Введение

Алексей Андреевич Сорокин

МГУ им. М. В. Ломоносова
весенний семестр 2022–2023 учебного года
Межфакультетский курс “Введение в компьютерную лингвистику” 1 марта, занятие 3
Нейронные сети.

Базовая схема текстовой классификации

- Общая схема классификации:

$$
\begin{aligned}
X &= [x_1, \ldots, x_n], \\
x_i &\in \mathbb{R}^D, \\
s &= \sum(X) = [\sum_i \{x_{i1}\}, \ldots, \sum_i \{x_{iD}\}], \\
p = [p_1, \ldots, p_K] &= \mathrm{softmax}(Ws + b)
\end{aligned}
$$

Базовая схема текстовой классификации

- Общая схема классификации:

$$\begin{aligned} X &= [x_1, \ldots, x_n], \\ x_i &\in \mathbb{R}^D, \\ s &= \sum(X) = [\sum_i \{x_{i1}\}, \ldots, \sum_i \{x_{iD}\}], \\ p = [p_1, \ldots, p_K] &= \mathrm{softmax}(Ws + b) \end{aligned}$$

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).

Базовая схема текстовой классификации

- Общая схема классификации:

$$\begin{aligned} X &= [x_1, \ldots, x_n], \\ x_i &\in \mathbb{R}^D, \\ s &= \sum(X) = [\sum_i \{x_{i1}\}, \ldots, \sum_i \{x_{iD}\}], \\ p = [p_1, \ldots, p_K] &= \mathrm{softmax}(Ws + b) \end{aligned}$$

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)

Базовая схема текстовой классификации

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)

Базовая схема текстовой классификации

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)
- В линейном классификаторе первые две компоненты детерминиро
  обучается только последняя.
- То есть мы требуем, чтобы классы разделялись плоскостью в исходном пространстве признаков.

Базовая схема текстовой классификации

- Сеть состоит из трёх компонент:
  - Вычисление векторов слов x_i по слову w_i (0/1-вектора, $x_i = w_i$).
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)
- В линейном классификаторе первые две компоненты детерминиро
  обучается только последняя.
- То есть мы требуем, чтобы классы разделялись плоскостью в исходном пространстве признаков.
- Если первые две части будут обучаемые, то сеть “сама” научится так представлять тексты, чтобы классы можно было разделить плоскостью.

Обучаемые вектора слов

- В качестве x_i можно брать предобученные вектора (word2vec, FastText, GloVe).
- В этом случае соотношения между векторами отражают общие семантические закономерности.

Обучаемые вектора слов

- В качестве x_i можно брать предобученные вектора (word2vec, FastText, GloVe).
- В этом случае соотношения между векторами отражают общие семантические закономерности.
- Они могут быть нерелевантны для конкретной задачи.
- Можно обучать вектора вместе с задачей.

Обучаемые вектора слов

- Можно обучать вектора вместе с задачей.
- Пусть j – индекс слова в словаре, вектор слова w_j – 0/1-вектор:

$$[0, \ldots, 0, 1_j, 0, \ldots, 0]$$

Обучаемые вектора слов

- Можно обучать вектора вместе с задачей.
- Пусть j – индекс слова в словаре, вектор слова w_j – 0/1-вектор:

$$[0, \ldots, 0, 1_j, 0, \ldots, 0]$$

- Пусть $x_j = Uw_j$, матрица U – обучаемая.
- То есть x_j – j-ый столбец матрицы U.
- То есть первый слой сети вычисляет вектора для каждого словарного слова.

Обучаемые вектора слов

- Можно обучать вектора вместе с задачей.
- Пусть j – индекс слова в словаре, вектор слова w_j – 0/1-вектор:

$$[0, \dots, 0, 1_j, 0, \dots, 0]$$

- Пусть $x_j = Uw_j$, матрица U – обучаемая.
- То есть x_j – j-ый столбец матрицы U.
- То есть первый слой сети вычисляет вектора для каждого словарного слова.
- Эти вектора необязательно отражают общую семантику, как word2vec, но можно взять word2vec в качестве начального приближения при обучении.

