онлайн-курс

СПЕЦИАЛЬНЫЕ АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ

Метрические задачи

Metric learning

Постановка задачи

Описание данных

Есть такой 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.

Основной инструмент: embedding

При решении подобных задач важно уметь превращать фото в вектора. То есть по фото X нужно получить числовой вектор e(X).

Причём фото одного и того же человека должны отображаться в близкие вектора.

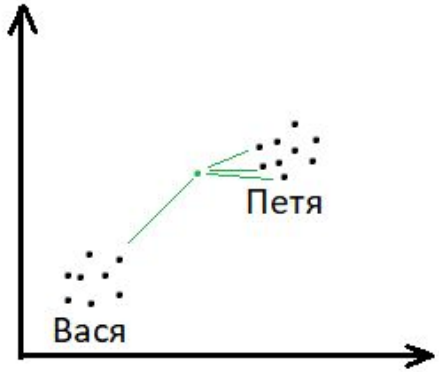
Допустим, что подобный embedding уже реализован.
Как с его помощью решать указанные выше задачи?

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).

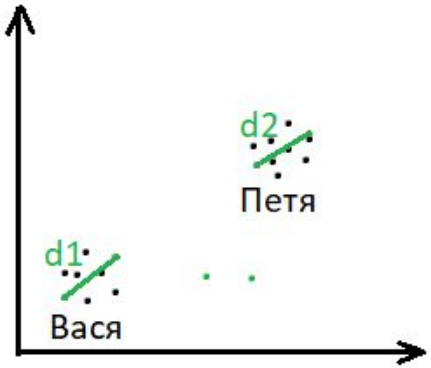
А дальше используем классические алгоритмы ML.



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.

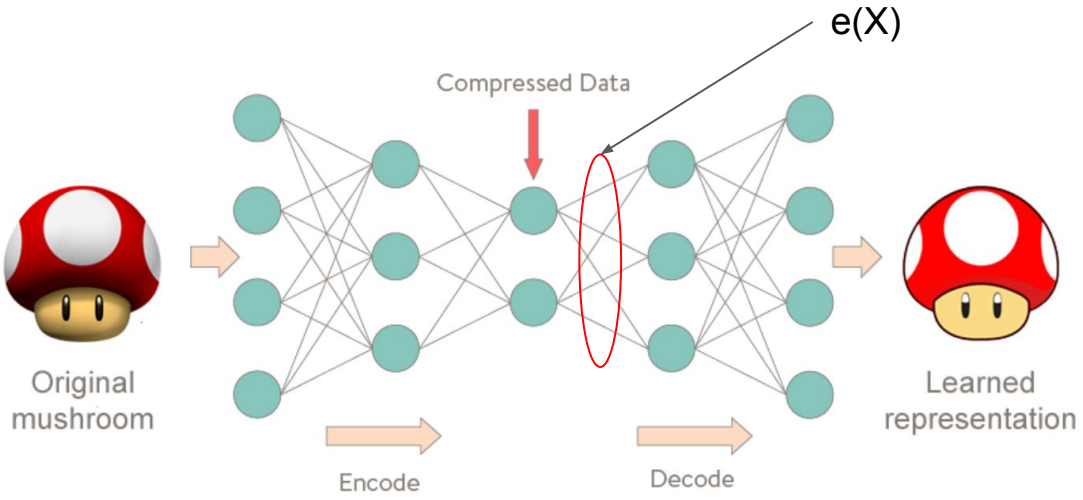
Как превращать фото в вектора?

Первый способ: автокодировщики

Мы знаем, что каждый АК с помощью бутылочного горлышка превращает объект X в числовой вектор e(X).

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

Но есть и другие методы...

