DEEP LEARNING FOR NATURAL LANGUAGE PROCESSING

Lecture 1: introduction, deep learning, language modeling, CNN

Caio Corro

ABOUT THE COURSE

Other (deep learning for) NLP courses

There are many very good courses available online, see the resource page on my website.

- **Good news for you:** plenty of resources to learn! :)
- **Bad news for me:** how can my course be useful for you? :(

How I build this course

1. I often observe many misconception in the NLP community
   (for example about attention, named entity recognition and BIO-tagging, CRFs, …)
2. I built the course around a selection of these topics so you don't make the same mistakes

ABOUT THE COURSE

Other (deep learning for) NLP courses

There are many very good courses available online, see the resource page on my website.

- **Good news for you:** plenty of resources to learn! :)
- **Bad news for me:** how can my course be useful for you? :(

How I build this course

1. I often observe many misconception in the NLP community
   (for example about attention, named entity recognition and BIO-tagging, CRFs, …)
2. I built the course around a selection of these topics so you don't make the same mistakes

Outline

- Study a few important neural architectures in details
  in order to understand the building blocks of most of neural networks used in NLP
- Introduction to structured prediction
- Learn « advanced » PyTorch tricks that are useful in NLP (and other fields)

ABOUT THE COURSE

Grading scheme

- 100%: lab exercises, 2-3 students per group

Lab exercises

- Lab 1: convolution for text classification (not graded)
- Lab 2: language modeling
- Lab 3: part-of-speech tagging

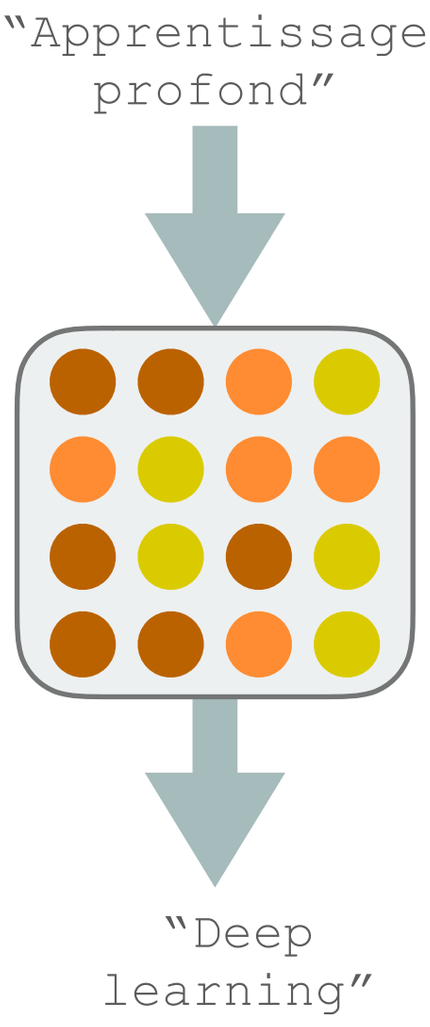
How will I grad projects?

- Experimental results are important,
  but the grade will be unrelated to how good your model works
- Quality of data analysis and result analysis
- Description of the model and what you explored (i.e. math, figures, etc)
- Explanation of your implementation (in the report!)

NATURAL LANGUAGE PROCESSING

Applications

- Machine Translation
- Information Retrieval
- Sentiment Analysis
- Automatic Summarization
- Human-Computer Interaction
- …



Data type

- Structured: language is sequential
- Discrete: text processing
- Continuous: speech processing

We will focus on text, even if many languages are unwritten!

Difficulties

- Many languages (> 6000)
- Ambiguity
- Constantly evolving
- Noisy

THE LIMIT OF RULE BASED SYSTEMS FOR NLP

- Programming languages: prescriptive grammar $\Rightarrow$ easy to build a rule-based system
- Natural languages: comes in many shapes and forms!

THE LIMIT OF RULE BASED SYSTEMS FOR NLP

- Programming languages: prescriptive grammar ⇒ easy to build a rule-based system
- Natural languages: comes in many shapes and forms!

Gender-inclusive language

Les étudiants sont en grève.

Les étudiant(e)s sont en grève.

Les étudiant.e.s sont en grève.

Les étudiant·e·s sont en grève.

Ils sont en grève.

Iels sont en grève.

THE LIMIT OF RULE BASED SYSTEMS FOR NLP

- Programming languages: prescriptive grammar ⇒ easy to build a rule-based system
- Natural languages: comes in many shapes and forms!

Gender-inclusive language

Les étudiants sont en grève.

Les étudiant(e)s sont en grève.

Les étudiant.e.s sont en grève.

Les étudiant·e·s sont en grève.

Ils sont en grève.

Iels sont en grève.

Sociocultural variation

Y a le mec de Milla

Y a le mec à Milla

Non-standard writing

I luv pizza.

I ❤️ 🍕

J'ème la pizza.

J'm la pizza.

Boomers and « l'Académie Française » are apparently not capable to adapt to variation and diversity in languages, but you want your software to be smarter than them!

THE LIMIT OF RULE BASED SYSTEMS FOR NLP

Problem: Named Entity Recognition (NER)

Task: given a large corpus a text files, extract all named entities:

- Person names
- Place names
- Dates
- Locations
- ...

After Sherborne, Turing studied as an undergraduate from 1931 to 1934 at King's College, Cambridge, where he was awarded first-class honours in mathematics. In 1935, at the age of 22, he was elected a fellow of King's on the strength of a dissertation in which he proved the central limit theorem. Unknown to the committee, the theorem had already been proven, in 1922, by Jarl Waldemar Lindeberg. A blue plaque at the college was unveiled on the centenary of his birth on 23 June 2012 and is now installed at the college's Keynes Building on King's Parade

Potential tags:
LOCATION
ORGANIZATION
DATE
MONEY
PERSON
PERCENT
TIME

Extract of https://en.wikipedia.org/wiki/Alan_Turing tagged with http://nlp.stanford.edu:8080/ner/

THE LIMIT OF RULE BASED SYSTEMS FOR NLP

Easy solution

Use a gazetteer:

Class	Gazetteer excerpt
Locations	sunderland, sundsvall, sundsvl, sunnyvale, sunrise, sunset beach, sunshine coast, suourland, suouroy
Organizations	xyratex, y e data, ya, yadkin, yahoo, yahoo maps, yahoo personal email, yahoo personals, yahoo sbc
Person Names	jerrine, jerrod, jerrol duane, jerrold, jerrome, jerry, jerry ben, jerry brown, jerry claude, jerry coleman
Other	january, february, sunday, monday, toyota prius, ford focus, apple ipod, microsoft zune, tylenol, advil

Gazetteer from [Carlson et al., 2009]

Downsides

- Language is ambiguous, e.g. « Tim Cook »
- Language is evolving, e.g. new people, new places, …
- Typos, « non-standard » writing (e.g. tweets)

In general for natural language processing

- More or less intuitive for a human who speaks a given language
- Fuzzy decision, lots of « weak » contradictory signals

Even for syntactic analysis!

*« The spokesperson, **Μιχάλης Χατζόπουλος**, has declared that… »*

SENTENCE CLASSIFICATION

Sentiment analysis

- Input: sentence
- Output: Positive? Neutral? Negative?

Inputs are of different length!

```
This movie is great!
I saw that movie some years ago.
The food was disgusting…
```

SENTENCE CLASSIFICATION

Sentiment analysis

- Input: sentence
- Output: Positive? Neutral? Negative?

Inputs are of different length!

```
This movie is great!
I saw that movie some years ago.
The food was disgusting…
```

Natural language inference

- Input: premise and hypothesis (2 sentences)
- Output: Entailment? Neutral? Contradiction?

```
John likes Baltimore a lot.
Johne likes Baltimore.
```

TAGGING

Part-of-speech tagging

- Input: sentence
- Output: grammatical category for each word of the sentence

```
PRP    VB    DET   NN
They   walk  the   dog
```

TAGGING

Part-of-speech tagging

- Input: sentence
- Output: grammatical category for each word of the sentence

PRP	VB	DET	NN
They	walk	the	dog

Named entity recognition with BIO tags

- Input: sentence
- Output: BIO tags + 6 classes
  (person, location, group, creative work, product, corporation)

B-Per	I-Per	O	O	B-Loc
Neil	Armstrong	visited	the	moon

CHUNKING

Named entity recognition

- Input: sentence
- Output: chunks (person, location, group, creative work, product, corporation)

CHUNKING

Named entity recognition

- Input: sentence
- Output: chunks (person, location, group, creative work, product, corporation)

Nested named entity recognition

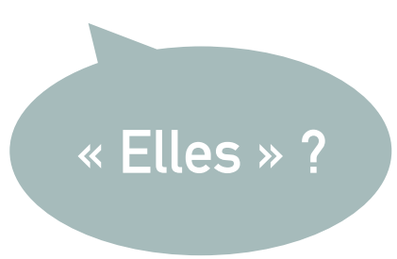
- Input: sentence
- Output: nested chunks

SEQUENCE GENERATION

Machine translation

- Input: sentence in a source language
- Output: sentence in a target language

They walk the dog ⇒ Ils promènent le chien



SEQUENCE GENERATION

Machine translation

- Input: sentence in a source language
- Output: sentence in a target language

`They walk the dog` ⇒ `Ils promènent le chien`

« Elles » ?

Image captioning

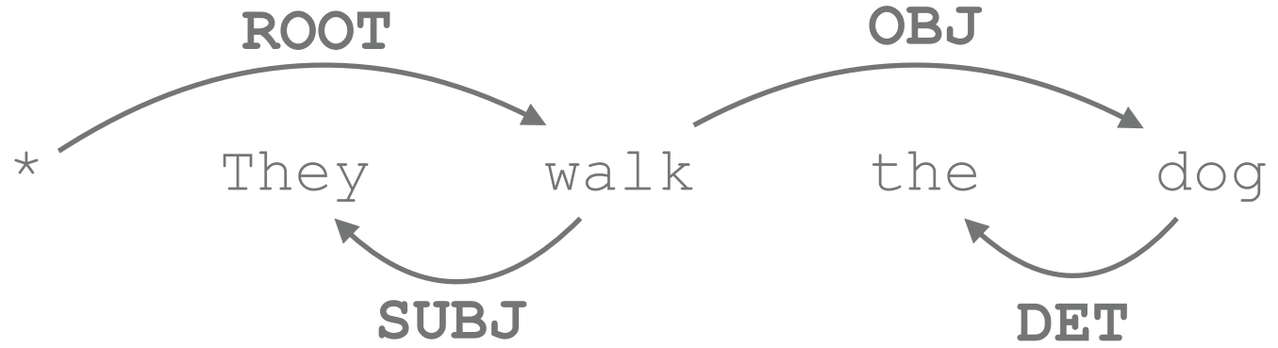
- Input: image
- Output: sentence describing the image

⇒ `Spongebob is cooking burgers`

SYNTACTIC PARSING

Syntactic dependency parsing

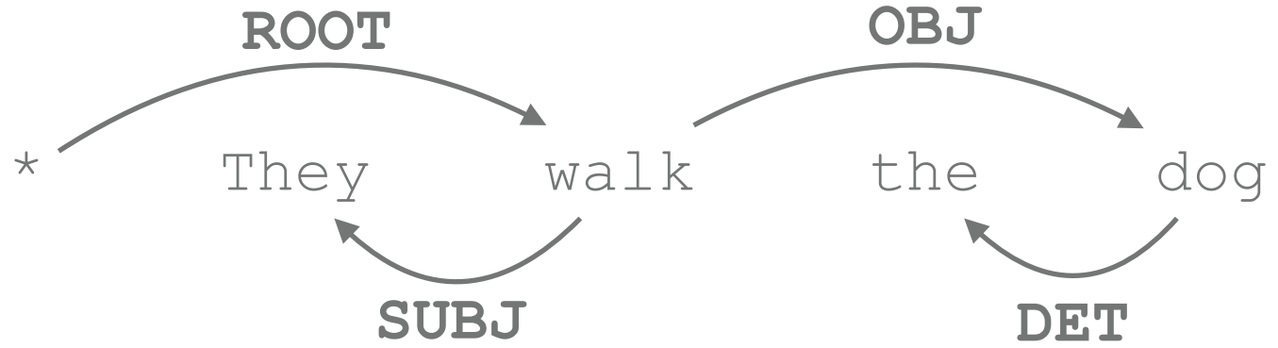
- Input: sentence
- Output: bi-lexical dependencies between words



SYNTACTIC PARSING

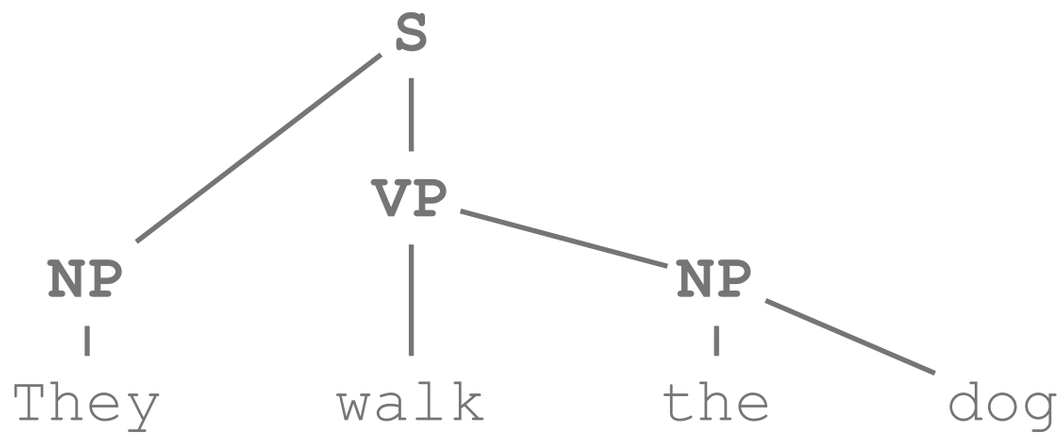
Syntactic dependency parsing

- Input: sentence
- Output: bi-lexical dependencies between words



Constituency parsing

- Input: sentence
- Output: hierarchical phrase-structure



SEMANTIC PARSING

SQL parsing

- Input: sentence
- Output: SQL query

```
I want to book a flight from Paris to Rome.
```

⇓

```
SELECT * FROM  flight WHERE  from = "paris" AND to = "rome"
```

SEMANTIC PARSING

SQL parsing

- Input: sentence
- Output: SQL query

```
I want to book a flight from Paris to Rome.
```

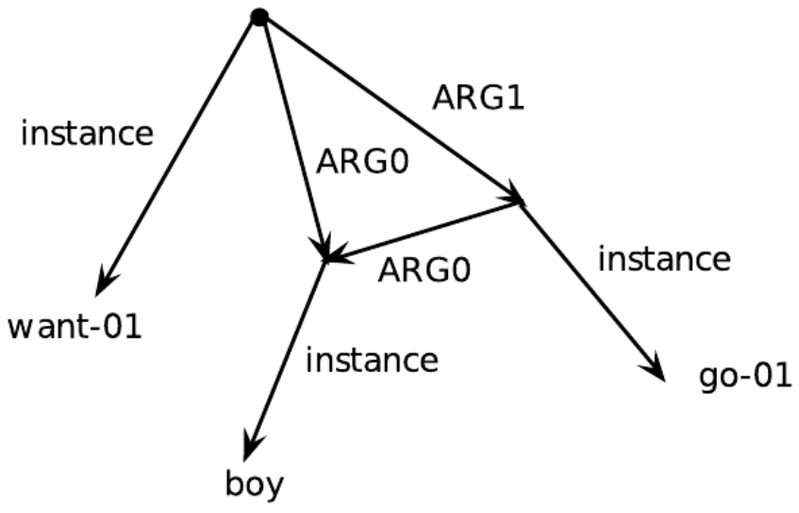
⇓

```
SELECT * FROM  flight WHERE  from = "paris" AND to = "rome"
```

Abstract Meaning Representation (AMR) parsing

- Input: sentence
- Output: graph

The boy want to go. ⇒



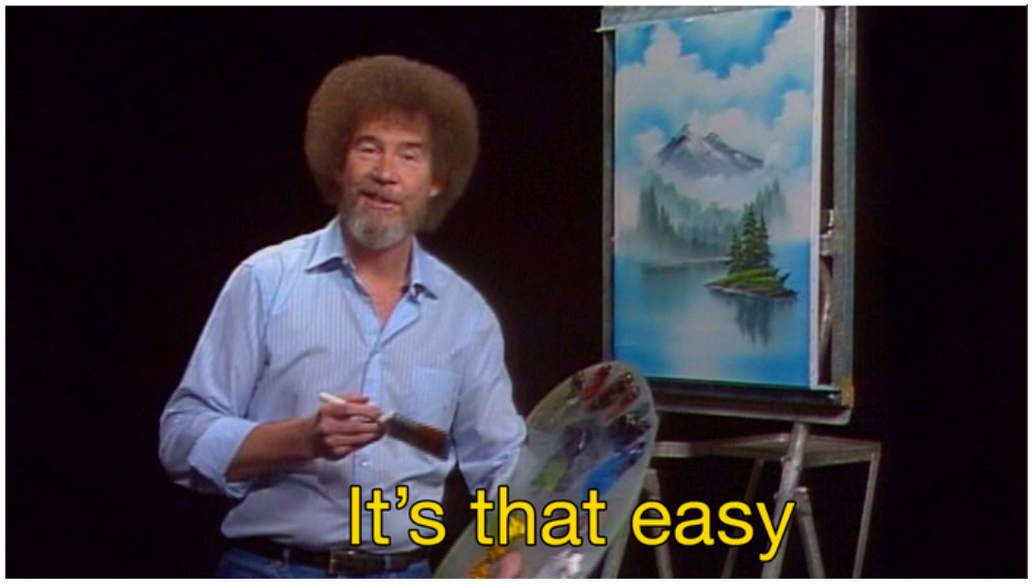
DEEP LEARNING

Neural networks

- Input: « raw » features, e.g. an image, a sentence
- Hidden layers: sequence of non-linear transformations
- Output: linear classification/regression

Parameter estimation cookbook

- Loss function minimization
- Stochastic gradient descent
- Regularization (weight decay, dropout, …)



DEEP LEARNING FOR NLP

Neural architectures

- How to efficiently represent text data?
- How to design the layers of the network?
- How to output structured representations?

This is what what makes deep learning for NLP interesting!

Training

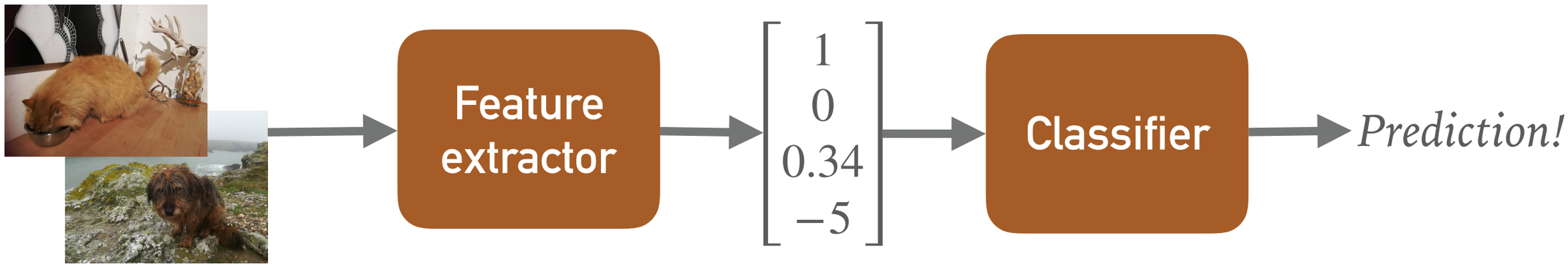
- Annotation in NLP is expensive: unsupervised pre-training is essential
- Structured outputs are difficult to learn: exponential search space

Problematic even for sequence generation

BACKGROUND: MACHINE LEARNING

RECALL

The « old school » machine learning pipeline



Example of classifiers

- Decision Tree:
  - Make a decision considering a limited number of features
  - Use conjunction of features to make a prediction
- K-nearest neighbors:
  - All features are used and considered equals
- Linear models:
  - Weight features so they are more or less important to make a decision

BINARY CLASSIFICATION

Data

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Output: $y \in \{0,1\}$ or $y \in \{-1,1\}$

Parameters

- Projection vector: $\mathbf{a} \in \mathbb{R}^d$
- Intercept: $b \in \mathbb{R}$

BINARY CLASSIFICATION

Data

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Output: $y \in \{0,1\}$ or $y \in \{-1,1\}$

Parameters

- Projection vector: $\mathbf{a} \in \mathbb{R}^d$
- Intercept: $b \in \mathbb{R}$

Decision

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{a}^\top \mathbf{x} + b \geq 0, \\ 0 & \text{otherwise.} \end{cases} \qquad \text{or} \qquad f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{a}^\top \mathbf{x} + b \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

BINARY CLASSIFICATION

Data

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Output: $y \in \{0,1\}$ or $y \in \{-1,1\}$

Parameters

- Projection vector: $\mathbf{a} \in \mathbb{R}^d$
- Intercept: $b \in \mathbb{R}$

