Машинное обучение (Machine Learning)

Attention

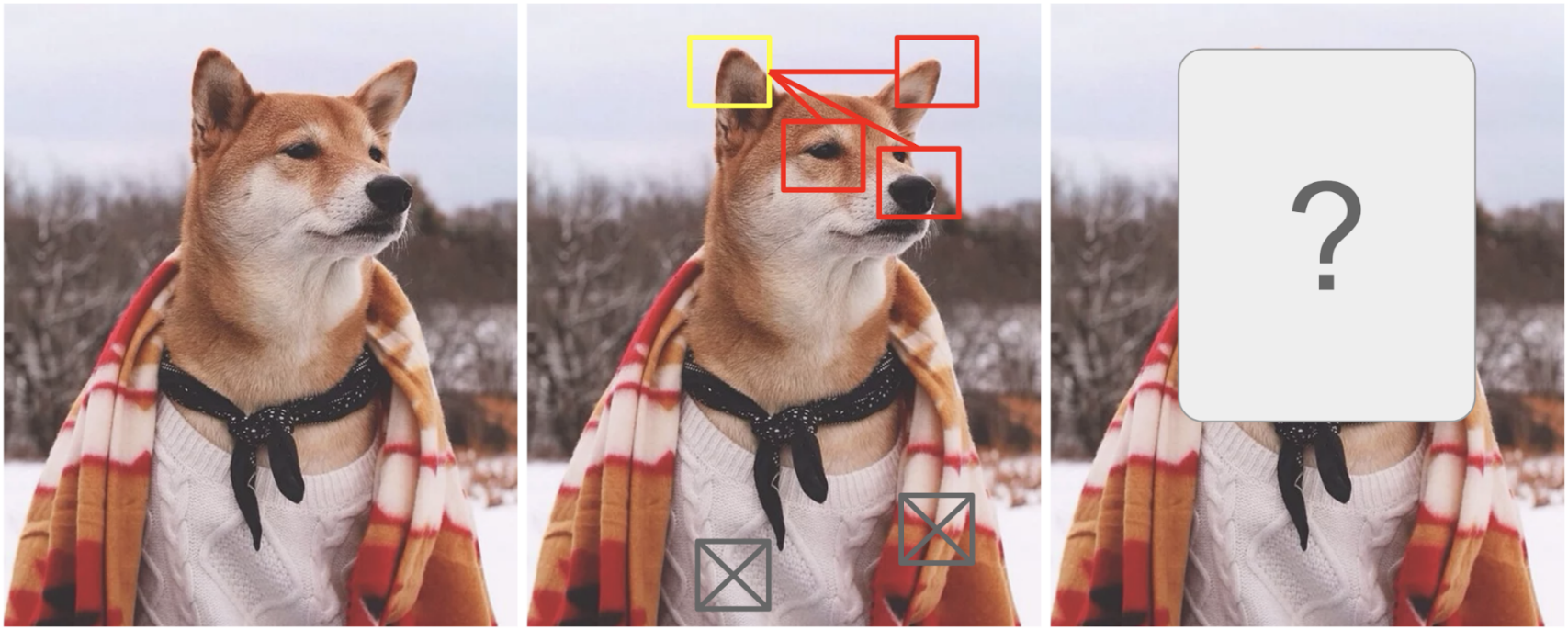
Уткин Л.В.

Санкт-Петербургский политехнический университет Петра Великого

Презентация является компиляцией и заимствованием материалов из замечательного курса и презентации К.В. Воронцова.

Мотивация

- Внимание в некоторой степени мотивируется тем, как мы обращаем визуальное внимание на различные области изображения или сопоставляем слова в одном предложении

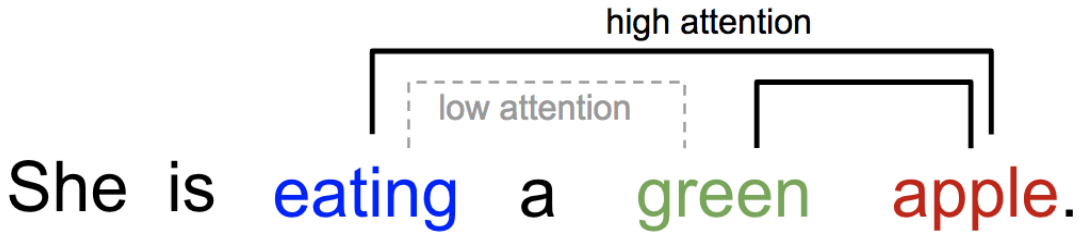


https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Мотивация

- Внимание человека фокусируется на определенной области с “большим разрешением” (взгляд на острое ухо в желтой рамке), воспринимая окружающее изображение в “малым разрешении” (снежный фон и одежда), а затем отрегулирует фокус или сделает вывод соответствующим образом.
- При имеющемся небольшом участке изображения, остальные части дают подсказки, что там должно изображаться. Ожидаем увидеть заостренное ухо в желтой рамке, потому что мы видели нос собаки, другое острое ухо справа и глаза Сибы (в красных рамках).
- Тем не менее, свитер и одеяло в нижней части не будут так полезны, как эти черты собаки.

Мотивация



- Когда мы видим «едят», мы ожидаем, что очень скоро встретимся со словом «еда».
- Цветом выделены слова описывающие еду, но, вероятно, не так, как «еда» напрямую.

Мотивация

- Attention можно интерпретировать как вектор весов важности (attention vector)
- Чтобы предсказать или сделать вывод об одном элементе, таком как пиксель в изображении или слово в предложении, оцениваем, используя этот вектор, насколько сильно он соотносится (коррелирует) с другими элементами

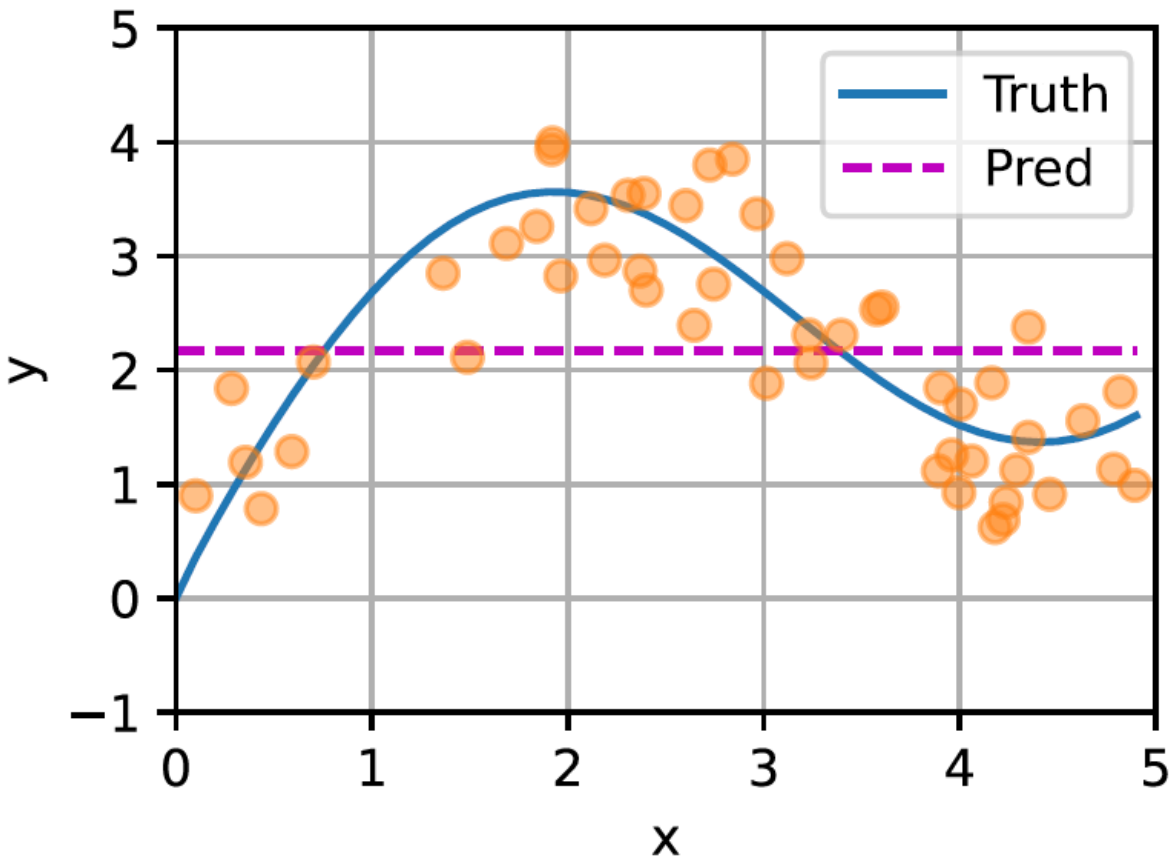
Регрессия Надарая-Уотсона (1)