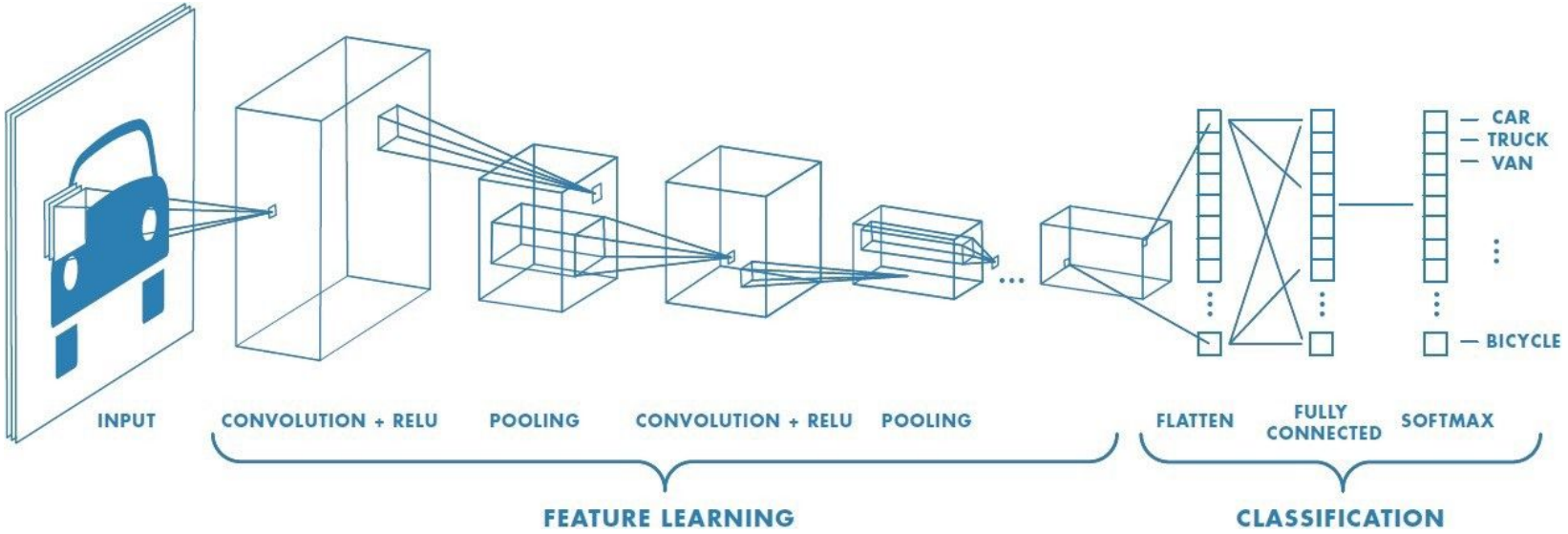


Второй способ: обрезанные CNC

Вот стандартная CNC.

Сначала у неё идут свёрточные слои, потом — полносвязные.

Сделаем ей чик-чик.

