Mathematical structures for word embeddings

Siddharth Bhat

IIIT Hyderabad

October 23th, 2021

What is a word embedding?

What is a word embedding?

- Map words to *mathematical objects*.

What is a word embedding?

- Map words to *mathematical objects*.
- Semantic ideas on words $\simeq$ mathematical operations on these objects.

What is a word embedding?

- Map words to *mathematical objects*.
- Semantic ideas on words $\simeq$ mathematical operations on these objects.
- Most common: *vector embeddings* (`word2vec`)

What is a word embedding?

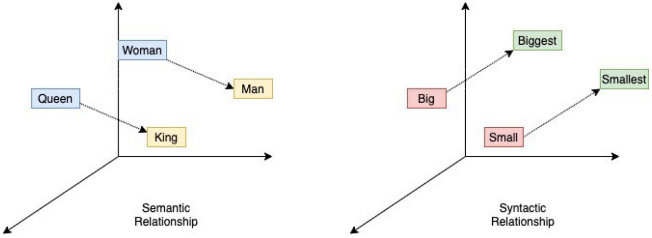
- Map words to *mathematical objects*.
- Semantic ideas on words $\simeq$ mathematical operations on these objects.
- Most common: *vector embeddings* (`word2vec`)

What's `word2vec`?

- Input: A corpus (sequence of words.). Output: mapping from words to *vectors*.

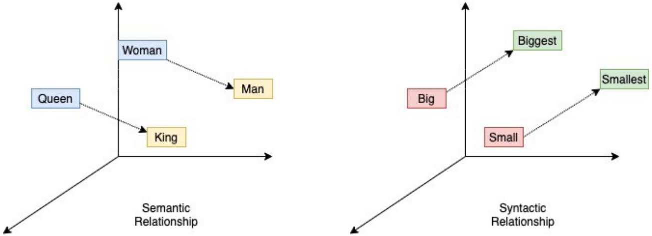
What's `word2vec`?

- Input: A corpus (sequence of words.). Output: mapping from words to *vectors*.



What's `word2vec`?

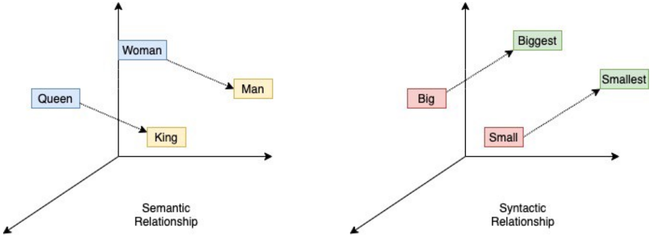
- Input: A corpus (sequence of words.). Output: mapping from words to *vectors*.



- How does one compute such a mapping?

What's word2vec?

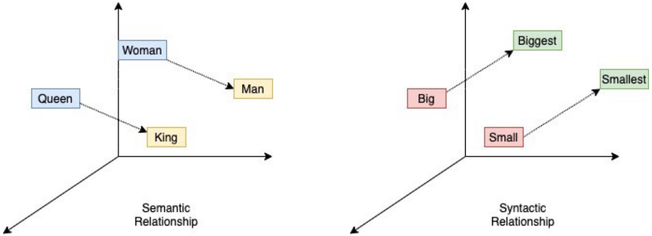
- Input: A corpus (sequence of words.). Output: mapping from words to *vectors*.



- How does one compute such a mapping?
- Distributional Hypothesis: words that occur in the same contexts tend to have similar meanings.

What's word2vec?

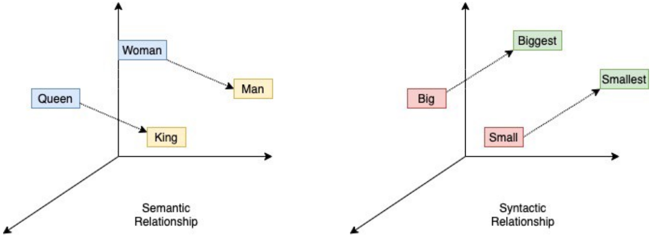
- Input: A corpus (sequence of words.). Output: mapping from words to *vectors*.



