

Skip-gram with Negative Sampling

The algorithm behind the theorem

James Pustejovsky

Brandeis University
Department of Computer Science

CS 135: Computational Semantics

Fall 2026

Where This Sits

Session 8 derived what SGNS converges to:

$$\mathbf{c}_c \cdot \mathbf{w}_w = \text{PMI}(w, c) - \log k$$

by setting a derivative to zero and solving.

That derivation says where the dot products *land*. It says nothing about how they get there, and the answer turns out to matter for three questions this course has open.

Why gradient descent at all?	Session 8d answered this in principle
Where do W and C come from?	and what is the relation to U and V
Does the theorem actually hold?	we will check it numerically

Today is the mechanics: two embedding tables, one loss, four gradients, and a verification.

The Starting Point: Words as Symbols

Before any of this, a word was an index. cat is entry 3472, dog is entry 8901, and the two numbers are unrelated.

As vectors, that means one-hot: for $\mathbf{e}_i, \mathbf{e}_j \in \mathbb{R}^{|V|}$ with $i \neq j$,

$$\mathbf{e}_i \cdot \mathbf{e}_j = 0 \quad \text{for every pair of distinct words.}$$

You have seen this object before. In Session 6B the one-hot vector was how we picked an individual out of the domain, and its orthogonality was a *feature* – it is what made $M\mathbf{e}_j$ select a column cleanly.

Here the same orthogonality is the problem. A representation in which every word is equidistant from every other encodes no similarity at all.

The goal is a map $\phi : V \rightarrow \mathbb{R}^d$ with $d \ll |V|$ in which distances mean something.

Two Tables, Not One

The parameterization

For vocabulary V and dimension d , SGNS keeps **two** sets of vectors:

$$W \in \mathbb{R}^{|V| \times d} \quad (\text{word as } \textit{target}), \quad C \in \mathbb{R}^{|V| \times d} \quad (\text{word as } \textit{context})$$

Parameters: $\theta = \{W, C\}$, a total of $2|V|d$ numbers.

Every word gets two vectors, and which one is used depends on the role it is playing in the pair being scored:

$$\text{score}(w, c) = \mathbf{c}_c \cdot \mathbf{w}_w$$

the *target* vector of w against the *context* vector of c .

The usual justification is practical: a word rarely appears in its own context, so tying the tables would force $\mathbf{w}_w \cdot \mathbf{w}_w$ to be small, which is awkward. But something more interesting is going on.

An Asymmetry the Model Has and Discards

Part I gave likes the type $\langle e, \langle e, t \rangle \rangle$ and Alice the type e . The asymmetry between a functor and its argument was **in the type**, and it governed what could combine with what.

SGNS also has an asymmetry. A word occupies one of two positions, and gets a different vector in each.

	Part I	SGNS
The distinction	functor vs. argument	target vs. context
Where it lives	in the type	in which table you index
Enforced by	the type system	nothing
Survives training	yes	no – C is thrown away

Both vectors are elements of the same $\mathbb{R}^d$. Nothing stops you from putting a context vector where a target vector belongs, and the arithmetic will not complain. The asymmetry is a bookkeeping convention, not a type.

W and C Are U and V

Once you know the theorem, the two tables stop looking like an implementation detail.

$$WC^\top \approx M^{\text{PMI}} - \log k \quad \text{against} \quad U_k \Sigma_k V_k^\top = M_k$$

	SVD	SGNS
Word side	U_k , rows indexed by words	W
Context side	V_k , rows indexed by contexts	C
Scale	Σ_k , explicit and separate	absorbed into the two factors
Orthogonality	guaranteed	none

Both methods produce a two-sided factorization of a PMI-like matrix. The SVD hands you the scale in a third factor you can place where you like; SGNS has already distributed it, and you cannot recover the split.

What To Do With C

After training, the standard move is to keep W and discard C . Also common: use $W + C$.

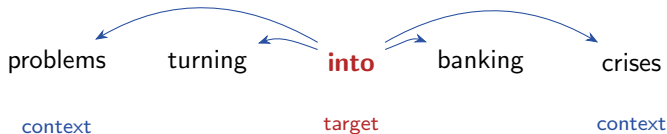
Choice	SVD analogue	
keep W only	$W = U_k \Sigma_k^\alpha$, discard V	the usual default
use $W + C$	add the two sides	sometimes slightly better
concatenate $[W; C]$	keep both, $2d$ dimensions	rarely used

This is the α question from Session 8 in a different costume. There we asked where the scale should go once V is discarded; here we ask whether to discard C at all. In both cases the reconstruction $WC^\top$ is unchanged and only the *word geometry* moves.

