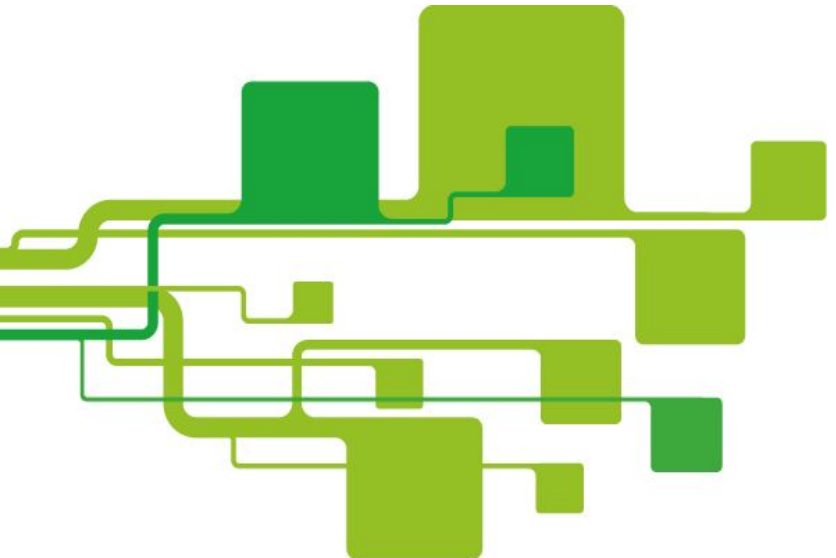


Нейронные сети

для рекомендательных
систем



Артем Просветов,
к.ф.-м.н.



Рекомендательная система

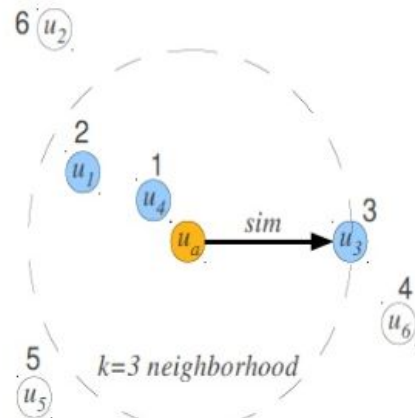


Коллаборативная фильтрация (user-based-вариант)

- По каждому клиенту ищем группу наиболее похожих на него клиентов
- Усредняем интересы данной группы
- Минус –kNN подход плох на больших масштабах

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	?	4.0	4.0	2.0	1.0	2.0	?	?
u_2	3.0	?	?	?	5.0	1.0	?	?
u_3	3.0	?	?	3.0	2.0	2.0	?	3.0
u_4	4.0	?	?	2.0	1.0	1.0	2.0	4.0
u_5	1.0	1.0	?	?	?	?	?	1.0
u_6	?	1.0	?	?	1.0	1.0	?	1.0
u_a	?	?	4.0	3.0	?	1.0	?	5.0
r_a	3.5	4.0			1.3	2.0		

$$\hat{r}_{aj} = \frac{1}{\sum_{i \in \mathcal{N}(a)} s_{ai}} \sum_{i \in \mathcal{N}(a)} s_{ai} r_{ij}$$

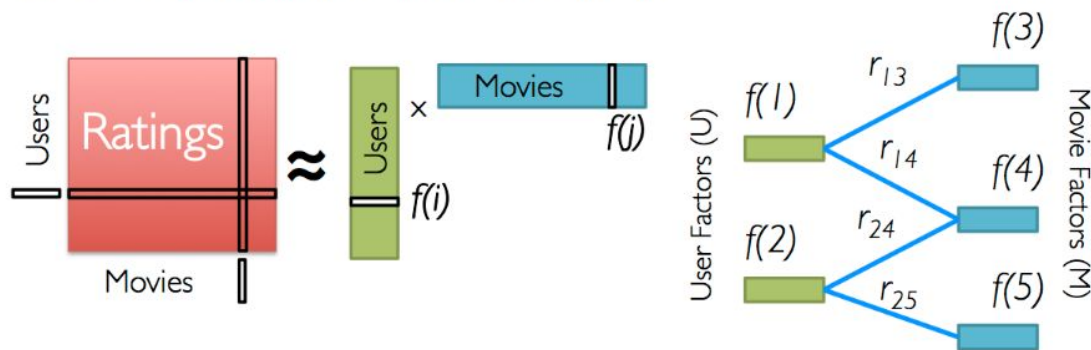


$$\text{sim}_{\text{Cosine}}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

Коллаборативная фильтрация

(Alternating Least Squares)

Low-Rank Matrix Factorization:



Iterate:

$$f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda ||w||_2^2$$

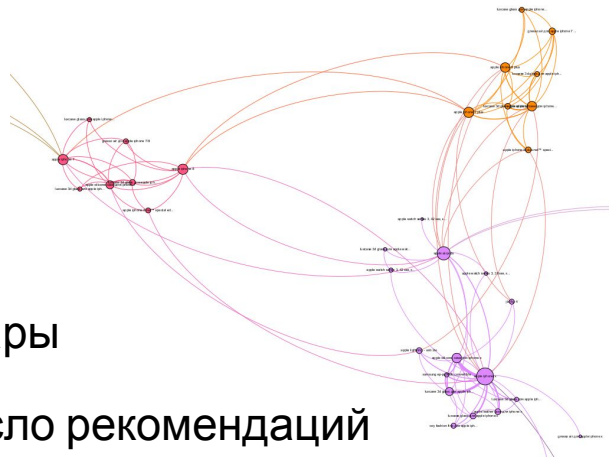
Taken from the BerkeleyX Course Big Data Analysis with Spark

Коллаборативная фильтрация

- Item-based – по каждому продукту ищем группу наиболее похожих продуктов с точки зрения пользовательских предпочтений
- Преимущества:
 - Меньше размерность матрицы расстояний
 - Легче вычислять
 - Модель более устойчива к переобучению
 - Можно реже обновлять
 - Меньше подвержены изменению предпочтений со временем

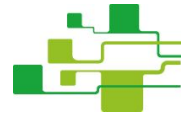
Ассоциативные правила

- Классический метод для поиска продуктовых ассоциаций
- Плюс
 - Получаем красивый граф связей
- Минусы
 - Можем рекомендовать только новые товары
 - Не всегда можем получить требуемое число рекомендаций



А если предпочтения меняются?

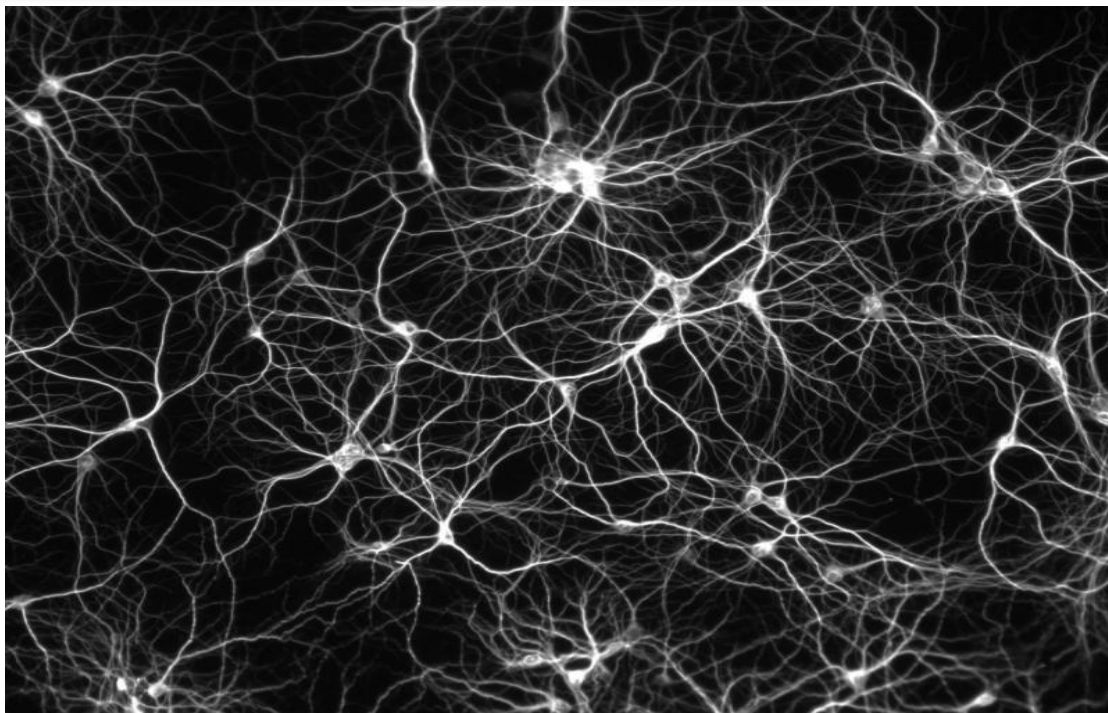
- Матрица предпочтений отражает предпочтения “в плоскости”
- Матрица предпочтений не отражает изменения предпочтений во времени
- Матрица предпочтений не учитывает близость продуктов по составу/цели/назначению