- Обучающая выборка: n примеров $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i = (x_{i1}, ..., x_{im}) \in \mathbb{R}^m$, $y_i \in \mathbb{R}$
- Регрессионная модель $f : \mathbb{R}^m \to \mathbb{R}$, прогнозирует предсказание $z = f(\mathbf{x})$ для нового примера $\mathbf{x}$
- Простая оценка:

$$z = f(\mathbf{x}) = \sum_{i=1}^{n} y_i$$

Регрессия Надарая-Уотсона (2)

$$z = f(\mathbf{x}) = \sum_{i=1}^{n} y_i$$



Регрессия Надарая-Уотсона (3)

$$z = f(\mathbf{x}) = \sum_{i=1}^{n} \alpha(\mathbf{x}, \mathbf{x}_i) y_i$$

- $\alpha(\mathbf{x}, \mathbf{x}_i)$ - вес внимания (attention weight) характеризует насколько пример $\mathbf{x}_i$ близок к $\mathbf{x}$ (по расстоянию)
- используем ядро $K(\mathbf{x}, \mathbf{x}_i)$ для оценки близости

$$\alpha(\mathbf{x}, \mathbf{x}_i) = \frac{K(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{n} K(\mathbf{x}, \mathbf{x}_j)}$$

- Гауссово ядро

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\left\|\mathbf{x} - \mathbf{x}_i\right\|^2 / 2\sigma\right)$$

Регрессия Надарая-Уотсона (4)

- Гауссово ядро $K(\mathbf{x}, \mathbf{x}_i) = \exp\left(\|\mathbf{x} - \mathbf{x}_i\|^2 / 2\sigma\right)$
- Вес внимания

$$\alpha(\mathbf{x}, \mathbf{x}_i) = \frac{\exp\left(-\|\mathbf{x} - \mathbf{x}_i\|^2 / 2\sigma\right)}{\sum_{j=1}^{n} \exp\left(-\|\mathbf{x} - \mathbf{x}_j\|^2 / 2\sigma\right)}$$
$$= \text{softmax}\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma}\right)$$

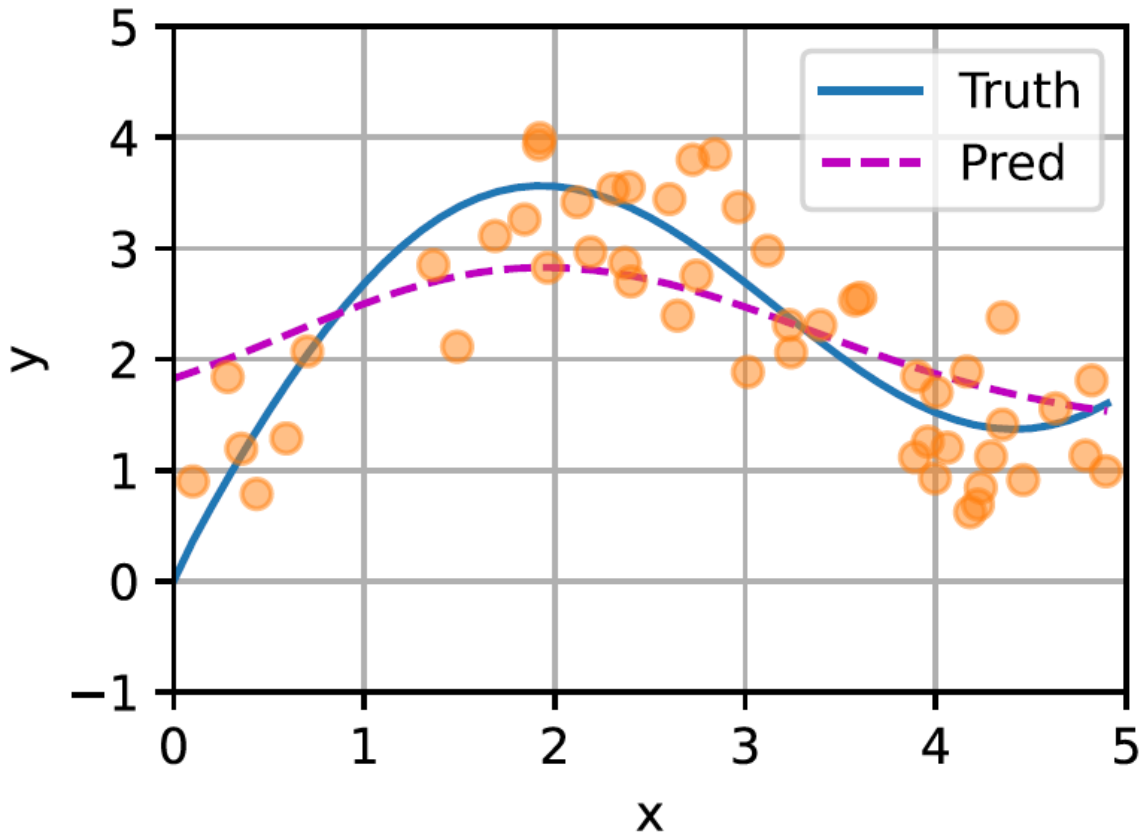
Регрессия Надарая-Уотсона (5)

- Регрессия

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha(\mathbf{x}, \mathbf{x}_i) y_i = \sum_{i=1}^{n} \text{softmax}\left(\frac{-\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma}\right) y_i$$

- $\mathbf{x}$ - *query*, $\mathbf{x}_i$ - *keys*, y_i - *values*
- Если σ задано, то получаем непараметрическую модель внимания

