Lecture 5

Корбут Даниил
Deep Learning Research Engineer, Insilico Medicine

telegram: @rtriangle
vk: rtriangle
email: korbut.daniel@gmail.com

План занятия

- Напоминание прошлой лекции
- Временные ряды и NN
- Рекомендательные системы и NN
- Кластеризация и методы понижения размерности
- Ранжирование
- Summary курса

Sequence labeling

Problem

- Given a sequence of *tokens*, infer the most probable sequence of *labels* for these tokens.

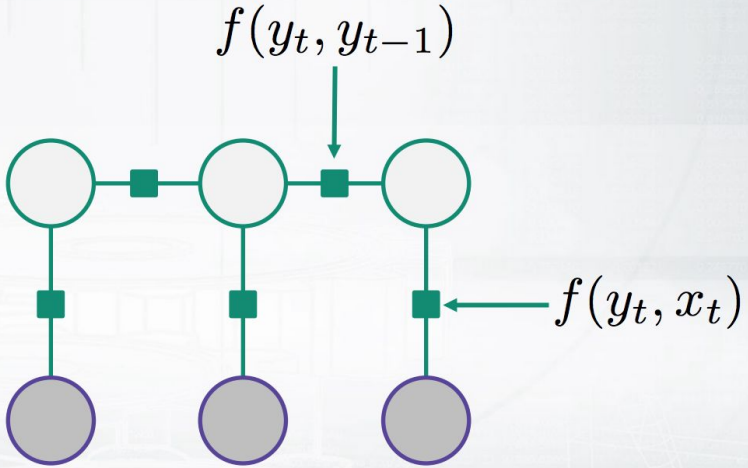
Examples

- part of speech tagging
- named entity recognition

Conditional Random Field (linear chain)

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t) \right)$$

Undirected graph:



https://www.coursera.org/learn/language-processing

Features engineering

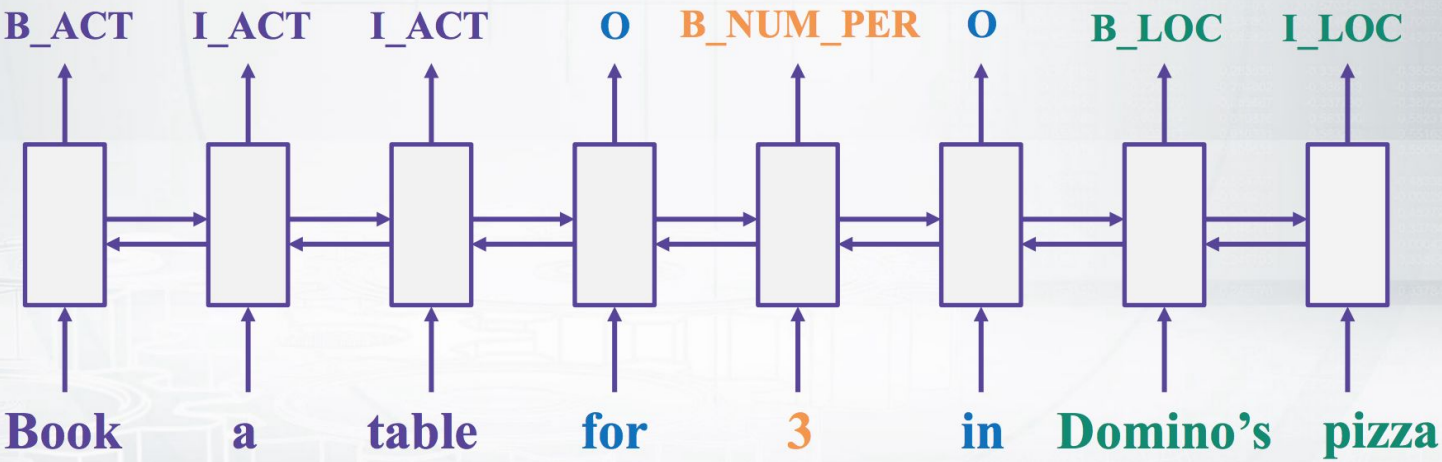
Label-observation features:

- $f(y_t, y_{t-1}, x_t) = [y_t = y]\, g_m(x_t)$
- $f(y_t, y_{t-1}, x_t) = [y_t = y][y_{t-1} = y']$
- $f(y_t, y_{t-1}, x_t) = [y_t = y][y_{t-1} = y']\, g_m(x_t)$

https://www.coursera.org/learn/language-processing

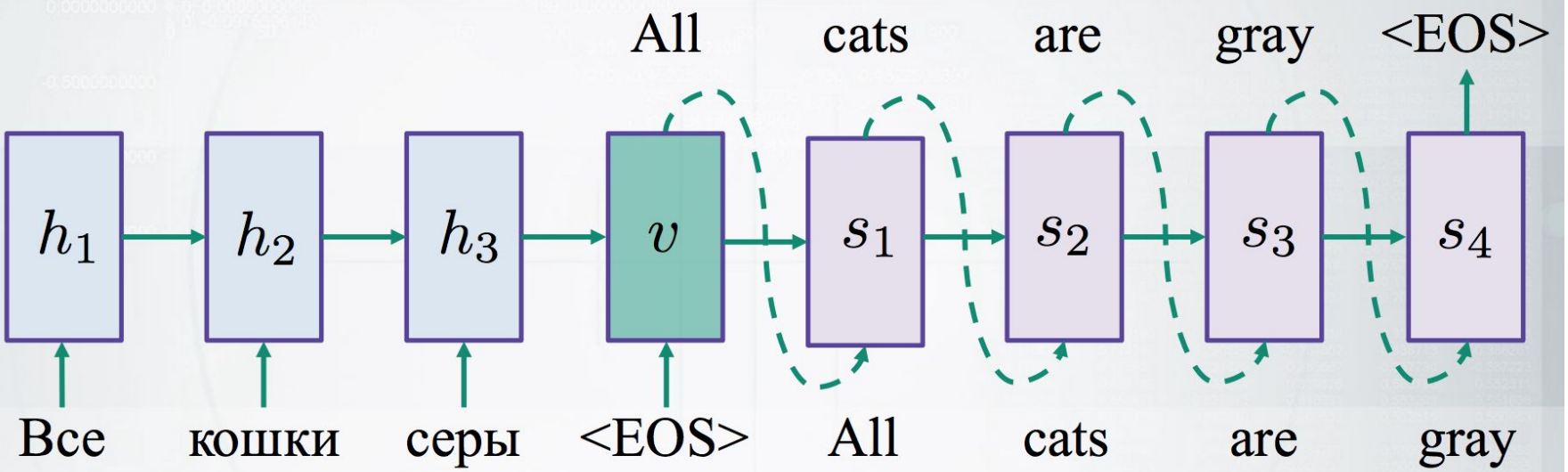
Bi-directional LSTM

- Universal approach for sequence tagging
- You can stack several layers + add linear layers on top
- Trained by cross-entropy loss coming from each position



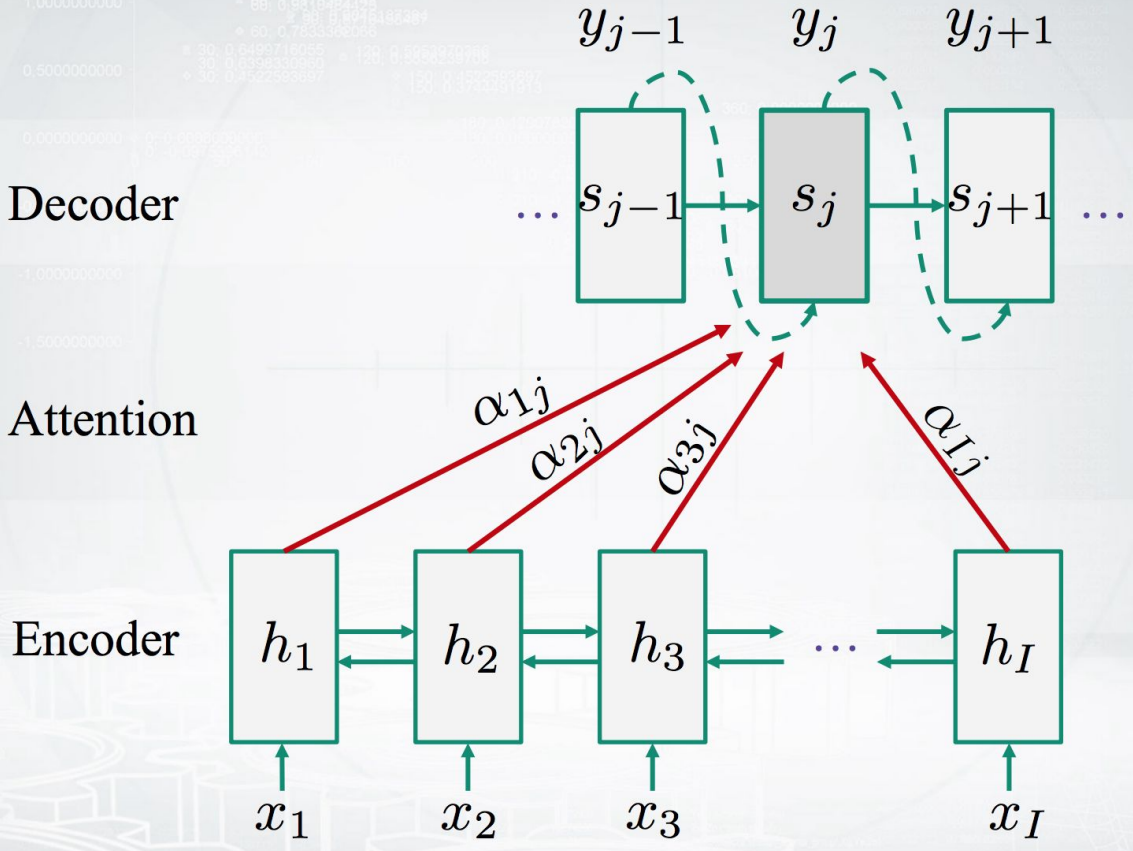
https://www.coursera.org/learn/language-processing

Sequence to sequence



https://www.coursera.org/learn/language-processing

Attention mechanism

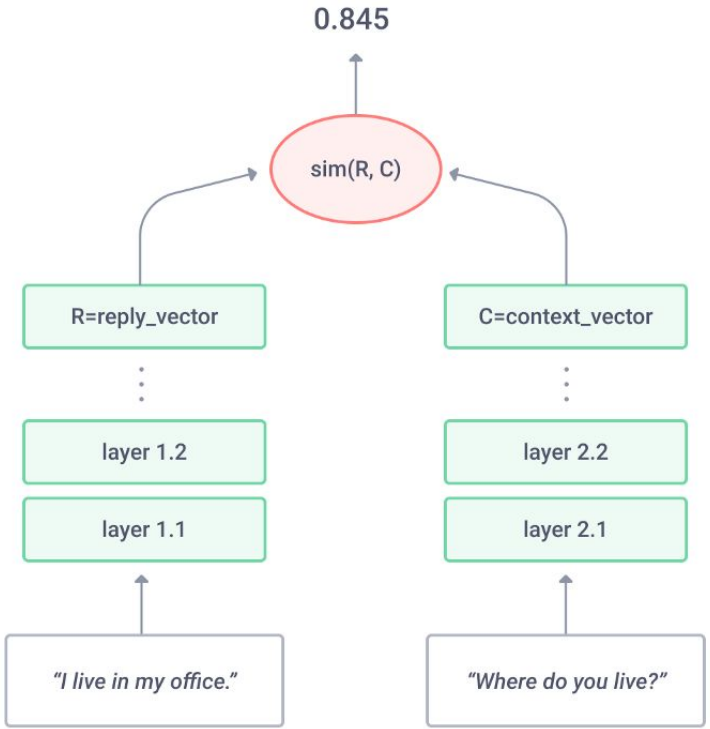


Bahdanau et. al - Neural Machine Translation by jointly learning to align and translate, 2015.

