Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра суперкомпьютеров и квантовой информатики

Кучеров Василий Дмитриевич

Анализ влияния выбора функции потерь на эффективность обучения по нескольким примерам

Отчет о проделанной работе в осеннем семестре
в рамках курсовой работы

Научный руководитель:
канд. физ.-мат. наук
Д.Ю.Буряк

Москва
2022

Содержание

1. Введение

Машинное обучение призвано решать задачи для которых сложно или невозможно предоставить алгоритм. Таким образом, с использованием методов машинного обучения решается широкий спектр задач: обнаружение и классификация объектов на изображениях и видео, обработка естественного языка, преобразование текста в аудио и обратно. Одной из основных проблем такого подхода к решению задач является сбор данных для обучения, которых зачастую требуется много, чтобы получить результат высокого качества. Малый объем данных типичен для таких задач как: распознование ключевых слов (Keyword spotting) для умных колонок и других устройств, поддерживающих голосовой интерфейс с пользователем, распознование лиц или отпечатков пальцев для использование вместа пароля (напрмер разблокировка телефона), открытие лекарств при помощи определения свойств новых молекул. Для решения данной проблемы было предложено обучение по нескольким примерам (Few Shot Learning или FSL).

2. Обзор обучения по нескольким примерам

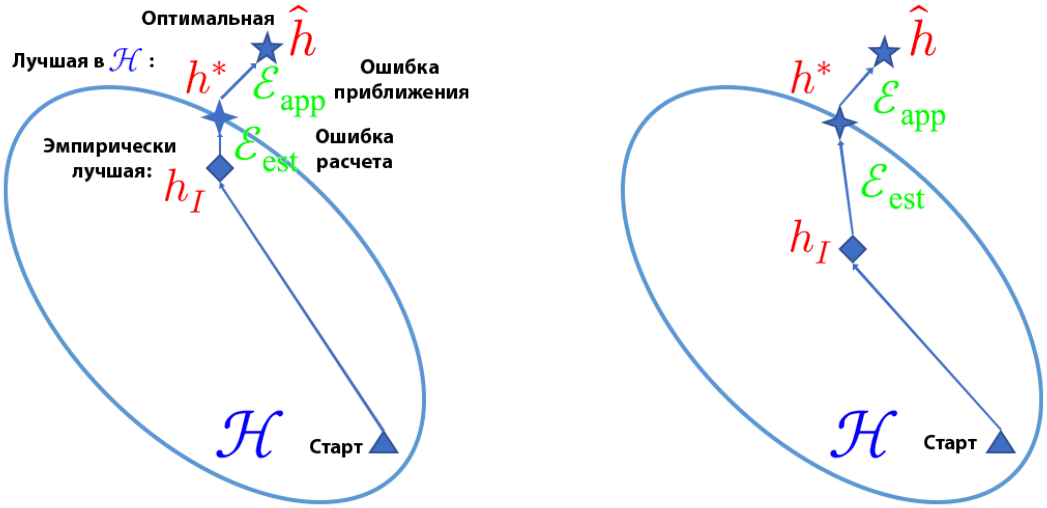
2.1. FSL

Для понимания принципа работы различных подходов в FSL удобно формализовать задачу: Пусть R - ожидаемый риск; $\mathcal{H}$ - пространство гипотез; $\hat{h}$ - функция, минимизирующая ожидаемый риск; h^* - функция в $\mathcal{H}$, минимизирующая ожидаемый риск; h_I - функция в $\mathcal{H}$, минимизирующая эм-

пирический риск. Тогда общую ошибку можно представить в виде [2]:

$$\mathbb{E}\left[R\left(h_I\right) - R(\hat{h})\right] = \underbrace{\mathbb{E}\left[R\left(h^*\right) - R(\hat{h})\right]}_{\mathcal{E}_{\text{app}}\left(\mathcal{H}\right)} + \underbrace{\mathbb{E}\left[R\left(h_I\right) - R\left(h^*\right)\right]}_{\mathcal{E}_{\text{est}}\left(\mathcal{H}, I\right)}$$

$\mathcal{E}_{\text{app}}$ - ошибка приближения (approximation), показывающая насколько хорошо могут функции в $\mathcal{H}$ аппроксимировать $\hat{h}$; $\mathcal{E}_{\text{est}}$ - ошибка расчета (estimation) показывает насколько хорошо мы можем приблизиться к h^* в $\mathcal{H}$, минимизируя эмпирический риск.



