POLITECNICO
MILANO 1863

Deep Learning: Theory, Techniques & Applications
- Word Embedding -

Prof. Matteo Matteucci – *matteo.matteucci@polimi.it*

Department of Electronics, Information and Bioengineering
Artificial Intelligence and Robotics Lab - Politecnico di Milano

Thanks Microsoft

Gold sponsor of Deep Learning and Image Classification courses in the 2017/2018 academic year!!!!

Recall Machine Learning Paradigms

Immagine you have a certain experience E, i.e., a dataset, and let's name it

$$D = x_1, x_2, x_3, \ldots, x_N$$

- **_Supervised Learnig_**: given the desired outputs $t_1, t_2, t_3, \ldots, t_N$ learn to produce the correct output given a new set of input
- **_Unsupervised learning_**: exploit regularities in D to build a representation to be used for reasoning or prediction
- **_Reinforcement learning_**: producing actions $a_1, a_2, a_3, \ldots, a_N$ which the environment, and receiving rewards $r_1, r_2, r_3, \ldots, r_N$ le to maximize rewards in the long term
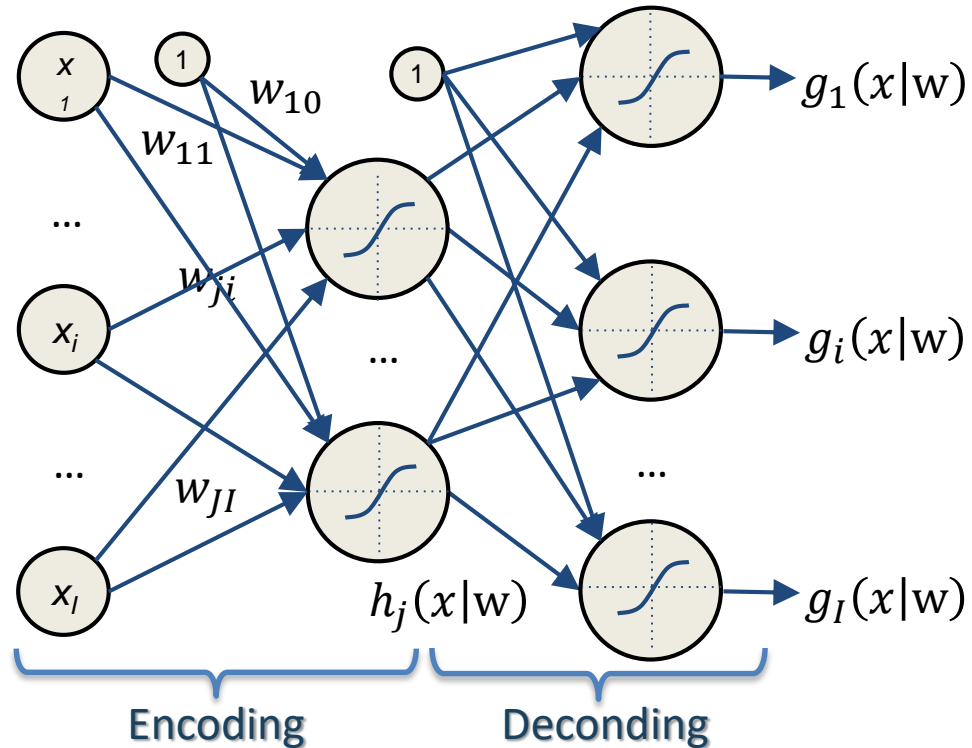
Haven't seen much of it, is it?

This course focuses mainly on Supervised and Unsupervised Learning …

Neural Autoencoder

Network trained to output the input (i.e., to learn the identity function)
- Limited number of units in hidden layers (compressed representation)
- Constrain the representation to be sparse (sparse representation)

$$x \in \Re^I \xrightarrow{enc} h \in \Re^J \xrightarrow{dec} g \in \Re^I$$
$$I << J$$

$$E = \|g_i(x_i|w) - x_i\|^2 + \lambda \sum_j \left| h_j \left(\sum_i w_{ji}^{(1)} x_i \right) \right|$$

Reconstruction error

$$g_i(x_i|w) \sim x_i$$
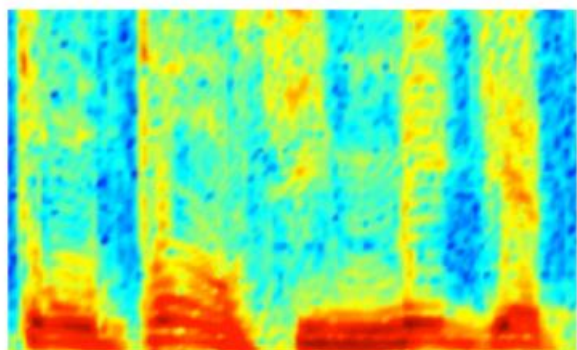
Sparsity term

$$h_j(x_i|w) \sim 0$$

Word Embedding Motivation

Natural language processing systems treat words as discrete atomic symbols

- 'cat' is encoded as Id537
- 'dog' is encoded as Id143
- …

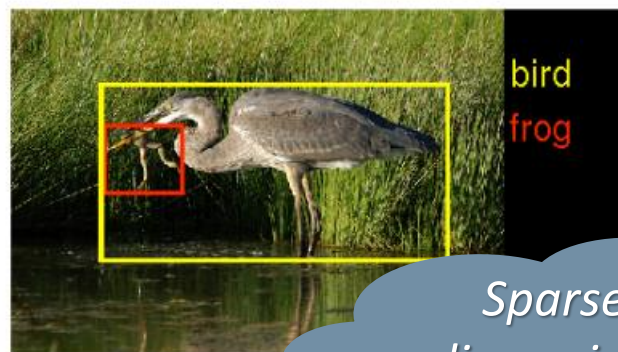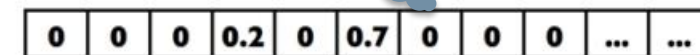*Items in a dictionary …*

A document becomes a Bag of Words

Sparse and high dimensional -> Curse of Dimensionality!

Audio Spectrogram

DENSE

Image pixels

DENSE

Word, context, or document vectors

SPARSE

Encoding Text is a Serious Thing

Performance of real-world applications (e.g., chatbot, document classifiers, information retrieval systems) depends on input encoding and several have been proposed:

Local representations
- N-grams
- Bag-of-words
- 1-of-N coding

$\xrightarrow{\text{Language Model}}$

Continuous representations
- Latent Semantic Analysis
- Latent Dirichlet Allocation
- Distributed Representations

Determine $P(s = w_1, \ldots, w_k)$ in some domain of interest

$$P(s_k) = \prod_i^k P(w_i \mid w_1, \ldots, w_{i-1})$$

In traditional n-gram language models "the probability of a word depends only on the context of n−1 previous words"

$$\hat{P}(s_k) = \prod_i^k P(w_i \mid w_{i-n+1}, \ldots, w_{i-1})$$

Typical ML-smoothing learning process (e.g., Katz 1987):
- compute $\hat{P}(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) = \frac{\#w_{i-n+1}, \ldots, w_{i-1}, w_i}{\#w_{i-n+1}, \ldots, w_{i-1}}$
- smooth to avoid zero probabilities

N-gram Language Model: Curse of Dimensionality

Let's assume you train a 10-gram LM on a corpus of 100.000 unique words

- The model lives in a 10D hypercube where each dimension has 100.000 slots
- Model training ↔ assigning a probability to each of the 100.000^{10} slots
- _Probability mass vanishes_ → more data is needed to fill the huge space
- The more data, the more unique words! → Is not going to work …

In practice:

- Corpuses can have 10^6 unique words
- Contexts are typically limited to size 2 (trigram model),
  e.g., famous Katz (1987) smoothed trigram model
- With short context length a lot of information is not captured

N-gram Language Model: Word Similarity Ignorance

Let assume we observe the following similar sentences

- *Obama speaks to the media in Illinois*
- *The President addresses the press in Chicago*

With classic one-hot vector space representations

- speaks   =   [0 0 1 0 … 0 0 0 0]
- addresses =   [0 0 0 0 … 0 0 1 0]

  speaks ⊥ addresses

- obama   =   [0 0 0 0 … 0 1 0 0]
- president =   [0 0 0 1 … 0 0 0 0]

  obama ⊥ president

- illinois   =   [1 0 0 0 … 0 0 0 0]
- chicago   =   [0 1 0 0 … 0 0 0 0]

  illinois ⊥ chicago

In each case, word pairs share no similarity, and we need word similarity to generalize
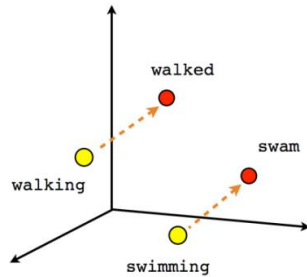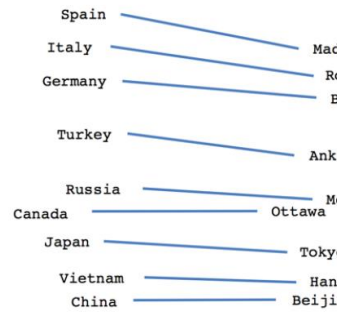
Embedding

Any technique mapping a word (or phrase) from it's original high-dimensional input space (the body of all words) to a lower-dimensional numerical vector space - so one *embeds* the word in a different space
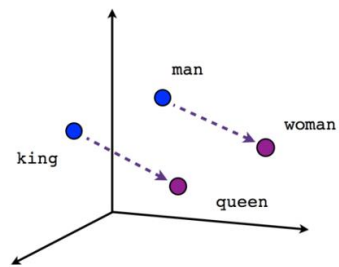
Male-Female
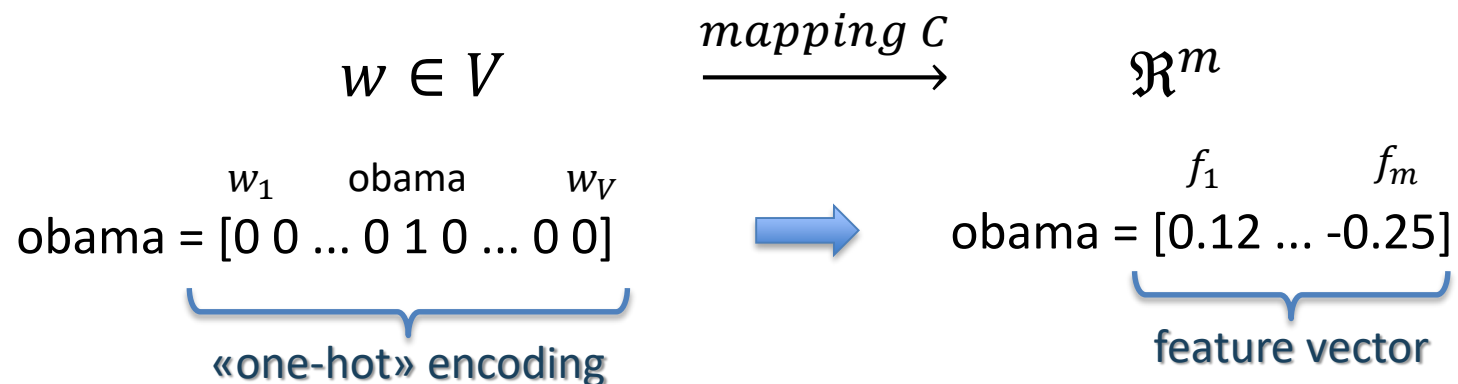
Verb tense

Country-Capital

Closer points are closer in meaning and they form clusters ...

Word Embedding: Distributed Representation

Each unique word w in a vocabulary V (typically $\|V\| > 10^6$) is mapped to a point in a real continuous m-dimensional space (typically $100 < m < 500$)

$$w \in V \xrightarrow{\;mapping\; C\;} \Re^m$$

$$\underset{\text{«one-hot» encoding}}{\underbrace{\overset{w_1 \qquad obama \qquad w_V}{obama = [0\ 0\ ...\ 0\ 1\ 0\ ...\ 0\ 0]}}} \qquad \Longrightarrow \qquad \underset{\text{feature vector}}{\underbrace{\overset{f_1 \qquad\quad f_m}{obama = [0.12\ ...\ \text{-}0.25]}}}$$

Fighting the curse of dimensionality with:

- Compression *(dimensionality reduction)*
- Smoothing *(discrete to continuous)*
- Densification *(sparse to dense)*

Similar words should end up to be close to each other in the feature space
...

Neural Net Language Model (Bengio et al. 2003)

For each training sequence: input = (context, target) pair: $(w_{t-n+1} \dots w_{t-1}, w_t)$

objective: minimize $E = -\log \widehat{P}(w_t \mid w_{t-n+1} \dots w_{t-1})$

softmax. i^{th} output $= \widehat{P}(w_i = w_t \mid w_{t-n+1} \dots w_{t-1})$

OUTPUT LAYER

$|V|$ probabilities that sum to 1

tanh

HIDDEN LAYER
nonlinear

$500 < h < 1000$ (typically)

concatenation

PROJECTION LAYER
linear

$(n-1) \cdot m$

$C(w_{t-n+1})$ $C(w_{t-2})$ $C(w_{t-1})$

table lookup in shared $C_{|V|,m}$

INPUT LAYER

0000......0010   ...   0010......0000   0000......1000 $(n-1) \cdot |V|$

w_{t-n+1} w_{t-2} w_{t-1}

input context: $(n-1)$ past words

Projection layer contains the word vectors in $C_{|V|,m}$
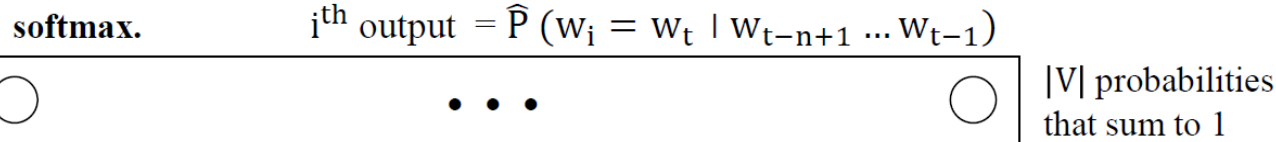
Neural Net Language Model (Bengio et al. 2003)

Neural Net Language Model (Bengio et al. 2003)

For each training sequence: input = (context, target) pair: $(w_{t-n+1} \dots w_{t-1}, w_t)$