And in both cases the answer is settled by downstream evaluation rather than by theory, which is the recurring situation in this part of the course.

Skip-gram, in One Line

Slide a window across the corpus. At each position the center word is the **target** and the surrounding words within distance L are its **contexts**.



$$P(c \mid w; \theta) = \frac{\exp(\mathbf{c}_c \cdot \mathbf{w}_w)}{\sum_{c' \in V} \exp(\mathbf{c}_{c'} \cdot \mathbf{w}_w)}$$

Note what carries the whole model: a single inner product, exponentiated. The probability that c occurs near w is a monotone function of $\mathbf{c}_c \cdot \mathbf{w}_w$ and of nothing else.

The Denominator, and the Way Around It

That sum runs over the whole vocabulary, so every gradient step costs $O(|V|d)$ just to normalize. With $|V| \approx 10^6$ that is fatal.

Negative sampling replaces the $|V|$ -way question with a binary one:

Did c actually occur in the context of w , or did I make it up?

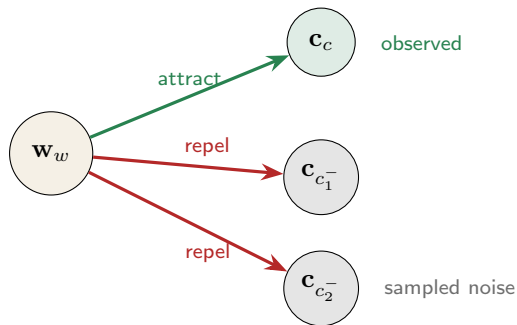
$$P(D = 1 \mid w, c) = \sigma(\mathbf{c}_c \cdot \mathbf{w}_w), \quad P(D = 0 \mid w, c) = \sigma(-\mathbf{c}_c \cdot \mathbf{w}_w)$$

The SGNS objective

$$\ell(w, c) = \underbrace{\log \sigma(\mathbf{c}_c \cdot \mathbf{w}_w)}_{\text{observed pair}} + \sum_{i=1}^k \underbrace{\log \sigma(-\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w)}_{k \text{ invented pairs}}$$

with each c_i^- drawn from a noise distribution P_n .

Attract and Repel



The first term is maximized by making $\mathbf{c}_c \cdot \mathbf{w}_w$ large, the second by making each $\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w$ small.

The word's position is determined by a tug of war between one thing it was seen with and k things it was not. There is no other force in the model.

The Noise Distribution

Negatives are drawn from a *smoothed* unigram distribution:

$$P_n(c) = \frac{\#(c)^\alpha}{\sum_{c'} \#(c')^\alpha}, \quad \alpha = 0.75$$

Word	$P(c)$	$P(c)^{3/4}$	effect
the	0.060	0.0306	sampled less
cat	10^{-4}	5.6×10^{-4}	sampled more
constitutionality	10^{-7}	1.8×10^{-6}	sampled much more

You have met $\alpha = 0.75$ before. It is the same exponent as context distribution smoothing in the PPMI pipeline, doing the same job in a different place: flattening the frequency distribution so that rare words are not swamped.

It also perturbs the theorem slightly. The clean identity assumes $\alpha = 1$; at $\alpha = 0.75$ the target is a smoothed PMI rather than PMI itself.

Why There Are Gradients Here At All

Session 8d established the reason, and it is worth restating before doing the calculus.

Eckart–Young	gives the optimal low-rank factorization for the <i>un-weighted</i> Frobenius loss, in closed form, via the spectrum
SGNS's loss	is logistic, and weights each cell by $\#(w, c)$
Consequence	no spectral solution exists; the problem is NP-hard in general

So the gradient is not an architectural choice. It is what remains available once you decide that a cell observed three thousand times deserves more attention than a cell observed twice.

Everything in the next five slides is the price of that decision, paid in arithmetic.

Two Identities

Everything follows from the derivative of the log-sigmoid.

$$\frac{d}{dx} \log \sigma(x) = \frac{\sigma'(x)}{\sigma(x)} = \frac{\sigma(x)(1 - \sigma(x))}{\sigma(x)} = 1 - \sigma(x)$$

$$\frac{d}{dx} \log \sigma(-x) = -(1 - \sigma(-x)) = -\sigma(x)$$

Read them as *error signals*. The first is how far the model's estimate of a true pair falls short of 1; the second is how much probability it wrongly assigns to a false pair.

