Wykorzystanie autoencoderów jako systemów rekomendacyjnych.

informacje
o mnie

Cześć! Tematyką data science i uczenia maszynowego zajmuję się od 2012. Zaczynałem jako programista ruby/python, stopniowo coraz bardziej skupiając się na zagadnieniach analizy danych. Mam doświadczenie w tworzeniu całego spektrum rozwiązań analitycznych – począwszy od systemów rekomendacyjnych, rozpoznawanie obrazów, poprzez prognozowanie szeregów czasowych, na computer vision skończywszy. Oprócz pracy zawodowej, jestem także w trakcie doktoratu, w ramach którego badam zastosowanie analizy asocjacyjnej, jako narzędzia wspomagania decyzji.

Filip Wójcik

Senior Data Scientist
PhD Candidate, UE Wroc

Filip.wojcik@outlook.com

https://filip-wojcik.com

agenda
prezentacji

systemy rekomendacyjne

Czym są i jak działają? Do czego zostały stworzone

klasyczne algorytmy

Jakie podejścia stosowano dotąd?

agenda prezentacji

autoencodery

Specyficzny rodzaj sieci neuronowych używanych do rekonstrukcji macierzy

przykład zastosowania

Porównanie działania algorytmów klasycznych i systemu *autorec*

Systemy rekomendacyjne

Czym są? Jak działają i do czego się je wykorzystuje? Jakie są najczęstsze problemy podczas ich używania?

01

systemy rekomendacyjne

Analiza preferencji

- Analiza zachowań konsumentów
- Próba odnalezienia wzorców zachowania
- Wyszukiwanie podobieństw pomiędzy osobami oraz produktami
- Trudne zadanie w czasach, gdy oferta produktowa jest bardzo szeroka

Reprezentacja

- Preferencje użytkowników albo klientów przejawiają się najczęściej w postaci **ocen (ratingów)**
- Oceny dawane przez klientów produktom stanowią podstawę systemu
- System nie ma dostępu do większości zmiennych opisujących osoby i produkty

Cel

- Próba rekonstrukcji **ukrytych (latentnych)** czynników, wpływających na decyzje
- Na podstawie takiej rekonstrukcji – przewidywanie przyszłych zachowań
- Rekomendowanie produktów zgodnych z preferencjami
- Tę ogólną ideę można zaimplementować na wiele sposobów

systemy rekomendacyjne

01 duża ilość produktów

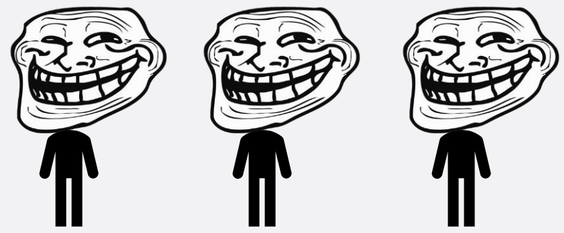
Zazwyczaj ilość produktów jest przytłaczająco większa niż ilość klientów. Co więcej – trudno jest odnaleźć osoby, które kupują DOKŁADNIE to samo

02 średnia ilość aktywnych klientów

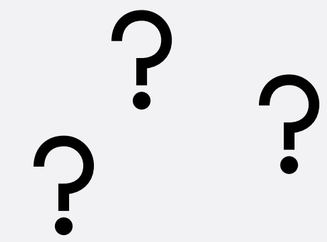
Ilość dóbr nabywanych przez klientów jest zazwyczaj mała w stosunku do całej oferty, a nierzadko ludzie dokonują pojedynczych zakupów w danym sklepie

03 znaczne obciążenie ocen

Problem negatywnych recenzji w Internecie jest powszechnie znany. Ludzie często są bardzo krytyczni, albo po prostu nie chce im się wystawiać pozytywnych opinii.

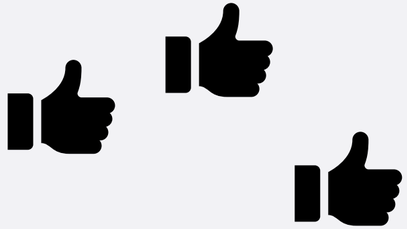


systemy rekomendacyjne



04 problem zimnego startu

Ciężko jest rekomendować cokolwiek nowym użytkownikom, jeśli ich preferencje są nieznane.