Текстовая классификация с обучаемыми векторами

- Схема классификации:

$$
\begin{aligned}
X &= [w_1, \ldots, w_n], \\
w_i &\in \mathbb{R}^D, \\
h_i &= U_{d \times D} w_i, \\
s &= \sum(H) = [\sum_i \{h_{i1}\}, \ldots, \sum_i \{h_{iD}\}], \\
p = [p_1, \ldots, p_K] &= \mathrm{softmax}(Ws + b)
\end{aligned}
$$

Текстовая классификация с обучаемыми векторами

- Схема классификации:

$$\begin{aligned} X &= [w_1, \ldots, w_n], \\ w_i &\in \mathbb{R}^D, \\ h_i &= U_{d \times D} w_i, \\ s &= \sum(H) = [\sum_i \{h_{i1}\}, \ldots, \sum_i \{h_{iD}\}], \\ p = [p_1, \ldots, p_K] &= \operatorname{softmax}(Ws + b) \end{aligned}$$

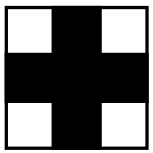
- Сеть состоит из трёх компонент:
  - Вычисление эмбеддингов $h_i \in \mathbb{R}^d$ для слов w_i (0/1-вектора, $x_i = w_i$). При этом $d \sim 100 \ldots 500$, то есть $d \ll D$.
  - Агрегация их в вектор предложения (покоординатный максимум, среднее или сумма).
  - Вычисления самой вероятной метки (однослойный персептрон)

Свёрточные сети

- В тексте нужно учитывать не только представления отдельных слов или символов, но и их групп (энграмм).
- В задачах распознавания образов для этого придуманы свёрточные сети.
- Для выделения пиксельных шаблонов на изображение накладывают фильтры.

Свёрточные сети

- В тексте нужно учитывать не только представления отдельных слов или символов, но и их групп (энграмм).
- В задачах распознавания образов для этого придуманы свёрточные сети.
- Для выделения пиксельных шаблонов на изображение накладывают фильтры.
- Например, изображению



соответствует фильтр

-1	1	-1
1	1	1
-1	1	-1

со свободным членом -4.

- Этот фильтр будет активирован, только если все чёрные квадраты будут заполнены, а ни один белый – нет.

Свёрточные сети

- Формально, двумерный фильтр ширины $d = 2k + 1$ — это матрица

$$\begin{pmatrix} f_{-k,-k} & \dots & f_{-k,k} \\ \dots & \dots & \dots \\ f_{k,-k} & \dots & f_{kk} \end{pmatrix}$$

и пороговое значение f_0.

- Результат применения фильтра в позиции i, j к изображению x:

$$a_{ij} = \sum_{r,s=-k}^{k} f_{r,s} x_{i-r,j-s} - f_0$$

Свёрточные сети

- Формально, двумерный фильтр ширины $d = 2k + 1$ — это матрица

$$\begin{pmatrix} f_{-k,-k} & \dots & f_{-k,k} \\ \dots & \dots & \dots \\ f_{k,-k} & \dots & f_{kk} \end{pmatrix}$$

и пороговое значение f_0.