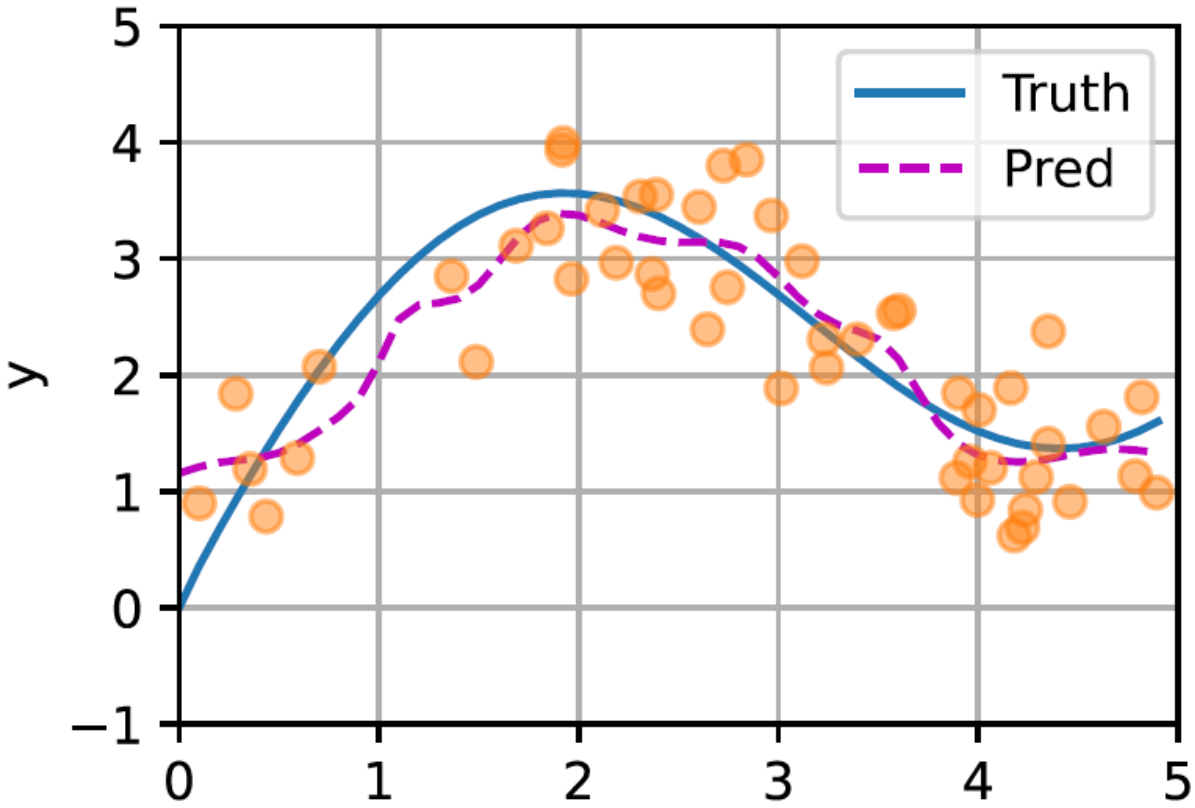
Регрессия Надарая-Уотсона (6)



Регрессия Надарая-Уотсона (7)

- Пусть $\sigma = 1/(2w)$ - параметр обучения

$$f(\mathbf{x}) = \sum_{i=1}^{n} \text{softmax}\left(-\|\mathbf{x} - \mathbf{x}_i\|^2 w\right) y_i$$

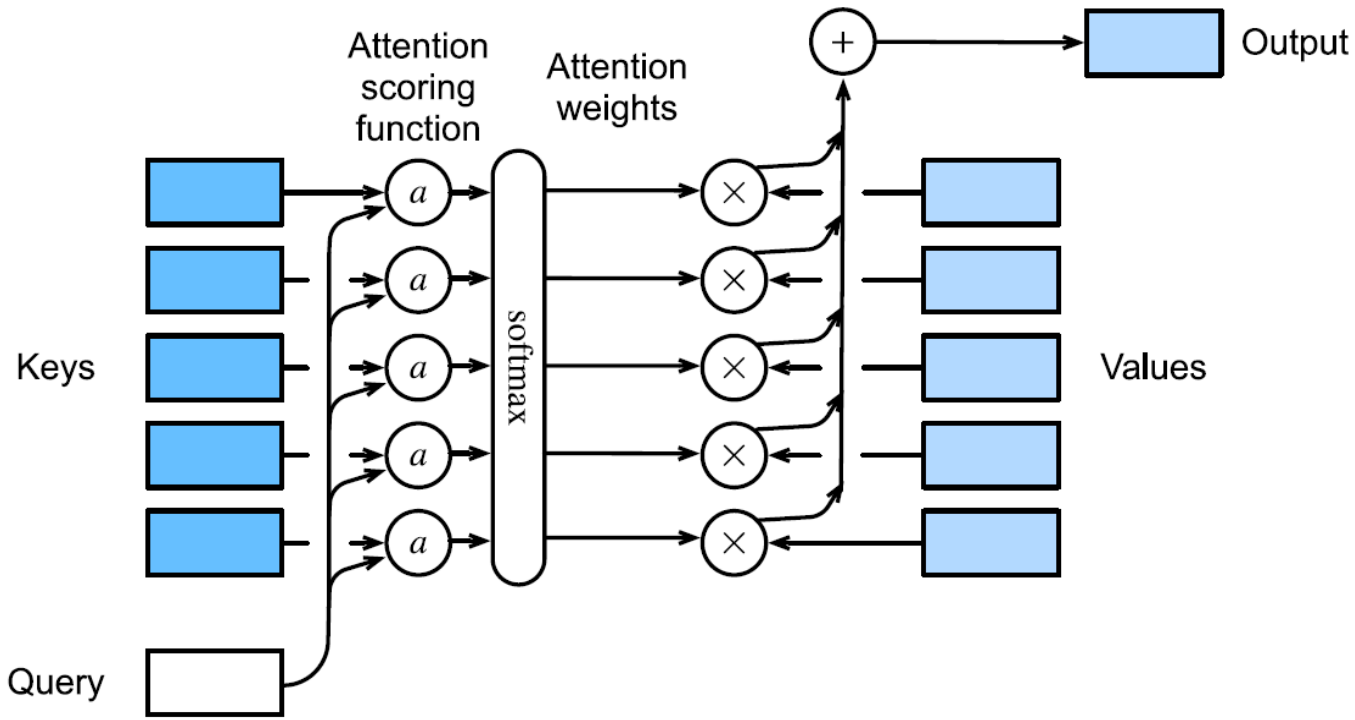


Регрессия Надарая-Уотсона (8)

- Ядерная регрессия Надарая-Уотсона — пример машинного обучения с механизмом внимания.
- Модель внимания в соответствии с регрессией Надарая-Ватсона - средневзвешенное значение меток обучающей выборки.
- Вес внимания присваивается примеру (value, y_i) на основе запроса (query, $\mathbf{x}$) и ключа (key, $\mathbf{x}_i$), связанного с примером.
- Модель внимания может быть непараметрической или параметрической.

Модель внимания в более общем виде

- $\exp\left(-\left\|\mathbf{x}-\mathbf{x}_i\right\|^2/2\sigma\right)$ - скоринговая функция внимания (функция оценки)



Модель внимания в более общем виде

- Имеются вектор query $\mathbf{q} \in \mathbb{R}^q$, пары векторов key-value $(\mathbf{k}_1, \mathbf{v}_1), ..., (\mathbf{k}_n, \mathbf{v}_n)$, $\mathbf{k}_i \in \mathbb{R}^k$, $\mathbf{v}_i \in \mathbb{R}^v$
- Модель внимания (пулинг внимания) f:

$$f\left(\mathbf{q}, (\mathbf{k}_1, \mathbf{v}_1), ..., (\mathbf{k}_n, \mathbf{v}_n)\right) = \sum_{i=1}^{n} \alpha(\mathbf{q}, \mathbf{k}_i)\mathbf{v}_i \in \mathbb{R}^v$$

где

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \text{softmax}\left(a(\mathbf{q}, \mathbf{k}_i)\right) = \frac{\exp\left(a(\mathbf{q}, \mathbf{k}_i)\right)}{\sum_{j=1}^{n} \exp\left(a(\mathbf{q}, \mathbf{k}_j)\right)} \in \mathbb{R}$$

- Выбор скоринговой функции a определяет вид модели внимания.

Additive attention

- Скоринговая функция a:

$$\alpha(\mathbf{q}, \mathbf{k}) = \mathbf{w}_v^T \tanh\left(\mathbf{W}_q \mathbf{q} + \mathbf{W}_k \mathbf{k}\right) \in \mathbb{R}$$

  где $\mathbf{W}_q \in \mathbb{R}^{h \times q}$, $\mathbf{W}_k \in \mathbb{R}^{h \times k}$, $\mathbf{w}_v \in \mathbb{R}^h$ - параметры обучения

- запрос и ключ (query и key) конкатенируются и передаются в нейронную сеть с одним скрытым слоем, количество скрытых нейронов которого равно гиперпараметру h.

Scaled Dot-Product attention (1)

- Более эффективная с вычислительной точки зрения скоринговая функция - скалярное произведение.
- Но операция скалярного произведения требует, чтобы и запрос, и ключ имели одинаковую длину вектора, скажем, d.
- Предположим, что все элементы запроса и ключа являются независимыми случайными величинами с нулевым средним и единичной дисперсией, тогда скалярное произведение обоих векторов имеет нулевое среднее значение и дисперсию d.