05 problem popularności

Niektóre produkty w ofercie są tak popularne, że algorytmy mają skłonność do rekomendowania ich tylko z tego powodu

06 trudność oceny wyników

Na etapie projektowania algorytmu ciężko jest ocenić jakość systemu rekomendacyjnego. Potrzeba interakcji z użytkownikiem

Klasyczne algorytmy

02

Istnieją rozwiązania pozwalające budować systemy rekomendacyjne, cieszące się dużą popularnością od dawna. Używane w nich koncepcje są wykorzystywane także przez autoencodery.

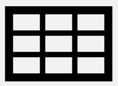
klasyczne algorytmy

podejście modelowe + *content based filtering*

Próba ubrania systemów rekomendacyjnych w klasyczne uczenie maszynowe. Wymaga znajomości **atrybutów** charakteryzujących użytkowników i produkty. Potem łączy się je ze sobą i dokonuje predykcji

collaborative filtering

Do działania tego rodzaju systemów wystarczy **macierz ratingów i miara odległości**. Wyszukuje się wektory podobne do danego użytkownika / produktu, rekomendując elementy, których brakuje w aktualnie przetwarzanym za pomocą odpowiedniego wzoru



model zmiennych latentnych

Podejście oparte na **rozkładzie macierzy** i **analizę własności ukrytych (latentnych).** Rozkład macierzy ma ujawnić niewidoczne połączenia pomiędzy użytkownikami i czynnikami latentnymi oraz produktami a czynnikami latentnymi. Na tej podstawie sugeruje się nowe elementy. Matematyczny rozkład macierzy – np. SVD

klasyczne algorytmy

podejście modelowe + *content based filtering*

Próba ubrania systemów rekomendacyjnych w klasyczne uczenie maszynowe. Wymaga znajomości **atrybutów** charakteryzujących użytkowników i produkty. Potem łączy się je ze sobą i dokonuje predykcji

User feature 1	User feature 2	…	User feature n	Item feature 1	Item feature 2	…	Item feature m	Rating

$$\mathbf{D} = \begin{pmatrix} & X_1 & X_2 & \cdots & X_d \\ \mathbf{x}_1 & x_{11} & x_{12} & \cdots & x_{1d} \\ \mathbf{x}_2 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_n & x_{n1} & x_{n2} & \cdots & x_{nd} \end{pmatrix}$$