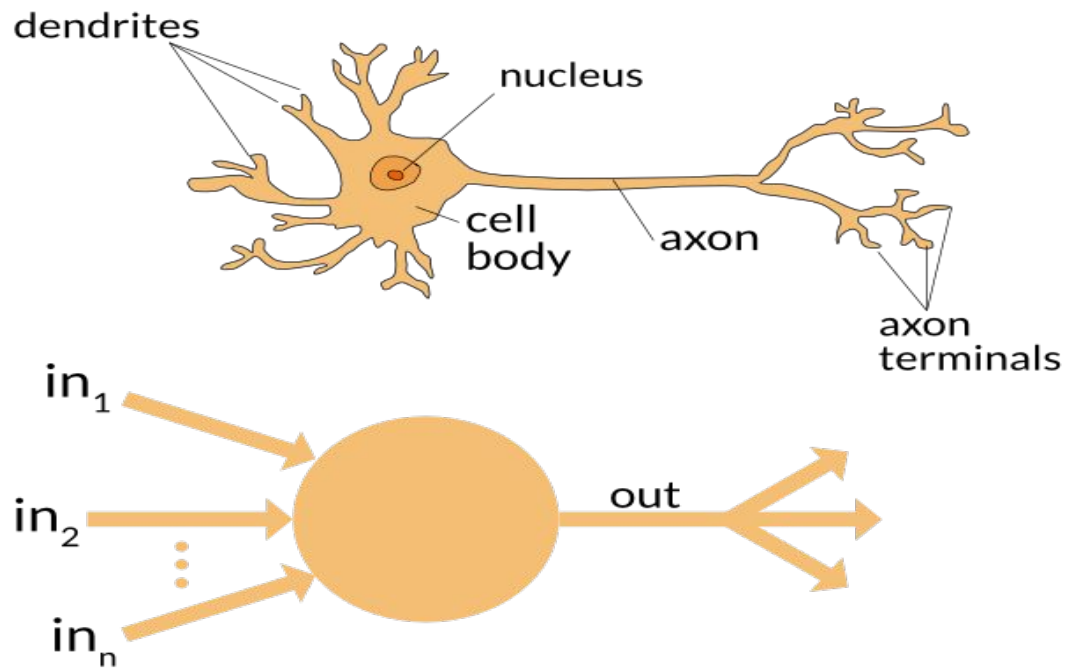
Нейронные сети: введение



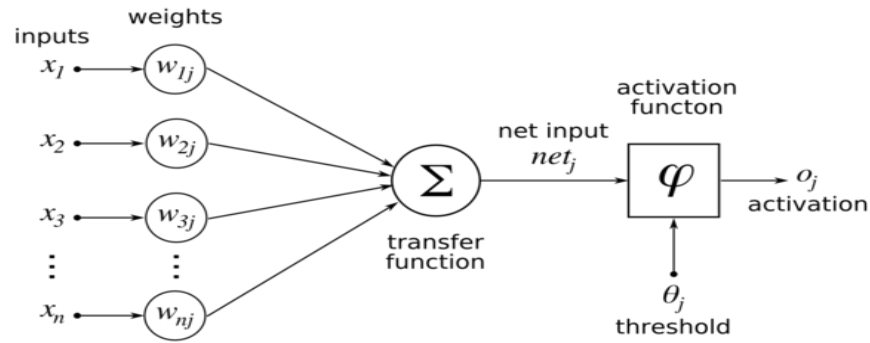
Нейронные сети



Модель перцептрона



Нейронные сети

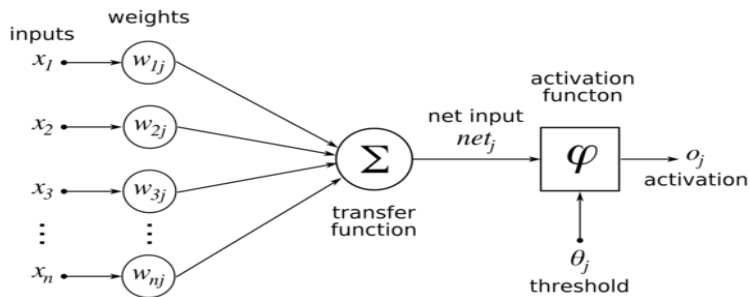


Бинарный классификатор:

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Обучение сети

Шаг 1: получаем вектора на выходе

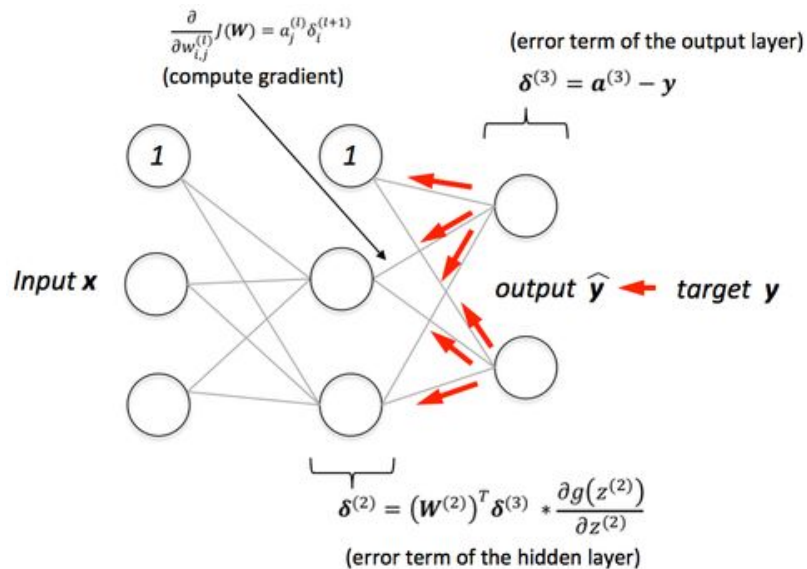
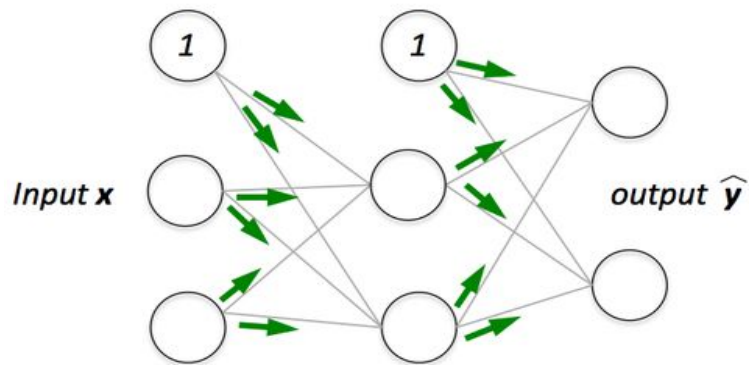


$$\begin{aligned} y_j(t) &= f[\mathbf{w}(t) \cdot \mathbf{x}_j] \\ &= f[w_0(t)x_{j,0} + w_1(t)x_{j,1} + w_2(t)x_{j,2} + \dots + w_n(t)x_{j,n}] \end{aligned}$$

Шаг 2: обновляем веса

$$w_i(t+1) = w_i(t) + (d_j - y_j(t))x_{j,i}$$

Обратное распространение ошибки

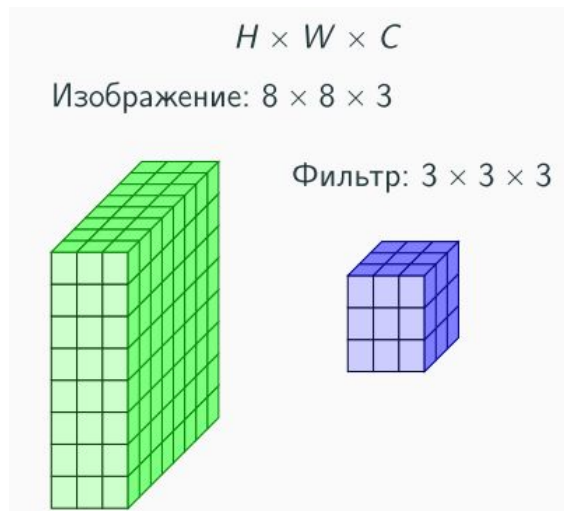




Нейронные сети: архитектуры



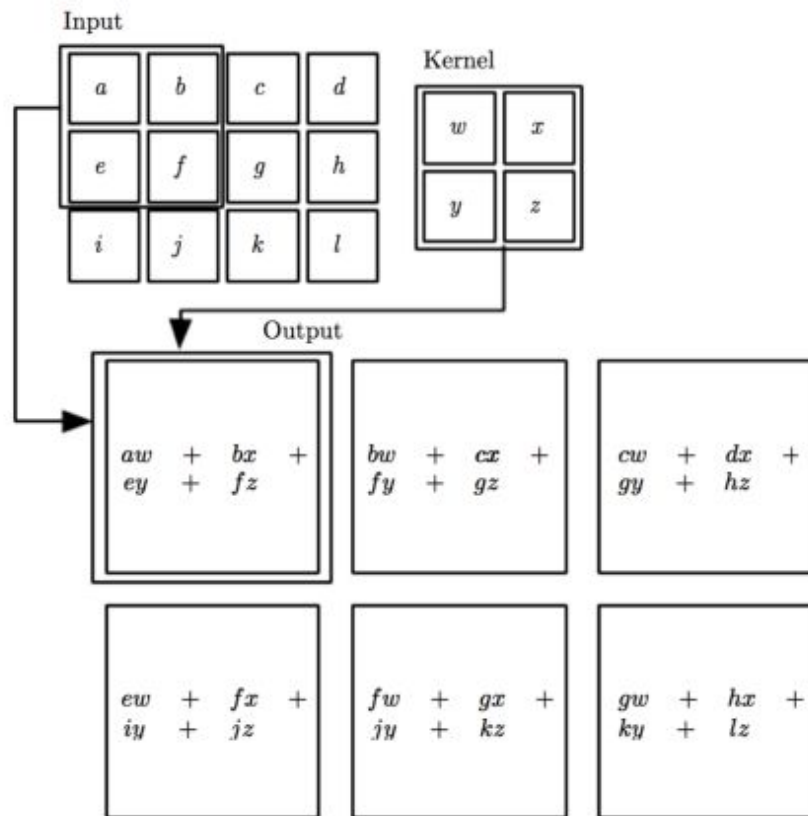
Сверточный слой



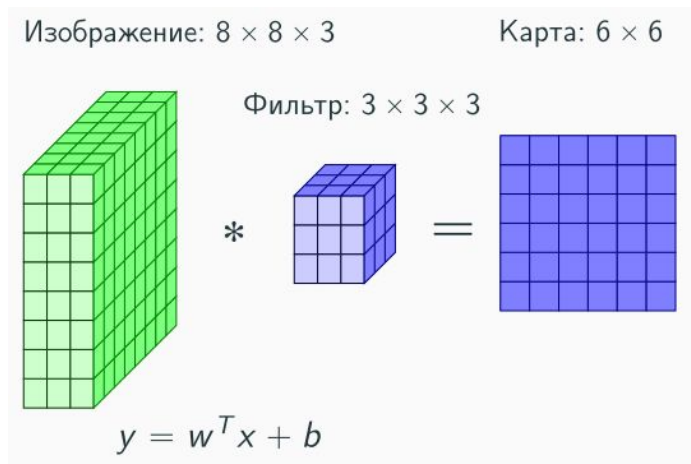
Свернуть изображение с
фильтром:

пробежать по изображению
(пространственно), вычисляя
скалярные произведения

Свертка изображения с ядром



Сверточный слой

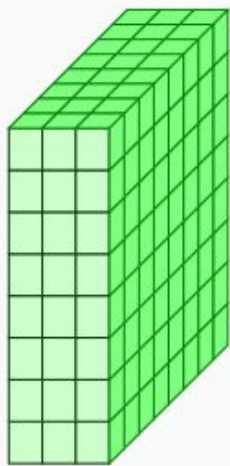


- Сколько параметров?
 $28 = 3 \times 3 \times 3 + 1$
- Сколько параметров у полносвязного слоя?