- How does one compute such a mapping?
- Distributional Hypothesis: words that occur in the same contexts tend to have similar meanings.
- Start with *random vectors*. Make words that occur close by come closer, words that don't occur close by become perpendicular.

What's word2vec?

- Input: A corpus (sequence of words.). Output: mapping from words to *vectors*.



- How does one compute such a mapping?
- Distributional Hypothesis: words that occur in the same contexts tend to have similar meanings.
- Start with *random vectors*. Make words that occur close by come closer, words that don't occur close by become perpendicular.

What's word2vec?

```
def train(corpus: list, DIMSIZE: int):
  """
  train word2vec of dimension DIMSIZE on the given corpus (list of words).
  Eg:train(["the", "man", "was" "tall", "the", "quick", "brown", "fox"], 20)
  """
  vocab = set(corpus); VOCABSIZE = len(vocab)
  # map each unique word to an index for array indexing.
  vocab2ix = dict([(word, ix) for (ix, word) in enumerate(corpus)])
  # +ve and -ve sample vectors.
  # +ve vectors are random initialized, -ve vectors are zero initialized
  poss = np.rand((VOCABSIZE, DIMSIZE)); negs = np.zeros((VOCABSIZE, DIMSIZE))

  for wix in range(len(corpus)): # for every location in the corpus
    w = vocab2ix[corpus[wix]]  # find word at location,
    l = max(wix-WINDOWSIZE, 0); r = min(wix+WINDOWSIZE, len(corpus)-1) # take a window

    for w2ix in range(l, r+1): # word in window
        w2 = vocab2ix[corpus[w2ix]] # prallel.
        learn(l=poss[w], r=negs[w2], target=1.0)

    for _ in range(NNEGSAMPLES): # random words outside window.
        w2ix = random.randint(0, len(corpus)-1) # random word.
        w2 = vocab2ix[corpus[w2ix]]
      learn(l=poss[w], r=negs[w2], target=0.0) # perpendicular
  return { v: poss[vocab2ix[v]] for v in vocab }
```

What's word2vec?

```
def learn(l: np.array, r:np.array, target: float):
  """
  gradient descent on
  loss = (target - dot(l, r))^2 where l = larr[lix]; r = rarr[rix]
  """
  dot = np.dot(l, r); grad_loss = 2 * (target - out)
  #dloss/dl = 2 * (target - dot(l, r)) r
  #dloss/dr = 2 * (target - dot(l, r)) l
  lgrad =  EPSILON * grad_loss * r; rgrad =  EPSILON * grad_loss * l
  # l -= eps * dloss/dl; r -= eps * dloss/dr
  l += EPSILON * grad_loss * r;
  r += EPSILON * grad_loss * l

def train(corpus: list, DIMSIZE: int):
    for w2ix in range(l, r+1): # positive samples, parallell
        w2 = vocab2ix[corpus[w2ix]] # word in window
        learn(l=poss[w], r=negs[w2], target=1.0)
    for _ in range(NNEGSAMPLES): # negative samples: perpendicular.
        w2ix = random.randint(0, len(corpus)-1) # random word outside window.
        learn(l=poss[w], r=negs[w2], target=0.0) # perpendicular
```

Part I: What's a philosopher to do?

- Montague semantics: The *meaning* of a word is the *set* of possible worlds where the meaning holds true.

Part I: What's a philosopher to do?

- Montague semantics: The *meaning* of a word is the *set* of possible worlds where the meaning holds true.
- A mathematical analogy: The *meaning* of an expression $\forall x \in \mathbb{Z}, x \leqslant 2$ is...

Part I: What's a philosopher to do?