Scaled Dot-Product attention (2)

- Чтобы гарантировать, что дисперсия скалярного произведения по-прежнему остается единицей независимо от длины вектора, масштабированная скоринговая функция имеет вид:

$$\alpha(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T \mathbf{k} / \sqrt{d}$$

- Scaled Dot-Product attention для n запросов:

$$\text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V} \in \mathbb{R}^{n \times n},\ \mathbf{Q} \in \mathbb{R}^{n \times d},\ \mathbf{K} \in \mathbb{R}^{m \times d},\ \mathbf{V} \in \mathbb{R}^{m \times v}$$

Модели внимания

- Можно вычислить результат модели внимания как средневзвешенное значение, где разные варианты скоринговых функций приводят к разному поведению модели внимания.
- Когда запросы и ключи являются векторами разной длины, можно использовать функцию Additive attention.
- Когда запросы и ключи одинаковы, масштабированная скоринговая функция скалярного произведения более эффективна в вычислительном отношении.

Модель внимания для временного ряда

- Модель внимания объединяет временные характеристики с использованием динамически генерируемых весов, позволяя напрямую фокусироваться на важных временных моментах времени в прошлом, даже если они очень далеки в скользящем окне.
- Модель внимания имеет вид:

$$\tilde{\mathbf{h}}_t = \sum_{\tau=0}^{k} \alpha(t, \tau)\, \mathbf{h}_{t-\tau}$$

где $\mathbf{h}_{t-\tau}$ - промежуточный вектор признаков, $\alpha(t, \tau) \in [0, 1]$ - вес внимания для $t - \tau$ момента времени, $\tilde{\mathbf{h}}_t$ - выходной вектор

Self-attention

- Дана последовательность токенов $\mathbf{x}_1, \ldots, \mathbf{x}_n$, где $\mathbf{x}_i \in \mathbb{R}^d$. Ее self-attention - последовательность такой же длины $\mathbf{y}_1, \ldots, \mathbf{y}_n$, где

$$\mathbf{y}_i = f(\mathbf{x}_i, (\mathbf{x}_i, \mathbf{x}_i), \ldots, (\mathbf{x}_n, \mathbf{x}_n)) \in \mathbb{R}^d$$

- Self-attention как non-local means denoising: $\sum_{i=1}^n \alpha(\mathbf{x}, \mathbf{x}_i) y_i = \sum_{i=1}^n \text{softmax}\left(\frac{-\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma}\right) y_i$

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha(\mathbf{y}, \mathbf{y}_i) y_i = \sum_{i=1}^n \frac{\exp\left(\frac{-\|\mathbf{y}-\mathbf{y}_i\|^2}{2\sigma}\right) y_i}{\sum_{j=1}^n \exp\left(\frac{-\|\mathbf{y}-\mathbf{y}_j\|^2}{2\sigma}\right)}$$

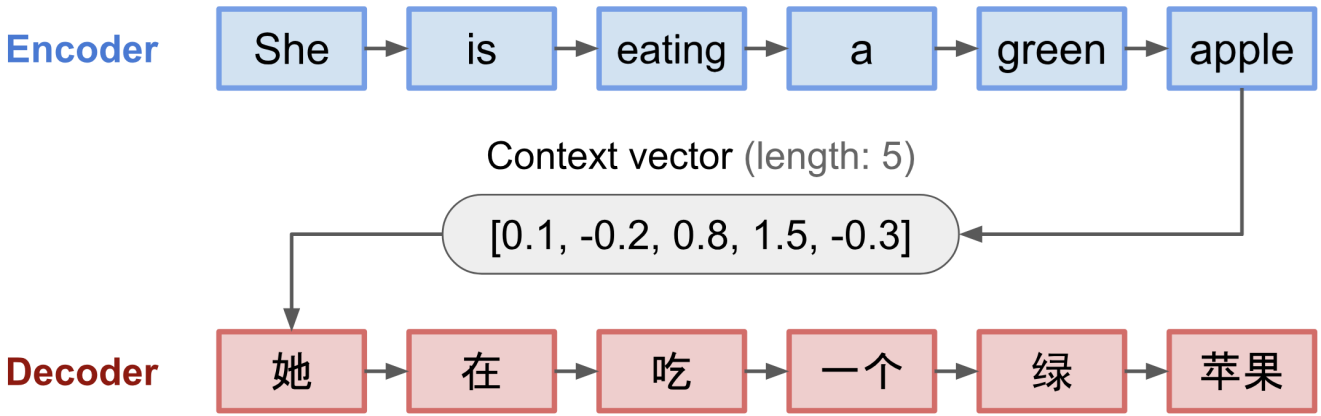
$$= \sum_{i=1}^n \text{softmax}\left(\frac{-\|\mathbf{y}-\mathbf{y}_i\|^2}{2\sigma}\right) y_i$$

- $\mathbf{q}_i = \mathbf{k}_i = \mathbf{v}_i = \mathbf{y}_i$: query=keys=values

Что не так с моделью Seq2Seq

- Модель seq2seq разработана для NLP
- seq2seq преобразует входную последовательность (источник) в новую (цель), и обе последовательности могут иметь произвольную длину
- Примеры - машинный перевод, генерация диалоговых вопросов и ответов и т.д.
- seq2seq имеет архитектуру кодер-декодер:
  - Кодер обрабатывает входную последовательность и сжимает информацию в контекстный вектор (embedding) фиксированной длины
  - Декодер инициализируется контекстным вектором, чтобы выдать преобразованный вывод
- Кодер и декодер являются РНН, использует LSTM

Что не так с моделью Seq2Seq



- Недостаток - невозможность запоминания длинных предложений, он забывает первую часть, когда завершает обработку всего ввода
- Attention пытается решить эту проблему