Decision

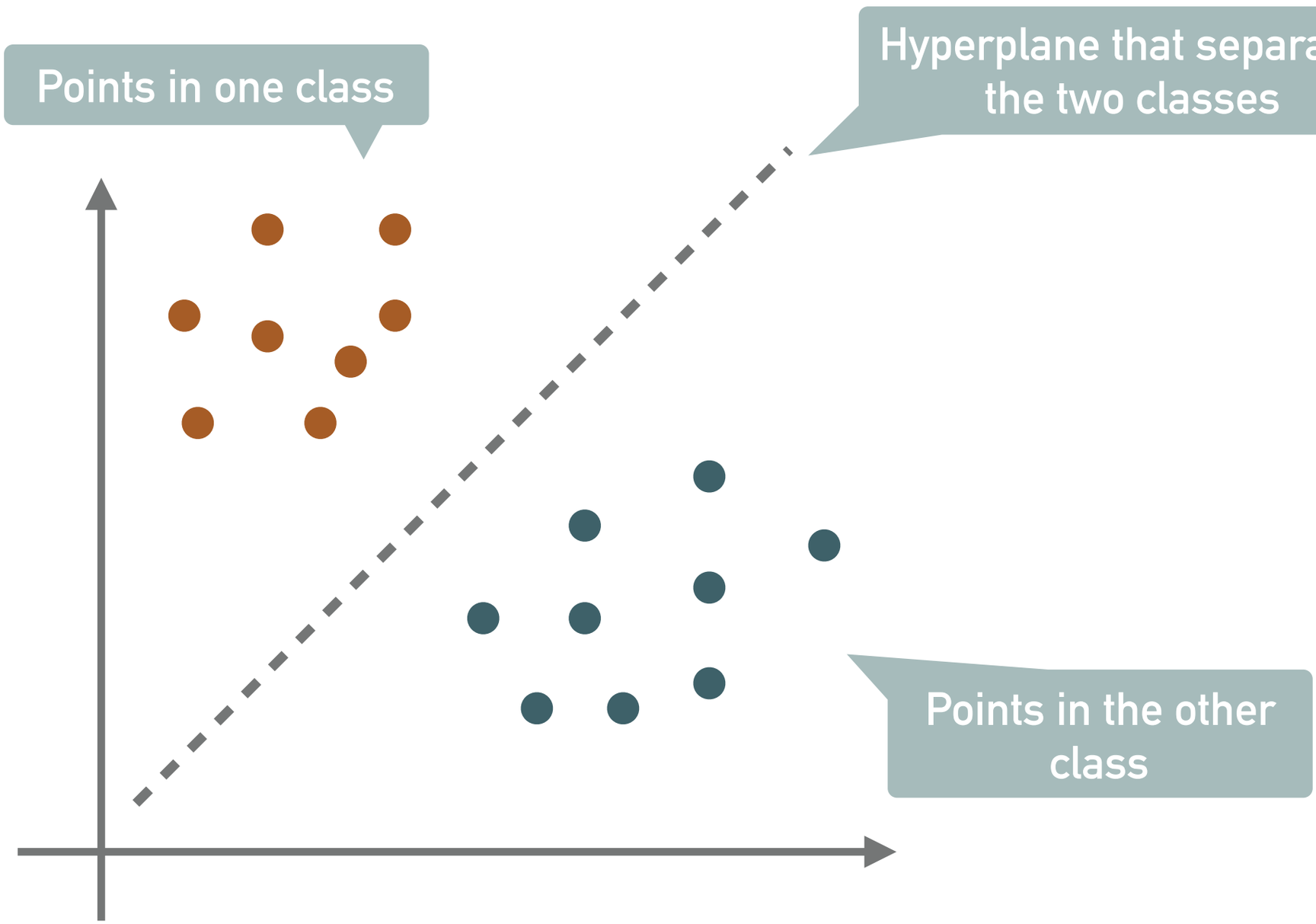
$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{a}^\top \mathbf{x} + b \geq 0, \\ 0 & \text{otherwise.} \end{cases} \quad \text{or} \quad f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{a}^\top \mathbf{x} + b \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

Probabilistic output

$$p(y = 1 \,|\, \mathbf{x}) = \sigma(\mathbf{a}^\top \mathbf{x} + b) = \frac{\exp(\mathbf{a}^\top \mathbf{x} + b)}{1 + \exp(\mathbf{a}^\top \mathbf{x} + b)}$$

$$p(y = 0 \,|\, \mathbf{x}) = 1 - \sigma(\mathbf{a}^\top \mathbf{x} + b)$$

INTUITION


Points in one class
Hyperplane that separates the two classes
Points in the other class

TRAINING 1/2

Training data

Actually not a set

Assume access to n iid labeled datapoints: $D = \left\{ \langle \mathbf{x}^{(i)}, y^{(i)} \rangle \right\}_{i=1}^{n}$

Loss function

Function defined as $\ell : \{0,1\} \times \mathbb{R}^d \to \mathbb{R}_+$

- Perceptron loss: $\ell(y, w) = \max(0, -w \times (2y - 1))$
- Binary hinge loss: $\ell(y, w) = \max(0, 1 - w \times (2y - 1))$
- Binary negative log-likelihood: $\ell(y, w) = -w \times y + \log(1 + \exp(w))$

Training objective

Regularization term

$$\min_{\substack{\mathbf{a} \in \mathbb{R}^d, \\ b \in \mathbb{R}}} \quad \frac{1}{n} \sum_{\langle \mathbf{x}, y \rangle \in D} \ell(y, \mathbf{a}^\top \mathbf{x} + b) \quad + \quad \alpha \times r(\mathbf{a})$$

TRAINING 2/2

Training objective

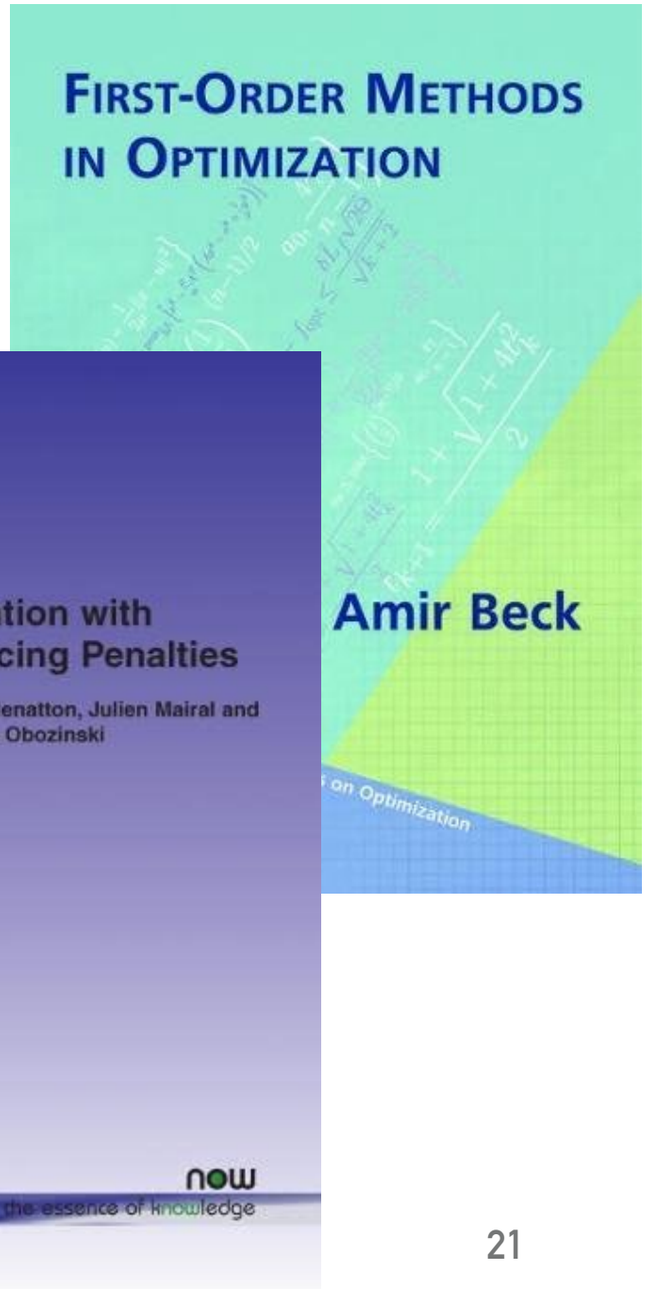
$$\min_{\substack{\mathbf{a} \in \mathbb{R}^d, \\ b \in \mathbb{R}}} \quad \frac{1}{n} \sum_{\langle \mathbf{x}, y \rangle \in D} \ell(y, \mathbf{a}^\top \mathbf{x} + b) \quad + \quad \alpha \times r(\mathbf{a})$$

Regularization term

Training algorithms

- Simple solution: (sub-)gradient descent
- Specialized solution
  (depending on the choice of loss and regularization)

Can be way faster!



(SUB-)GRADIENT DESCENT

$$\theta^{(t+1)} = \theta^{(t)} - \eta \times \nabla_\theta g(\theta)$$

For linear models!
Neural networks optimization relies on different heuristics

How to choose the step-size?

- Fixed step size
- Diminishing step size
- Line-search (search for the best step-size at a given point)

Convergence

- Theoretical guarantee
- Fast and efficient in practice

MULTICLASS CLASSIFICATION WITH K CLASSES

Data

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Output: $\mathbf{y} \in E(k)$ or $y \in \{1...k\}$

Set of one-hot vectors of size k

Parameters

- Projection vector: $\mathbf{A} \in \mathbb{R}^{k \times d}$
- Intercept: $\mathbf{b} \in \mathbb{R}^k$

MULTICLASS CLASSIFICATION WITH K CLASSES

Data

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Output: $\mathbf{y} \in E(k)$ or $y \in \{1...k\}$

Set of one-hot vectors of size k

Parameters

- Projection vector: $\mathbf{A} \in \mathbb{R}^{k \times d}$
- Intercept: $\mathbf{b} \in \mathbb{R}^k$

Decision

$$f(\mathbf{x}) = \underset{\mathbf{y} \in E(k)}{\operatorname{argmax}} \quad \mathbf{y}^\top (\mathbf{A}\mathbf{x} + \mathbf{b}) \qquad \textit{or} \qquad f(\mathbf{x}) = \underset{y \in \{1...k\}}{\operatorname{argmax}} \quad [\mathbf{A}\mathbf{x} + \mathbf{b}]_y$$

MULTICLASS CLASSIFICATION WITH K CLASSES

Data

- Input: $\mathbf{x} \in \mathbb{R}^d$
- Output: $\mathbf{y} \in E(k)$ or $y \in \{1...k\}$

Set of one-hot vectors of size k

Parameters

- Projection vector: $\mathbf{A} \in \mathbb{R}^{k \times d}$
- Intercept: $\mathbf{b} \in \mathbb{R}^k$

Decision

$$f(\mathbf{x}) = \underset{\mathbf{y} \in E(k)}{\operatorname{argmax}} \; \mathbf{y}^\top (\mathbf{A}\mathbf{x} + \mathbf{b}) \qquad or \qquad f(\mathbf{x}) = \underset{y \in \{1...k\}}{\operatorname{argmax}} \; [\mathbf{A}\mathbf{x} + \mathbf{b}]_y$$

Probabilistic output

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{\exp(\mathbf{y}^\top (\mathbf{A}\mathbf{x} + \mathbf{b}))}{\sum_{\mathbf{y}'} \exp(\mathbf{y}'^\top (\mathbf{A}\mathbf{x} + \mathbf{b}))}$$

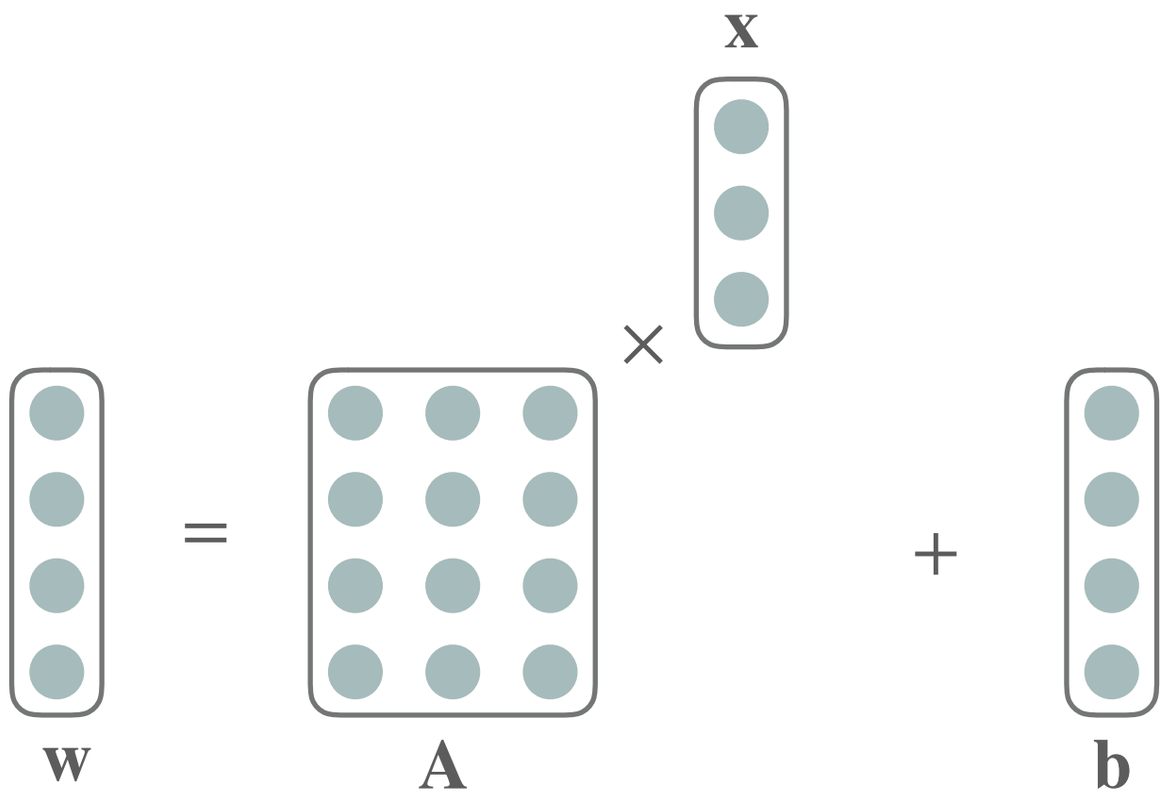
ILLUSTRATION

Problem

- Input: features
- Output: 1-in-k prediction
  (e.g. select the part-of-speech tag associated with a given word)

Linear classifier $\mathbf{w} = \mathbf{A}\mathbf{x} + \mathbf{b}$

- Input dim: 3
- Output dim: k=4 classes



TRAINING

Training data

Actually not a set

Assume access to n iid labeled datapoints: $D = \left\{ \langle \mathbf{x}^{(i)}, \mathbf{y}^{(i)} \rangle \right\}_{i=1}^{n}$

Loss function

Function defined as $\ell : E(k) \times \mathbb{R}^d \to \mathbb{R}_+$

- Hinge loss: $\ell(\mathbf{y}, \mathbf{w}) = \max(0, -\mathbf{w}^\top \mathbf{y} + 1 + \max_{\mathbf{y}' \neq \mathbf{y}} \mathbf{w}^\top \mathbf{y}')$
- Negative log-likelihood: $\ell(\mathbf{y}, \mathbf{w}) = -\mathbf{w}^\top \mathbf{y} + \log \sum_{\mathbf{y}'} \exp \mathbf{w}^\top \mathbf{y}'$

Training objective

Regularization term

$$\min_{\substack{\mathbf{A} \in \mathbb{R}^{k \times d}, \\ \mathbf{b} \in \mathbb{R}^k}} \quad \frac{1}{n} \sum_{\langle \mathbf{x}, \mathbf{y} \rangle \in D} \ell(\mathbf{y}, \mathbf{A}^\top \mathbf{x} + \mathbf{b}) \quad + \quad \alpha \times r(\mathbf{A})$$

LIMITS OF LINEAR CLASSIFIERS

TEXT CLASSIFICATION

Task

Predict if a movie review is good or bad.

« This movie is great! » *« Worst movie ever… »*

Features

1. Uni-gram features:
   - ➤ One feature per word in the vocabulary
   - ➤ The feature is equal to one if the word is in the review
2. Bi-gram features:
   - ➤ One feature per couple of words in the vocabulary
   - ➤ The feature is equal to one if the words appear consecutively in the review

FEATURE SET

Let V be the vocabulary.

Unigram features

- Number of features: |V|
- We can use a dictionary to map a word to an index in a vector

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix} \begin{matrix} bad \\ movie \\ great \\ worst \\ \\ house \end{matrix}$$

Bigram features

- Number of features: |V|x|V|
- Similarly, use a dict!

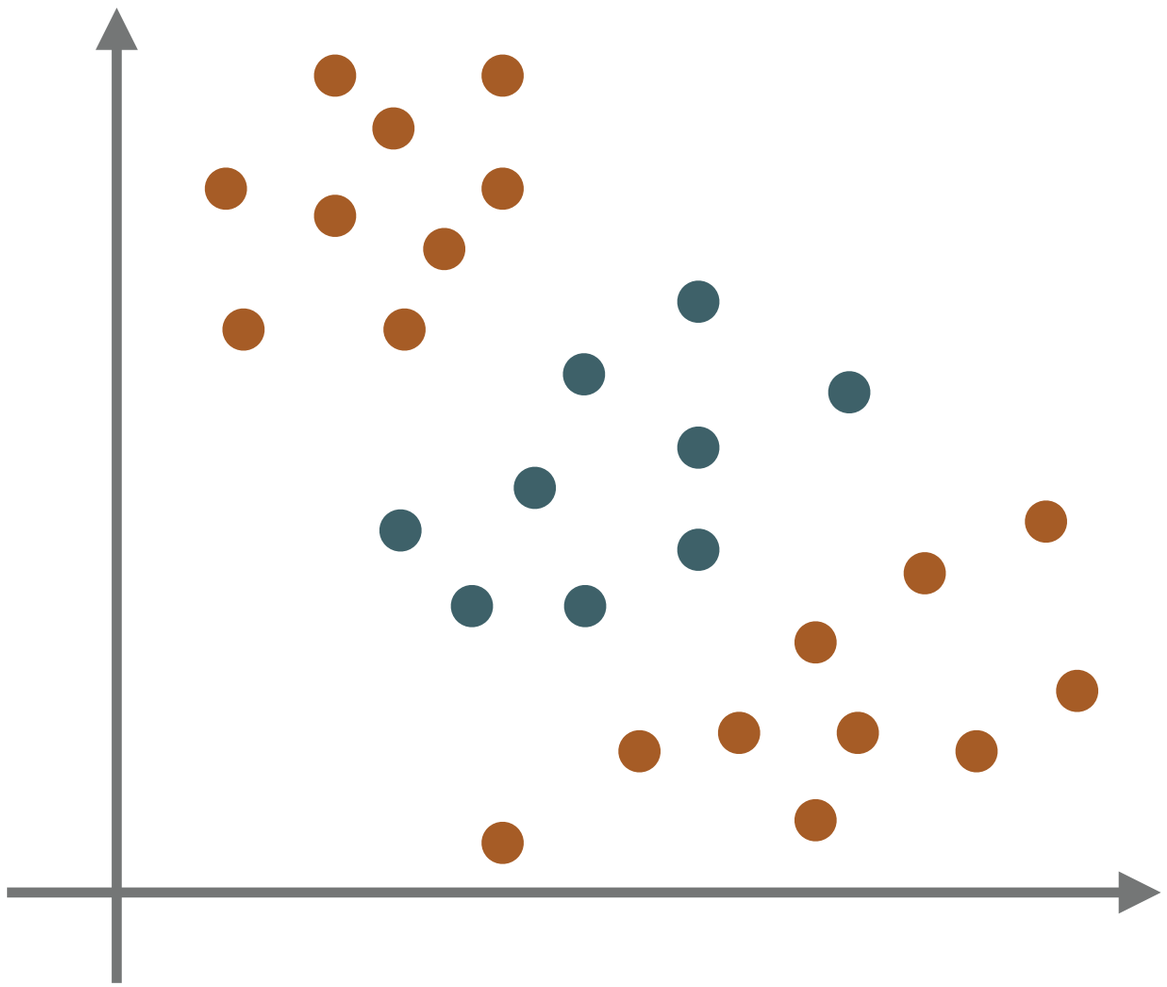
$$x = \begin{bmatrix} 1 \\ 0 \\ \dots \\ 1 \end{bmatrix} \begin{matrix} This\ movie \\ The\ movie \\ \\ is\ great \end{matrix}$$

Problem

- The feature vector is sparse
- The dot product can be unnecessarily expensive!
- There is no inductive bias: a sentence is not bag-of-words, it is a sequence!