- Montague semantics: The *meaning* of a word is the *set* of possible worlds where the meaning holds true.
- A mathematical analogy: The *meaning* of an expression $\forall x \in \mathbb{Z}, x \leqslant 2$ is...
- the *set* of possible values where the meaning holds true: $(-\infty, 2] = \{x \in \mathbb{Z} : x \leqslant 2\}$.

Part I: What's a philosopher to do?

- Montague semantics: The *meaning* of a word is the *set* of possible worlds where the meaning holds true.
- A mathematical analogy: The *meaning* of an expression $\forall x \in \mathbb{Z}, x \leqslant 2$ is...
- the *set* of possible values where the meaning holds true: $(-\infty, 2] = \{x \in \mathbb{Z} : x \leqslant 2\}$.
- Meaning $\simeq$ subsets. Is `word2vec` subsets?

Part I: What's a philosopher to do?

- Montague semantics: The *meaning* of a word is the *set* of possible worlds where the meaning holds true.
- A mathematical analogy: The *meaning* of an expression $\forall x \in \mathbb{Z}, x \leqslant 2$ is...
- the *set* of possible values where the meaning holds true: $(-\infty, 2] = \{x \in \mathbb{Z} : x \leqslant 2\}$.
- Meaning $\simeq$ subsets. Is `word2vec` subsets? Yes, *fuzzy sets*.

Part I: What's a philosopher to do?

- Montague semantics: The *meaning* of a word is the *set* of possible worlds where the meaning holds true.
- A mathematical analogy: The *meaning* of an expression $\forall x \in \mathbb{Z}, x \leqslant 2$ is...
- the *set* of possible values where the meaning holds true: $(-\infty, 2] = \{x \in \mathbb{Z} : x \leqslant 2\}$.
- Meaning $\simeq$ subsets. Is `word2vec` subsets? Yes, *fuzzy sets*.
- Set: binary membership. $(1 \in_? \{1, 2\} = T$, $3 \notin_? \{1, 2\} = F)$.

Part I: What's a philosopher to do?

- Montague semantics: The *meaning* of a word is the *set* of possible worlds where the meaning holds true.
- A mathematical analogy: The *meaning* of an expression $\forall x \in \mathbb{Z}, x \leqslant 2$ is...
- the *set* of possible values where the meaning holds true: $(-\infty, 2] = \{x \in \mathbb{Z} : x \leqslant 2\}$.
- Meaning $\simeq$ subsets. Is `word2vec` subsets? Yes, *fuzzy sets*.
- Set: binary membership. $(1 \in_? \{1, 2\} = T$, $3 \notin_? \{1, 2\} = F)$.
- Fuzzy set: probabilistic membership. $(1 \in_{fuz} F = 0.1$, $2 \in_{fuz} F = 0.5)$.
- Formally: $A \equiv \{(x, \mu_A(x)), x\}$. x is an element of set A with a probability $\mu_A(x)$ such that $0 \leqslant \mu_A(x) \leqslant 1$

The hidden sets in `word2vec`

- Given the set of vectors, normalize the ith component of the vector across *all vectors*.

The hidden sets in word2vec

- Given the set of vectors, normalize the ith component of the vector across *all vectors*.
- $\texttt{fuzembed}_{\texttt{word}}[i] \equiv e^{\texttt{vecembed}_{\texttt{word}}[i]} / \sum_{w \in \texttt{CORPUS}} e^{\texttt{vecembed}_{w}[i]}$.

The hidden sets in `word2vec`

- Given the set of vectors, normalize the ith component of the vector across *all vectors*.
- $\texttt{fuzembed}_{\texttt{word}}[i] \equiv e^{\texttt{vecembed}_{\texttt{word}}[i]} / \sum_{w \in \texttt{CORPUS}} e^{\texttt{vecembed}_w[i]}$.
- Fuzzy set embedding from `word2vec` embeddings.