https://www.coursera.org/learn/language-processing

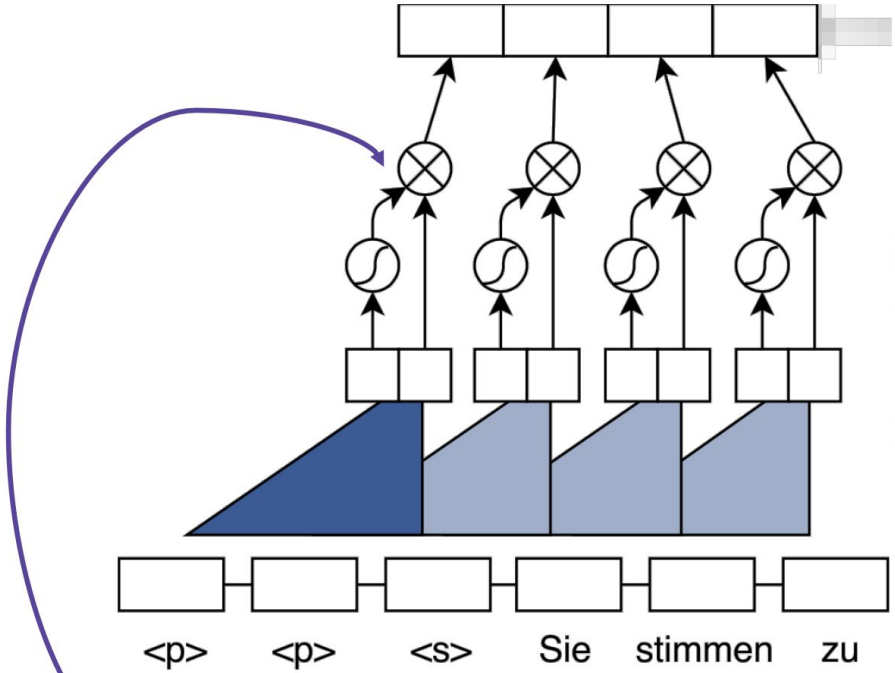
Selective models

Instead of estimating probability p(reply | context; w), selective models learn similarity function—sim(reply, context; w), where a reply is one of the elements in a predefined pool of possible answers.



$$L = \max\left(0, sim(\texttt{ctx}, \texttt{reply}_{wrong}) - sim(\texttt{ctx}, \texttt{reply}_{correct}) + \alpha\right) \to \min$$

CNN for sequences: Gated Linear Unit



Stacking 6 layers with kernel size 5 results in an input field of 25 elements

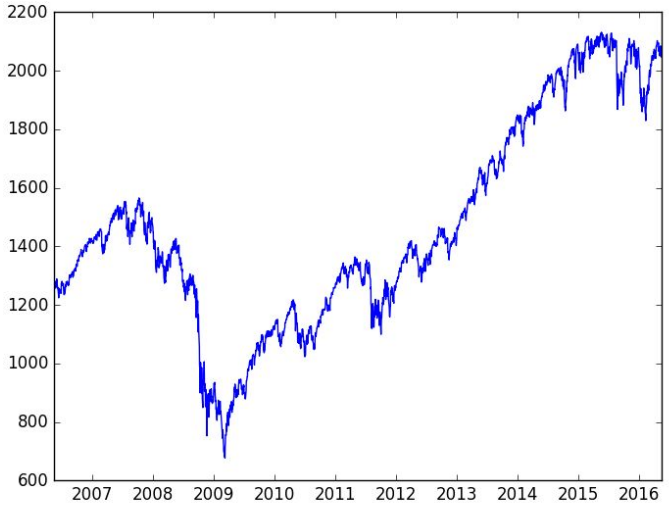
$$h_l(\mathbf{X}) = (\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c})$$

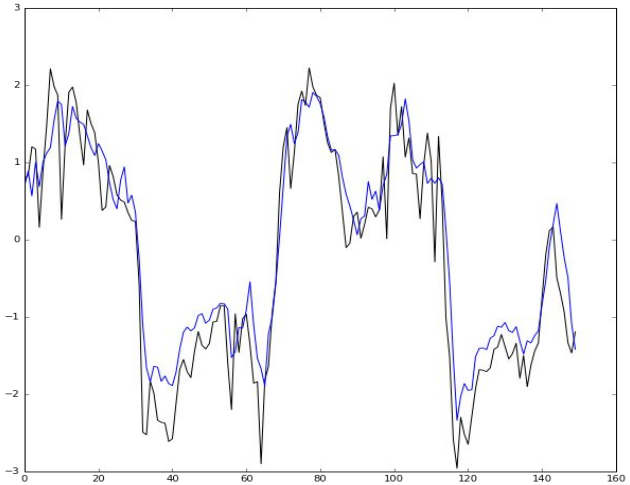
https://arxiv.org/pdf/1612.08083.pdf https://arxiv.org/pdf/1705.03122.pdf

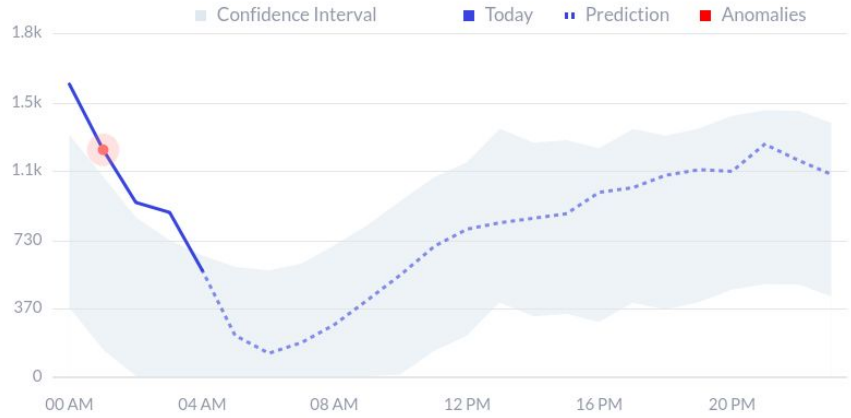
https://www.coursera.org/learn/language-processing

Time series

- Предсказание следующего значения: регрессия в случае точного показателя, классификация в случае определения направления изменения цены
- Поиск различных аномалий







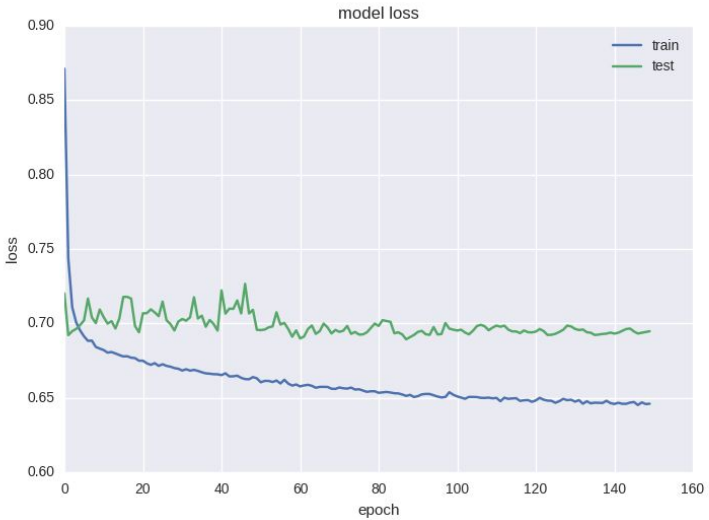
1. Часто при решении задачи классификации (движении цены) стоит решать задачу регрессии
2. Есть несколько способов предсказания на период t+k:
   a. если нужно предсказать на небольшой период вперёд, по 1 модели на каждый shift
   b. одна модель с параметром shift
   c. одна модель на shift=1 и предсказание на основе своих предсказаний
3. Проблема: не стационарный ряд (mean, var, max, min). Тест Дики-Фуллера для проверки на стационарность

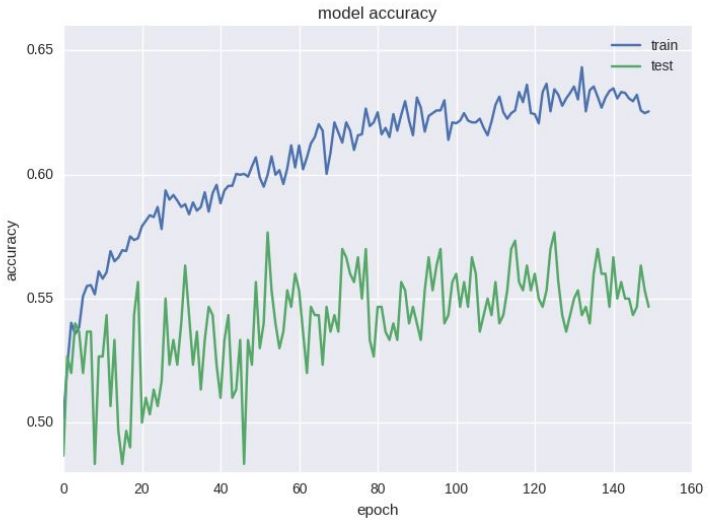
https://medium.com/@alexrachnog/neural-networks-for-algorithmic-trading-part-one-simple-time-series-forecasting-f992daa1045a

Как можно предотвратить переобучение?

```
model = Sequential()
model.add(Dense(64, input_dim=30))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
```

Первая попытка: обучим сеть на классификацию повышения/понижения с loss cross_entropy

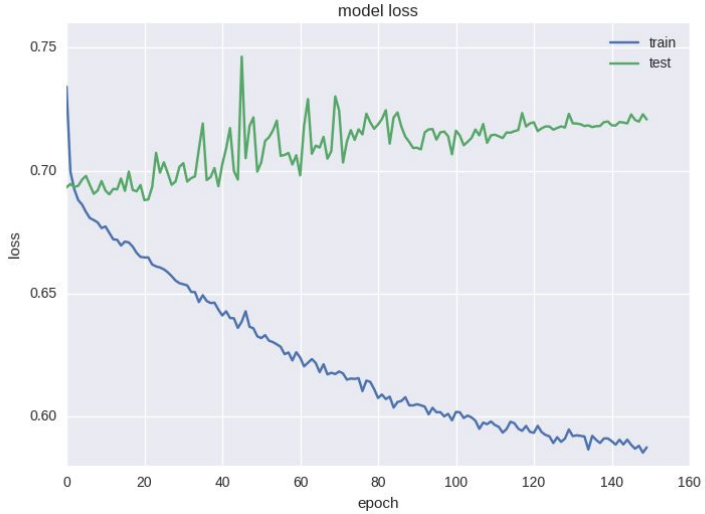


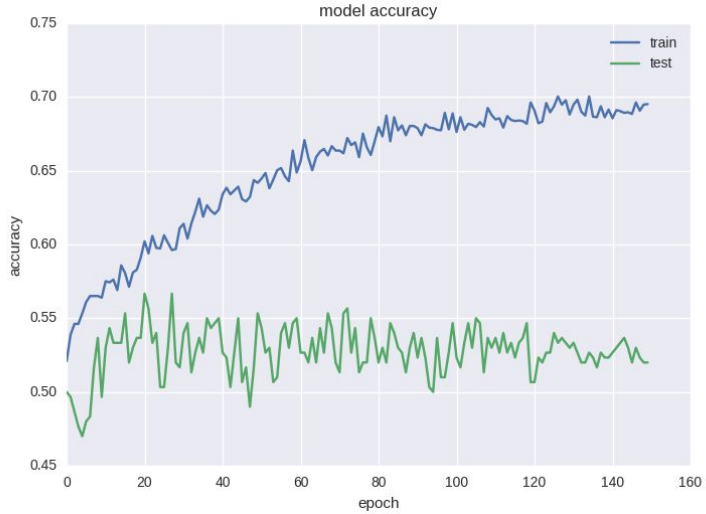


Как можно предотвратить переобучение?

```
model = Sequential()
model.add(Dense(64, input_dim=30))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(16))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2)) model.add(Activation('softmax'))
```