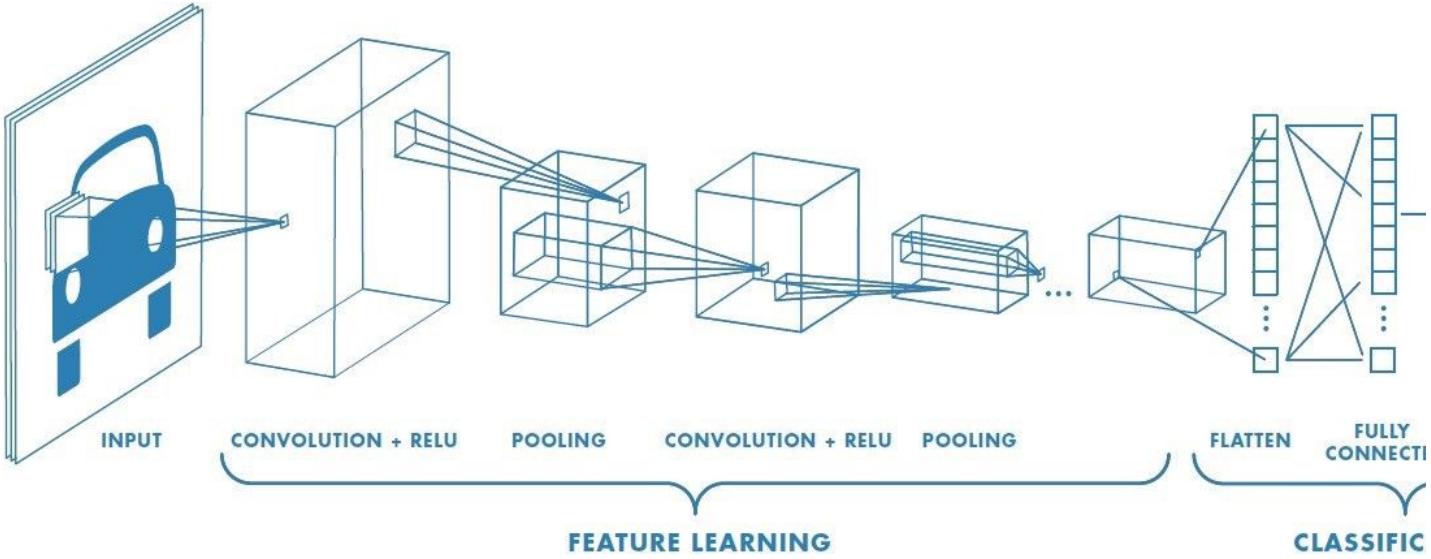


Второй способ: обрезанные CHC

Обрежем ей конец (простите за каламбур).

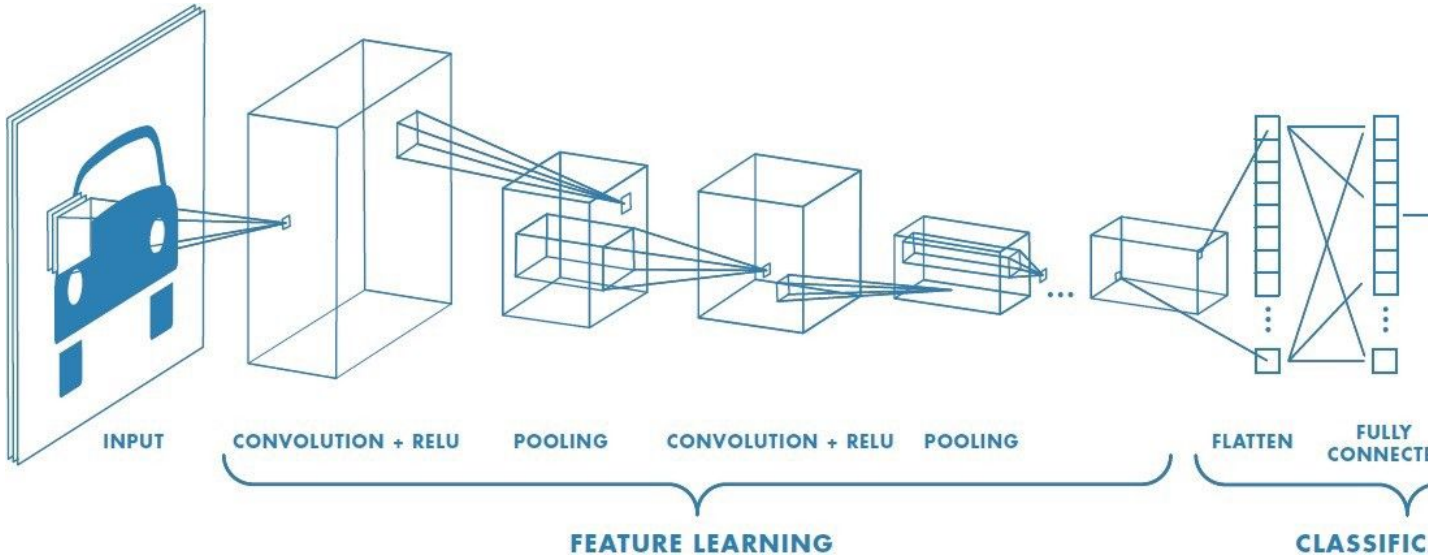
Теперь CHC заканчивается полносвязным слоем из n нейронов.

Выходы этих нейронов — это и будут координаты вектора e(X).