The hidden sets in `word2vec`

- Given the set of vectors, normalize the ith component of the vector across *all vectors*.
- $\texttt{fuzembed}_{\texttt{word}}[i] \equiv e^{\texttt{vecembed}_{\texttt{word}}[i]} / \sum_{w \in \texttt{CORPUS}} e^{\texttt{vecembed}_{w}[i]}$.
- Fuzzy set embedding from `word2vec` embeddings.
- The projection of $\vec{v}$ on a dimension i normalized is to be interpreted as if this dimension i were a property, what is probability that v would possess that property?

What does this buy us anyway? (Set operations)

$$(A \cap B)[i] \equiv A[i] \times B[i] \quad \text{(set intersection)}$$

$$(A \cup B)[i] \equiv A[i] + B[i] - A[i] \times B[i] \, \text{(set union)}$$

$$(A \sqcup B)[i] \equiv \max(1, \min(0, A[i] + B[i])) \, \text{(disjoint union)}$$

$$(\neg A)[i] \equiv 1 - A[i] \quad \text{(complement)}$$

$$(A \setminus B)[i] \equiv A[i] - \min(A[i], B[i]) \quad \text{(set difference)}$$

$$(A \subseteq B) \equiv \forall x \in \Omega : \mu_A(x) \leqslant \mu_B(x) \, \text{(set inclusion)}$$

$$|A| \equiv \sum_{i \in \Omega} \mu_A(i) \quad \text{(cardinality)}$$

What does this buy us anyway? (Set intersection)

$\hat{N}$	$\hat{M}$	$\hat{G}$	$\hat{N} \cap \hat{M}$	$\hat{N} \cap \hat{G}$
nobility	metal	bad	fusible	good
isotope	fusible	manners	unreactive	dharma
fujwara	ductility	happiness	metalloids	morals
feudal	with	evil	ductility	virtue
clan	alnico	excellent	heavy	righteous
$\vec{N}$	$\vec{M}$	$\vec{G}$	$\vec{N} + \vec{M}$	$\vec{N} + \vec{G}$
noblest	trivalent	bad	fusible	gracious
auctoritas	carbides	natured	metals	virtuous
abies	metallic	humoured	sulfides	believeth
eightfold	corrodes	selfless	finntroll	savages
vojt	alloying	gracious	rhodium	hedonist

- Polysemy of the word `noble`, in the context of the words `good` and `metal`.
- `noble` is represented by N, `metal` by M and `good` by G.
- We also provide the word2vec analogues of the same, under $\vec{N}$, $\vec{M}$, and $\vec{G}$.
- See that `word2vec` has no analogue for set-intersection. We use the closest possible analogue (addition), which performs worse semantically.

Crash course on entropy

- Given a probability $p \in [0, 1]$, define the *surprisal* to be $-\log_2 p$.

Crash course on entropy

- Given a probability $p \in [0, 1]$, define the *surprisal* to be $-\log_2 p$.
- If $p = 1$, and the event happens, then we are never $(-\log_2 1 = 0)$ surprised.

Crash course on entropy

- Given a probability $p \in [0, 1]$, define the *surprisal* to be $-\log_2 p$.
- If $p = 1$, and the event happens, then we are never $(-\log_2 1 = 0)$ surprised.
- If $p = 0$, and the event happens, then we are terrifically $(-\log_2 0 = - - \infty = +\infty)$ surprised.

Crash course on entropy

- Given a probability $p \in [0,1]$, define the *surprisal* to be $-\log_2 p$.
- If $p = 1$, and the event happens, then we are never $(-\log_2 1 = 0)$ surprised.
- If $p = 0$, and the event happens, then we are terrifically $(-\log_2 0 = --\infty = +\infty)$ surprised.
- If $p = 1/2$, and the event happens, then we are terrifically $(-\log_2 1/2 = --1 = +1)$ surprised.