Both are bounded in $(0, 1)$, which is why SGNS trains stably without a learning-rate schedule doing much work.

The Gradient for the Target Vector

From the positive term:

$$\frac{\partial}{\partial \mathbf{w}_w} \log \sigma(\mathbf{c}_c \cdot \mathbf{w}_w) = [1 - \sigma(\mathbf{c}_c \cdot \mathbf{w}_w)] \mathbf{c}_c$$

From the i th negative term:

$$\frac{\partial}{\partial \mathbf{w}_w} \log \sigma(-\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w) = -\sigma(\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w) \mathbf{c}_{c_i^-}$$

$$\frac{\partial \ell}{\partial \mathbf{w}_w} = [1 - \sigma(\mathbf{c}_c \cdot \mathbf{w}_w)] \mathbf{c}_c - \sum_{i=1}^k \sigma(\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w) \mathbf{c}_{c_i^-}$$

The target moves toward the observed context and away from each sampled one, each step scaled by how surprised the model currently is.

The Gradients for the Context Vectors

$$\frac{\partial \ell}{\partial \mathbf{c}_c} = [1 - \sigma(\mathbf{c}_c \cdot \mathbf{w}_w)] \mathbf{w}_w \qquad \frac{\partial \ell}{\partial \mathbf{c}_{c_i^-}} = -\sigma(\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w) \mathbf{w}_w$$

Symmetric to the last slide, with the roles exchanged: the context vectors move along $\pm \mathbf{w}_w$.

How much gets touched

One step updates $\mathbf{w}_w$, $\mathbf{c}_c$, and the k negatives: $2 + k$ vectors out of $2|V|$. With $|V| = 10^6$ and $k = 5$, that is seven rows out of two million.

Cost per step is $O(dk)$, independent of $|V|$. This is the entire practical argument for SGNS over the SVD, and it is an argument about sparsity of *updates*, not about representations.

Every Update Is a Multiple of the Other Vector

Look at the four gradients together. Each one has the same shape:

$$\Delta \mathbf{w}_w \in \text{span}\{\mathbf{c}_c, \mathbf{c}_{c_1}^-, \dots\}, \quad \Delta \mathbf{c} = \text{scalar} \times \mathbf{w}_w$$

No vector is ever moved in a direction that is not already a vector in the other table, and the step size is always a function of an inner product.

The entire learning dynamics take place inside the inner product. The model has no access to any other operation on its vectors – no outer product, no component-wise interaction, nothing that could raise the order of the representation.

Session 8d argued that distributional semantics made a silent choice of the inner product. Here is that choice at the level of the update rule, where it is not a matter of interpretation.

The Update Rules

For one pair (w, c) with negatives $c_1^-, \dots, c_k^-$ and learning rate η :

1. **Scores.** $s^+ = \mathbf{c}_c \cdot \mathbf{w}_w, \quad s_i^- = \mathbf{c}_{c_i^-} \cdot \mathbf{w}_w$

2. **Target.**

$$\mathbf{w}_w \leftarrow \mathbf{w}_w + \eta \left[(1 - \sigma(s^+)) \mathbf{c}_c - \sum_{i=1}^k \sigma(s_i^-) \mathbf{c}_{c_i^-} \right]$$

3. **Observed context.** $\mathbf{c}_c \leftarrow \mathbf{c}_c + \eta (1 - \sigma(s^+)) \mathbf{w}_w$

4. **Each sampled context.** $\mathbf{c}_{c_i^-} \leftarrow \mathbf{c}_{c_i^-} - \eta \sigma(s_i^-) \mathbf{w}_w$

Typical settings: $d = 100$ – 300 , $L = 5$ – 10 , $k = 5$ – 20 on small corpora and 2 – 5 on large ones, η starting at 0.025 and decaying linearly.