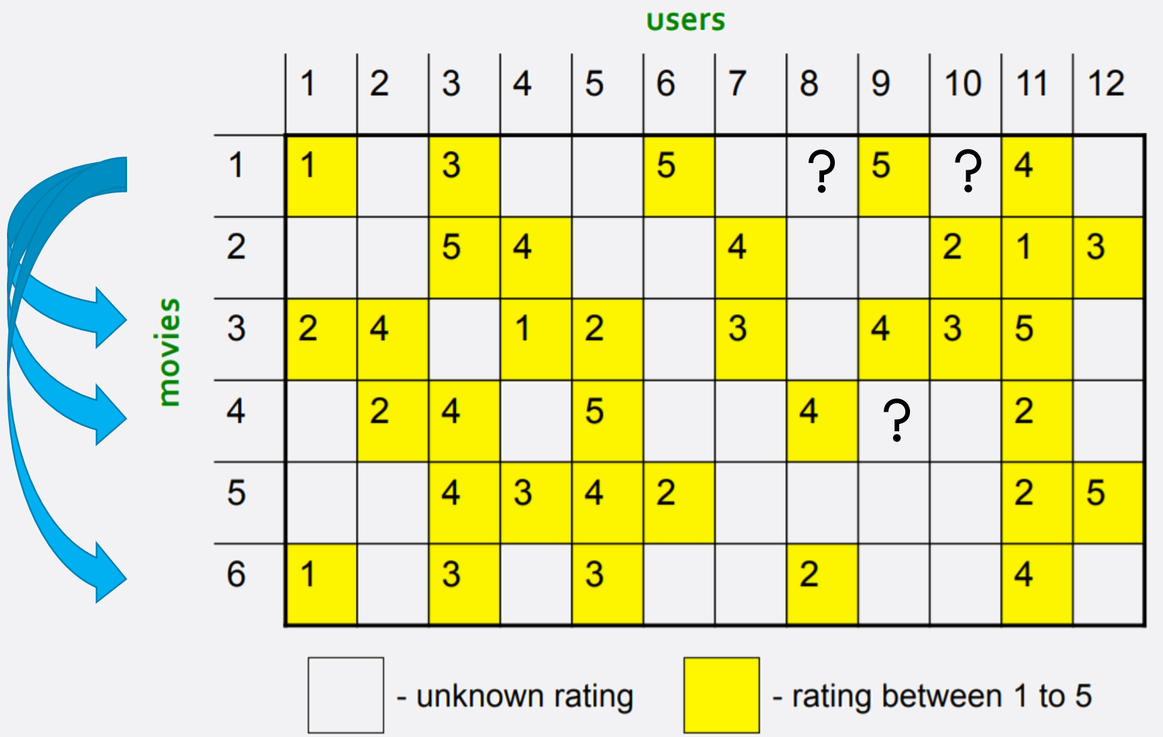
klasyczne
algorytmy

collaborative filtering

Do działania tego rodzaju systemów wystarczy **macierz ratingów i miara odległości**. Wyszukuje się wektory podobne do danego użytkownika / produktu, rekomendując elementy, których brakuje w aktualnie przetwarzanym

KNN

$$s_{ij}(x_i, x_j) = \text{cosine}(\theta)$$



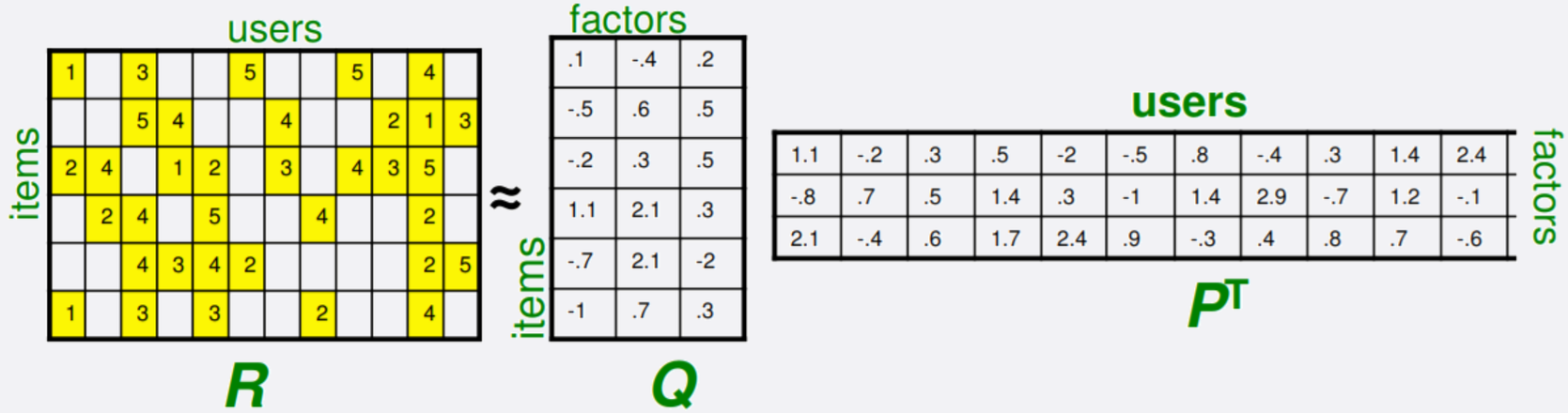
$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

Rajaraman, A. and Ullman, J.D., 2011. *Mining of massive datasets*. Cambridge University Press.

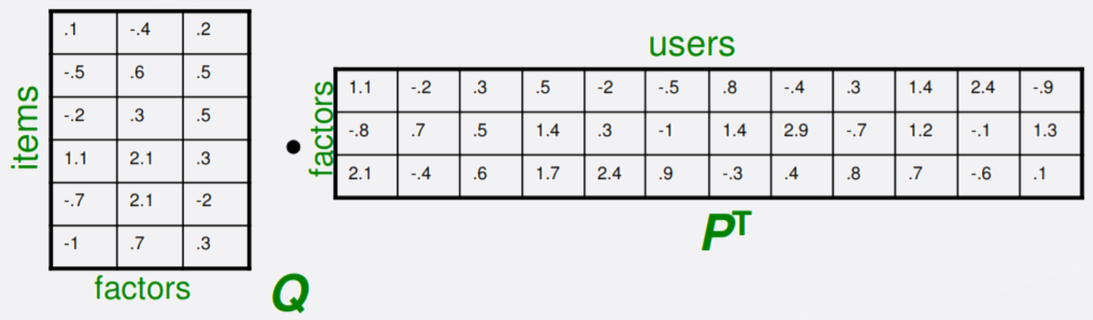
klasyczne
algorytmy

model zmiennych latentnych

Podejście oparte na **rozkładzie macierzy** i **analizę własności ukrytych (latentnych).** Rozkład macierzy ma ujawnić niewidoczne połączenia pomiędzy użytkownikami i czynnikami latentnymi oraz produktami a czynnikami latentnymi. Na tej podstawie sugeruje się nowe elementy. Matematyczny rozkład macierzy – np. SVD



$$r =$$



Rajaraman, A. and Ullman, J.D., 2011. *Mining of massive datasets*. Cambridge University Press.