Crash course on entropy

- Given a probability $p \in [0, 1]$, define the *surprisal* to be $-\log_2 p$.
- If $p = 1$, and the event happens, then we are never $(-\log_2 1 = 0)$ surprised.
- If $p = 0$, and the event happens, then we are terrifically $(-\log_2 0 = - - \infty = +\infty)$ surprised.
- If $p = 1/2$, and the event happens, then we are terrifically $(-\log_2 1/2 = - - 1 = +1)$ surprised.
- Given a distribution $P : X \to [0, 1]$, the entropy of P is the *average surprise*, given by $\sum_{x \in X} P(x) \cdot -\log P(x)$.

What does this buy us anyway? (Entropy)

Fuzzy entropy is a measure of the uncertainty of the elements belonging to the set.

What does this buy us anyway? (Entropy)

Fuzzy entropy is a measure of the uncertainty of the elements belonging to the set.

$$
\begin{aligned}
H(A) &\equiv \sum_i H(X_i^A) \\
&\equiv \sum_i -p_i^A \ln p_i^A - (1 - p_i^A) \ln\left(1 - p_i^A\right) \\
&\equiv \sum_i -A[i] \ln A[i] - (1 - A[i]) \ln(1 - A[i])
\end{aligned}
$$

and	the	in	one	which	to	however	*two*	for	*eight*
this	of	of	in	the	*zero*	to	is	a	for
as	and	only	a	also	*nine*	it	as	but	*s*

- Function words are words which are largely syntactic rather than semantic.
- On the left: Top 15 words with highest entropy with frequency $\geqslant 100$. (note that all of them are function words).
- On the right: Top 15 words with the highest frequency.
- Non-function words are emphasized for comparison.

What does this buy us anyway? (KL divergence)

- K-L (Kullback Leibler) divergence is an asymmetric measure of similarity.
- Given data d which follows distribution P, the extra bits need to store it under the false assumption that the data d follows distribution Q is the K-L divergence between the distributions P and Q.
- Let P be the distribution that assigns 0.25 probability to a, b, c, d. Since all are equiprobable, we use 2 bits per character.
- Let Q be the distribution that assigns 0.5 probability to a, b and 0 probability to c, d. We use 1 bit to represent if we are storing a or b.
- If the real distribution is Q and we store data using P, then we really need only $\{a, b\}$, but we are trying to store $\{a, b, c, d\}$. P(false assumption) needs twice as many bits as Q(true distribution) to store the message c.
- If the real distribution is P and we store data using Q, then we really need $\{a, b, c, d\}$, but we *can only store* $\{a, b\}$. Q(false assumption) need *infinitely* more bits to store the message c than P (true distribution).

What does this buy us anyway? (KL divergence)

$$KL(S,T) \equiv \sum_i KL(X_i^S, X_i^T) = \sum_i p_i^S \log\left(p_i^S / p_i^T\right)$$

Example 1	$KL(ganges, delta)$	6.3105
	$KL(delta, ganges)$	6.3040
Example 2	$KL(north \cap korea, china)$	1.02923
	$KL(china, north \cap korea)$	10.60665

- K-L divergence shows the relation between two words.
- Can also consider phrases when composed using feature intersection as in the case of north korea.
- We demonstrate human annotator judgement of the distance between China and North Korea, where human annotators considered "North Korea" to be very similar to "China", while the reverse relationship was rated as significantly less strong ("China" is not very similar to "North Korea")

What does this buy us anyway? (Cross entropy)

- cross-entropy of two distributions P and Q is the sum of the entropy of P and the K-L divergence between P and Q.
- captures both the *uncertainty in* P, as well as the distance from P to Q.
- Gives information theoretic difference between the concepts of P and Q.

