Конспект лекции

А.Н. Шевляков

Введение в нейронные сети

Обработка текстов

# превращение текста в численный объект

При исследовании текстов нужно превратить текст в численный объект (например, в вектор чисел).

Перед дальнейшей обработкой из текста нужно **удалить всё лишнее и несущественное**: знаки препинания и служебные слова (союзы, предлоги, артикли…). Кроме того, все слова нужно привести к **начальной форме**.

То есть сделать **embedding** текста в пространство векторов.

Существует много способов представить информацию в тексте в виде вектора. Самые простые методы представляют текст в виде длиннющего вектора чисел, причем размерность вектора равна количеству слов в рассматриваемом языке.

# Контекст слова и работа с ним

Откуда исторически возникла проблема анализа контекстов? Это проблема **синонимов!**

В рассмотренных выше методах «оцифровки текста» возникает проблема синонимов. Они считаются разными словами. Например, возникнут два разных столбца, соответствующих словам «жена» и «супруга».

По-хорошему, **синонимы нужно отождествлять**. А как это сделать в автоматическом режиме?

**Контекст** — окрестность слова радиуса R в тексте. То есть 2-контекст слова w — это окрестность радиуса 2 с центром в слове w.

Центр окрестности в некоторых алгоритмах может не включатся в контекст.

Самое простое применение контекста - поиск синонимов.

Пусть A1,...An — весь корпус текстов. Для слов wi, wj и радиуса r можно вычислить величину cr,i,j — число одинаковых r-контекстов слов wi, wj .

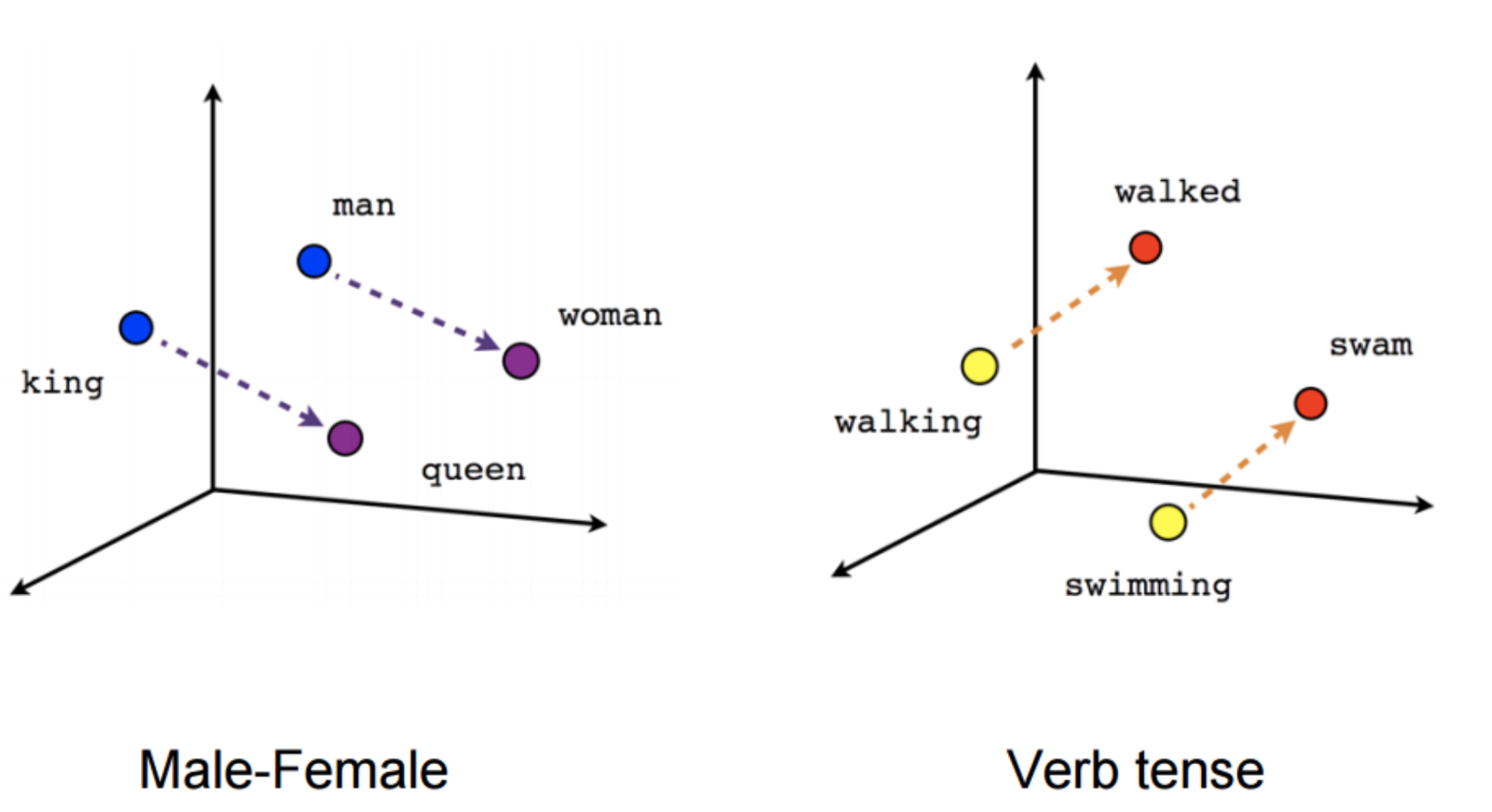
**Смысл**: **у слов-синонимов значение cr,i,j будет очень высоким.** Поэтому можно отождествить все слова, у которых cr,i,j выше определенного порога (порог определяется экспертом-лингвистом).