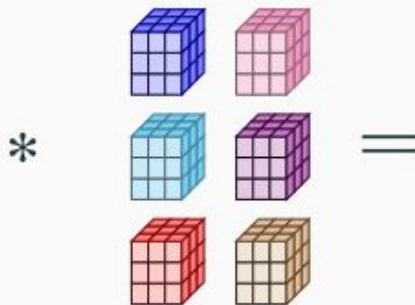
$$6913 = (8 \times 8 \times 3) \times (6 \times 6) + 1$$

Сверточный слой

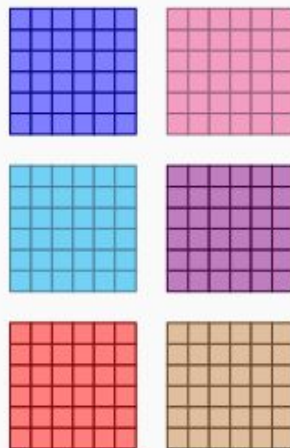
Изображение: $8 \times 8 \times 3$



Фильтры $3 \times 3 \times 3$

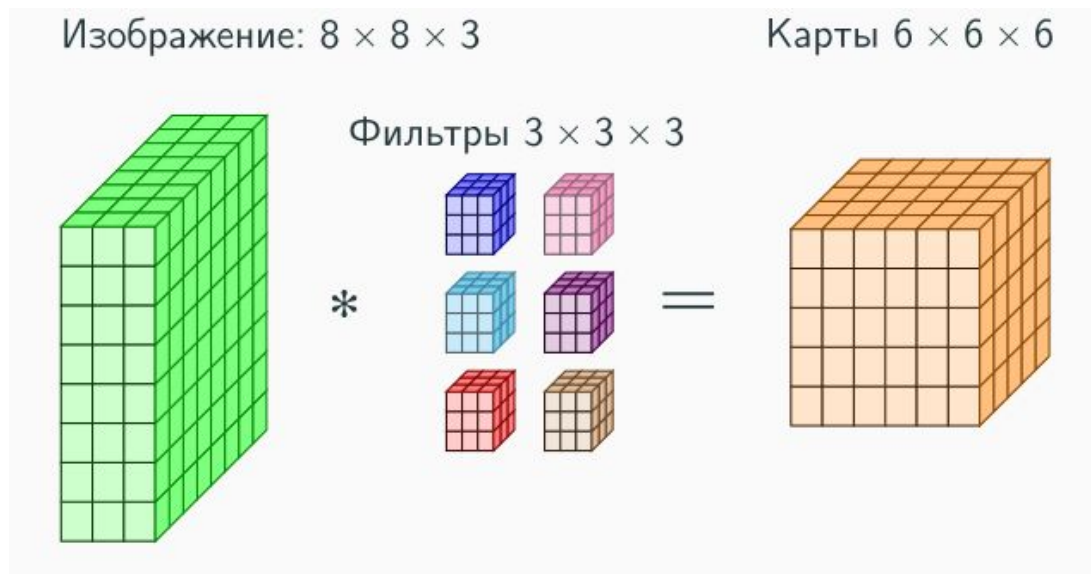


Карты 6×6



Число фильтров F : гиперпараметр

Сверточный слой



Тензор $6 \times 6 \times 6$: входное изображение
следующего сверточного слоя

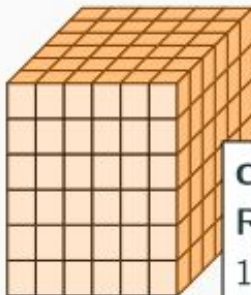
Сверточная сеть: последовательность сверток

$8 \times 8 \times 3$



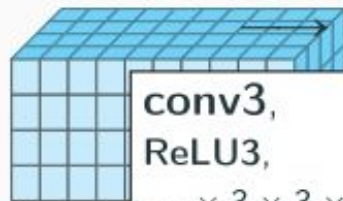
conv1,
ReLU1,
 $6 \times 3 \times 3 \times 3$

$6 \times 6 \times 6$



conv2,
ReLU2,
 $10 \times 3 \times 3 \times 6$

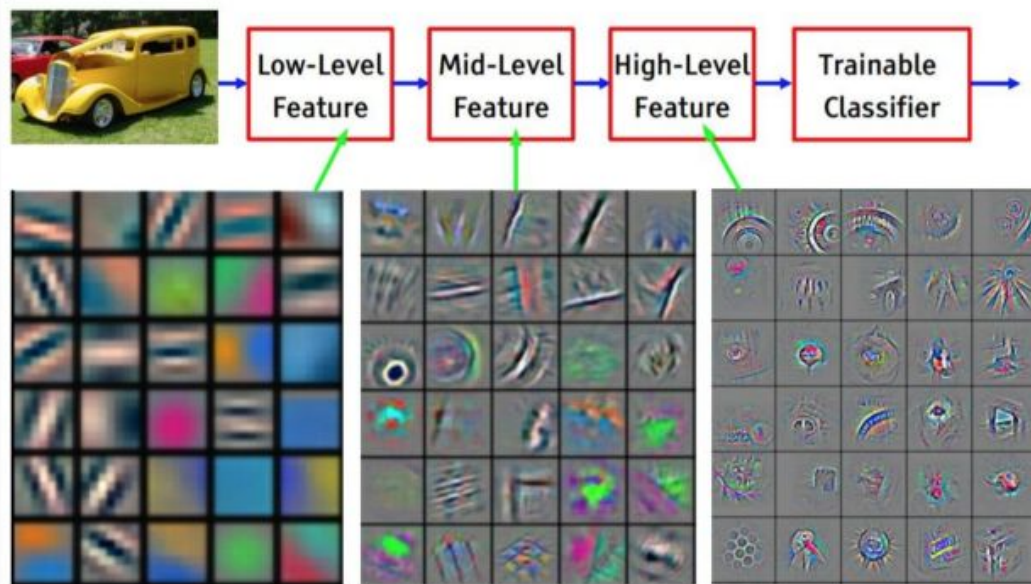
$4 \times 4 \times 10$



...

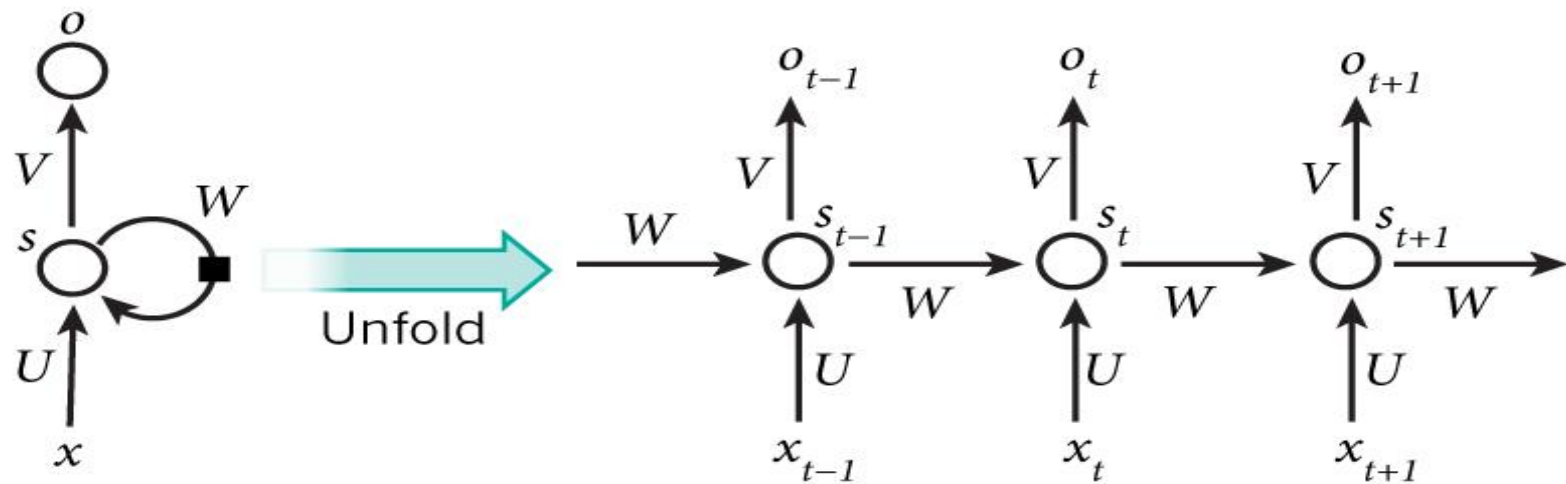
conv3,
ReLU3,
 $\dots \times 3 \times 3 \times 10$