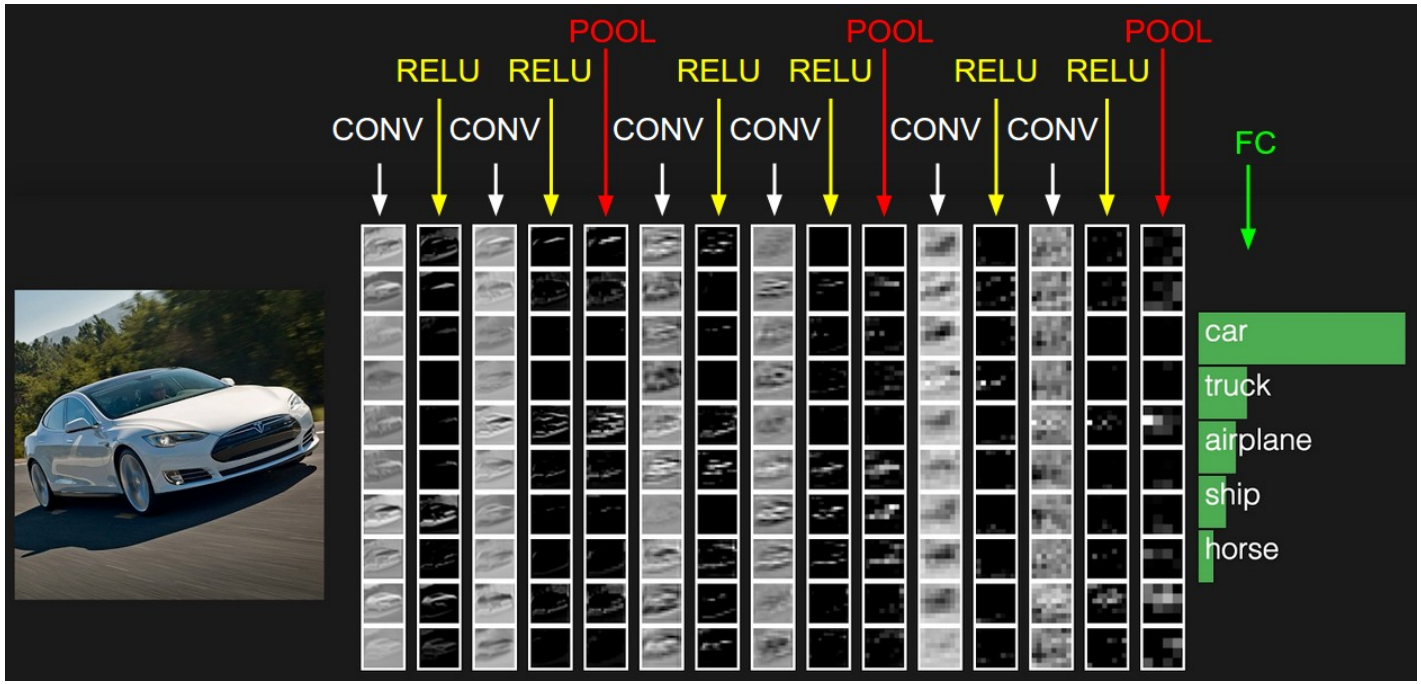
- Результат применения фильтра в позиции i, j к изображению x:

$$a_{ij} = \sum_{r,s=-k}^{k} f_{r,s} x_{i-r,j-s} - f_0$$

- Часто к фильтрам применяют ReLU-активацию $z_{ij} = \max(a_{ij}, 0)$.
- То есть фильтр активируется в случае наличия определённого шаблона.

Свёрточные сети

- Нейронные сети для изображений состоят из десятков последовательных слоёв параллельных фильтров. (то есть к каждому сектору пикселей одновременно применяются несколько фильтров).



Свёрточные сети

- На n-ом слое каждая позиция изображения задаётся вектором $z_{ij}^{(n)}$ из f_n чисел.

Свёрточные сети

- На n-ом слое каждая позиция изображения задаётся вектором $z_{ij}^{(n)}$ из f_n чисел.
- На финальном слое с номером N строится единый вектор $Z \in \mathbb{R}^{f_z}$ для изображения:

$$Z_r = \max_{i,j}(z_{i,j}^{(n)})_r$$

- То есть финальный max-слой проверяет наличие шаблона, заданного компонентой вектора где-нибудь в изображении.

Свёрточные сети

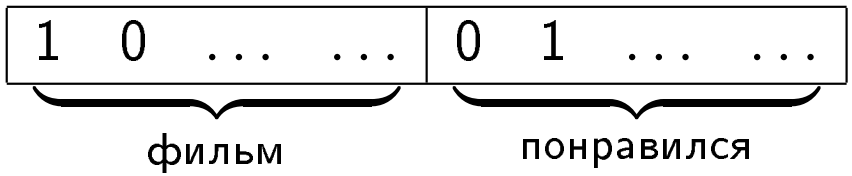
- На n-ом слое каждая позиция изображения задаётся вектором $z_{ij}^{(n)}$ из f_n чисел.
- На финальном слое с номером N строится единый вектор $Z \in \mathbb{R}^{f_z}$ для изображения:

$$Z_r = \max_{i,j}(z_{i,j}^{(n)})_r$$

- То есть финальный max-слой проверяет наличие шаблона, заданного компонентой вектора где-нибудь в изображении.
- Если решается задача классификации изображений, то вектор Z пропускается через дополнительный линейный классификатор.
- Поскольку свёрточные слои сводятся к операциям с матрицами и векторами, то обучать их параметры можно градиентным спуском.

