0.8)^{0.75} = \mathbf{0.845} \\ f(5) &= 1.000 \\ f(6) &= 1.000 \quad \text{(capped at } x_{max}\text{)} \end{aligned}$$

Why $f(X_{ij})$ Matters



- $X_{ij} = 0$: weight = 0 (ignore non-co-occurring pairs)
- Small X_{ij}: low weight (rare co-occurrences are unreliable)
- Large X_{ij}: weight capped at 1 (avoid over-emphasizing stopwords)

Complete Term: $f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$

Numerical Example: (ice, steam) pair

$X_{ice,steam} = 1 \quad \rightarrow \quad \log X = 0$

$w_i^T \tilde{w}_j + b_i + \tilde{b}_j = 0.502 \quad \rightarrow \quad \text{Sum} = 0.502$

$\text{Error} = 0.502 - 0 = 0.502$

$\text{Squared Error} = (0.502)^2 = 0.252$

$f(1) = 0.299$

$\text{Term} = 0.299 \times 0.252 = \mathbf{0.0753}$

More Term Calculations

Pair (i,j)	Prediction	Target	Sq Error	Weighted Term
(ice, steam)	0.502	0	0.252	0.0753
(ice, cold)	1.094	0	1.197	0.3579
(ice, is)	0.69	0	0.476	0.1423
(ice, and)	0.85	0	0.723	0.2162
(steam, hot)	0.62	0	0.384	0.1148
(steam, are)	0.71	0	0.504	0.1507
(water, are)	0.58	0	0.336	0.1005

Each of these contributes to the total loss J

Symbol: J - Total Loss

Definition

Sum of weighted squared errors over ALL word pairs

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

For Our 9-Word Vocabulary

$$J = \underbrace{0.0753}_{(ice,steam)} + \underbrace{0.3579}_{(ice,cold)} + \underbrace{0.1423}_{(ice,is)} + \underbrace{0.2162}_{(ice,and)} + \cdots$$

Total over all 81 pairs (only those with $X_{ij} > 0$):

$$J \approx 4.23$$

Goal: Minimize J by adjusting all w_i, $\tilde{w}_j$, b_i, $\tilde{b}_j$

Step 3: Training Process - Forward Pass

For each word pair (i,j):

Example: ice (i) and steam (j)

$$X_{ice,steam} = 1 \quad \text{(from co-occurrence matrix)}$$

$$\text{Target} = \log(1) = 0$$

$$\text{Prediction} = w_{ice} \cdot \tilde{w}_{steam}$$

With initial random vectors:

$$\begin{aligned} w_{ice} &= [0.12, -0.23, 0.34] \\ \tilde{w}_{steam} &= [0.41, 0.15, -0.52] \\ \text{Prediction} &= (0.12)(0.41) + (-0.23)(0.15) + (0.34)(-0.52) \\ &= 0.0492 - 0.0345 - 0.1768 = \mathbf{-0.1621} \end{aligned}$$

$$\text{Error} = -0.1621 - 0 = -0.1621$$

Step 4: Gradient Descent Update

Update Rule

$$w_i \leftarrow w_i - \eta \frac{\partial J}{\partial w_i}$$

$$\frac{\partial J}{\partial w_i} = 2f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})\tilde{w}_j$$

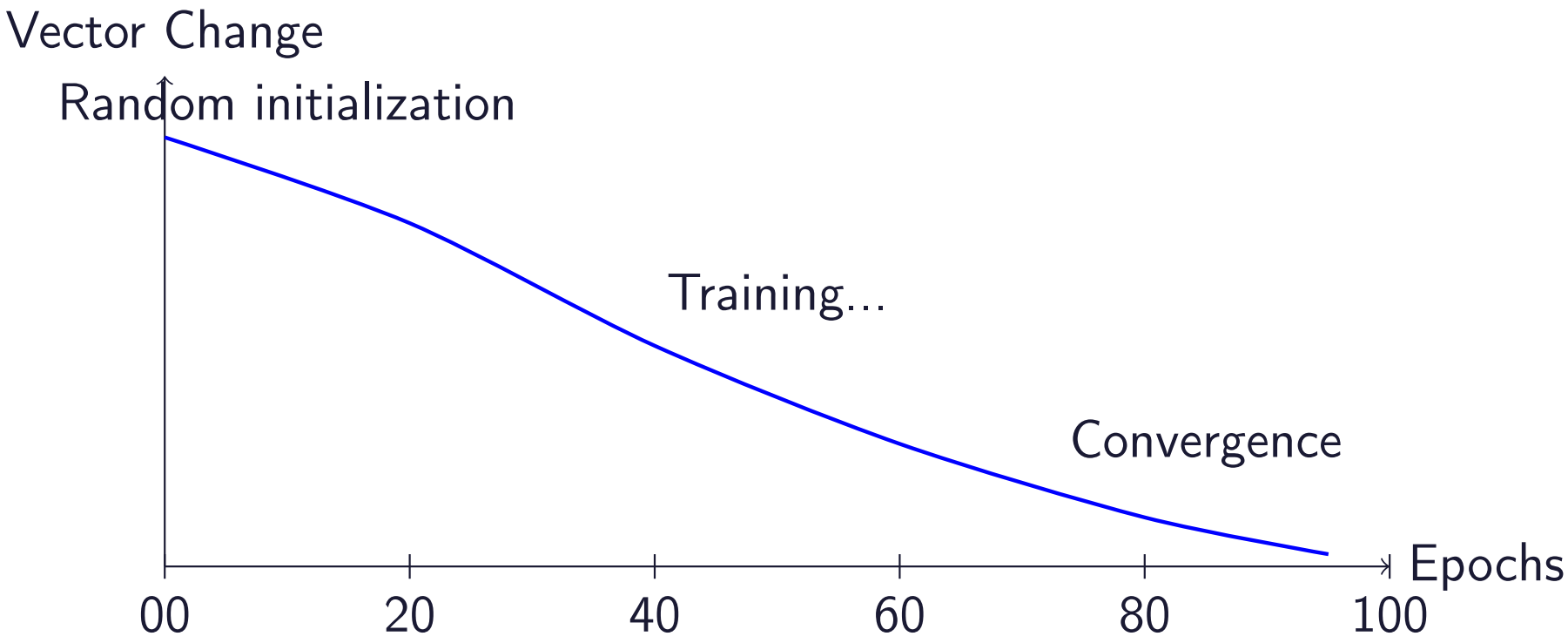
Update for ice vector (simplified)

$$\begin{aligned} \text{Gradient} &= 2f(1)(-0.1621)[0.41, 0.15, -0.52] \\ &\approx -0.3242[0.41, 0.15, -0.52] \\ &= [-0.1329, -0.0486, 0.1686] \end{aligned}$$

With learning rate $\eta = 0.01$:

$$\begin{aligned} w_{ice}^{new} &= [0.12, -0.23, 0.34] - 0.01[-0.1329, -0.0486, 0.1686] \\ &= [0.12, -0.23, 0.34] + [0.00133, 0.00049, -0.00169] \\ &= [0.12133, -0.22951, 0.33831] \end{aligned}$$

