Московский государственный университет им. М. В. Ломоносова
Факультет вычислительной математики и кибернетики

Отчет

По заданию №5
по практикуму на ЭВМ

Выполнил студент 317 группы
Измаилов Павел Алексеевич

Москва, 24 февраля 2016

Содержание

1 Описание проделанной работы

В данном отчете содержатся результаты экспериментов, проведенных мной в соответствии с пятым заданием по курсу практикума на ЭВМ на кафедре ММП ВМК МГУ. Все необходимые эксперименты были проведены, и по ним были получены соответствующие выводы. Мной были выполнены все части задания, кроме одного бонусного задания, заключающегося в реализации части функций на GPU.

2 Теория и вывод используемых формул

В данном разделе приводится теория, относящаяся к заданию, и вывод используемых формул.

2.1 Нейронные сети

Нейронные сети — класс моделей, используемых в машинном обучении в задачах обучения с учителем. Многослойные нейронные сети позволяют автоматически извлекать из данных иерархию признаков и использовать эти признаки для решения тех или иных задач. Нейронная сеть состоит из соединенных между собой некоторым образом нейронов — специальных вычислительных элементов. Эти элементы имеют некоторое число входов и один выход. Нейроны складывают значения со всех своих входов, применяют к ним некоторое нелинейное преобразование f, и полученное число подают на выход. Мы будем рассматривать так называемые feed-forward нейронные сети с полносвязными слоями. В этих сетях нейроны объединены в последовательность из L слоев, причем выход каждого нейрона слоя i подается на вход всем нейронам слоя $i+1$ с некоторым весом w. Вес w свой для каждой пары нейронов. Кроме того в каждом слое, кроме последнего, добавляется один нейрон, не имеющий входов, и подающий на вход всем нейронам следующего слоя значения b, свои для каждого нейрона. Этот дополнительный нейрон будем называть константным.

Первый и последний слои нейронной сети называются видимыми, а остальные — скрытыми. На вход первому слою нейросети подается вектор x с числом компонент, равным числу нейронов в первом слое. На первом слое нелинейное преобразование f не применяется, то есть на выход нейроны первого слоя подают просто соответствующие компоненты вектора x. Значения, поданные на выход на последнем слое являются "ответом" нейронной сети.

Действие нейронной сети можно описать следующим образом. На вход ей поступает вектор x. Матрицу весов для нейронов слоев l и $l+1$ будем обозначать через W, то есть W_{ij} — вес, соответствующий i-му нейрону слоя l и j-му нейрону слоя $l+1$. Выход нейрона i слоя l будем обозначать через a_i^l, а вход — через z_i^l. Эти велечины объедим в вектора a^l и z^l соответственно. Ответ нейронной сети на объекте (векторе) x будем обозначать через $\mathrm{net}(x)$.Описанные выше величины описываются следующей системой уравнений.

$$a^1 = x$$
$$z^l = W^l a^{l-1} + b^l, \quad l = 1, \ldots, L$$
$$a^l = f(z^l), \quad l = 1, \ldots, L.$$

Обучением нейронной сети называется процесс подбора весов w и b таких, что ответ нейронной сети на данных из обучающей выборки близок к известным правильным ответам. Это достигается путем минимазации некоторого функционала качества.

Мы будем рассматривать частный случай нейронной сети — автокодировщик.