Второй способ: кастрированные CNC

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



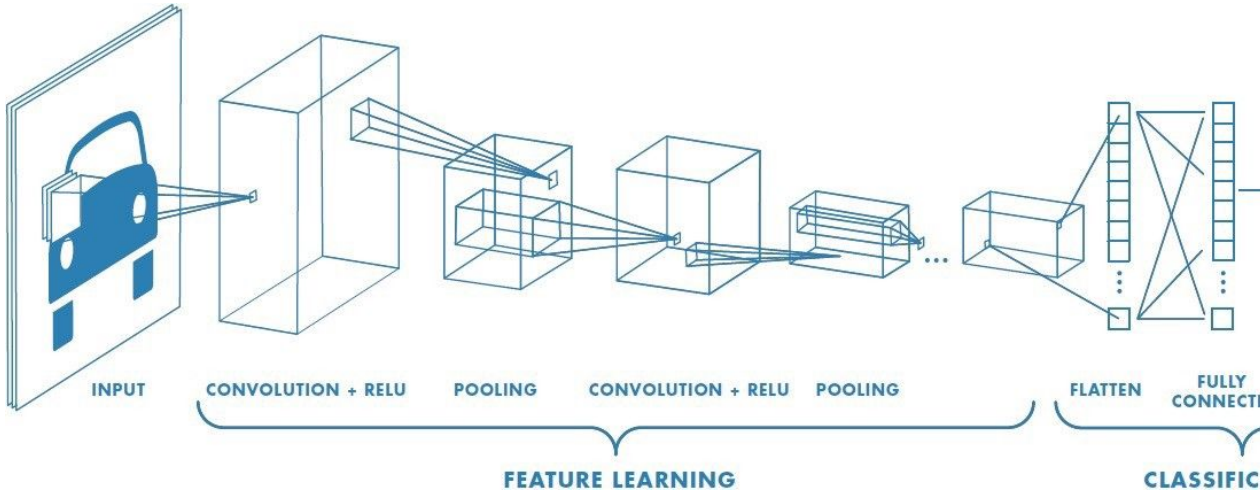
Тренировка embedding-а

Тренировка обрезанной CHC

А как тренировать обрезанную CHC (OCHC)?

CHC же тренируется по ТВ, то есть должны быть известны ответы, в какие числовые вектора переводить изображения.

А это неизвестно.



Какая тут функция потерь?

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**.

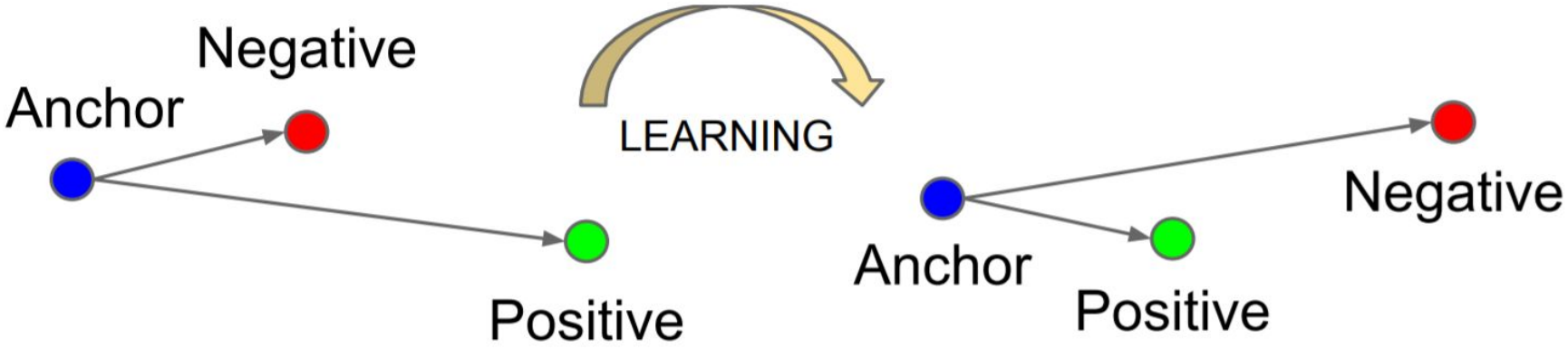
Каждая тройка состоит из изображений (A,P,N), где

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

Какая тут функция потерь?