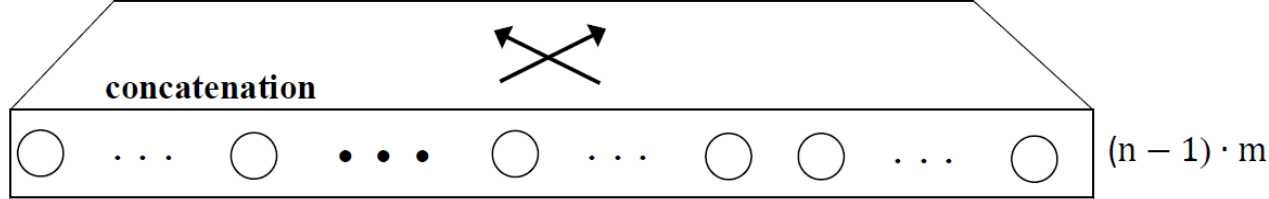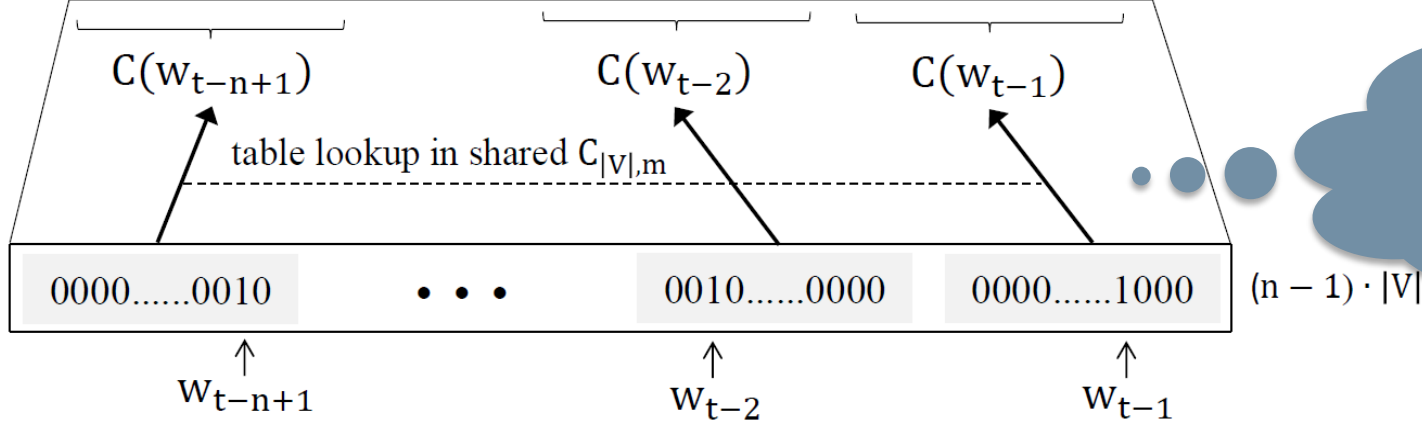objective: minimize $E = -\log \widehat{P}(w_t | w_{t-n+1} \dots w_{t-1})$

OUTPUT LAYER

softmax. i^{th} output $= \widehat{P}(w_i = w_t | w_{t-n+1} \dots w_{t-1})$

Training by stochastic gradient descent has complexity
$$n \times m + n \times m \times h + \boldsymbol{h} \times |V|$$

tanh

HIDDEN LAYER
nonlinear

$500 < \dots$
(typically)

concatenation

PROJECTION LAYER
linear

Softmax is used to output a multinomial distribution
$$\widehat{P}(w_i = w_t | w_{t-n+1}, \dots, w_{t-1}) = \frac{e^{y_{w_i}}}{\sum_{i'}^{|V|} e^{y_{w_{i'}}}}$$

$C(w_{t-n+1})$ $C(w_{t-2})$

table lookup in shared $C_{|V|,m}$

- $y = b + U \cdot \tanh(d + H \cdot x)$
- x is the concatenation $C(w)$ of the context weight vectors
- d and b are biases (respectively h and $|V|$ elements)
- U is the $|V| \times h$ matrix with hidden-to-output weights
- H is the $(h \times (n-1) \cdot m)$ projection-to-hidden weights matrix

INPUT LAYER

0000......0010 $\cdots$ 0010......

input context:
$(n-1)$ past words

w_{t-n+1} w_{t-2} w_{t-1}

Neural Net Language Model (Bengio et al. 2003)

For each training sequence: input = (context, target) pair: $(w_{t-n+1} \ldots w_{t-1}, w_t)$
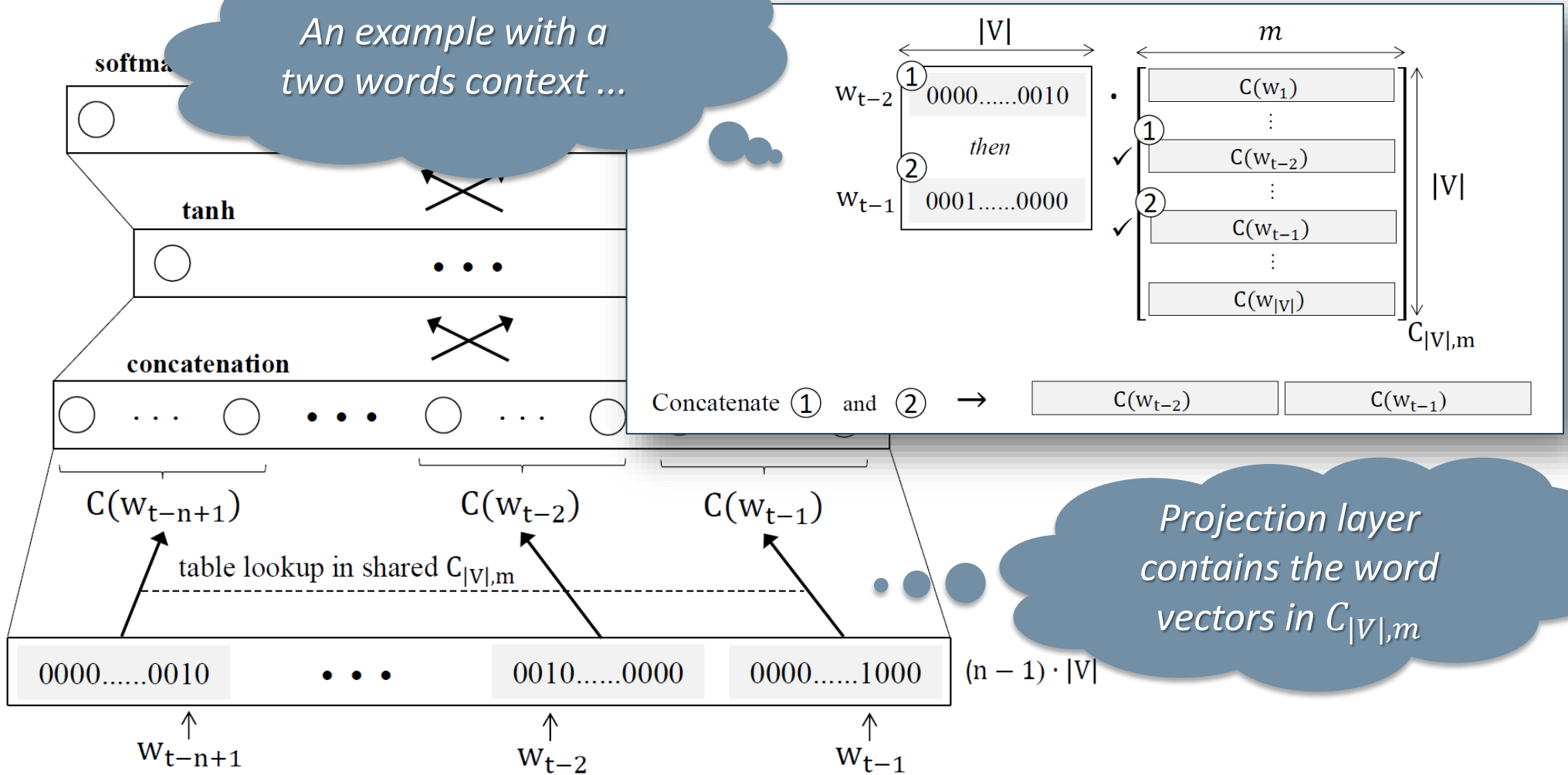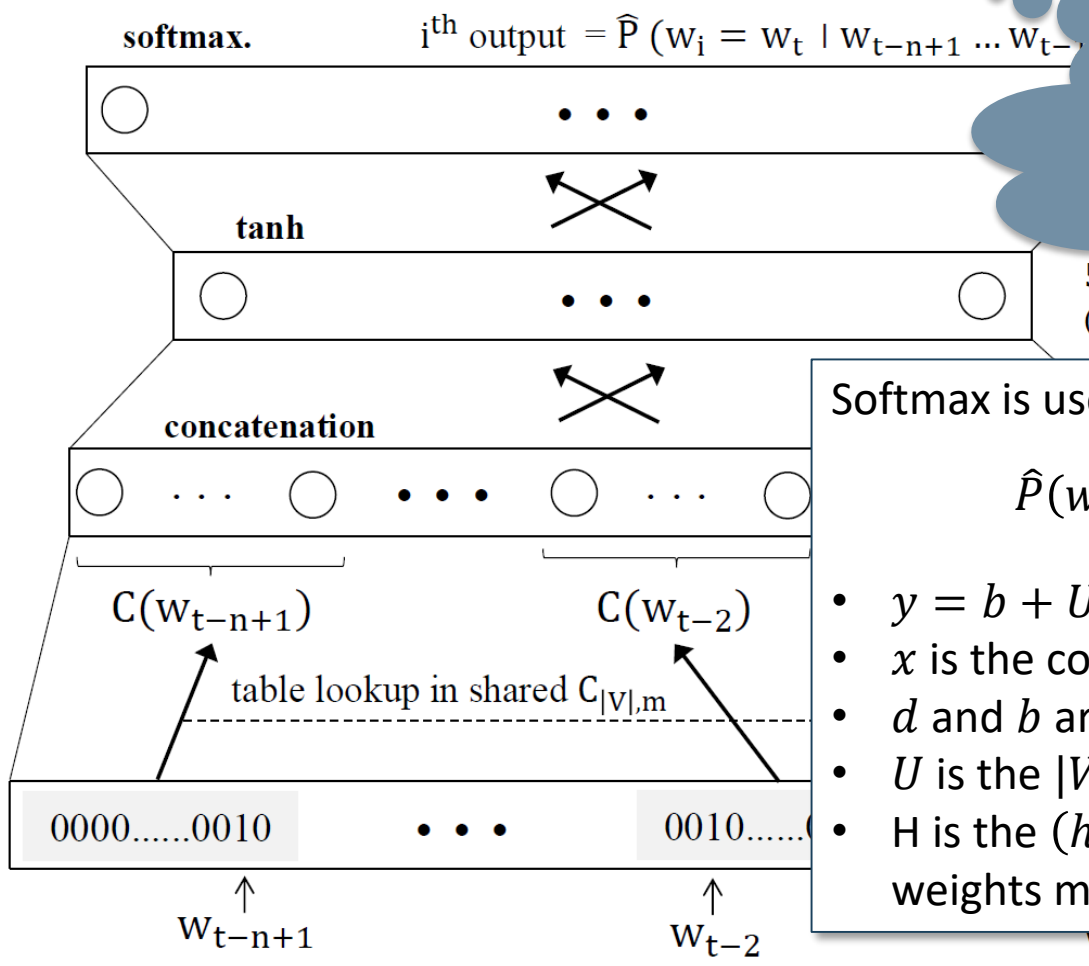objective: minimize $E = -\log \widehat{P}(w_t \mid w_{t-n+1} \ldots w_{t-1})$