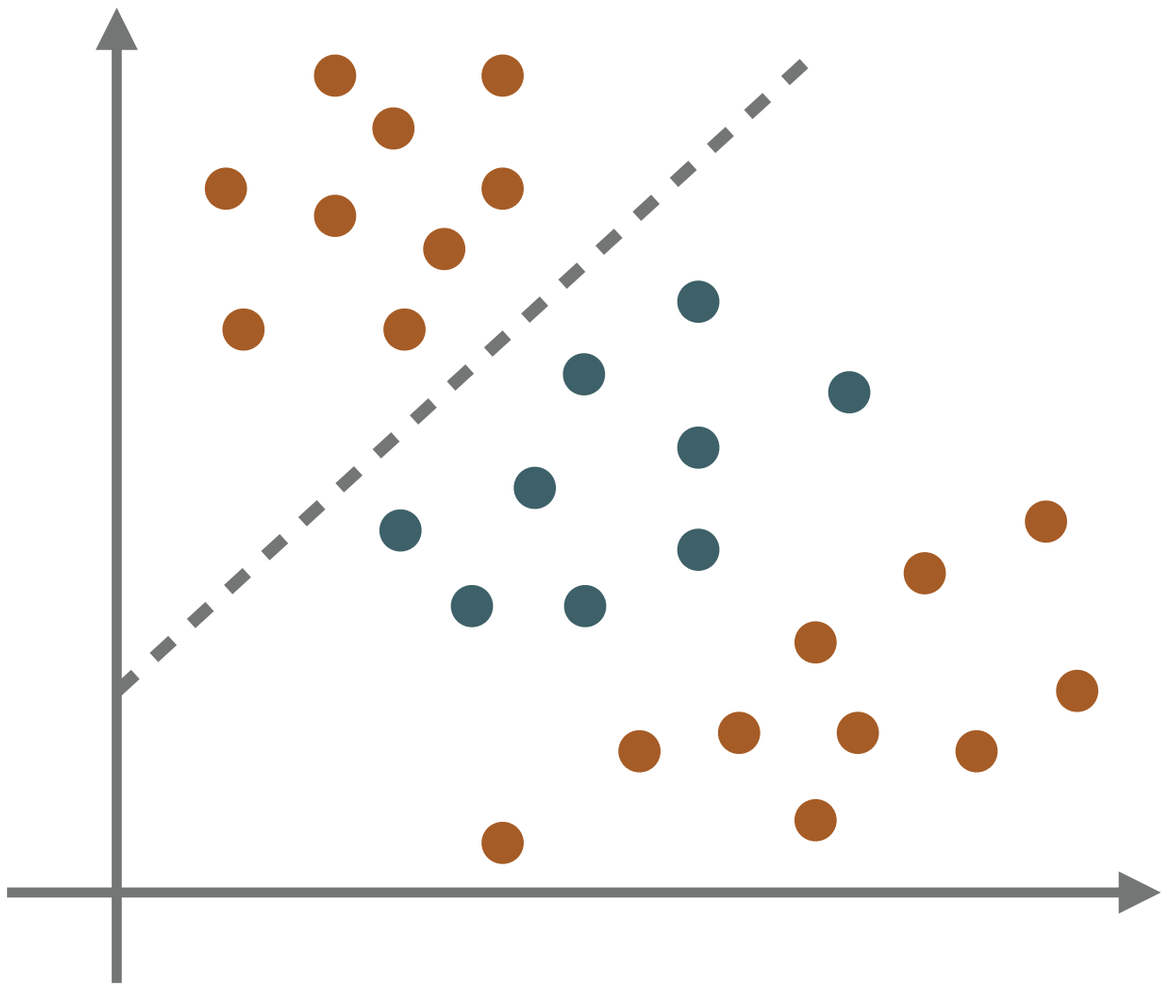
PROBLEMATIC CASES

- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**



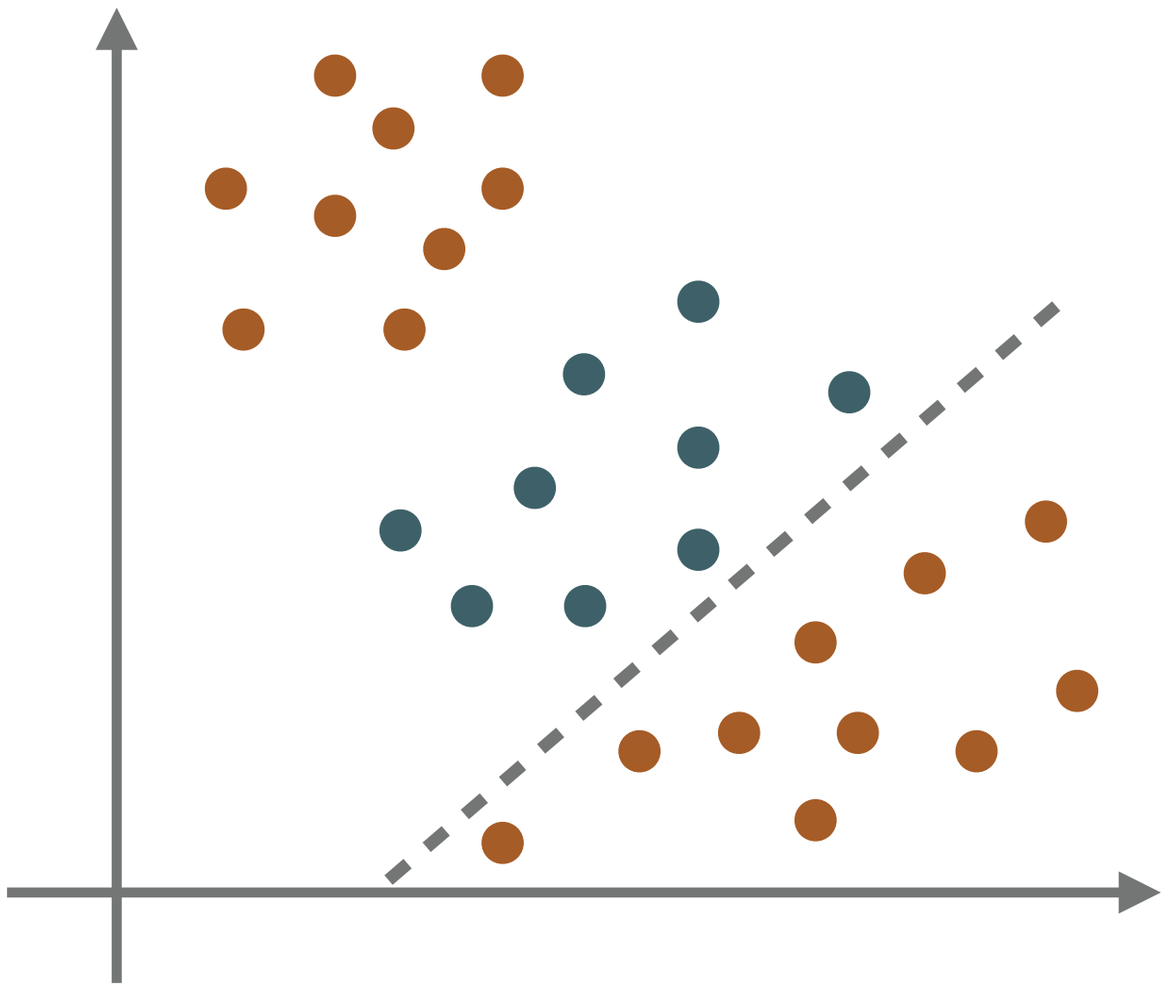
PROBLEMATIC CASES

- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**



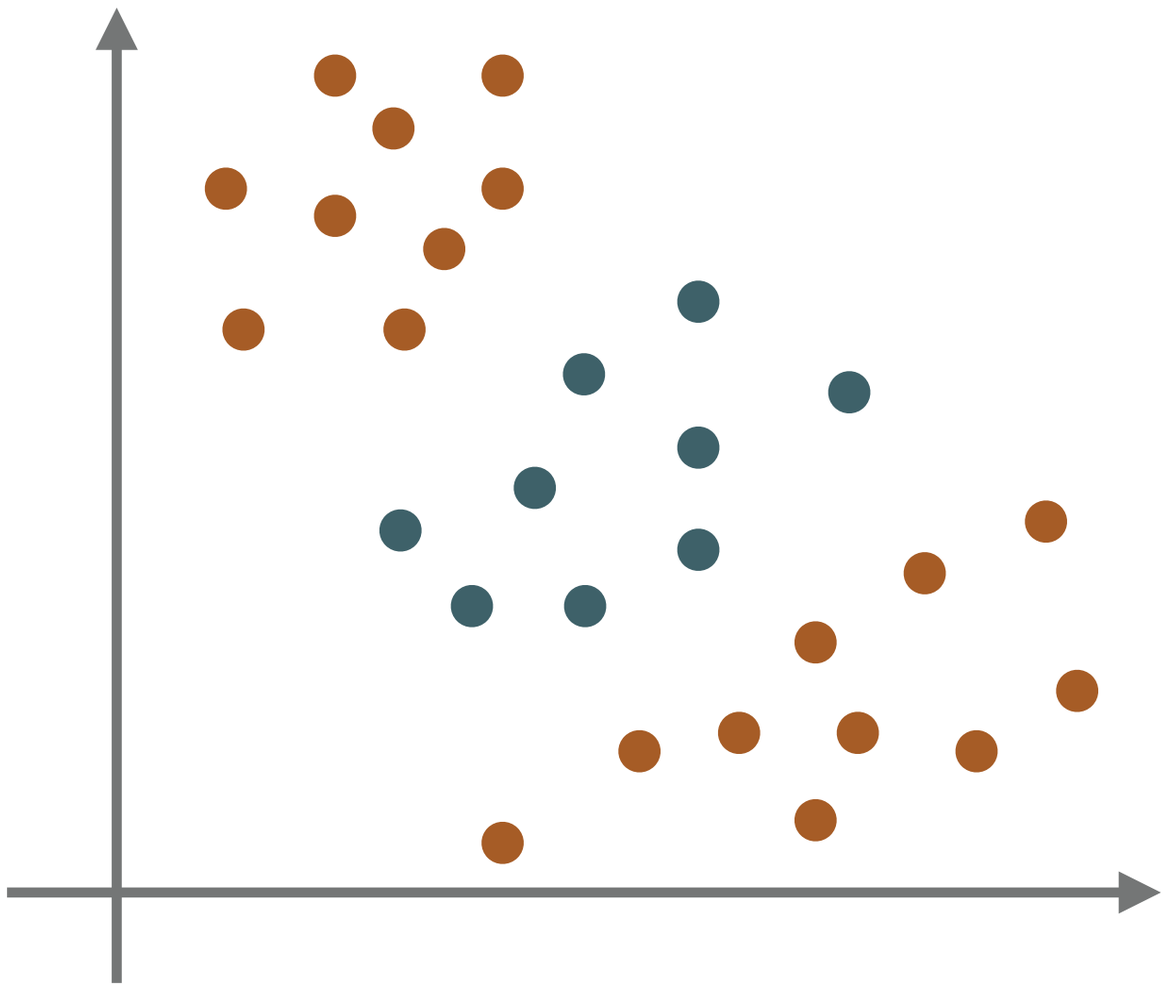
PROBLEMATIC CASES

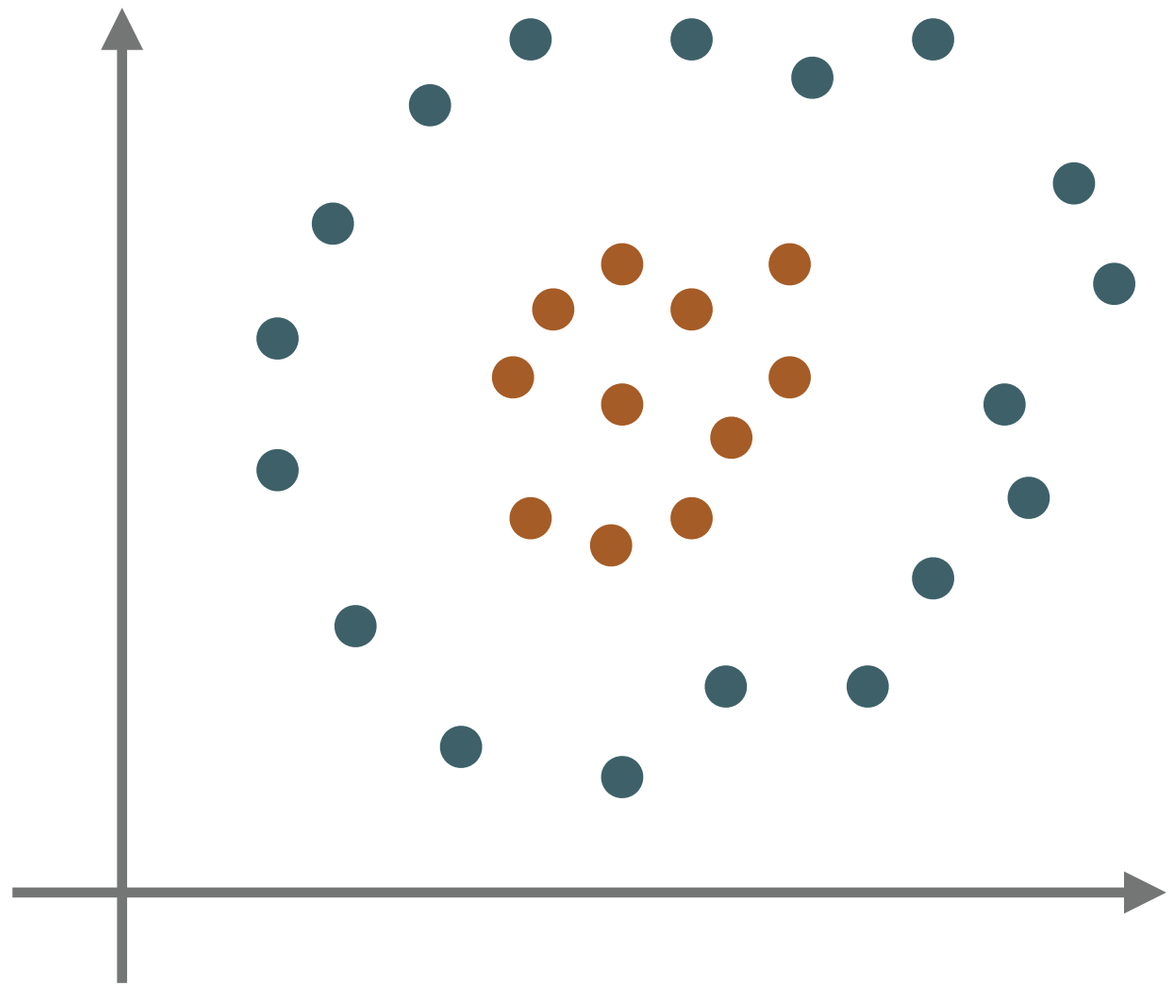
- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**



PROBLEMATIC CASES

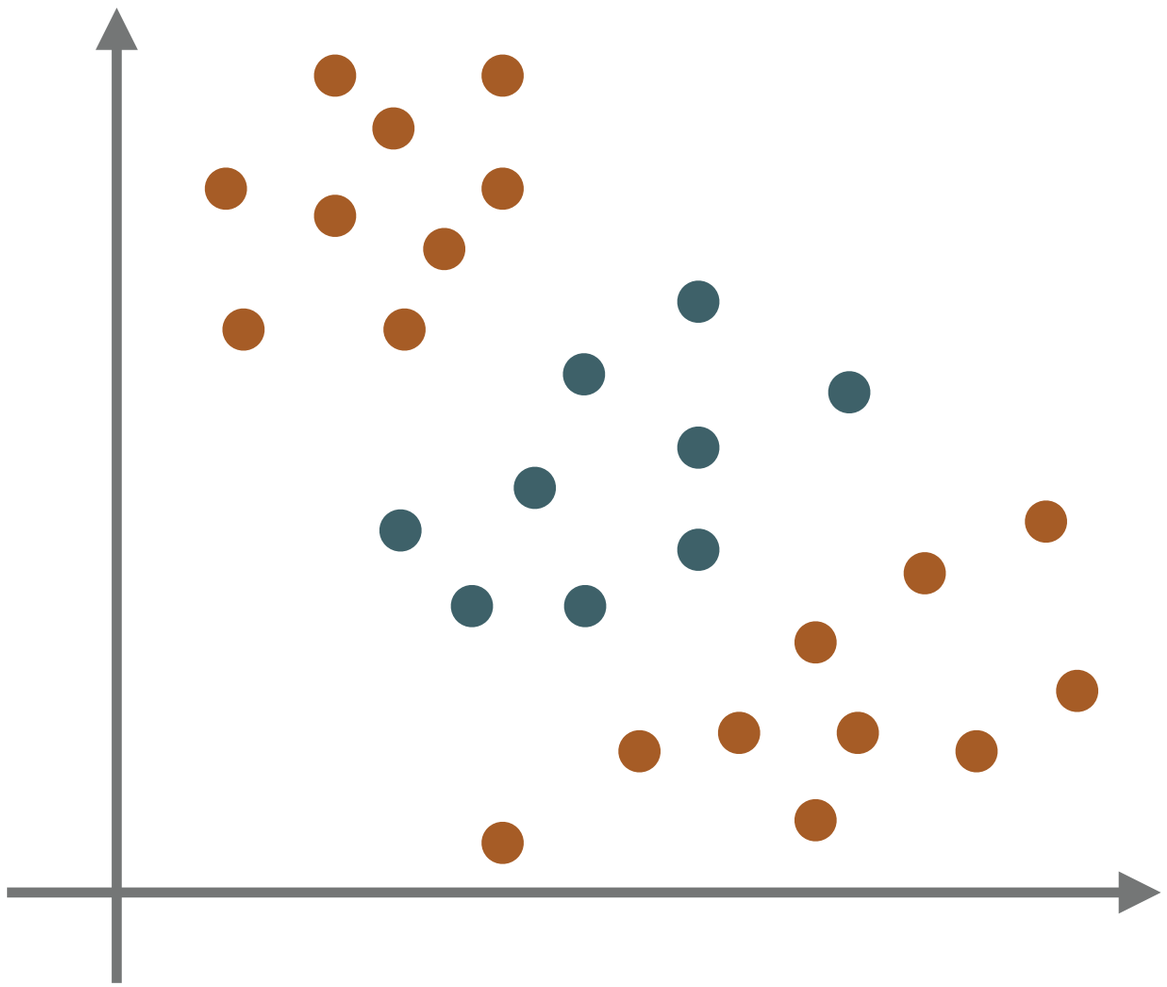
- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**

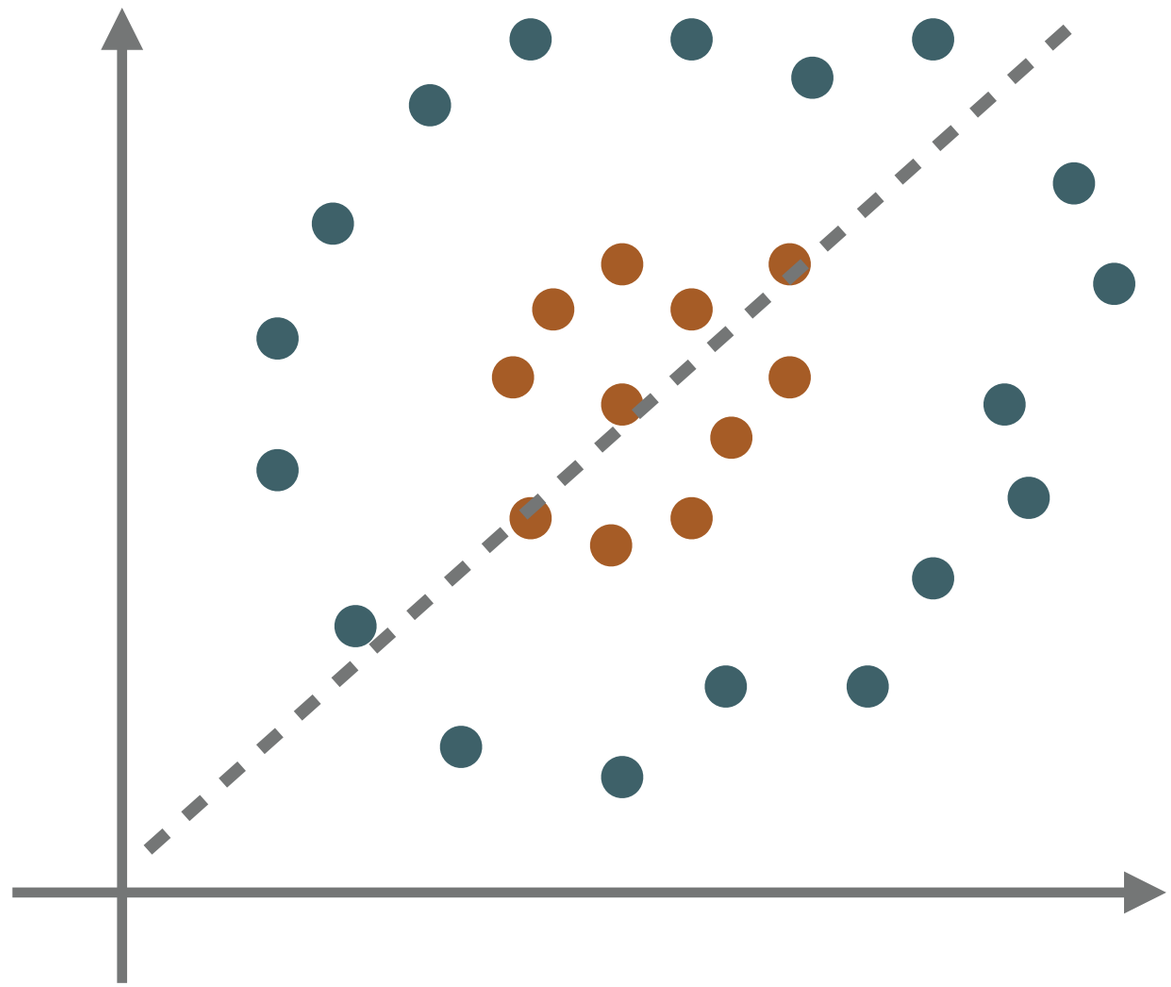




PROBLEMATIC CASES

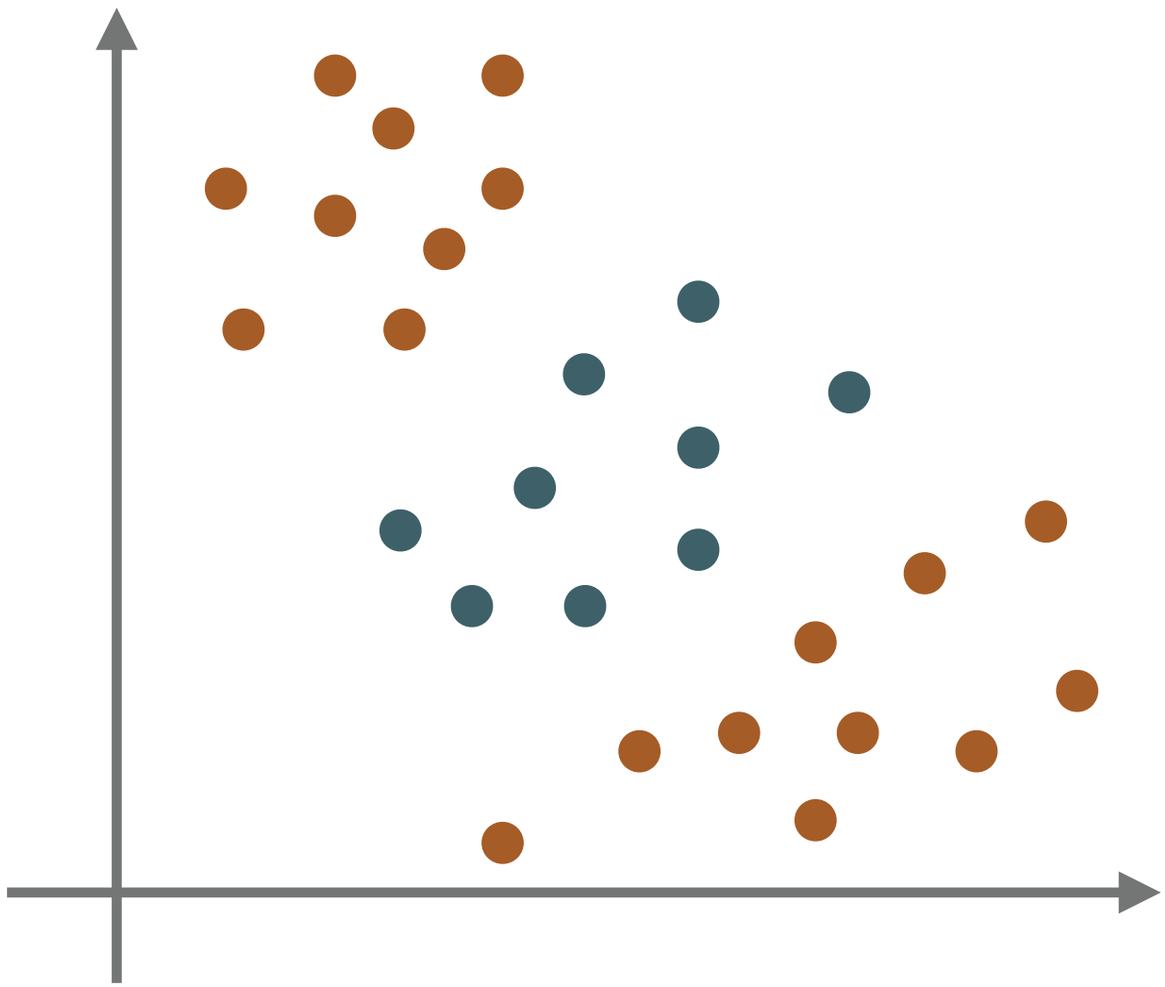
- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**