(а) Большая I **(б) Малая I**

Рис. 2.1. Влияние размера обучающей выборки I на h_I

Используя некотрые предварительные знания о задаче можно выделить три подхода к уменьшению $\mathcal{E}_{\text{est}}$:

- Данные. Увеличение обучающей выборки за счет преобразования объектов заданного датасета, использования объектов из неразмеченных, слаборазмеченных датасетов или похожих датасетов.
- Модель. Упрощение пространства гипотез $\mathcal{H}$. Тогда в новом пространстве $\tilde{\mathcal{H}}$ имеющейся обучающей выборки будет достаточно для обучения. Различают многозадачное обучение, обучение эмбеддингов, обучение с внешней памятью и генеративное моделлирование.

- Алгоритм. Выбор хорошей стартовой точки для обучения или добавление дополнительного шага в процессе обучения помимо шага минимизатора эмпирического риска. Выделяют такие методы, как уточнение существующих параметров, уточнение мета-обученных параметров и обучение оптимизатора.

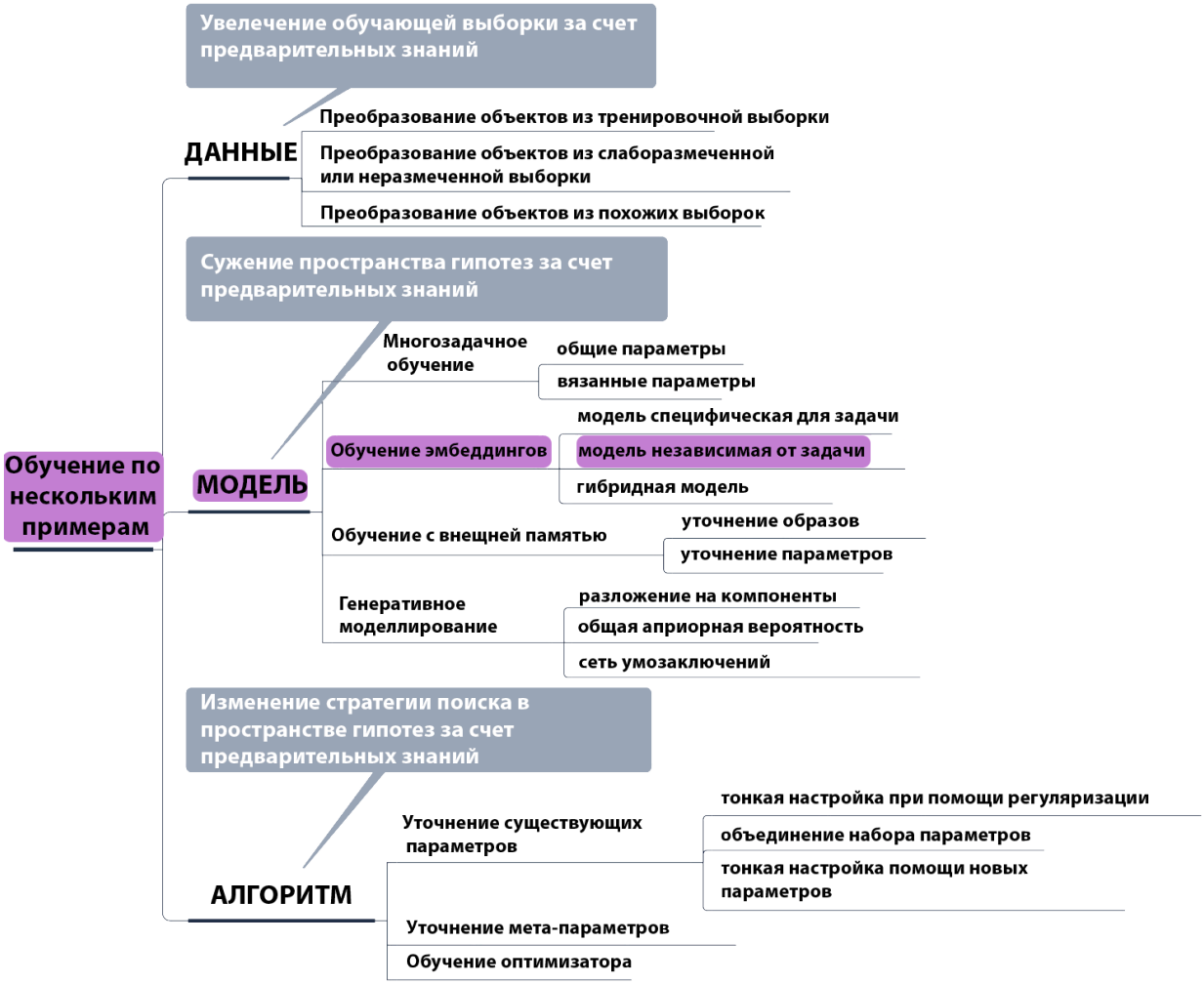


Рис. 2.2. Схема подходов в FSL

Поскольку текущая работа тесно связана с обучением эмбеддингов остановимся подробнее на данной теме.

2.2. Обучение эмбеддингов

Суть построения эмбеддингов - перевод объктов из начального пространства $x_i \in \mathcal{X} \subset \mathbb{R}^n$ в пространство меньшей размерности $z_i \in \mathcal{Z} \subset \mathbb{R}^m, m < n$, в котором уже и производить классификацию. Причем $\mathcal{Z}$ должно быть таким, что похожие примеры должны быть близки друг к другу, а различные - удалены.

Одним из наиболее простых методов построения сети эмбеддингов является использование части классификатора: сначала строится сеть, классифицирующая на большое число классов, а после удаляется последний слой (слой вывода), оставшая часть и есть генератор эмбеддингов (Рис. 2.3).