Второй шаг: сделаем сеть глубже и посмотрим на результат

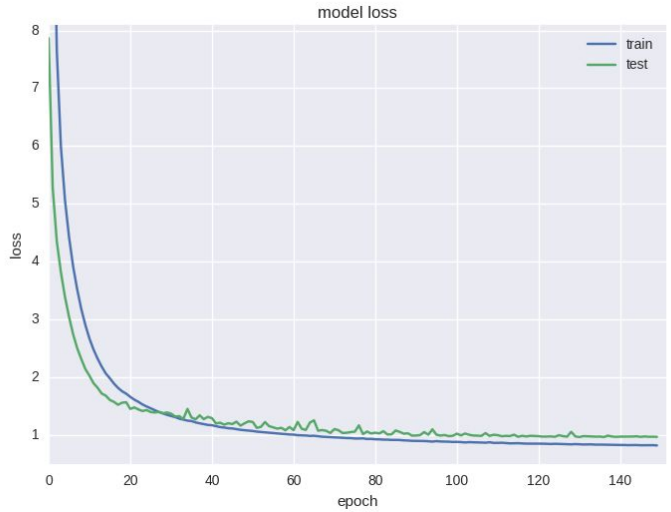


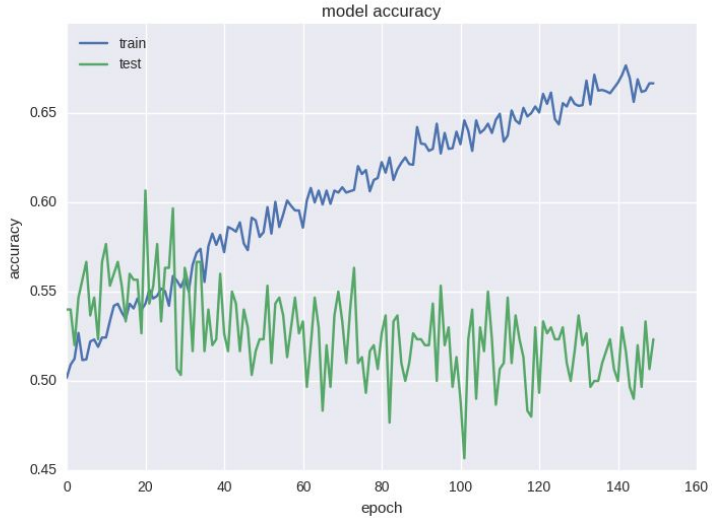


Как можно предотвратить переобучение?

```
model = Sequential()
model.add(Dense(64, input_dim=30,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(16,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
```

Третий шаг: добавим регуляризацию.
Проблема: лосс падает, качество не растёт

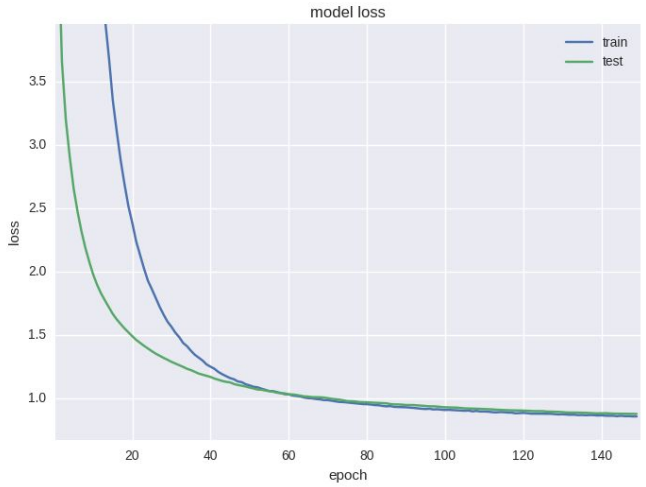


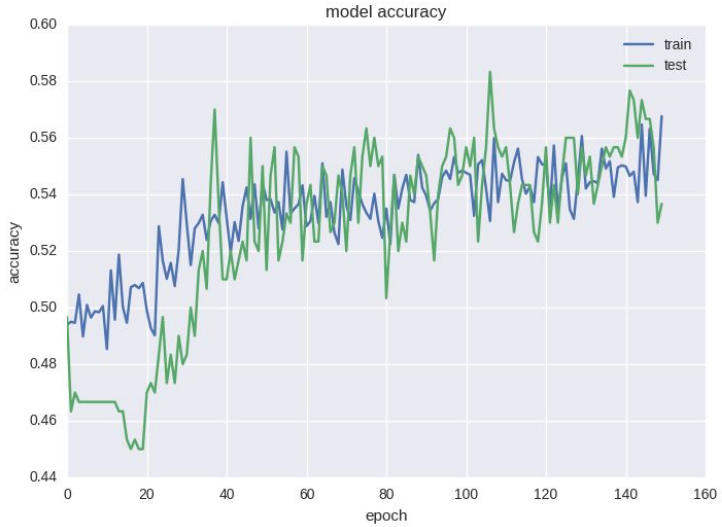


Как можно предотвратить переобучение?

```
model = Sequential()
model.add(Dense(64, input_dim=30,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dropout(0.5))
model.add(Dense(16,
activity_regularizer=regularizers.l2(0.01)))
model.add(BatchNormalization())
model.add(LeakyReLU())
model.add(Dense(2))
model.add(Activation('softmax'))
```

Четвёртый шаг: добавим дропаут, получили 58% accuracy





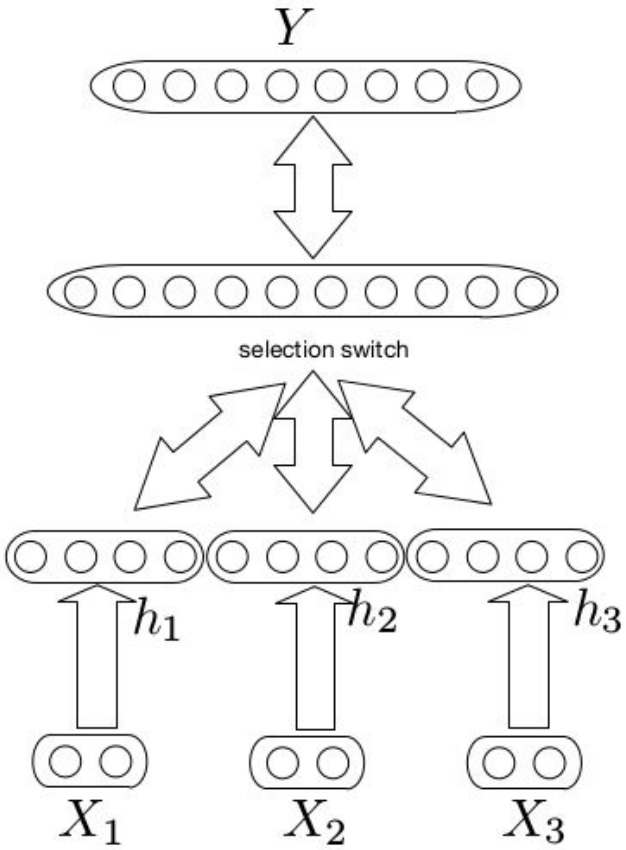
Как ещё можно обучаться и предсказывать TS?

```
main_input = Input(shape=(len(X[0]), ), name='main_input')
x = GaussianNoise(0.05)(main_input)
x = Dense(64, activation='relu')(x)
x = GaussianNoise(0.05)(x)
output = Dense(1, activation = "linear", name = "out")(x)


final_model = Model(inputs=[main_input], outputs=[output])
opt = Adam(lr=0.002)
final_model.compile(optimizer=opt, loss='mse')
```

“Novel” point here is adding small noise to the input and to the output of the single layer of our neural network. It can work very similar to L2 regularization. Proof here.

Как ещё можно обучаться и предсказывать TS?

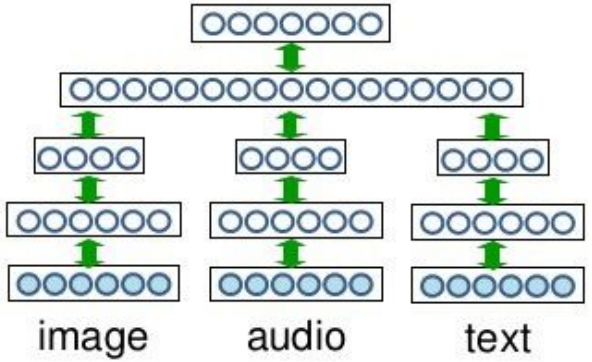


Multimodal learning model

Motivation: Multi-modal learning

Single learning algorithms that combine multiple input domains

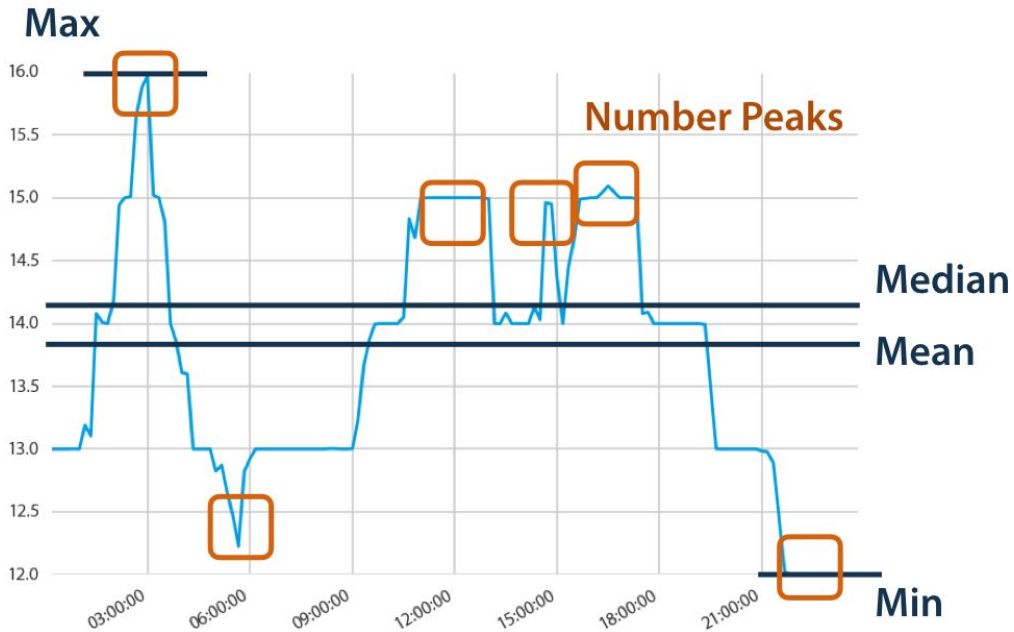
- Images
- Audio & speech
- Video
- Text
- Robotic sensors
- Time-series data
- Others



https://becominghuman.ai/neural-networks-for-algorithmic-trading-multimodal-and-multitask-deep-learning-5498e0098caf

Полезные библиотеки для Time Series

Tsfresh - позволяет автоматически генерировать дополнительные признаки для временных рядов: квантили, локальные максимумы, скользящие средние, дисперсии, количество пиков и т.п.



