

# Нейронные сети

Екатерина Черняк

Факультет компьютерных наук НИУ ВШЭ

April 11, 2018

## 1 Сети прямого распространения

## 2 Сверточные нейронные сети

- CNN для классификации предложений
- CNN для определения близости между предложениями
- CNN для извлечения отношений
- Символьные CNN для классификации предложений

## 1 Сети прямого распространения

## 2 Сверточные нейронные сети

- CNN для классификации предложений
- CNN для определения близости между предложениями
- CNN для извлечения отношений
- Символьные CNN для классификации предложений

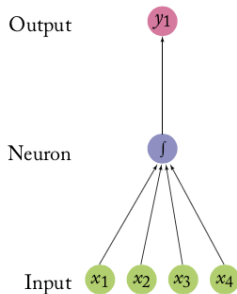
# Искусственный нейрон

$x$  — ВХОД

$y$  — ВЫХОД

$g$  — нелинейная функция активации

$$NN_P(x) = y = W^2 g(xW^1 + b)$$



# Сеть прямого распространения

$$\text{NN}_{\text{MLP2}}(x) = y$$

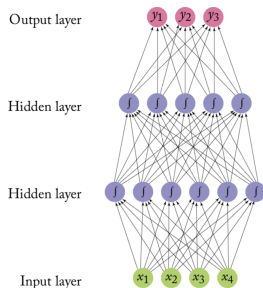
$$\begin{aligned}h_1 &= g^1(xW^1 + b^1) \\h_2 &= g^2(h^1W^2 + b^2) \\y &= h^2W^3\end{aligned}$$

$$x \in \mathbb{R}^{d_{in}}, y \in \mathbb{R}^{d_{out}}$$

$$W^1 \in \mathbb{R}^{d_{in} \times d_1}, b^1 \in \mathbb{R}^{d_1}$$

$$W^2 \in \mathbb{R}^{d_1 \times d_2}, b^2 \in \mathbb{R}^{d_2}$$

$$W^3 \in \mathbb{R}^{d_2 \times d_{out}}$$



# dropout-регуляризация

$$\text{NN}_{\text{MLP2}}(x) = y$$

$$h_1 = g^1(xW^1 + b^1)$$

$$m^1 \sim \text{Bernouli}(r^1)$$

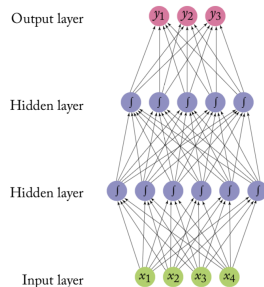
$$\hat{h}^1 = m^1 \odot h^1$$

$$h_2 = g^2(\hat{h}^1 W^2 + b^2)$$

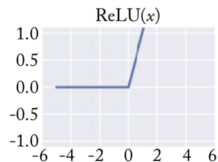
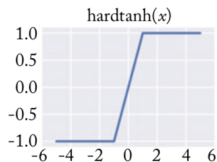
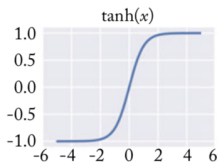
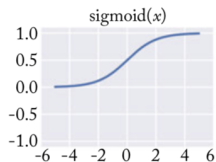
$$m^2 \sim \text{Bernouli}(r^2)$$

$$\hat{h}^2 = m^2 \odot h^2$$

$$y = \hat{h}^2 W^3$$



# Нелинейные функции активации



# Векторное представление текста

## 1 Мешок слов [Bag of Words, BoW]

- ▶  $|\text{word} \in V| = N$  – словарь
- ▶  $x \in D$  – документ,  $|x| = k$
- ▶  $\vec{x}$  –  $N$ -мерный вектор,  $\vec{x}_i = f(\text{word}_i, x_i)$ , в котором  $k$  ненулевых компонент