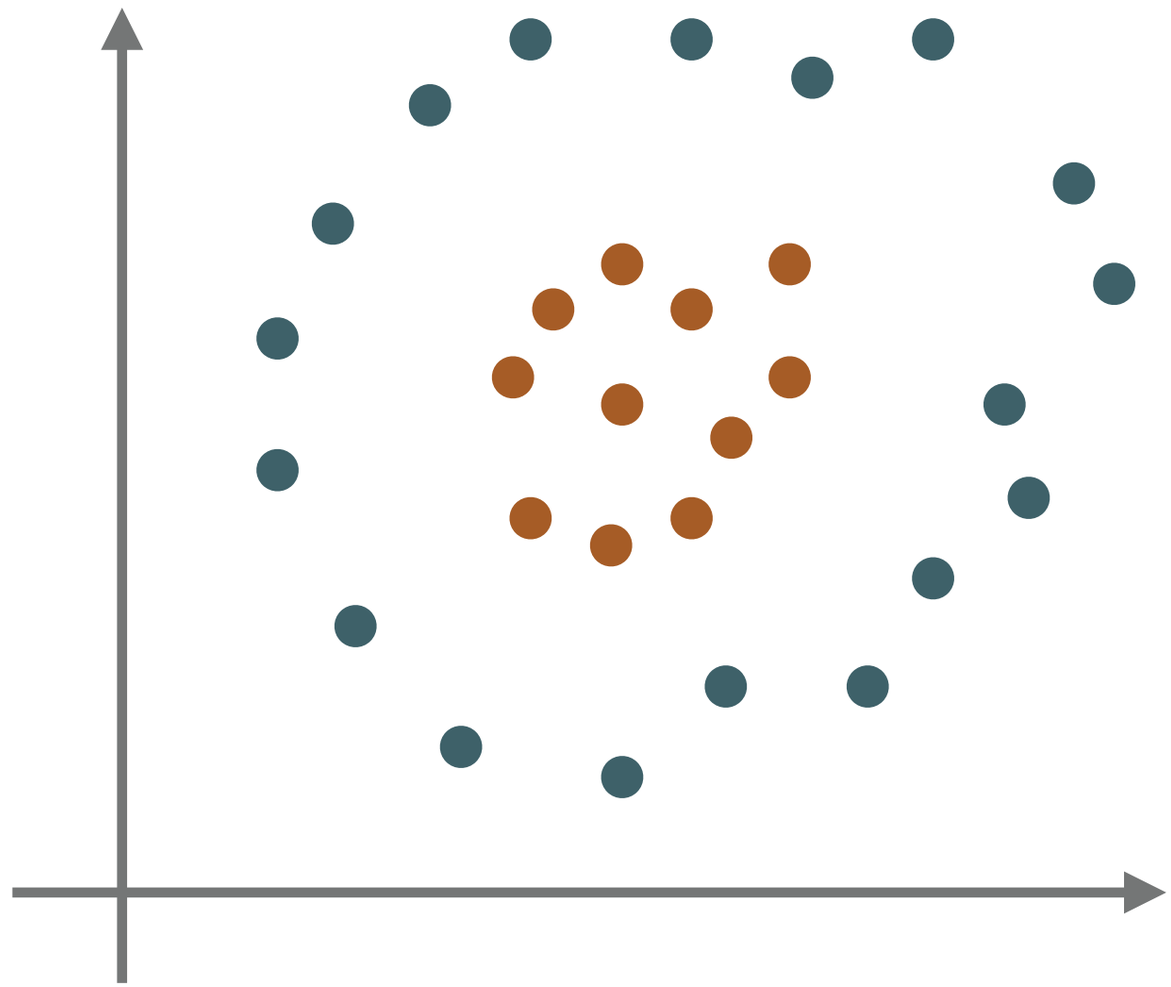




PROBLEMATIC CASES

- Can we always find a hyperplane that separate classes? **NO**
- Can we characterize formally in which cases we can? **YES**





A VERY SHORT INTRODUCTION TO DEEP LEARNING

MAIN IDEA

« Latent » hidden representation

- Compute a sequence of hidden representations so that classes are linear separable
- Train everything end-to-end via gradient descent!

Classifier

Parameterized function $f_\theta : \mathcal{X} \to \mathcal{Y}$

Feature space

Output space

Parameters

Neural Network

« Complicated » non convex parameterized function $f_\theta : \mathcal{X} \to \mathcal{Y}$

MULTILAYER PERCEPTRON 1/2

$$\mathbf{z}^{(1)} = \sigma\left(\mathbf{A}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right)$$

First hidden layer

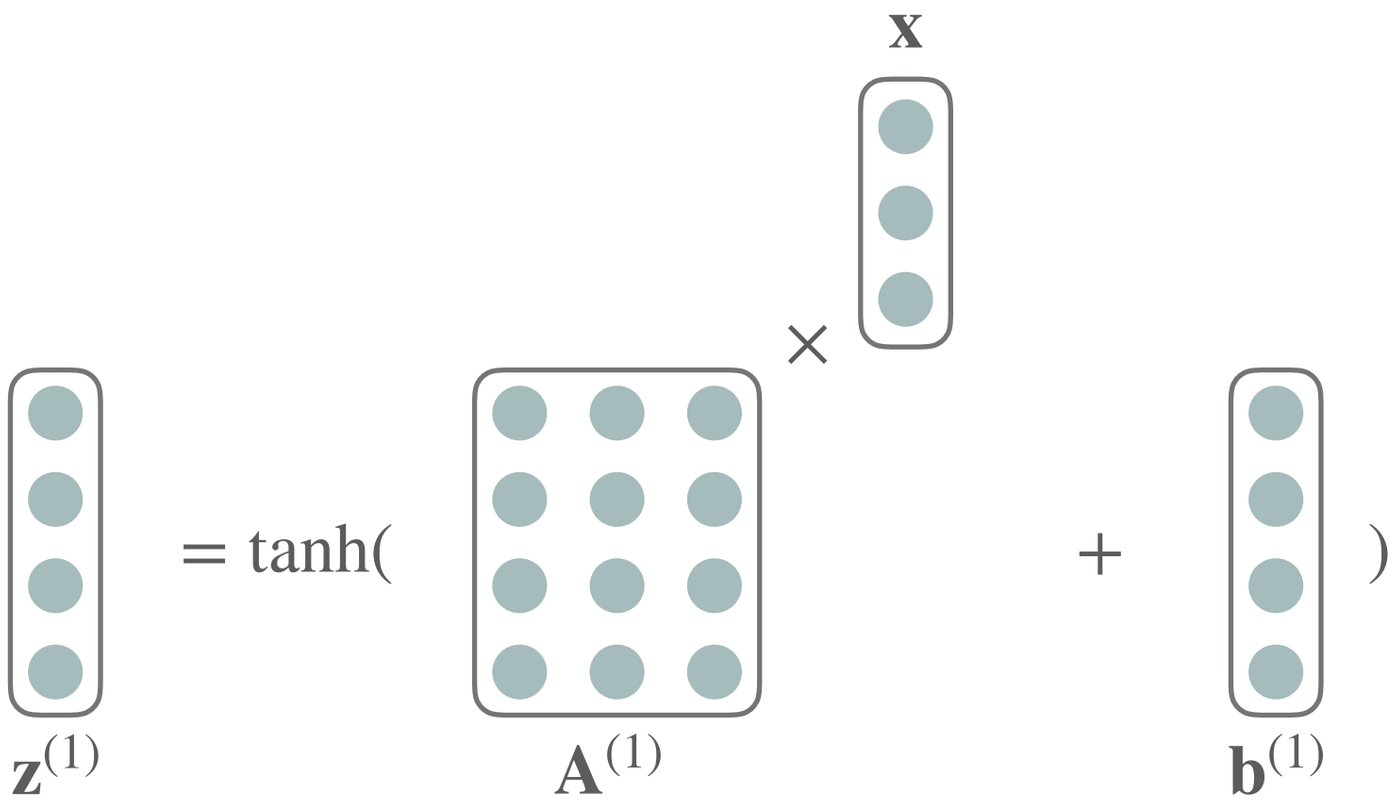
$$\mathbf{z}^{(2)} = \sigma\left(\mathbf{A}^{(2)}\mathbf{z}^{(1)} + \mathbf{b}^{(2)}\right)$$

Second hidden layer

$$\mathbf{w} = \mathbf{A}^{(3)}\mathbf{z}^{(2)} + \mathbf{b}^{(3)}$$

Output projection

- $\mathbf{x}$: input features
- $\mathbf{z}^{(i)}$: hidden representations
- $\mathbf{w}$: output logits
- $\theta = \{\mathbf{A}^{(1)}, \mathbf{b}^{(1)}, \dots\}$: trainable parameters
- σ : piecewise non-linear activation function



OPTIMIZATION

Backpropagation algorithm

1. Build computation graph
2. Back-propagate to compute the (sub-)gradient

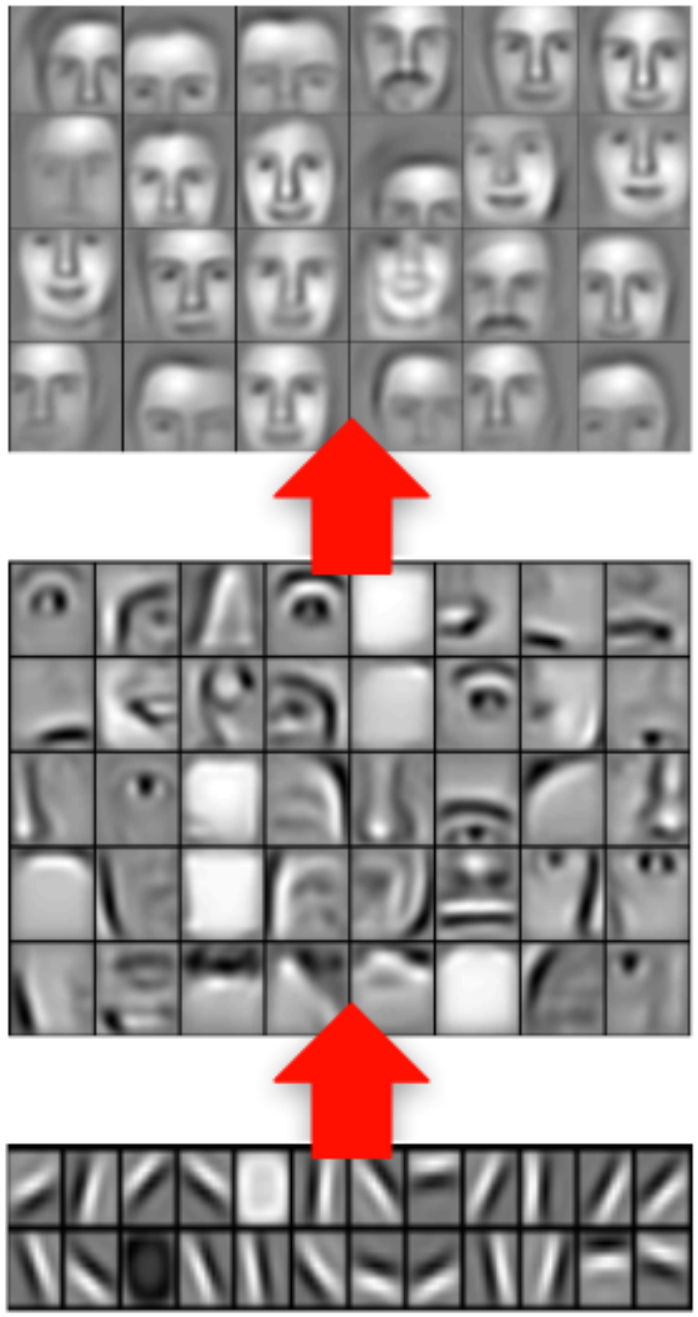
Important: NLP often requires to allow to build dynamic computation graphs!

Beyond gradient descent (GD)

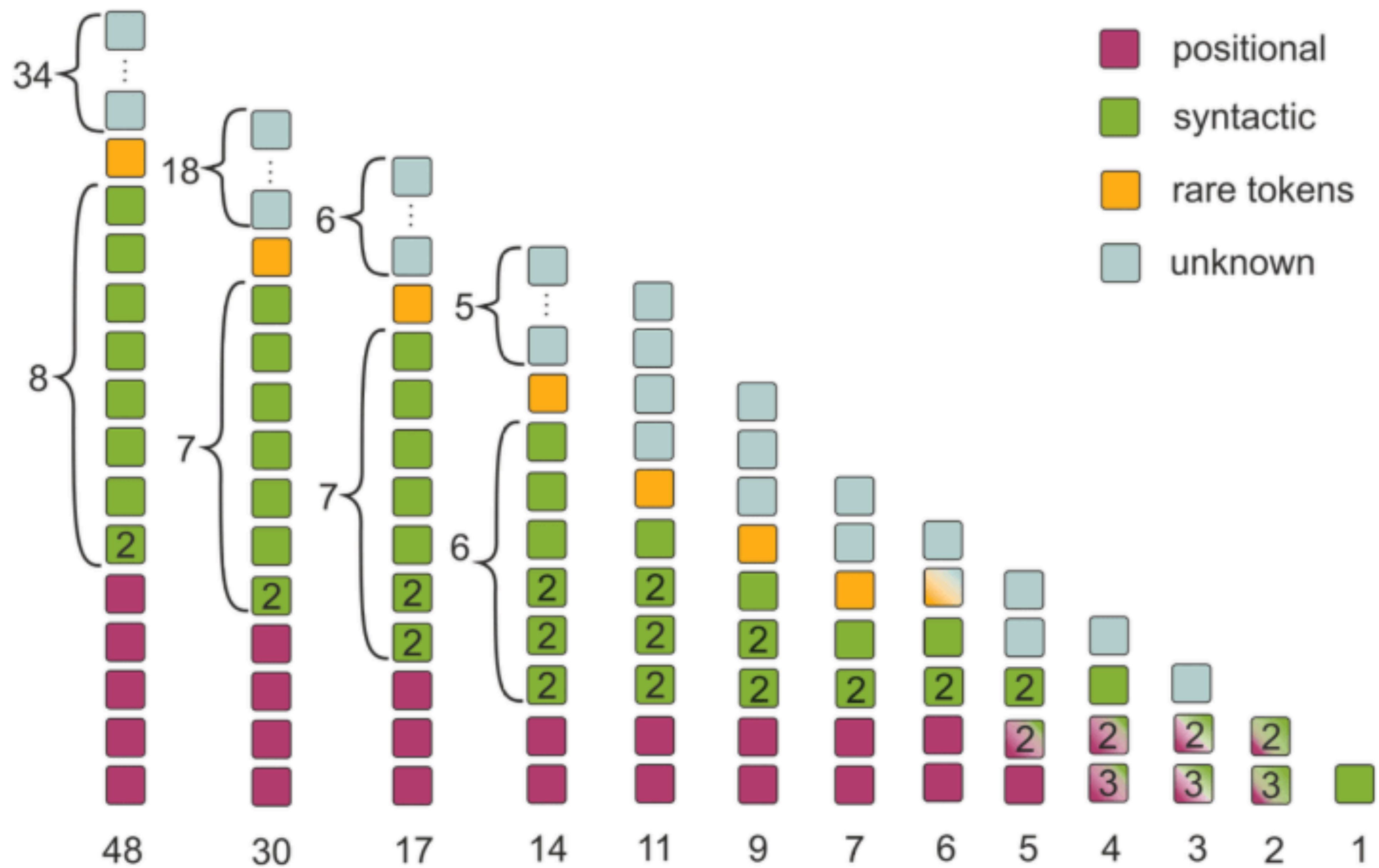
- Stochastic GD: approximate the objective using a random subset of datapoints (i.e. using a minibatch, or even a single datapoint)
- « improved » GD optimization algorithms:
  - Momentum
  - Adam
  - …

In NLP, use Adam as default :)

REPRESENTATION LEARNING: COMPUTER VISION [LEE ET AL., 2009]



REPRESENTATION LEARNING: NATURAL LANGUAGE PROCESSING [VOITA ET AL., 2019]



GENERALIZATION

Overparameterized neural networks

- Networks where the number of parameters exceed the training dataset size.
- Can learn by heart the dataset,
  i.e. **overfit the data** -> does not generalize well to unseen data
- Are easier to optimize in practice

Monitoring the training process

- Loss should go down
- Training accuracy should go up
- Dev accuracy should go up

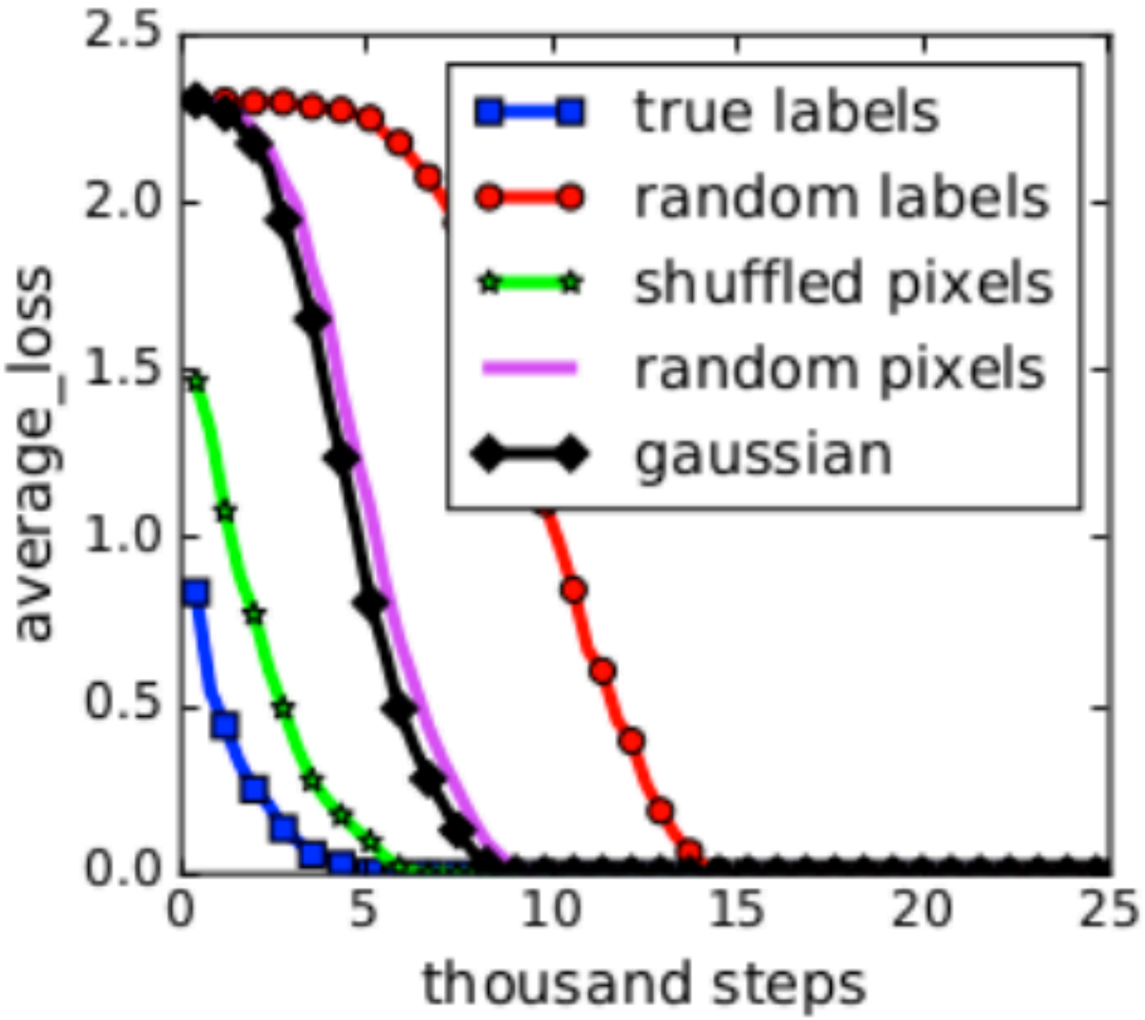
Regularization

Techniques to control parameters during learning and prevent overfitting

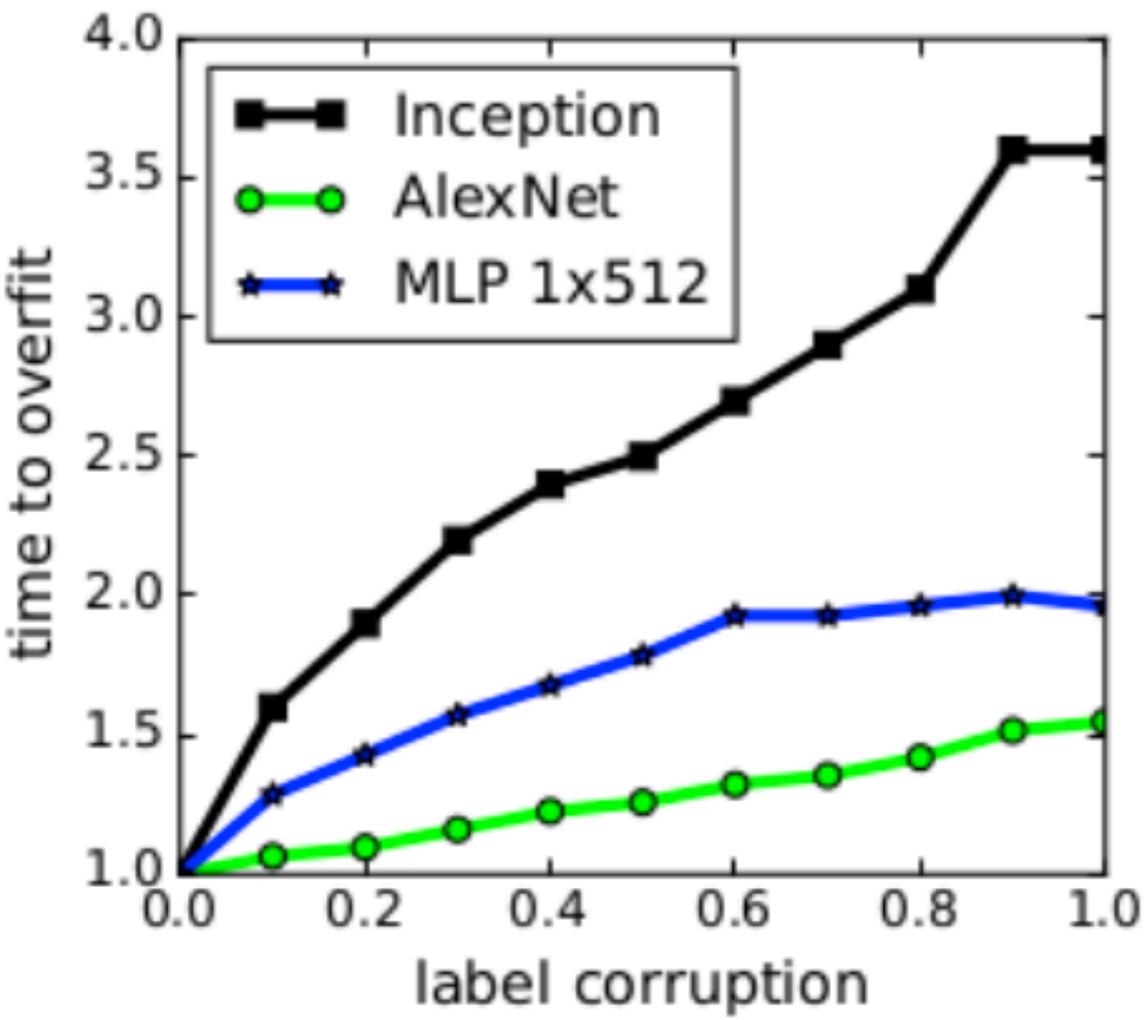
LEARNING WITH RANDOM INPUTS AND LABELS 1/2 [ZHANG ET AL., 2017]

model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

LEARNING WITH RANDOM INPUTS AND LABELS 2/2 [ZHANG ET AL., 2017]