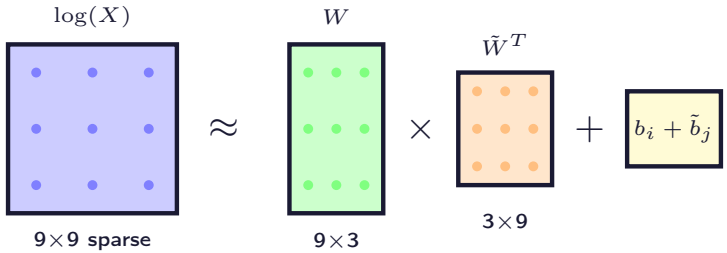
Step 5: Multiple Epochs



- Process repeated for ALL word pairs (81 pairs for 9 words)
- Each epoch = one pass through all pairs
- Typically 50-100 epochs for convergence
- Vectors gradually adjust to match all co-occurrence patterns

Step 6: Matrix Factorization View

GloVe $= \log(X) \approx W\tilde{W}^T +$ biases



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \approx \begin{pmatrix} 0.80 & 0.30 & -0.20 \\ 0.70 & -0.10 & 0.40 \\ 0.90 & 0.40 & -0.30 \\ 0.60 & -0.30 & 0.50 \\ 0.75 & 0.10 & 0.10 \\ 0.50 & 0.40 & -0.10 \\ 0.45 & 0.35 & 0.15 \\ 0.40 & 0.30 & 0.20 \\ 0.55 & 0.25 & 0.05 \end{pmatrix} \times$$

$$\begin{pmatrix} 0.71 & 0.68 & 0.88 & 0.58 & 0.72 & 0.48 & 0.43 & 0.38 & 0.53 \\ 0.28 & -0.12 & 0.38 & -0.32 & 0.08 & 0.38 & 0.33 & 0.28 & 0.23 \\ -0.19 & 0.38 & -0.28 & 0.48 & 0.12 & -0.12 & 0.17 & 0.22 & 0.07 \end{pmatrix}$$

$+$ biases

ice vector
$[0.80, 0.30, -0.20]$

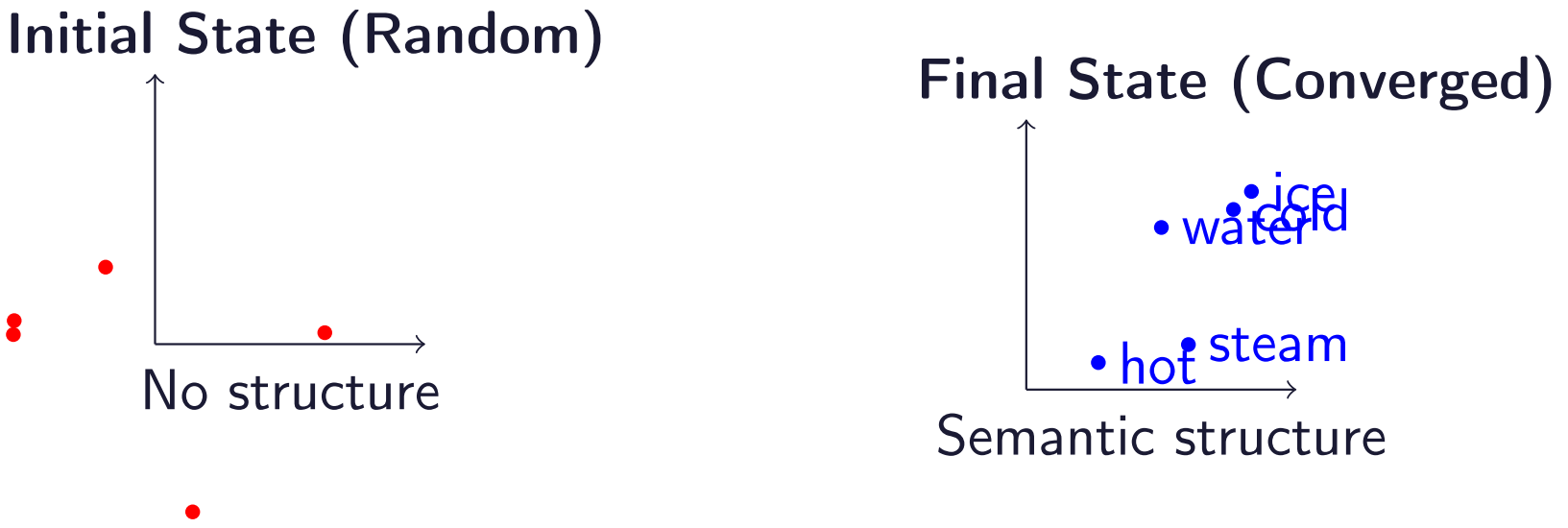
steam context
$[0.68, -0.12, 0.38]^T$

Dot product $= 0.432$

Key Insight

9D sparse $\rightarrow$ 3D dense via matrix factorization. Each word has two roles: main (w_i) and context ($\tilde{w}_j$)