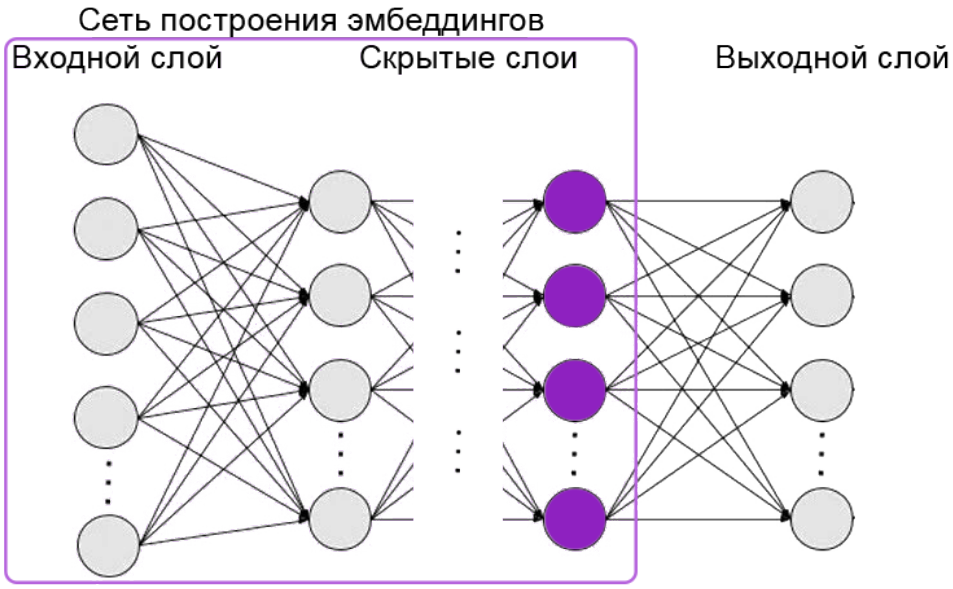


Рис. 2.3. Построение генератора эмбеддингов с помощью классификатора

Основные компоненты обучения эмбеддингов: функция f, переводящая тестовый пример $x_{\text{test}} \in D_{\text{test}}$ в пространство эмбеддингов $\mathcal{Z}$, функция g, переводящая объекты из тренировочного датасета $x_i \in D_{\text{train}}$ в $\mathcal{Z}$, функция схожести (расстояния) $s(x_i, x_j)$. Обычно функции f и g совпадают.

По тому, как меняются параметры функций f и g от задачи к задаче выделяют три метода построения эмбеддингов: индивидуальный для каждой задачи (task-specific), независимый (task-invariant) и гибридный.

- **Task-specific (пецифическая для каждой задачи)**. Обучение проводится только на объектах из обучающей выборки для конкретной задачи.

- **Task-invariant (модель независящая от задачи)**. Обучение проводится на большой обучающей выборке, а после обученная модель применяется без переобучения для специфической задачи.

- **Гибридная модель**. Гибридная модель объединяет в себе предыдущие два подхода: эмбеддинги, полученные при использовании независимой модели, используются как параметр для построения эмбеддингов для конкретной задачи.

