Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

Николаев Владимир Владимирович

Дифференцируемые метрические методы на основе моделей с памятью

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

н.с.

Д. А. Кропотов

Москва, 2017

Содержание

1 Введение **2**

1.1 Memory Networks 2

1.2 End-To-End Memory Networks 5

2 Подход **7**

2.1 Классификация с помощью сети с памятью 7

2.2 Улучшение для классификации изображений 8

2.3 Технические сложности 8

2.4 Обсуждение 9

3 Вычислительные эксперименты **9**

4 Дальнейшие исследования **10**

4.1 Увеличение сети 10

4.2 Разделение опорных объектов 11

4.3 Использование разреженных операций чтения и записи 12

5 Заключение **13**

Список литературы **14**

1 Введение

В последнее время активное развитие получают методы, использующие концепцию внешней памяти, когда по выборке не только учатся параметры модели, но и запоминается сама выборка для дальнейшего использования. В работах [1], [2] предложен один из видов памяти для построения вопросно-ответных систем. При этом самым «классическим» методом, использующим концепцию памяти, является метод k-ближайших соседей. Его минус в том, что он не имеет параметров, которые могут сами настраиваться под выборку, а какие-либо улучшения качества можно получить только подбирая различные метрики и предобработку данных.

В данной работе рассматривается метод на основе модели с памятью, решающий задачу классификации. Он имеет настраиваемые по выборке параметры, похожие на метрики в kNN. При этом получающаяся в предложенном методе «метрика» может иметь произвольную сложность, потому что является на самом деле небольшой нейронной сетью.

В разделах (1.1) и (1.2) описываются архитектуры сетей с памятью, на основе которых строится предложенный в работе метод классификации. В разделе (2.1) описывается базовый вид метода, а в (2.2) его модификация для классификации изображений. В разделе (2.3) рассмотрены технические сложности реализации предложенного метода. В (2.4) проводится параллель между предложенным методом и классическим алгоритмом k-ближайших соседей. В (3) описаны проведенные эксперименты. В разделе (4) предложены возможные дальнейшие исследования.

1.1 Memory Networks

В сети с памятью (memory network) [1] совмещается концепция долговременной памяти (как в методе k ближайших соседей) и обучаемые дискриминативные функции (нейронные сети). В отличии от обычных рекурентных сетей или LSTM [3], в этой модели память является полностью долговременной и хранит всю историю как есть, не сжимая ее в какое-то небольшое внутреннее представление. В оригинальной

статье описывается как общая концепция метода, так и конкретная реализация для вопросно-ответной системы.

Сеть с памятью состоит из памяти m и четырех модулей **I**, **G**, **O** и **R**:

- **I**: (input feature map) – переводит вход сети в некоторое внутреннее представление
- **G**: (generalization) – обновляет старые ячейки памяти согласно новым входным данным
- **O**: (output feature map) – генерирует выход сети по входу и текущему состоянию памяти
- **R**: (response) – переводит выход сети в ответ нужного формата (например, для вопросно-ответных систем ответом является слово или предложение)

Для каждого входа x (в зависимости от задачи входом может быть буква, слово, аудиосигнал, картинка или что-то еще) выполняется ряд действий:

1. x переводится во внутреннее представление $I(x)$
2. Обновляется память: $m_i = G(m_i, I(x), m),\ \forall i$, в простейшем случае старая память не меняется, а новый $I(x)$ записывается в новую ячейку
3. Вычисляется выход сети по входу и состоянию памяти: $o = O(I(x), m)$
4. Выход сети преобразуется в ответ: $r = R(o)$

Обучаемыми элементами в простейшем случае являются **O** и **R**. Модуль **O** строит выход по k опорным ячейкам памяти. Для $k = 1$:

$$o_1 = O_1(x, m) = \underset{i=1,\dots,N}{\operatorname{argmax}} s_O(x, m_i), \tag{1}$$

где s_O некоторая оценивающая близость функция.

При $k = 2$ ищется вторая опорная ячейка памяти:

$$o_2 = O_2(x, m) = \underset{i=1,\dots,N}{\operatorname{argmax}} s_O([x, m_{o_1}], m_i) \tag{2}$$

То есть учитывается предыдущая найденная ячейка o_1. Таким образом можно искать последовательность опорных ячеек для любого k. Выходом модуля **O** является вход x и все опорные ячейки памяти: $[x, m_{o_1}, m_{o_2}]$ (в случае $k = 2$).