03

Sieci neuronowe o odpowiedniej strukturze mogą zostać wykorzystane do odwzorowania zmiennych latentnych. Od dawana jest to stosowane np. przy kompresji obrazów

autoencodery

Autoencodery rekomendacyjne

AutoRec: Autoencoders Meet Collaborative Filtering

Suvash Sedhain[†*], Aditya Krishna Menon[†*], Scott Sanner[†*], Lexing Xie[*†]
[†] NICTA, [*] Australian National University
suvash.sedhain@anu.edu.au, { aditya.menon, scott.sanner }@nicta.com.au,
lexing.xie@anu.edu.au

ABSTRACT

This paper proposes AutoRec, a novel autoencoder framework for collaborative filtering (CF). Empirically, AutoRec's compact and efficiently trainable model outperforms state-of-the-art CF techniques (biased matrix factorization, RBM-CF and LLORMA) on the Movielens and Netflix datasets.

Categories and Subject Descriptors D.2.8 [**Information Storage and Retrieval**]Information Filtering

Keywords Recommender Systems; Collaborative Filtering; Autoencoders

1. INTRODUCTION

Collaborative filtering (CF) models aim to exploit information about users' preferences for items (e.g. star ratings) to provide personalised recommendations. Owing to the Netflix challenge, a panoply of different CF models have been proposed, with popular choices being matrix factorisation [1, 2] and neighbourhood models [5]. This paper proposes *AutoRec*, a new CF model based on the autoencoder paradigm; our interest in this paradigm stems from the recent successes of (deep) neural network models for vision and speech tasks. We argue that AutoRec has representational and computational advantages over existing neural approaches to CF [4], and demonstrate empirically that it outperforms the current state-of-the-art methods.

2. THE AUTOREC MODEL

In rating-based collaborative filtering, we have m users,

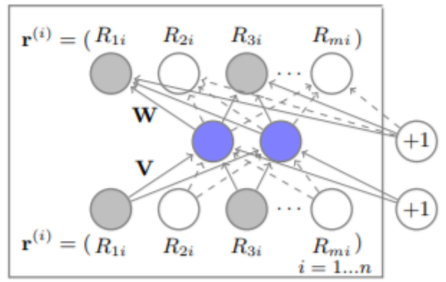


Figure 1: Item-based AutoRec model. We use plate notation to indicate that there are n copies of the neural network (one for each item), where $\mathbf{W}$ and $\mathbf{V}$ are tied across all copies.

where $h(\mathbf{r};\theta)$ is the *reconstruction* of input $\mathbf{r} \in \mathbb{R}^d$,

$$h(\mathbf{r};\theta) = f\left(\mathbf{W} \cdot g(\mathbf{V}\mathbf{r} + \boldsymbol{\mu}) + \mathbf{b}\right)$$

for *activation functions* $f(\cdot), g(\cdot)$. Here, $\theta = \{\mathbf{W}, \mathbf{V}, \boldsymbol{\mu}, b\}$ for transformations $\mathbf{W} \in \mathbb{R}^{d \times k}, \mathbf{V} \in \mathbb{R}^{k \times d}$, and biases $\mu \in \mathbb{R}^k, \mathbf{b} \in \mathbb{R}^d$. This objective corresponds to an auto-associative neural network with a single, k-dimensional hidden layer. The parameters θ are learned using backpropagation.