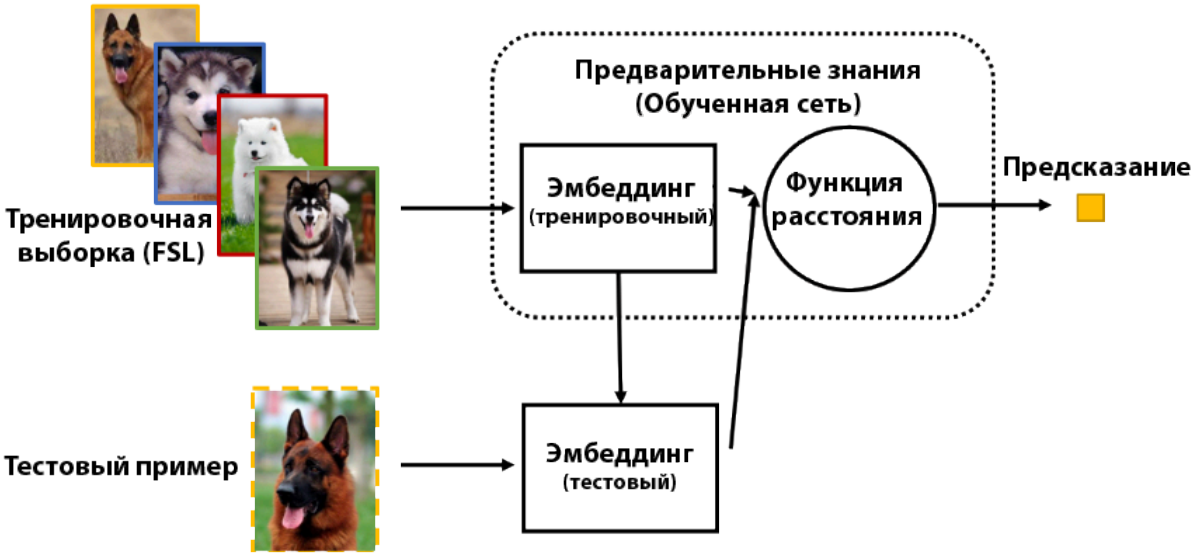


Рис. 2.4. Гибридная модель построения эмбеддингов

3. Кластеризация

3.1. Введение

Как уже было сказано в главе обучение эмбеддингов мы хотим, чтобы в новом пространстве объекты были хорошо разделимы и образовывали кластера. Это особенно важно для экстремальных случаев FSL: обучение по одному примеру (One Shot Learning или OSL) или обучение без примеров (Zero Shot Learning или ZSL). На Рис. 3.1 точками изображены примеры двух синтетических классов. Если кластеры достаточно удалены, то нам не важно какие точки выбрать (какие примеры нам попадутся в обучающей выборке) для построения разделяющей прямой, а если кластеры близки или вообще пересекаются, то стабильно строить хорошую разделяющую прямую не получится, все будет зависеть от полученных примеров.