(a) learning curves

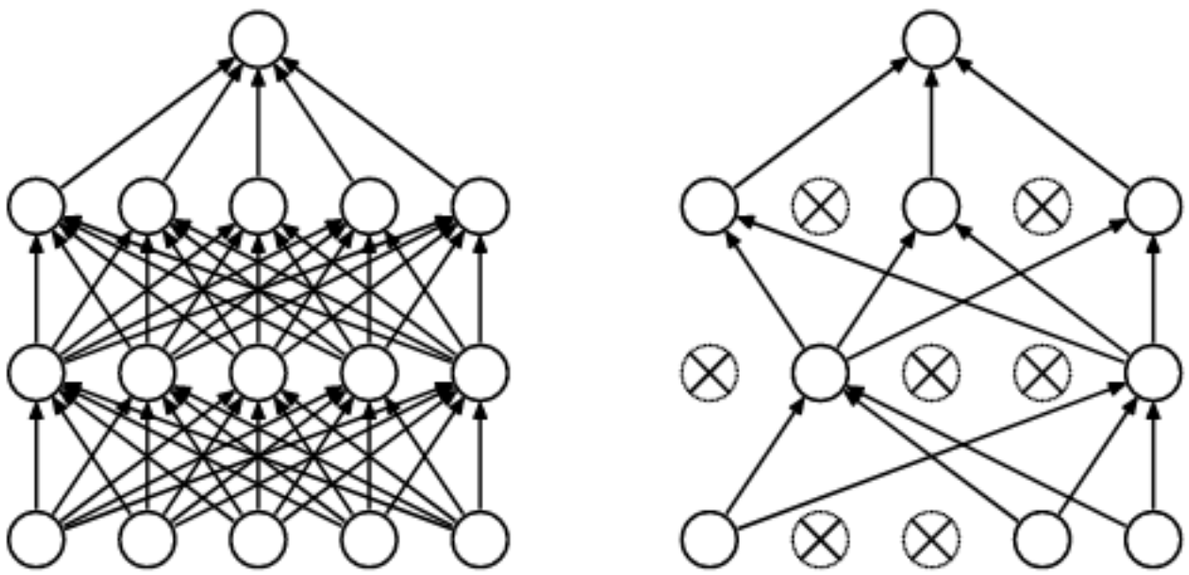


(b) convergence slowdown

DROPOUT 1/4 [HINTON ET AL., 2012, SRIVASTAVA ET AL., 2014]

How does dropout work?

- During training, we randomly "turn off" neurons,
  i.e. we randomly set elements of hidden layers to 0
- During test, we do use the full network



(a) Standard Neural Net

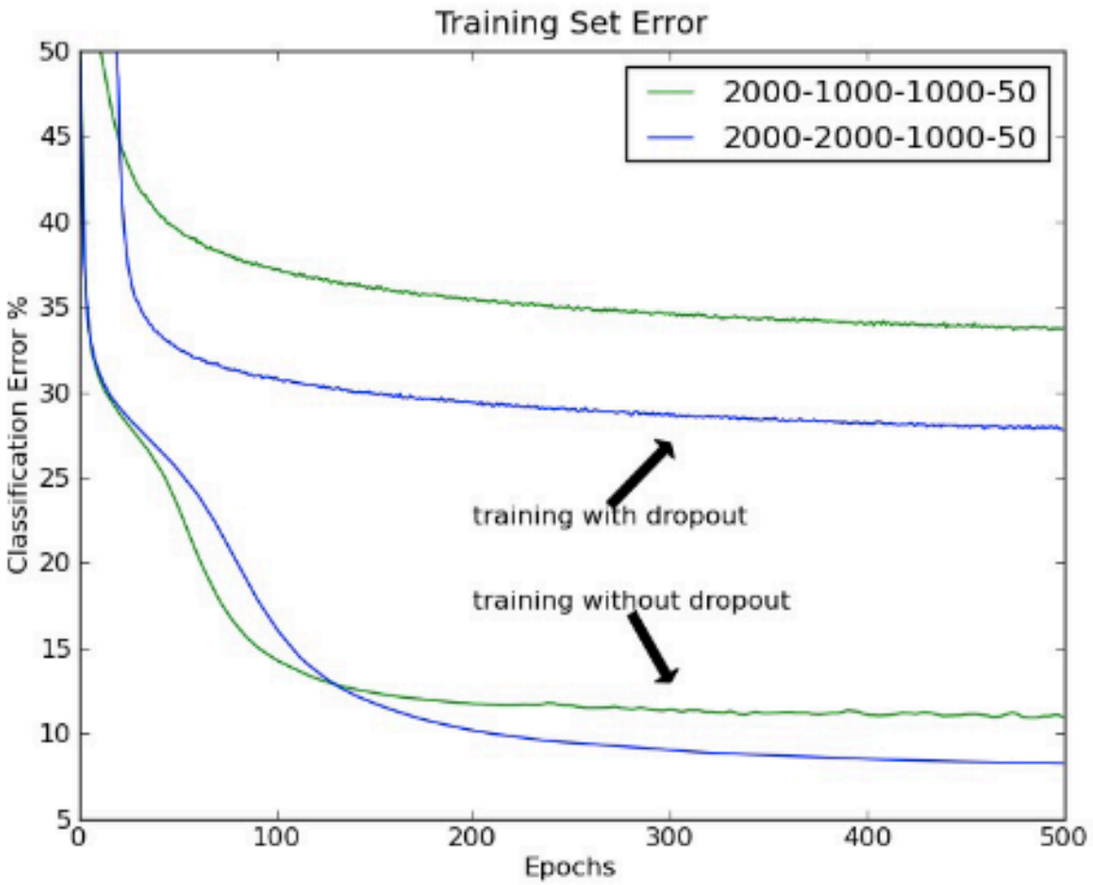
(b) After applying dropout.

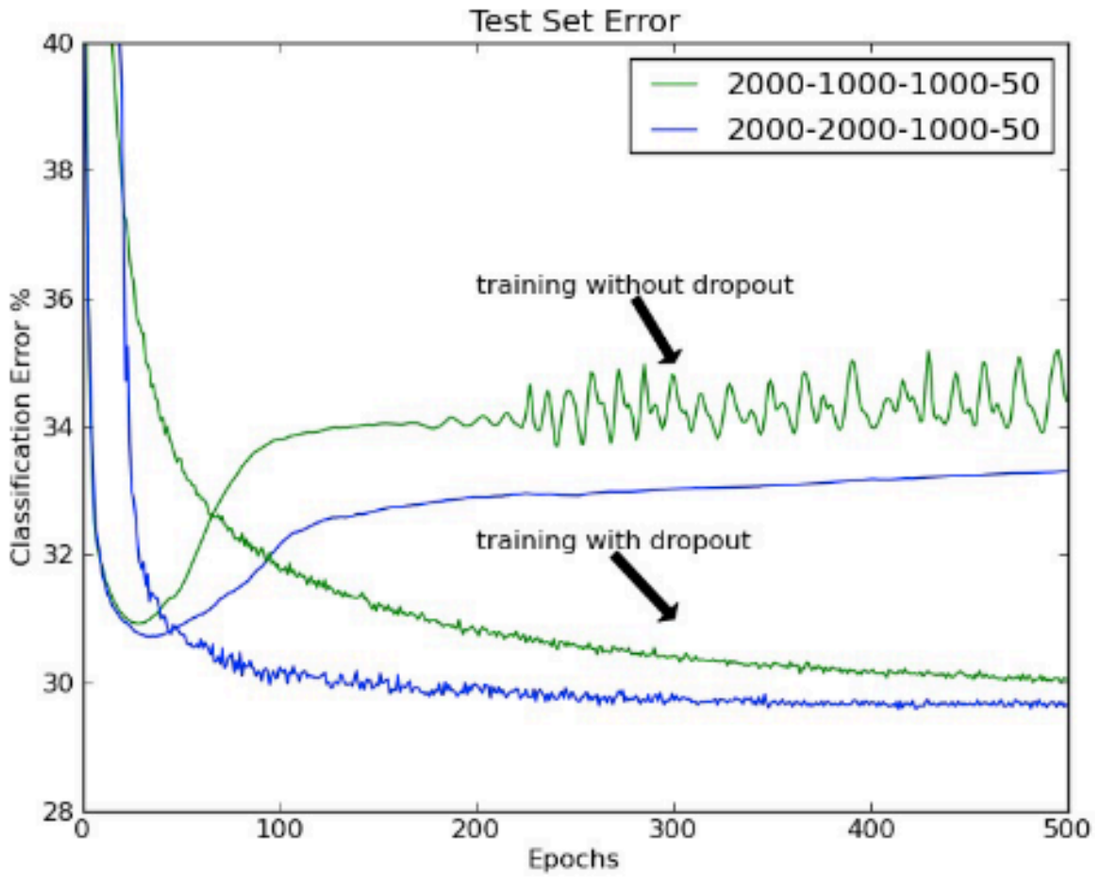
Intuition

- prevents co-adaptation between units
- equivalent to averaging different models

Units must be « independent »

DROPOUT 2/4 [HINTON ET AL., 2012]





DROPOUT 3/4 [HINTON ET AL., 2012]

Dropout layer

A dropout layer is parameterized by the probability of "turning off" a neuron

$$z' = \text{Dropout}(z, p = 0.5)$$

Implementation

$\boldsymbol{z} \in \mathbb{R}^n$: output of a hidden layer

$p \in [0, 1]$: dropout probability

$\boldsymbol{m} \in \{0, 1\}^n$: mask vector

$\boldsymbol{z}'$: hidden values after dropout application

Forward pass:

$$\boldsymbol{m} \sim \text{Bernoulli}(1 - p)$$

$$z'_i = \frac{z_i * m_i}{1 - p}$$

Backward pass:

$$\frac{\partial z'_i}{z_i} = \frac{m}{1 - p}$$

$\Rightarrow$ no gradient for "turned off" neurons.

The mask $\mathbf{m}$ a vector of booleans stating if neuron is z_i kept (m_i=1) or « turned off" (m_i=0).

DROPOUT 4/4 [HINTON ET AL., 2012]

Where do you apply dropout?

- On the input of the neural network
- After activation function
- Do not apply dropout on the output logits

Which dropout probability should you use?

- Empirical question: you have to test!
- Dropout probability at different layers can be different (especially input vs. hidden layers)
- Usually between 0.1 and 0.5

Dropout variants

Dropout can be applied differently for special neural network architectures (e.g. convolutions, recurrent neural networks)

LANGUAGE MODELING

LANGUAGE MODELING

The language modeling task

« Unsupervised » problem

Build a model that assigns a probability to any given sentence

$$p(y_1, y_2, \ldots, y_n)$$

LANGUAGE MODELING

The language modeling task

« Unsupervised » problem

Build a model that assigns a probability to any given sentence

$$p(y_1, y_2, \ldots, y_n)$$

Usefulness

- Text generation

$$y_1 \sim p(y_1) \qquad y_2 \sim p(y_2 \mid y_1) \qquad y_3 \sim p(y_3 \mid y_1, y_2) \qquad \ldots$$

LANGUAGE MODELING

The language modeling task

« Unsupervised » problem

Build a model that assigns a probability to any given sentence

$$p(y_1, y_2, \ldots, y_n)$$

Usefulness

- Text generation

$$y_1 \sim p(y_1) \qquad y_2 \sim p(y_2 \mid y_1) \qquad y_3 \sim p(y_3 \mid y_1, y_2) \qquad \ldots$$

- Test if a neural architecture is able to learn a good representation of language

$$p(\text{«}\textit{The dog is eating}\text{»}) > p(\text{«}\textit{Is eating dog the}\text{»})$$

LANGUAGE MODELING

The language modeling task

« Unsupervised » problem

Build a model that assigns a probability to any given sentence

$$p(y_1, y_2, \ldots, y_n)$$

Usefulness

- Text generation

$$y_1 \sim p(y_1) \qquad y_2 \sim p(y_2 \mid y_1) \qquad y_3 \sim p(y_3 \mid y_1, y_2) \qquad \ldots$$

- Test if a neural architecture is able to learn a good representation of language

$$p(\text{«The dog is eating»}) > p(\text{«Is eating dog the»})$$

- Improve sequence generation problems by using unlabeled data, e.g. machine translation, speech recognition

$$\hat{y}_1 \ldots \hat{y}_n = \max_{y_1 \ldots y_n} \quad \lambda \times p(y_1 \ldots y_n \mid x_1 \ldots x_m) \quad + \quad (1 - \lambda) \times p(y_1 \ldots y_n)$$

Translation accuracy

Target language adequacy

LANGUAGE MODEL EVALUATION

I love pizza with ____ { mushrooms, pineapples, C++, …

LANGUAGE MODEL EVALUATION

I love pizza with ____ { mushrooms, pineapples, C++, …

Difficult for large scale evaluation

Evaluation

- Predict next word: suitable words should have a higher probability than incoherent ones
- Evaluate on data: better language model should give a higher probability to the test set

LANGUAGE MODEL EVALUATION

I love pizza with ____ { mushrooms, pineapples, C++, …

Difficult for large scale evaluation

Evaluation

- Predict next word: suitable words should have a higher probability than incoherent ones
- Evaluate on data: better language model should give a higher probability to the test set

Perplexity

- Probability of test sentences normalized by the number of words
- The lower the better

$$PP(y_1, \ldots, y_n) = p(y_1, \ldots, y_n)^{-\frac{1}{n}}$$

$$= \sqrt[n]{\frac{1}{p(y_1, \ldots, y_n)}}$$

Intuition: longer sentences are less probable so we want to normalize to able to compare sentences of different sizes

N-GRAM MODEL 1/2

Auto-regressive model

The probability of a word is conditioned on previous words only,
or, in other words, we assume a generative model that produce words from left to right.

$$p(y_1 \dots y_n) = p(y_1, \dots, y_{n-1}) p(y_n \,|\, y_1, \dots, y_{n-1})$$

N-GRAM MODEL 1/2

Auto-regressive model

The probability of a word is conditioned on previous words only,
or, in other words, we assume a generative model that produce words from left to right.

$$p(y_1 \dots y_n) = p(y_1, \dots, y_{n-1}) p(y_n \,|\, y_1, \dots, y_{n-1})$$

Markov chain

Memory-less auto-regressive model:

Bi-gram model

- 1st order Markov chain: $p(y_1, \dots, y_n) = p(y_1) \prod_{i=2}^{n} p(y_i \,|\, y_{i-1})$

$$p(\text{«}\textit{The dog is eating}\text{»}) = p(\text{«}\textit{The}\text{»}) p(\text{«}\textit{dog}\text{»} \,|\, \text{«}\textit{The}\text{»}) p(\text{«}\textit{is}\text{»} \,|\, \text{«}\textit{dog}\text{»}) p(\text{«}\textit{eating}\text{»} \,|\, \text{«}\textit{is}\text{»})$$

N-GRAM MODEL 1/2

Auto-regressive model

The probability of a word is conditioned on previous words only,
or, in other words, we assume a generative model that produce words from left to right.

$$p(y_1 \dots y_n) = p(y_1, \dots, y_{n-1}) p(y_n \,|\, y_1, \dots, y_{n-1})$$

Markov chain

Memory-less auto-regressive model:

Bi-gram model

- 1st order Markov chain: $p(y_1, \dots, y_n) = p(y_1) \prod_{i=2}^{n} p(y_i \,|\, y_{i-1})$

$$p(\text{«The dog is eating»}) = p(\text{«The»}) p(\text{«dog»} \,|\, \text{«The»}) p(\text{«is»} \,|\, \text{«dog»}) p(\text{«eating»} \,|\, \text{«is»})$$

- 2nd order Markov chain: $p(y_1, \dots, y_n) = p(y_1) \; p(y_2 \,|\, y_1) \prod_{i=3}^{n} p(y_i \,|\, y_{i-1}, y_{i-2})$

Tri-gram model

N-GRAM MODEL 2/2

1st order Markov chain estimation

$$p(\text{« cat »} \mid \text{« the »}) = \frac{\textit{Number of times « the cat » appears in the training data}}{\textit{Number of times « the » appears in the training data}}$$

Smoothing

Worse for high order models!

The distribution is really sparse, many sequences of words have a probability of zero.

N-GRAM MODEL 2/2

1st order Markov chain estimation

$$p(\text{« cat »} \mid \text{« the »}) = \frac{\textit{Number of times « the cat » appears in the training data}}{\textit{Number of times « the » appears in the training data}}$$

Smoothing

Worse for high order models!

The distribution is really sparse, many sequences of words have a probability of zero.

- Introduce fake occurrences

$$p(\text{« cat »} \mid \text{« the »}) = \frac{1 + \textit{Number of times « the cat » appears in the training data}}{1 + \textit{Number of times « the » appears in the training data}}$$

N-GRAM MODEL 2/2

1st order Markov chain estimation

$$p(\text{« cat »} \mid \text{« the »}) = \frac{\textit{Number of times « the cat » appears in the training data}}{\textit{Number of times « the » appears in the training data}}$$

Smoothing

Worse for high order models!

The distribution is really sparse, many sequences of words have a probability of zero.

- Introduce fake occurrences

$$p(\text{« cat »} \mid \text{« the »}) = \frac{1 + \textit{Number of times « the cat » appears in the training data}}{1 + \textit{Number of times « the » appears in the training data}}$$

- Interpolation

$$\tilde{p}(\text{« cat »} \mid \text{« the big »}) = \lambda^{(1)} \times p(\text{« cat »} \mid \text{« the big »}) \quad \text{Tri-gram}$$
$$+\lambda^{(2)} \times p(\text{« cat »} \mid \text{« big »}) \quad \text{Bi-gram}$$
$$+\lambda^{(3)} \times p(\text{« cat »}) \quad \text{Uni-gram}$$
$$\textit{such that } \lambda^{(1)}, \lambda^{(2)}, \lambda^{(3)} \geq 0 \textit{ and } \lambda^{(1)} + \lambda^{(2)} + \lambda^{(3)} = 1$$

LIMITATIONS OF N-GRAMS MODELS

Pros

- Easy to estimate

Cons

- Unobserved sequences will have small probabilities

```
The big cat
The fat cat
```

« big » and « fat » are semantically similar, therefore we would like both examples to have approx. the same probability even if « fat cat » doest not appear in training data.

- Limited context, no long range dependencies

The **cats**, I have been told, **are** big.

N-GRAM MODEL WITH A MULTILINEAR PERCEPTRON

NEURAL N-GRAM MODEL

Task

Predict the next word give a fixed size history.

- Bi-gram model: $p(\text{« } cat \text{ » } | \text{ « } big \text{ »})$
- Tri-gram model: $p(\text{« } cat \text{ » } | \text{ « } The\ big \text{ »})$

NEURAL N-GRAM MODEL

Task

Predict the next word give a fixed size history.

- Bi-gram model: $p(\text{« cat »} \mid \text{« big »})$
- Tri-gram model: $p(\text{« cat »} \mid \text{« The big »})$

Output

~100.000 words is standard

It's a classification problem, similar to MNIST digit classification!

Each output class represent a word, i.e. we map word to integers in $V = \{0,1,...\}$

NEURAL N-GRAM MODEL

Task

Predict the next word give a fixed size history.

- Bi-gram model: $p(\text{« cat »} \mid \text{« big »})$
- Tri-gram model: $p(\text{« cat »} \mid \text{« The big »})$

Output

~100.000 words is standard

It's a classification problem, similar to MNIST digit classification!