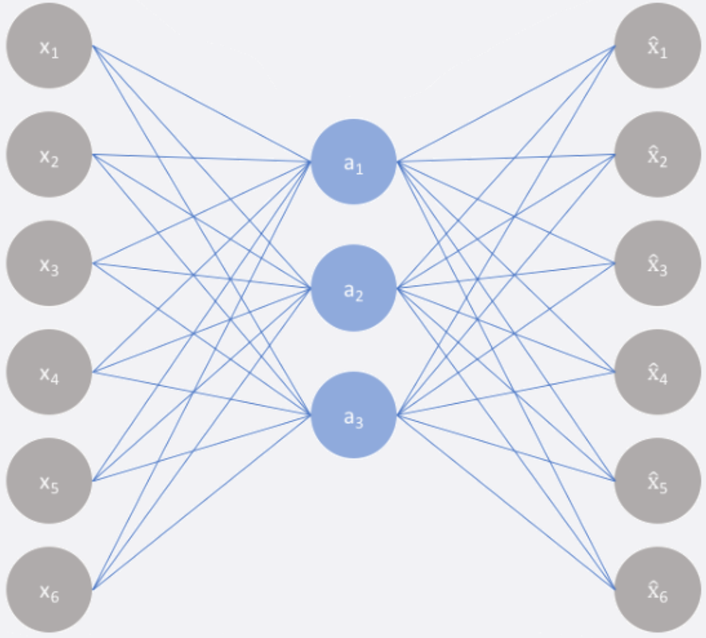
The item-based AutoRec model, shown in Figure 1, applies an autoencoder as per Equation 1 to the set of vectors $\{\mathbf{r}^{(i)}\}_{i=1}^n$, with two important changes. First, we account for the fact that each $\mathbf{r}^{(i)}$ is partially observed by only updating during backpropagation those weights that are associated with observed inputs, as is common in matrix factorisation

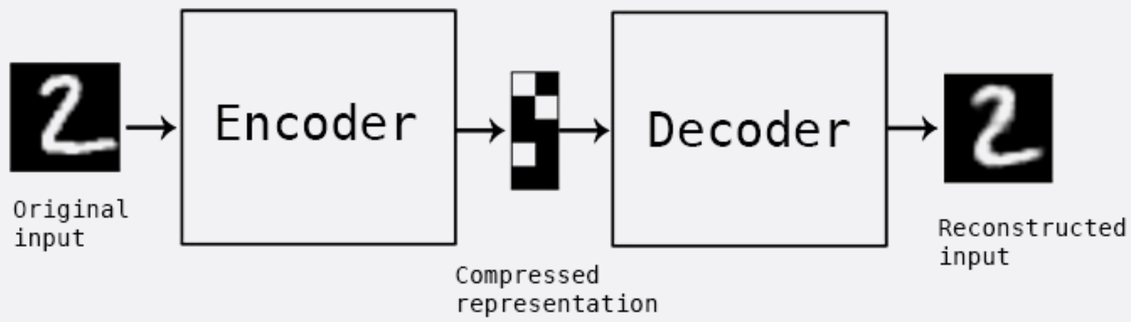
2015

autoencodery
działanie

odwzorowanie wejścia na wyjściu

Autoencodery przyjmują wejście i **odwzorowują je na wyjściu**. Nie ma więc klasycznej klasyfikacji ani regresji – chodzi o odtworzenie. Klasycznym przykładem wykorzystania jest odszumianie obrazu.