## 2 Распределенное представление слов [Continuous Bag of Words, CBoW]

- ▶ one-hot кодировка: каждое слово  $\text{word}$  –  $N$ -мерный вектор,  $\vec{\text{word}}_i = 1$ , иначе – 0
- ▶ плотные вектора – эмбединги: каждое слово  $\text{word}$  –  $d$ -мерный вектор,  $\vec{\text{word}}_i \in \mathbb{R}$   
Матрица эмбедингов:  $E \in \mathbb{R}^{|V| \times d}$
- ▶  $\text{CBOW}(x) = \frac{1}{k} \sum_i^k E_i$
- ▶  $x = [\vec{\text{word}}_1, \dots, \vec{\text{word}}_k]$



## Другие признаки

Как учесть часть речи или регистр слова? Специальные аффиксы?

Является ли оно именованной сущностью?

Конкатенируем эмбединги для части речи и для слова. Эмбединги для части речи инициализируются случайно и дообучаются во время обучения.

# Классификатор на основе сетей прямого распространения

- На последнем слое сети находится функция активации softmax для классификации на  $K$  классов:

$$z = W_3 l_3 + b_3 \in \mathbb{R}^K$$

- То есть, на выходном слое находится  $K$  нейронов, каждый соответствует своему классу
- Для итоговой классификации:

$$y_j = \text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k \in K} e^{z_k}}$$

- $y_j$  может быть интерпретировано как вероятность класса  $i$
- Как выбрать число скрытых слоев? Например,  
2000  $\rightarrow$  1000  $\rightarrow$  500  $\rightarrow$  100

## 1 Сети прямого распространения

## 2 Сверточные нейронные сети

- CNN для классификации предложений
- CNN для определения близости между предложениями
- CNN для извлечения отношений
- Символьные CNN для классификации предложений

# Сверточные нейронные сети

Сверточные нейронные сети [англ. convolutional neural network]:

- Заимствованы из области компьютерного зрения
- Пик популярности пришелся на 2014 (до +10% аккуратности в задачах классификации), со временем были вытеснены рекуррентными нейронными сетями

Помогают справиться с двумя проблемами:

- 1 Часто входы бывают переменной длины (тексты, абзацы, предложения)
- 2 Если использовать подход, основанный на представлении предложениями окнами, то:
  - ▶ число параметров увеличивается,
  - ▶ нужно подбирать размер окна.

# Слой свертки

Фильтр [англ. filter]:

$w_{1:n}$  – входная последовательность

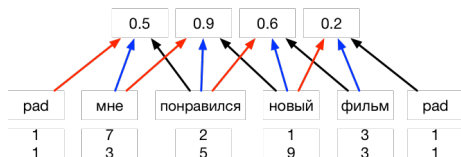
слов,  $E_{w_i}$  – эмбединг слова  $w_i$

$x_i = \oplus(w_{i:i+k-1})$  – окно длины  $k$

Фильтр:  $p_i = g(x_i u)$

$p_i \in \mathbb{R}, x_i \in \mathbb{R}^{k \cdot d_{emb}}, u \in \mathbb{R}^{k \cdot d_{emb}}$

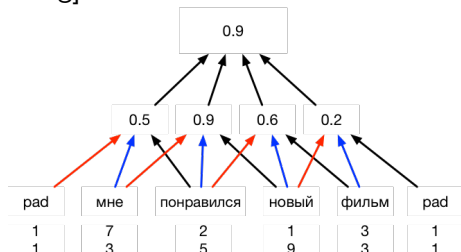
Преобразуем каждое входное окно, но пока размерность входа не уменьшается!