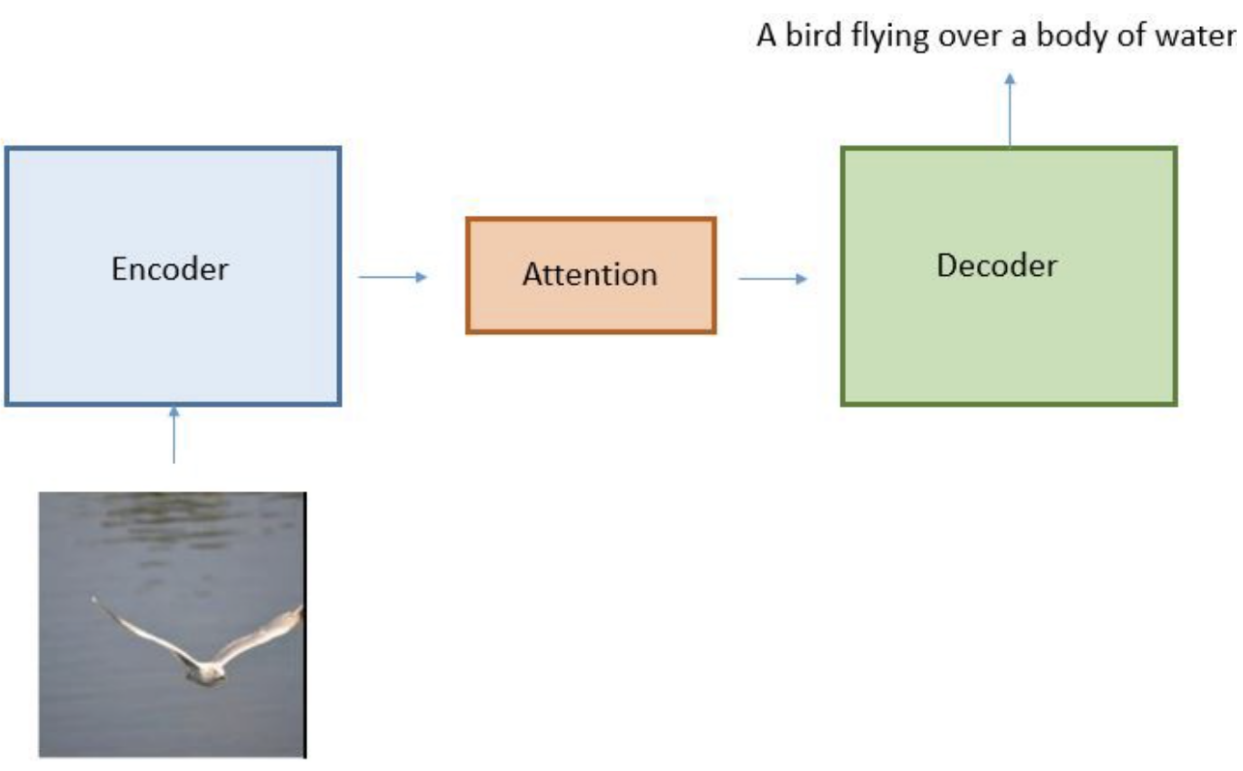
Зачем Attention

- Attention был создан, чтобы помочь запомнить длинные исходные предложения в машинном переводе (NMT)
- Вместо создания одного вектора контекста из последнего скрытого состояния кодера, идея - создание shortcuts между вектором контекста и всем исходным вводом
- Вес этих shortcuts настраивается для каждого элемента вывода

Attention для перевода

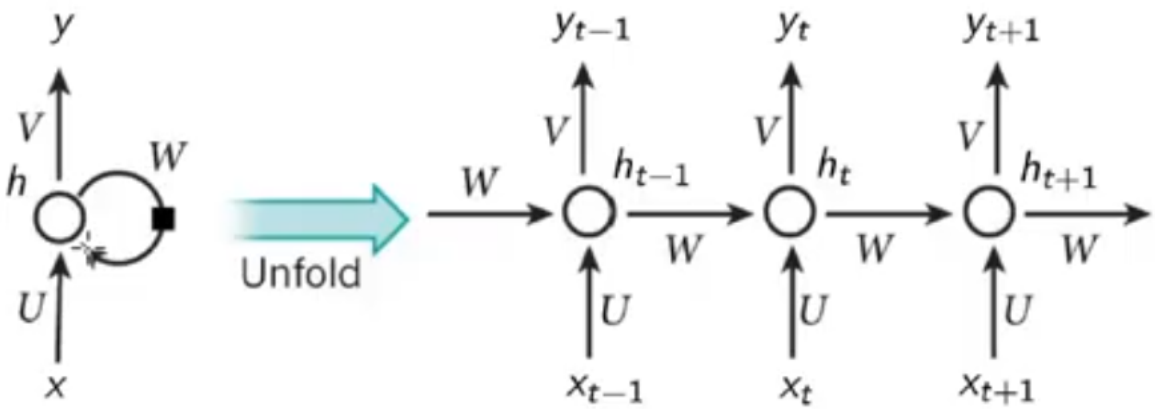
- Хотя контекстный вектор имеет доступ ко всей входной последовательности, необходимо беспокоиться о забывании
- Выравнивание между источником и целью обучается и контролируется контекстным вектором
- Контекстный вектор использует три фрагмента информации:
  1. кодер скрытых состояний
  2. декодер скрытых состояний
  3. выравнивание между источником и целью

Attention для почти перевода



RNN

- x_t - входной веектор в момент $t = 1, ..., T$
- h_t - веектор скрытого состояния в момент t
- y_t - входной веектор
- $h_t = \sigma_h(Ux_t + Wh_{t-1})$; $y_t = \sigma_y(Vh_t)$



RNN

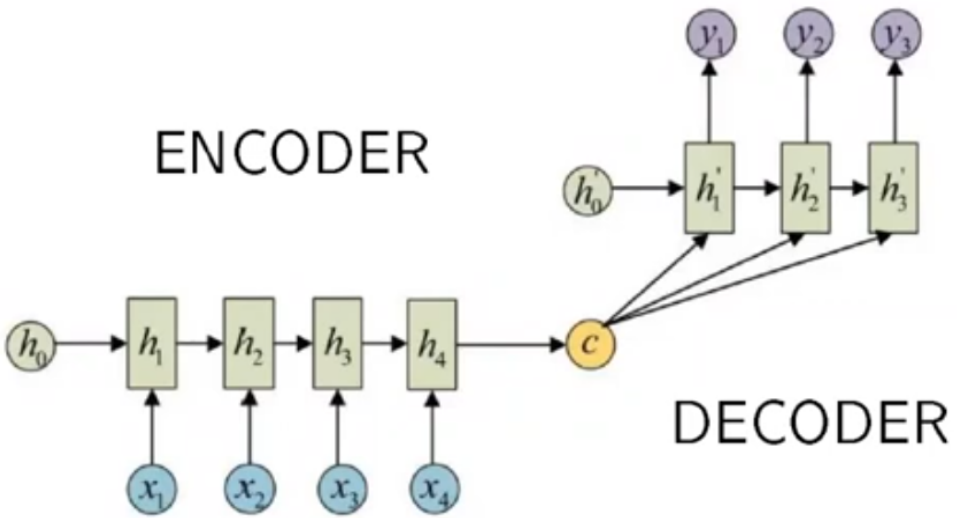
- x_t - входной веектор в момент $t = 1, ..., T$
- h_t - веектор скрытого состояния в момент t
- y_t - входной веектор
- $h_t = \sigma_h(Ux_t + Wh_{t-1})$; $y_t = \sigma_y(Vh_t)$

Обучение RNN: $\sum_{t=0}^{T} L_t(U, V, W) \to \min_{U,V,W}$

- длины входного и выходного сигнала должны совпадать, невозможно загадывание вперед

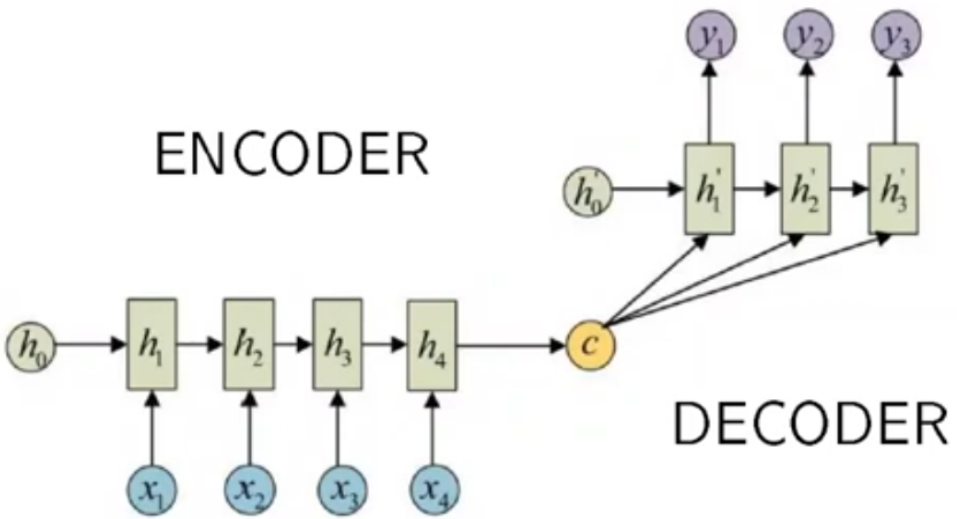
RNN: Seq2Seq (более формально)

$X = (x_1, \ldots, x_n)$ - входная последовательность
$Y = (y_1, \ldots, y_m)$ - выходная последовательность
$c \equiv h_n$ кодирует всю информацию про X для синтеза Y
$h_i = f_{\text{in}}(x_i, h_{i-1})$; $h'_t = f_{\text{out}}(h'_{t-1}, y_{t-1}, c)$; $y_t = f_{\text{y}}(h'_t, y_{t-1})$