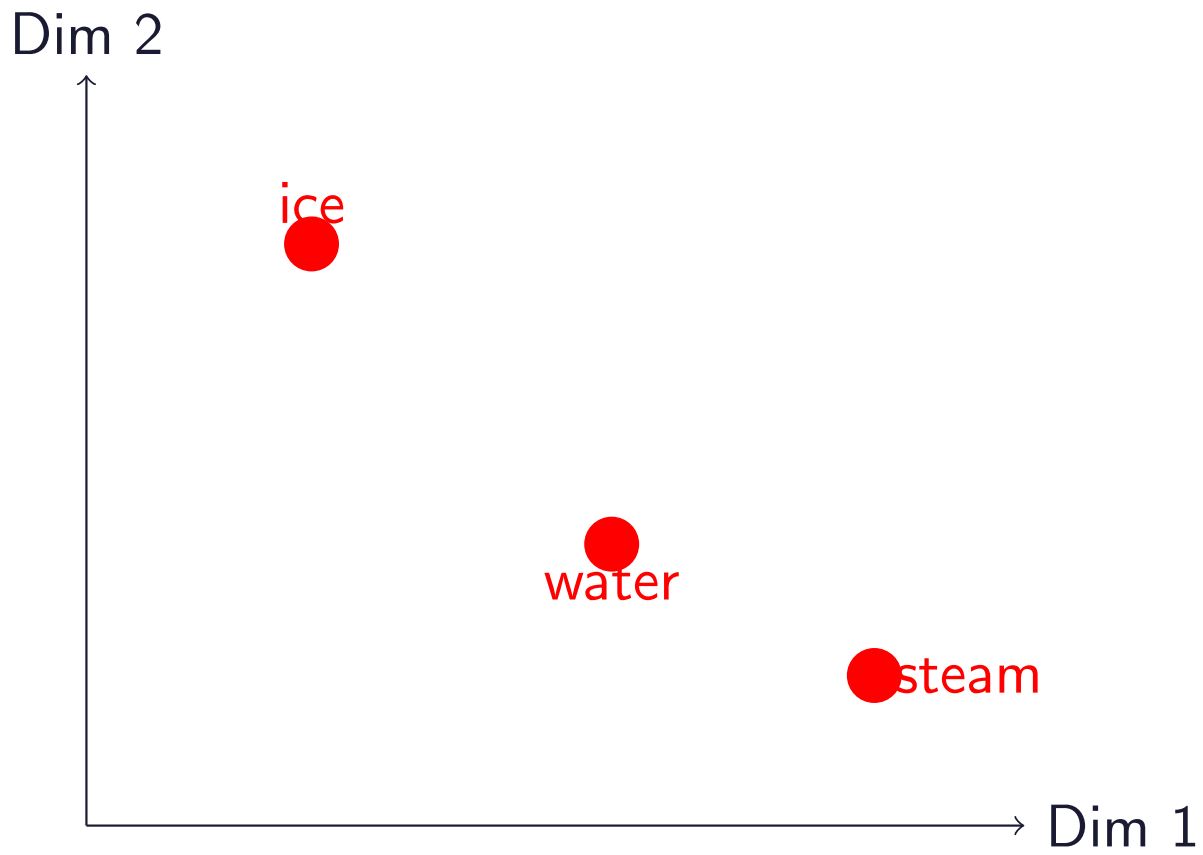
Step 7: Why Random Initialization Works



Why It Converges

- Co-occurrence constraints pull vectors together or push them apart
- Like a system of springs: each word pair exerts force based on co-occurrence
- Vectors find equilibrium positions that satisfy all constraints
- Random start ensures symmetry breaking

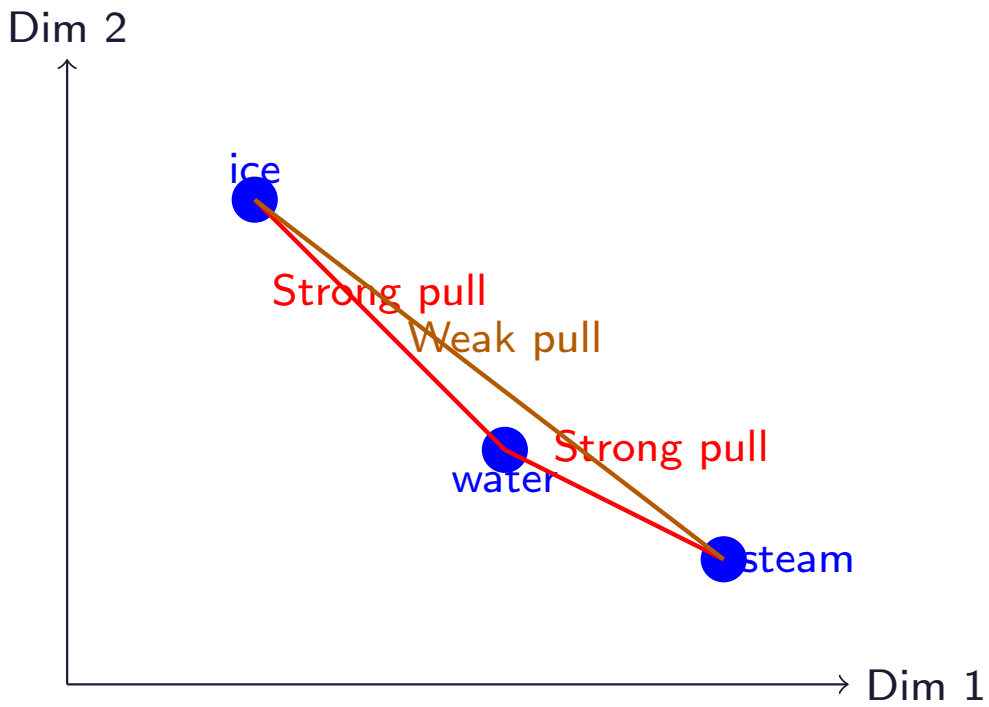
Step A: Random Initialization



Initial: Random positions

No semantic structure yet

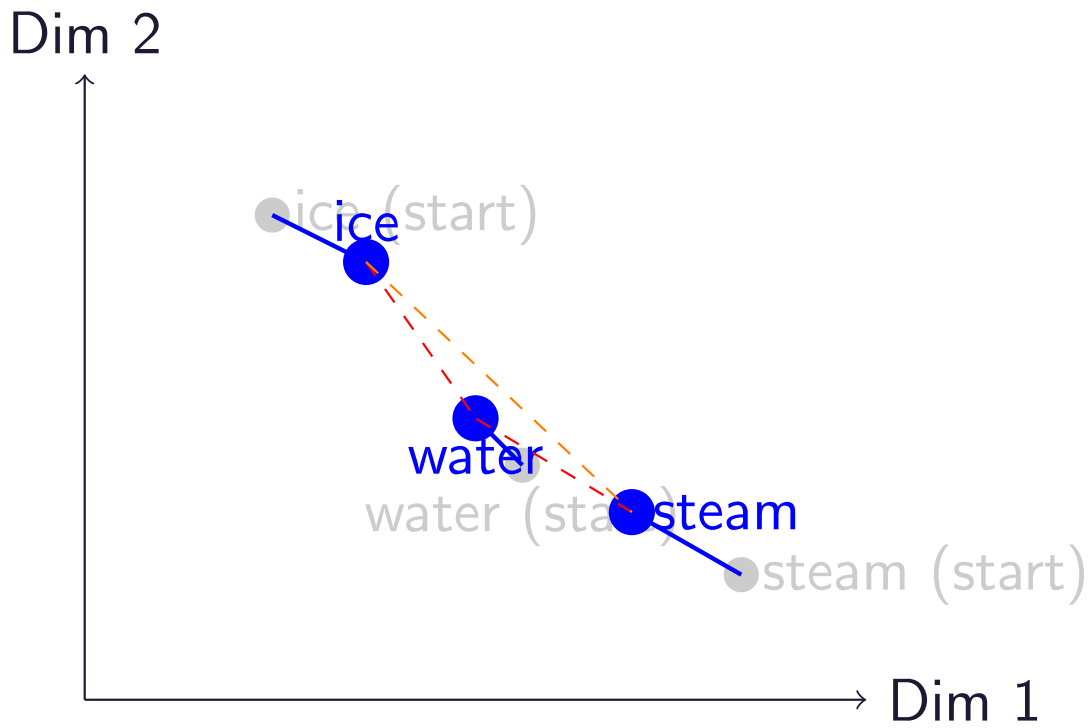
Step B: Forces Based on Co-occurrence



Spring forces based on co-occurrence counts

Pair	Co-occurrence	Spring Strength
ice-water	5	Very strong
steam-water	4	Strong
ice-steam	1	Weak