Иерархия представлений

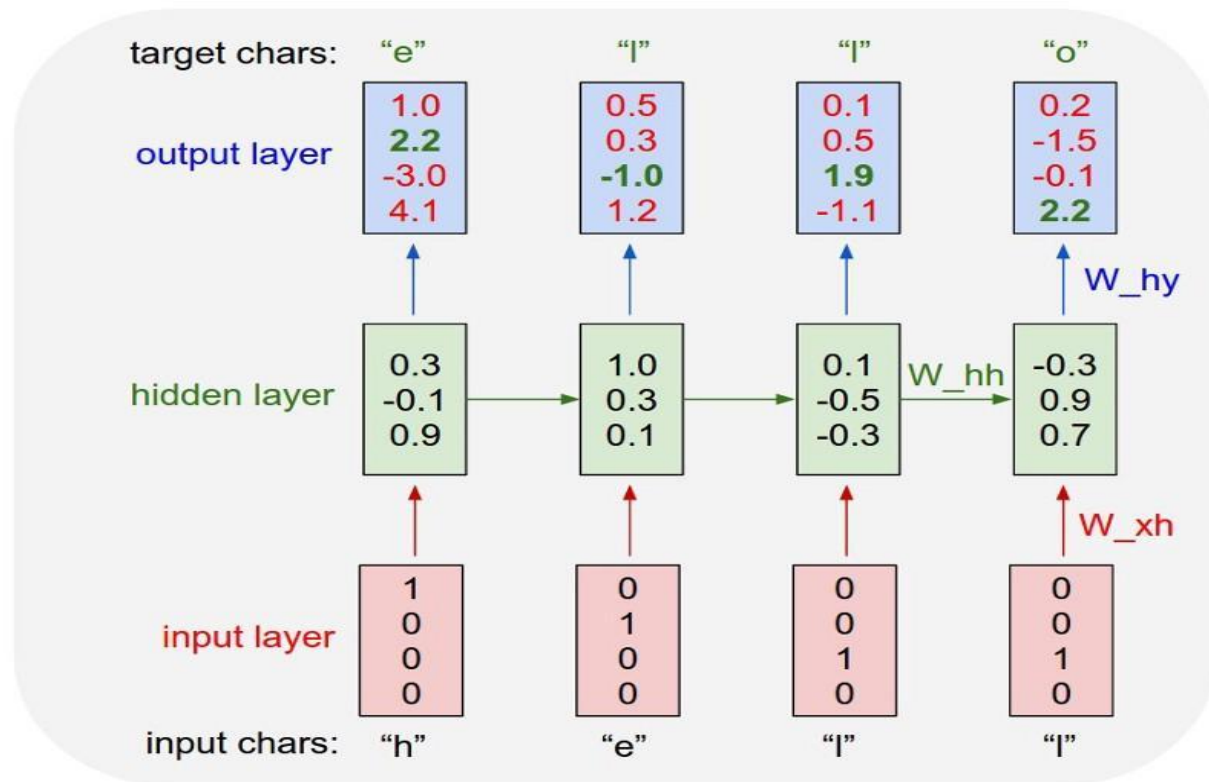


(Слайд Yann LeCun)

Recurrent Neural Network

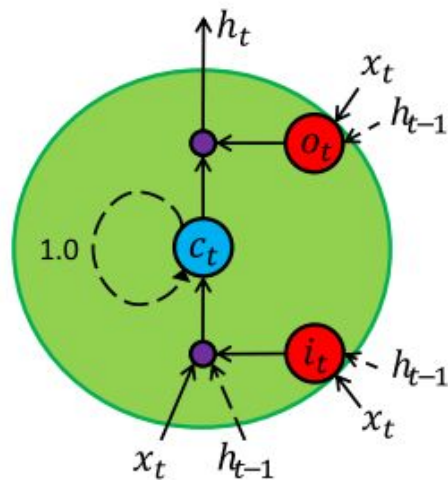


Recurrent Neural Network



Long short term memory

Version 0



Gate values in $[0,1]$

i_t, o_t - input/output gates

c_t - memory

$$i_t = \sigma(V_i x_t + W_i h_{t-1} + b_i)$$

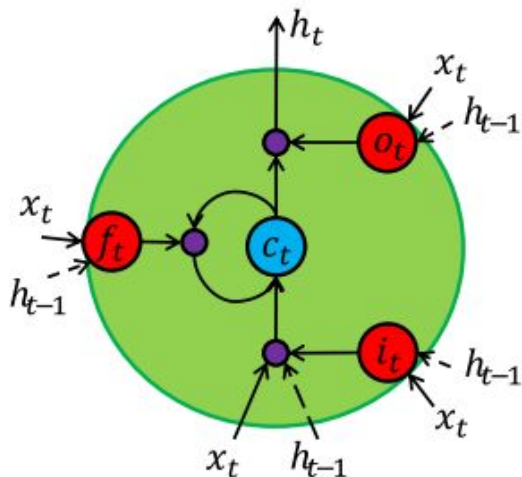
$$o_t = \sigma(V_o x_t + W_o h_{t-1} + b_o)$$

$$c_t = c_{t-1} + i_t \cdot g(V_c x_t + W_c h_{t-1} + b_c)$$

$$h_t = o_t \cdot g(c_t)$$

Long short term memory

Version 1



Gate values in $[0,1]$

i_t, o_t, f_t - input/output/forget gates
 c_t - memory

$$i_t = \sigma(V_i x_t + W_i h_{t-1} + b_i)$$

$$f_t = \sigma(V_f x_t + W_f h_{t-1} + b_f)$$

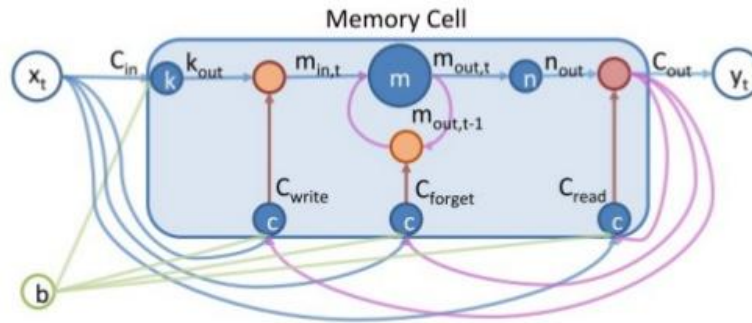
$$o_t = \sigma(V_o x_t + W_o h_{t-1} + b_o)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g(V_c x_t + W_c h_{t-1} + b_c)$$

$$h_t = o_t \cdot g(c_t)$$

Long Short-Term Memory

- recurrent neural networks with memory
- able to connect and recognize long-range patterns between words in text

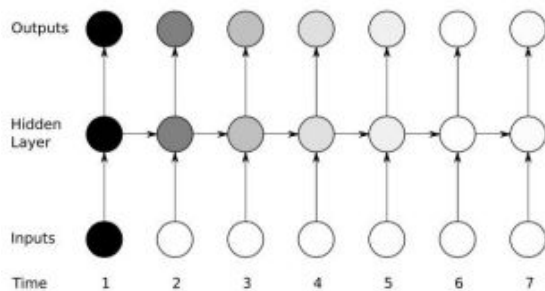


LSTM-сети хороши в случаях событий, разделенных временными лагами с неопределённой продолжительностью и границами

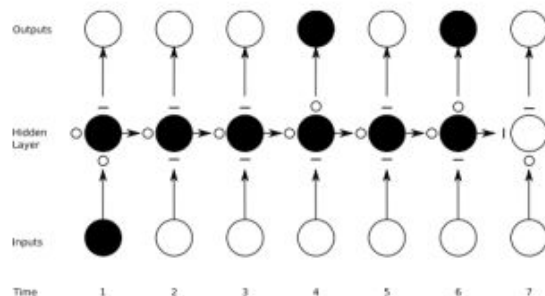
Long short term memory

Examples

RNN



LSTM



— - gate is close

○ - gate is open

[\[Graves, 2012\]](#)



Нейронные сети для рекомендаций

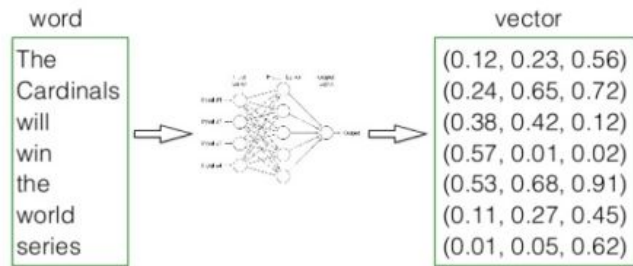




Word2Vec?



Word2Vec



После преобразования слова можно складывать, вычитать и т. д.

- Word2Vec – структура данных, первоначально разработанная и используемая компанией Google для анализа поисковых запросов
- На вход подается большое количество необработанных текстов, а также другие дополнительные параметры
- На выходе для каждого слова имеем вектора одинаковой длины, состоящие из не интерпретируемых числовых признаков

- Пусть для каждого покупателя последовательность покупок закодирована в текст типа:
покупка1 покупка2 покупка3 ... покупка4
- Получаем вместо разреженной матрицы корпус текстов
- Обучаем Word2Vec на полученном корпусе
- Получаем все преимущества, которые дает Word2Vec

преимущества Word2Vec

- Кластеризация векторов покупателей
- Поиск ближайшего “слова-продукта” посредством косинусного расстояния
- Вычисление “расстояния” между покупателями
- Векторные операции с продуктами:
 принтер + ? = кофеварка + капсулы



Restricted Boltzmann Machine



Boltzmann Machine

- Является обобщённым вариантом машины Хопфилда.
- Сеть имеет стохастическую природу
- Нейроны поделены на две группы, описывающие наблюдаемые и скрытые состояния

Boltzmann Machine

- Энергия сети:

$$E = - \sum_{i < j} w_{ij} s_i s_j - \sum_i \theta_i s_i$$

w_{ij} сила связи между нейронами j и i .

s_i состояние нейрона i , $s_i \in \{0, 1\}$

θ_i порог для нейрона i .

