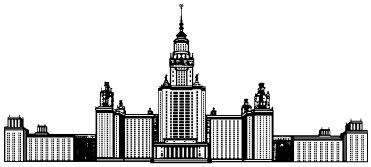
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики
Кафедра Математических Методов Прогнозирования

КУРСОВАЯ РАБОТА

«Постановка и реализация различных задач многомерного шкалирования на основе модели глубинного обучения PyTorch»
«Formulation and implementation of various multidimensional scaling problems based on PyTorch deep learning model»

Выполнил:
студент 3 курса 317 группы
Сенин Александр Николаевич

Научный руководитель:
к.ф.-м.н.
Майсурадзе Арчил Ивериевич

Москва, 2020

Содержание

Аннотация

В данной работе уделяется значительное внимание понятию метрики, ее аксиомам. Рассмотрены обобщения метрики, приведены их аксиомы, даны соответствующие примеры в анализе данных. Сформулирована общая постановка задачи многомерного шкалирования, рассмотрены основные подходы к ее решению: классическое многомерное шкалирование, метрическое многомерное шкалирование, неметрическое многомерное шкалирование. В работе представлена модель машинного обучения, использующая полносвязную нейронную сеть, решающая поставленную задачу, описана логика ее обучения на случай метрического и неметрического многомерного шкалирования. Описана конкретная реализация алгоритма, использующего представленную модель, даны советы по реализации, обращено внимание на потенциальные ошибки. Приведены примеры работы реализованного алгоритма.

Аксиомы метрики

Задачи, рассмотренные далее в этой работе, существенно используют понятие метрики и его обобщения. Предварительно вспомним строгое определение метрики, перечислим возможные его обобщения.

Для построения системы аксиом метрики достаточно формализировать естественные представления о расстоянии: неотрицательность, равенство нулю на совпадающих объектах (и только на них), симметричность и справедливость неравенства треугольника. Соответственно, пусть на множестве X определена функция $\rho : X \times X \to \mathbb{R}$, такая что для любых $x, y, z \in X$:

- $\rho(x, y) \geqslant 0$, причем $\rho(x, y) = 0 \Leftrightarrow x = y$
- $\rho(x, y) = \rho(y, x)$
- $\rho(x, y) \leqslant \rho(x, z) + \rho(z, y)$

Тогда функция ρ называется метрикой.

Классический пример метрики — расстояние Минковского:

$$l_p(x, y) = \left(\sum_{i=1}^{d} |x^i - y^i|^p \right)^{1/p}$$

Частные случаи l_p:

- Евклидово расстояние при $p = 2$.
- Расстояние городских кварталов при $p = 1$.
- Метрика Чебышева при $p = \infty : l_\infty(x, y) = \max\limits_{i=1\ldots d} |x_i - y_i|$.

l_p не является метрикой при $p < 1$ (нарушается неравенство треугольника).

Понятие метрики в машинном обучении и анализе данных используется в метрических методы для решения задач классификации или регрессии (например, kNN), задаче кластеризации и обнаружения выбросов, задаче понижения размерности и многомерного шкалирования.

Интересно, что представленная система аксиом не будет являться независимой. Чтобы показать это, перепишем ее в развернутом виде $\forall x, y, z \in X$:

1. $\rho(x, y) \geqslant 0$
2. $\rho(x, y) = 0 \Rightarrow x = y$
3. $x = y \Rightarrow \rho(x, y) = 0$
4. $\rho(x, y) = \rho(y, x)$
5. $\rho(x, y) \leqslant \rho(x, z) + \rho(z, y)$

Сперва покажем, что аксиома неотрицательности следует из 3, 4, 5 аксиом:

$$0 = \rho(x, x) \leqslant \rho(x, y) + \rho(y, x) = 2\rho(x, y) \Rightarrow \rho(x, y) \geqslant 0$$

Независимой будет система из аксиом лишь со второй по пятую. Докажем это. Будем рассуждать так: предположим, что в любом метрическом пространстве X аксиома с номером i вытекает из аксиом $\{2, 3, 4, 5\}\backslash\{i\}$. Чтобы показать, что это не так, для каждого $i = 2, 3, 4, 5$ приведем соответствующий контрпример.

Для доказательства независимости второй аксиомы положим $\rho(x, y) \equiv 0$. Тогда автоматически будут выполнены требования 3, 4, 5, но из равенства нулю метрики не будет следовать равенство элементов из X.

Для третьей аксиомы положим $\rho(x, y) \equiv 1$. Тогда функция никогда не будет равна нулю, значит 2 аксиома будет всегда верна в силу ложности посылки. 4 и 5 аксиомы тоже верны, но $\rho(x, x) = 1 \neq 0$.