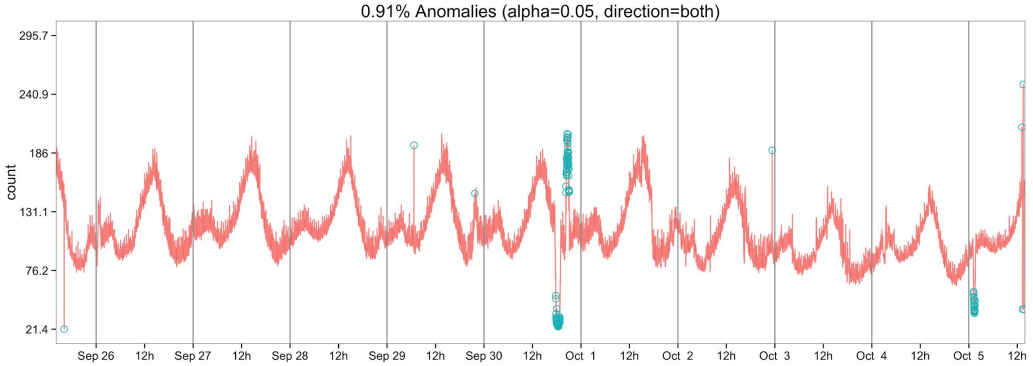
https://github.com/blue-yonder/tsfresh

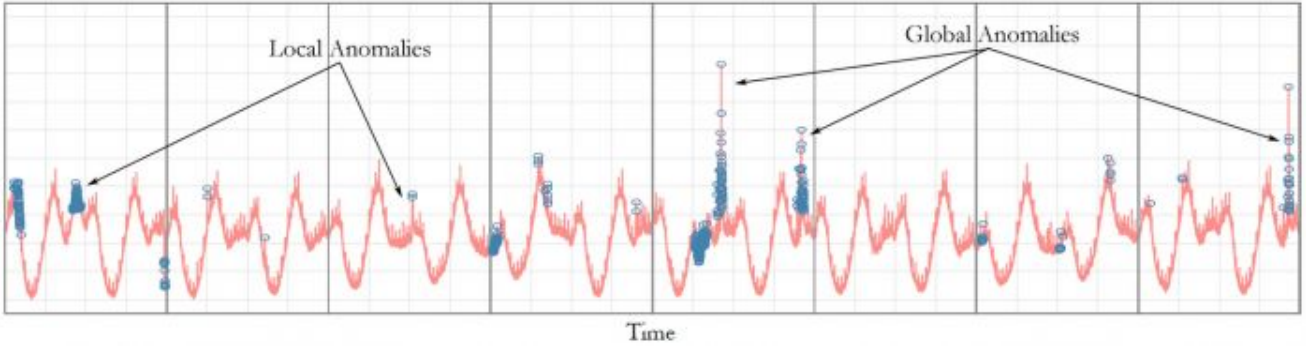
https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html

Полезные библиотеки для Time Series

Anomaly detection from Twitter - готовое решение для поиска аномалий во временных рядах, есть настройки разной чувствительности.

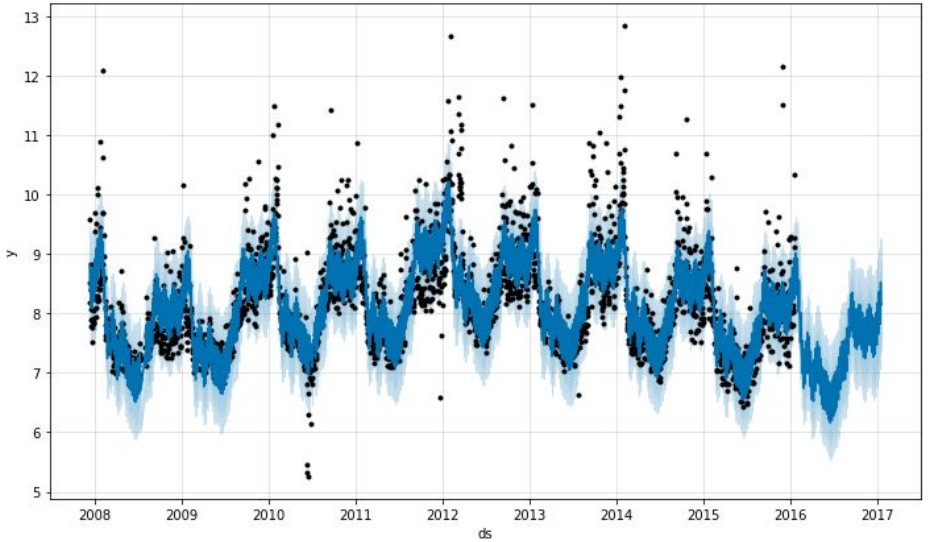
https://github.com/twitter/AnomalyDetection

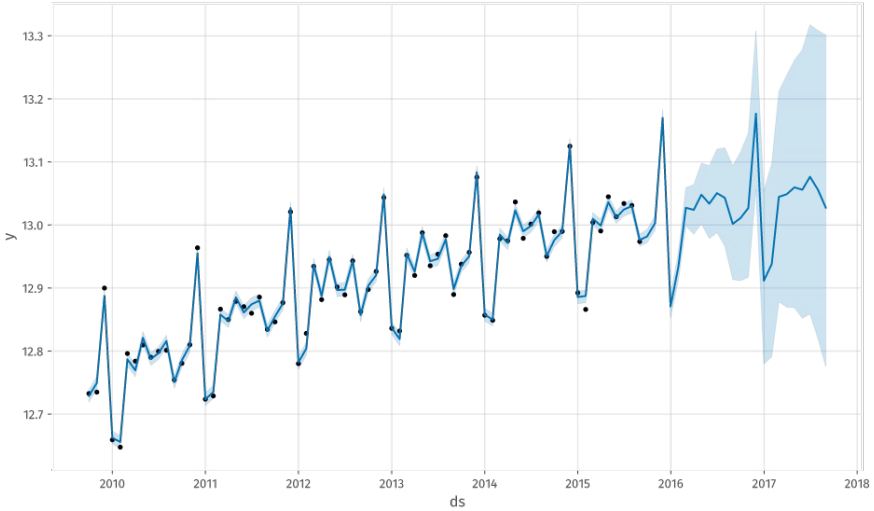




Полезные библиотеки для Time Series

Prophet - библиотека facebook для предсказания временных рядов. Неплохо работает из коробки.



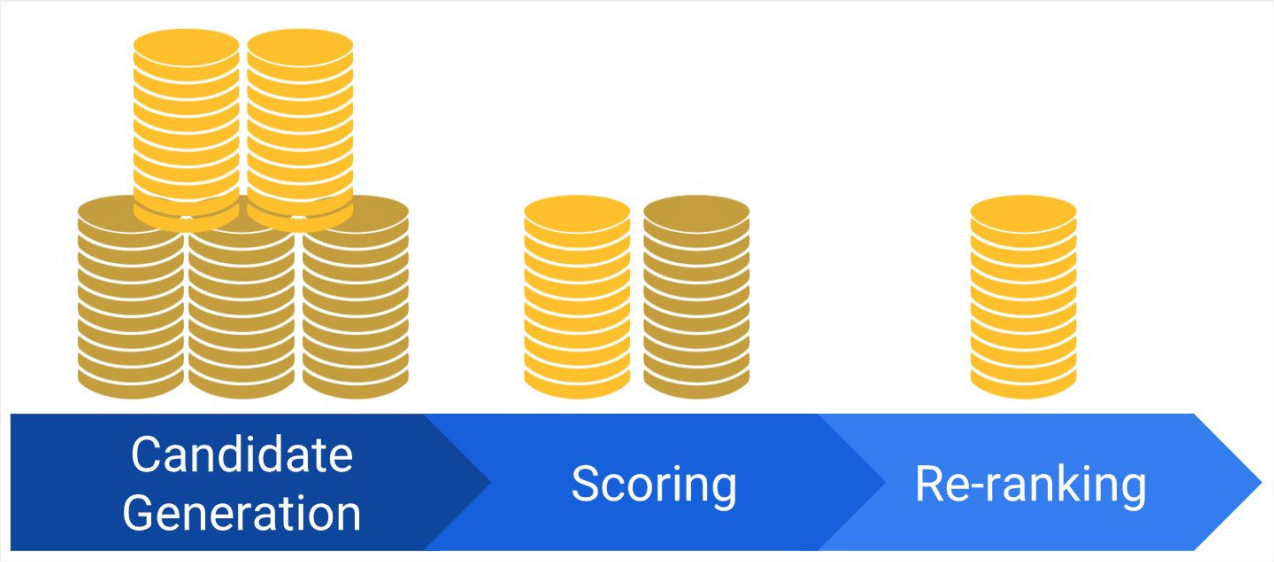


https://facebook.github.io/prophet/

Recommender systems

Современные рекомендательные системы состоят из 3-х основных компонент:

- candidate generation
- scoring
- re-ranking



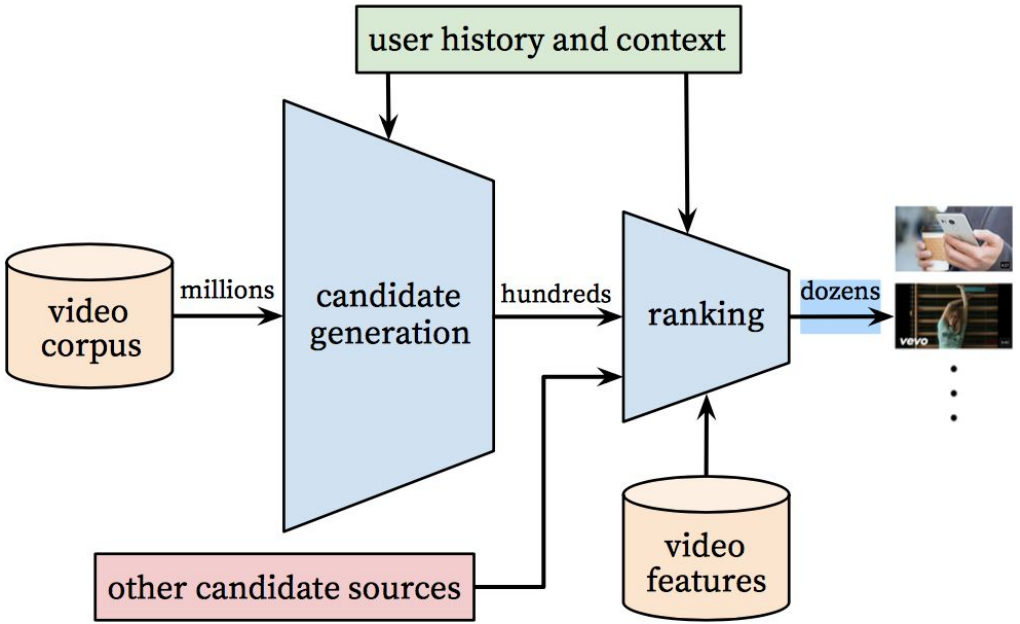
https://developers.google.com/machine-learning/recommendation

Recommender systems

Candidate generation - the system starts from a potentially huge corpus and generates a much smaller subset of candidates.

Scoring - another model scores and ranks the candidates in order to select the set of items (on the order of 10) to display to the user.

Re-ranking - must take into account additional constraints for the final ranking.



https://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/45530.pdf

Candidate Generation

Type	Definition	Example
content-based filtering	Uses *similarity between items* to recommend items similar to what the user likes.	If user A watches two cute cat videos, then the system can recommend cute animal videos to that user.
collaborative filtering	Uses *similarities between queries and items simultaneously* to provide recommendations.	If user A is similar to user B, and user B likes video 1, then the system can recommend video 1 to user A (even if user A hasn't seen any videos similar to video 1).

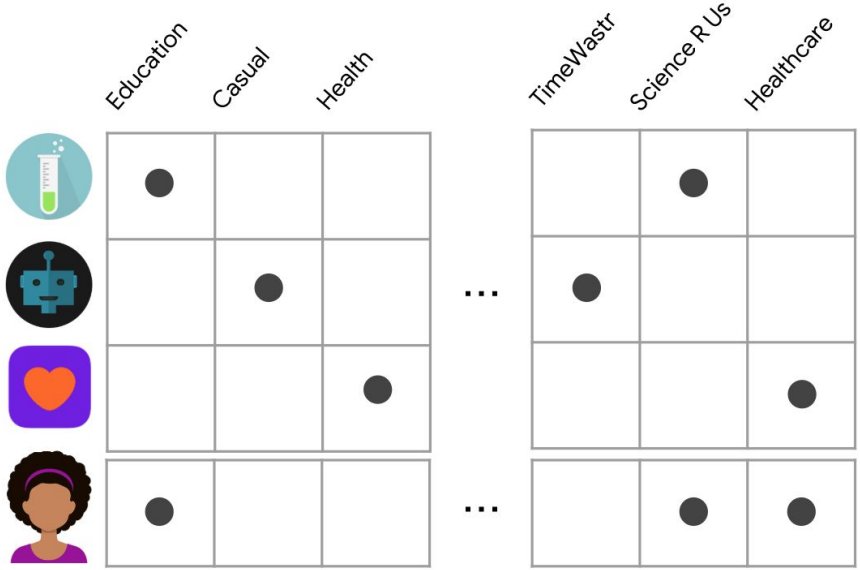
Both content-based and collaborative filtering map each item and each query (or context) to an embedding vector in a common embedding space E=R^d.