Boltzmann Machine

- Идея использования «теплового шума» для выхода из локальных минимумов и повышения вероятности попадания в более глубокие минимумы принадлежит С. Кирпатрику.
- Введем некоторый параметр t — аналог уровня теплового шума. Тогда вероятность активности k -го нейрона:

t — уровень теплового шума в сети;

E_k — сумма весов связей k -го нейрона со всеми активными в данный момент нейронами.

$$P_k = 1 / (1 + e^{-E_k/t}),$$

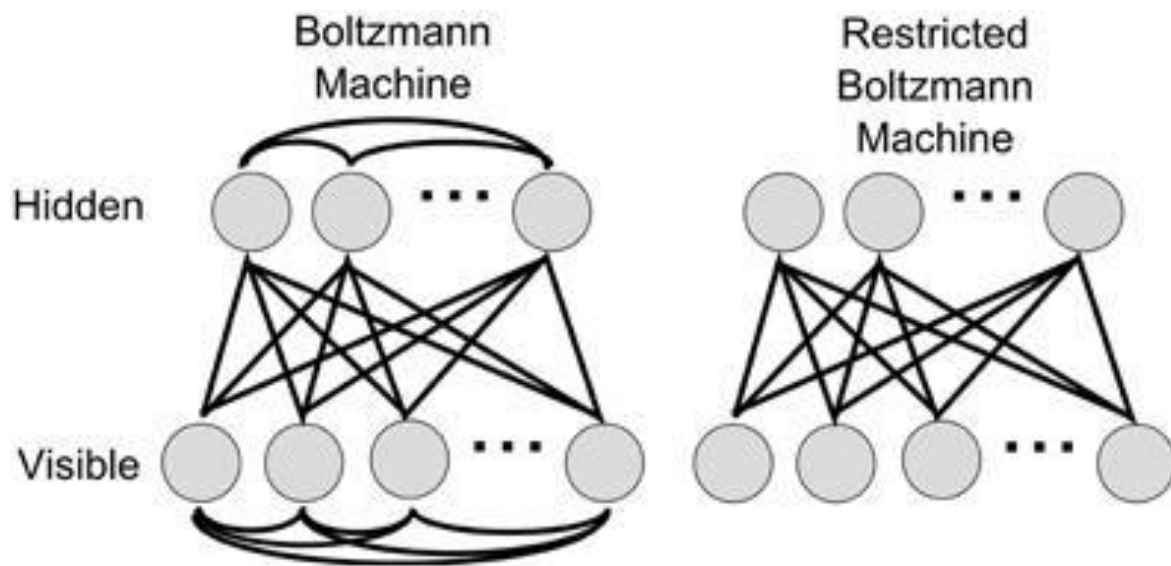
Интерпретация

- Есть ряд состояний, которые мы можем наблюдать
- Есть ряд состояний, которые скрыты.
- Мы можем сделать вероятностный вывод относительно скрытых состояний, опираясь на состояния, которые мы можем наблюдать.
- Обучив такую модель, мы получаем возможность делать выводы относительно видимых состояний, зная скрытые -> генерировать данные из вероятностного распределения, на котором обучена модель.

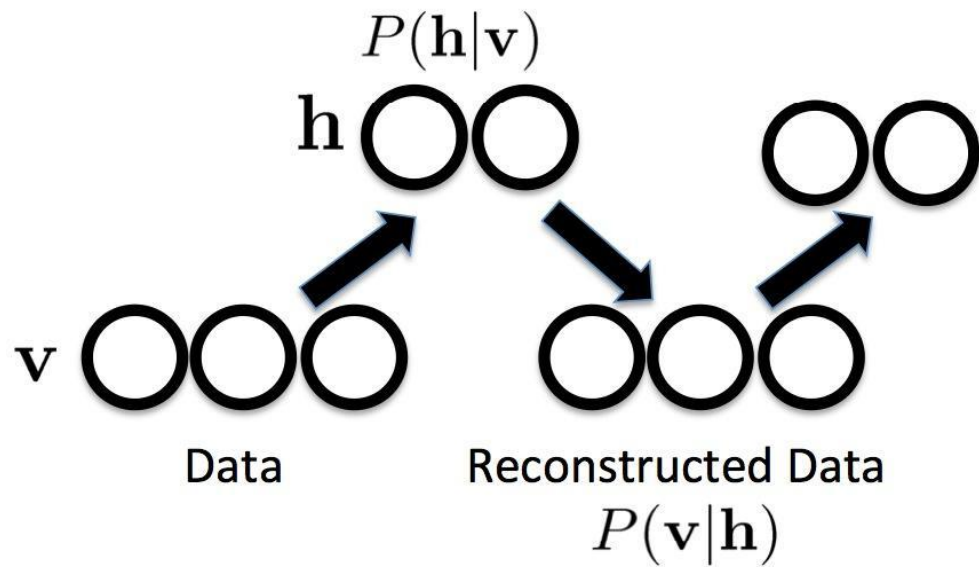
недостатки машины Больцмана

- Время, необходимое для достижения состояния равновесия, растет экспоненциально с размером машины и силами связи
- Связи осциллируют из-за сетевого эффекта и в итоге “залипают” на крайних значениях (0 / 1)
- В итоге, машины Больцмана представляют из себя теоретический конструкт, практически не используемый на практике

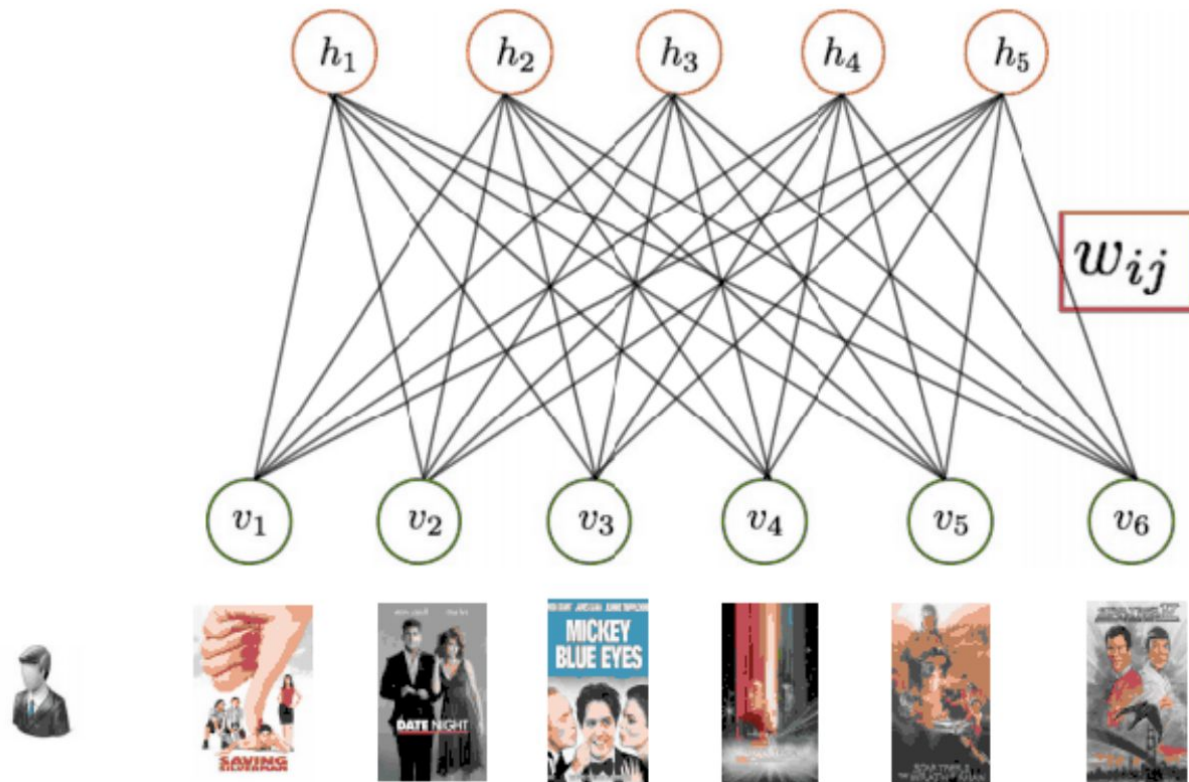
Архитектура BM / RBM



Тренировка RBM

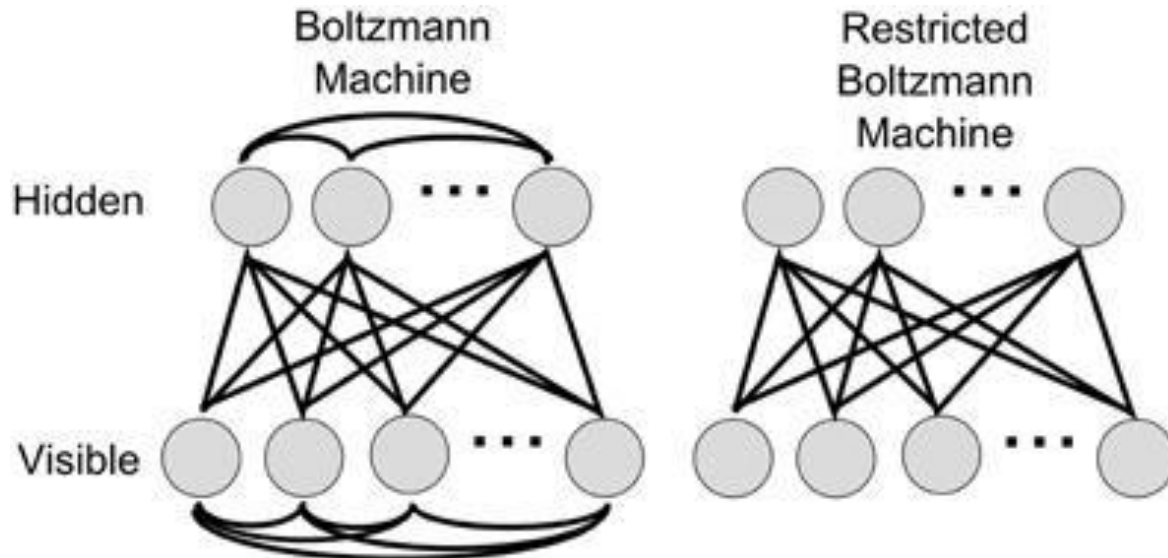


Restricted Boltzmann Machine

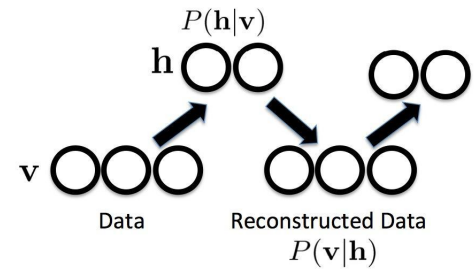


Restricted Boltzmann Machine

- Каждый продукт является элементом видимого слоя
- Количество нейронов скрытого слоя - параметр