Each output class represent a word, i.e. we map word to integers in $V = \{0,1,...\}$

Last hidden representation

$\mathbf{z} \in \mathbb{R}^m$

Output logits

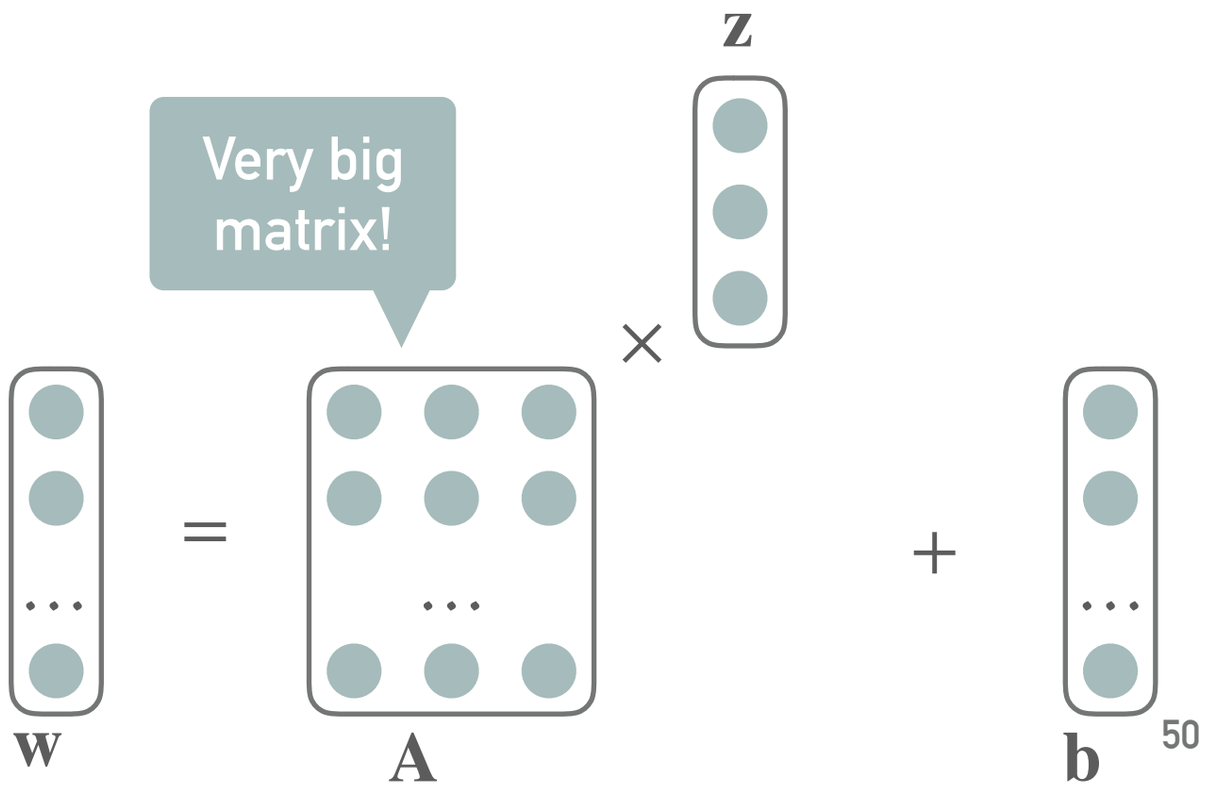
$\mathbf{w} \in \mathbb{R}^{|V|}$

Output projection

$\mathbf{W} \in \mathbb{R}^{|V| \times m}$

$\mathbf{b} \in \mathbb{R}^{|V| \times m}$

Very big matrix!



NEURAL N-GRAM MODEL

Input

- Bi-gram model: 1 word
- Tri-gram model: 2 words

One hot encoding of words

- Vocabulary mapped to a set of integers: $V = \{0,1,...\}$
- Input words are mapped to a one hot encoding vector

NEURAL N-GRAM MODEL

Input

- Bi-gram model: 1 word
- Tri-gram model: 2 words

One hot encoding of words

- Vocabulary mapped to a set of integers: $V = \{0,1,...\}$
- Input words are mapped to a one hot encoding vector

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ \dots \\ 0 \end{bmatrix}$$

One-hot encoding of the word mapped to 2

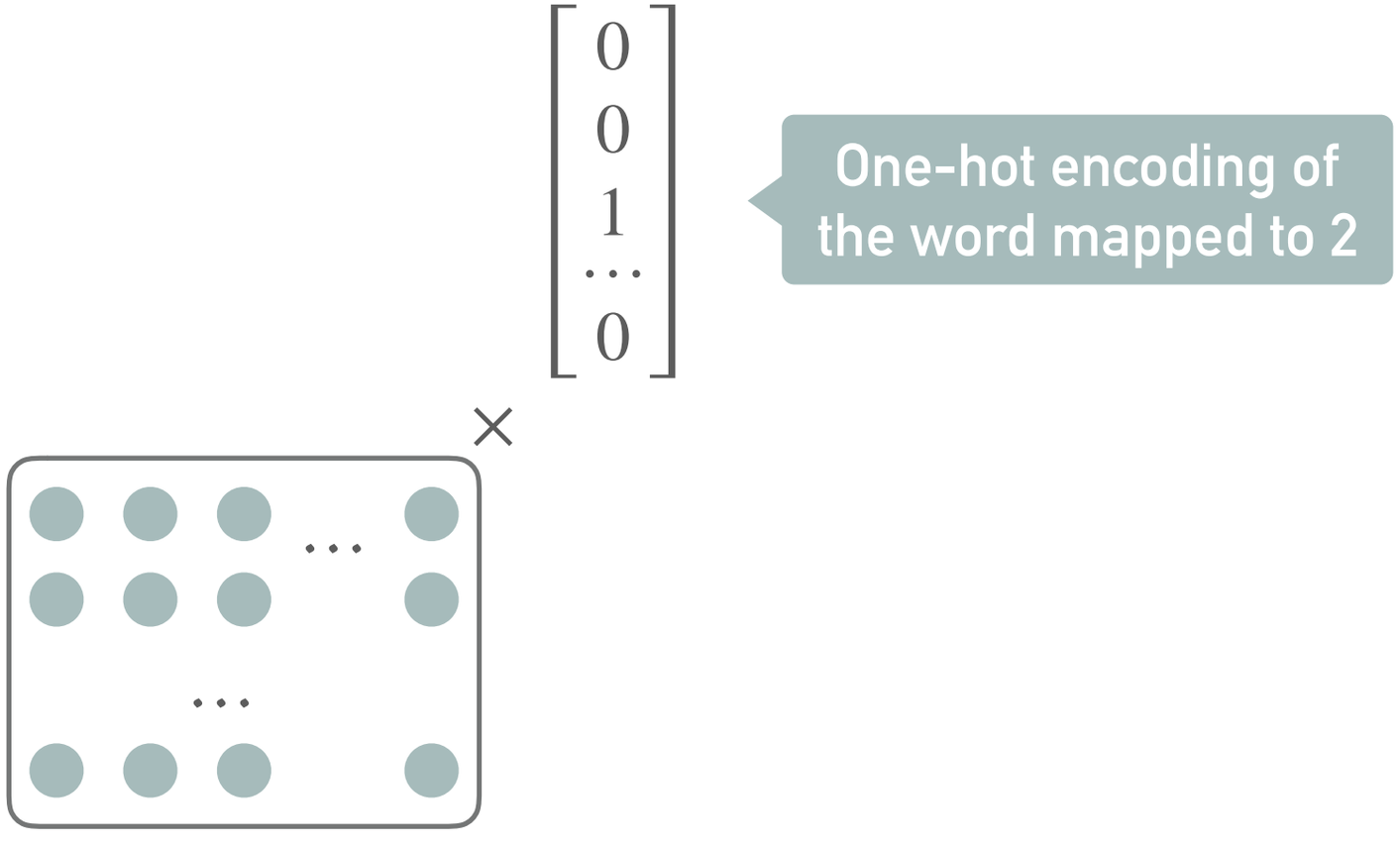
NEURAL N-GRAM MODEL

Input

- Bi-gram model: 1 word
- Tri-gram model: 2 words

One hot encoding of words

- Vocabulary mapped to a set of integers: $V = \{0,1,...\}$
- Input words are mapped to a one hot encoding vector



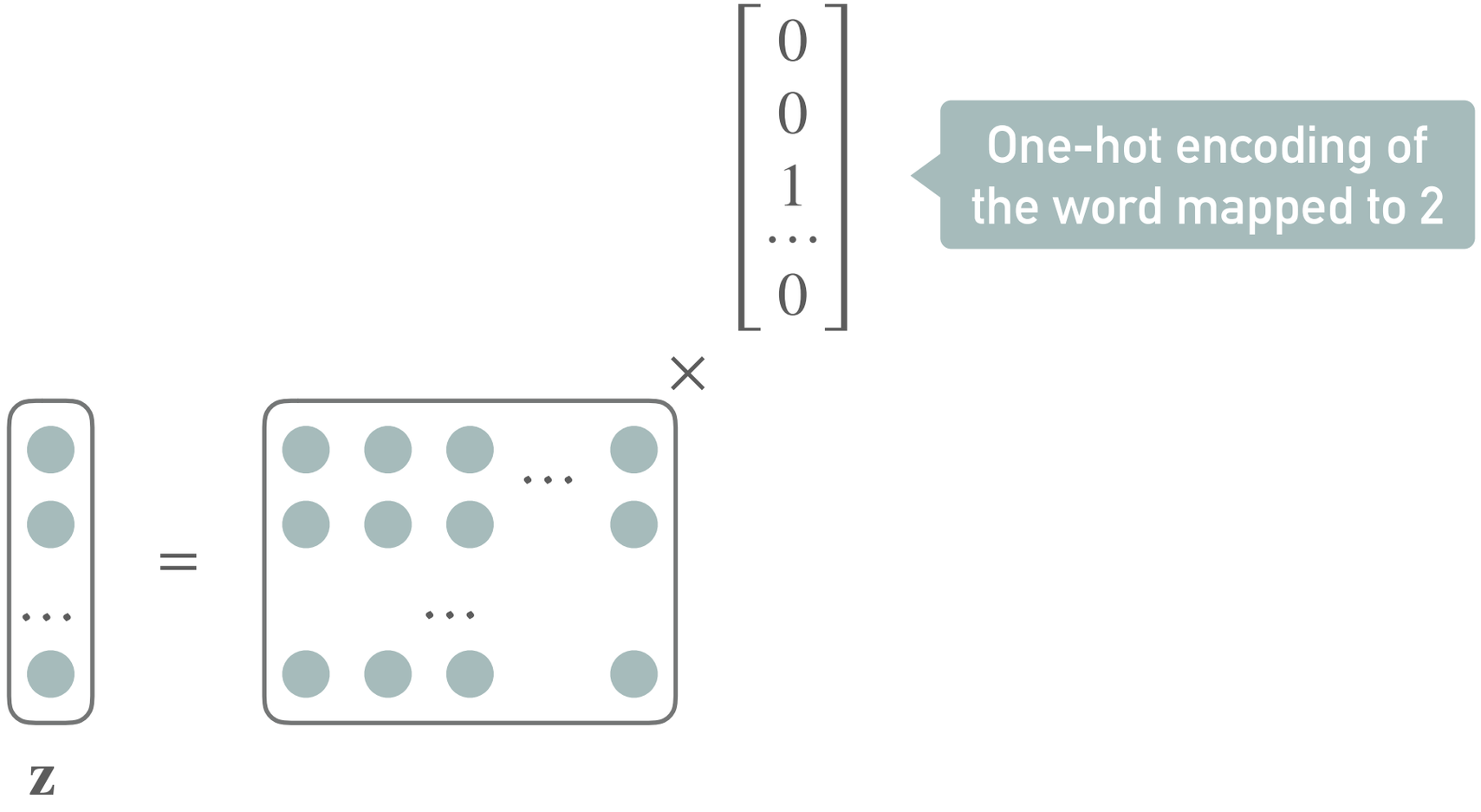
NEURAL N-GRAM MODEL

Input

- Bi-gram model: 1 word
- Tri-gram model: 2 words

One hot encoding of words

- Vocabulary mapped to a set of integers: $V = \{0,1,...\}$
- Input words are mapped to a one hot encoding vector



LIMITATIONS OF NEURAL N-GRAM MODELS

Pros

- Generalize to unseen n-grams

Cons

- More expensive to estimate than the non-neural model
- The output layer is expensive to compute!
- Limited context, no long range dependencies

Recurrent networks solve this problem: next week!

WORD EMBEDDINGS

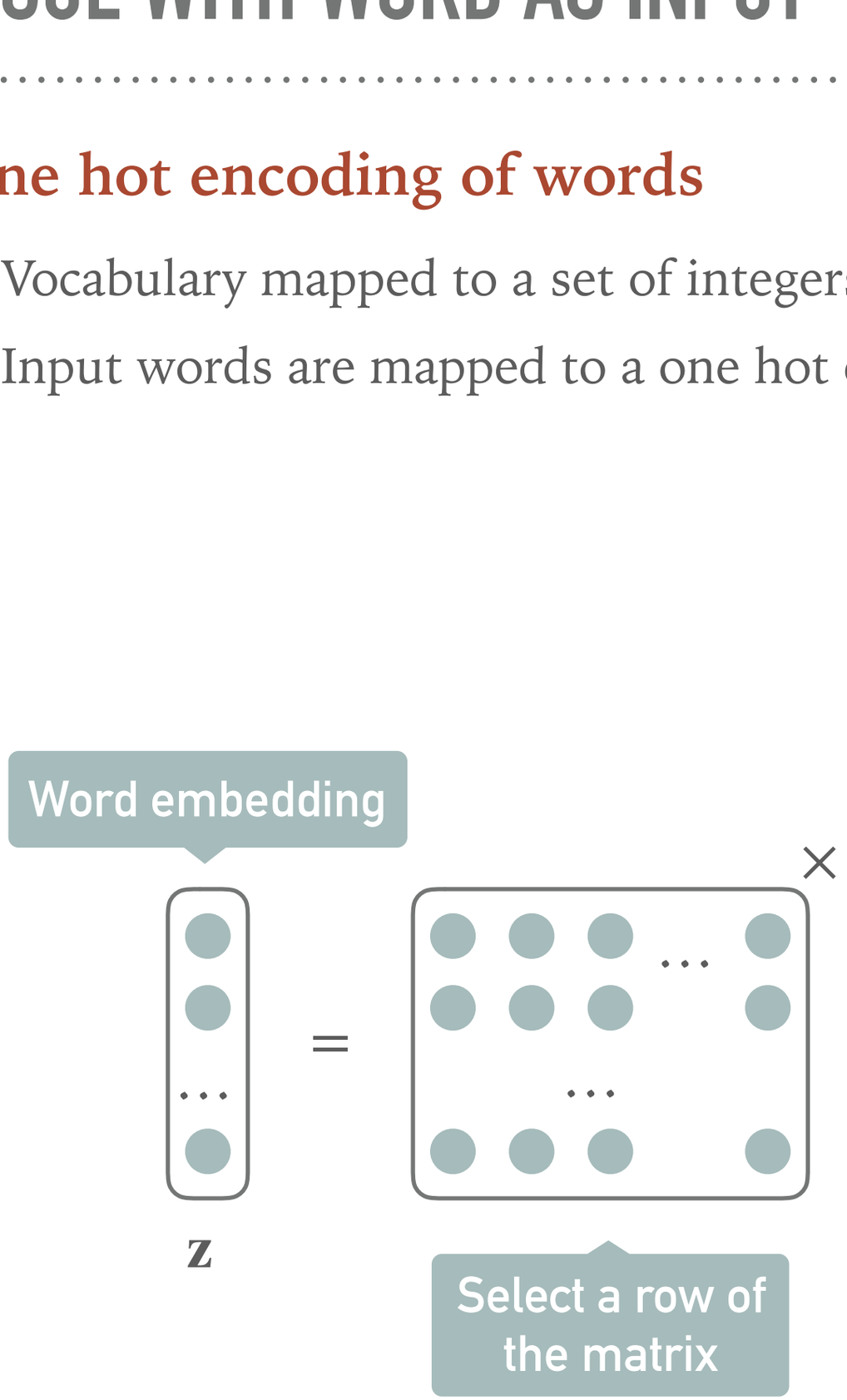
ISSUE WITH WORD AS INPUT

One hot encoding of words

- Vocabulary mapped to a set of integers: $V = \{0,1,...\}$
- Input words are mapped to a one hot encoding vector

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ \cdots \\ 0 \end{bmatrix}$$

One-hot encoding of the word mapped to 2



Explicitly computing this matrix multiplication is expensive: use embedding tables instead!

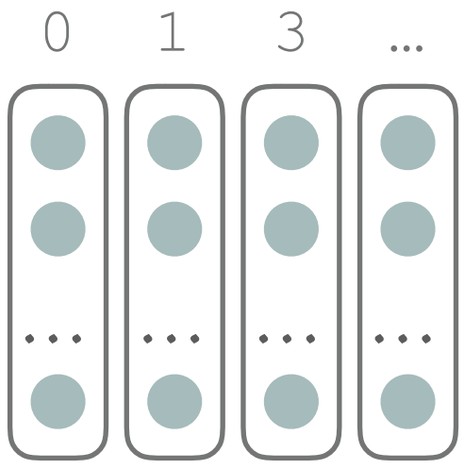
WORD EMBEDDING

Dictionary

- Dictionary: map each word to an integer
- Embedding table: map an integers to a vector

These vectors are parameters that are trained with the rest of the network!

```
car: 0
red: 1
big: 2
…
```



Unknown word

At test time, you may see words that where not present in the training set :(

1. Add a special « unknown word » to the dictionary/embedding table
2. Learn it by randomly replacing other words with the « unknown word » during training

PYTORCH IMPLEMENTATION

```
emb_table = torch.nn.Embedding(
    num_embeddings = 10,
    embedding_dim = 100
)
```

Construct embedding table

```
words = torch.LongTensor([2, 5])
# x.shape = (2, 100)
x = emb_table(words)
```

Retrieve embeddings of the input

```
# x.shape = (1, 200)
x = x.reshape(1, -1)
```

Reshape so that it looks like a mini batch of size one with 2 input words

CONVOLUTIONAL NEURAL NETWORK

SEQUENCE REPRESENTATION

Motivation

- Build token representation for out-of-vocabulary words (e.g. for tagging)
- Build sentence-level representation for sentence classification

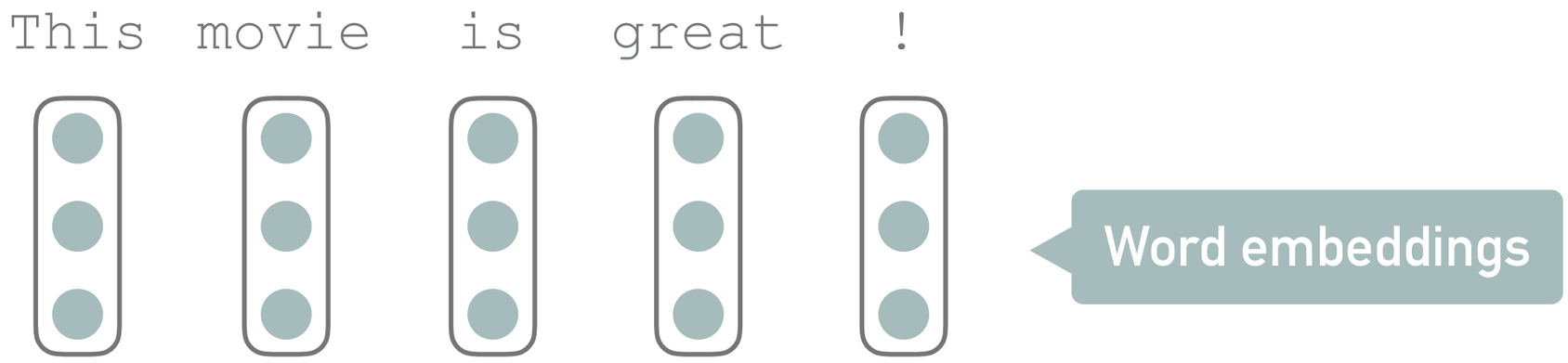
```
This movie is great!
I saw that movie some years ago.
The food was disgusting…
```

To classify these sentences, we need to build a fixed size hidden representation to feed it to a MLP!

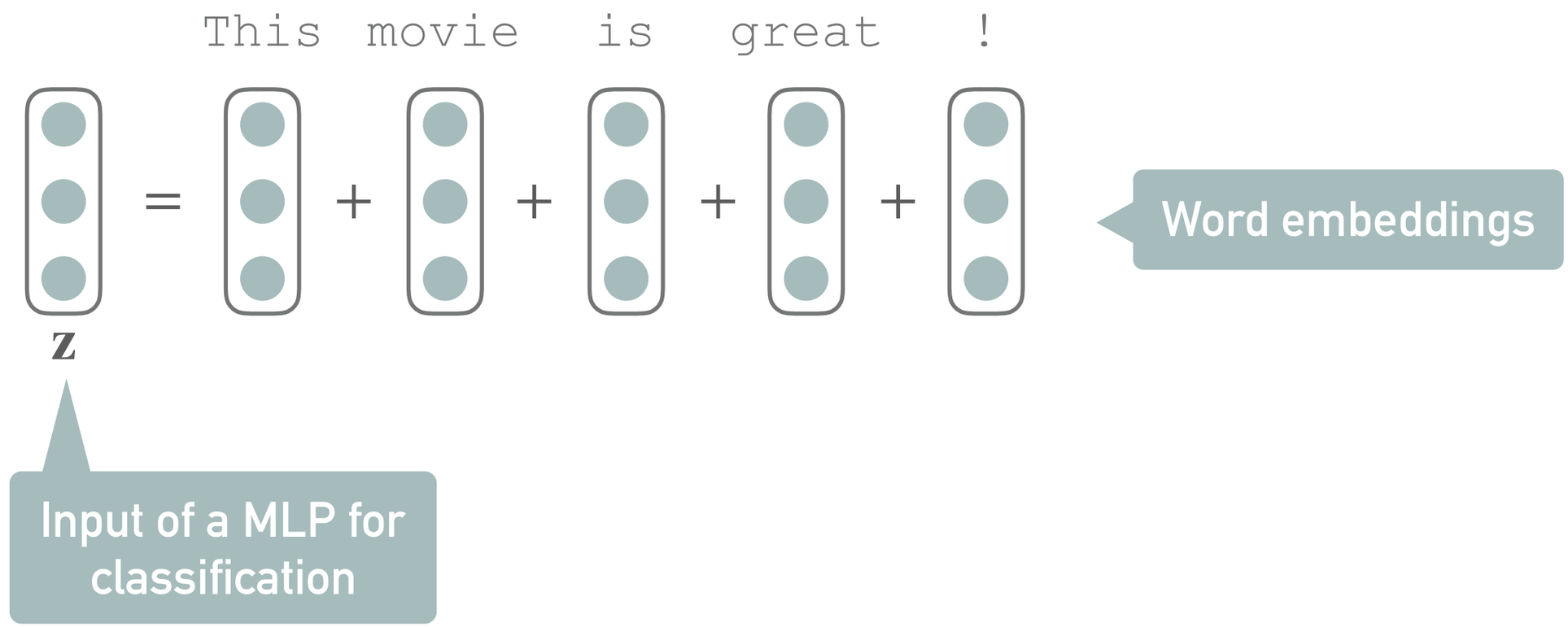
BAG-OF-WORDS MODEL

This movie is great !