A similarity measure is a function $s : E \times E \to \mathbb{R}$ that takes a pair of embeddings and returns a scalar measuring their similarity. The embeddings can be used for candidate generation as follows: given a query embedding $q \in E$, the system looks for item embeddings $x \in E$ that are close to q, that is, embeddings with high similarity $s(q, x)$.

Candidate Generation: Content-based Filtering

- The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
- The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

- Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.
- The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.



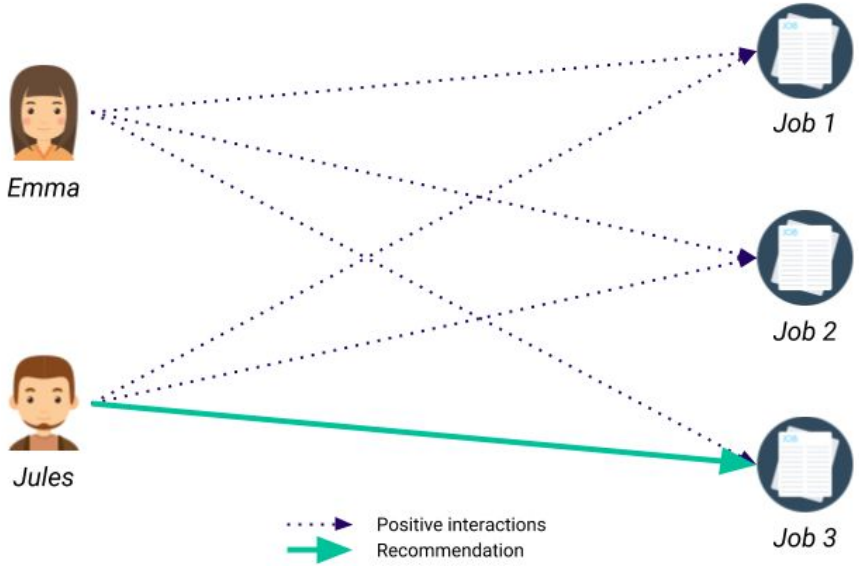
https://developers.google.com/machine-learning/recommendation/content-based/basics

https://developers.google.com/machine-learning/recommendation

Candidate Generation: Collaborative Filtering

- We don't need domain knowledge because the embeddings are automatically learned.
- The model can help users discover new interests.
- To some extent, the system needs only the feedback matrix to train a matrix factorization model.

- Cold start problem
- Hard to include side features for query/item



https://www.welcometothejungle.co/en/articles/collaborative-filtering-job-recommendations

https://developers.google.com/machine-learning/recommendation

Candidate Generation: Matrix Factorization

Observed Only MF

$$\Sigma_{(i,\,j)\,\in\,\text{obs}}\ (A_{ij} - U_i \cdot V_j)^2$$

Weighted MF

$$\Sigma_{(i,\,j)\,\in\,\text{obs}}\ (A_{ij} - U_i \cdot V_j)^2 + w_0 \Sigma_{(i,\,j)\,\notin\,\text{obs}}\ (0 - U_i \cdot V_j)^2$$

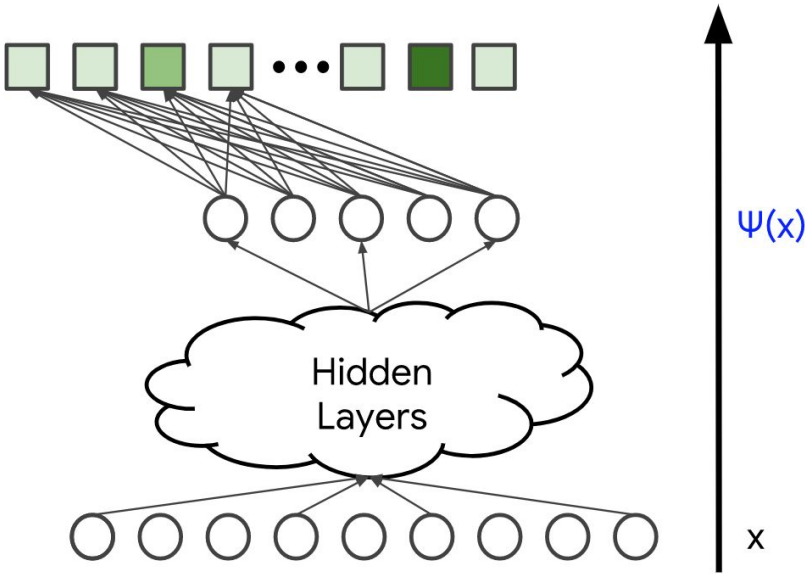
SVD

$$|A - U V^T|_F^2 = \Sigma_{(i,\,j)}\ (A_{ij} - U_i \cdot V_j)^2$$

https://developers.google.com/machine-learning/recommendation

Candidate Generation: Deep Learning

Deep neural network (DNN) models can address these limitations of matrix factorization. DNNs can easily incorporate query features and item features (due to the flexibility of the input layer of the network), which can help capture the specific interests of a user and improve the relevance of recommendations.



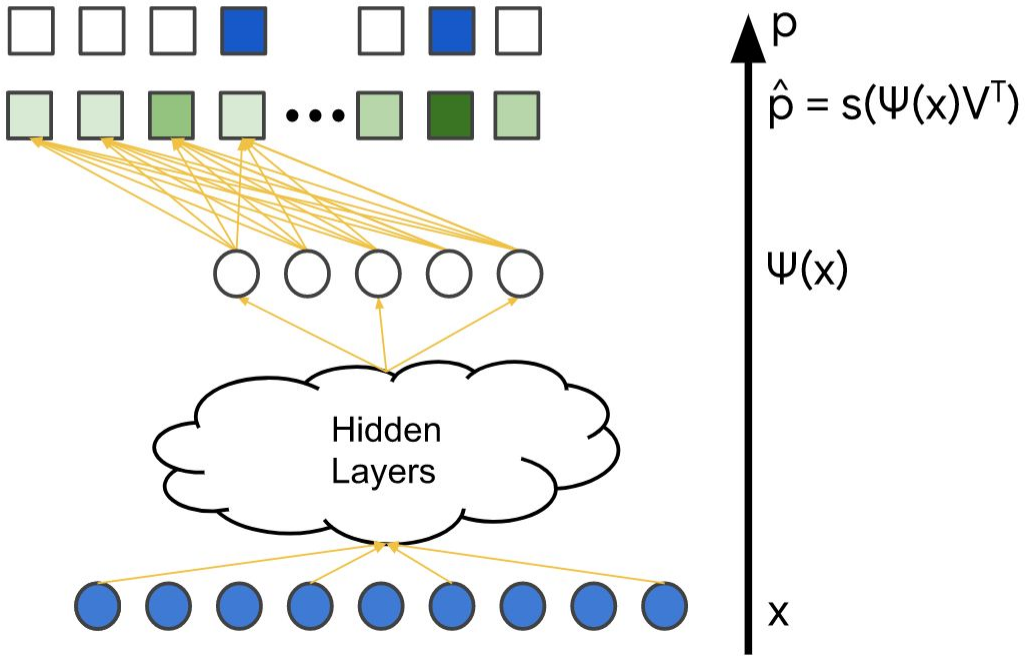
One possible DNN model is softmax, which treats the problem as a multiclass prediction problem in which:

- The input is the user query.
- The output is a probability vector with size equal to the number of items in the corpus, representing the probability to interact with each item; for example, the probability to click on or watch a YouTube video.

https://developers.google.com/machine-learning/recommendation

Candidate Generation: Deep Learning

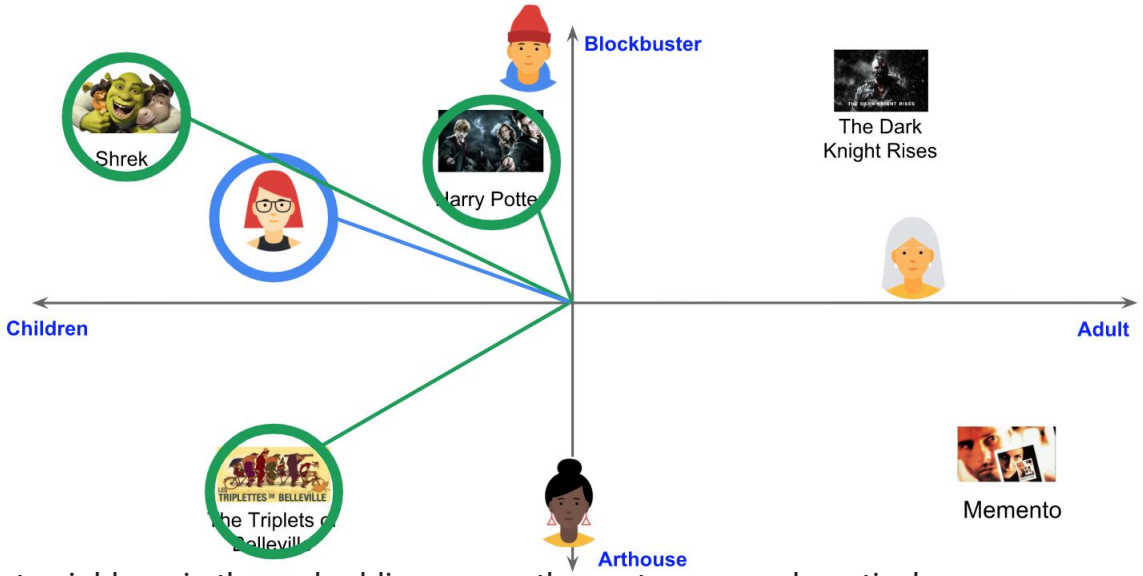
The softmax training data consists of the query features x and a vector of items the user interacted with (represented as a probability distribution p). These are marked in blue in the following figure. The variables of the model are the weights in the different layers. These are marked as orange in the following figure. The model is typically trained using any variant of stochastic gradient descent.



https://developers.google.com/machine-learning/recommendation

Scoring

Once you have the query embedding q, search for item embeddings V_j that are close to q in the embedding space. This is a nearest neighbor problem. For example, you can return the top k items according to the similarity score $s(q, V_j)$.



To compute the nearest neighbors in the embedding space, the system can exhaustively score every potential candidate. Exhaustive scoring can be expensive for very large corpora, but you can use either of the following strategies to make it more efficient:

- If the query embedding is known statically, the system can perform exhaustive scoring offline, precomputing and storing a list of the top candidates for each query. This is a common practice for related-item recommendation.
- Use approximate nearest neighbors.

Re-ranking

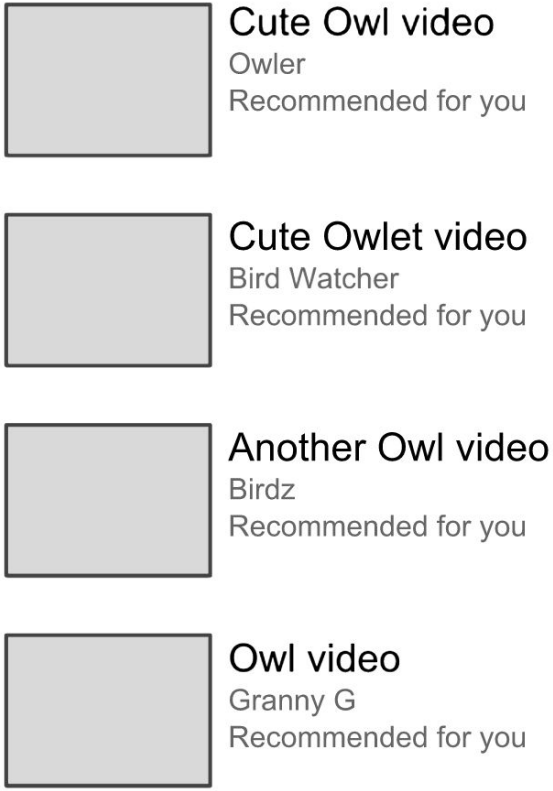
You can implement re-ranking on a video recommender by doing the following:

1. Training a separate model that detects whether a video is click-bait.
2. Running this model on the candidate list.
3. Removing the videos that the model classifies as click-bait.

