Конспект лекции

А.Н. Шевляков

Введение в нейронные сети

Метрические задачи машинного обучения

# Постановка задачи

Есть датасет MegaFace: **4.7** миллиона фото; **672,057** уникальных людей;

в среднем **7** фото у одного человека (3 минимум, 2469 максимум).

**Нужно решить такие задачи:**

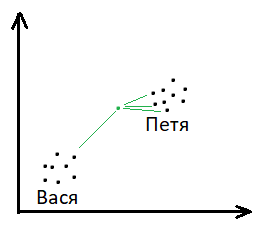
* дана одна новая фотка. Кто из имеющихся в датасете людей изображён на ней? Или такого человека в датасете нет?
* даны две фотки. Эти фотки принадлежат одному человеку или разным (НС должна дать ответ, даже если этого человека нет в датасете)?

Это **не задачи классификации**! Тут очень много классов и мало представителей одного класса.

При решении подобных задач важно уметь превращать фото в вектора. То есть по фото X нужно получить числовой вектор е(Х). Причём фото одного и того же человека должны отображаться в близкие вектора.

Допустим, что подобный **embedding** уже реализован. Как с его помощью решать указанные выше задачи?

**Кто из людей изображен на фото?** Превращаем всех людей в вектора.  
Каждому человеку соответствует облако точек (векторов). Новую фотку тоже превращаем в вектор (зелёная точка). А дальше используем классические алгоритмы ML.



Например, можно найти **k ближайших соседей** зелёной точки (метод kNN). Человек, **чьи фото преобладают среди соседей** зелёной точки, будет считаться человеком с нового фото.

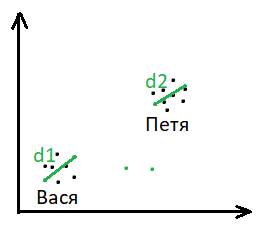
**На двух фото один человек?** Превращаем всех людей в вектора.  
Каждому человеку соответствует облако точек (векторов).

У каждого облака есть диаметр (расстояние между

самыми далёкими точками).

Пусть D — самый большой диаметр среди диаметров всех облаков.

Две новых фотки тоже превращаем в вектора (две зелёных точки).



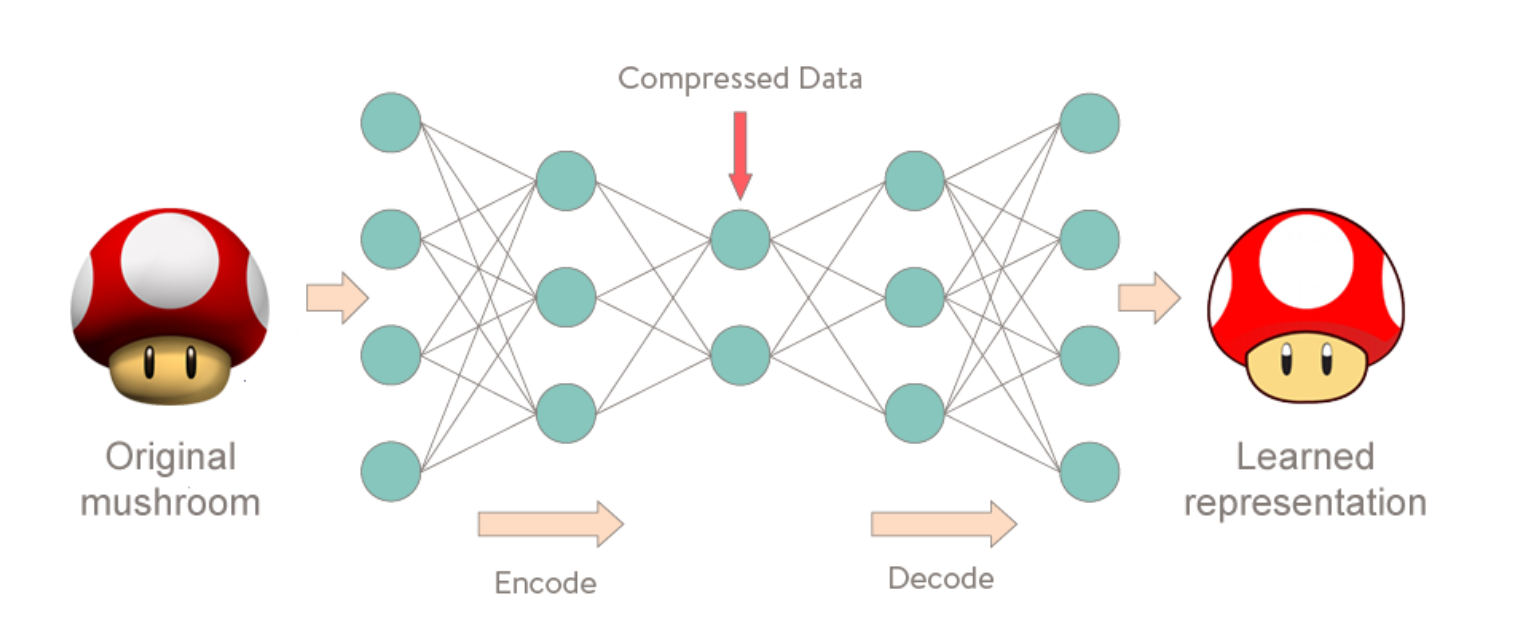
Правило: если расстояние между зелёными точками меньше D, то считается, что эти фото принадлежат одному человеку.