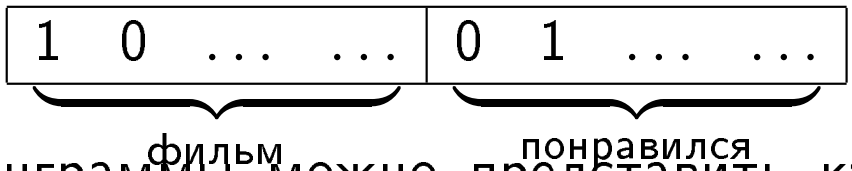
Матричное представление для энграмм

- Контекстные энграммы представляются как конкатенация векторов:



Матричное представление для энграмм

- Контекстные энграммы представляются как конкатенация векторов:

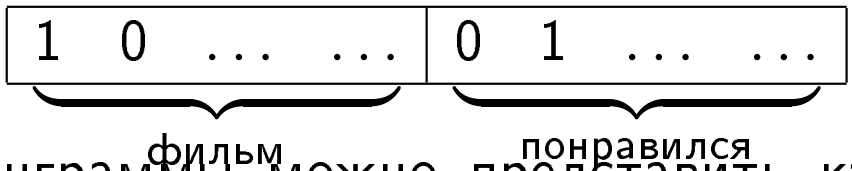


- Извлечение энграммы можно представить как наложение фильтра f:

$$f = [1, 0, \ldots, 0, 1, \ldots]$$

Матричное представление для энграмм

- Контекстные энграммы представляются как конкатенация векторов:



- Извлечение энграммы можно представить как наложение фильтра f:

$$f = [1, 0, \ldots, 0, 1, \ldots]$$

- Более точно, операция $y = \max(\langle f, C \rangle - 1, 0)$ вернёт 1 только для энграммы *фильм понравился*.
- Если задать матрицу F из подобных фильтров, то $y = FC$ переведёт вектор контекста C в вектор энграмм Y.

Свёрточные слои

- Если применять фильтры не к 0 – 1-векторам, а к плотным векторам, то одним фильтром можно “выхыватывать” несколько энграмм.
- Эти энграммы будут похож друг на друга семантически или синтаксически (в зависимости от исходных векторных представлений).
- Однако веса энграмм фильтров по-прежнему настраиваются вручную.

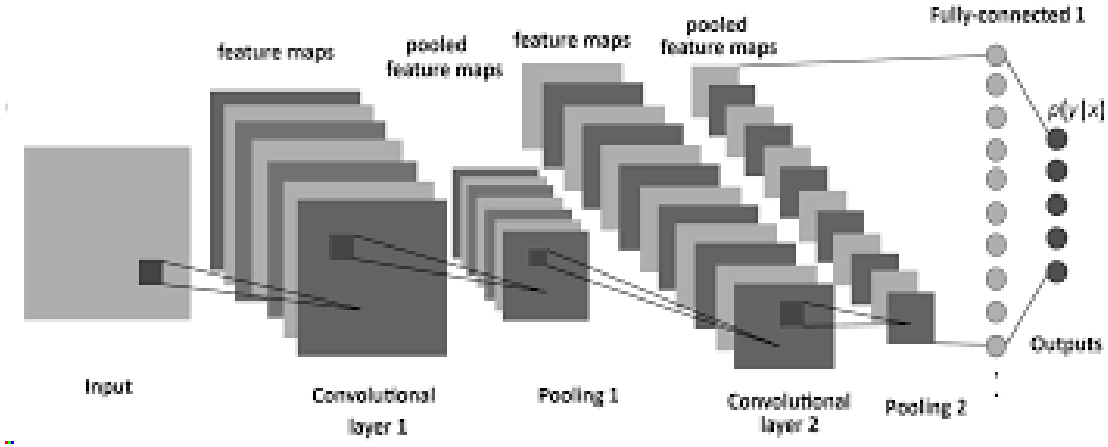
Свёрточные слои

- Если применять фильтры не к $0-1$-векторам, а к плотным векторам, то одним фильтром можно "выхыватывать" несколько энграмм.
- Эти энграммы будут похож друг на друга семантически или синтаксически (в зависимости от исходных векторных представлений).
- Однако веса энграмм фильтров по-прежнему настраиваются вручную.
- Можно определять матрицу C автоматически, обучая сеть с помощью обратного распространения ошибок:

$$\begin{aligned} X &= [x_1, \ldots, x_n] \text{ – матрица, задающая текст,} \\ C_i &= [\![x_{i-k}, \ldots, x_i, \ldots, x_{i+k}]\!] \text{ – вектор контекста в позиции } i \text{ ,} \\ z_i &= FC_i = \sum F_{js}(C_i)_s \text{ – сжатый вектор в позиции } i \end{aligned}$$