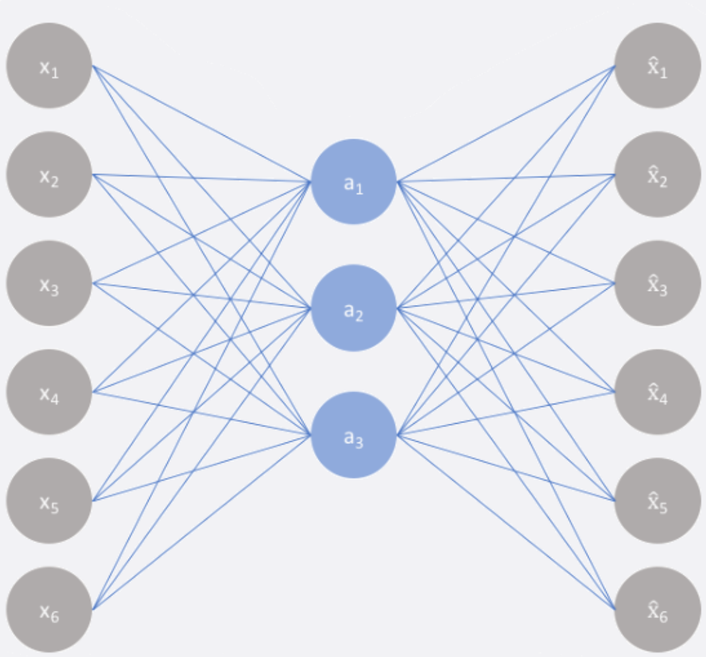


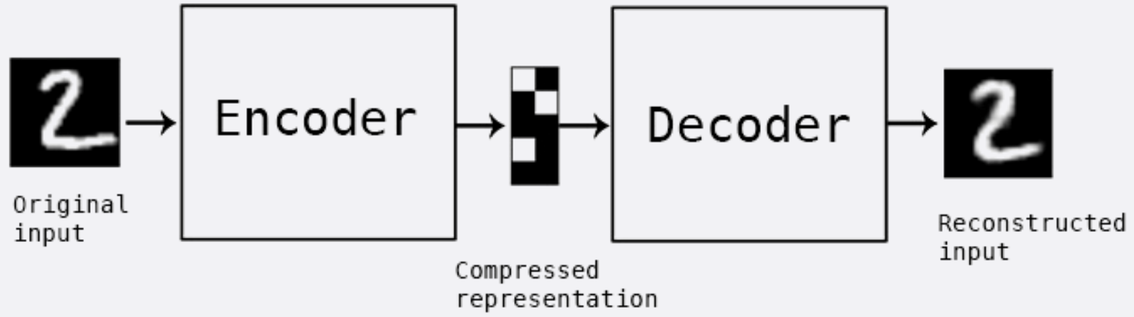
https://blog.keras.io/building-autoencoders-in-keras.html

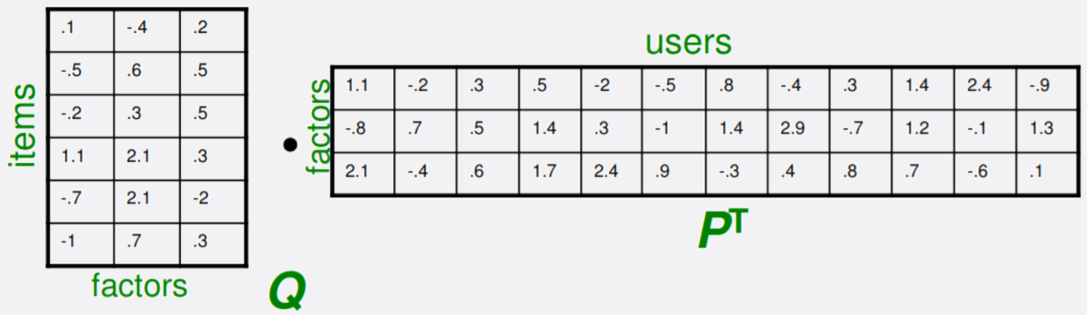
autoencodery
działanie

kompresja wewnętrznej reprezentacji

Podczas przetwarzania wejścia w wyjście, autoencoder dokonuje **kompresji.** Jest to tożsame z dokonaniem nieliniowej redukcji wymiarowości. Taki skompresowany wymiar można interpretować jako **zmienne latentne!**