В этом алгоритме центр контекста мы не включаем в контекст.

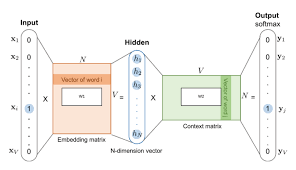
# Технологии типа word2vec

Технологии word2vec - это алгоритмы, которые каждому слову ставят в соответствие числовой вектор со свойством:

1. семантически близкие слова должны отображаться в близкие вектора (относительно некоторой метрики)
2. должна поддерживаться некоторая арифметика над векторами.



**Word2vec с помощью нейросетей (НС).** Здесь будут использоваться НС «матричной» архитектуры. Фактически будет искаться разложение матрицы не «оранжевый» и «зелёный» множитель.



А уж если используются НС, то нужна тренировочная выборка (ТВ). Как её получить?

Нужен **один очень-очень большой текст**. Например, можно объединить **все** статьи Википедии.

Естественно, нужно сделать предобработку текста (удалить служебные слова, знаки препинания, все слова привести к начальной форме):

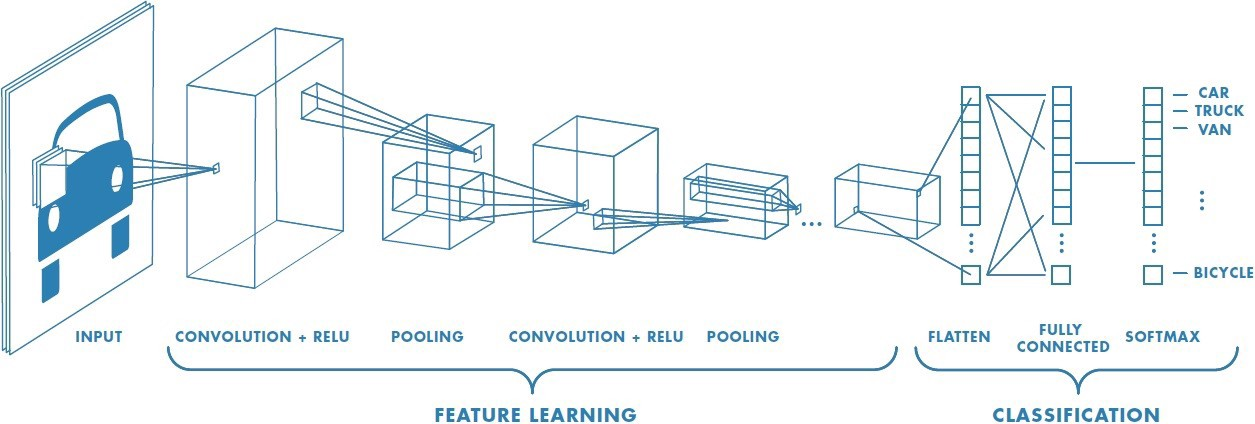
**мести мести весь земля весь предел свеча гореть стол свеча гореть**

Теперь фиксируем ширину (радиус контекста). Пусть он будет равен 1.

Начинаем перебирать все контексты и формируем пары вида: **(центральное слово контекста, слово из контекста).**

Тренировка НС будет заключаться в предсказании Y по Х.