В модуле **R** можно выдавать последнюю опорную ячейку или наиболее подходящее слово из словаря, которое можно искать как

$$r = \underset{w \in W}{\operatorname{argmax}} s_R([x, m_{o_1}, m_{o_2}], w) \tag{3}$$

Также в модуле **R** можно использовать небольшую рекурентную сеть, если нужно построить ответ в виде связного предложения.

В качестве функций s_O и s_R в оригинальной статье использовалась

$$s(x, y) = \Phi_x(x)^T U^T U \Phi_y(y), \tag{4}$$

где Φ – отображения оригинального текста в некоторое внутреннее представление (например, мешок слов), а U – обучаемая матрица.

Обучается модель (в случае $k = 2$) с помощью минимизирования по параметрам функций s_O и s_R функционала

$$\begin{aligned} &\sum_{\bar{f} \neq m_{o_1}} max(0, \gamma - s_O(x, m_{o_1}) + s_O(x, \bar{f})) + \\ &\sum_{\bar{f}' \neq m_{o_2}} max(0, \gamma - s_O([x, m_{o_1}], m_{o_2}) + s_O([x, m_{o_1}], \bar{f}')) + \\ &\sum_{\bar{r} \neq r} max(0, \gamma - s_R([x, m_{o_1}, m_{o_2}], r) + s_R([x, m_{o_1}, m_{o_2}], \bar{r})), \end{aligned} \tag{5}$$

где $\bar{f}$, $\bar{f}'$ и $\bar{r}$ – все возможные неправильные выборы опорных ячеек/слова ответа, а γ – некоторый зазор. Этот функционал оптимизируется стохастическим градиентным спуском, на каждом шагу семплируется несколько вариантов f, $\bar{f}$ и $\bar{r}$, а не считаются полные суммы.

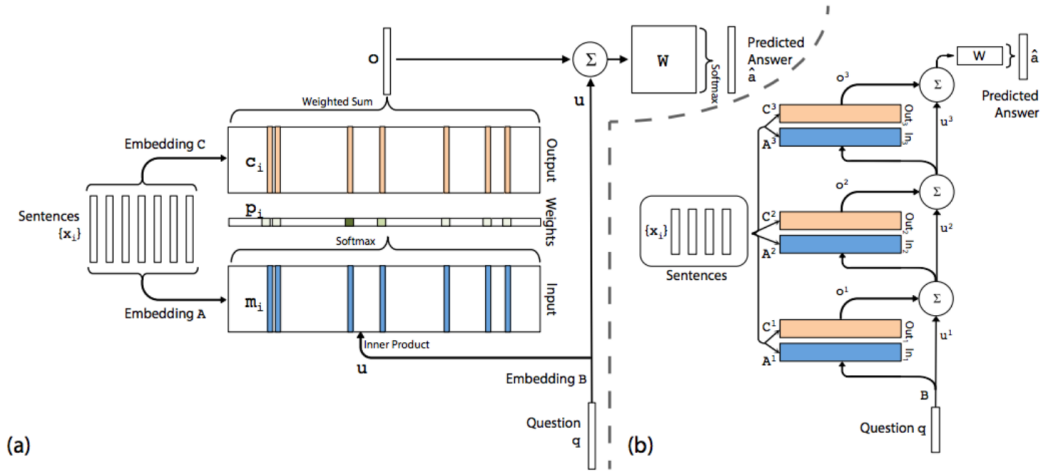

Рис. 1: End-To-End Memory Network

Проблема классической модели Memory Network в том, что обучающая выборка для каждой пары (вопрос, ответ) должна содержать еще набор опорных фактов (так называемый hard supervision). С практической точки зрения это сильно увеличивает затраты на разметку данных.

1.2 End-To-End Memory Networks

Модификация сети с памятью, описанная в [2], не требует различного рода опорных фактов. В оригинальной статье с помощью этой модели тоже строится вопросно-ответная система.

На вход модель принимает конечный набор предложений $x_1, x_2, \ldots, x_n$ и запрос q, а выдает ответ a. Входы и выход сети x_i, q и a состоят из слов некоторого словаря мощности V. Модель записывает все предложения x в память и находит некоторое непрерывное представление для x и q. Затем это представление с помощью нескольких прыжков (hops) по памяти преобразуется в ответ a. Эти прыжки являются непрерывным аналогом поиска опорных ячеек памяти в прошлой модели. Так как все операции в этом преобразовании дифференцируемы, то параметры сети можно обучать с помощью метода обратного распространения ошибки.