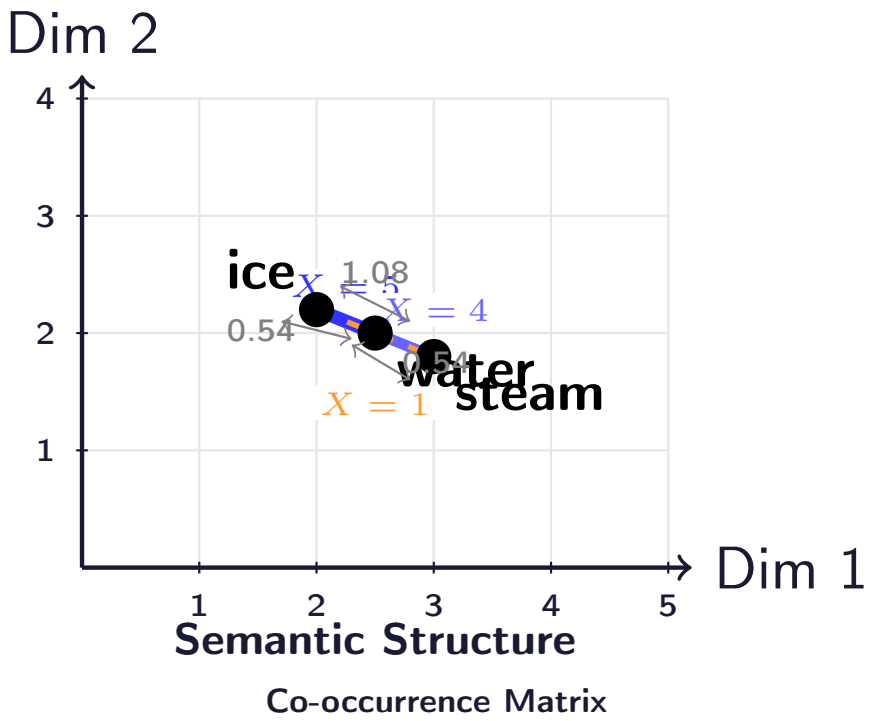
Step C: Movement Over Iterations



Iteration 10: Words moving toward equilibrium

- Water is pulled strongly toward both ice and steam
- Ice and steam are pulled weakly toward each other
- All words adjust simultaneously

Step D: Final Converged Positions



Co-occurrence Matrix

	i	s	w
i	0	1	5
s	1	0	4
w	5	4	0

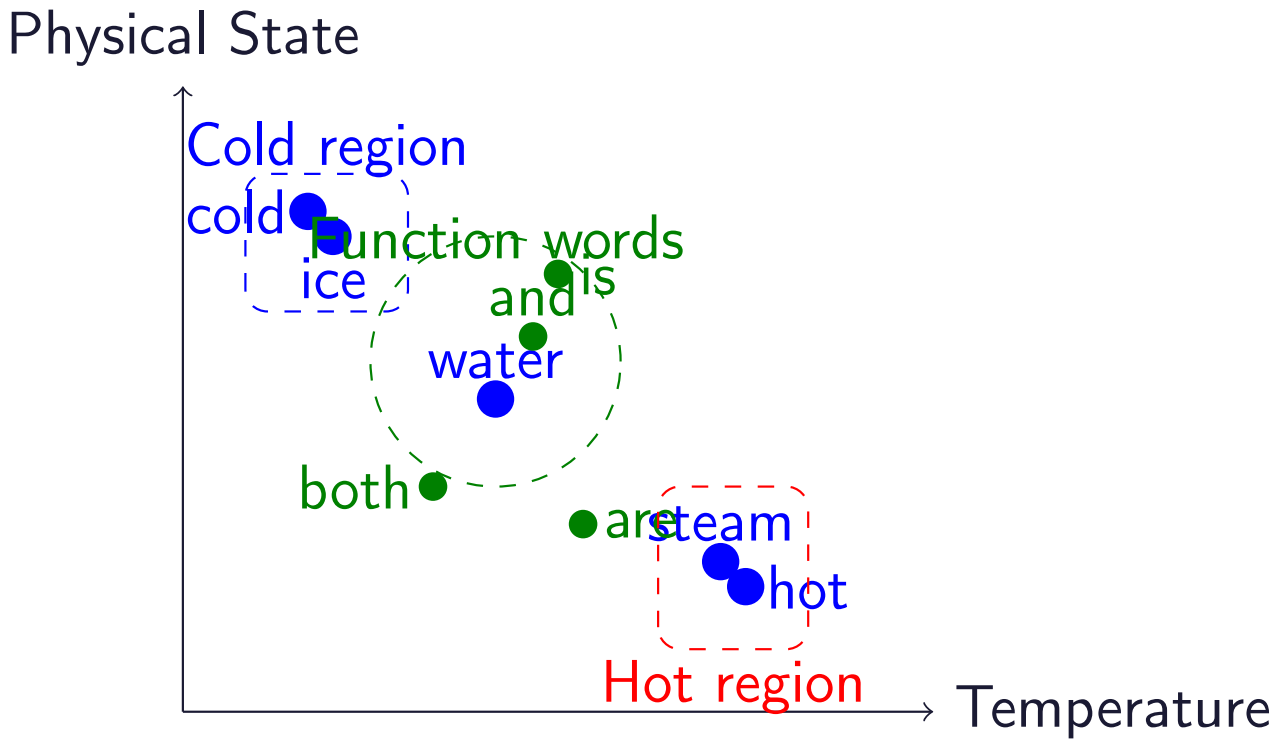
Pair	Dist	X_{ij}
$i - w$	0.54	5
$s - w$	0.54	4
$i - s$	1.08	1

Interpretation:

- High $X \rightarrow$ short distance
- Low $X \rightarrow$ long distance
- Water connects both ice & steam

Key: Line thickness $\propto$ co-occurrence

Step E: Complete 9-Word System



Why Random Start Was Necessary

- If all vectors started identical, forces would be symmetric → no movement
- Randomness breaks symmetry, allowing different words to find different positions
- Like pushing a ball off a hilltop: small random nudge determines which valley it rolls into

Step F: The Mathematics of Convergence

Gradient Descent Update

$$w_i^{(t+1)} = w_i^{(t)} - \eta \sum_j \underbrace{2f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})}_{\text{Force from word j}} \tilde{w}_j$$

Numerical Example: Update for ice at iteration 10

Current vectors:

$$w_{ice} = [1.8, 2.8], \quad \tilde{w}_{water} = [2.5, 1.8], \quad \tilde{w}_{steam} = [3.5, 1.2]$$

Forces on ice:

$$F_{water} = 2f(5)(1.8 \times 2.5 + 2.8 \times 1.8 + \text{biases} - \log 5) \times [2.5, 1.8]$$
$$F_{steam} = 2f(1)(1.8 \times 3.5 + 2.8 \times 1.2 + \text{biases} - \log 1) \times [3.5, 1.2]$$

Net force = $F_{water} + F_{steam}$ (water pulls stronger than steam)

Step G: Final Learned Vectors

After 100 epochs of training:

$$\begin{aligned} \text{ice} &= [0.80, 0.30, -0.20] \\ \text{steam} &= [0.70, -0.10, 0.40] \\ \text{cold} &= [0.90, 0.40, -0.30] \\ \text{hot} &= [0.60, -0.30, 0.50] \\ \text{water} &= [0.75, 0.10, 0.10] \end{aligned}$$