Как следует из ТВ, нейронная сеть на вход получает слова и выдаёт слова. Проще разобраться с **выходным слоем**. По сути наша задача — **многоклассовая задача предсказания**. Тут есть стандартная архитектура: слой softmax и выдача вероятностей принадлежности к классам.



**Размер** выходного слоя совпадает с количеством слов в языке!

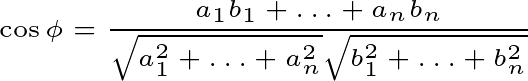
НС для word2vec матричная, то есть данные там представляются матрицами и векторами. Каждому входу (центру контекста) соответствует свой вектор, **вектор слова**. Координаты этого вектора — пока неизвестные веса НС.

**Длина векторов** является архитектурной характеристикой НС, и определяется заранее. Фактически — это размерность embedding-а.

Вторая матрица — это матрица контекстов. Эти вектора будет рисовать вертикально. Понятно, что ширина матрицы с прошлого слайда должна равняться высоте матрицы контекста (чтобы происходило умножение).

У нас есть вектор слова и вектор контекста (и у них одинаковая длина). Как найти меру близости между векторами?

**Косинусное сходство** векторов A=(a1,...,an), B=(b1,...,bn) :



Как следует из формулы, близость векторов пропорциональна сумме (скалярное произведение)

{a_1b_1+\ldots+a_nb_n}

**На этом и будет основана тренировка НС.**

У нас в ТВ есть пара (мести, весь). Этим словам соответствую вектора

**«мести»**=(w11,w12) — вектор слова

**«весь»**=(u21,u22) — вектор контекста.

Вычисляем их скалярное произведение: w11u21+w12u22

У нас в ТВ есть пара (мести, весь). Этим словам соответствуют вектора

**«мести»**=(w11,w12), «**весь»**=(u21,u22).

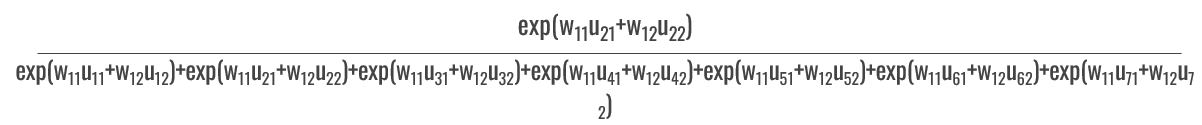
Вычисляем их скалярное произведение: w11u21+w12u22

А также вычисляем скалярное произведение вектора-слова «мести» со **всеми** остальными векторами-контекстами:

w11u11+w12u12 w11u21+w12u22 w11u31+w12u32 w11u41+w12u42

w11u51+w12u52 w11u61+w12u62 w11u71+w12u72

Выписываем функцию потерь



В функцию потерь НС L(w) будет записано слагаемое: …-ln(дробь со слайда)-...

Подобное слагаемое нужно выписать для каждой строки из ТВ.

А потом нужно минимизировать это выражение.

Что даст минимизация функции потерь L(w)? Она даст оптимальные значения координат векторов-слов **w** и векторов-контекстов **u**. Нам важнее вектора **w** — они дают представление слов в виде векторов.

Более тонкая настройка word2vec:

* Можно варьировать размерность векторов-слов и векторов-контекстов (в нашем примере она равнялась 2).
* Можно брать более широкий радиус контекста (в нашем примере он был равен 1).

Но смысл тренировки НС не поменяется.

# Negative sampling

Общий вид функции потерь для НС word2vec

1. **Это сумма логарифмов** с обратными знаками (это следует из общей теории).
2. **Количество логарифмов** равно объёму ТВ.
3. Внутри каждого логарифма стоит дробь. В знаменателе дроби — сумма экспонент. **Количество слагаемых в знаменателе совпадает с количеством слов в словаре.**

Например, если в русском языке порядка 100.000 слов, то в **каждом** знаменателе **каждого** логарифма в функции потерь будет 100.000 слагаемых.

Количество логарифмов равно количеству слов **во всех статьях** русской Википедии (мы условились тренировать НС на всей Википедии)!

Эту функцию потерь L(w) нужно минимизировать.

Таким образом, функция потерь L(w) в задаче word2vec просто огромная. В ней миллионы слагаемых, а внутри каждого слагаемого еще порядка 100 тысяч слагаемых в знаменателе:

L(w)=-ln(...)-ln(...)-...-ln(...)

Даже для современных компьютеров вычисления с участием такой функции — дело медленное.