The system re-ranks videos by modifying the score as a function of:

- video age (perhaps to promote fresher content)
- video length

- Freshness
- Diversity



Re-ranking

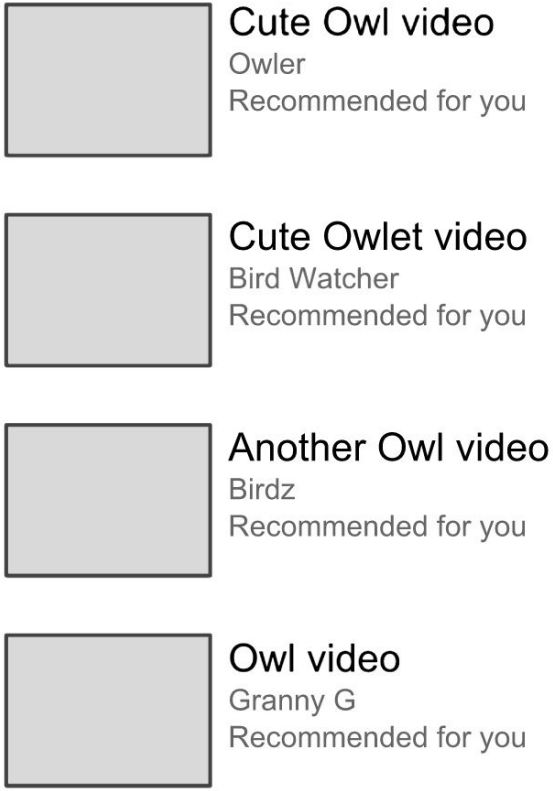
You can implement re-ranking on a video recommender by doing the following:

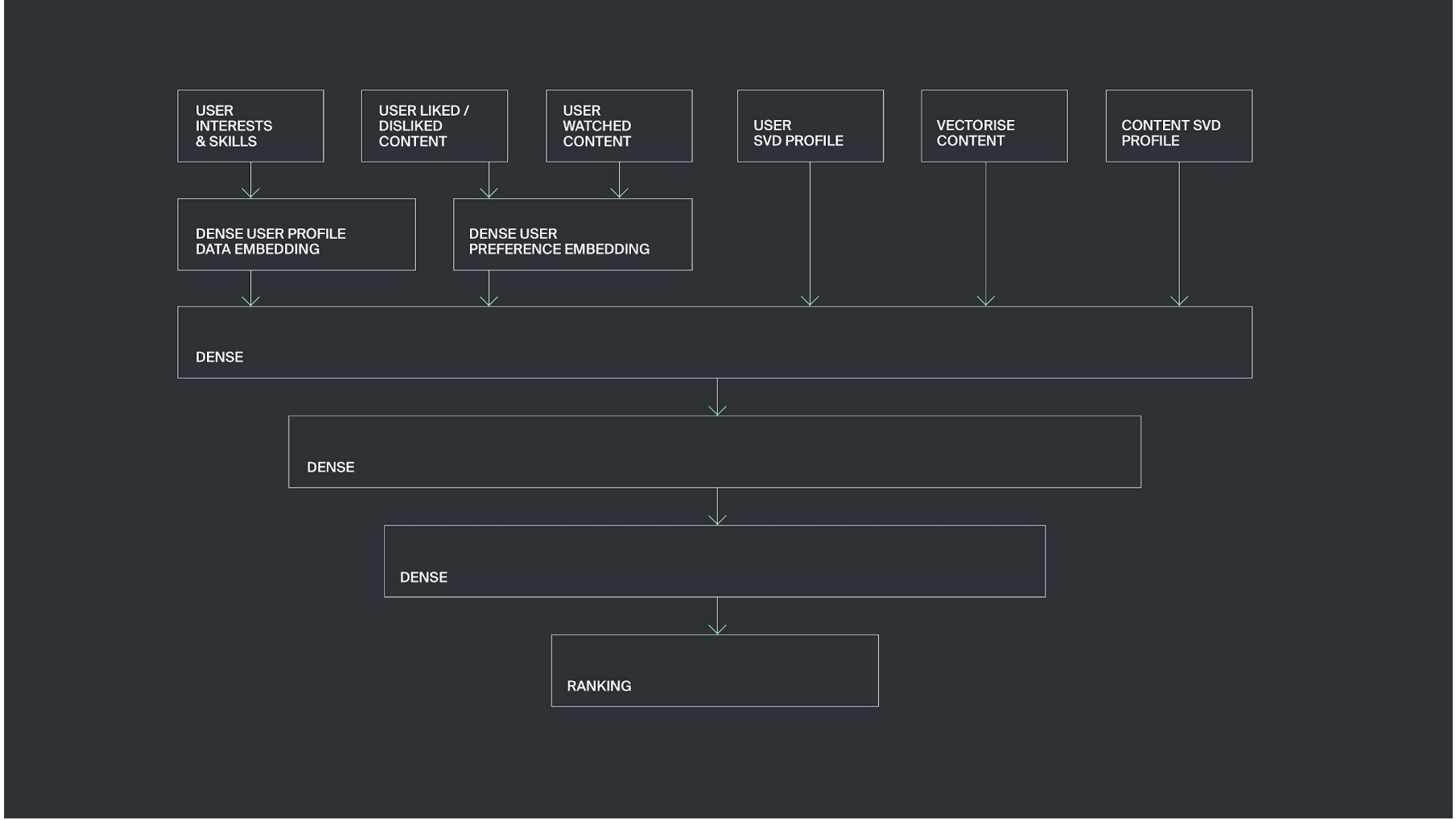
1. Training a separate model that detects whether a video is click-bait.
2. Running this model on the candidate list.
3. Removing the videos that the model classifies as click-bait.

The system re-ranks videos by modifying the score as a function of:

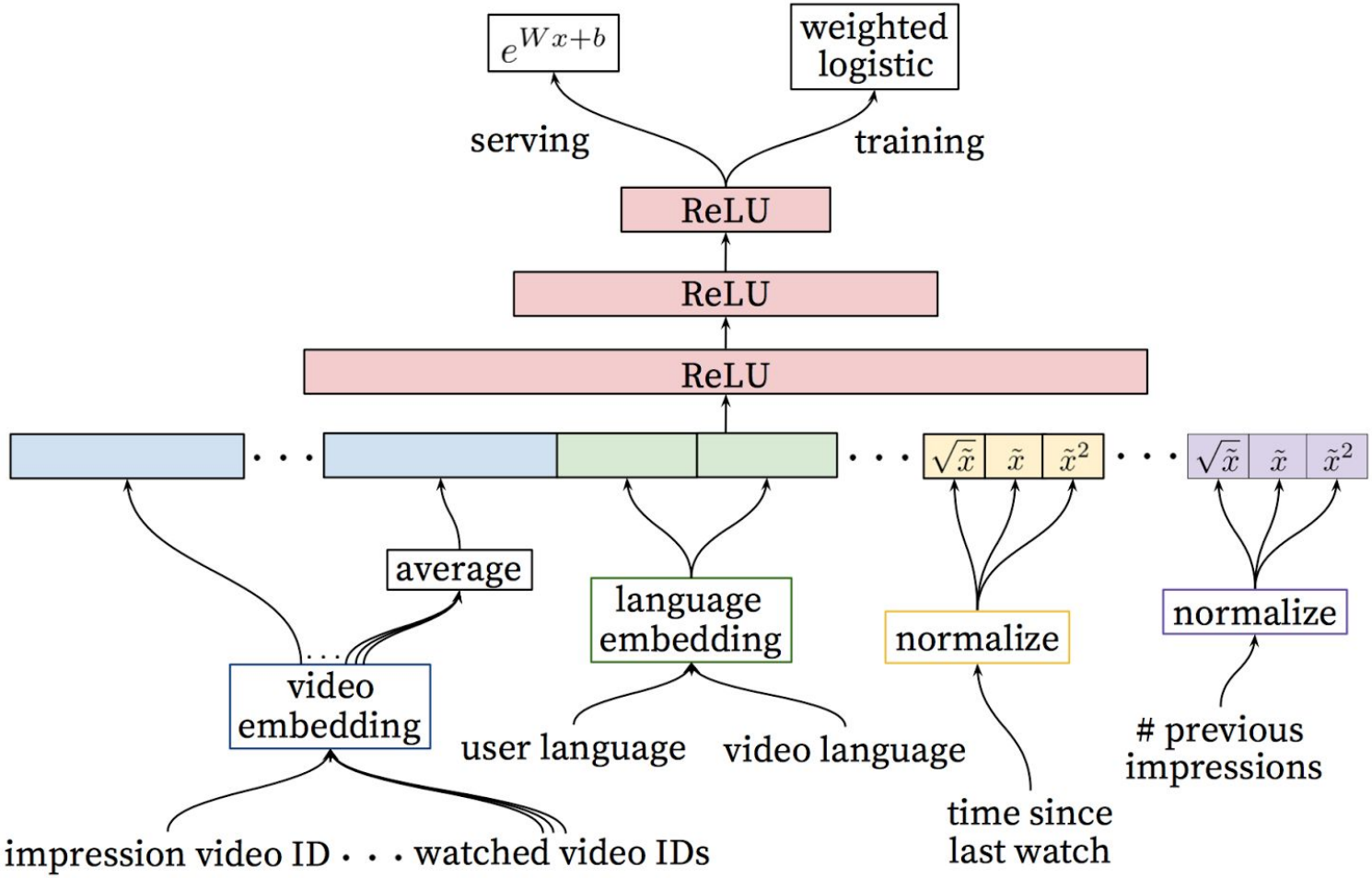
- video age (perhaps to promote fresher content)
- video length

- Freshness
- Diversity





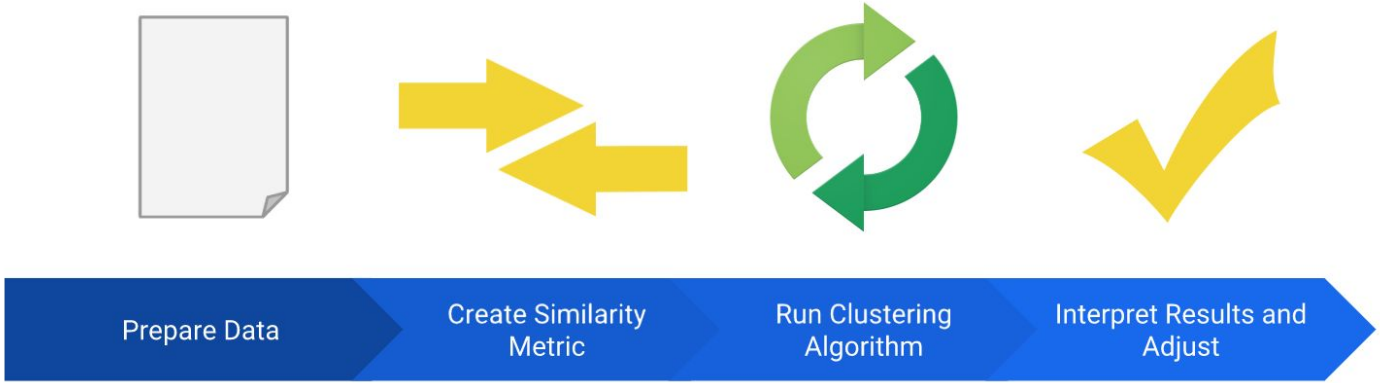
https://towardsdatascience.com/recommender-systems-with-deep-learning-architectures-1adf4eb0f7a6



https://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/45530.pdf

Clustering

1. As with any ML problem, you must normalize, scale, and transform feature data. While clustering however, you must additionally ensure that the prepared data lets you accurately calculate the similarity between examples
2. Before a clustering algorithm can group data, it needs to know how similar pairs of examples are. You quantify the similarity between examples by creating a similarity metric. Creating a similarity metric requires you to carefully understand your data and how to derive similarity from your features
3. A clustering algorithm uses the similarity metric to cluster data