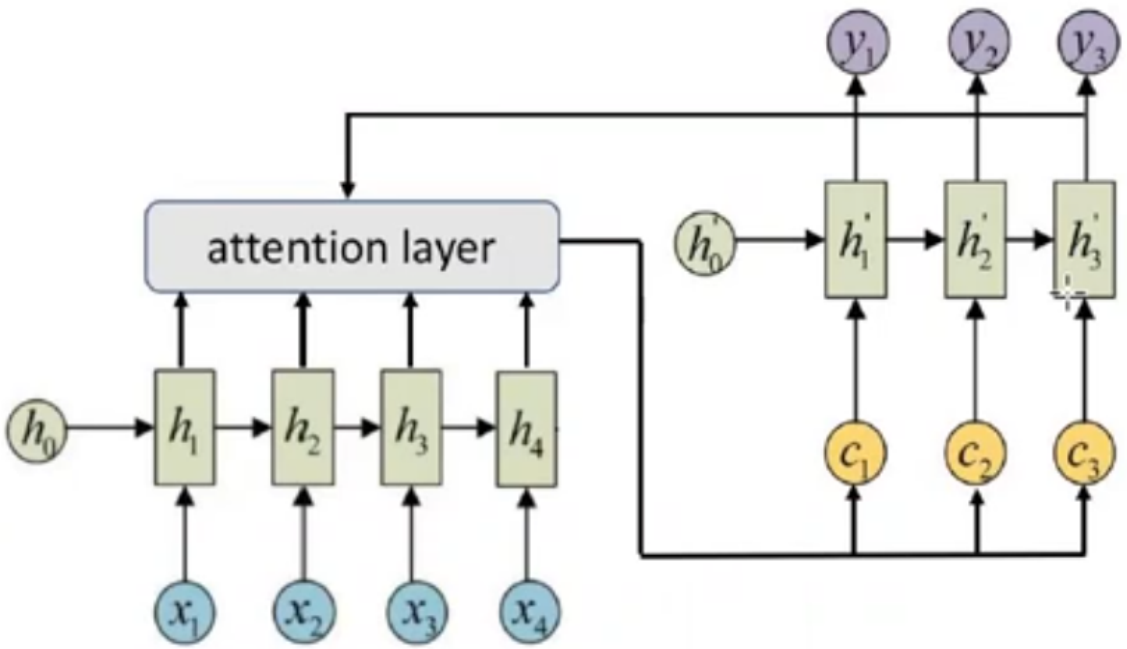
RNN: Seq2Seq (более формально)

- h_n лучше помнит конец последовательности, чем начало
- чем больше n, тем труднее упаковать информацию в c
- придется контролировать затухание градиента
- RNN трудно распараллеливать



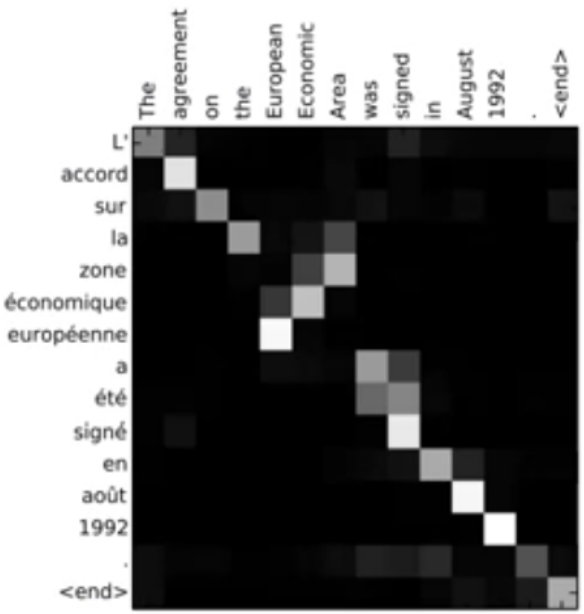
Attention и RNN

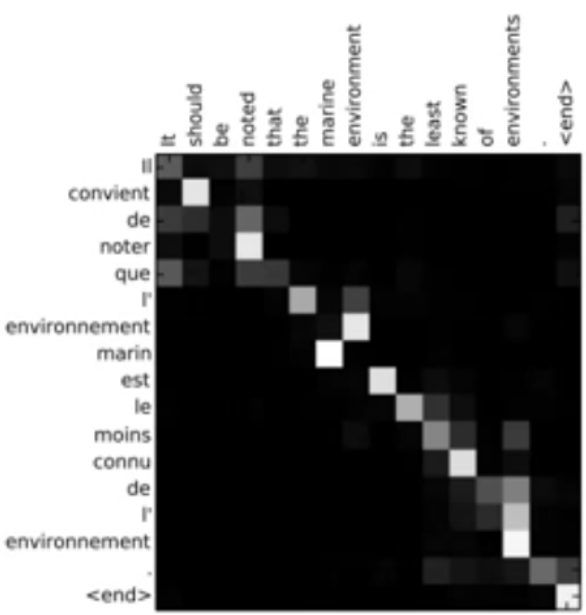
- можно отказаться от рекуррентности как по h_i, так и по h'_t
- можно вводить обучаемые параметры в a и c

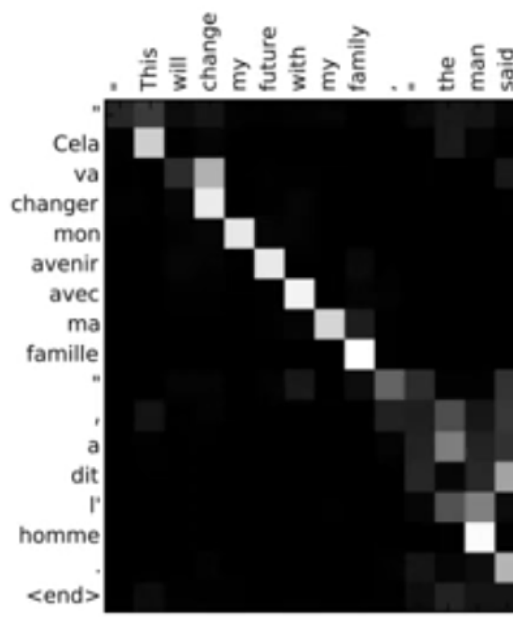


Attention - интерпретируемость

При обработке конкретной последовательности X визуализация матрицы α_{ti} показывает, на какие слова x_i модель обращает внимание, генерируя слово перевода y_t





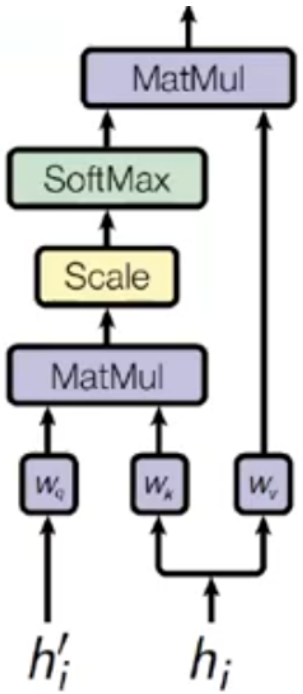


Функции сходства

- $a(h, h') = h^{\mathrm{T}} h'$ - скалярное произведение
- $a(h, h') = \exp\left(h^{\mathrm{T}} h'\right)$ - norm превращается в SoftMax
- $a(h, h') = h^{\mathrm{T}} W h'$ - с матрицей обучаемых параметров W
- $a(h, h') = w^{\mathrm{T}} th\left(Uh + Vh'\right)$ - аддитивное внимание с w, U, V

Функции сходства

Линейное преобразование векторов query, key, value:
$a(h_i, h'_{t-1}) = (W_k h_i)^T (W_q h'_{t-1}) / \sqrt{d}$;
$\alpha_{ti} = \mathrm{SoftMax}_i a(h_i, h'_{t-1})$; $c_t = \sum_i \alpha_{ti} W_v h_i$
$W_{q,d\times\dim(h')}$, $W_{k,d\times\dim(h)}$, $W_{v,d\times\dim(h)}$ - матрицы весов линейных нейронов (обучаемые линейные преобразования в пространство размерностм d); часто $W_k = W_v$для простоты