Сначала опишем устройство одного слоя сети, который делает один прыжок по памяти. Каждое предложение x_i (оно представлено в виде мешка слов) конвертируется в два вектора m_i и c_i (в простейшем случае, как и описано в оригинальной статье, с помощью умножения на embedding матрицы A и C соответственно). Запрос q тоже конвертируется в некоторое внутреннее представление u (в простейшем случае так же с помощью умножения на embedding матрицу B). Далее считается сходство u с каждой ячейкой памяти m_i:

$$p_i = Softmax(u^T m_i), \tag{6}$$

где $Softmax(z_i) = \exp(z_i)/\sum_j \exp(z_j)$. Таким образом получается вектор вероятностей над памятью. Смысл этого вектора – насколько каждое предложение x_i похоже на запрос q.

Затем по вектору вероятностей p и вектору c строится выходной вектор o:

$$o = \sum_i p_i c_i \tag{7}$$

А затем, в случае однослойной сети, по векторам u и o строится вектор ответа $\hat{a}$:

$$\hat{a} = Softmax(W(o + u)) \tag{8}$$

Матрица W имеет размерность $V \times d$ и $\hat{a}$ является вектором вероятностей над словарем. В таком случае считается, что правильным ответом модели должно быть одно слово.

Один слой сети представлен на рисунке (1)(a). Из таких слоев можно построить многослойную сеть с памятью. Сумма выхода и входа одного слоя $o^k + u^k$ является входом для следующего слоя:

$$u^{k+1} = u^k + o^k, \tag{9}$$

а вектор ответа строится как

$$\hat{a} = Softmax(W(u^{K+1})) = Softmax(W(o^K + u^K)), \tag{10}$$

где K – число слоев в сети.

При этом embedding матрицы A^k и C^k могут быть свои для каждого слоя, или же быть связанными какими-нибудь ограничениями.

2 Подход

В данной работе концепция сети с памятью использовалась для построения классификатора изображений.

2.1 Классификация с помощью сети с памятью

Построим классификатор на основе End-to-End Memory Network. Будем при этом использовать только один слой, а вместо представлений предложений c – ответы на объектах обучающей выборки. Опишем метод более детально.

Рассмотрим одну эпоху обучения сети. Разобъем обучающую выборку на N батчей. Пусть (X_{batch}, y_{batch}) – признаковое описание и ответы объектов одного батча выборки. Обозначим через $(X_{support}, y_{support})$ – признаковое описание и ответы всех остальных объектов из обучающей выборки (назовем их *опорными*). Будем считать, что в матрице $X_{support}$ объекты записаны по строкам. Возьмем $x \in X_{batch}$.

Линейно преобразуем x и $X_{support}$ с помощью умножения на некоторую матрицу A, а потом посчитаем близость преобразованного вектора x к опорным:

$$u = Ax \tag{11}$$

$$M = AX_{support}^T \tag{12}$$

$$p_i = Softmax(u^T M_i) \tag{13}$$

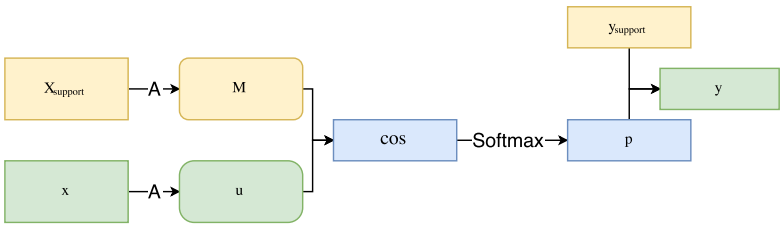

Рис. 2: Архитектура сети

Переведем ответы $y_{support}$ в бинарные вектора $Y_{support}$, где $Y_{i,j} = 1$, если x_i принадлежит j-ому классу, и 0 иначе.

Тогда в качестве класса объекта x возьмем $y = Y_{support}^T \cdot p$. Все операции дифференцируемы, поэтому такую сеть можно обучать методом обратного распространения ошибки.

Архитектура сети представлена на рисунке (2).

2.2 Улучшение для классификации изображений

Как известно, для классификации изображений плохо работают методы, не учитывающие их структуры, а хорошее качество показывают сверточные нейронные сети. Поэтому вместо одного простого линейного преобразования (11), (12) можно вставить небольшую нейронную сеть со сверточными слоями. В экспериментах использовалась сеть вида

$$Conv \to MaxPool \to ReLU \to Linear \tag{14}$$

Такая архитектура приведена на рисунке (3).