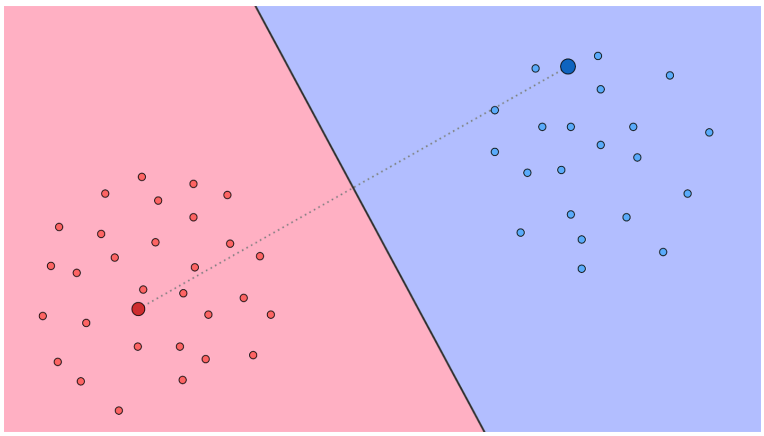
3.2. Оценка качества

Для определения качества кластеризации вводятся следующие функции [1]:

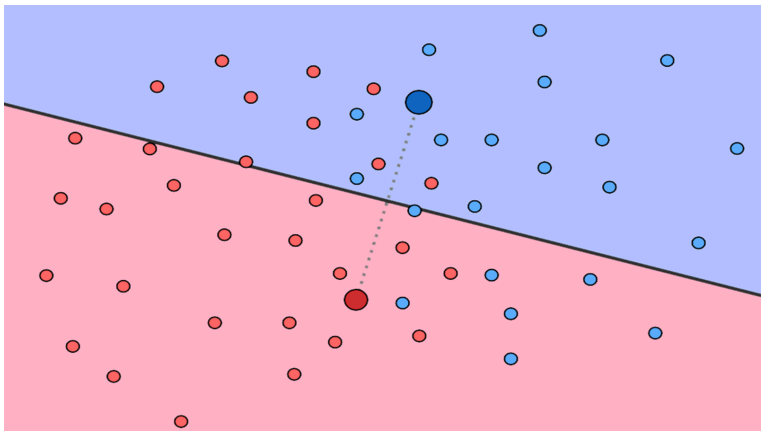
- Отношение внутриклассовой к межклассовой дисперсии (intra-class to inter-class variance ratio):

$$\frac{\sigma^2_{\text{within}}}{\sigma^2_{\text{between}}} = \frac{\frac{\sum_{i,j}\left\|\phi_{i,j}-\mu_i\right\|_2^2}{N}}{\frac{\sum_i\left\|\mu_i-\mu\right\|_2^2}{C}} = \frac{C}{N}\frac{\sum_{i,j}\left\|\phi_{i,j}-\mu_i\right\|_2^2}{\sum_i\left\|\mu_i-\mu\right\|_2^2},$$

  где C - количество классов, N - количество объектов в классе, $\phi_{i,j}$ - эмбеддинг j-ого объекта в i-ом классе, μ_i - среднее эмбеддингов i-ого класса, μ - среднее эмбеддингов всех классов. Чем меньше значение полученного выражения, тем лучше кластеризация.