В противном случае — это фото разных людей.

# превращение фото в вектора

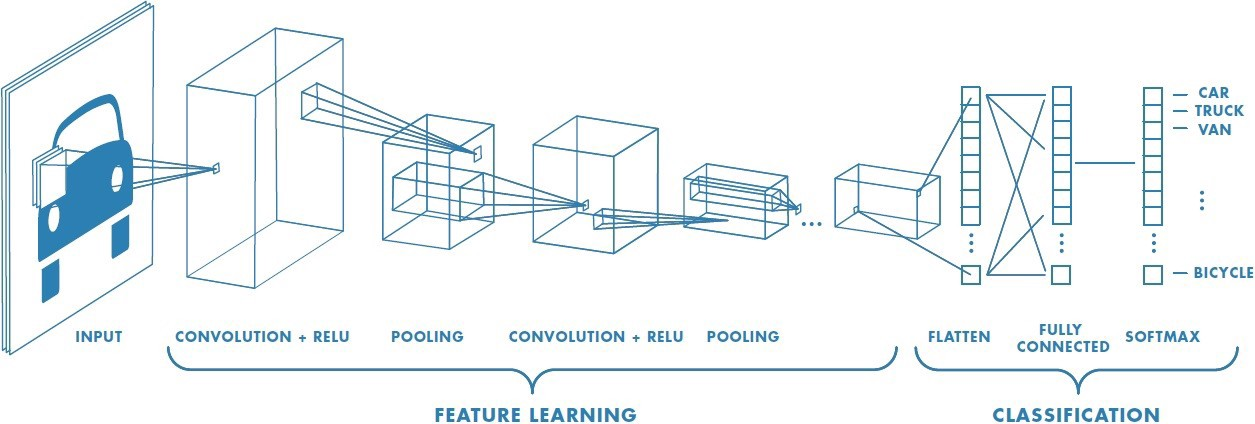
Первый способ: **автокодировщики.** Мы знаем, что каждый АК с помощью бутылочного горлышка превращает объект Х в числовой вектор e(X).

Но есть недостаток: размерность вектора e(X) мала (по сравнению с входом Х).

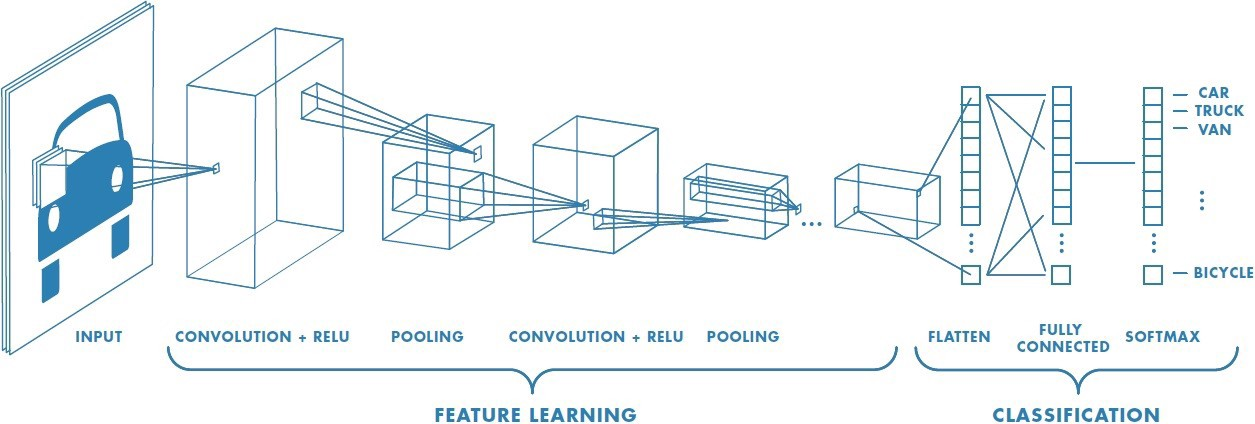


Второй способ: **обрезанные СНС.** Вот стандартная СНС.