OUTPUT LAYER

HIDDEN LAYER
nonlinear

PROJECTION LAYER
linear

INPUT LAYER

input context:
$(n-1)$ past words

softmax.

concatenation

$C(w_{t-n+1})$ $C(w_{t-2})$ $C(w_{t-1})$

table lookup in shared $C_{|V|,m}$

0000......0010   0010...

w_{t-n+1} w_{t-2}
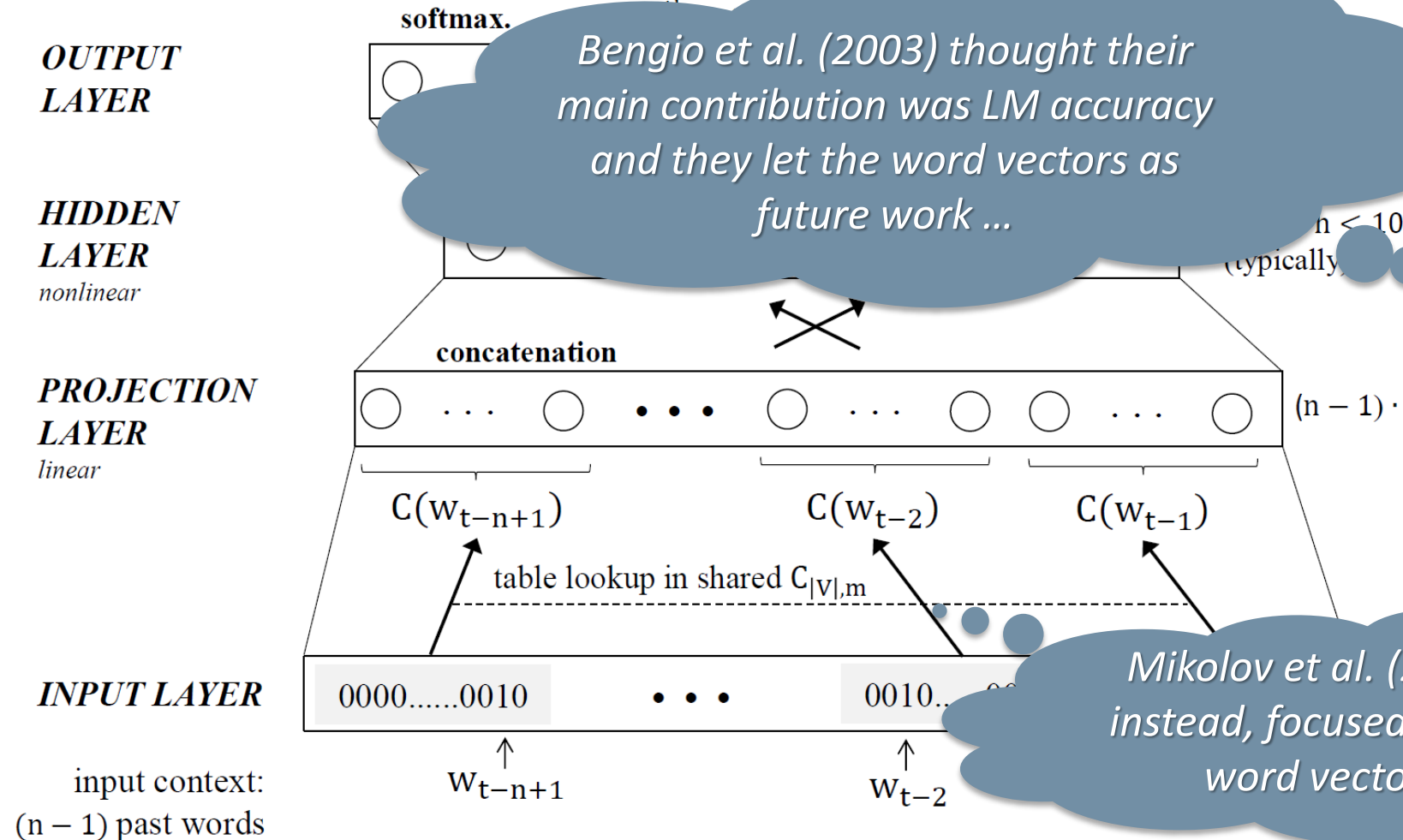
Bengio et al. (2003) thought their main contribution was LM accuracy and they let the word vectors as future work ...