# Слой субдискретизации (пулинга)

Субдискретизация / пулинг [англ. pooling]:

- $p_i$  – выходные значения фильтра

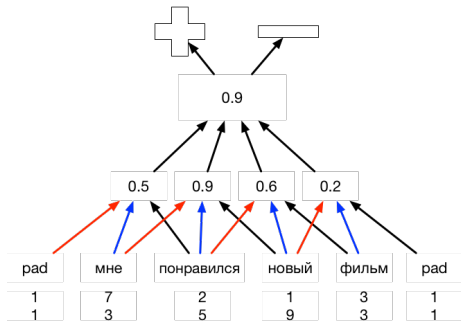


max-пулинг:  $c = \max_i p_i$

- Выбираем самый важный признак из полученных на предыдущем шаге
- Можем использовать и min, и усреднение

# Классификатор на основе сверточной сети

- $y \in [0, 1]$  - истинные значения
- $\hat{y} = c$  - предсказанные значения



- Для обучения сверточной сети можно использовать обычный алгоритм распространения ошибки
- Одномерные фильтры – это сильное ограничение. Что делать, если  $c = 0.5$ ?

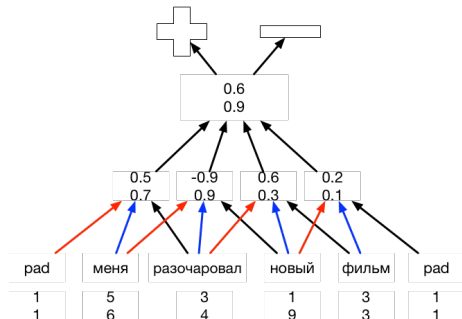
# Многомерные фильтры

Применяем фильтр  $l$  раз:

$$p_i = g(x_i \cdot U + b)$$

$$p_i \in \mathbb{R}^l, x_i \in \mathbb{R}^{k \cdot d_{emb}}$$

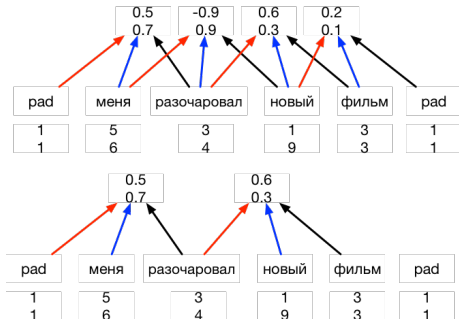
$$U \in \mathbb{R}^{k \cdot d_{emb} \times l}, b \in \mathbb{R}^l$$





# Шаг окна

- Можно использовать непересекающиеся окна, чтобы уменьшить объем вычисления



# Как выбирать вектора слов?

- Случайная инициализация (если нет обученных моделей word2vec, GloVe)
- word2vec, GloVe без обновления
- word2vec, GloVe с обновлением на каждой эпохе (увеличивается количество параметров!)
- Несколько каналов: копируем два входа и
  - ▶ на один подаем word2vec и не обновляем эти входы во время обучения, на второй подаем word2vec и обновляем эти входы во время обучения
  - ▶ на один вход подаем word2vec, на второй – GloVe

# Как использовать pad?

[[мое первое короткое предложение], [второе очень длинное предложение, которое никогда не заканчивается], [третье предложение]]

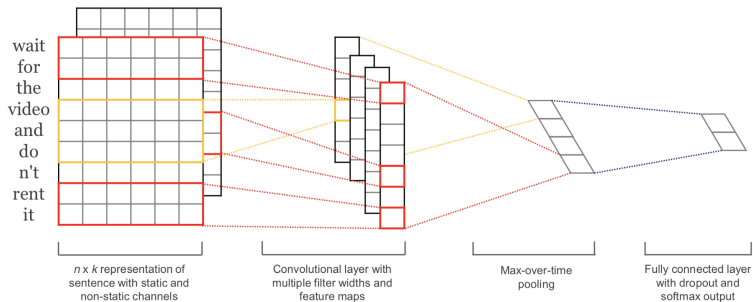
- 1 Неэффективный способ: одно предложение – одна эпоха
- 2 Окружить все предложения балластными символами pad и сделать их одной длины
  - ▶ Надо убедиться, что max-пулинг не выберет значения, соответствующие pad
  - ▶ Надо убрать выбросы, то есть, супер-длинные предложения, возникшие, например, из-за ошибок сегментатора