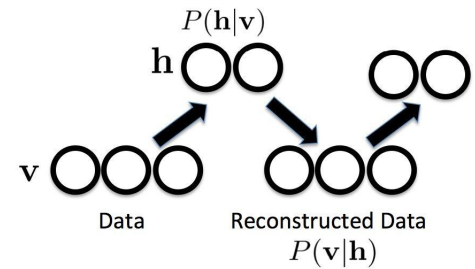
Restricted Boltzmann Machine



Процесс тренировки:

- Если пользователь купил продукт, то соотв. нейрон видимого слоя активируется
- Состояния нейронов видимого слоя подаются на вход скрытого слоя
- Вычисляются состояния скрытого слоя
- состояния скрытого слоя подаются на вход видимого слоя и состояния нейронов видимого слоя пересчитывается
- Разность между текущим состоянием видимого слоя и прошлым состоянием используется для обновления весов связей между нейронами

Restricted Boltzmann Machine



Для вычислений рекомендаций на RBM:

- Состояние видимого слоя соотв. продуктам пользователя
- На основе состояния видимого слоя и натренированных весов, вычисляется состояние скрытого слоя
- Состояния скрытого слоя используются для пересчета состояния видимого слоя
- Вероятности активации видимого слоя используются как рейтинги рекомендаций

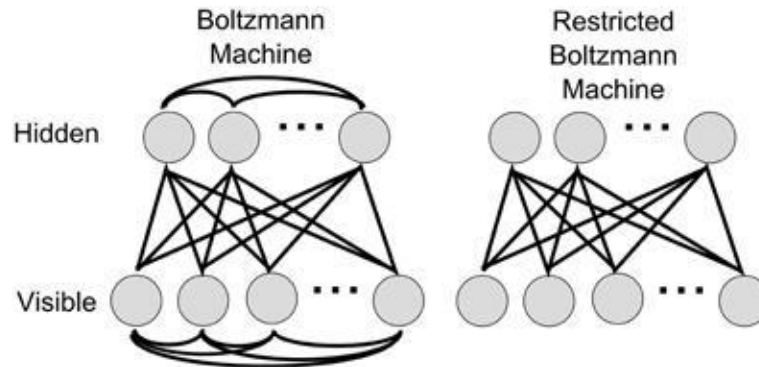
Restricted Boltzmann Machine

2007 Netflix Progress Prize:

SVD - Prize RMSE: 0.8914

RBM - Prize RMSE: 0.8990

В 2014 году Netflix для рекомендаций использовал ансамбль с RBM

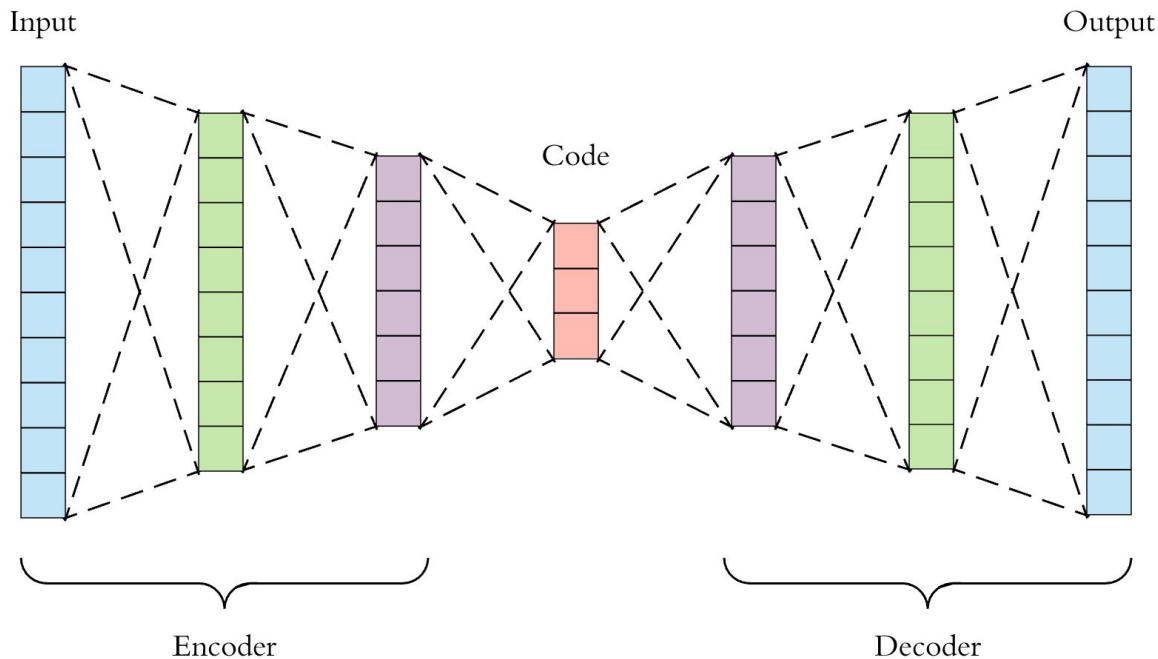




NVIDIA DeepRecommender

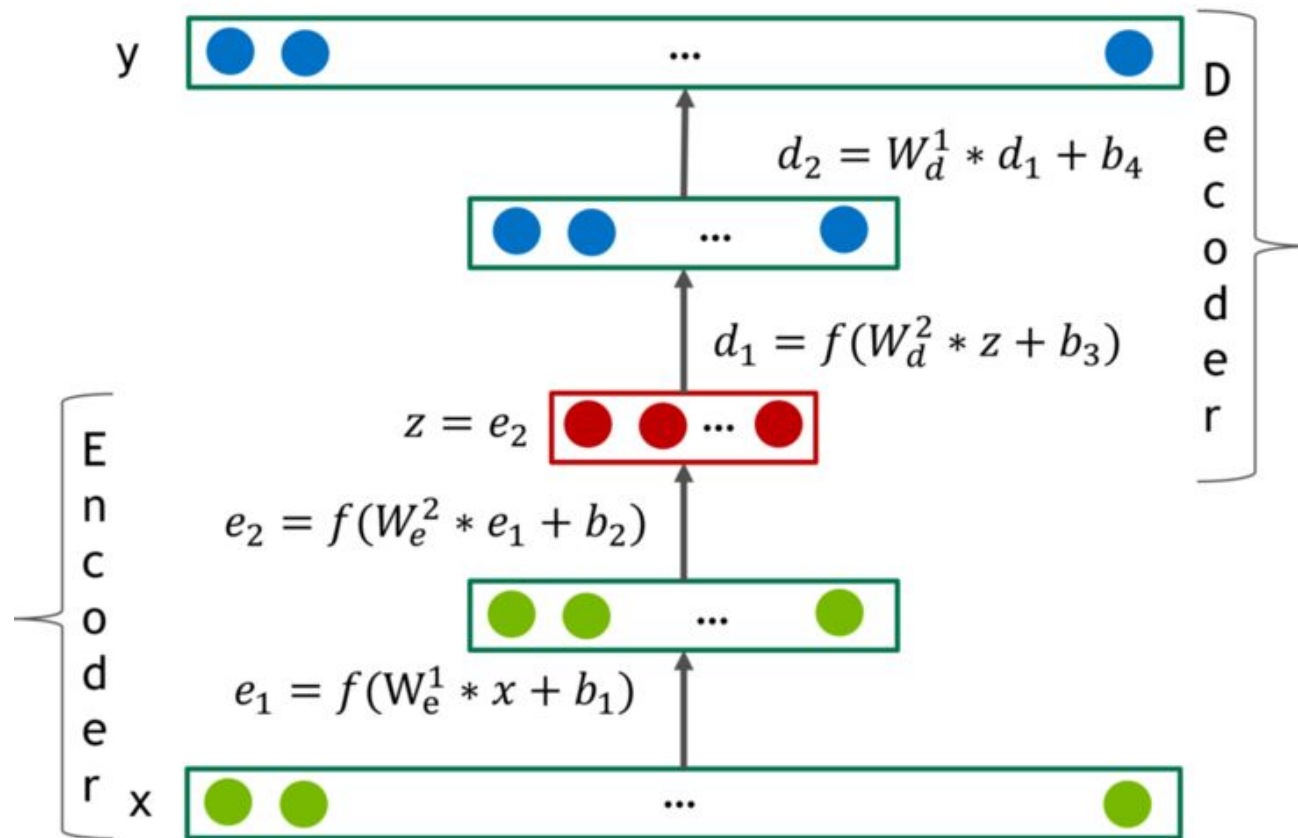


Автокодировщики



обучение без учителя при использовании метода обратного распространения ошибки.

DeepRecommender



DeepRecommender

- Deep autoencoder с 6 слоями
- нелинейная функция активации SELU (scaled exponential linear units)
- Dropout
- iterative dense refeeding -???

- Пусть у нас идеальный случай f :

$$f(\mathbf{x})_i = x_i \text{ для всех } i: x_i \neq 0$$

$f(\mathbf{x})_i$ точно предсказывает все будущие рейтинги

Это означает, что если пользователь оценивает новый продукт k (т.е. для нового вектора \mathbf{x}'):

$$f(\mathbf{x})_k = x'_k \text{ и } f(\mathbf{x}) = f(\mathbf{x}').$$

Следовательно, можно вернуть выход на вход в автоэнкодер для увеличения набора данных (augmentation).