Mikolov et al. (2013), instead, focused on the word vectors

Tested on Brown (1.2M words, $V \cong$ 16K, 200K test set) and AP News (14M words, $V \cong$ 150K reduced to 18K, 1M test set)
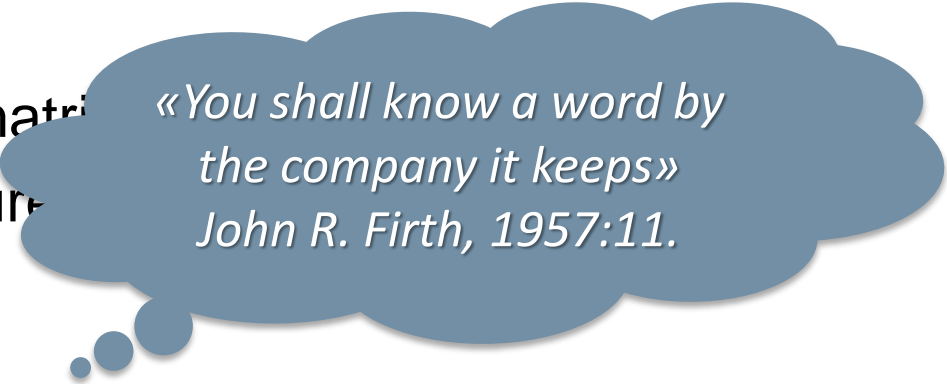
Brown: h=100, n=5, m=30
AP News: h=60, n=6, m=100
- **3 week** training using **40 cores**
- 24% (Brown) and 8% (AP News) relative improvement wrt traditional smoothed n-gram in terms of test set perplexity

Due to **complexity**, NNLM can't be applied to large data sets and it shows ...nce on rare words

Google's word2vec (Mikolov et al. 2013a)

Idea: achieve better performance allowing a simpler (shallower) model to be trained on much larger amounts of data

- No hidden layer (leads to 1000X speed up)
- Projection layer is shared (not just the weight matri...
- Context contain words bot from history and futur...

«You shall know a word by the company it keeps» John R. Firth, 1957:11.

...Pelé has called **Neymar** an excellent player...
...At the age of just 22 years, **Neymar** had scored 40 goals in 58 internationals...
...occasionally as an attacking midfielder, **Neymar** was called a true phenomenon...

These words will represent **Neymar**

Google word2vec Flavors

Input   projection   output
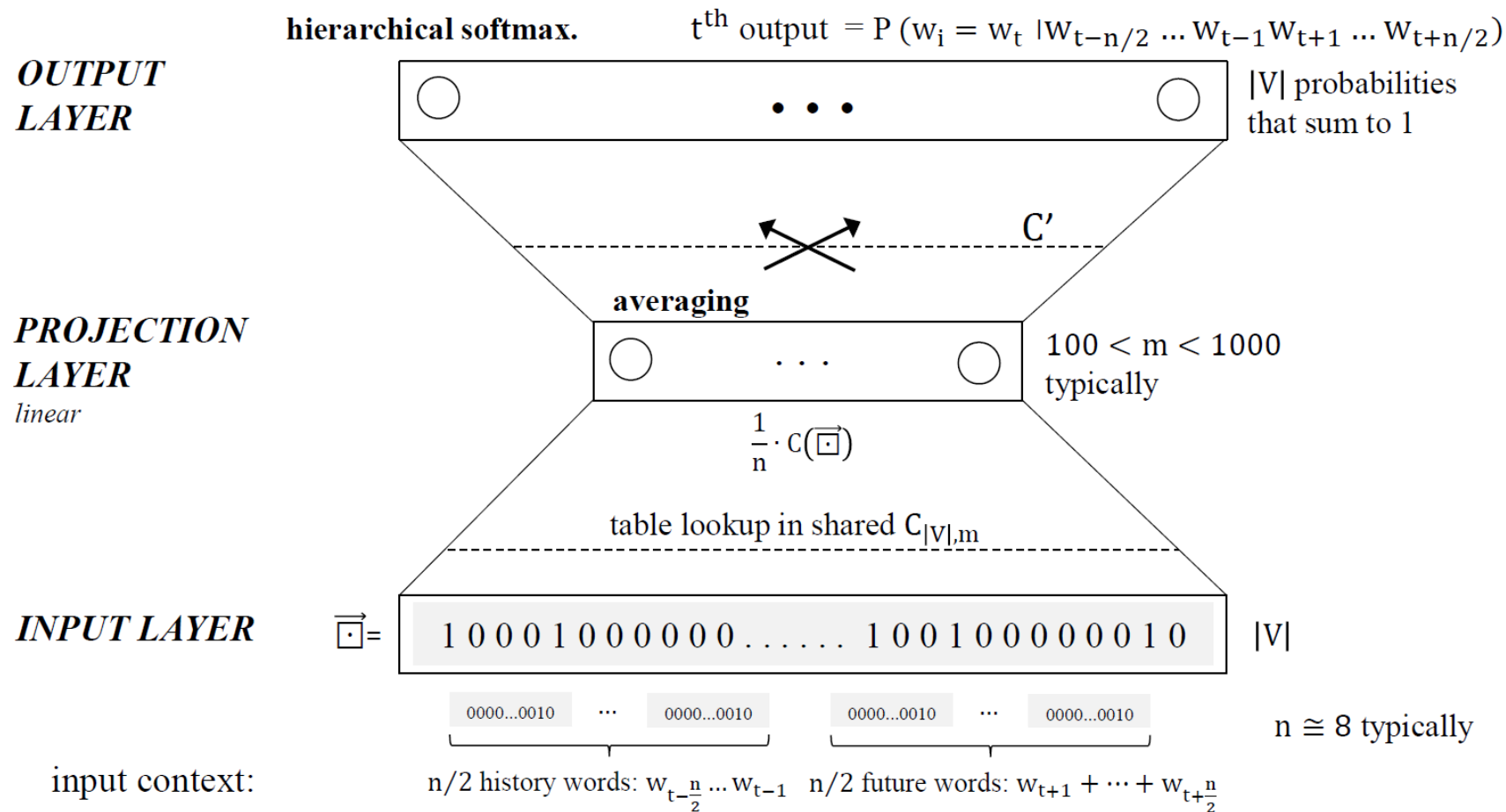
w(t)

w(t-2)

w(t-1)

w(t+1)