What does this buy us anyway? (Analogy)

$$a : b :: x : y_?$$
$$y_? = b - a + x \implies y_? = (b + x) - a$$
$$y = (b \sqcup x) \setminus a \quad \text{(Set-theoretic interpretation)}$$

- given a pairing $(a : b)$, and a prior x, we are asked to compute an unknown word $y_?$ such that $a : b :: x : y_?$
- In the vector space model, analogy is computed based on vector distances. But this is semantically incoherent, as we must then re-normalize vectors.
-

Word 1	Word 2	Word 3	word2vec	Our representation
bacteria	tuberculosis	virus	polio	hiv
cold	freezing	hot	evaporates	boiling
ds	nintendo	dreamcast	playstation	sega
pool	billiards	karate	taekwondo	judo

- Examples of analogy compared to the analogy in word2vec. We see here that the comparisons constructed by feature representations are similar to those given by the standard word vectors.

Evaluation: Similarity

Dims.	**word2vec**	**Our Representation**	
		K-L Divergence	Cross-Entropy
20	0.2478	0.2690	**0.2744**
50	0.2916	0.2966	**0.2981**
100	0.2960	0.3124	**0.3206**
200	0.3259	0.3253	**0.3298**

- Similarity scores on the SimLex-999 dataset for various dimension sizes (Dims.).

Evaluation: Analogy

Category		word2vec		Our representation	
		50	100	50	100
Capital Common Countries		21.94	37.55	**39.13**	**47.23**
Capital World		13.02	20.10	**27.30**	**26.54**
Currency		12.24	18.60	**25.27**	**24.90**
City-State		10.38	16.70	**23.24**	**23.51**
Family		10.61	17.34	**23.67**	**23.88**
Adjective-Adverb	Syntactic	4.74	3.23	**7.26**	**3.83**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	9.92	15.68	**21.73**	**21.52**
Opposite	Syntactic	4.06	3.66	**7.61**	**4.92**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	9.36	14.73	**20.60**	**20.26**
Comparative	Syntactic	8.86	12.63	**16.88**	**15.39**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	10.10	15.96	**21.67**	**21.39**
Superlative	Syntactic	7.59	11.30	**14.32**	**13.36**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	9.54	15.20	**20.35**	**20.15**
Present-Participle	Syntactic	7.51	10.96	**14.31**	**13.14**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	9.34	14.73	**19.84**	**19.49**
Nationality	Syntactic	12.51	19.07	**21.64**	**21.96**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	11.51	18.16	**22.71**	**22.97**
Past Tense	Syntactic	11.65	17.09	**20.43**	**19.76**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	11.16	17.21	**21.96**	**27.72**
Plural	Syntactic	11.76	17.23	**20.53**	**19.89**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	11.26	17.28	**21.90**	**21.64**
Plural Verbs	Syntactic	11.36	16.60	**19.88**	**19.46**
	Semantic	10.61	17.34	**23.67**	**23.88**
	Overall	11.05	16.91	**21.46**	**21.30**

- Comparison of Analogies between word2vec and our representation for 50 and 100 dimensions (Dims.).
- Outperform word2vec on every single metric.

Evaluation: Function word detection

top n words	**word2vec**	**Our Representation**
15	10	**15**
30	21	**30**
50	39	**47**

- Function word detection using entropy (in our representation) and by frequency in word2vec.
- We see that we consistently detect more function words than word2vec, based on the 176 function word list (Making and Using Word Lists for Language Learning and Teaching).
- The metric is *number of words*, i.e. the number of words chosen by frequency for word2vec and entropy for our representation.
- We detect more function words than the baseline frequency based methods.

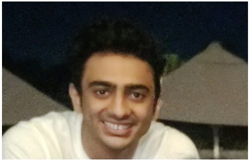
Evaluation: Compositionality detection

Dims.	Metric	word2vec	Our Representation
50	Spearman	0.3946	**0.4117**
	Pearson	0.4058	**0.4081**
100	Spearman	0.4646	**0.4912**
	Pearson	0.4457	**0.4803**
200	Spearman	0.4479	**0.4549**
	Pearson	**0.4163**	0.4091

- Predict whether two words combine to create a phrase or not (eg. monkey business, silver bullet)
- We decide that a phrase $w_1 w_2$ is a phrase if $|KL(w_1, w_2) - KL(w_2, w_1)|$ is large, as this implies information asymmetry.
- We see that almost across the board, we perform better.