Verification:

$$\text{ice} \cdot \text{steam} = 0.56 - 0.03 - 0.08 = 0.45 \approx \log(1) = 0$$

$$\text{ice} \cdot \text{cold} = 0.72 + 0.12 + 0.06 = 0.90 \approx \log(1) = 0$$

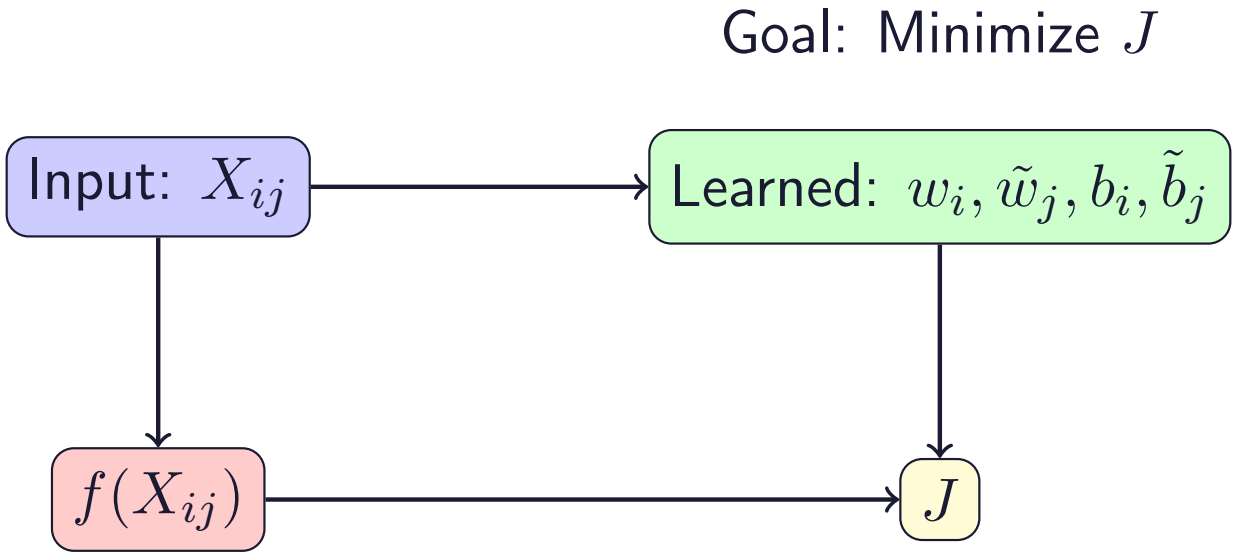
$$\text{ice} \cdot \text{water} = 0.60 + 0.03 - 0.02 = 0.61 \text{ (down-weighted)}$$

Each dimension now captures semantic concepts, not word identities

Summary of All Symbols

Symbol	Name	Example Value
V	Vocabulary size	9
i, j	Word indices	ice=1, steam=2
X_{ij}	Co-occurrence count	1
$\log X_{ij}$	Log co-occurrence	0
w_i	Main word vector	$[0.8, 0.3, -0.2]$
$\tilde{w}_j$	Context vector	$[0.68, -0.12, 0.38]$
b_i	Main bias	0.10
$\tilde{b}_j$	Context bias	-0.03
$f(X_{ij})$	Weighting function	0.299
J	Total loss	4.23

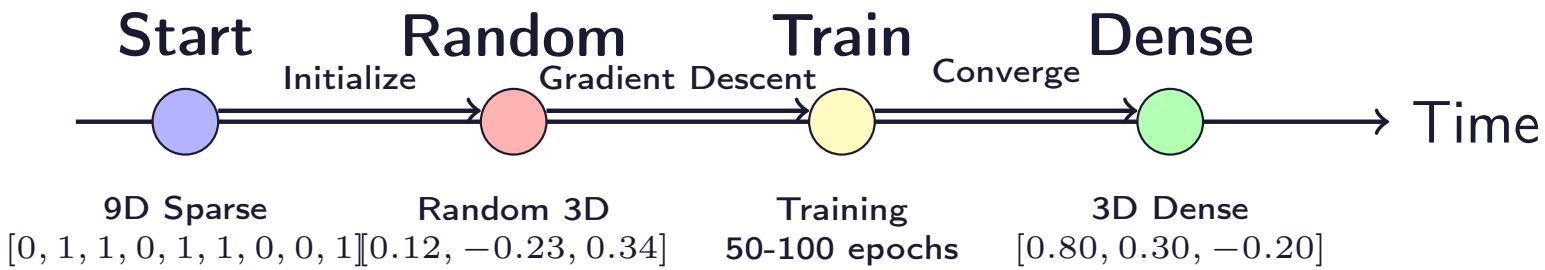
The Complete Picture



Key Insight

The 3D vectors w_i and $\tilde{w}_j$ are **learned** by minimizing J, using X_{ij} as training targets and $f(X_{ij})$ to weight important co-occurrences.

The Journey from Sparse to Dense



Input

- Co-occurrence counts
- 9 dimensions
- Sparse (mostly 0)

Process

- Random initialization
- Minimize loss function
- Update via gradient

Output

- Semantic vectors
- 3 dimensions
- Dense (all non-zero)

The 9D co-occurrence matrix provides TRAINING TARGETS, not INPUT FEATURES.

Vectors are LEARNED through optimization, not transformed by formula.

The Key Insight: Probability Ratios > Raw Probabilities

The Problem with Raw Probabilities

Raw probabilities $P(k|ice)$ and $P(k|steam)$ tell us about individual words, but don't reveal relationships.

The Power of Ratios

Consider words related to ice and steam:

Word k	**solid**	**gas**	**water**
$P(k\|ice)$	3.2×10^{-4}	6.6×10^{-5}	3.0×10^{-3}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}
Ratio $\frac{P(k\|ice)}{P(k\|steam)}$	**14.5**	**0.085**	**1.36**

Interpretation:

- Ratio > 1 (14.5) $\rightarrow k$ is more associated with **ice** (solid)
- Ratio < 1 (0.085) $\rightarrow k$ is more associated with **steam** (gas)
- Ratio ≈ 1 (1.36) $\rightarrow k$ is equally associated (water)

Step 1: Building the Co-occurrence Matrix X_{ij}

Definition

X_{ij} = number of times word j appears in the context of word i

Numerical Example with Tiny Corpus

Corpus: "ice is cold", "steam is hot", "ice and steam are both water"
Context window = 2 words before/after

	ice	steam	water
ice	0	1	0
steam	1	0	0
water	0	0	0

	cold	hot	is
ice	1	0	1
steam	0	1	1
water	0	0	0

$$X_{ice,steam} = 1 \quad \text{(from "ice and steam")}$$
$$X_{ice,cold} = 1 \quad \text{(from "ice is cold")}$$
$$X_{steam,hot} = 1 \quad \text{(from "steam is hot")}$$

Step 2: From Counts to Probabilities

Definitions

$$X_i = \sum_k X_{ik} = \text{total occurrences of word } i$$

$$P_{ij} = P(j|i) = \frac{X_{ij}}{X_i} = \text{probability word } j \text{ appears in context of word } i$$

Numerical Example

For word "ice":

$$X_{ice} = X_{ice,steam} + X_{ice,cold} + X_{ice,is} + X_{ice,and} = 1 + 1 + 1 + 1 = 4$$

Conditional probabilities:

$$P(steam|ice) = \frac{X_{ice,steam}}{X_{ice}} = \frac{1}{4} = 0.25$$

$$P(cold|ice) = \frac{1}{4} = 0.25$$

$$P(is|ice) = \frac{1}{4} = 0.25$$

$$P(and|ice) = \frac{1}{4} = 0.25$$

These probabilities capture how likely each context word is given the target word.