(a)



(b)

Рис. 3.1. Примеры хорошей (a) и плохой (b) класетризации

- Ограничение разброса гиперплоскостей: Пусть x_1, x_2 - объекта одного класса, y_1, y_2 - объекты другого класса, $f_\theta(x)$ - эмбеддинг объекта x, тогда выражение $f_\theta(x_1) - f_\theta(y_1)$ показывает направление разделяющей гиперплоскости с наибольшим зазором. Следуешее выражение показывает насколько разные гиперплоскости можно построить, выбирая различные пары объектов из двух классов:

$$R_{HV}\left(f_\theta(x_1), f_\theta(x_2), f_\theta(y_1), f_\theta(y_2)\right) = \frac{\|(f_\theta(x_1) - f_\theta(y_1)) - (f_\theta(x_2) - f_\theta(y_2))\|_2}{\|(f_\theta(x_1) - f_\theta(y_1)\|_2 + \|f_\theta(x_2) - f_\theta(y_2)\|_2}.$$

Как и в первом случае чем меньше значение R_{HV}, тем лучше кластеризация.

4. Функции потерь

Теперь рассмотрим специфические для задачи построения эмбеддингов функции потерь [3], предполагаемо улучшающие кластеризацию.

4.1. Contrastive loss

Contrastive loss для каждой пары объектов (x_i, x_j) минимизирует расстояние эмбеддингов, если они из одного класса, и максимизирует, если они из разных классов:

$$\mathcal{L}_{\text{cont}}\left(\mathbf{x}_i, \mathbf{x}_j, \theta\right) = \mathbb{1}\left[y_i = y_j\right] \left\|f_\theta\left(\mathbf{x}_i\right) - f_\theta\left(\mathbf{x}_j\right)\right\|_2^2 + \mathbb{1}\left[y_i \neq y_j\right] \max\left(0, \epsilon - \left\|f_\theta\left(\mathbf{x}_i\right) - f_\theta\left(\mathbf{x}_j\right)\right\|_2\right)^2$$

Где ϵ это гиперпараметр, определяющий минимальное расстояние между объектами разных классов.

4.2. Triplet loss

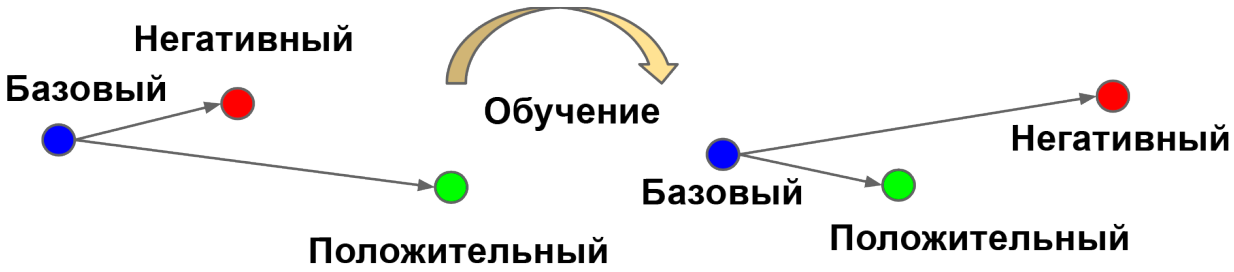


Рис. 4.1. Схема работы Triplet Loss

Triplet Loss развивает идею Contrastive loss, но расчет идет сразу по трем объектам: базовый пример $\mathbf{x}$, положительный пример (того же класса, что и базовый) $\mathbf{x}^+$ и один негативный пример (другого класса) $\mathbf{x}^-$. Triplet loss одновременно увеличивает расстояние между базовым и негативным объек-

тами и уменьшает расстояние между базовым и положительным. (Рис. 4.1):

$$\mathcal{L}_{\text{triplet}}\left(\mathbf{x}, \mathbf{x}^{+}, \mathbf{x}^{-}\right) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \left\|f(\mathbf{x}) - f\left(\mathbf{x}^{+}\right)\right\|_2^2 - \left\|f(\mathbf{x}) - f\left(\mathbf{x}^{-}\right)\right\|_2^2 + \epsilon\right)$$

4.3. Lifted Structured loss

Lifted Structured loss работает по тому же принципу, что и Triplet loss, однако он использует все пары объектов внутри одного тренировочного батча (Рис. 4.2).