Свёрточные слои

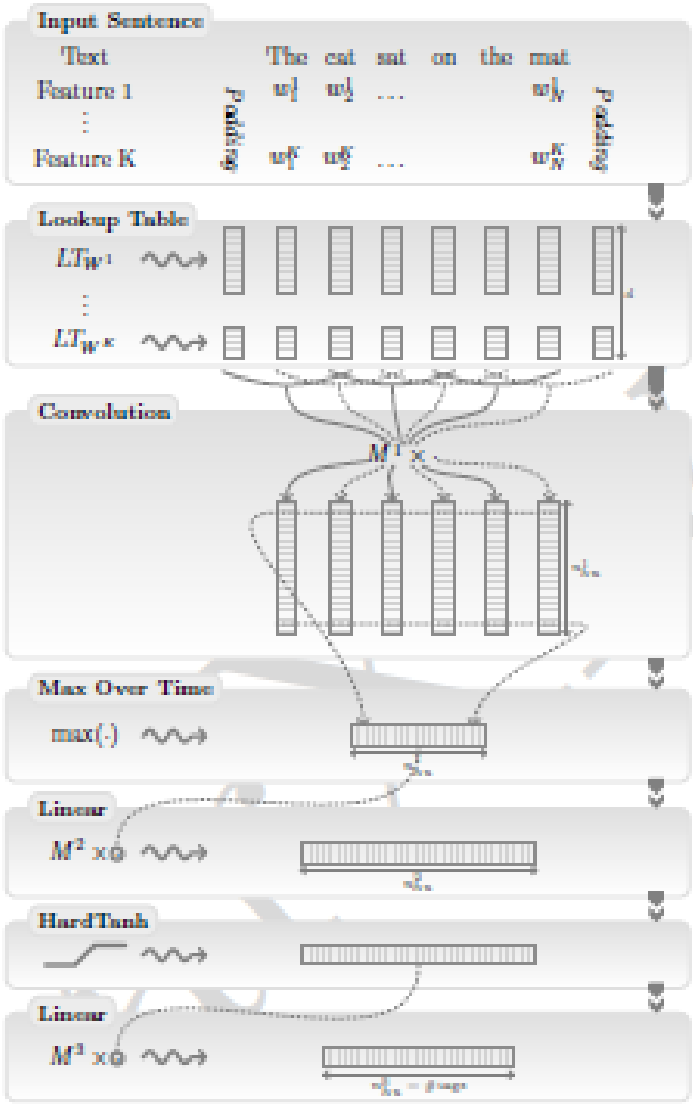
- Можно делать несколько свёрточных слоёв подряд.
- Каждый следующий служит входом предыдущего.



- В конце концов вектора контекстов агрегируются в вектор предложения.

$$\begin{aligned} z &= \max(z_1, \ldots, z_n) \\ p &= \mathrm{softmax}(W * z + b) \end{aligned}$$

Агрегация


Input Sentence
Text
The cat sat on the mat
Feature 1
Feature K
padding
padding
Lookup Table
Convolution
Max Over Time
max(·)
Linear
HardTanh
Linear

Агрегация

- Свёрточный слой собирает информацию из окна ширины m вокруг каждого слова.
- Нужно агрегировать эту информацию для всего предложения.
- Если h_i — представление для позиции i после свёрточного слоя, то каждый из признаков h_{ij} соответствует наличию энграмм определённого вида в окне вокруг данного слова.

Агрегация

- Свёрточный слой собирает информацию из окна ширины m вокруг каждого слова.
- Нужно агрегировать эту информацию для всего предложения.
- Если h_i — представление для позиции i после свёрточного слоя, то каждый из признаков h_{ij} соответствует наличию энграмм определённого вида в окне вокруг данного слова.
- Положим $z_j = \max h_{ij}$.
- То есть высокое значение z_j — наличие энграммы определённого вида где-либо в предложении.