Независимость четвертой аксиомы покажем с помощью функции $\rho(x, y) = x - y$ при $x - y \geqslant 0$ и $\rho(x, y) = 1$ в противном случае ($X = \mathbb{R}$). Тогда легко проверяются аксиомы 2 и 3. Аксиома 4 не выполняется для $x - y > 0$. Неравенство треугольника проверяется перебором случаев.

Для доказательства независимости 5 аксиомы применим известный факт о нарушении неравенства треугольника для метрики Минковского при $0 < p < 1$. Тогда достаточно в качестве контрпримера рассмотреть расстояние Минковского для $p = 0.5$

Было показано, что не обязательно требовать от ρ неотрицательности, однако в некоторых источниках принято включать это требование в систему аксиом метрики, чтобы сохранить соответствие с интуицией о расстоянии.

Обобщения метрики

В анализе данных далеко не все функции на парах объектах, имеющие смысл расстояния, являются метриками. Таким образом, приходим к следующим обобщениям метрики (используем терминологию в соответствии с [2]):

Псевдометрика — функция $X \times X \to \mathbb{R}$, удовлетворяющая аксиомам метрики, за исключением быть может равенства нулю только на равных элементах из X (удаляем аксиому 2).

Метаметрика — функция $X \times X \to \mathbb{R}$, удовлетворяющая аксиомам метрики, за исключением быть может равенства нулю на равных элементах из X (удаляем аксиому 3).

Квазиметрика — функция $X \times X \to \mathbb{R}$, удовлетворяющая аксиомам метрики, за исключением быть может аксиомы симметричности (удаляем аксиому 4).

Полуметрика — функция $X \times X \to \mathbb{R}$, удовлетворяющая аксиомам метрики, за исключением быть может неравенства треугольника (удаляем аксиому 5).

Если оставить только аксиомы 1 и 3 получим определение *праметрики*, а если к аксиомам праметрики добавить аксиому 2 получим определение *дивергенции*. Приведем некоторые примеры описанных обобщений.

Классический пример псевдометрики в анализе данных — расстояние Махаланобиса: $\rho_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}, S \succeq 0$ (S — симметричная неотрицательно определенная матрица). Аксиомы псевдометрики легко проверяются с учетом неотрицательной определенности S, неравенство треугольника следует из неравенства Коши-Буняковского для скалярного произведения по отношению к S: $(x,y)_S = x^T S y$. Аксиома 2 может нарушаться именно из-за слабого требования к S — неотрицательной определенности. Если потребовать $S \succ 0$ получим метрику, а значит ρ_M будет метрикой, если матрица объекты-признаки будет иметь полный ранг.

Эта псевдометрика является модификацией обычного евклидового расстояния $l_2(x,y) = \sqrt{(x-y)^T(x-y)}$. Идея такой модификации состоит в том, что для S можно построить ортонормированный базис из собственных векторов $\Rightarrow SQ = Q\Lambda \Rightarrow S = Q\Lambda Q^T \Rightarrow S^{-1} = Q\Lambda^{-1}Q^T \Rightarrow \rho_M(x,y) = l_2(\widetilde{x},\widetilde{y})$, где $\widetilde{x} = \Lambda^{-1/2}Q^T x$. Другими словами, ρ_M представляет собой евклидово расстояние в новом признаковом пространстве, переход в которое осуществляется по правилу: $\widetilde{x} = \Lambda^{-1/2}Q^T x$.

Пусть $S = \frac{1}{N-1}\sum\limits_{i=1}^{N}(x_i - \overline{x})(x_i - \overline{x})^T = \frac{1}{N-1}X^T X$ — несмещенная выборочная оценка ковариационной матрицы (выборка предварительно центрирована, $\overline{x} = 0$). То-

гда собственные векторы описывают «направления разброса» данных, а собственные значения — степень разброса в направлении соответствующего вектора. При таком выборе матрицы S расстояние Махаланобиса $\rho_M(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$ «автоматический» приводит пространство к тому, в котором данные имеют единичную ковариационную матрицу (whitening).

Обратимся к примерам других обобщений. Квазиметрикой будет функция расстояния, учитывающая порядок своих аргументов, например, время, соответствующее кратчайшему пути между городскими остановками. В этом случае естественно считать, что время в пути от A до B не равно времени от B до A.

Полуметрикой будет расстояние Минковского $l_p(x,y) = \left(\sum_{i=1}^{d} |x^i - y^i|^p\right)^{1/p}$ при $p < 1$. Для обоснования невыполнения неравенства треугольника достаточно рассмотреть контрпример: пусть $a = (0,0); b = (0,1); c = (1,1)$. В таком случае

$$l_p(a,c) = 2^{1/p} > l_p(a,b) + l_p(b,c) = 2.$$

Дивергенцией будет расхождение Кульбака-Лейблера:

$$D_{KL}(p||q) = \int p(x) ln(\frac{p(x)}{q(x)}) dx,$$

имеющее смысл расстояния между двумя вероятностными распределениями p и q.