Conclusion

- `word2vec` is performant but poorly understood.
- We extract fuzzy set embeddings from `word2vec`, giving richer, understandable variants of vector-based operations!
- TL;DR: Mathematical modelling (fuzzy sets) is useful to extend empirical results (`word2vec`)!
- `https://www.aclweb.org/anthology/2020.repl4nlp-1.4/`
- Collaborators: Alok Debnath, Souvik Banerjee.
- Advisors: Dr. Kannan Srinathan, Dr Manish Shrivastava.

Extensions

- The only thing we need to train `word2vec` is a dot-product.

Extensions

- The only thing we need to train `word2vec` is a dot-product.
- Can we generalize? `word2vec` consider *directions*, which are 1D subspaces.

Extensions

- The only thing we need to train `word2vec` is a dot-product.
- Can we generalize? `word2vec` consider *directions*, which are 1D subspaces.
- We should generalize! Use nD subspaces.

Extensions

- The only thing we need to train `word2vec` is a dot-product.
- Can we generalize? `word2vec` consider *directions*, which are 1D subspaces.
- We should generalize! Use nD subspaces.
- The collection of all nD subspaces of a vector space is known as the *Grassmanian*.

Extensions

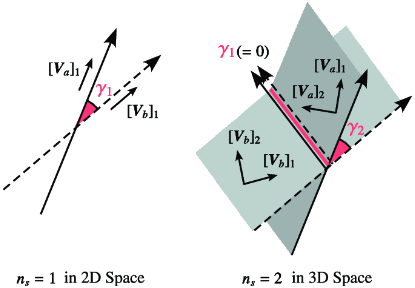
- The only thing we need to train `word2vec` is a dot-product.
- Can we generalize? `word2vec` consider *directions*, which are 1D subspaces.
- We should generalize! Use nD subspaces.
- The collection of all nD subspaces of a vector space is known as the *Grassmanian*.

Training on the grassmanian

- The collection of all nD subspaces of a vector space is known as the *Grassmanian*.

Training on the grassmanian

- The collection of all nD subspaces of a vector space is known as the *Grassmanian*.
- What is the analogue of the dot-product? It is an operation known as *Angle between flats*.



- How does one take gradients on the Grassmanian?

Results for training on the grassmanian [Sovik Banerjee's research]

- We evaluate the quality of our document embed-dings with the 20newsgroups topic classification.
- k-NN algorithm on Grassmannian manifold is performed by replacing the euclidean metric with grassmanian-compatible metric

Embedding	F1-Macro	F1-Micro
Avg. W2V	0.630	0.631
SIF	0.552	0.549
Doc2Vec	0.648	0.645
JoSE	0.703	0.707
Grassmannian	**0.749**	**0.752**

- we performed sentiment analysis on the imdb movie reveiew dataset using kernel SVM.
- We see that our model slightly outperforms the other embeddings model.

Embedding	Accuracy
Avg. W2V	88.02
SIF	85.32
Doc2Vec	88.52
JoSE	87.95
Grassmannian	**88.90**

Abstract interpretation

- $\alpha : (C, \leqslant) \to (A, \sqsubset)$: monotone Abstraction from the Concrete to the Abstract.
- $\gamma : (A, \sqsubset) \to (C, \sqsubset)$: monotone Concretization from Abstract to the Concrete. (γ for Galois)
- $c \leqslant \gamma(\alpha(c))$: Abstraction can lose information
- $a = \alpha(\gamma(c))$: Concretization is faithful: abstract objects are well represented by concrete objects.