Сначала у неё идут свёрточные слои, потом — полносвязные.



Обрежем ей конец. Теперь СНС заканчивается полносвязным слоем из n нейронов. Выходы этих нейронов — это и будут координаты вектора e(X).



**Преимущества**: размерность вектора e(X) может быть достаточно большой, так как можно создать полносвязный слой с произвольным числом нейронов.

# Тренировка embedding-а

Как тренировать обрезанную СНС (ОСНС)?

СНС же тренируется по ТВ, то есть должны быть известны ответы, в какие числовые вектора переводить изображения. А это неизвестно.

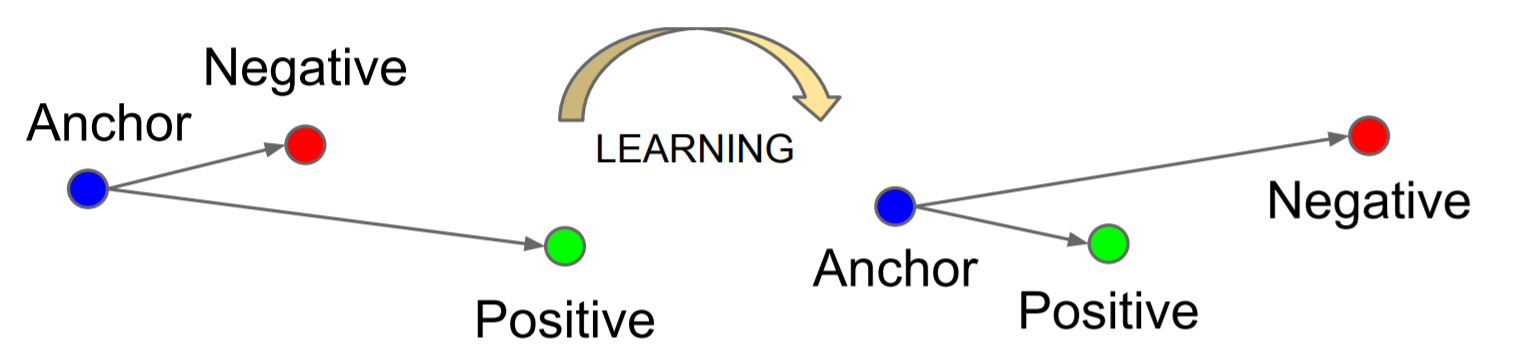
Функция потерь у ОСНС должна стимулировать отображать фото одного человека X в близкие вектора e(X), а фотки разных людей должны отображаться в далёкие вектора.

**А вот как это завернуть в виде формулы?**

На самом деле ОСНС обучается не на единичных изображениях, а на **тройках**. Функция потерь в этом случае называется **triplet loss**.

Каждая тройка состоит из изображений (A,P,N), где

* изображение А называется **якорем** тройки;
* на изображениях P и А изображен один и тот же человек;
* на изображениях N и А изображены разные люди.



**Задача**: вектора e(A), e(P) должны быть близки друг к другу, а вектора e(A), e(N) далеки друг от друга. В общем, величина d(e(A),e(P)) — d(e(A),e(N)) должна быть как можно меньше.

Но не всё так просто. Если сразу минимизировать эту величину

d(e(A),e(P)) - d(e(A),e(N)),то ОСНС будет отображать **все фото одного человека в один вектор** (тогда занулится выражение d(e(A),e(P))).

**Но это плохо**: для решения наших задач важно, чтобы фотки одного человека формировали облако, а не отображались в один вектор.

Следовательно, **нужен компромисс** между желанием отобразить все фото одного человека в один вектор и формированием облака (то есть облако должно быть нетривиальным и достаточно компактным).

Поэтому в качестве **triplet loss** рассматривают величину

L(A,P,N)=max(d(e(A),e(P))-d(e(A),e(N))+𝛂, 0) и уже у него ищут минимум.

Минимизация triplet loss-a стремится к достижению равенства

L(A,P,N)=max(d(e(A),e(P))-d(e(A),e(N))+𝛂, 0)=0, то есть стремится к достижению равенства d(e(A),e(P))-d(e(A),e(N))+𝛂<0.

Или: d(e(A),e(N))>d(e(A),e(P))+𝛂 и поэтому d(e(A),e(N))>𝛂

То есть 𝛂 - это **минимальный порог, который отделяет друг от друга объекты разных классов.**