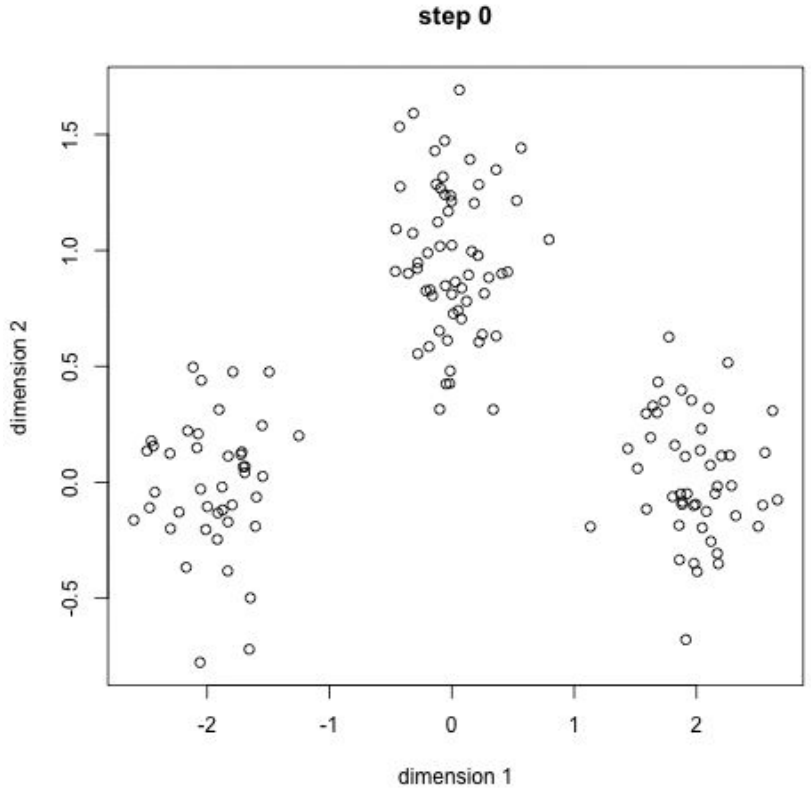
https://developers.google.com/machine-learning/clustering

Clustering: k-means

1) Инициализируем центры случайными объектами
2) Относим каждый объект к ближайшему из центров и присваиваем ему соответствующий класс
3) Двигаем центры в центры масс получившихся кластеров
4) Goto 2) пока сдвиг на 3) >= eps

Одна итерация за O(nk)

http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html



Clustering: EM

- Observations $x_1 \dots x_n$
  - K=2 Gaussians with unknown μ, σ^2

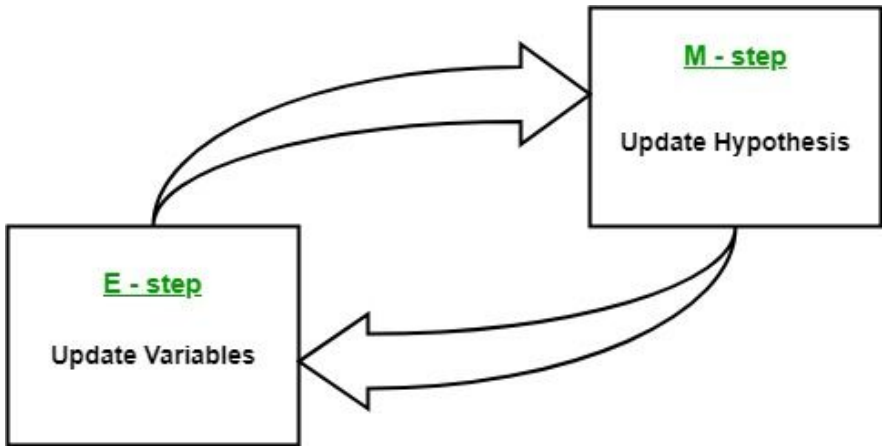
- Chicken and egg problem
  - need (μ_a, σ_a^2) and (μ_b, σ_b^2) to guess source of points
  - need to know source to estimate (μ_a, σ_a^2) and (μ_b, σ_b^2)
- EM algorithm
  - start with two randomly placed Gaussians (μ_a, σ_a^2), (μ_b, σ_b^2)

E-step:
  - for each point: $P(b|x_i)$ = does it look like it came from b?

M-step:
  - adjust (μ_a, σ_a^2) and (μ_b, σ_b^2) to fit points assigned to them
  - iterate until convergence

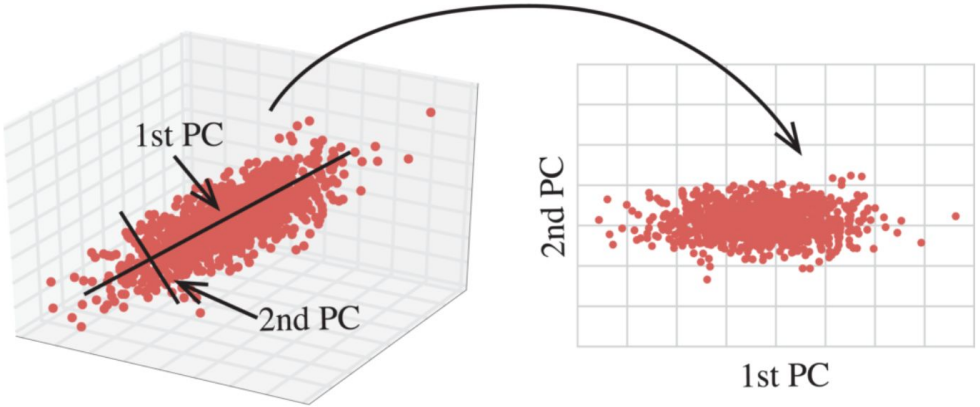
https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Clustering: EM



https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

Dimensionality Reduction: PCA

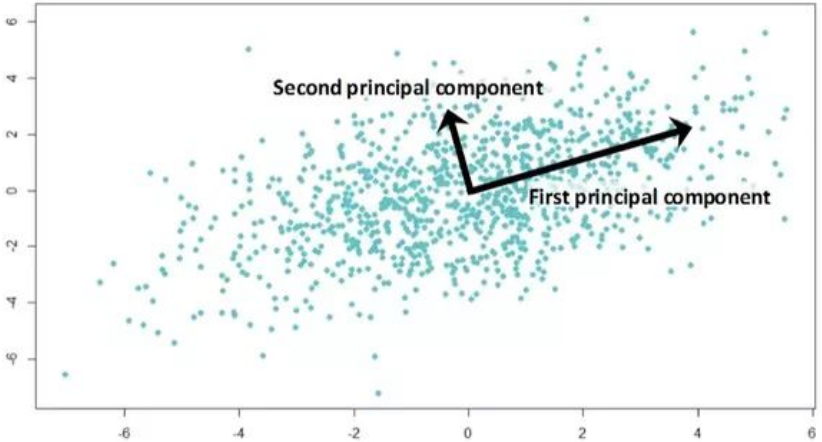


Один из основных способов уменьшить размерность данных, потеряв наименьшее количество информации.

https://habr.com/ru/post/304214/

Dimensionality Reduction: PCA

1) Вычисляем матрицу ковариаций признаков
2) Находим собственные вектора матрицы ковариаций
3) Первые k векторов соответствующих k максимальным собственным значениям - компоненты нашего разложения



https://medium.com/@sadatnazrul/the-dos-and-donts-of-principal-component-analysis-7c2e9dc8cc48

Плюсы: возможность регуляции получаемой размерности (добавлении компонент по одной в зависимости от объяснённой дисперсии); скорость алгоритма; интерпретация

Минусы: линейность, предположение об ортогональности

Dimensionality Reduction: t-SNE

У нас есть набор данных с точками, описываемыми многомерной переменной с размерностью пространства существенно больше трех. Необходимо получить новую переменную, существующую в двумерном или трехмерном пространстве, которая бы в максимальной степени сохраняла структуру и закономерности в исходных данных.

$$p_{j|i} = \frac{exp(- \parallel x_i - x_j \parallel^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(- \parallel x_i - x_k \parallel^2 / 2\sigma_i^2)}$$

Эта формула показывает, насколько точка Xj близка к точке Xi при гауссовом распределении вокруг Xi с заданным отклонением σ. Сигма будет различной для каждой точки. Она выбирается так, чтобы точки в областях с большей плотностью имели меньш

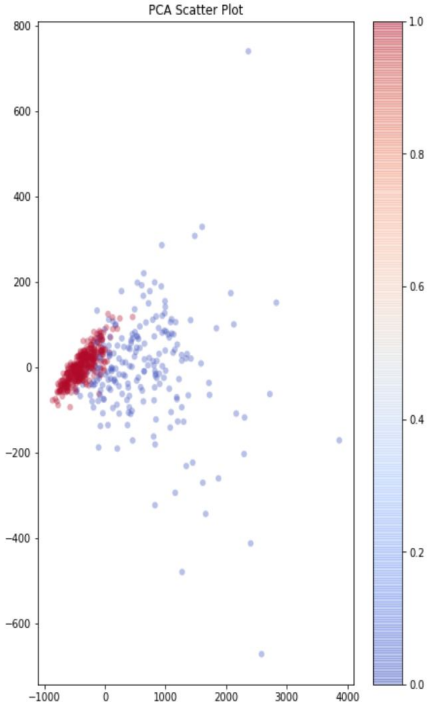
Для двумерных или трехмерных «коллег» пары Xi и Xj, назовем их для ясности Yi и Yj, не представляет труда оценить условную вероятность, используя ту же формулу 1. Стандартное отклонение предлагается установить в 1/√2:

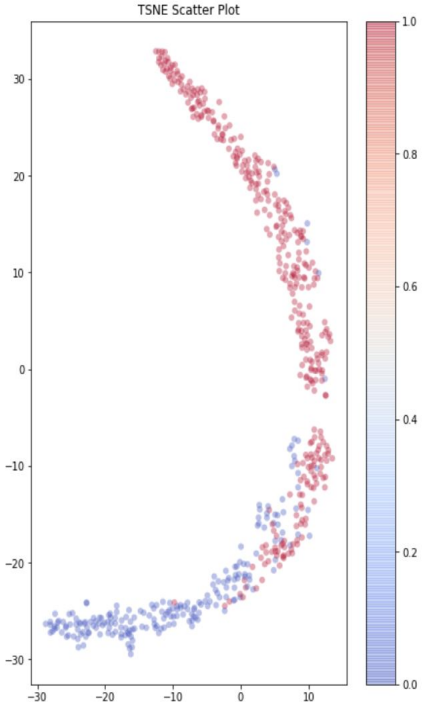
$$q_{j|i} = \frac{exp(- \parallel y_i - y_j \parallel^2)}{\sum_{k \neq i} exp(- \parallel y_i - y_k \parallel^2)}$$

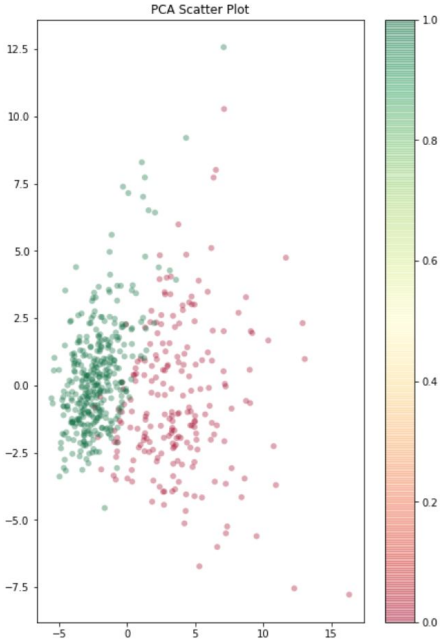
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n},$$