Постановка задачи MDS

В ряде задач, оперирующих понятием метрики, мы перечислили в том числе многомерное шкалирование. Остановимся на этой задаче подробнее.

Рассмотрим общую постановку задачи многомерного шкалирования (MDS) [3]: Известно, что существует некоторая коллекция из N объектов произвольной природы, на которой определена некоторая функция расстояния (distance function):

d_{ij} := расстояние между i-м и j-м объектом

Таким образом, про коллекцию объектов известна лишь матрица попарных различий (dissimilarity matrix): $D = \{d_{ij}\}_{i,j=1}^{N}$

Задача многомерного шкалирования: по матрице различий D найти такие N векторов $x_1, ..., x_N \in \mathbb{R}^p$, что $||x_i - x_j|| \approx d_{ij}$. В общем случае строго не оговаривается, что подразумевается под «приближением» расстояний: может рассматриваться в том числе и сохранение отношения порядка. Размерность полученного представления p обычно выбирается небольшой, например, $p = 2, 3$ для задач визуализации.

Обратим внимание на то, что в общей постановке не оговаривается вид функции расстояния. В этом смысле, в качестве d может быть использована не обязательно строго метрика, но существуют теоретические результаты, гарантирующие нахождение итоговой конфигурации $x_1, ..., x_N \in \mathbb{R}^p$, что $||x_i - x_j|| = d_{ij}$, для которых существенен вид функции расстояния d (d должно быть не только строго метрикой, но еще и евклидовой метрикой).

Существует три основных подхода к решению поставленной задачи:

1. Классическое многомерное шкалирование (cMDS):

   Алгоритм cMDS предполагает, что исходная функция расстояния, с помощью которой была построена матрица различий D, является евклидовой: $d(x, y) = ((x - y)^T(x - y))^{1/2}$. Алгоритм восстанавливает по D матрицу Грама, находит ее спектральное разложение и строит такие координаты $\{x_i\}_{i=1}^N$, что выполняется точное равенство $||x_i - x_j|| = d_{ij}$. Размерность полученного представления p может оказаться неудовлетворительной для поставленной задачи. Алгоритм позволяет построить координаты произвольной меньшей размерности, но в таком случае точное равенство не гарантируется.

   Алгоритм работает и в случае неевклидового расстояния d (non-euclidean distance), но в таком случае точное равенство также не гарантируется.

2. Метрическое многомерное шкалирование (mMDS):

   Основная идея подхода метрического многомерного шкалирования — найти такую конфигурацию $\{x_i\}_{i=1}^N$, что для каждой пары расстояние $||x_i - x_j||$ будет наиболее близко к d_{ij}. Это требование может быть сформулировано через минимизацию функционала стресса:

   $$Stress(x_1, ..., x_N) = \left(\sum_{i \neq j = 1, ..., N} (d_{ij} - ||x_i - x_j||)^2 \right)^{1/2}$$

   Решением задачи mMDS в таком случае будет набор $\{x_i\}_{i=1}^N$, на котором достигается минимум функционала стресса.

3. Неметрическое многомерное шкалирование (nMDS):

   В постановке неметрического многомерного шкалирования решается следующая задача: найти такой набор $\{x_i\}_{i=1}^N$, что $\forall i, j, k, l \ ||x_i - x_j|| < ||x_k - x_l|| \iff d_{ij} < d_{kl}$. Ключевая идея этого подхода: сохранить порядок на всевозможных парах расстояний.

Общая идея решения задачи MDS

Рассмотрим общую идею решения задачи:

1. Про исходную коллекцию объектов известна лишь матрица попарных различий $D = \{d_{ij}\}_{i,j=1}^{N}$. Тогда под объектом модели машинного обучения x будем понимать пару исходных объектов из коллекции, а под целевой переменной y — соответствующее расстояние:

   $$(\underbrace{obj_i, obj_j}_{x}, \underbrace{d_{ij}}_{y})$$

2. Общая схема модели машинного обучения следующая:

   $$\left.\begin{array}{c} x \longrightarrow \text{граф вычислений} \longrightarrow \text{output} \\ y \end{array}\right\} \longrightarrow \text{loss}$$

   Вид графа вычислений и loss-функции конкретизируем позднее для конкретной задачи многомерного шкалирования.