Агрегация

- Свёрточный слой собирает информацию из окна ширины m вокруг каждого слова.
- Нужно агрегировать эту информацию для всего предложения.
- Если h_i — представление для позиции i после свёрточного слоя, то каждый из признаков h_{ij} соответствует наличию энграмм определённого вида в окне вокруг данного слова.
- Положим $z_j = \max h_{ij}$.
- То есть высокое значение z_j — наличие энграммы определённого вида где-либо в предложении.
- После этого получаем вектор признаков $\mathbf{z} = [z_1, \ldots, z_M]$ для всего предложения.
- Финальное распределение вероятностей $\mathbf{p} = [p_1, \ldots, p_n]$:

$$\mathbf{p} = softmax(W^{out} z + W_0^{out})$$

Рекурретные сети

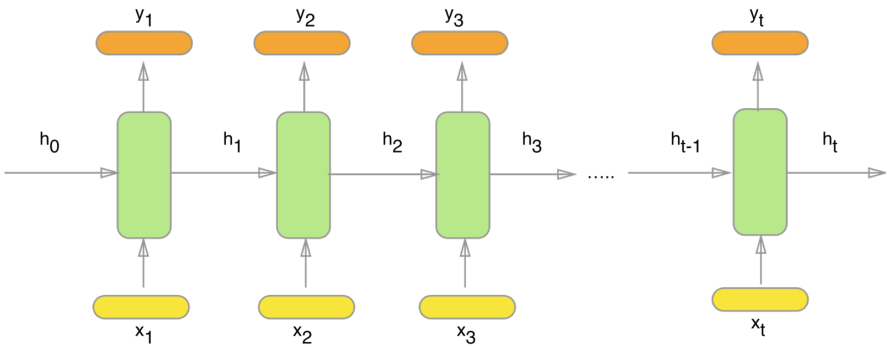
- Свёрточные сети не учитывают порядок внутри последовательности.
- Они не могут отслеживать нелокальные зависимости (только внутри энграмм).

Рекурретные сети

- Свёрточные сети не учитывают порядок внутри последовательности.
- Они не могут отслеживать нелокальные зависимости (только внутри энграмм).
- Можно поддерживать внутреннее состояние модели, обновляя его вместе с каждым новым символом:

$$\begin{aligned} h_t &= h(h_{t-1}, x_t) \\ y_t &= y(h_t) \end{aligned}$$

- h — функция, пересчитывающая скрытое состояние по предыдущему текущему состоянию и текущему входу (энкодер).
- y — функция, вычисляющая выходную метку по скрытому состоянию (декодер).



Рекуррентные сети

- Общая формула пересчёта:

$$\begin{aligned} h_t &= h(h_{t-1}, x_t) \\ y_t &= y(h_t) \end{aligned}$$

Рекуррентные сети

- Общая формула пересчёта:

$$\begin{aligned} h_t &= h(h_{t-1}, x_t) \\ y_t &= y(h_t) \end{aligned}$$

- Простейший вариант для функций h и o — персептрон:

$$\begin{aligned} h(s, x) &= g_1(Us + Vx + b_1), \\ y(s) &= g_2(Ws + b_2), \\ g_1, g_2 &- \text{нелинейности (сигмоида, ReLU)} \end{aligned}$$

- Однако у него есть недостатки.

Неустойчивые градиенты

- Обучение сети происходит градиентным спуском (изменение параметров в направлении антиградиента штрафа).
- Долговременные зависимости устанавливаются за счёт связи h_t с $h_{t-1}, h_{t-2}, \ldots$.

Неустойчивые градиенты

- Обучение сети происходит градиентным спуском (изменение параметров в направлении антиградиента штрафа).
- Долговременные зависимости устанавливаются за счёт связи h_t с $h_{t-1}, h_{t-2}, \ldots$.
- Формально:

$$\frac{\partial h_t}{\partial h_{t-k}} = \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_{t-k+1}}{\partial h_{t-k}},$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial g_1(z)}{\partial z}\Big|_{z = Us + Vx + b_1} U$$

Неустойчивые градиенты

- Обучение сети происходит градиентным спуском (изменение параметров в направлении антиградиента штрафа).
- Долговременные зависимости устанавливаются за счёт связи h_t с $h_{t-1}, h_{t-2}, \ldots$.
- Формально:

$$\frac{\partial h_t}{\partial h_{t-k}} = \frac{\partial h_t}{\partial h_{t-1}} \cdots \frac{\partial h_{t-k+1}}{\partial h_{t-k}},$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial g_1(z)}{\partial z}\Big|_{z = Us + Vx + b_1} U$$

- Если обозначить $A = |\frac{\partial g_1(z)}{\partial z} U|$, то $\frac{\partial h_t}{\partial h_{t-k}} \sim A^k$.
  - $A < 1$ — градиенты затухают к 0.
  - $A > 1$ — градиенты неконтролируемо растут, малое изменение в h_{t-k} ведёт к большим изменениям в h_t.