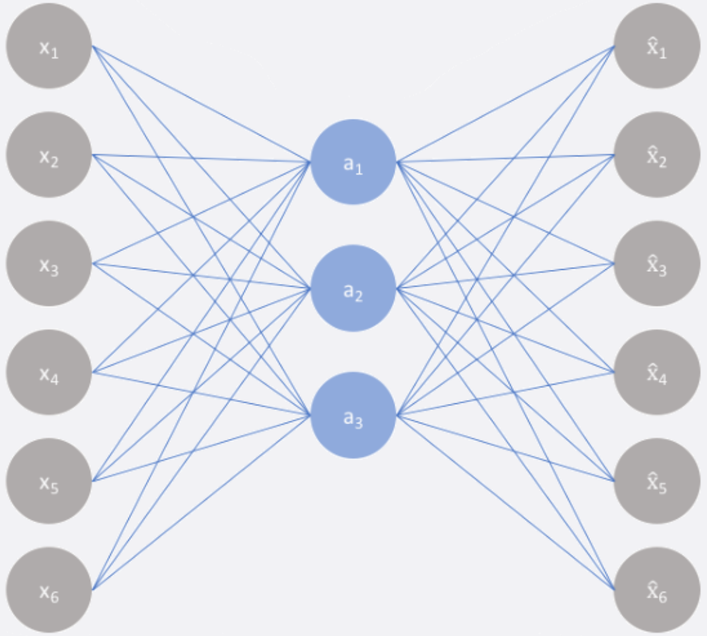


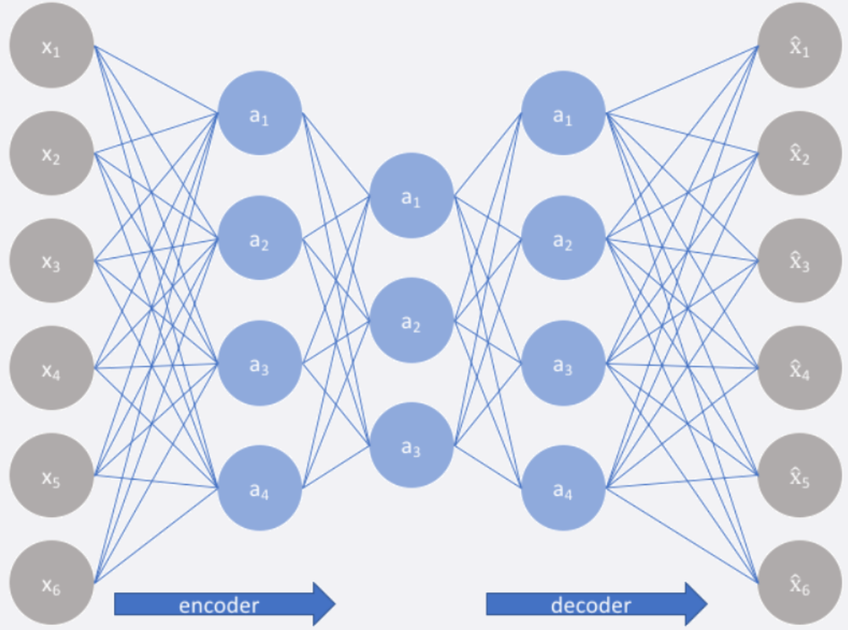
https://blog.keras.io/building-autoencoders-in-keras.html

autoencodery
działanie

elastyczna architektura

Autoencodery mogą przyjmować dowolną postać – od prostych sieci z jedną warstwą ukrytą, do sieci głębokich z wieloma warstwami kompresji, aż po *deep stacked autoencoders* (autoencodery zestawione niezależnie w jedną sieć)





https://www.jeremyjordan.me/autoencoders/

autoencodery
działanie

proste i elastyczne szkolenie

Autoencodery mogą być szkolone jak **zwyczaje sieci głębokie,** bądź też z użyciem **greedy layer-wise pretraining**, gdzie każdy autoencoder składowy jest szkolony samodzielnie, a potem „zestawiany” do wspólnego układu.

Można też szkolić autoencodery z użyciem współdzielonych wag na wejściu i wyjściu

l – *loss function*
o – *wyjście autoencodera*
$\hat{a}$ – *aktywacja na wyjściu autoencodera*
$h(x)$ – *aktywacja na wejściu autoencodera*
W – *wagi (współdzielone)*

$$o(\hat{a}(x)) = o(c + W^T h(x)) = o(c + W^T \sigma(b + Wx))$$

$$\frac{\partial l}{\partial W_{ij}} = \frac{\partial l}{\partial \hat{a}_j}\frac{\partial \hat{a}_j}{\partial W_{ij}} = \frac{\partial l}{\partial \hat{a}_j}(h_i + W_{ij}\frac{\partial h_i}{\partial W_{ij}}) = \frac{\partial l}{\partial \hat{a}_j}h_i + \frac{\partial l}{\partial a_i}x_j$$

autoencodery
działanie

schemat treningu autoencodera

W przypadku systemu rekomendacyjnego sama **rekonstrukcja macierzy** ze zmiennych latentnych jest zadaniem pośrednim.
Prawdziwym celem jest sprawdzenie możliwości generalizacji sytemu.

Błąd rekonstrukcji – mówi o stopniu „zrozumienia” zmiennych latentnych.
Błąd predykcji – ocenia trafność predykcji brakujących ratingów (rekomendacja)