3. По исходной коллекции формируется выборка $\{(x_i, y_i)\}_{i=1}^{l}$. Модель обучается исходя из минимизации функционала $Q = \sum_{i=1}^{l} \text{loss}(x_i, y_i)$, параметрами модели являются параметры графа вычислений.

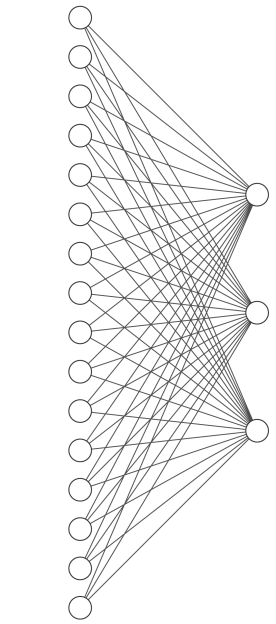
Решение задачи метрического многомерного шкалирования

Конкретизируем теперь описанную выше идею решения на случай метрического многомерного шкалирования mMDS. Итоговую цель работы нашего алгоритма можно описать так: object $\longrightarrow$ representation, где representation — построенное координатное представление для объекта, вектор x_i в общей постановке MDS.

Схема модели будет выглядеть так:

$$\left.\begin{array}{l} object_i \longrightarrow \text{representation}_i \\ object_j \underbrace{\longrightarrow}_{\text{учим}} \text{representation}_j \end{array}\right\} \longrightarrow \text{distance} \left.\begin{array}{c} \\ \\ \text{actual distance} \end{array}\right\} \longrightarrow \text{loss}$$

Граф вычислений, осуществляющий преобразование object $\longrightarrow$ representation может быть устроен следующим образом [1]:



В этом примере граф вычислений представляет из себя один полносвязный слой нейронной сети без активации (Dense), принимающий на вход $object_i$ — one-hot закодированное представление номера i-го объекта, возвращающий искомое представление representation_i размерности p (например, $p = 3$). По сути, для каждого объекта $object_i$ такая сеть будет хранить p чисел — веса соответствующие i-му входу. Таким образом, предложенная модель эквивалентна случайному заданию координат $\{x_i\}_{i=1}^N$ с последующим нахождением оптимальной конфигурации путем минимизации функционала стресса.

Реализация алгоритма mMDS на PyTorch

Рассмотрим конкретную реализацию предложенного алгоритма решения задачи mMDS с помощью фреймворка глубокого обучения PyTorch, обратим внимание на особенности, которые нужно учесть при реализации, оценим качество полученных представлений $\{x_i\}_{i=1}^N$. Наша задача состоит в обучении сети построению для объектов координатных представлений в некотором маломерном координатном пространстве. Наша модель предполагает, что сеть получает параллельно на вход два

разных объекта i и j, а затем сравнивает расстояние между полученными представлениями с реальным расстоянием на этой паре, известным еще из матрицы попарных различий D. Логику получения сетью пары объектов реализуем с помощью сиамской сети (siamese neural network). Выходы сиамской сети — пару координатных представлений подадим на вход некоторой функции расстояния (в наших экспериментах эта функция расстояния совпадает с d — функцией, с помощью которой строилась матрица попарных различий D, но, вообще говоря, в качестве функции расстояния на представлениях можно использовать произвольную другую функцию). Полученное расстояние сравниваем с заданным различием из матрицы различий с помощью *loss*-функции, откуда и начинаем обратное распространение ошибки. В качестве функции потерь в экспериментах использовалась средняя квадратическая ошибка.

Объекты i и j, идущие на вход сети, описываем с помощью one-hot закодированного вектора с единицей в позиции, соответствующей номеру объекта (под номером объекта мы понимаем его номер в матрице различий D), и нулями на всех остальных позициях. Сиамскую сеть реализуем с помощью цикла for по входным данным в методе forward класса Net, описывающего сеть. Таким образом будет соблюдено ключевое требование сиамской сети — общее разделение весов.