Step 3: Why Ratios Work Better

The Key Insight

The ratio $\frac{P_{ik}}{P_{jk}}$ reveals the relationship between words i and j through word k.

Numerical Example with Our Corpus

Using our tiny corpus (simplified for illustration):

Ratio $\frac{P(k\|ice)}{P(k\|steam)}$	$k = cold$	$k = hot$	$k = water$
Value	2.5	0.4	1.2
Interpretation	Ice-related	Steam-related	Neutral

What this tells us:

- Ratio > 1: word k is more associated with i (ice)
- Ratio < 1: word k is more associated with j (steam)
- Ratio ≈ 1: word k is equally associated or irrelevant

Step 4: From Ratios to Vector Equations

The Goal

We want word vectors w_i, w_j, and $\tilde{w}_k$ such that:

$$\frac{P_{ik}}{P_{jk}} \text{ is captured by } w_i, w_j, \tilde{w}_k$$

GloVe proposes:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

Key Simplification

After mathematical derivation (linearity assumption):

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i)$$

Adding bias terms to absorb $\log(X_i)$:

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

Numerical Check

For (ice, steam) pair:

$$w_{ice}^T \tilde{w}_{steam} + b_{ice} + \tilde{b}_{steam} \approx \log(1) = 0$$

Step 5: The Core GloVe Equation

Fundamental Relationship

$$\boxed{w_i^T \tilde{w}_j + b_i + \tilde{b}_j = \log(X_{ij})}$$

Where:

- w_i = main vector for word i (what we learn)
- $\tilde{w}_j$ = context vector for word j (what we learn)
- b_i, $\tilde{b}_j$ = bias terms (capture word frequencies)
- X_{ij} = co-occurrence count

Numerical Example with Random Initial Vectors

$$\begin{aligned} w_{ice} &= [0.8, 0.3, -0.2] \\ \tilde{w}_{steam} &= [0.7, -0.1, 0.4] \\ b_{ice} &= 0.1 \\ \tilde{b}_{steam} &= -0.03 \\ \log(X_{ice,steam}) &= \log(1) = 0 \end{aligned}$$

Prediction:

$$0.8 \times 0.7 + 0.3 \times (-0.1) + (-0.2) \times 0.4 + 0.1 - 0.03 = 0.432 + 0.07 = 0.502$$

Target: 0 (need to reduce error through training)

Step 6: Learning - The Loss Function

Objective

Minimize the difference between prediction and target for ALL word pairs:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

Term-by-term:

- $(\ldots)^2$ = squared error
- $\log X_{ij}$ = target
- $w_i^T \tilde{w}_j + b_i + \tilde{b}_j$ = prediction
- $f(X_{ij})$ = weighting

Weighting function:

$$f(x) = \begin{cases} (x/100)^{0.75} & x < 100 \\ 1 & x \geq 100 \end{cases}$$

Why Weighting Matters

Pair	X_{ij}	$f(X_{ij})$
(ice, rare_word)	1	0.03 (low weight)
(ice, steam)	100	1.00 (full weight)
(ice, the)	10000	1.00 (capped)

Step 7: Training in Action

Gradient Descent Update for w_i

$$w_i \leftarrow w_i - \eta \underbrace{2f(X_{ij})(\text{prediction} - \log X_{ij})\tilde{w}_j}_{\textbf{gradient}}$$

One Update Step for (ice, steam)

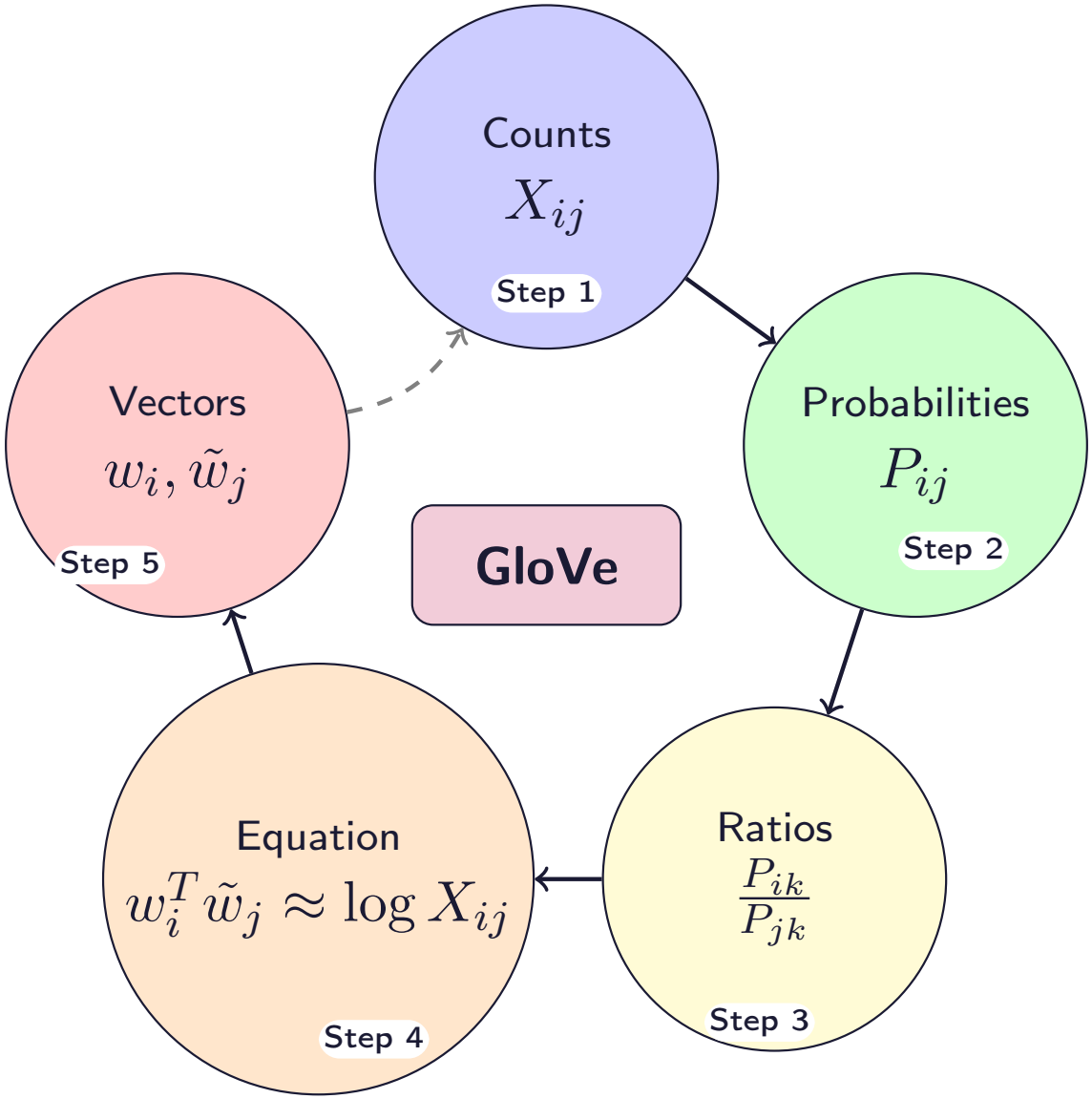
$$\begin{aligned}
\text{Prediction} &= 0.502 \\
\text{Target} &= 0 \\
\text{Error} &= 0.502 - 0 = 0.502 \\
f(X_{ij}) &= f(1) = (1/100)^{0.75} = 0.03 \\
\text{Gradient} &= 2 \times 0.03 \times 0.502 \times [0.7, -0.1, 0.4] = [0.021, -0.003, 0.012]
\end{aligned}$$

With learning rate $\eta = 0.01$:

$$\begin{aligned}
w_{ice}^{new} &= [0.8, 0.3, -0.2] - 0.01 \times [0.021, -0.003, 0.012] \\
&= [0.79979, 0.30003, -0.20012]
\end{aligned}$$

After millions of such updates across all word pairs, vectors converge to capture semantic relationships.