Worked Example: the Setup

$d = 3$, target $w = \text{coffee}$, observed context $c = \text{cup}$, one negative $c^- = \text{elephant}$, $\eta = 0.1$.

$$\mathbf{w}_{\text{coffee}} = \begin{pmatrix} 0.5 \\ -0.2 \\ 0.3 \end{pmatrix}, \quad \mathbf{c}_{\text{cup}} = \begin{pmatrix} 0.4 \\ 0.1 \\ 0.6 \end{pmatrix}, \quad \mathbf{c}_{\text{elephant}} = \begin{pmatrix} -0.1 \\ 0.8 \\ -0.3 \end{pmatrix}$$

Scores.

$$s^+ = (0.4)(0.5) + (0.1)(-0.2) + (0.6)(0.3) = 0.360$$

$$s^- = (-0.1)(0.5) + (0.8)(-0.2) + (-0.3)(0.3) = -0.300$$

Error signals. $1 - \sigma(0.360) = 1 - 0.5890 = 0.4110$, $\sigma(-0.300) = 0.4256$

Worked Example: the Step

$$\Delta \mathbf{w} = 0.1 [\mathbf{0.4110} \mathbf{c}_{\text{cup}} - \mathbf{0.4256} \mathbf{c}_{\text{elephant}}] = \begin{pmatrix} 0.0207 \\ -0.0299 \\ 0.0374 \end{pmatrix}$$

	before	after
$\mathbf{w}_{\text{coffee}}$	(0.500, -0.200, 0.300)	(0.521, -0.230, 0.337)
$\mathbf{c}_{\text{cup}}$	(0.400, 0.100, 0.600)	(0.421, 0.092, 0.612)
$\mathbf{c}_{\text{elephant}}$	(-0.100, 0.800, -0.300)	(-0.121, 0.809, -0.313)

	before	after	change
$\mathbf{w}_{\text{coffee}} \cdot \mathbf{c}_{\text{cup}}$	0.3600	0.4045	+0.0445
$\mathbf{w}_{\text{coffee}} \cdot \mathbf{c}_{\text{elephant}}$	-0.3000	-0.3546	-0.0546
$\cos(\text{coffee, cup})$	0.8022	0.8167	+0.0145
$\cos(\text{coffee, elephant})$	-0.5657	-0.6122	-0.0465

The Theorem, and Whether to Believe It

Session 8 derived

$$\mathbf{c}_c \cdot \mathbf{w}_w = \text{PMI}(w, c) - \log k$$

from a first-order condition on a single pair, holding everything else fixed.

The derivation assumes	but in practice
each dot product is free to take any value	$d \ll V $ constrains them jointly
the optimum is reached	SGD stops early
P_n is the unigram, $\alpha = 1$	$\alpha = 0.75$

So the identity is a statement about an idealization. It is worth asking how closely the idealization is met – and that is something we can just measure.

The Experiment

Corpus	three topics, Zipfian word frequencies, 15% cross-topic leakage
Vocabulary	$ V = 15$
Window	$L = 2$, giving 180,132 observed (w, c) tokens
Model	$d = 12$, $k = 5$, noise $\alpha = 1$
Trained by	gradient ascent on the expected objective

The leakage matters. Without it every within-topic pair has nearly the same PMI, and there is no range over which to test anything. With it, $\text{PMI} - \log k$ spans $[-3.12, -0.59]$.

Then compare, for all 225 observed pairs, the learned $\mathbf{c}_c \cdot \mathbf{w}_w$ against the $\text{PMI}(w, c) - \log k$ computed directly from the counts.

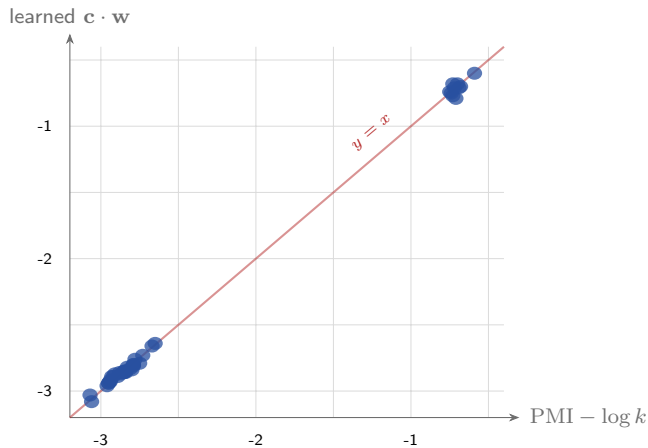
The Result

correlation	0.9996	
mean absolute error	0.0203	against a range of 2.52
best-fit slope	0.9990	theory says 1
best-fit intercept	-0.0010	theory says 0

pair	$\#(w, c)$	PMI	PMI - $\log k$	learned
coffee / cup	3093	0.863	-0.746	-0.691
coffee / brew	2384	0.898	-0.711	-0.657
river / water	4371	0.903	-0.707	-0.794
laptop / screen	4382	0.886	-0.724	-0.773
coffee / laptop	990	-1.232	-2.841	-2.837
river / battery	345	-1.274	-2.884	-2.911
coffee / river	930	-1.280	-2.890	-2.837

A quantity nobody computed, and nobody put in the loss, recovered to two decimal places.