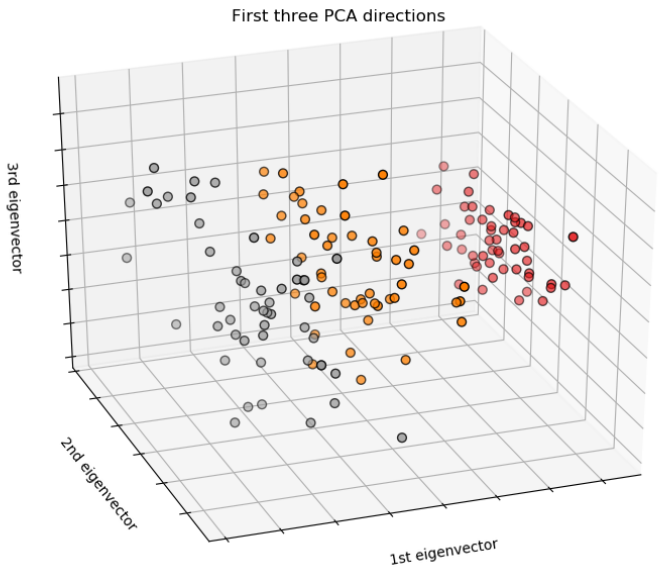
Очевидно, что хранить всю матрицу попарных различий в памяти не представляется возможным, потому что она занимает квадратичную память от исходного размера выборки. Для обучения сети в нашем случае все равно придется перебрать все пары объектов, поэтому будет естественным подавать данные на вход батчами, а для генерации батчей по матрице различий написать итератор. В нашем случае, в экспериментах сеть для тестирования сети использовались полноценные датасеты, где объекты представлены полноценным признаковым описанием, а не номером строки или столбца в матрице попарных расстояний. Поэтому в этом случае итератор перебирал всевозможные пары объектов, самостоятельно вычислял расстояние на каждой паре, строил one-hot векторы, а затем возвращал батчи, где каждая строка соответствует некоторой паре, и для этой пары возвращается тройка: one-hot векторы на вход сети и истинное расстояние на вход функции потерь.

Так как мы приняли решение обучать сеть на батчах, то обратное распространение ошибки с последующим смещением весов сети реализуем с помощью стохастического градиентного спуска *SGD*. Шаг обучения *learning rate* подбирался в диапазоне от 10^{-4} до 10^{-7} из соображений сходимости. Под эпохой в нашем случае будем понимать прохождение по одному разу каждой пары объектов в выборке

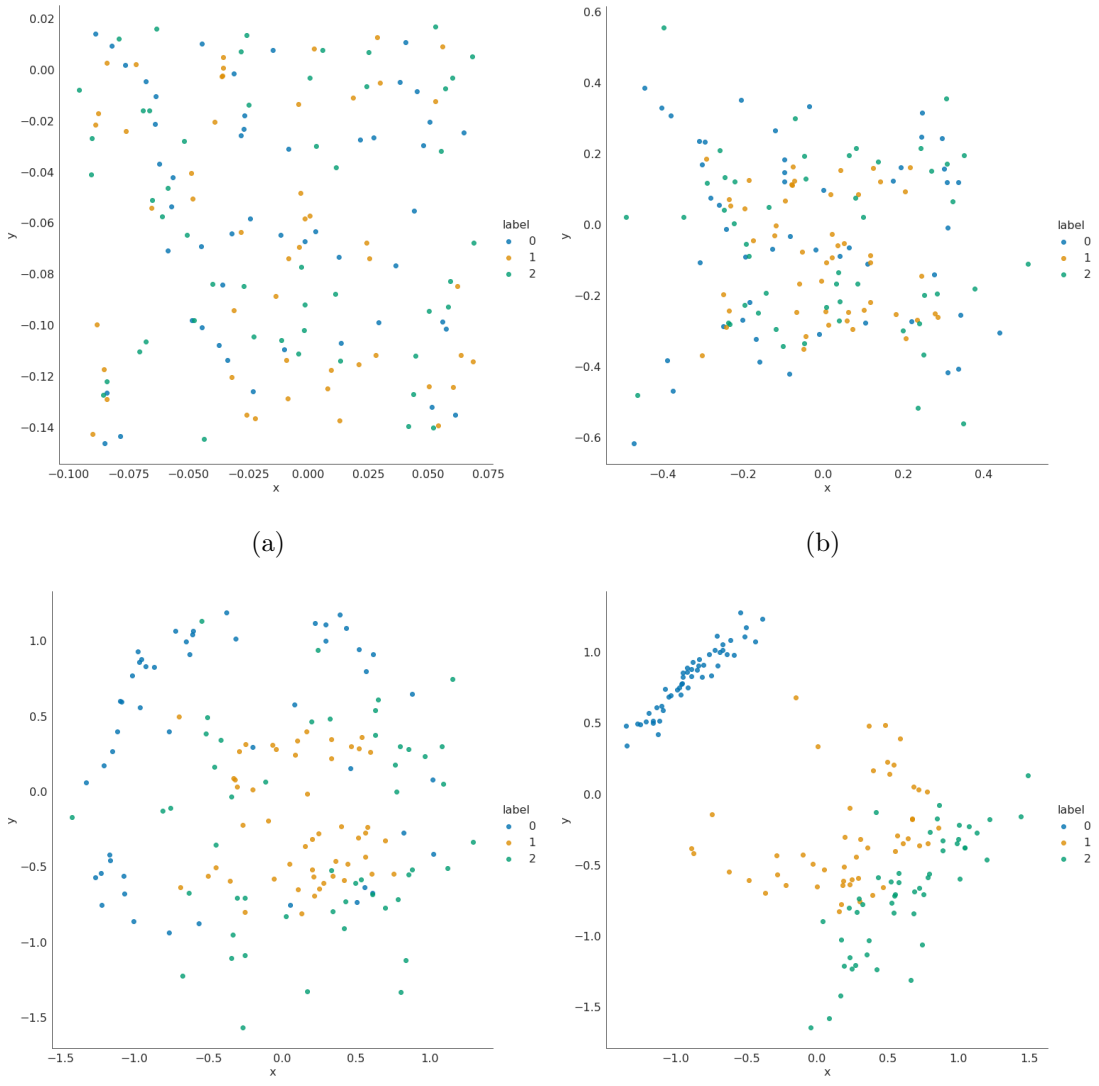
при обучении сети. Для визуализации работы алгоритма по ходу обучения сети и по окончании работы использовался непосредственно полносвязный слой, осуществляющий преобразование объекта в координатное представление. При этом особенно важно вычисления сети, использующиеся в визуализации, не учитывать в общем графе вычислений, по которому затем распространяется ошибка, то есть использовать, например, менеджер контекста $with\ torch.no\_grad()$.