Метод состоит из следующих шагов:

- На основе разреженной матрицы x вычисляем $f(x)$ и лосс (прямое распространение).
- Делаем обратное распространение ошибки и обновляем веса.
- Используем $f(x)$ как новый тренировочный пример и находим $f(f(x))$
- Снова используем метода обратного распространения ошибки.
- Повторяем шаги несколько раз

DeepRecommender

DataSet	RMSE	Model Architecture
Netflix 3 months	0.9373	n,128,256,256,dp(0.65),256,128,n
Netflix 6 months	0.9207	n,256,256,512,dp(0.8),256,256,n
Netflix 1 year	0.9225	n,256,256,512,dp(0.8),256,256,n
Netflix full	0.9099	n,512,512,1024,dp(0.8),512,512,n



Классификация / Регрессия



Возможно свести рекомендательную систему к задаче классификации множеством способов. Один из них:

- Последовательность покупок пользователя подается на вход
- Подается на вход новый продукт / продукты
- Если новый продукт есть в следующей покупке пользователя, то целевая функция равна 1
- Если новый продукт отсутствует в следующей покупке пользователя, то целевая функция равна 0

В нейронной сети можно использовать:

- Эмбединги
- Сверточные слои
- Рекуррентные слои
- LSTM / GRU
- скалярное произведение тензоров
- конкатенацию
- dropout
- etc

Построение рекомендаций:

- Последовательность текущих покупок пользователя подается на вход
- Подается на вход новый продукт
- Результат работы модели используется для ранжирования продуктов в рекомендациях



Deep Semantic Similarity Model



DSSM-модель

Posterior probability
computed by softmax

Relevance measured
by cosine similarity

Semantic feature

y

Multi-layer non-
linear projection

l_3

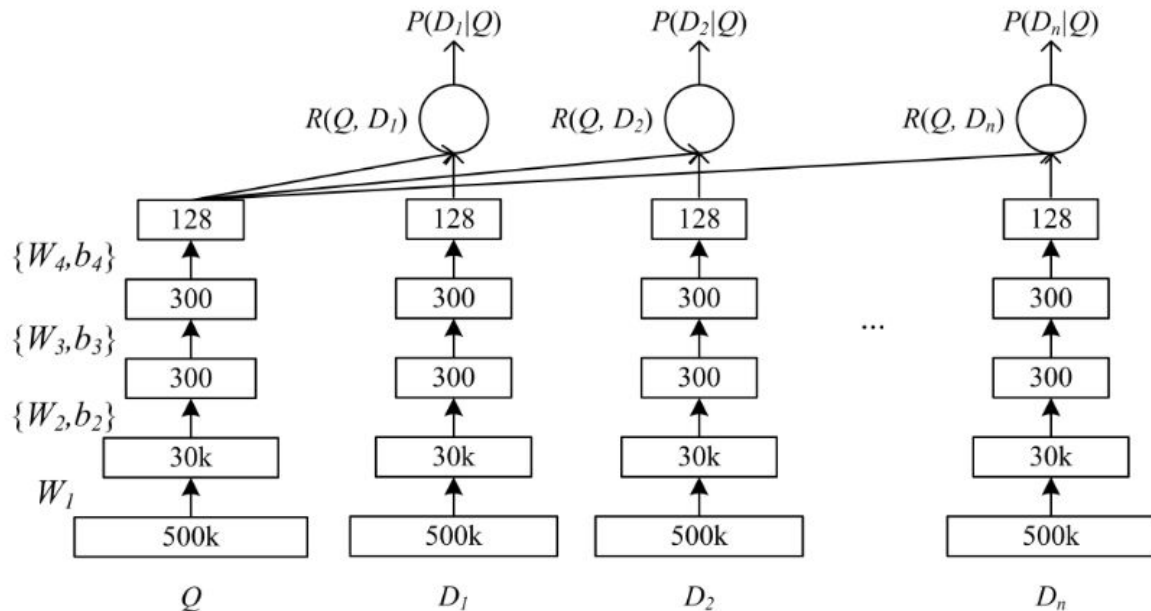
l_2

Word Hashing

l_1

Term Vector

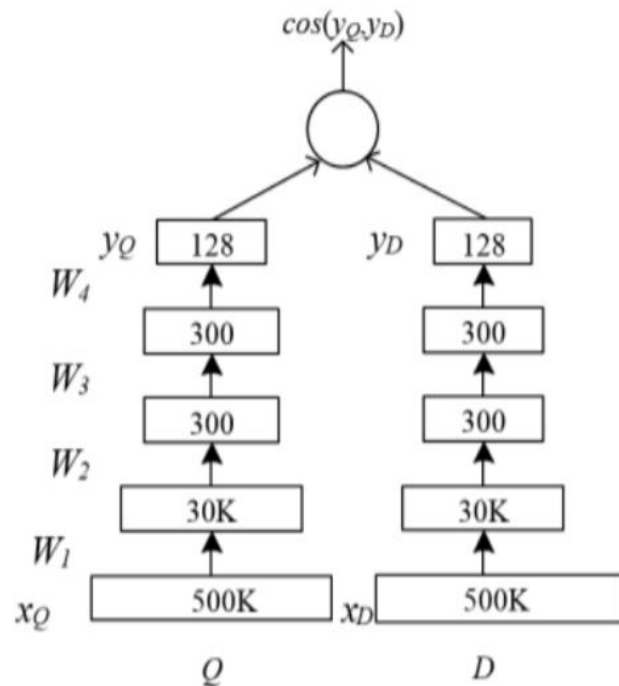
x



DSSM-модель

- Имея запрос Q , надо вывести релевантный ответ D
- Для решения этой задачи нужна функция, оценивающая связь $R(Q, D)$

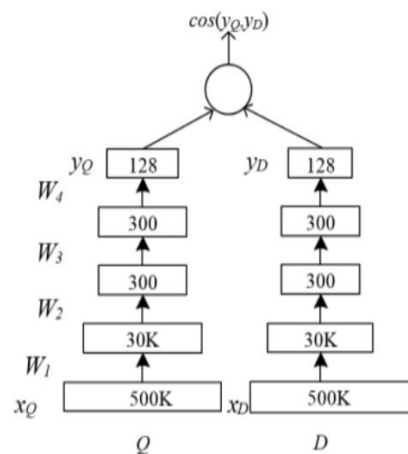
$$\text{sim}_A(Q, D) = \frac{\hat{Q}^T \hat{D}}{\|\hat{Q}\| \|\hat{D}\|}$$



Po-Sen Huang et al 2013

DSSM-модель

- на сегодняшний день является “рабочей лошадкой” Яндекс Дзен и других сервисов рекомендаций Яндекса
- В наших рекомендательных системах является одной из основных моделей





Что дальше?



Что дальше?

Какие грани рекомендательных систем сегодня можно совершенствовать?

- Необходимо принимать во внимание контекст рекомендации (где находится пользователь, что он делает, что делал ранее)
- В системе может оцениваться несколько характеристик товара, в этом в случае при показе рекомендаций должна учитываться каждая в отдельности
- Иметь возможность работать не с пользователями, а с группами пользователей (всей группе рекомендуем товар X)

Что дальше?

Какие грани рекомендательных систем сегодня можно совершенствовать?

- При показе рекомендаций учитывать различные бизнес-правила, фильтры и ограничения
- Фокусироваться на ранжировании рекомендаций, а не на оценке конкретного рейтинга
- Обращать внимание на accuracy/novelty tradeoff (часто интереснее советовать что-то принципиально новое)

Что дальше?

Какие грани рекомендательных систем сегодня можно совершенствовать?

- Вместо близости предпочтений учитывать социальную близость между пользователями (друзья по фейсбуку)
- Система должна подстраиваться под пользователя, пользователь должен иметь возможность настраивать систему
- Прозрачность рекомендаций (важно понимать почему показана именно такая)
- Развивать устойчивость к искусственным манипуляциям (чтобы фейковые отзывы не влияли на оценку)



Выводы



Выводы

- Классические методы рекомендаций срабатывают не всегда
- Нейронные сети позволяют строить более гибкие рекомендации
- Нейронные сети позволяют учитывать последовательность покупок



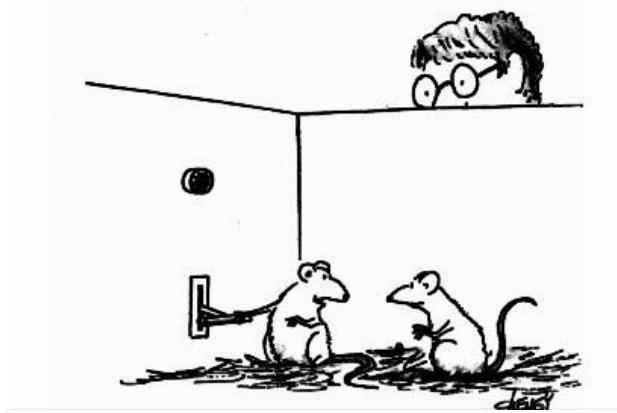
Как получить эмбединги



Как описать поведение людей?

Помогает в решении задач:

- Предсказание поведения клиента
- Предсказание оттока
- Предсказание мошеннических действий
- etc



Интересный феномен: каждый раз, когда я дергаю рычаг, экспериментатор вздыхает с облегчением!

Проблема

- Лог поведения — неструктурированная информация

User 4234123 at 2018-06-12 09:00:32, email opening

User 7472836 at 2018-06-12 09:00:49, email opening

User 7546243 at 2018-06-12 09:01:11, click url = <https://blabla.com/bla/bla>

User 4234187 at 2018-06-12 09:01:14, email bounced

User 3618934 at 2018-06-12 09:01:27, email received

User 1534803 at 2018-06-12 09:01:41, email opening

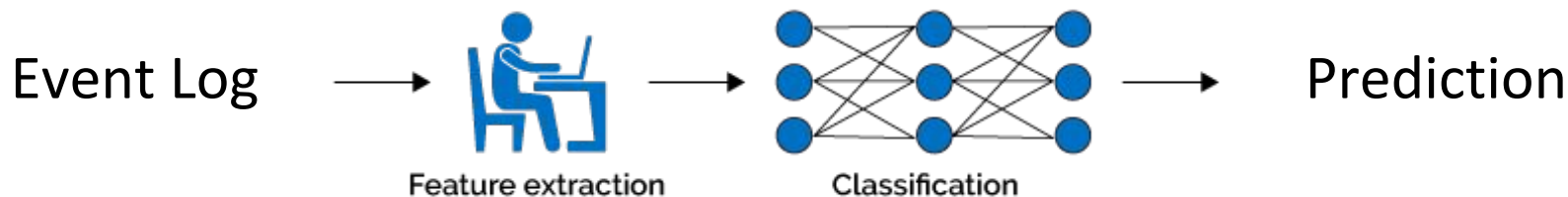
.....