w(t+2)
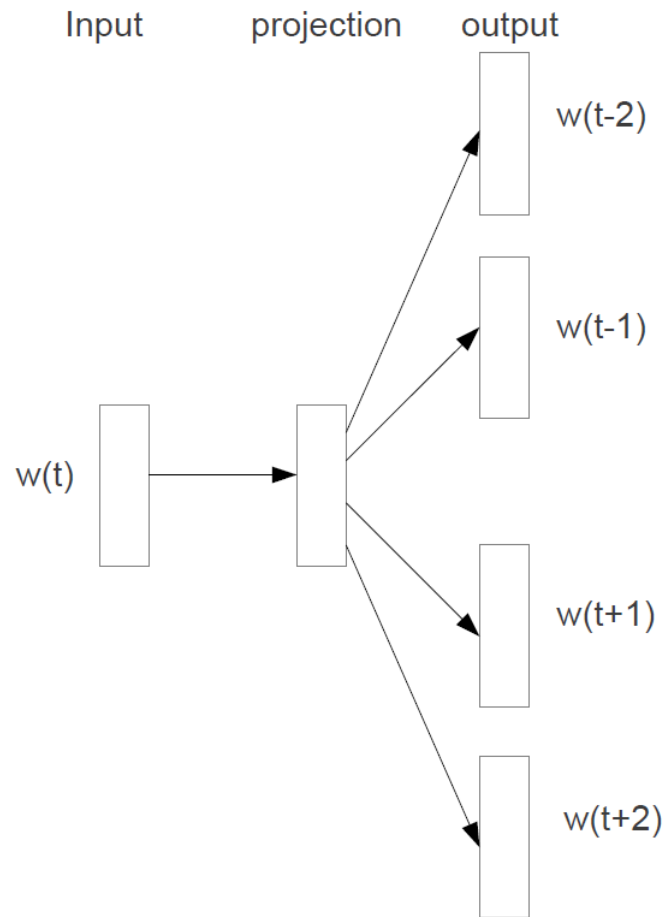
Skip-gram architecture

Word2vec's Continuous Bag-of-Words (CBOW)

For each training sequence: input = (context, target) pair: $(w_{t-\frac{n}{2}} \dots w_{t-1} w_{t+1} \dots w_{t+\frac{n}{2}}, w_t)$

objective: minimize $E = -\log \widehat{P}(w_t \mid w_{t-n/2} \dots w_{t-1} w_{t+1} \dots w_{t+n/2})$

hierarchical softmax. t^{th} output $= P(w_i = w_t \mid w_{t-n/2} \dots w_{t-1} w_{t+1} \dots w_{t+n/2})$

OUTPUT LAYER $|V|$ probabilities that sum to 1

C'

averaging

PROJECTION LAYER $100 < m < 1000$ typically
linear

$\frac{1}{n} \cdot C(\overrightarrow{\square})$

table lookup in shared $C_{|V|,m}$

INPUT LAYER $\overrightarrow{\square} =$ 1 0 0 0 1 0 0 0 0 0 0 . . . . . . 1 0 0 1 0 0 0 0 0 0 1 0 $|V|$
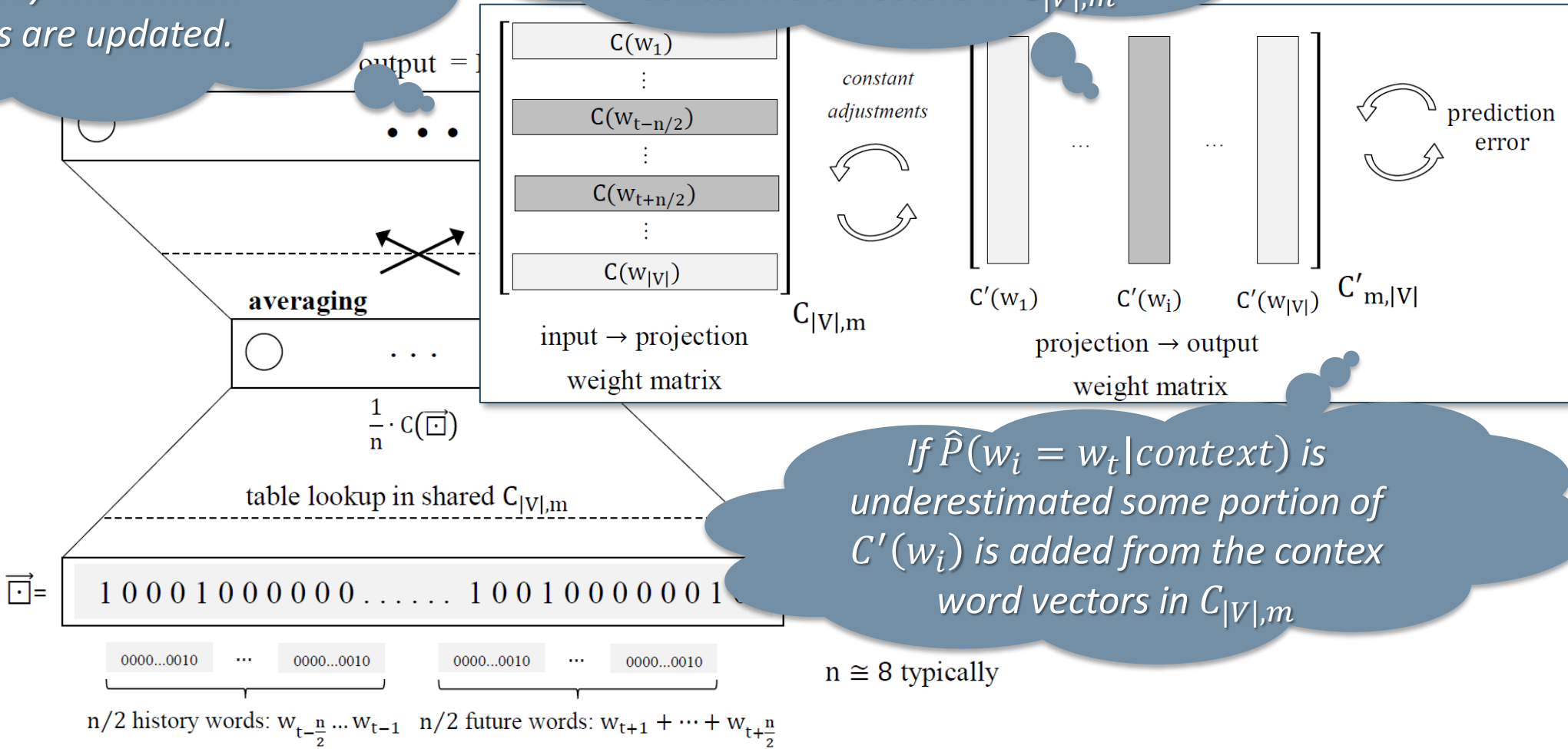
0000...0010 ... 0000...0010   0000...0010 ... 0000...0010 $n \cong 8$ typically

input context: n/2 history words: $w_{t-\frac{n}{2}} \dots w_{t-1}$ n/2 future words: $w_{t+1} + \dots + w_{t+\frac{n}{2}}$

Word2vec's Continuous Bag-of-Words (CBOW)

Word2vec facts

Word2vec shows significant improvements w.r.t. the NNML

- Complexity is $n \times m + m \times log|V|$ (Mikolov et al. 2013a)
- On Google news 6B words training corpus, with $|V| \sim 10^6$
  - CBOW with m=1000 took 2 days to train on 140 cores
  - Skip-gram with m=1000 took 2.5 days on 125 cores
  - NNLM (Bengio et al. 2003) took 14 days on 180 cores, for m=100 only!
- word2vec training speed $\cong$ 100K-5M words/s
- Best NNLM: 12.3% overall accuracy vs. Word2vec (with Skip-gram): 53.3%

Capital-Country	Past tense	Superlative	Male-Female	Opposite
Athens: **Greece**	walking: **walked**	easy: **easiest**	brother: **sister**	ethical: **unethical**

Adapted from Mikolov et al. (2013a)