Еще одна потенциальная ошибка, с которой можно столкнуться при реализации, это переполнение при вычислении $loss$ функции. В некоторых экспериментах первоначально при генерации батчей использовалась функция расстояния, являющаяся квадратом евклидового расстояния. На многомерных датасетах при достаточно большом размере батчей квадрат такого расстояния на одной паре объектов уже был порядка 10^4, а общий $loss$ переполнялся, становился nan и сеть не обучалась. Решением проблемы оказалось использование классического евклидового расстояния, то есть потребовалось просто добавить в функцию расстояния корень, чтобы не получать таких больших расстояний на парах.

Теперь продемонстрируем работу алгоритма. Для примера возьмем 4-мерный датасет ирисов, каждый объект которого принадлежит одному из трех классов. Трехмерная визуализация этого датасета с помощью метода главных компонент выглядит так:



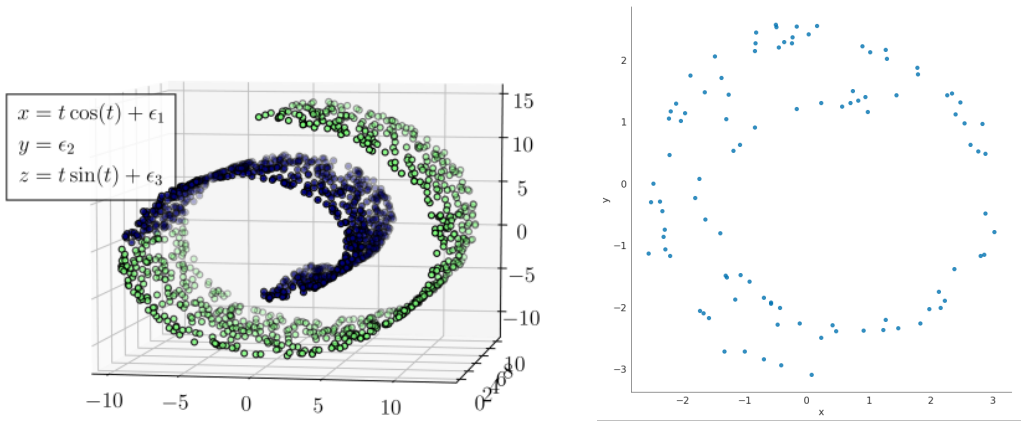
Визуализируем динамику обучения сети и качество полученных представлений относительно числа эпох. Будем с помощью построенного алгоритма находить представления $\{x_i\}_{i=1}^N$ при $p = 2$ (Рис. 1).



(a) (b)

(c) (d)

Рис. 1: (a) 0 эпох (b) 100 эпох (c) 350 эпох (d) 1000 эпох



(a) (b)

Рис. 2: (a) Swiss Roll (b) Представление при $p = 2$

В соседней для MDS задаче понижения размерности для демонстрации того, что алгоритм сохраняет структуру в данных часто используют специфические синтетические датасеты. Возьмем для примера такого датасета Swiss Roll датасет (Рис. 1a). Датасет представляет собой 100 трехмерных объектов. Наш алгоритм после нескольких сотен эпох воспроизводит «спираль» (Рис. 2b).