- Как перевести логи в формат, подходящий для моделей ML?

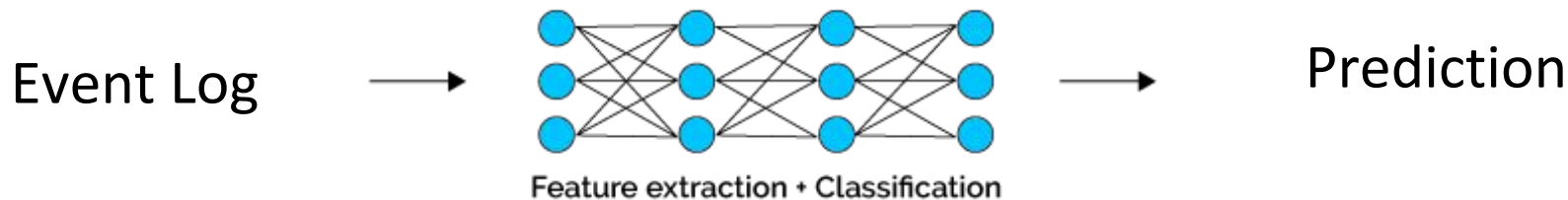
Проблема

Как перевести логи в формат, подходящий для моделей ML?

Machine Learning



Deep Learning



Кодирование событий

получил письмо	1 0 0 0 0 0 0
открыл письмо	0 1 0 0 1 0 0
кликнул на ссылку	0 0 1 0 0 0 0
зашел на сайт	0 0 0 1 0 1 0
совершил покупку	0 0 0 0 0 0 1

Проблема: у разных людей разное количество действий!

Кодирование событий

Добавим нули:

получил письмо	000010000000
открыл письмо	00000100100
кликнул на ссылку	00000010000
зашел на сайт	00000001010
совершил покупку	00000000001

Но здесь не учитывается время,
прошедшее между действиями!

Кодирование событий

получил письмо	0	0	0	0	1	0	0	0	0	0	0
открыл письмо	0	0	0	0	0	1	0	0	1	0	0
кликнул на ссылку	0	0	0	0	0	0	1	0	0	0	0
зашел на сайт	0	0	0	0	0	0	0	1	0	1	0
совершил покупку	0	0	0	0	0	0	0	0	0	0	1
норм. лог(Δt)	0	0	0	0	.1	.3	.2	.5	.2	.1	.1

Теперь можно обучать нейросеть!

Обучение нейронной сети

Сравнение метрик качества нейронных сетей в процессе обучения автокодировщика

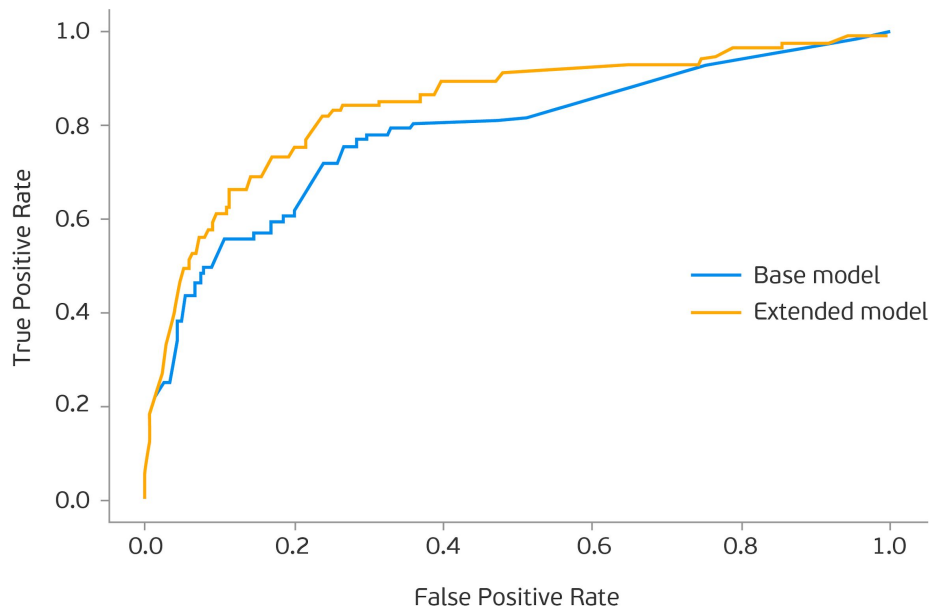
	LSTM	CNN	CNN+LSTM
Binary cross-entropy loss	0.1398	0.0968	0.0957

Применение нейронной сети

Сравнение метрик качества предсказательной модели, построенной на полученных признаках

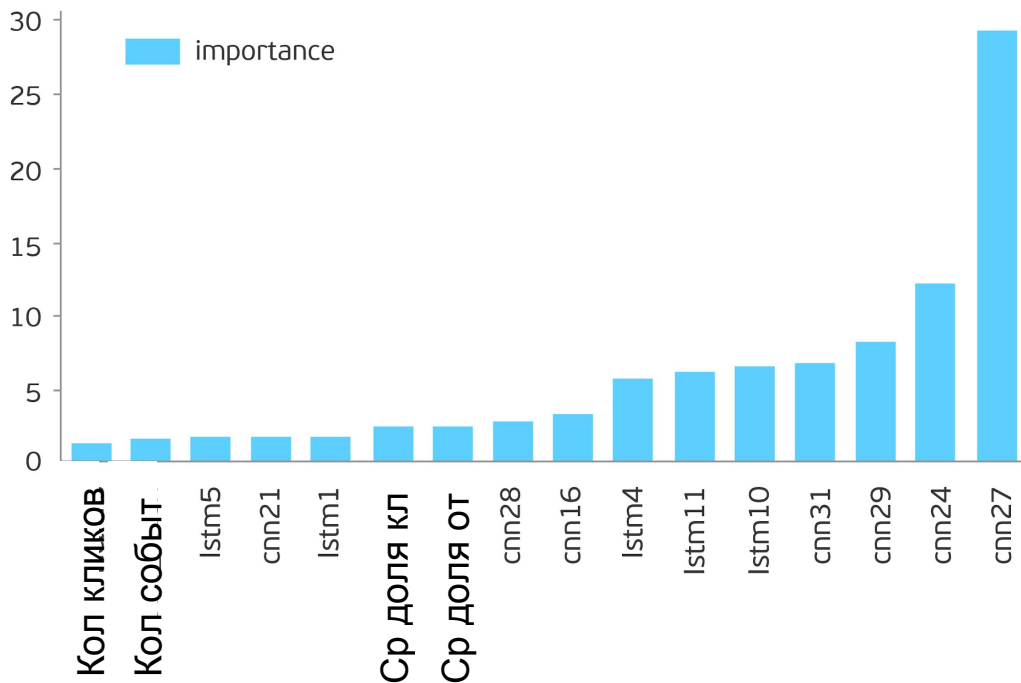
	LSTM	CNN	CNN+LSTM
Roc Auc	0.825 +/- 0.015	0.845 +/- 0.015	0.845 +/- 0.025

Сравнение ROC-кривых



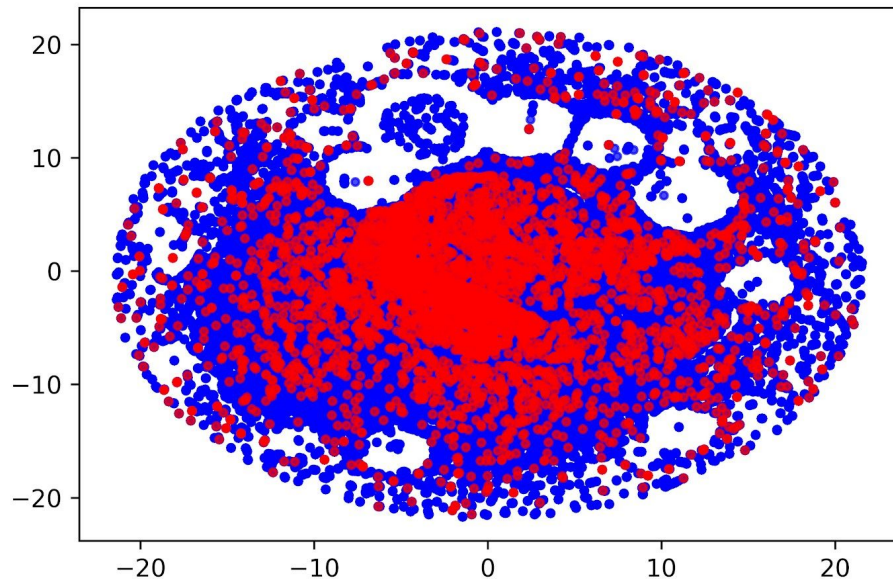
Добавление признаков от автокодировщика дает
прирост roc-auc +10%

Значимость признаков



Доминируют признаки от CNN автокодировщика

Проекция на 2D закодированного поведения



Красные точки — получатели, совершившие покупку
Синие точки — получатели, не совершившие покупку

Выводы

- Перевод неструктурированной информации в структурированную проблематичен и требует изобретения признаков
- Нейронные сети позволяют закодировать поведение человека без изобретения признаков экспертами
- Добавление закодированного поведения в качестве признаков дает прирост качества в предсказательных моделях



info@cleverdata.ru
cleverdata.ru