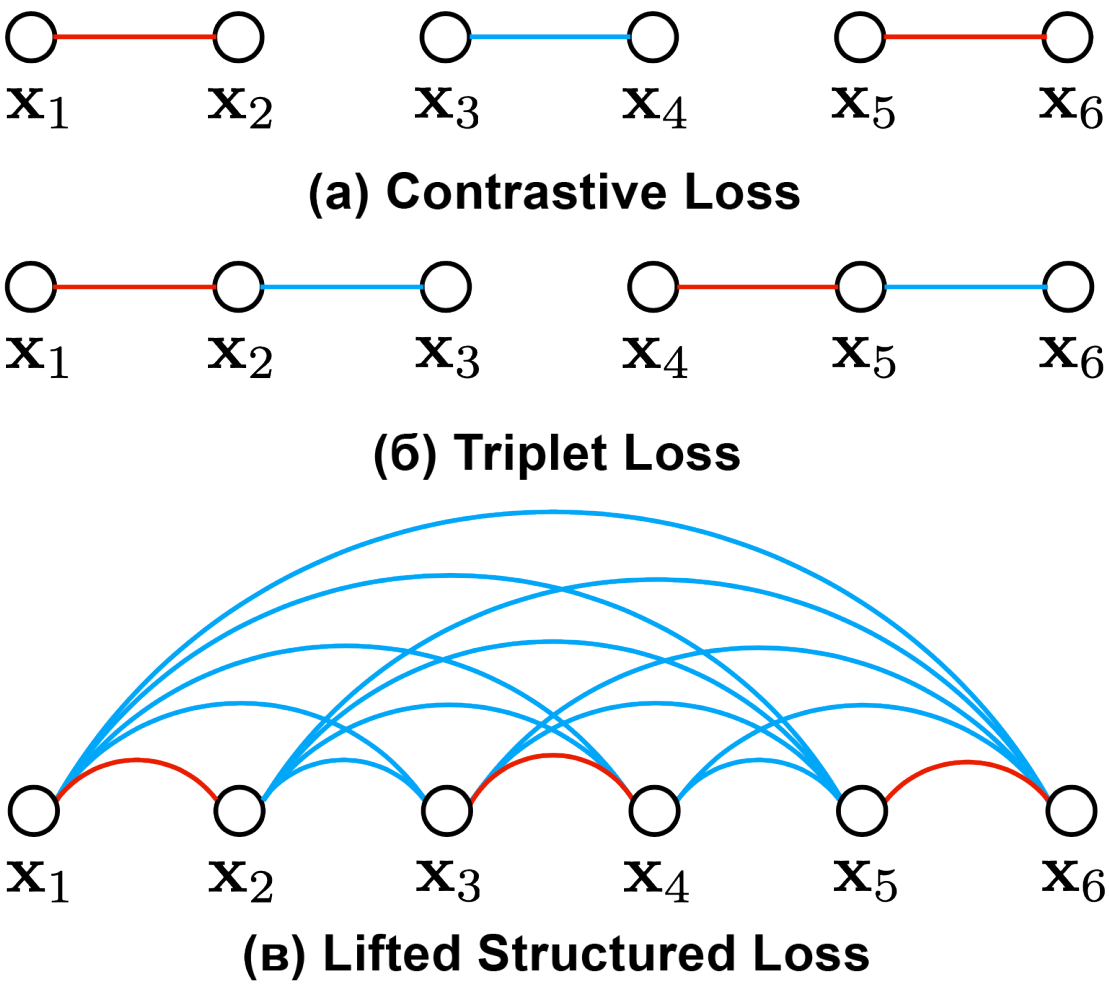


Рис. 4.2. Сравнение функций потерь по выбору объектов

Пусть $D_{ij} = \left|f\left(\mathbf{x}_i\right) - f\left(\mathbf{x}_j\right)\right|_2$, $\mathcal{P}$ - набор положительных примеров, $\mathcal{N}$ - набор отрицательных примеров, тогда:

$$\mathcal{L}_{\text{struct}} = \frac{1}{2|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \max\left(0, \mathcal{L}_{\text{struct}}^{(ij)}\right)^2$$

$$\text{where } \mathcal{L}_{\text{struct}}^{(ij)} = D_{ij} + \max\left(\max_{(i,k)\in\mathcal{N}} \epsilon - D_{ik}, \max_{(j,l)\in\mathcal{N}} \epsilon - D_{jl}\right)$$

Правая часть в $\mathcal{L}_{\text{struct}}^{(ij)}$ призвана искать сложные негативные примеры (наиближайшие отрицательные примеры к базовому). Однако это функция ступенчатая, что может вызвать проблемы во время обучения, поэтому используют её сглаженную версию:

$$\mathcal{L}_{\text{struct}}^{(ij)} = D_{ij} + \log\left(\sum_{(i,k)\in\mathcal{N}} \exp\left(\epsilon - D_{ik}\right) + \sum_{(j,l)\in\mathcal{N}} \exp\left(\epsilon - D_{jl}\right)\right)$$

4.4. N-pair loss

N-pair loss - это обобщение triplet loss, использующее не один, а несколько ($N-1$) негативных примеров. Таким образом, выбирается базовый пример, один положительный и ($N-1$) негативных примеров.

$$\mathcal{L}_{\text{N-pair}}\left(\mathbf{x}, \mathbf{x}^+, \{\mathbf{x}_i^-\}_{i=1}^{N-1}\right) = \log\left(1 + \sum_{i=1}^{N-1} \exp\left(f(\mathbf{x})^\top f\left(\mathbf{x}_i^-\right) - f(\mathbf{x})^\top f\left(\mathbf{x}^+\right)\right)\right)$$