Решение задачи неметрического многомерного шкалирования

Рассмотрим теперь случай неметрического многомерного шкалирования. Ключевая идея этого подхода — сохранить отношение порядка на парах расстояний. Будем добиваться выполнения этого требования с помощью triplet-loss. Классический подход к использованию triplet loss подразумевает формирование по выборке троек — точки интереса (anchor point), точки из того же класса, что и точка интереса (anchor-positive), точки другого класса (anchor-negative).

Пример triplet loss:

$$L = max(0, d_{ap} - d_{an} + m)$$

Здесь d_{ap} — расстояние от представления anchor point до представления anchor-positive, d_{ap} — до представления anchor-negative соответственно, m — отступ, гарантирующий выполнение $|d_{ap} - d_{an}| \geqslant m$.

В нашей задаче отсутствуют метки anchor-positive и anchor-negative на объектах. Однако, мы можем использовать в качестве anchor-positive и anchor-negative для объекта с номером i такие объекты с номерами j и k соответственно, что $d_{ij} < d_{ik}$.

Тогда наша модель по-прежнему осуществляет преобразование object $\longrightarrow$ representation, содержит в себе параметры, которые нужно обучить (например, можно по-прежнему использовать полносвязный слой без активации), а схема обучения модели будет выглядеть следующим образом:

$$\left.\begin{array}{l}\left.\begin{array}{l}\text{representation}_i \\ \text{representation}_j\end{array}\right\} \longrightarrow \text{distance}_{ij} \\ \left.\begin{array}{l}\text{representation}_i \\ \text{representation}_k\end{array}\right\} \longrightarrow \text{distance}_{ik}\end{array}\right\} \longrightarrow \text{triplet loss}$$

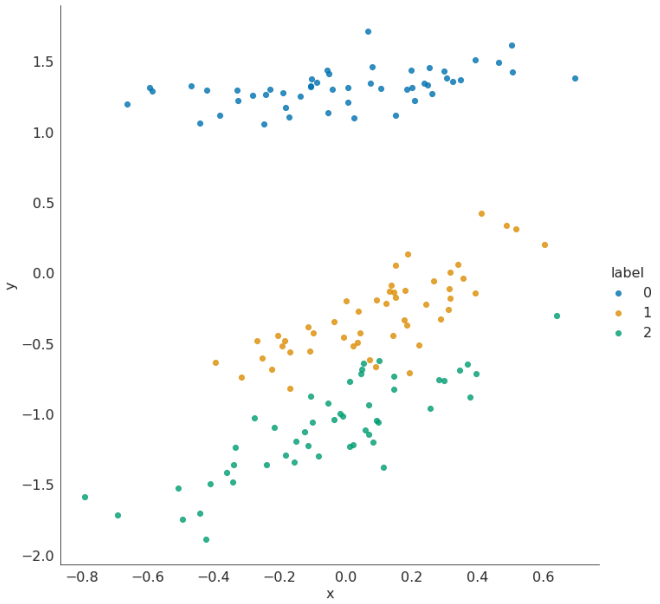
Возможны модификации представленной модели. Например, кажется более естественным использовать четверки объектов: про одну пару (i, j) известно, что d_{ij} мало, объекты близки друг к другу, а про вторую пару (k, l) известно, что наоборот d_{kl} велико, объекты значительно удалены друг от друга. Полученные расстояния на представлениях для каждой пары будем подавать уже в quadruplet loss, логика работы которого та же, что и у triplet loss — штрафовать за большие расстояния на изначально близких объектах и за маленькие расстояния на изначально удаленных.

Реализация алгоритма nMDS на PyTorch

Рассмотрим конкретную реализацию предложенного алгоритма решения задачи nMDS с помощью фреймворка глубокого обучения PyTorch, обратим внимание на особенности, которые нужно учесть при реализации, оценим качество полученных представлений $\{x_i\}_{i=1}^N$, укажем на отличия в реализации по сравнению с mMDS. В случае метрического многомерного шкалирования нам требовалось параллельно считать представления для каждого объекта в паре, чтобы затем вычислить расстояние на этой паре представлений. В случае же неметрического многомерного шкалирования нам при использовании triplet loss потребуется параллельно считать представления для каждого объекта в тройке. Поэтому будем также использовать сиамскую сеть, выходы которой — тройку координатных представлений будем подавать на вход в triplet loss, предварительно взяв некоторую точку как anchor point, а затем определив anchor-positive и anchor-negative точки так, что $d_{ap} < d_{an}$. Вся остальная

логика работы сети и ее обучения строится по аналогии с сетью, решающей задачу mMDS: используем сиамскую сеть на тройках объектов, тройки строятся по аналогии с предыдущей сетью: итератор перебирает всевозможные пары объектов, третий объект к паре выбирается случайным образом. Далее полученные представления вместе с разметкой anchor, anchor positive и anchor negative подаются на вход в triplet loss, после чего следует обратное распространение ошибки. Данные для обучения аналогично разбиваются на батчи, оптимизация происходит стохастическим градиентным спуском. Под одной эпохой в этом случае будем понимать проход итератором всех пар по одному разу (напомним, что пара достраивается до тройки добавлением случайного объекта). Из-за случайного выбора третьего объекта к каждой паре, для достижения сходимости теперь потребуется больше эпох.