The Same Thing, as a Picture



The two clusters are within-topic pairs (upper right) and cross-topic pairs (lower left). Both sit on the diagonal.

What the Fit Is Telling You

Slope 0.999	the relation is one-to-one, not merely monotone. SGNS is not learning “something correlated with PMI”; it is learning PMI
Intercept -0.001	the $-\log k$ offset is exactly right, which confirms that k enters as a constant shift and nothing else
Error 0.02	what is left is the finite d and finite optimization, and it is small

Raise k and every point slides down the diagonal by $\log k$ without changing shape. Lower d and the points scatter around the line, because the factorization can no longer fit every cell independently – which is the low-rank approximation reappearing.

The word “implicit” in “implicit matrix factorization” is doing no hedging. The matrix is there, to two decimal places.

Cosine, Not Dot Product

Training drives $\mathbf{c} \cdot \mathbf{w}$ toward PMI. Evaluation almost never uses the dot product.

$$\cos(\mathbf{w}_a, \mathbf{w}_b) = \frac{\mathbf{w}_a \cdot \mathbf{w}_b}{\|\mathbf{w}_a\| \|\mathbf{w}_b\|}$$

Dot product	grows with vector norm, and norm grows with frequency
Cosine	measures direction only

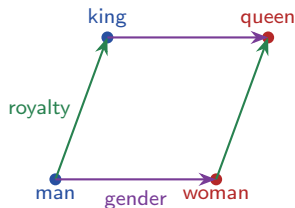
Note the small inconsistency this introduces. The objective optimizes a dot product; the evaluation reads an angle. Nothing guarantees that a model good at the first is best at the second, and the norm it discards carries real information – roughly, how confidently the word was placed.

This is a gap between what is optimized and what is measured, and it is rarely remarked on.

Analogies

The result that made word2vec famous:

$$\mathbf{w}_{\text{king}} - \mathbf{w}_{\text{man}} + \mathbf{w}_{\text{woman}} \approx \mathbf{w}_{\text{queen}}$$



The claim is that a semantic relation is a *constant displacement* – the same arrow, wherever in the space you apply it.

Why Analogies Follow From the Theorem

The analogy $a : b :: c : d$ holds when the PMI differences match across every context:

$$\text{PMI}(a, c') - \text{PMI}(b, c') \approx \text{PMI}(c, c') - \text{PMI}(d, c') \quad \text{for all } c'$$

Substituting $\mathbf{w}_w \cdot \mathbf{c}_{c'} \approx \text{PMI}(w, c') - \log k$, the $\log k$ cancels in each difference:

$$(\mathbf{w}_a - \mathbf{w}_b) \cdot \mathbf{c}_{c'} \approx (\mathbf{w}_c - \mathbf{w}_d) \cdot \mathbf{c}_{c'} \quad \text{for all } c'$$

If that holds against enough context vectors to span the space, then

$$\mathbf{w}_a - \mathbf{w}_b \approx \mathbf{w}_c - \mathbf{w}_d$$

So the parallelogram is not an emergent mystery. It is what a PMI-ratio regularity in the corpus looks like after you take logarithms, and logarithms turn ratios into differences.

What the Parallelogram Commits You To

Read the claim structurally. If a relation R is a fixed vector $\mathbf{r}$ with $\mathbf{w}_{R(x)} = \mathbf{w}_x + \mathbf{r}$, then:

Property of vector addition	Property it forces on the relation
$\mathbf{r} + \mathbf{s} = \mathbf{s} + \mathbf{r}$	relations commute
$\mathbf{r} + (-\mathbf{r}) = \mathbf{0}$	every relation is invertible
$\mathbf{r}$ does not depend on $\mathbf{x}$	the relation applies uniformly

Meanings under relations form an abelian group. That is a strong claim, and it is why the successes are what they are: plural, comparative, past tense, capital-of, currency-of are all near-invertible and near-uniform.

And it is why *cause*, *part-of*, *entails* do not work this way. Those relations are not invertible, and some are not even symmetric in the relevant sense.