Regularities in word2vec Embedding Space

Country and Capital Vectors Projected by PCA
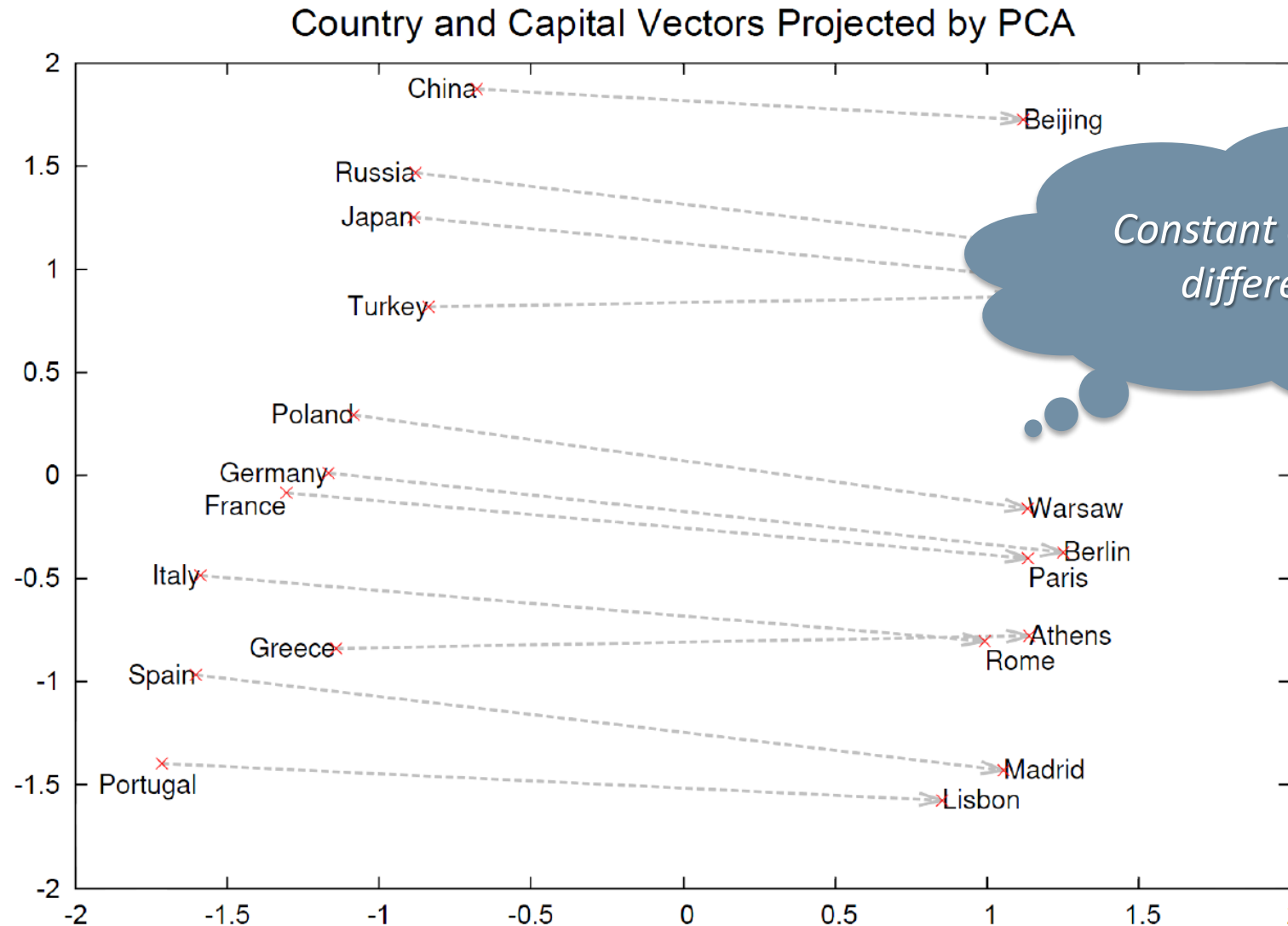
Constant country-capital difference vector.

Mikolov et al. (2013b)

Picture taken from:
https://www.scribd.com/document/285890694/NIPS-DeepLearningWorkshop-NNforText

Regularities in word2vec Embedding Space

Country and Capital Vectors Projected by PCA

Constant female-male difference vector.

Mikolov et al. (2013b)

Regularities in word2vec Embedding Space

Vector operations are supported make «intuitive sense»:

- $w_{king} - w_{man} + w_{woman} \cong w_{queen}$
- $w_{paris} - w_{france} + w_{italy} \cong w_{rome}$
- $w_{windows} - w_{microsoft} + w_{google} \cong w_{android}$
- $w_{einstein} - w_{scientist} + w_{painter} \cong w_{picasso}$
- $w_{his} - w_{he} + w_{she} \cong w_{her}$
- $w_{cu} - w_{copper} + w_{gold} \cong w_{au}$
- ...
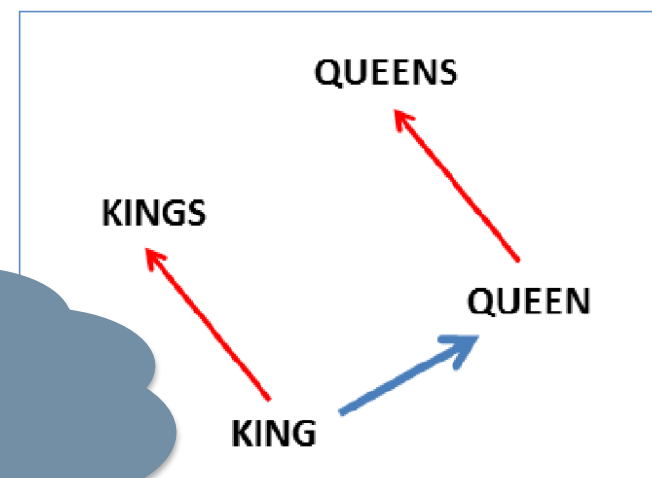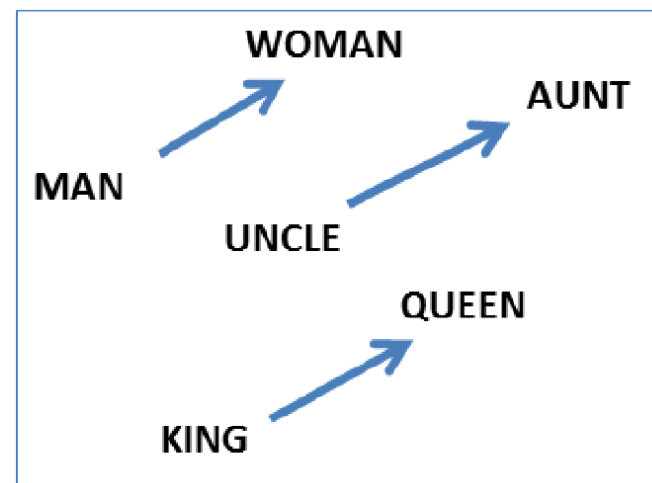
«You shall know a word by the company it keeps»
John R. Firth, 1957:11.

Picture taken from:
https://www.scribd.com/document/285890694/NIPS-DeepLearningWorks

Applications of word2vec in Information Retrieval

Example query: "restaurants in mountain view that are not very good"

Forming the phrases: "restaurants in (mountain view) that are (not very good)"

Adding the vectors: "restaurants + in + (mountain view) + that + are + (not very good)"

Expression	Nearest tokens
Czech + currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa, flag carrier Lufthansa
Russian + river	Moscow, Volga River, upriver, Russia
French + actress	Juliette Binoche, Vanessa Paradis, Charlotte Gainsbourg

(Very simple and efficient, but will not work well for long sentences or documents)

Applications of word2vec in Document Classification/Similarity

With BoW D_1 and D_2 are equally similar to D_0.

document 1

Obama
speaks
to
the
media
in
Illinois

'Obama' 'greets'
'President' 'speaks'
'Chicago'
'media'
'Illinois' 'press'

word2vec embedding

document 2

The
President
greets
the
press
in
Chicago

D_1 **Obama** **speaks** to the **media** in **Illinois.**

$1.07 = 0.45 + 0.24 + 0.20 + 0.18$

D_0 The **President** **greets** the **press** in **Chicago**.