Нужно упрощать вычисления с функцией L(w). Возможно, даже в ущерб качеству обучения (**спойлер**: качество остаётся удовлетворительным).

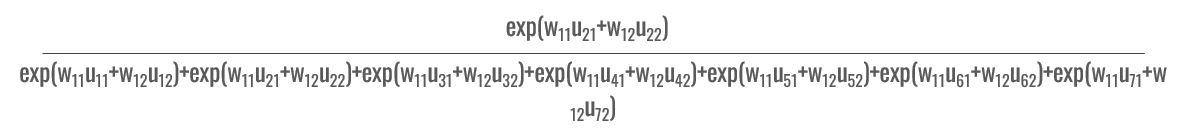
Как упрощать вычисления?

**Стохастический ГС.** Этот метод должен быть вам знаком, он изучается в любом общем курсе по нейронным сетям.

Суть: на каждой итерации ГС случайно выбирается ровно одно слагаемое из функции потерь, и градиент считается только по нему.

Это означает, что на каждой итерации ГС рассматривается ровно один логарифм из суммы. Но внутри логарифма стоит дробь, знаменатель которой содержит порядка 100 тыс. слагаемых. А для решения этой проблемы см. **negative sampling.**

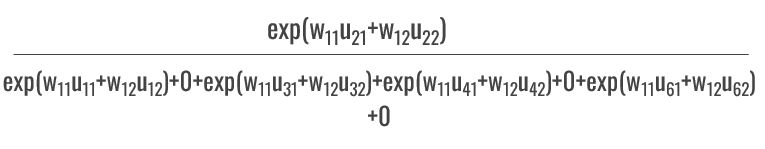
**Negative sampling.** Рассмотрим «под-логарифмическое» выражение. В нашем примере оно имеет знаменатель с 7 слагаемыми, поскольку в нашем «языке» всего 7 слов: **мести весь земля предел свеча гореть стол**



В «великом-и-могучем» русском языке слагаемых будет вообще порядка 100 тыс.

**Идея**: на каждой итерации ГС будет оставлять в знаменателе лишь несколько случайно выбранных слагаемых!

То есть на каждой итерации ГС от «великого-и-могучего» остаётся лишь несколько слов, по весам которых и вычисляются градиенты.



Фактически на каждой итерации мы работаем со случайной и небольшой подвыборкой нашей тренировочной выборки [X,Y].

**Слова** языка — это **теперь вектора в пространстве** достаточно небольшой (по сравнению с размером словаря) размерности.

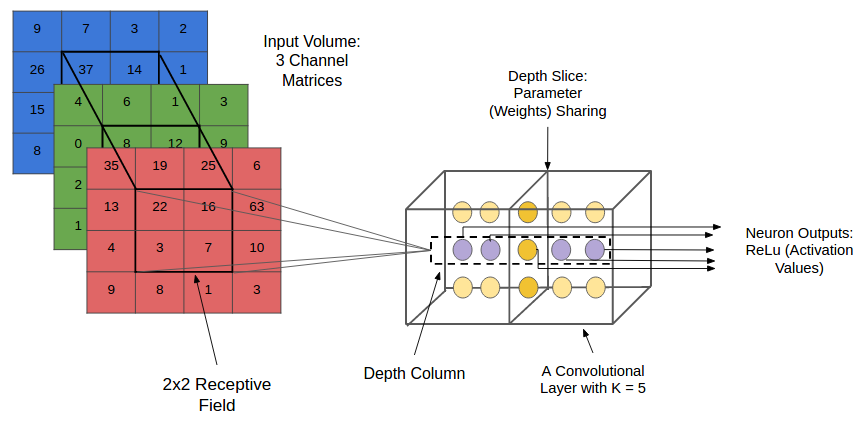
**Текст теперь — это последовательность векторов** T=(v1,v2,...,vk).

Это позволяет для распознавания текстов **применять НС, которым на вход подаются многомерные массивы**.

В частности, тексты могут распознаваться **свёрточными НС**.

Да-да, между картинками и текстами много общего, и там и там значение элемента определяется его контекстом (окрестностью).

**СНС для картинок.** Фильтр скользит по «поверхности» и сворачивает окрестность каждого пиксела с помощью фильтра. Каждый пиксел — это вектор длины 3.



**СНС для текстов.** Фильтр скользит по тексту и сворачивает окрестность каждого слова с помощью фильтра. В этом примере каждое слово представлено вектором длины 6.