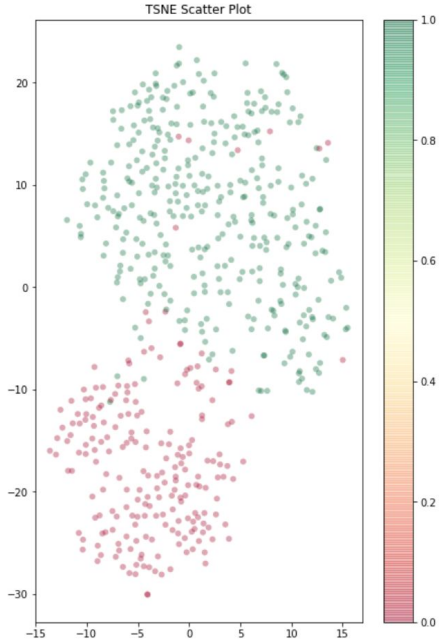
$$Cost = \sum_i KL(P_i \parallel Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}} \quad Cost = KL(P \parallel Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}},$$

Dimensionality Reduction: PCA vs t-SNE

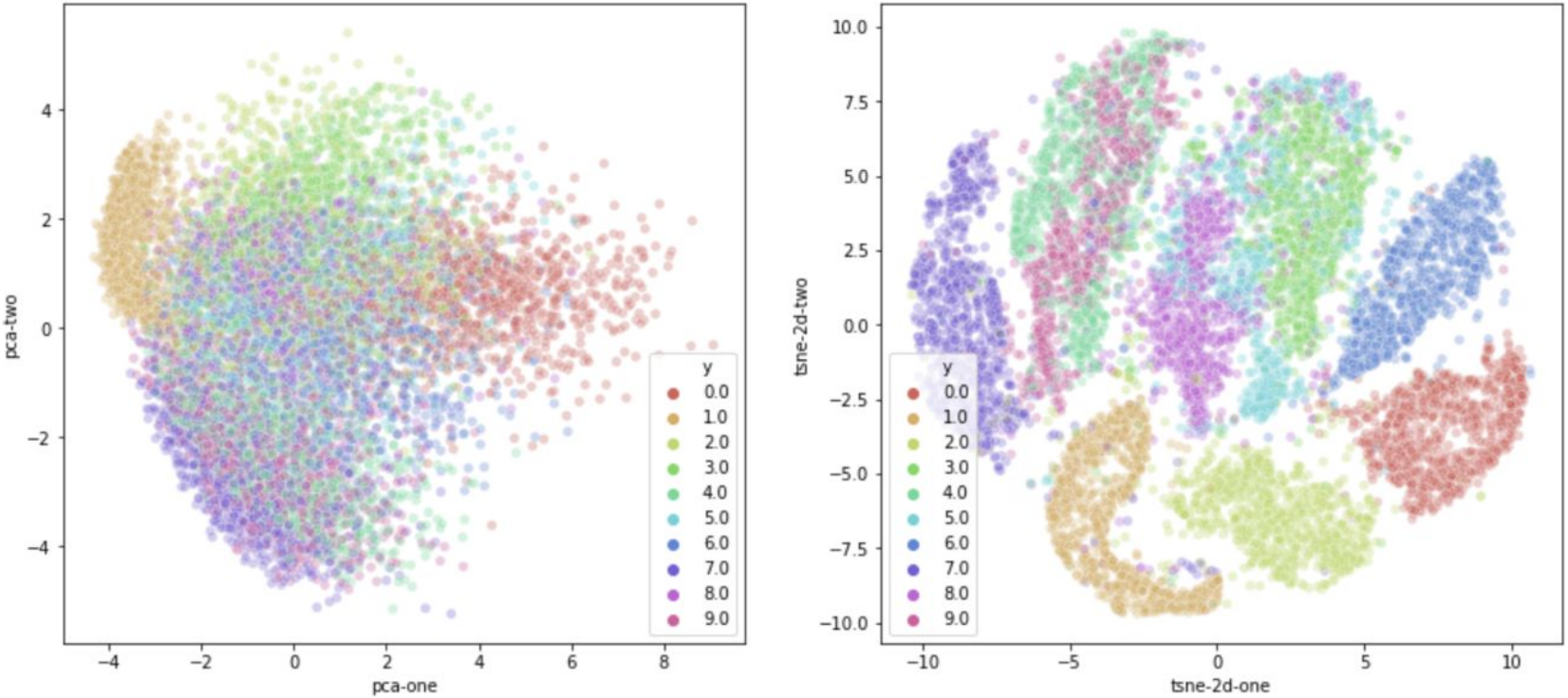








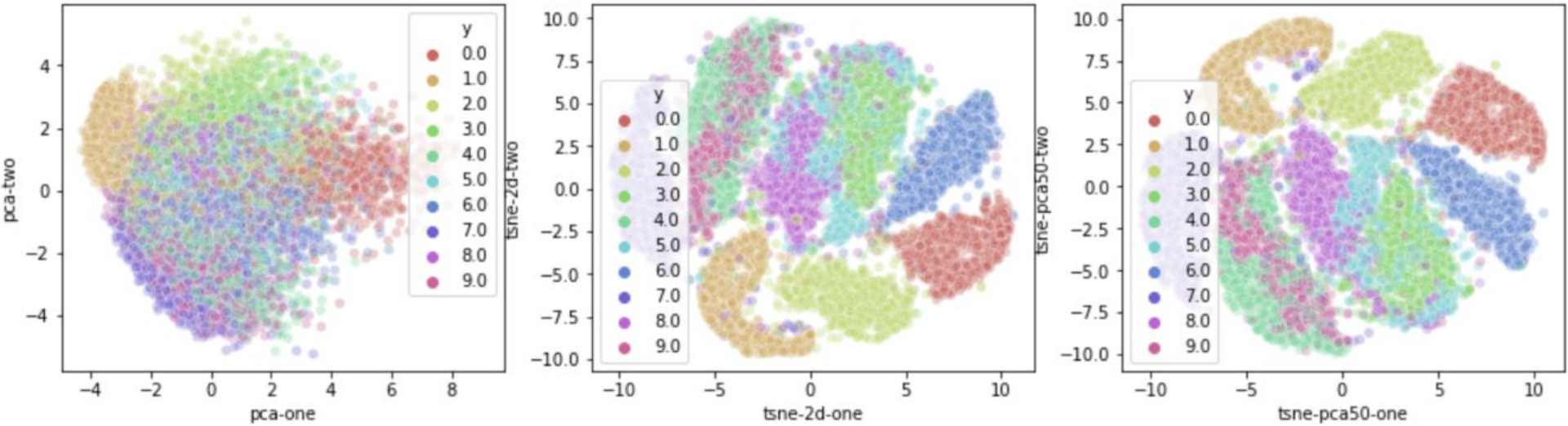
Dimensionality Reduction: PCA vs t-SNE



PCA (left) vs T-SNE (right)

https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

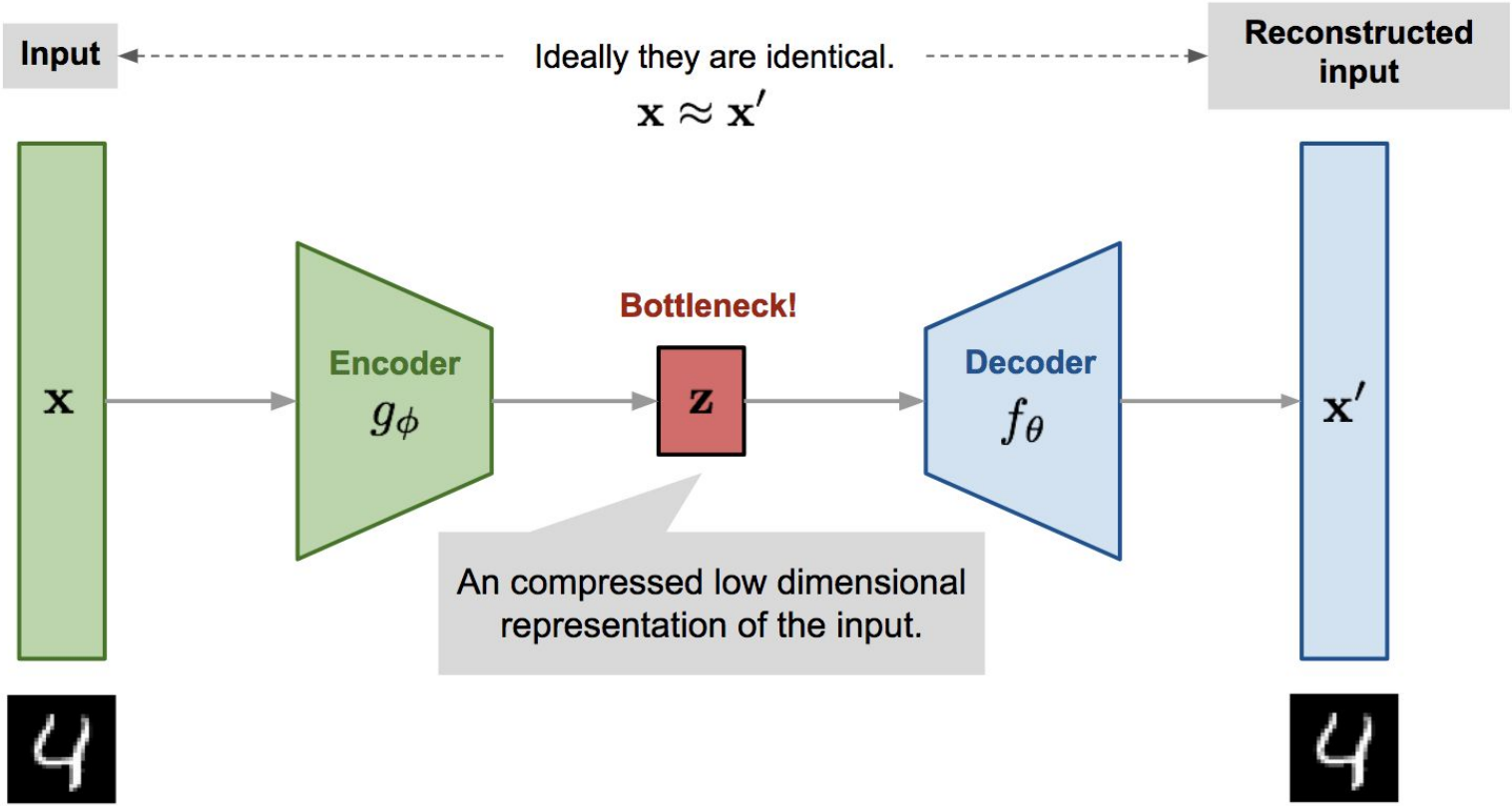
Dimensionality Reduction: PCA vs t-SNE



PCA (left) vs T-SNE (middle) vs T-SNE on PCA50 (right)

https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b

Dimensionality Reduction: Autoencoder



Summary

Links

- https://www.toptal.com/machine-learning/clustering-algorithms
- https://web.stanford.edu/class/cs345a/slides/12-clustering.pdf
- https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68
- https://medium.com/dailymotion/building-modern-recommender-systems-when-deep-learning-meets-product-principles-c79b16375109
- https://towardsdatascience.com/recommender-systems-with-deep-learning-architectures-1adf4eb0f7a6
- https://blog.statsbot.co/time-series-anomaly-detection-algorithms-1cef5519aef2
- https://arxiv.org/pdf/1707.07435.pdf
- https://blog.statsbot.co/recommendation-system-algorithms-ba67f39ac9a3
- https://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/45530.pdf
- https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-datSc4-EMmixt.pdf