User \ Movie	M1	M2	…	Mn
U1	5.0	3.0		-
U2	3.5	-		3.5
U3	4.0	4.5		-
U4	1.0	1.5		5.0
U5	3.5	5.0		4.0
U6	3.5	4.5		3.5

autoencodery
działanie

Dane treningowe
rekonstrukcja

User \ Movie	M1	M2	…	Mn
U1	5.0	3.0		-
U2	3.5	-		3.5

Dane walidacyjne
rekonstrukcja

U3	4.0	4.5		-
U4	1.0	1.5		5.0

Dane testowe
rekonstrukcja
predykcja

U5	**?**	5.0		**?**
U6	3.5	**?**		3.5

autoencodery
działanie

W_{ih}, W_{ho} $-$ *wagi na wejściu/wyściu*
b_{in}, b_{out} $-$ *bias wejścia*
θ $-$ *zestaw wszystkich parmetrów autoencodera (W, b)*
$g(x)$ $-$ *nieliniowa funkcja aktywacji warstwy wewnętrznej*
$f\ (z)$ $-$ *funkcja aktywacji na wyjściu*
$\boldsymbol{r}$ $-$ *macierz* **obserwowalnych** *(istniejących w zbiorze) ratingów*
$\boldsymbol{y}$ $-$ *wektor* **"ukrytych" ratingów** ze zbioru testowego
$h(\boldsymbol{r}; \boldsymbol{\theta})$ $-$ *zapis autoencodera jako funkcji ratingów i parmetrów*

$$\mathrm{h}(\mathbf{r}; \theta) = \mathrm{f}(W_{out} \cdot g(W_{in} \cdot \boldsymbol{r} + b_{in}) + b_{out})$$

Dane treningowe
rekonstrukcja

Dane walidacyjne
rekonstrukcja

Dane testowe
rekonstrukcja
predykcja

$$\mathcal{L}_1\big(\boldsymbol{r}, h(\boldsymbol{r}; \theta)\big) = \mathcal{L}_1(\boldsymbol{r}, \hat{\boldsymbol{r}}) = \frac{1}{|\boldsymbol{r}|}\sqrt{\sum_{i=1}^{r}(r_i - \hat{r}_i)}$$

błąd rekonstrukcji: **MSE**

$$\mathcal{L}_2(\boldsymbol{y}, \hat{\boldsymbol{y}})$$

błąd predykcji: **dowolny**

Goodfellow, I., Bengio, Y. i Courville, A. (2016) *Deep Learning*. MIT Press.

autoencodery
działanie

Dane treningowe
rekonstrukcja

Dane walidacyjne
rekonstrukcja

Dane testowe
rekonstrukcja
predykcja

$$\mathcal{L}_1\big(\boldsymbol{r}, h(\boldsymbol{r};\theta)\big) = \mathcal{L}_1(\boldsymbol{r}, \hat{\boldsymbol{r}}) = \frac{1}{|\boldsymbol{r}|}\sqrt{\sum_{i=1}^{\boldsymbol{r}}(r_i - \hat{r}_i)}$$

$$\mathcal{L}_2(\boldsymbol{y}, \hat{\boldsymbol{y}})$$

błąd rekonstrukcji: **MSE**

$\boldsymbol{r}$

User \ Movie	M1	M2	…	Mn
U1	5.0	3.0		-
U2	3.5	-		3.5

$\hat{\boldsymbol{r}}$

User \ Movie	M1	M2	…	Mn
U1	4.874	4.001		-
U2	3.222	-		3.654

błąd predykcji: **dowolny**

U5	**?**	5.0		**?**
U6	3.5	**?**		3.5

$$\boldsymbol{y} = [3.5, 4.5, 4.0\ ]$$

$$\hat{\boldsymbol{y}} = [3.223, 4.894, 3.999\ ]$$

Goodfellow, I., Bengio, Y. i Courville, A. (2016) *Deep Learning*. MIT Press.


ale_setting = FloatProperty(
name="Scale",
min=0.01, max=1000.0,
default=1.0,
)
execute(self, context):
# get the folder
folder_path = (os.path.dirname(self.filepath))
# get objects selected in the viewport
viewport_selection = bpy.context.selected_objects
# get export objects
obj_export_list = viewport_selection
if self.use_selection_setting == False:
obj_export_list = [i for i in bpy.context.scene.objects]
deselect all objects
py.ops.object.select_all(action='DESELECT')
or item in obj_export_list:
item.select = True
if item.type == 'MESH':
file_path = os.path.join(folder_path, "{}.obj".format(ite
bpy.ops.export_scene.obj(filepath=file_path, use_selectio
axis_forward=self.axis_forward_
axis_up=self.axis_up_setting,
use_animation=self.use_animat
use_mesh_modifiers=self.use
use_edges=self.use_edges
use_smooth_groups=self
use_smooth_groups_bi
use_normals=self
use_uvs=self

demo

Dziękuję za uwagę