## 1 Сети прямого распространения

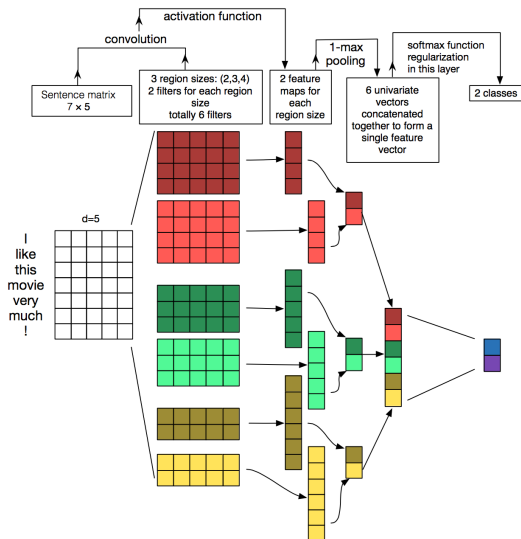
## 2 Сверточные нейронные сети

- CNN для классификации предложений
- CNN для определения близости между предложениями
- CNN для извлечения отношений
- Символьные CNN для классификации предложений

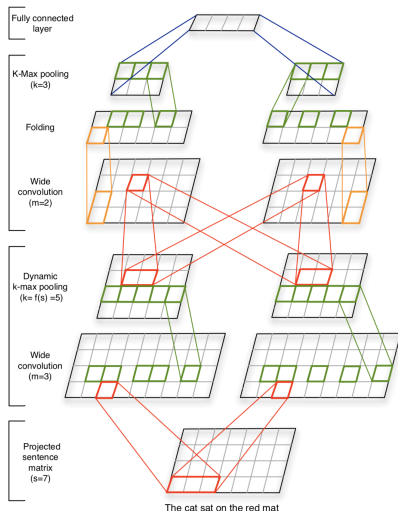
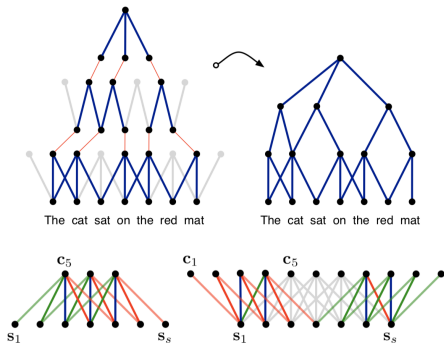
# CNN для классификации предложений [Kim14]



# CNN для классификации предложений [ZW15]



# CNN для классификации предложений [KGB14]



## 1 Сети прямого распространения

## 2 Сверточные нейронные сети

- CNN для классификации предложений
- CNN для определения близости между предложениями
- CNN для извлечения отношений
- Символьные CNN для классификации предложений



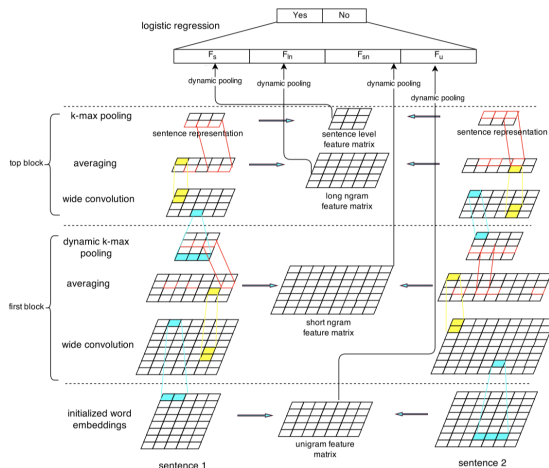
# Bi-CNN-MI для определения парафраза [paraphrase detection] [YS15]