Варианты рекуррентных сетей

- Назначение более сложных архитектур (Gated Recurrent Unit, Long-Short Term Memory) — сделать $A = 1$.
- GRU (Gated Recurrent Unit):

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_z) \\ h_t &= z_t h_{t-1} + (1 - z_t) tanh(W_h x_t + U_h(r_t h_{t-1}) + b_h) \end{aligned}$$

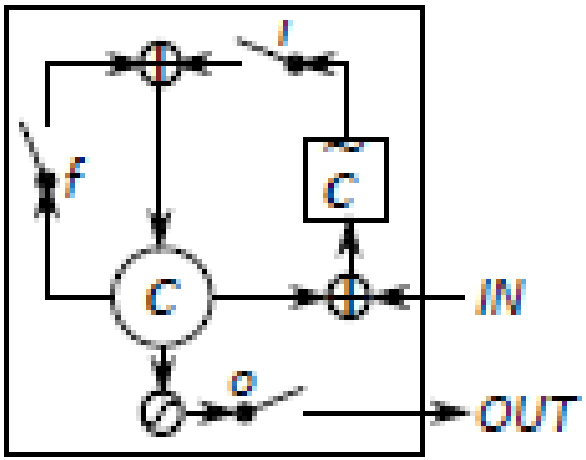
Варианты рекуррентных сетей

- Назначение более сложных архитектур (Gated Recurrent Unit, Long-Short Term Memory) — сделать $A = 1$.
- GRU (Gated Recurrent Unit):

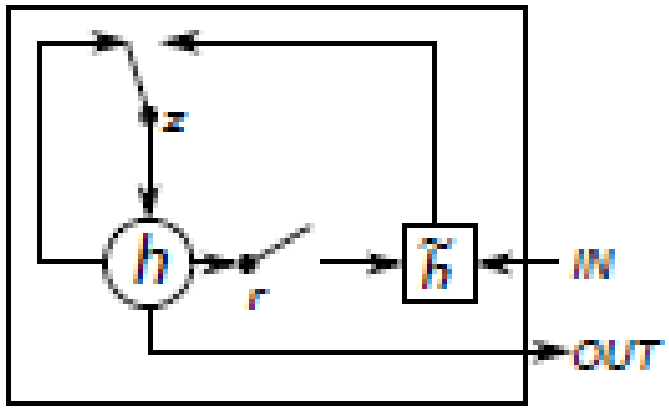
$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_z) \\ h_t &= z_t h_{t-1} + (1 - z_t) tanh(W_h x_t + U_h(r_t h_{t-1}) + b_h) \end{aligned}$$

- На каждом шаге сеть выбирает, как много информации унаследовать из предыдущего состояния, а как много обновить на основе текущего входа.
- r_t позволяет “забыть” предыдущее состояние и перезагрузить сеть.
- z_t балансирует вклад входа и предыдущего состояния.

Рекуррентные архитектуры: иллюстрация



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

Рекуррентные нейронные сети: LSTM

- LSTM (long-short term memory), формула пересчёта:

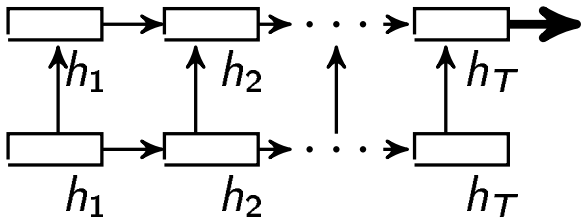
$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t c_{t-1} + i_t tanh(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \sigma_h(c_t)
\end{aligned}
$$

- Смысл параметров:

$$
\begin{aligned}
f_t &— \text{мера вклада предыдущего состояния} \\
i_t &— \text{мера вклада текущего входа} \\
o_t &— \text{мера выходной активации} \\
c_t &— \text{скрытое состояние сети} \\
h_t &— \text{выходной вектор}
\end{aligned}
$$