2.3 Технические сложности

При использовании сверточных слоев объем памяти, требуемый при обучении сети, становится гораздо больше. Отличие сети с памятью от обычной нейронной

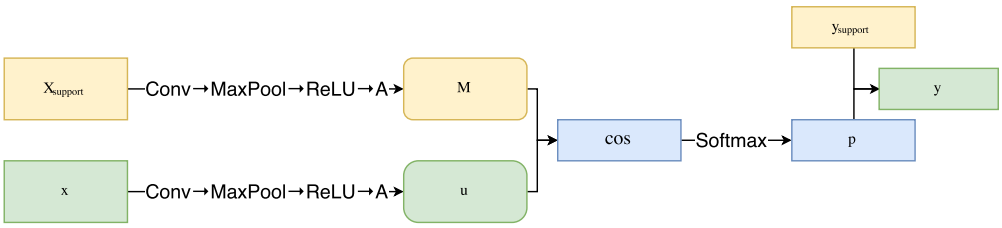

Рис. 3: Добавление сверточного слоя

сети в том, что на каждом батче нужно посчитать расстояния до большого числа объектов выборки. Соответственно нужно пропустить все объекты через сверточные слои и хранить в памяти, а не только маленький батч. Можно предложить несколько способов борьбы с этом проблемой.

Например, при настройке на каждый батч в качестве опорных объектов можно брать не всю оставшуюся выборку, а только случайные M объектов из нее. Такой способ исследован в ходе экспериментов. Другие способы описаны в разделе (4).

2.4 Обсуждение

На самом деле, предложенный метод является чем-то вроде дифференцируемого метода k-ближайших соседей. Он может выучивать нужную метрику по выборке, ограничения только структурные – наличие и количество линейных преобразований и сверточных слоев.

3 Вычислительные эксперименты

Для тестирования алгоритма были использованы классические датасеты – MNIST [4], CIFAR-10 и CIFAR-100 [5]. В экспериментах со сверточным слоем использовалось $M = 10^4$ случайных объектов в качестве опорных. Полученные результаты:

Результат kNN на датасете MNIST взят из статьи [4] для метода k-ближайших соседей с tangent distance, на датасете CIFAR-10 из статьи [6] при уменьшении раз-

Метод	MNIST	CIFAR-10	CIFAR-100
kNN	98.90%	41.78%	7.6%
MeMN2N Linear	97.76%	38.55%	19.85%
MeMN2N 1Conv	98.87%	62.69%	31.70%
Лучший известный	99.79%	96.53%	75.72%

Таблица 1: Точность (accuracy) методов на исследуемых наборах данных. kNN – метод k-ближайших соседей, MeMN2N Linear – предложенный метод с сетью, состоящей из одного линейного слоя, MeMN2N 1Conv – с сетью архитектуры $Conv \to MaxPool \to ReLU \to Linear$

мерности до 30 первых компонент с помощью PCA, на датасете CIFAR-100 получен самостоятельно. Представленные результаты на kNN используют только оригинальные признаки (пиксели изображений), на датасетах CIFAR-10 и CIFAR-100 используется евклидова метрика.

Видно, что предложенный метод показывает качество сравнимое или даже лучше, чем метод k-ближайших соседей, то есть сам выучивает хорошие метрики между объектами. При этом качество хуже, чем лучшее известное на данный момент (достигаемое с помощью глубоких сверточных нейронных сетей). Но целью работы было не улучшить известное качество нейросетей, а показать возможность обучения нелинейных метрик на основе модели с памятью.

На рисунке (4) приведена зависимость качества предложенного метода от эпохи обучения. Видно, что качество достаточно стабильно растет и выходит на плато.

4 Дальнейшие исследования

4.1 Увеличение сети

Понятно, что одного сверточного слоя мало для извлечения всей информации из изображений. Например, даже в самых старых реализациях сверточных сетей (LeNet-5) [4] использовалось три сверточных слоя. Так что увеличение сети, счи-