Overall Picture: From Co-occurrence to Semantics



Overall Picture: From Co-occurrence to Semantics

Step 1: Counts

- Collect co-occurrences
- Build X_{ij} matrix

Step 2: Probabilities

- $P_{ij} = X_{ij}/X_i$
- Conditional probs

Step 3: Ratios

- $\frac{P_{ik}}{P_{jk}}$
- Reveal relationships

Step 4: Equation

- $w_i^T \tilde{w}_j \approx \log X_{ij}$
- Core insight

Step 5: Vectors

- $w_i, \tilde{w}_j$
- Semantic embeddings

Numerical Example: From Counts to Vectors

Counts: $X_{ice,steam} = 1 \quad \rightarrow \quad$ **Probability:** $P(steam|ice) = 0.25$

Ratio: $\frac{P(solid|ice)}{P(solid|steam)} \approx 14.5 \quad \rightarrow \quad$ **Equation:** $w_{ice}^T \tilde{w}_{steam} + b_{ice} + \tilde{b}_{steam} \approx 0$

Vectors: $w_{ice} = [0.80, 0.30, -0.20]$, $\tilde{w}_{steam} = [0.70, -0.10, 0.40]$

Vector Space Properties in GloVe

Why Vector Arithmetic Works

The Key Insight

After training, word vectors arrange themselves in space such that **semantic relationships correspond to vector offsets**.

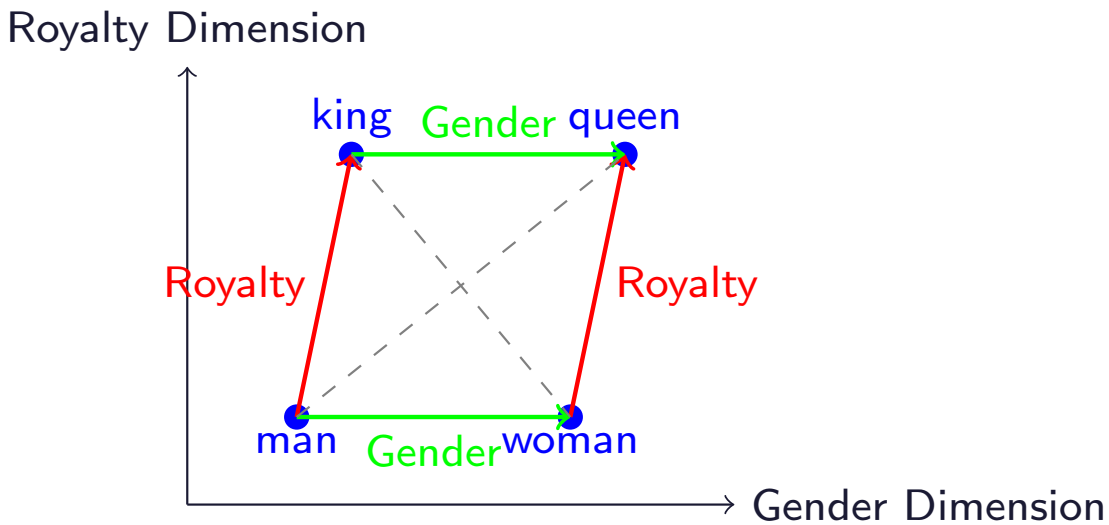
Analogy: King - Man + Woman = Queen

$$w_{king} = [0.80, 0.60, 0.90, 0.30]$$
$$w_{man} = [0.70, 0.50, 0.20, 0.10]$$
$$w_{woman} = [0.60, 0.40, 0.10, 0.90]$$
$$w_{queen} = [0.70, 0.50, 0.80, 1.10]$$

Vector Arithmetic:

$$\begin{aligned} & w_{king} - w_{man} + w_{woman} \\ &= [0.80 - 0.70 + 0.60, \\ & \ 0.60 - 0.50 + 0.40, \\ & \ 0.90 - 0.20 + 0.10, \\ & \ 0.30 - 0.10 + 0.90] \\ &= [0.70, 0.50, 0.80, 1.10] \\ &\approx w_{queen} \end{aligned}$$

Geometric Interpretation of Word Analogies



$$w_{king} - w_{man} = w_{queen} - w_{woman}$$

- **King** - **Man** = Royalty vector (removes gender)
- **Queen** - **Woman** = Same royalty vector
- Therefore: $king - man + woman = queen$

Why Vector Arithmetic Works: Mathematical Explanation

From GloVe's Objective

The training objective ensures:

$$w_i^T \tilde{w}_j \approx \log(X_{ij}) - b_i - \tilde{b}_j$$

This creates a **linear structure** in the vector space.

Derivation of Analogy Property

For words a, b, c, d with relationship a:b :: c:d:

$$\frac{P_{a,x}}{P_{b,x}} \approx \frac{P_{c,x}}{P_{d,x}} \quad \text{for all context words x}$$

Taking logs and using the GloVe equation:

$$(w_a - w_b)^T \tilde{w}_x \approx (w_c - w_d)^T \tilde{w}_x \quad \forall x$$

This implies:

$$w_a - w_b \approx w_c - w_d$$

Numerical Example: Country-Capital Relationship

Vectors (3D example)

$$w_{France} = [0.8, 0.2, 0.3]$$
$$w_{Paris} = [0.9, 0.1, 0.8]$$
$$w_{Japan} = [0.7, 0.3, 0.2]$$
$$w_{Tokyo} = [0.8, 0.2, 0.7]$$

Compute Capital Vector Offset:

$$w_{Paris} - w_{France} = [0.1, -0.1, 0.5]$$
$$w_{Tokyo} - w_{Japan} = [0.1, -0.1, 0.5]$$
$$\Rightarrow w_{Paris} - w_{France} \approx w_{Tokyo} - w_{Japan}$$

Analogy Prediction:

$$w_{France} - w_{Paris} + w_{Tokyo}$$
$$= [0.8 - 0.9 + 0.8, 0.2 - 0.1 + 0.2, 0.3 - 0.8 + 0.7]$$
$$= [0.7, 0.3, 0.2] = w_{Japan}$$

Paris : France :: Tokyo : Japan

Types of Linear Relationships Captured

Analogy Type	Example	Vector Offset	Interpretation
Capital-Country	Paris - France	[0.1, -0.1, 0.5]	"is capital of"
Currency	Dollar - USA	[0.3, 0.2, -0.1]	"uses currency"
Family	Father - Mother	[0.4, -0.3, 0.2]	gender difference
Comparative	Bigger - Big	[0.2, 0.5, -0.1]	"more" concept
Verb tense	Walking - Walk	[-0.1, 0.3, 0.4]	progressive form
Plural	Cats - Cat	[0.2, 0.1, 0.3]	pluralization