Attention - веса внимания - 2

- q - вектор-запрос, для которого вычисляется контекст
- $K = (k_1, ..., k_n)$ - векторы-ключи, сравниваемые с запросом
- $V = (v_i, ..., v_n)$ - векторы-значения, образующие контекст
- $a(k_i, q)$ - оценка сходства ключа k_i запросу q
- c - искомый вектор контекста, релевантный запросу

Attention - веса внимания -2

- *Модель внимания* - это 3х-слойная сеть, вычисляющая выпуклую комбинацию значений v_i, релевантных запросц q

$$c = \mathrm{Attn}(q, K, V) = \sum_i v_i \mathrm{SoftMax}_i a(k_i, q)$$

$c_t = \mathrm{Attn}(W_q h'_{t-1}, W_k H, W_v H)$ - пример с пред. слайда, $H = (h_1, ..., h_n)$ - входные векторы, h'_{t-1} - выходной

- *Самовнимание* (self-attention):
$c_t = \mathrm{Attn}(W_q h_i, W_k H, W_v H)$ - частный случай, когда $h_i \in H$

Многомерное внимание (multi-head attention)

Идея: J разных моделей внимания совместно обучаются выделять различные аспекты информации (части речи, синтаксис, и т.д.):

$$c^j = \mathrm{Attn}(W_q^j q, W_k^j H, W_v^j H),\ j = 1, ..., J$$

Варианты агрегирования выходного вектора:

- $c = J^{-1} \sum_{j=1}^{J} c^j$ - усреднение
- $c = [c^1, ..., c^J]$ - конкатенация
- $c = [c^1, ..., c^J] W$ - возврат к нужной разм-ти

  Регуляризация: чтобы аспекты внимания были максимально различны, строки $J \times n$ матриц A, $\alpha_{ji} = \mathrm{SoftMax}_i a(W_k^j h_i, W_q^j q)$ декоррелируются $(\alpha_j^{\mathrm{T}} \alpha_j \to 1)$:

$$\left\| AA^{\mathrm{T}} - I \right\|^2 \to \min_{\{W_k^j, W_q^j\}}$$

Иерархическое внимание (hierarchical attention - 1)

Вложенная структура: слова $\in$ предложения $\in$ документы

x_{it} - слова $t = 1, ..., T_i$ в предложениях $i = 1, ..., L$

Сеть первого (нижнего) уровня, обучение эмбедингов s_i:

$h_{it} = BidirGRU(W_0 x_{it})$ - GRU для векторизации слов

$h_{it} = th(W_1 h_{it} + b_1)$ - обучаемое преобразование Key

$s_i = \sum_t h_{it} \mathrm{SoftMax}_t(u_{it}^T q_1)$ - эмбединг предложения, Query q_1

Иерархическое внимание (hierarchical attention - 2)

Сеть второго (верхнего) уровня, обучение эмбедингов v:
$h_i = BidirGRU(s_i)$ - GRU для векторизации предложений
$u_i = th(W_2 h_i + b_2)$ - обучаемое преобразование Key
$v = \sum_i h_i \mathrm{SoftMax}_t(u_i^T q_2)$ - эмбединг предложения, Query q_2
Максимизация правдоподобия для классификации документов:
$\sum_d \sum_y \ln\left(\mathrm{SoftMax}_y(W_y v + b_y)\right) \to \max$

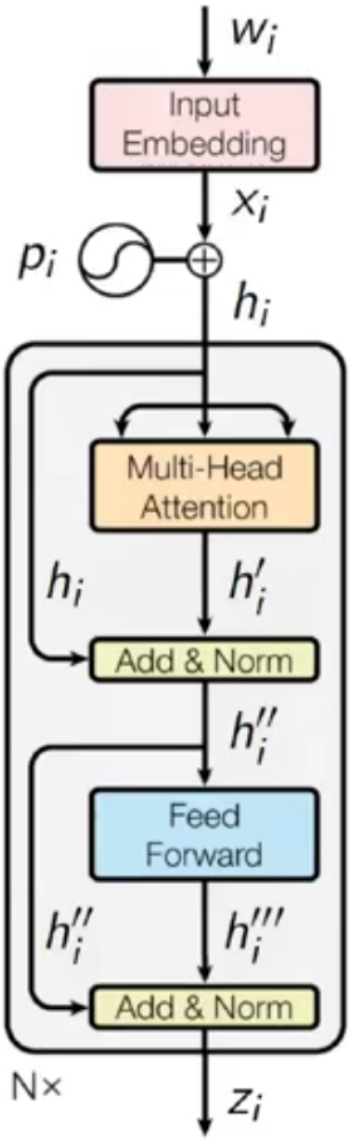
Трансформер для машинного перевода

Трансформер - это нейросетевая архитектура на основе моделей внимания и полносвязанных слоев, без RNN

Схема преобразований данных в машинном переводе:

- $S = (w_1, ..., w_n)$- слова предложения на входном языке
  $\downarrow$ *обучаемая или предобученная векторизация слов*
- $X = (x_1, ..., x_n)$- эмбединги слов входного предложения
  $\downarrow$ *трансформер-кодировщик*
- $Z = (z_1, ..., z_n)$- контекстные эмбединги слов
  $\downarrow$ *трансформер-декодировщик*
- $Y = (y_1, ..., y_m)$- эмбединги слов выходного предложения
  $\downarrow$ *генерация слов из построенной языковой модели*
- $\tilde{S} = (\tilde{w}_1, ..., \tilde{w}_m)$- слова предложения на выходном языке

Архитектура трансформера - кодировщика (1)



Архитектура трансформера - кодировщика (2)

1. Добавляются позиционные векторы p_i: $h_i = x_i + p_i$, $H = (h_1, ..., h_n)$
2. Многомерное самовнимание: $h_i^j = \mathrm{Attn}(W_q^j h_i, W_k^j H, W_v^j H)$
3. Конкатенация: $h_i' = \mathrm{MH}_j(h_i^j) \equiv \left[h_i^1, ..., h_i^J\right]$
4. Сквозная связь + нормировка уровня: $h_i'' = \mathrm{LN}(h_i' + h_i; \mu_1, \sigma_1)$
5. Полносвязная 2х-слойная сеть FFN: $h_i''' = W_2 \mathrm{ReLU}(W_1 h_i'' + b_1) + b_2$
6. Сквозная связь + нормировка уровня: $z_i = \mathrm{LN}(h_i''' + h_i''; \mu_2, \sigma_2)$

Замечания - 1

- вычисления параллельны по элементам последовательности $(x_1, ..., x_n) \to (z_1, ..., z_n)$, что было невозможно в RNN
- $N = 6$ блоков $h_i \to \square \to z_i$ соединяются последовательно
- возможно использование предобученных эмбедингов x_i