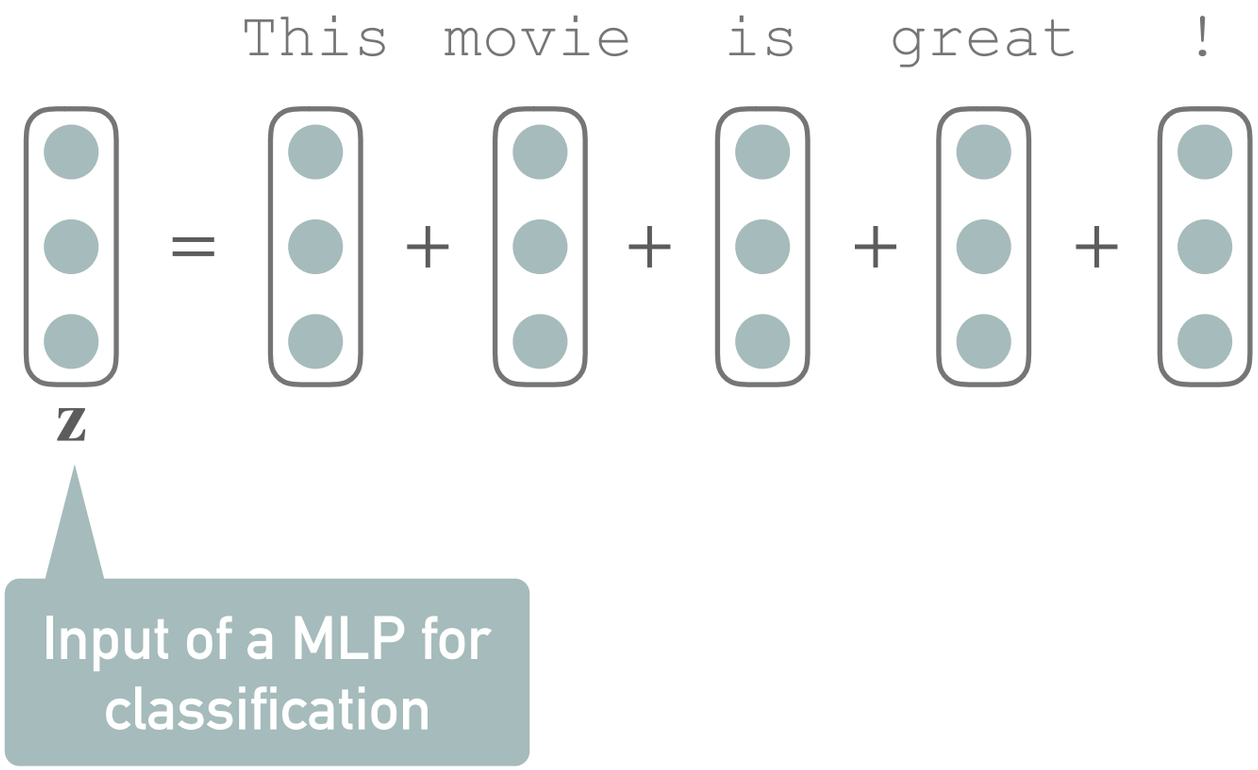
BAG-OF-WORDS MODEL



BAG-OF-WORDS MODEL



BAG-OF-WORDS MODEL



Word embeddings

This movie is great, acting is **not** bad.

This movie is **not** great, acting is bad.

These 2 sentences will have the same representation!

SEQUENCE REPRESENTATION

Motivation

- Build token representation for out-of-vocabulary words (e.g. for tagging)
- Build sentence-level representation for sentence classification

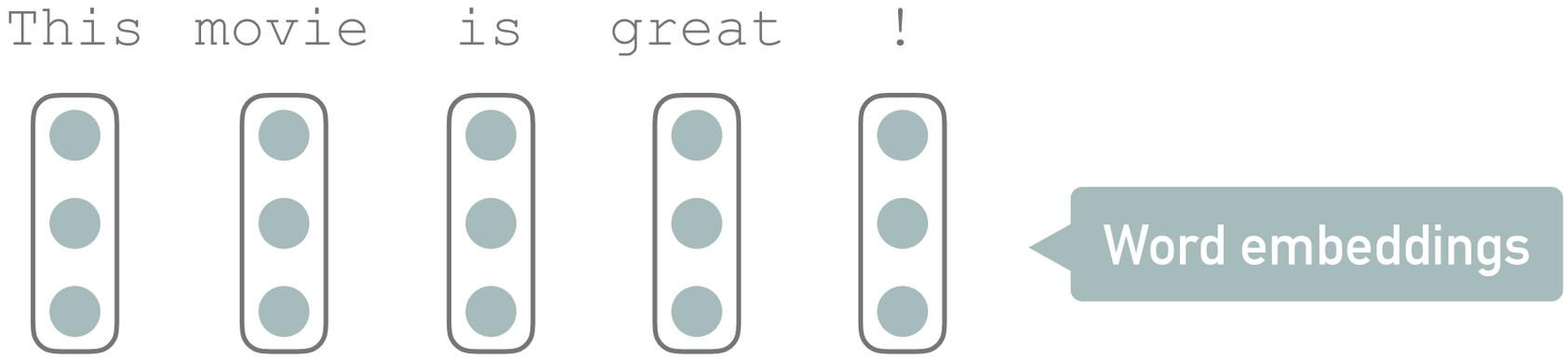
```
This movie is great!
I saw that movie some years ago.
The food was disgusting…
```

To classify these sentences, we need to build a fixed size hidden representation to feed it to a MLP!