A1: Detroit  
manufacturers have  
raised vehicle prices by  
ten percent

A2: GM, Ford and  
Chrysler have raised car  
prices by five percent

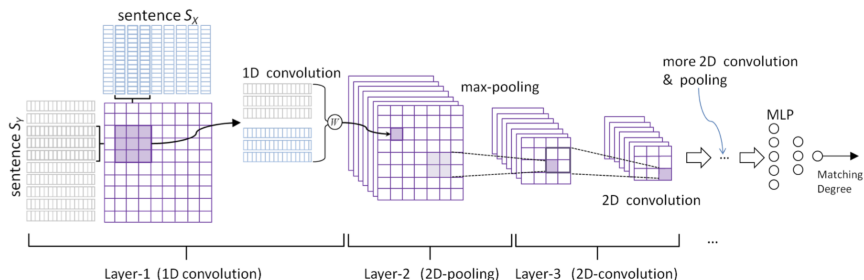
B1: Mary gave birth to  
a son in 2000

B2: He is 18 years old  
and his mother is Mary



# CNN для определения близости между предложениями[HLLC14]

- 1 Дополнение предложения [sentence completion]
- 2 Определение ответа на вопрос [matching a response to a tweet]
- 3 Определения парафраза [paraphrase detection]



## 1 Сети прямого распространения

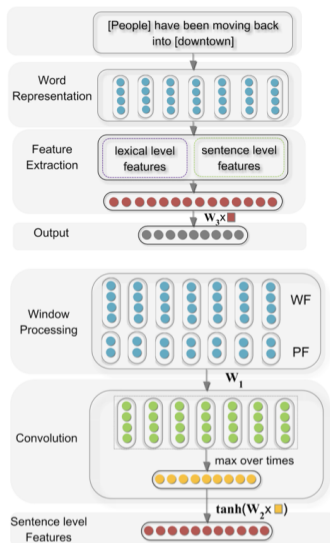
## 2 Сверточные нейронные сети

- CNN для классификации предложений
- CNN для определения близости между предложениями
- **CNN для извлечения отношений**
- Символьные CNN для классификации предложений

# CNN для извлечения отношений [relation extraction] [ZLL<sup>+</sup>14]

The [fire]<sub>e1</sub> inside WTC was caused by  
exploding [fuel]<sub>e2</sub>

The [company]<sub>e1</sub> fabricates [plastic  
chairs]<sub>e2</sub>



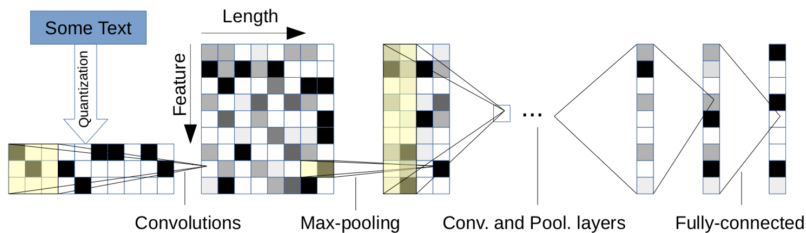
## 1 Сети прямого распространения

## 2 Сверточные нейронные сети





- CNN для классификации предложений
- CNN для определения близости между предложениями
- CNN для извлечения отношений
- Символьные CNN для классификации предложений

# Символьные CNN для классификации предложений [ZZL15]




Представление текста: one-hot вектора для 70 алфавитных и неалфавитных символов



# Источники I

-  Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen, *Convolutional neural network architectures for matching natural language sentences*, Advances in neural information processing systems, 2014, pp. 2042–2050.
-  Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom, *A convolutional neural network for modelling sentences*, arXiv preprint arXiv:1404.2188 (2014).
-  Yoon Kim, *Convolutional neural networks for sentence classification*, arXiv preprint arXiv:1408.5882 (2014).
-  Wenpeng Yin and Hinrich Schütze, *Convolutional neural network for paraphrase identification*, Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2015, pp. 901–911.

## Источники II

-  Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao, *Relation classification via convolutional deep neural network*, Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, pp. 2335–2344.
-  Ye Zhang and Byron Wallace, *A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification*, arXiv preprint arXiv:1510.03820 (2015).
-  Xiang Zhang, Junbo Zhao, and Yann LeCun, *Character-level convolutional networks for text classification*, Advances in neural information processing systems, 2015, pp. 649–657.