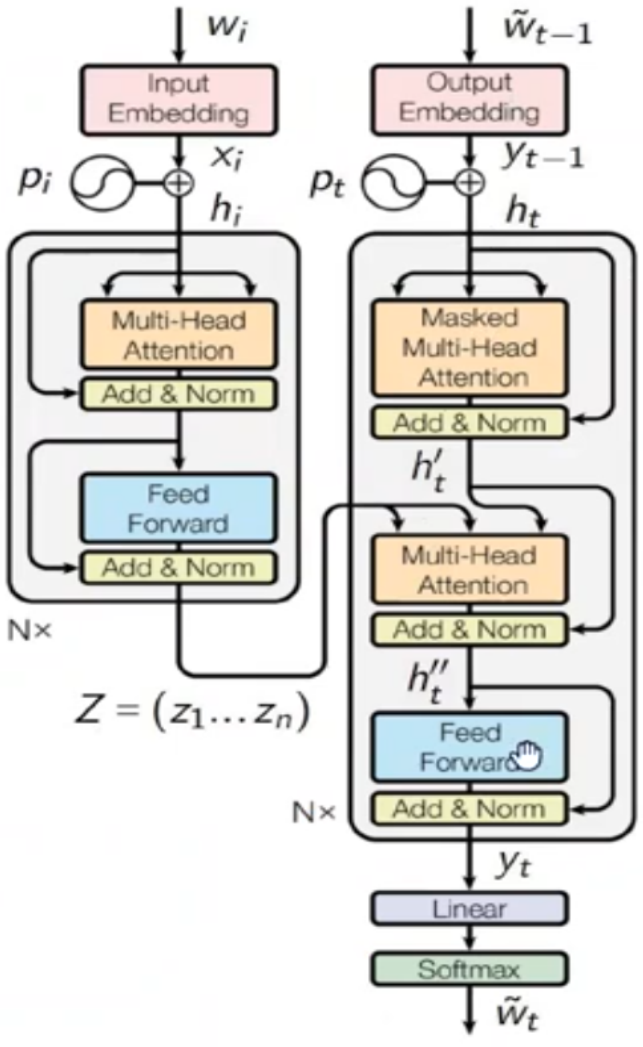
Замечания - 2

- возможно обучение эмбедингов $x_i \in \mathbb{R}^d$ слов $w_i \in V$:
  $x_i = u_{w_i}$ или в матричной записи $X_{d \times n} = U \cdot B_{d \times |V| \times n}$,
  где
  V - словарь слов входных последовательностей,
  U - матрица обучаемых векторных предст-ний слов,
  $b_{vi} = [w_i = v]$ - матрица one-hot кодирования
- нормировка уровня (Layer Normalization), $x, \mu, \sigma \in \mathbb{R}^d$:

$$LN_s(x; \mu, \sigma) = \sigma_s \frac{x_s - \bar{x}}{\sigma_x} + \mu_s, \ \ s = 1, ..., d,$$

$\bar{x} = \frac{1}{d} \sum_s x_s$, $\sigma_x^2 = \frac{1}{d} \sum_s (x_s - \bar{x})^2$ - среднее и дис-я x

Архитектура трансформера - декодировщика (1)



Архитектура трансформера - декодировщика (2)

Авторегрессионный синтез последовательности:
$y_0 = \langle \mathrm{BOS} \rangle$ - эмбединг символа начала

1. Маскирование "данных из будущего" $h_t = y_{t-1} + p_t$, $H_t = (h_1, ..., h_t)$
2. Многомерное самовнимание: $h_i^j = LN \circ MH_j \circ \mathrm{Attn}(W_q^j h_t, W_k^j H_t, W_v^j H_t)$
3. Многомерное внимание на кодировку Z: $h_i'' = LN \circ MH_j \circ \mathrm{Attn}(\tilde{W}_q^j h_t', \tilde{W}_k^j Z, \tilde{W}_v^j Z)$
4. Полносвязная 2х-слойная сеть: $y_t = LN \circ FFN(h_t'')$
5. Линейный предсказательный слой: $p(\tilde{w}|t) = \mathrm{SofMax}(W_v y_t + b_v)$

Генерация $\tilde{w}_t = \arg\max_{\tilde{w}} p(\tilde{w}|t)$ пока $\tilde{w}_t \neq \langle \mathrm{EOS} \rangle$

Критерии обучения и валидации для машинного перевода - 1

Критерий для обучения параметров нейронной сети W по обучающей выборке предложений S с переводом $\tilde{S}$:

$$\sum_{(S,\tilde{S})} \sum_{\tilde{w}_t \in \tilde{S}} \ln p\left(\tilde{w} | t, S, W\right) \to \max_W$$

Критерии обучения и валидации для машинного перевода - 2

Критерии оценивания моделей (недифференцируемые) по выборке пар предложений “перевод S, эталон S_0”:
BiLingual Evaluation Understudy:

$$\text{BLEU} = \min\left(1, \frac{\sum \text{len}(S)}{\sum \text{len}(S_0)}\right) \times \text{mean}_{(S,S_0)}\left(\prod_{n=1}^{4} \frac{\#n\text{-грамм из } S, \text{ входящих в } S_0}{\#n\text{-грамм в } S}\right)^{\frac{1}{4}}$$

Word Error Rate:

$$\text{WER} = \text{mean}_{(S,S_0)}\left(\prod_{n=1}^{4} \frac{\#\text{вставок}+\#\text{удалений}+\#\text{замен}}{\text{len}(S)}\right)$$

BERT (Bidirectional Encoder Representations from Transformer)

Трансформер BERT - это кодировщик без декодировщика, предобучаемый для решения широкого класса задач NLP

Схема преобразований данных:

- $S = (w_1, ..., w_n)$ - токены предложения на входном языке

  $\downarrow$ *обучение эмбедингов вместе с трансформером*
- $X = (x_1, ..., x_n)$ - эмбединги токенов входного предложения

  $\downarrow$ *трансформер кодировщика*
- $Z = (z_1, ..., z_n)$ - трансформированные эмбединги

  $\downarrow$ *дообучение на конкретную задачу*
- Y - выходной текст/разметка/классификация и т.д.

Критерии MLM (masked language modeling) для обучения BERT

Критерий маскированного языкового моделирования MLM строится автоматически по текстам (self-supervised learning)

$$\sum_{S} \sum_{i \in M(S)} \ln p\left(w_i | i, S, W\right) \to \max_{W},$$

где $M(S)$ - подмножество маскированных токенов из S,

$$p\left(w | i, S, W\right) = \mathrm{SoftMax}_{w \in V}(W_z z_i(S, W_T) + b_z)$$

- языковая модель, предсказывющая i-ый токен предложения S; $z_i(S, W_T)$ - контекстный эмбединг i-го токена предложения S на выходе Трансформера с параметрами W_T; W - все параметры Трансформера и языковой модели.

Критерии NSP (next sentence prediction) для обучения BERT

Критерий предсказания связи между предложениями NSP строится автоматически по текстам (self-supervised learning):

$$\sum_{(S,S')} \ln p\left(y_{SS'}|S,S',W\right) \to \max_{W}$$

где $y_{SS'} = [\text{за } S \text{ следует } S']$ - классификация пары предложений,

$$p\left(y_{SS'}|S,S',W\right) = \mathrm{SoftMax}_{y\in\{0,1\}}\left(W_y th\left(W_z z_0(S,S',W_T) + b_s\right)+b\right.$$

- вероятностная модель бинарной классификации пар (S,S'); $z_0(S,S',W_T)$ - контекстный эмбединг токена $\langle CLS\rangle$ для пары предложений, записанной в виде $\langle CLS\rangle$ S $\langle SEP\rangle$ S' $\langle SEP\rangle$

Замечания по трансформерам

- **Fine-tuning**: для дообучения на задаче задается модель $f(Z(S, W_T), W_f)$, выборка $\{S\}$ и критерий $L(S, f) \to \max$
- **Multi-task learning**: для дообучения на наборе задач $\{t\}$ задаются модели $f(Z(S, W_T), W_t)$, выборки $\{S\}_t$ и сумма критериев $\sum_t \lambda_t \sum_S L_t(S, f_t) \to \max$
- GLUE, SuperGLUE, Russian SuperGLUE - наборы текстовых задач на понимание естественного языка
- Трансформеры обычно строятся не на словах, а на токенах, получаемых BPE (Byte-Pair Encoding) или WordPiece
- $\text{BERT}_{\text{BASE}}$, GPT1: $N = 12, d = 768, J = 8$ весов 65M
- $\text{BERT}_{\text{LARGE}}$: $N = 24, d = 1024, J = 16$ весов 340M

Еще замечения

- Модели внимания сначала встраивались в RNN или CNN, но оказалось, что они самодостаточны
- Модель внимания работаем точнее и быстрее RNN
- Легко предобучается и используется для многих задач
- Легко обобщается на тексты, графы, изображения
- Доказано, что модель внимания multi-head self-attention (MHSA) эквивалентна сверточной сети [Cordonnier, 2020 On the relationship between self-attention and convolutional layers]
- Модель внимания лежит в основе трансформера