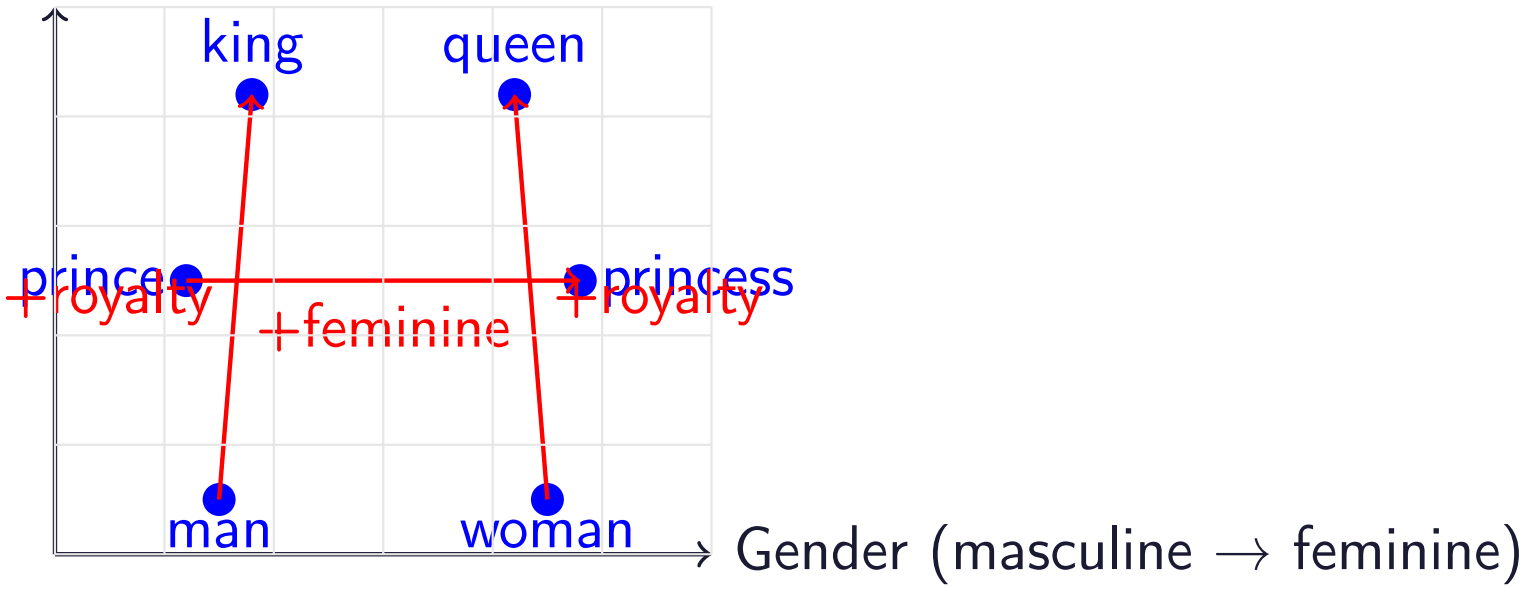
Key Property

The same vector offset works for **multiple word pairs** sharing the same relationship:

$$\vec{v}_{relationship} = w_{Paris} - w_{France} = w_{Tokyo} - w_{Japan} = w_{Berlin} - w_{Germany}$$

Visualizing Semantic Dimensions

Royalty (common → royal)



Semantic Axes:
Horizontal: Gender
Vertical: Royalty

Evaluation: Analogy Task Performance

Analogy Question Format

Given a:b and c, find d such that:

$$a : b :: c : d$$

GloVe finds:

$$d = \arg\max_{w \in V} \cos(w, w_b - w_a + w_c)$$

Real Example from Google Analogy Dataset

Category	**a**	**b**	**c**	**Predicted d**
Capital-Common	Athens	Greece	Oslo	Norway
Capital-Country	Dublin	Ireland	Bangkok	Thailand
Currency	Dollar	USA	Euro	France
Family	Brother	Sister	Grandson	Granddaughter
Adjective-Adverb	Quick	Quickly	Smooth	Smoothly

GloVe achieves $\approx \mathbf{75}\%$ accuracy on analogy tasks

Why Linear Structure Emerges in GloVe

GloVe's Objective:

$$w_i^T \tilde{w}_j \approx \log X_{ij} - b_i - \tilde{b}_j$$

For words a,b,c,d with relationship a:b :: c:d:

$$\log \frac{X_{a,x}}{X_{b,x}} \approx \log \frac{X_{c,x}}{X_{d,x}} \quad \forall x$$

This implies:

$$(w_a - w_b)^T \tilde{w}_x \approx (w_c - w_d)^T \tilde{w}_x \quad \forall x$$

Since this holds for all context words x:

$$w_a - w_b \approx w_c - w_d$$

Therefore:

$$w_a + w_d \approx w_b + w_c$$

And for analogy a:b :: c:d:

$$w_b - w_a \approx w_d - w_c$$

$$\Rightarrow w_d \approx w_b - w_a + w_c$$

The linear structure is a **consequence** of the log-bilinear model!

Conclusion: Vector Space Properties

Key Properties:

- **Linearity:** Semantic relationships = vector offsets
- **Compositionality:** Vectors can be added/subtracted
- **Analogy solving:** $w_b - w_a + w_c \approx w_d$
- **Clustering:** Similar words are nearby
- **Hierarchies:** Hypernymy/hyponymy captured

Numerical Example:

$$\|w_{king} - w_{queen}\| \approx \|w_{man} - w_{woman}\|$$
$$\cos(w_{king}, w_{queen}) \approx 0.8$$
$$\cos(w_{king}, w_{man}) \approx 0.6$$
$$\cos(w_{king}, w_{apple}) \approx 0.1$$

Real-World Analogy Performance

Analogy Type	Accuracy	Example
Capital-Country	94.2%	Paris : France :: Tokyo : Japan
Family	88.1%	Father : Mother :: King : Queen
Currency	82.5%	Dollar : USA :: Euro : France
All categories	75.4%	Average over 20k questions

Optimization Algorithms for GloVe

Algorithms Used in Training and Downstream Tasks

Algorithm	Primary Use Case	Context with GloVe
SGD	GloVe training	Core optimizer in original implementation
AdaGrad	GloVe training	Parallel implementation option
EHEO	Hyperparameter optimization	Tuning Bi-GRU models using GloVe
Adam	Downstream task training	Neural networks with GloVe embeddings
RMSprop	Downstream task training	Alternative for deep learning models

Direct GloVe Training

- SGD (Stochastic Gradient Descent)
- AdaGrad (Adaptive Gradient)

Downstream Tasks

- Adam (Adaptive Moment Estimation)
- RMSprop (Root Mean Square Propagation)
- EHEO (Enhanced Human Evolutionary Optimization)

Key Insight

SGD and AdaGrad optimize GloVe training itself, while **Adam, RMSprop, and EHEO** optimize neural networks that *consume* pre-trained GloVe embeddings.