Аналогично продемонстрируем работу алгоритма на датасете ирисов. Требовалось найти координатные представления для $p = 2$, обучение сети заняло 2000 эпох (впрочем, конфигурация точек незначительно изменилась с конфигурации на 1000 эпох). Визуально более явно прослеживается граница между классами 1 и 2 по сравнению с аналогичной конфигурацией, полученной с помощью алгоритма mMDS.



Дальнейшие исследования

Опишем направления дальнейших исследований по решению поставленной задачи, а также идеи, которые в дальнейшем планируется реализовать.

1. Более естественно при решению nMDS использовать четверки объектов, вместо

троек: пара заведомо близких объектов и пара заведомо далеких. Задача на будущее: сконструировать функцию потерь, которая будет принимать четверки объектов — *quadruplet loss*, перестроить структуру на сети на случай такой функции потерь, обучить сеть и получить результаты.

2. Обратить внимание на то, каким образом строить тройки для triplet loss и четверки для quadruplet loss — *triplet mining* и *quadruplet mining*.

3. В ходе подготовки к обзору подходов к решению задачи многомерного шкалирования была написана программа, позволяющая курсором расставлять точки на экране и двигать их. Дальнейшая задача: совместить эту программу с реализацией одного из алгоритмов многомерного шкалирования — *интерактивный визуальный анализ устойчивости решения*.

4. Рассмотреть случай метрик, зависящих от параметра. Для этого случая будет уже не одна матрица попарных различий, а трехмерный тензор. Рассмотреть аналогичную MDS задачу для тензора различий — *модель индивидуальных различий*.

Результаты

1. Изучены аксиомы метрики, показана их зависимость как системы аксиом, найдена соответствующая независимая подсистема, перебором контрпримеров доказана независимость.

2. Проведена классификация обобщений понятия метрики (псевдометрика, полуметрика, квазиметрика и т.д.), рассмотрен ряд соответствующих примеров, в том числе из анализа данных.

3. Сделана заготовка для визуализации процесса решения задачи многомерного шкалирования на .NET, позволяющая пользователю задать начальные координаты в двумерном признаковом пространстве мышью, а также передвигать уже заданные точки.

4. Проведен обзор существующих подходов к решению задачи многомерного шкалирования. Изучен классический алгоритм cMDS, проверена его эквивалентность методу главных компонент. На эту тему проведен доклад.

5. Проведен обзор нейросетевых подходов к решению задачи многомерного шкалирования. Описана общая идея решения задачи MDS, используя нейронные сети. Эта идея конкретизирована на случай метрического и неметрического многомерного шкалирования.

6. Предложенные алгоритмы решения mMDS и nMDS реализованы с помощью фреймворка PyTorch. Описаны подробности реализации, подчеркнуты особенности при реализации конкретных алгоритмов. Отдельно выделены потенциальные ошибки при реализации, сформулированы советы по их исправлению.

7. С помощью реализованных алгоритмов проведены эксперименты с несколькими датасетами, показана работоспособность предложенных алгоритмов, результаты визуализированы.

8. Поставлены дальнейшие задачи, сформулированы идеи, которые планируется реализовать.

Заключение

В рамках проведенной работы была поставлена задача, рассмотрены основные подходы к ее решению, был проведен анализ существующих идей ее решения, в том числе использующих нейросети. В результате была предложена модель, решающая поставленную задачу в случае метрического и неметрического многомерного шкалирования, была описана логика ее работы и обучения. Алгоритм, использующий предложенную модель, реализован на PyTorch, приведены примеры его работы.

Литература

[1] Wezel, Michiel & Kok, Joost & Kosters, Walter. (1997). Two Neural Network Methods for Multidimensional Scaling.

[2] Wikipedia contributors. (2020, May 18). Metric (mathematics). In Wikipedia, The Free Encyclopedia. Retrieved 20:30, June 1, 2020, from https://en.wikipedia.org/w/index.php?title=Metric_(mathematics)&oldid=957282191

[3] Wikipedia contributors. (2020, April 13). Multidimensional scaling. In Wikipedia, The Free Encyclopedia. Retrieved 20:33, June 1, 2020, from https://en.wikipedia.org/w/index.php?title=Multidimensional_scaling&oldid=950612463