$$= -\log \frac{\exp\left(f(\mathbf{x})^\top f\left(\mathbf{x}^+\right)\right)}{\exp\left(f(\mathbf{x})^\top f\left(\mathbf{x}^+\right)\right) + \sum_{i=1}^{N-1} \exp\left(f(\mathbf{x})^\top f\left(\mathbf{x}_i^-\right)\right)}$$

5. Цель и постановка задачи

Цель работы - провести исследование влияния выбора функции потерь на эффективность обучения по нескольким примерам.

Постановка задачи:

- Реализовать описанные в разделе 4 функции потерь
- Реализовать обучение эмбеддингов для задачи распознавания голосовых команд на наборе данных Google Speech. Выбран task-invariant подход, то есть обучение будет проводится на большом объеме данных, но не будет производится переобучения под новые специфические команды.
- Провести анализ разделимости получаемых кластеров с использованием описанных в разделе разделе 3.2 метрик качества.
- Составить список функций потерь, при использовании которых достигнуто наиболее высокое качество кластеризации пространства эмбеддингов.

Источники

[1] Goldblum, M., Reich, S., Fowl, L., Ni, R., Cherepanova, V., and Goldstein, T. Unraveling meta-learning: Understanding feature representations for few-shot tasks.

[2] WANG, Y., YAO, Q., KWOK, J. T., and NI, L. M. Generalizing from a few examples: A survey on few-shot learning.

[3] Weng, L. Contrastive representation learning. lilianweng.github.io (May 2021).