$1.63 = 0.49 + 0.42 + 0.44 + 0.28$

D_2 The **band** **gave** a **concert** in **Japan.**
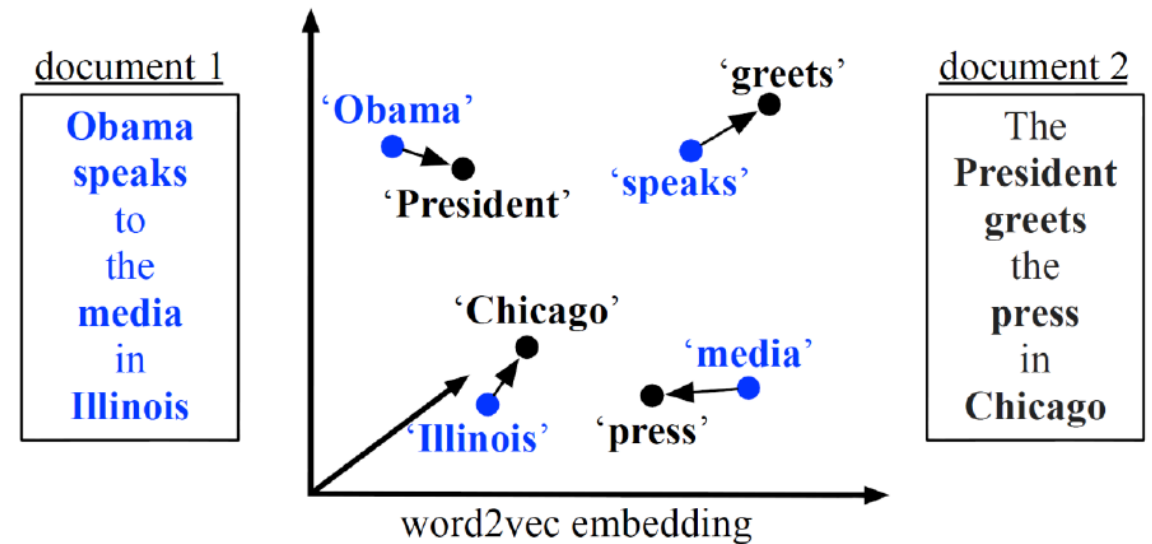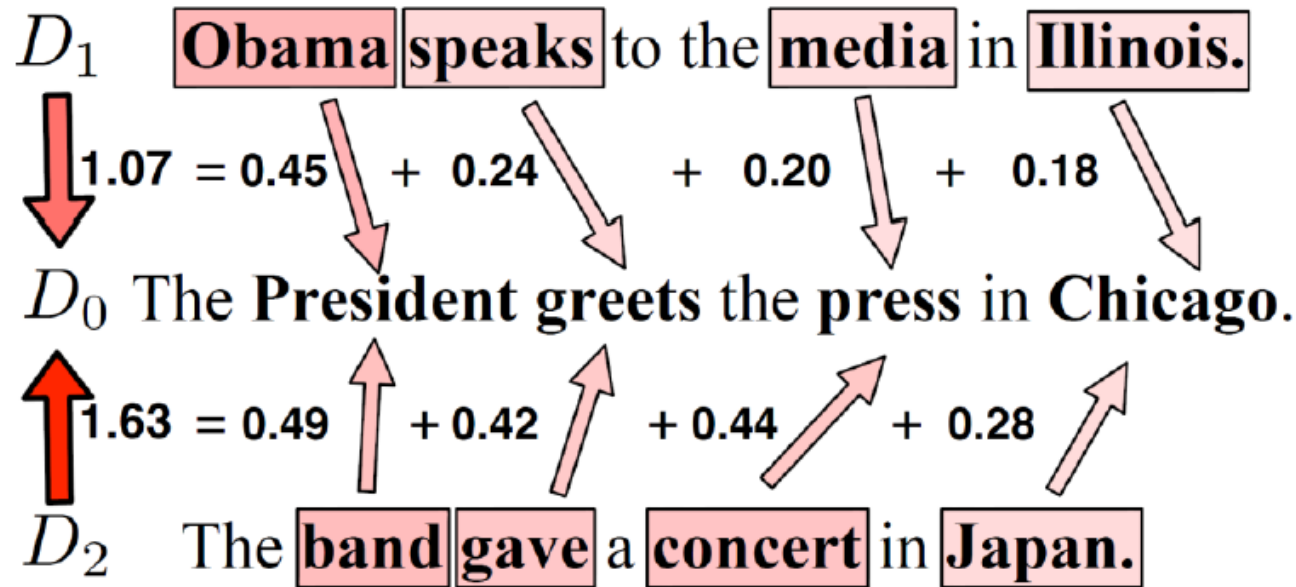
Word embeddings allow to capture the «semantics» of the document ...

Applications of word2vec in Sentiment Analysis

«You shall know a word by the company it keeps»
John R. Firth, 1957:11.

No need for classifiers, just use cosine distances …

```
Enter word or sentence (EXIT to break): sad

Word: sad   Position in vocabulary: 4067

                              Word        Cosine distance
--------------------------------------------------------------
                         saddening               0.727309
                               Sad               0.661083
                          saddened               0.660439
                      heartbreaking              0.657351
                      disheartening              0.650732
                      Meny_Friedman              0.648706
              parishioner_Pat_Patello            0.647586
                         saddens_me              0.640712
                        distressing              0.639909
                    reminders_bobbing            0.635772
                    Turkoman_Shiites             0.635577
                           saddest                0.634551
                        unfortunate               0.627209
                             sorry                0.619405
                         bittersweet              0.617521
                             tragic                0.611279
                           regretful              0.603472
```

GloVe: Global Vectors for Word Representation (Pennington et al. 2014)

GloVe makes explicit what word2vec does implicitly

- Encodes meaning as vector offsets in an embedding space
- Meaning is encoded by ratios of co-occurrece probabilities

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	

Trained by weighted least squares

Refer to Pennington et al. paper for details on this loss function ...

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

GloVe makes explicit what word2vec does implicitly

Tra

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

Nearest Neighbours with GloVe

What are the closest words to the target word *frog*:

1. *Frog*
2. Frogs
3. Toad
4. Litoria
5. Leptodactylidae
6. Rana
7. Lizard
8. Eleutherodactylus

3. litoria

4. leptodactylidae

5. rana

7. eleutherodactylus

Recall Machine Learning Paradigms

Immagine you have a certain experience E, i.e., a dataset, and let's name it

$$D = x_1, x_2, x_3, \ldots, x_N$$

- **_Supervised Learnig_**: given the desired outputs $t_1, t_2, t_3, \ldots, t_N$ learn to produce the correct output given a new set of input
- **_Unsupervised learning_**: exploit regularities in D to build a representation to be used for reasoning or prediction
- **_Reinforcement learning_**: producing actions $a_1, a_2, a_3, \ldots, a_N$ which affect the environment, and receiving rewards $r_1, r_2, r_3, \ldots, r_N$ learn to act in order to maximize rewards in the long term

This course focuses mainly on Supervised and Unsupervised Learning …

Deep Learning: Theory, Techniques & Applications
- Word Embedding -

Prof. Matteo Matteucci – *matteo.matteucci@polimi.it*

Department of Electronics, Information and Bioengineering
Artificial Intelligence and Robotics Lab - Politecnico di Milano