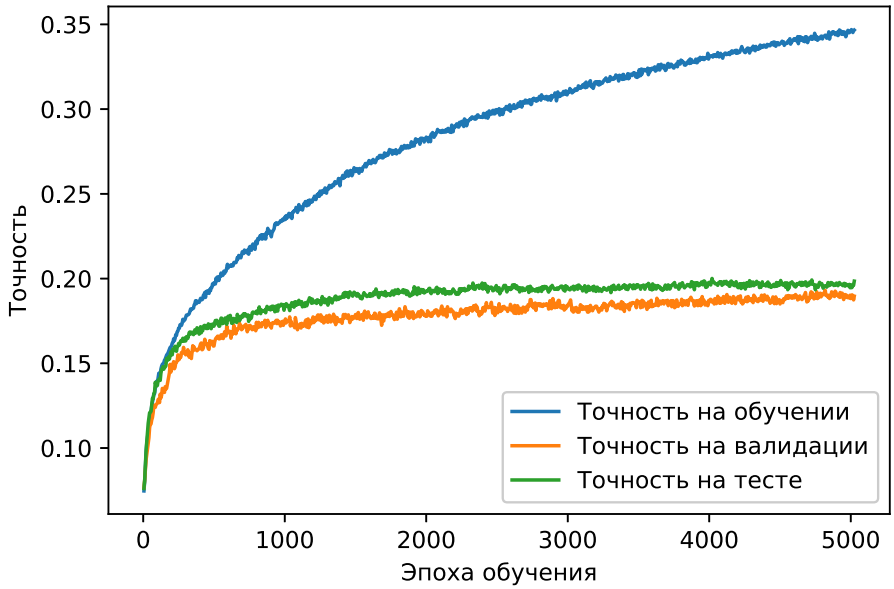

Рис. 4: Зависимость качество алгоритма на CIFAR-100 от эпохи на обучающей, валидационной и тестовой выборке сети с памятью с одним линейным слоем

тающей внутреннее представление данных, может значительно улучшить качество классификации.

Но при этом и время обучения сети, и объем потребляемой памяти сильно увеличится.

4.2 Разделение опорных объектов

В данной работе при использовании сверточного слоя для уменьшения потребления памяти считалось не честное расстояние до всех опорных объектов, а только для небольшой (25%) части из них. Понятно, что при использовании всех объектов качество должно быть лучше.

Можно предложить способ честного подсчета расстояний до всех объектов, при котором не надо хранить все в памяти. Разделим опорные объекты на несколько частей и будем считать близости раздельно по каждой части. Если делать это прямо

так, то в softmax'е (13) будет неточная нормировка, за счет того, что мы не видим все объекты. Эту проблему можно решить, если предварительно посчитать

$$max = \max_j z_j \tag{15}$$

$$Z = \sum_j \exp(z_j - max) \tag{16}$$

Это можно сделать, разбивая опорные объекты на части (подсчет максимума нужен для устойчивой нормировки в дальнейшем – чтобы Z не получалось слишком близким к нулю). Тогда уже для обучения параметров сети можно разбить опорные объекты на несколько частей, но уже считать не softmax на каждой, а

$$exp(z_i - max)/Z \tag{17}$$

Таким образом будет получаться честная несмещенная оценка.

4.3 Использование разреженных операций чтения и записи

Описанный выше способ честного подсчета расстояний до всех объектов решает проблему потребления памяти, но не времени обучения. В работе [7] представлена память с разреженным доступом (Sparse Access Memory), которая одновременно уменьшает объем потребляемой оперативной памяти и время обучения. Основная идея работы заключается в том, что при каждом доступе к памяти считается распределение вероятности (13) не на всей памяти, а только на K ближайших ячейках. При этом близкие ячейки ищутся с помощью приближенного метода k-ближайших соседей.

5 Заключение

В данной работе был предложен метод классификации, основанный на модели с памятью. При этом все операции в методе дифференцируемые и его параметры могут обучаться с помощью метода обратного распространения ошибки.

Предложена модификация метода, улучшающая его качество для задачи классификации изображений. Проведены эксперименты, показывающие, что классические методы машинного обучения, основанные на подсчете близости между объектами, работают немного хуже.

В конце работы предложены идеи для дальнейшего исследования, которые могут улучшить как качество работы метода, так и уменьшить время обучения.

Список литературы

[1] *Jason Weston Sumit Chopra Antoine Bordes*. Memory Networks. — 2014.

[2] *Sainbayar Sukhbaatar Arthur Szlam Jason Weston Rob Fergus*. End-To-End Memory Networks. — 2015.

[3] *Sepp Hochreiter Jurgen Schmidhuber*. Long short-term memory. — 1997.

[4] Gradient-Based Learning Applied to Document Recognition / Y. LeCun, L. Bottou, Y. Bengio, P. Haffner // *Proceedings of the IEEE*. — 1998. — November. — Vol. 86, no. 11. — Pp. 2278–2324.

[5] *Krizhevsky Alex*. Learning Multiple Layers of Features from Tiny Images. — 2009.

[6] *Yehya Abouelnaga Ola S. Ali Hager Rady Mohamed Moustafa*. CIFAR-10: KNN-based Ensemble of Classifiers. — 2016.

[7] *Jack W Rae Jonathan J Hunt Tim Harley Ivo Danihelka Andrew Senior Greg Wayne Alex Graves Timothy P Lillicrap*. Scaling Memory-Augmented Neural Networks with Sparse Reads and Writes. — 2016.