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

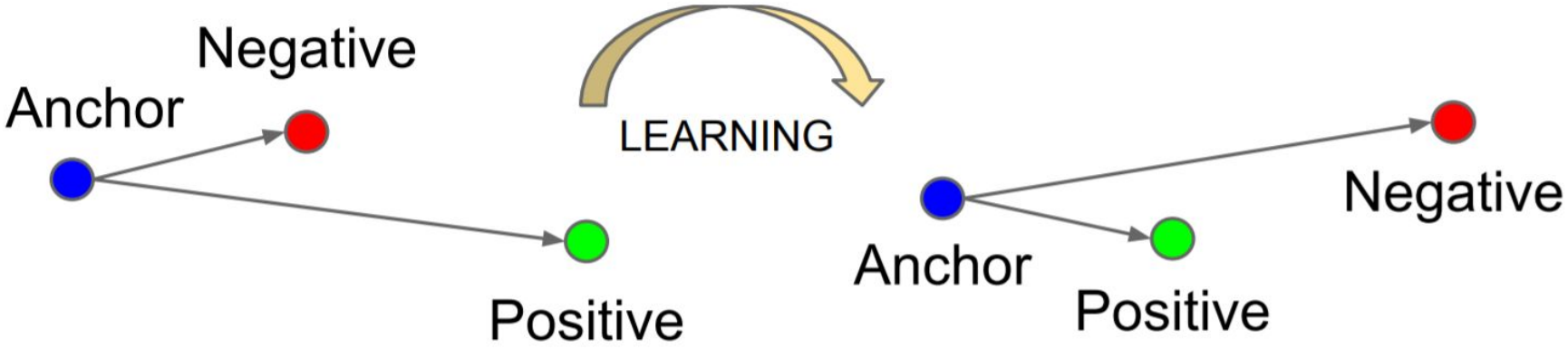
- изображение A называется **якорем** тройки;
- на изображениях P и A изображен один и тот же человек;
- на изображениях N и A изображены разные люди.



Какая тут функция потерь?

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.



Какая тут функция потерь?

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.

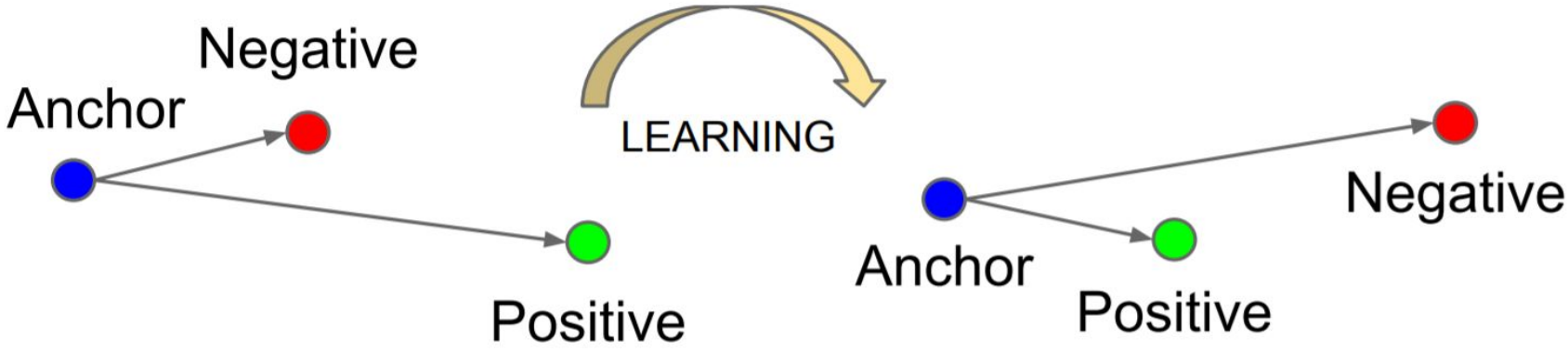
Следовательно, **нужен компромисс** между желанием отобразить все фото одного человека в один вектор и формированием облака (то есть облако должно быть нетривиальным и достаточно компактным).