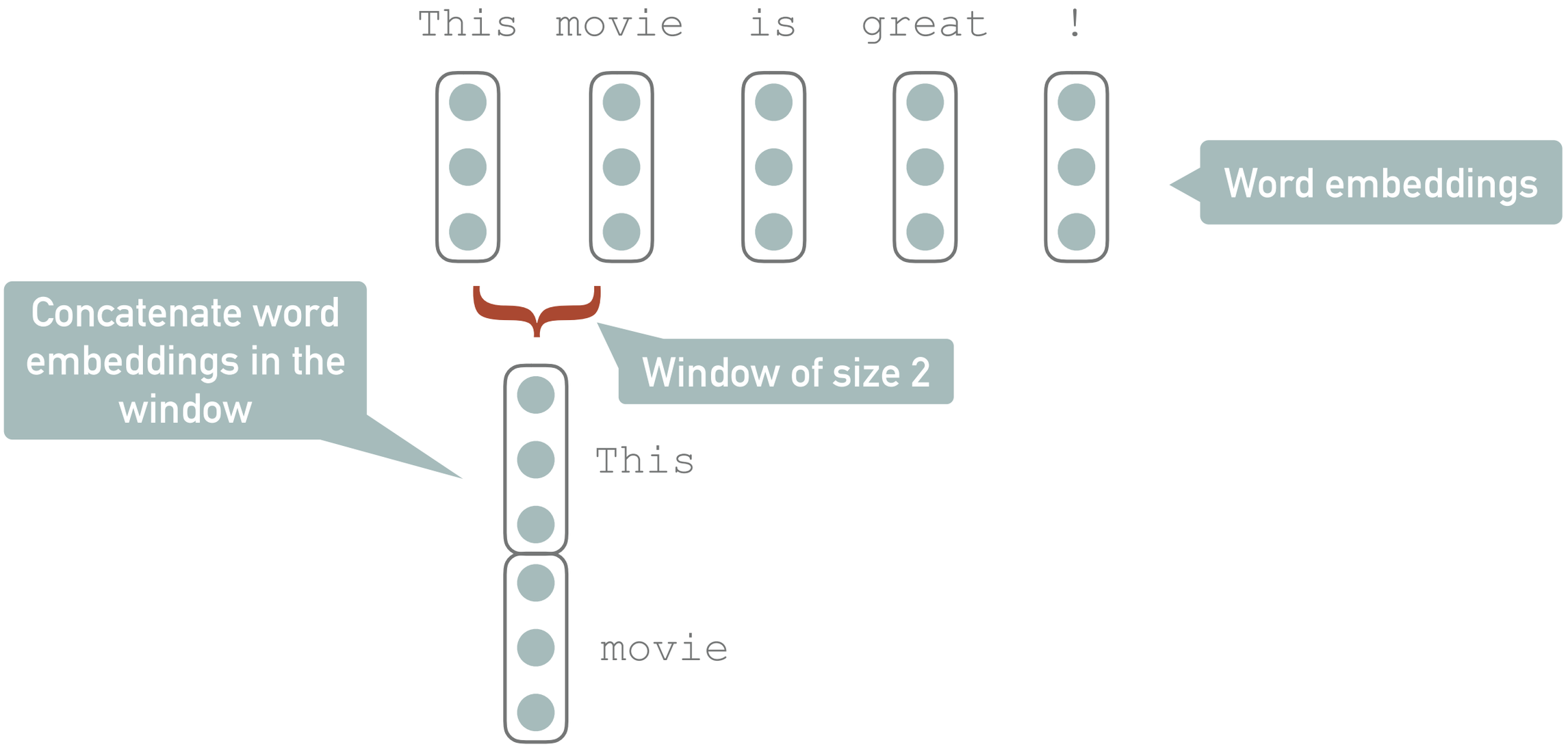
Convolutional Neural Network

1. Convolution over words with a fixed size sliding window
2. Pooling operation

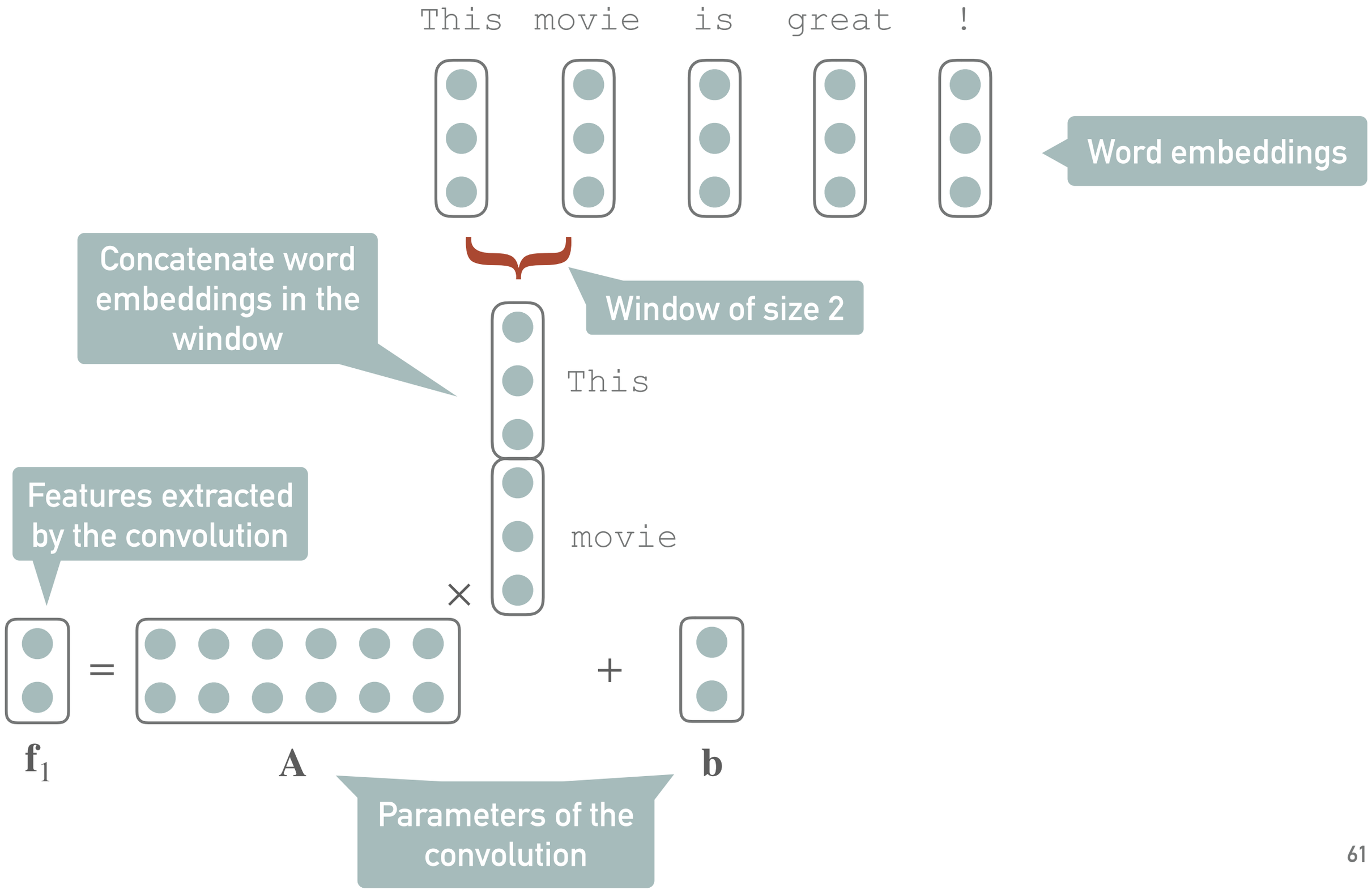
SLIDING WINDOW OF CONVOLUTIONS



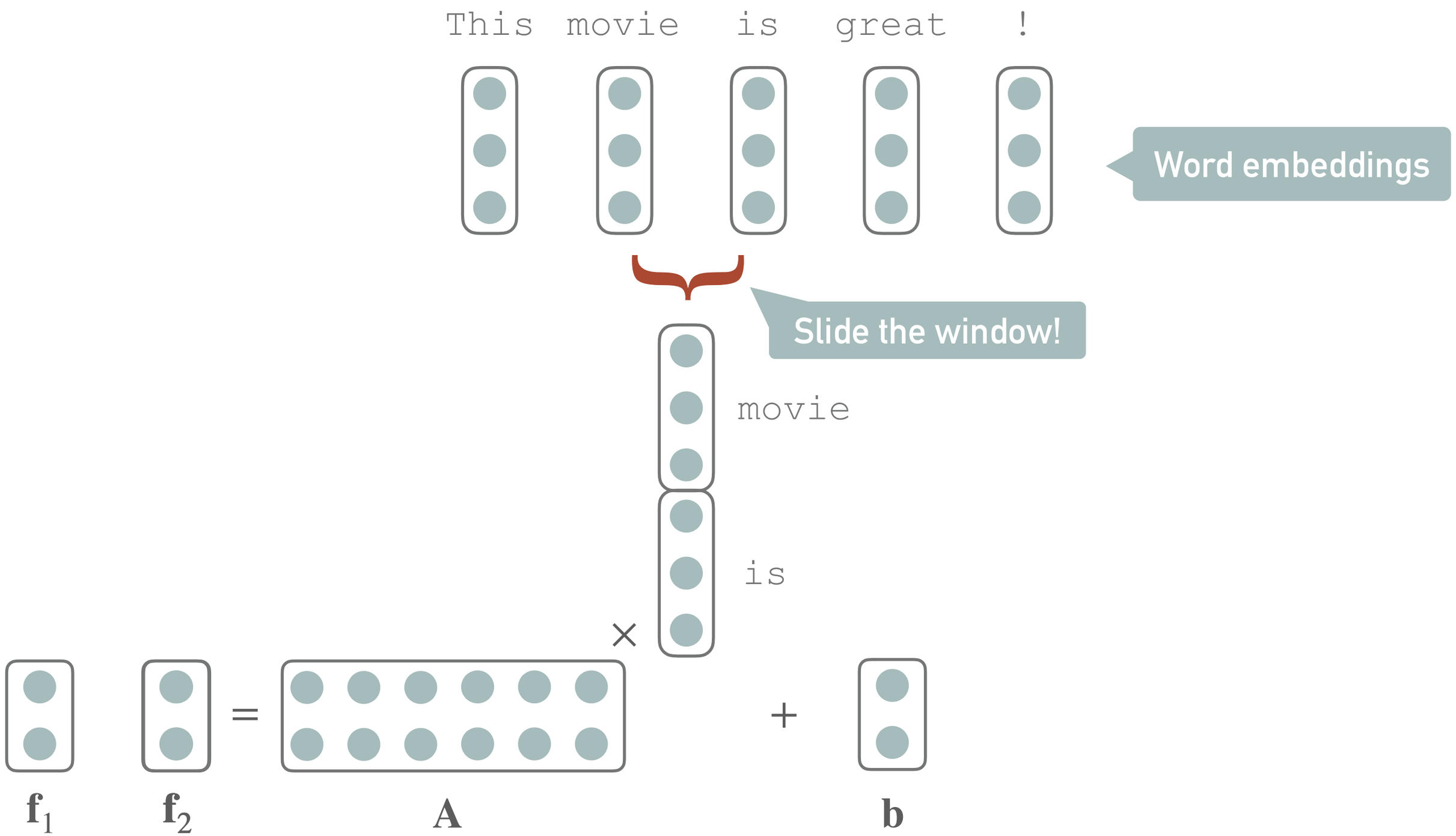
SLIDING WINDOW OF CONVOLUTIONS



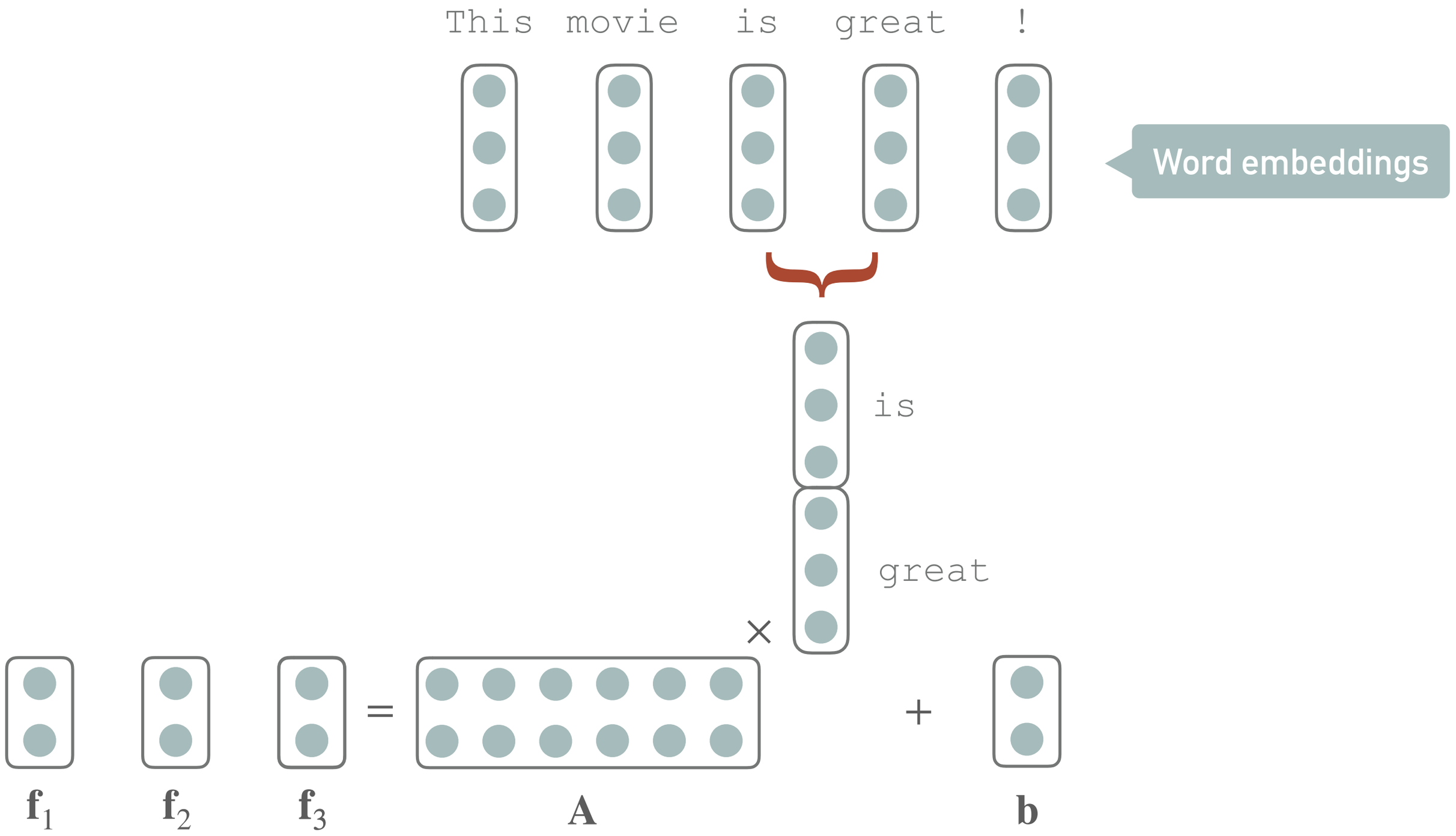
SLIDING WINDOW OF CONVOLUTIONS



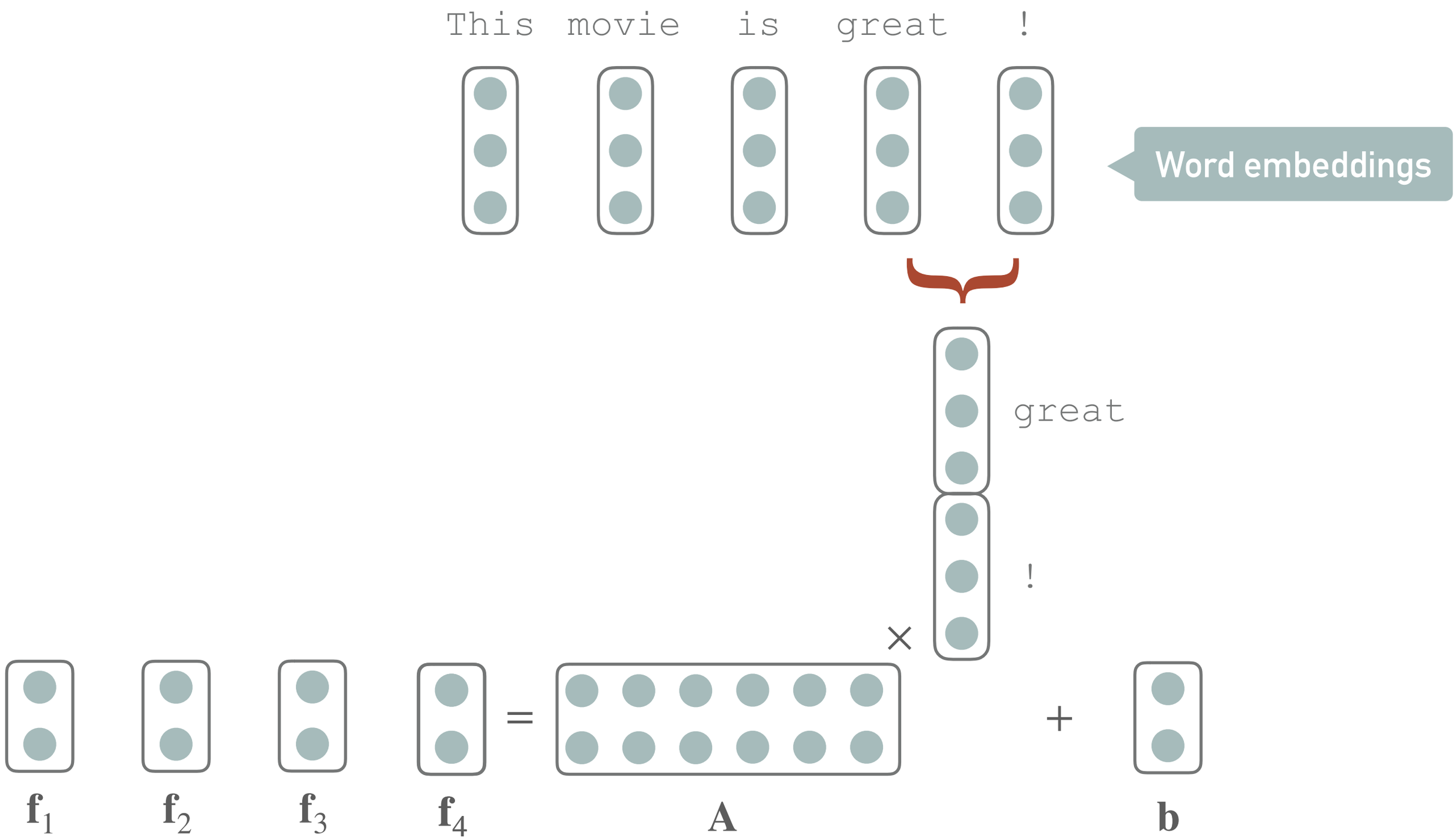
SLIDING WINDOW OF CONVOLUTIONS



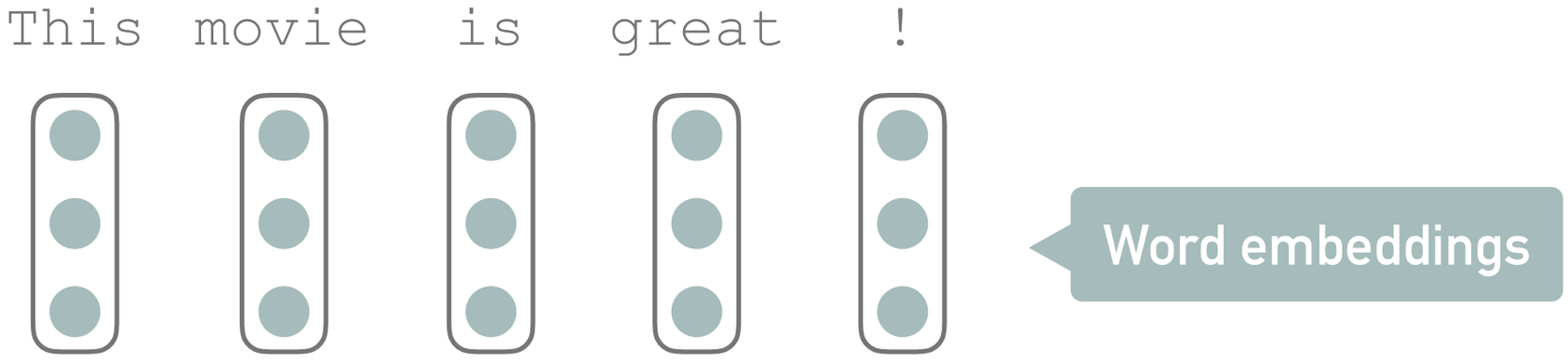
SLIDING WINDOW OF CONVOLUTIONS

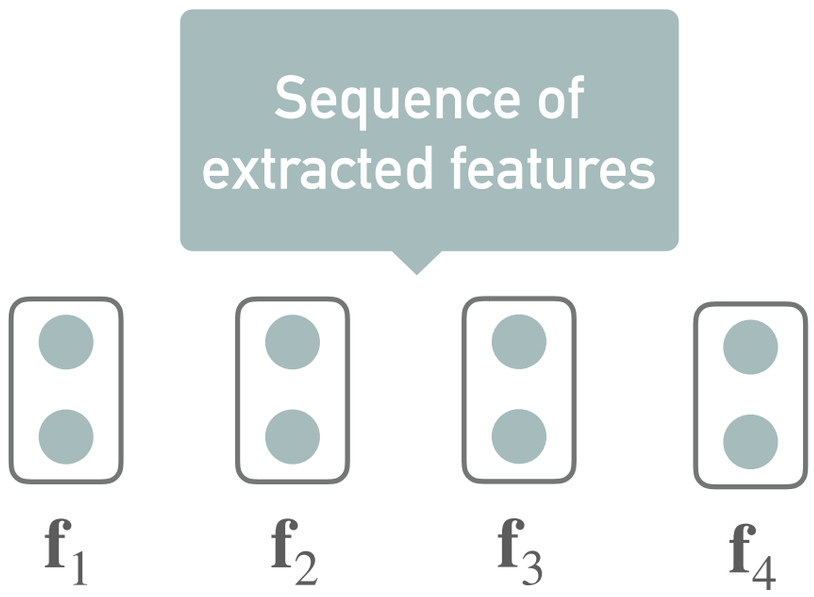


SLIDING WINDOW OF CONVOLUTIONS

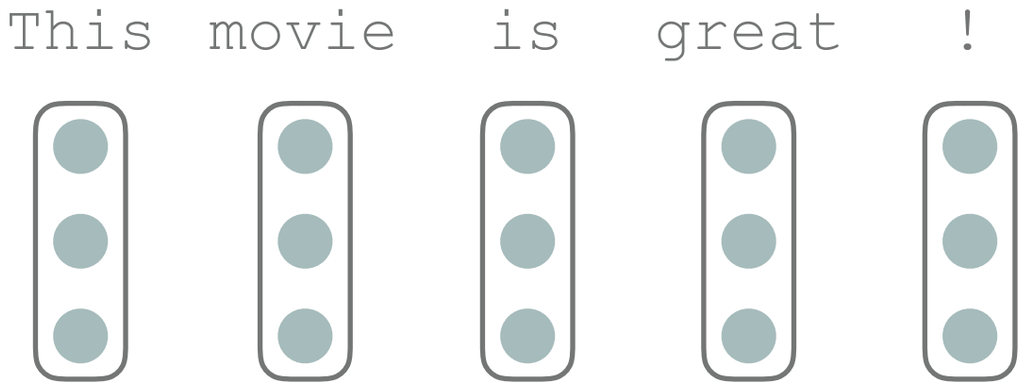


SLIDING WINDOW OF CONVOLUTIONS

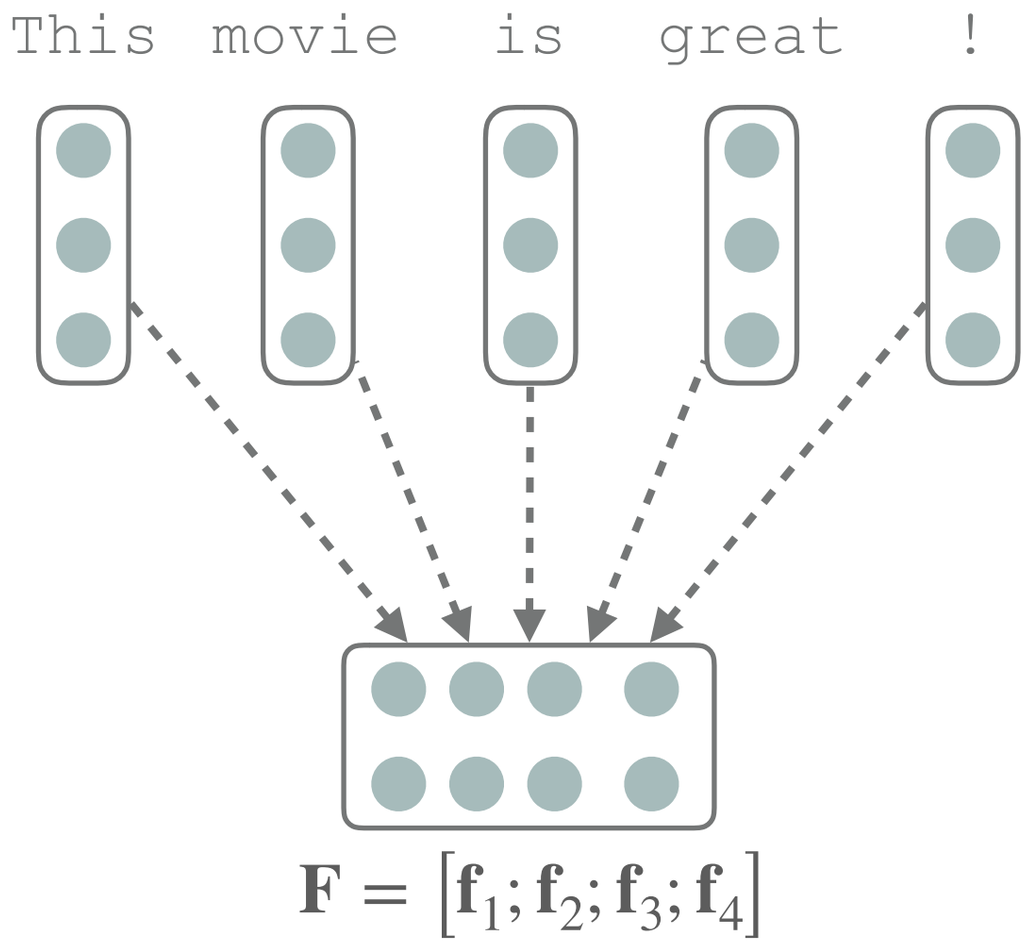




POOLING OPERATION



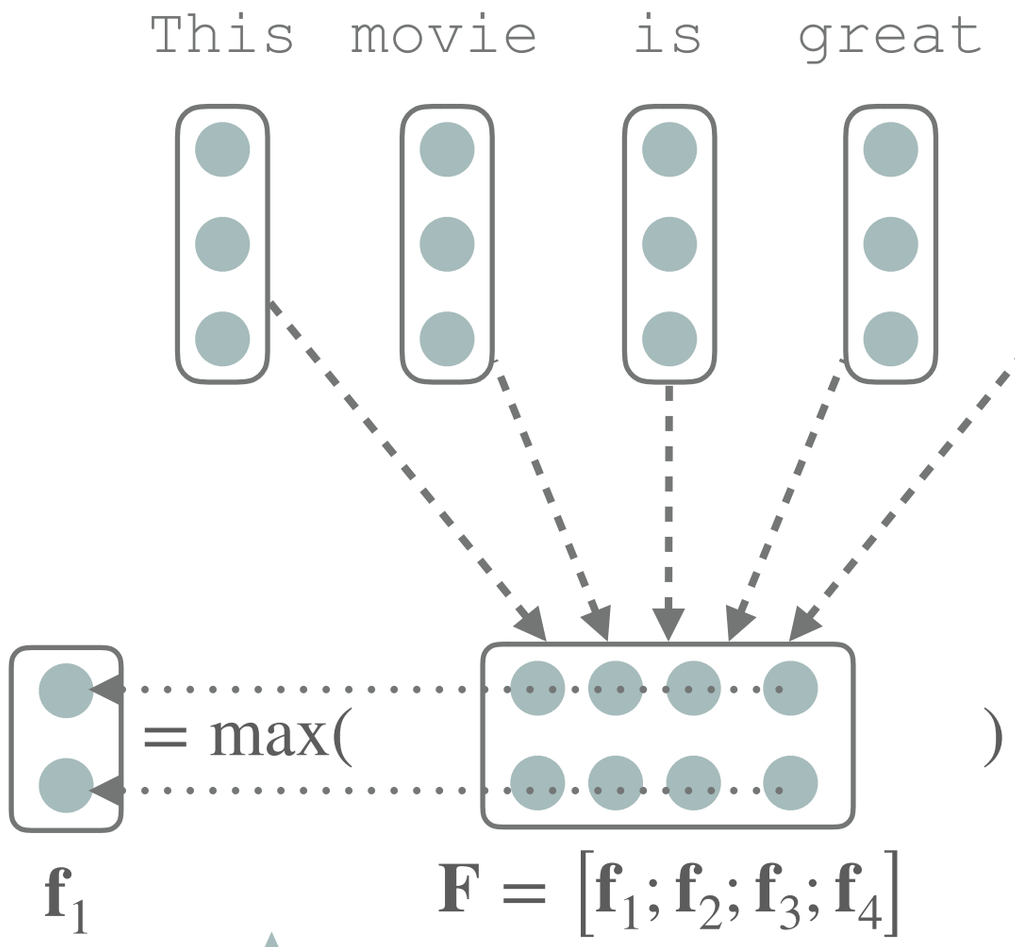
POOLING OPERATION



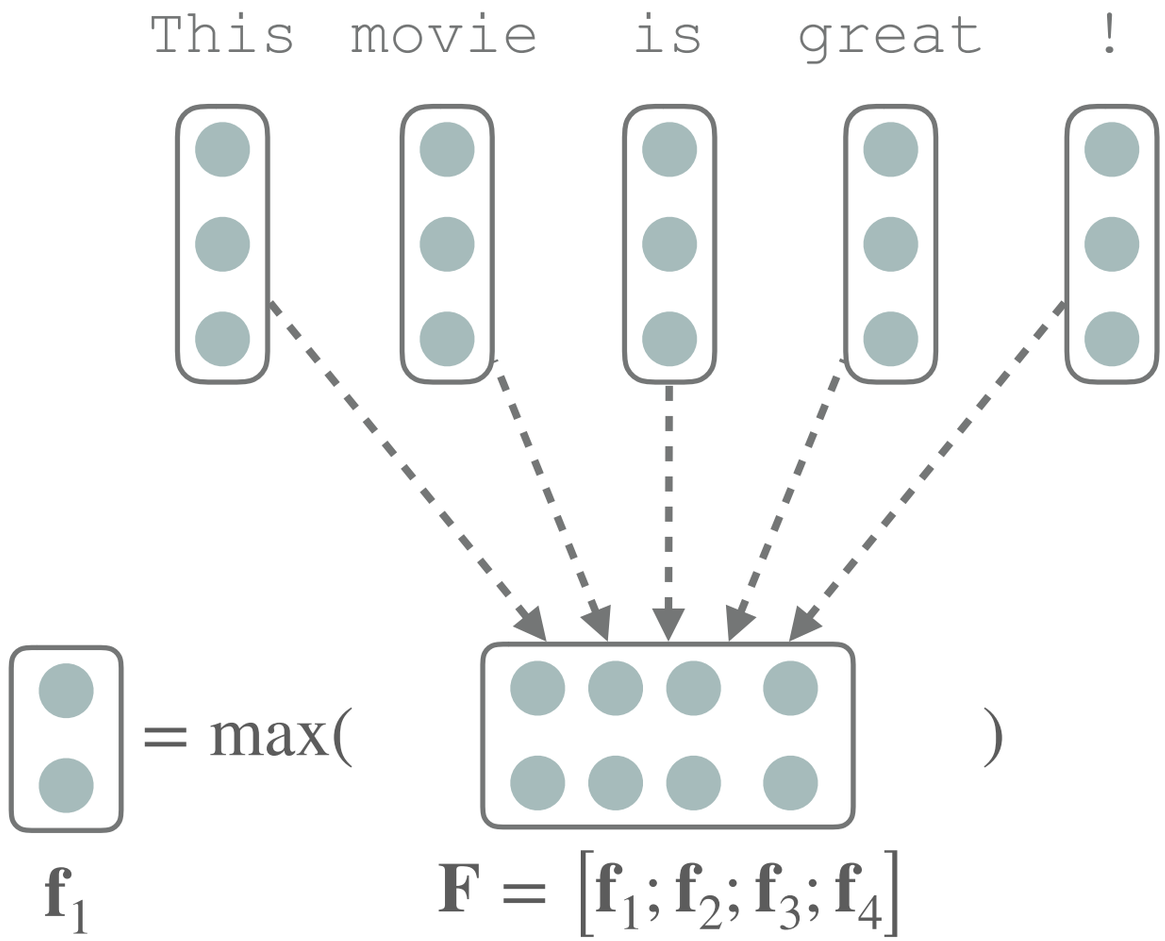
$$\mathbf{F} = [\mathbf{f}_1; \mathbf{f}_2; \mathbf{f}_3; \mathbf{f}_4]$$

Local feature extraction with a sliding window

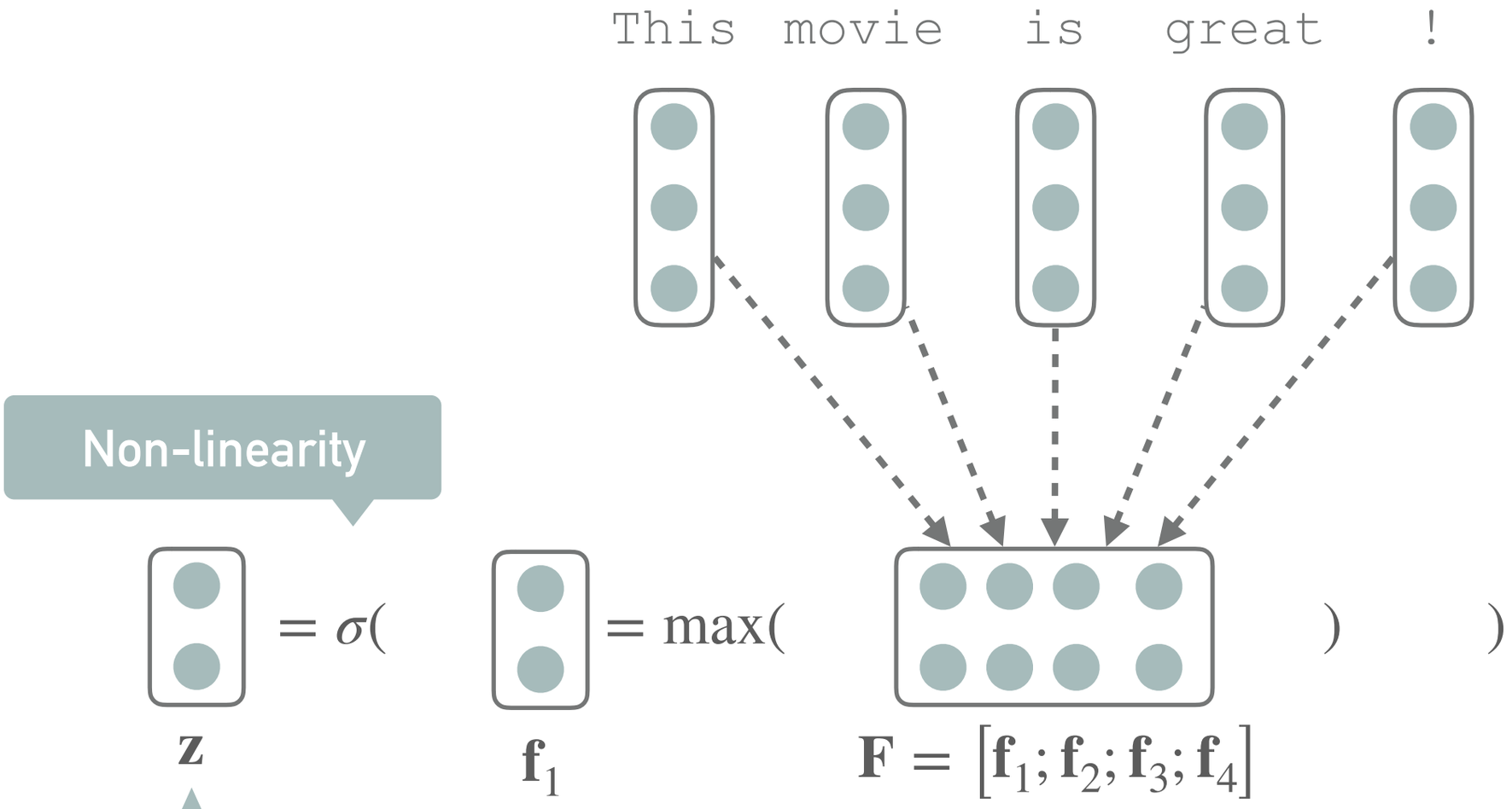
POOLING OPERATION



POOLING OPERATION



POOLING OPERATION



SUMMARY

Convolution

- The size of sliding window is an hyper-parameter (2, 3, 4…)
- The hidden representation of convolutions with different sliding window sizes can be concatenated
- The sentence can be padded with begin/end of sentence tokens

Pooling and activation functions

- Pooling functions: max, min, mean
- Activation functions: tanh, relu, sigmoid

Word representation

Word representation can be constructed with a CNN on character embeddings (and possibly concatenated with a word embedding)