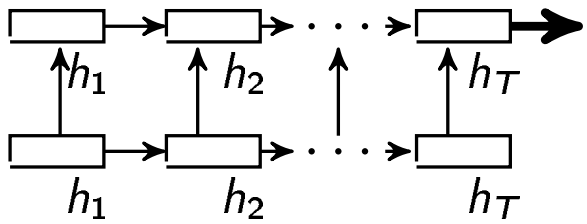
Рекуррентные нейронные сети: использование

- Сжатие информации о последовательности в один вектор h_T (T — длина последовательности):



Рекуррентные нейронные сети: использование

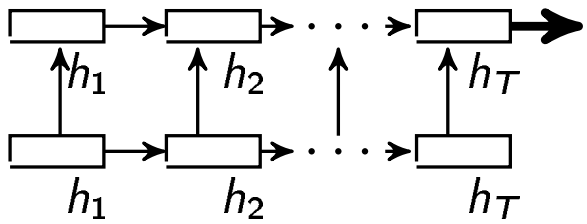
- Сжатие информации о последовательности в один вектор h_T (T — длина последовательности):



- Применение:
  - Все задачи классификации текстов — сеть строит вектор текста из векторов слов (этап агрегации в общей схеме классификации).

Рекуррентные нейронные сети: использование

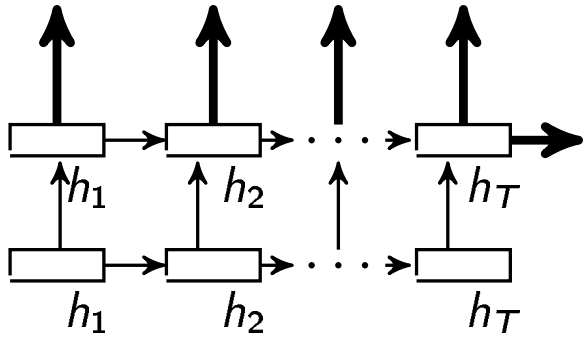
- Сжатие информации о последовательности в один вектор h_T (T — длина последовательности):



- Применение:
  - Все задачи классификации текстов — сеть строит вектор текста из векторов слов (этап агрегации в общей схеме классификации).
  - Вопросно-ответные системы — ответ раскодируется по сжатому вектору вопроса (до 2018г.).
  - Машинный перевод — перевод раскодируется по сжатому исходному тексту (до 2018г.).

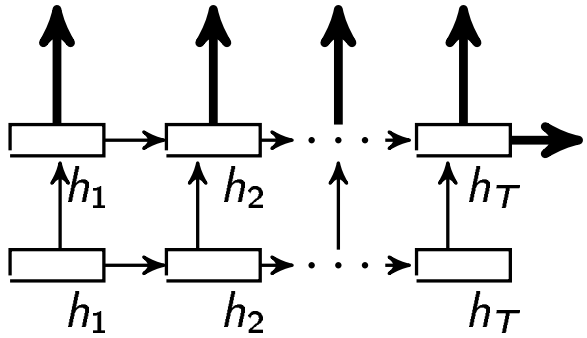
Рекуррентные нейронные сети: использование

- Аккумуляция контекстной информации для каждой позиции.



Рекуррентные нейронные сети: использование

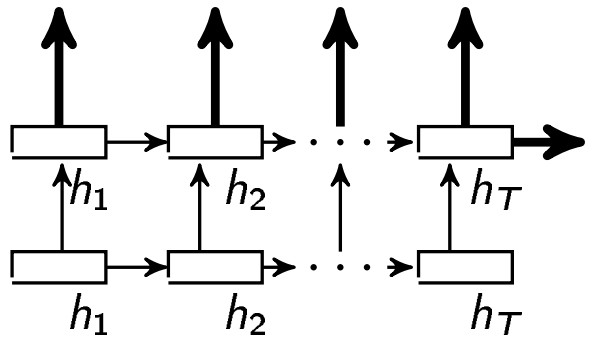
- Аккумуляция контекстной информации для каждой позиции.



- Применение:
  - Построение высокоуровневых представлений слов в контексте (кодируется не только само слово, но и его окружение).

Рекуррентные нейронные сети: использование

- Аккумуляция контекстной информации для каждой позиции.



- Применение:
  - Построение высокоуровневых представлений слов в контексте (кодируется не только само слово, но и его окружение).
  - Задачи простановки меток: морфологический анализ, распознавание именованных сущностей, деление на морфемы...
  - Задача выделения фрагмента текста (автоматический ответ на вопрос к тексту, финальные стадии поиска).