Какая тут функция потерь?

Поэтому в качестве **triplet loss** рассматривают величину

$L(A,P,N)=\max(d(e(A),e(P))-d(e(A),e(N))+\alpha, 0)$

и уже у него ищут минимум.



Смысл параметра α

Минимизация triplet loss-а стремится к достижению равенства
$L(A,P,N)=\max(d(e(A),e(P))-d(e(A),e(N))+\alpha, 0)=0,$

то есть стремится к достижению равенства
$d(e(A),e(P))-d(e(A),e(N))+\alpha<0.$

Или:
$d(e(A),e(N))>d(e(A),e(P))+\alpha$

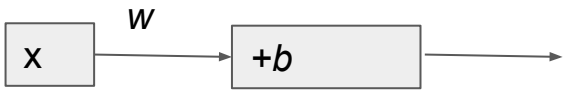
и поэтому $d(e(A),e(N))>\alpha$

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

Пример

Дана ТВ

И НС, которая осуществляет embedding:



Объекты	X	Y
A1	-2	0
A2	-1	0
B1	1	1
B2	2	1

Нужно перебрать все возможные тройки (A,P,N) и для каждой из них выписать triplet loss, потом все triplet loss-ы суммировать — это и будет итоговая функция потерь, которую и надо минимизировать.

Итак, для входа X НС вычисляет **embedding e(x)=wx+b.**
Перебираем все допустимые тройки: (A1,A2,B1), (A1,A2,B2), (A2,A1,B1), (A2,A1,B2), (B1,B2,A1), (B1,B2,A2), (B2,B1,A1), (B2,B1,A2).

Тройка	**d(e(A),e(P))**	**d(e(A),e(N))**	**triplet loss**
(A1,A2,B1)	d(-2w+b,-w+b)=w	d(-2w+b,w+b)=3w	max(w-3w+5,0)
(A1,A2,B2)	d(-2w+b,-w+b)=w	d(-2w+b,2w+b)=4w	max(w-4w+5,0)
(A2,A1,B1)	d(-w+b,-2w+b)=w	d(-w+b,w+b)=2w	max(w-2w+5,0)
(A2,A1,B2)	d(-w+b,-2w+b)=w	d(-w+b,2w+b)=3w	max(w-3w+5,0)
(B1,B2,A1)	d(w+b,2w+b)=w	d(w+b,-2w+b)=3w	max(w-3w+5,0)
(B1,B2,A2)	d(w+b,2w+b)=w	d(w+b,-w+b)=2w	max(w-2w+5,0)
(B2,B1,A1)	d(2w+b,w+b)=w	d(2w+b,-2w+b)=4w	max(w-4w+5,0)
(B2,B1,A2)	d(2w+b,w+b)=w	d(2w+b,-w+b)=3w	max(w-3w+5,0)

e(x)=wx+b. Метрика евклидова. Пусть α =5

Итоговая функция потерь для ТВ:

$$L(w,b)=\max(w-3w+5,0)+\max(w-4w+5,0)+\max(w-2w+5,0)+\max(w-3w+5,0)+$$
$$+\max(w-3w+5,0)+\max(w-2w+5,0)+\max(w-4w+5,0)+\max(w-3w+5,0)$$

$$L(w,b)=\max(-2w+5,0)+\max(-3w+5,0)+\max(-w+5,0)+\max(-2w+5,0)+$$
$$+\max(-2w+5,0)+\max(-w+5,0)+\max(-3w+5,0)+\max(-2w+5,0)$$

Её и нужно минимизировать относительно весов w,b с помощью ГС.

Вот так и тренируется сеть для задачи embedding!

Выводы

- Мы рассмотрели типичные задачи metric learning.
- Обсудили, как можно их решать, если известно представление объектов (embedding) в виде векторов.
- Рассмотрели архитектуру сети и принципы её тренировки для решения задач metric learning.