The Same Commutativity, Twice

Analogies	vector addition is commutative, and that is exactly what makes the parallelogram work
Composition	vector addition is commutative, and that is exactly what makes <i>dog bites man</i> and <i>man bites dog</i> identical

One property, celebrated in one place and fatal in the other. The analogy result is not evidence that the space is compositional. It is evidence that the space is **additive**, and additivity is the specific thing that composition cannot be.

Session 8d put four operations on a table and asked which keeps structure. The analogy result is the best case for the inner-product world, and it is a case about relations between *single words*, never about combining two of them into a third.

The Same Geometry, Read as Bias

Bolukbasi et al. (2016)

$$\mathbf{w}_{\text{man}} - \mathbf{w}_{\text{woman}} \approx \mathbf{w}_{\text{programmer}} - \mathbf{w}_{\text{homemaker}}$$

This is the analogy result, unchanged. The same displacement structure that gives king – queen gives this, and for the same reason: the PMI ratios in the corpus have that shape.

The model is not malfunctioning. It is reporting an association that is present in the text, using the only mechanism it has.

Debiasing by projecting out a “gender direction” removes the measurement, not the structure. Gonen & Goldberg (2019) showed the affected words remain clustered together afterward – the information survives the projection that was supposed to delete it.

The Window Chooses What “Similar” Means

Small window ($L \leq 3$)	Large window ($L \geq 8$)
syntactic similarity	topical association
dog $\sim$ cat, dogs	dog $\sim$ pet, veterinarian
tight substitutability	broad relatedness

Word2vec also samples the window size uniformly from $\{1, \dots, L\}$ at each position, which weights nearer words more heavily without any explicit weighting term.

This is the Session 7 knob, unchanged. The definition of “context” was always the place where a linguistic decision entered the pipeline, and it still is – it is simply now a hyperparameter rather than a stated choice.

Dependency-based contexts sharpen the effect further, and are the closest this tradition comes to the Word Sketch idea.

What SGNS Fixed

Scale	$O(dk)$ per step, independent of $ V $; streaming; no matrix to hold
Availability	one implementation, fast, that anyone could run on their own corpus
Differentiability	the representation became a layer, which is what made the next five years possible

These are substantial and they are engineering. Levy, Goldberg & Dagan showed that with the same hyperparameters, the count-based pipeline matches SGNS on most benchmarks.

What changed in 2013 was not the theory of meaning. It was the cost of acting on one.

What It Did Not Fix

One vector per type	bank still has one vector – Session 9
Antonyms	hot and cold share contexts, so they stay close
Negation	not distributionally marked
Composition	no operation takes two vectors to a third of a new type

The pattern from Session 8's ledger holds. The prediction route improved everything that was a consequence of *how the matrix was built*, and nothing that was a consequence of *distribution being the evidence*.

And the mechanism is now visible in the update rule. Every gradient moves a vector along another vector, scaled by an inner product. Nothing in that loop can produce an object of a different type from the ones it started with.

Where This Leaves Us

	operation	keeps
Distributional, including SGNS	$\mathbf{a} \cdot \mathbf{b}$	similarity
Tensor products	$\mathbf{a}\mathbf{b}^\top$	structure, at a cost in dimension
VSA	folded outer product	structure, fixed dimension
Geometric algebra	$\mathbf{a}\mathbf{b} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}$	both, as grades

SGNS belongs to the first row, and today showed why in the strongest available way: not by interpreting the model, but by reading its gradients. There is no step in the training loop that is not an inner product.

The next question is whether a representation that kept both parts of the product would be any use for semantics. That is what the second half of the course is for.

Exercises

- 1 In the worked example, $\cos(\text{coffee}, \text{elephant})$ moved by -0.047 while $\cos(\text{coffee}, \text{cup})$ moved by $+0.015$. The learning rate was the same for both. Explain the asymmetry from the two error signals.
- 2 The verification used noise exponent $\alpha = 1$. Predict, with reasons, what happens to the slope and intercept of the fit when $\alpha = 0.75$, and say which quantity the learned dot products would then be tracking.
- 3 Suppose you keep $W + C$ rather than W . Show that the reconstruction WC^T is unaffected by this choice, and explain why the word-to-word cosines are not.
- 4 **Harder.** The parallelogram makes relations an abelian group. Name a lexical relation that is genuinely non-invertible, predict what the analogy test would report for it, and say what a representation would need in order to do better.

Reading: Mikolov et al. (2013a, 2013b); Levy & Goldberg (2014); Levy, Goldberg & Dagan (2015); Bolukbasi et al. (2016); Gonen & Goldberg (2019).