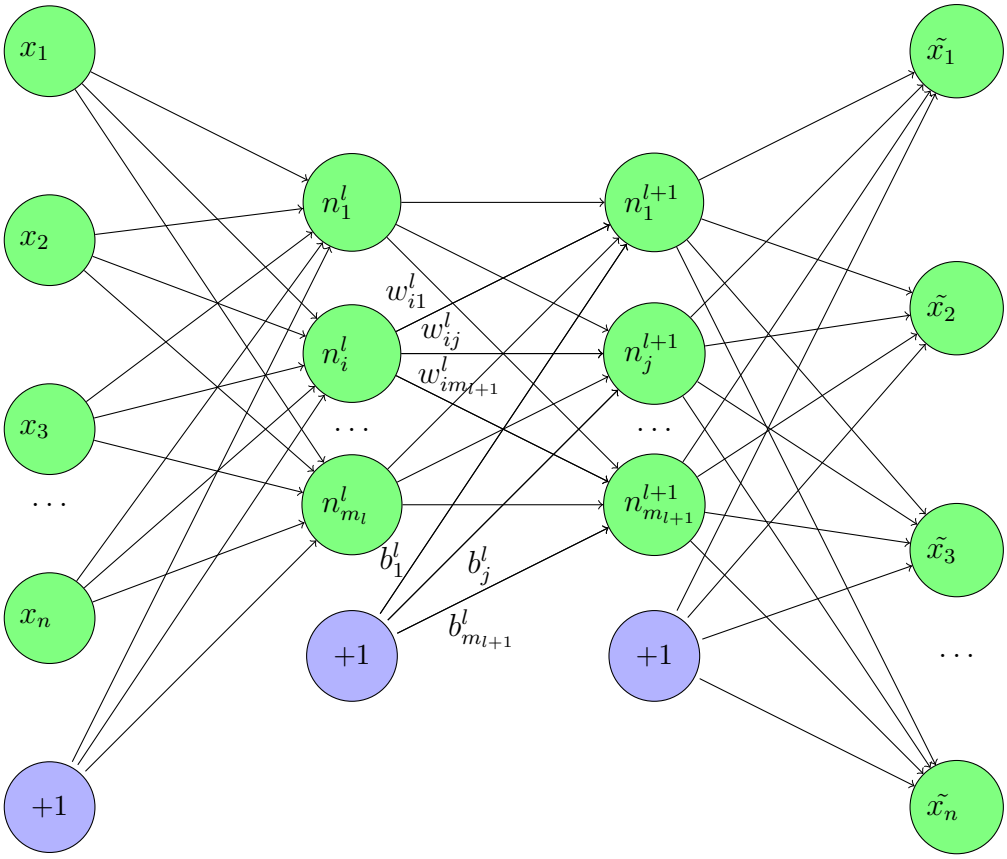


Рис. 1: Схема автокодировщика

2.2 Автокодировщики

Автокодировщики — специальный вид нейронных сетей, которые пытаются на выходе получить то же, что им подано на вход. Правильным ответом на объекте x для такой нейронной сети является сам объект x. Будем обозначать число нейронов (не считая константного) в слое l через m_l. В автокодировщиках выполняется соотношение $m_l = m_{L-l+1}$, то есть они "симметричны". При этом обычно число скрытых слоев

в автокодировщике нечетно, и число нейронов в среднем слое меньше числа нейронов в видимых слоях. Таким образом, автокодировщик позволяет закодировать поданный ему на вход объект x. Схема работы автокодировщика показана на рис. 1.

Применение автокодировщика полезно, например, если имеется большая неразмеченная выборка объектов и небольшая размеченная, причем требуется решить задачу классификации или регрессии. Обучив автокодировщик на неразмеченной выборке, можно получить хорошее сжатое представление объектов (код, получаемый на среднем слое), после чего преобразовав соответствующим образом размеченную выборку обучить на полученных признаках какой-нибудь классификатор.

В данной работе осуществлялась классификация изображений. Автокодировщик обучался на неразмеченной выборке из патчей (небольших кусочков изображений), случайным образом вырезанных из изображений. После этого все объекты размеченной выборки разделялись на такие патчи, которые преобразовывались с помощью автокодировщика в сжатое представление. Коды для всех патчей конкатенировались в один вектор признаков. На полученных признаках обучались логистическая регрессия и случайный лес.

2.3 Функционал качества

Пусть имеется выборка из N объектов $x \in \mathbb{R}^D$. Объединим эти объекты в матрицу $X \in \mathbb{R}^{N \times D}$. В качестве функционала качества используется регуляризованная среднеквадратическая ошибка:

$$J(W, b) = \frac{1}{2N} \sum_{i=1}^{N} ||X_i - \text{net}(X_i)||^2 + R(W, b),$$

где через X_i обозначен i-й объект выборки, а слагаемое $R(W, b)$ отвечает за регуляризацию и будет описано ниже.

На коэффициенты автокодировщика накладываются два ограничения — во-первых они должны быть не очень большими, а во-вторых нейроны должны активироваться редко. Поэтому регуляризатор $R(W, b)$ состоит из двух слагаемых:

$$R(W, b) = \frac{\lambda}{2} \sum_{l=1}^{L-1} ||W^l||_2^2 + \beta \sum_{j=1}^{H} \left(\rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \right),$$

где λ — коэффициент регуляризации, $||W^l||_2$ — норма Фробениуса матрицы W^l, ρ — коэффициент разреженности, β — регуляризатор разреженности, H — общее число нейронов во всех скрытых слоях, $\hat{\rho}_j$ — среднее значение активации нейрона j по всем объектам обучающей выборки X, то есть

$$\hat{\rho}_j = \frac{1}{N} \sum a_j(x_i),$$

где под a_j понимается значение активации нейрона j на i-м объекте. Суммирование в формуле для $R(W, b)$ ведется по всем нейронам в скрытых слоях.

2.4 Метод обратного распространения ошибки и подсчет градиента

В данном разделе описывается метод вычисления градиента целевой функции $J(W, b)$.

В данном разделе мы несколько изменим обозначения. Напомним, что $X \in \mathbb{R}^{N,D}$ — матрица объекты-признаки. Для всех величин z^l и a^l сконкатенируем их значения для всех объектов из обучающей выборки и теперь будем через z^l и a^l обозначать полученные матрицы: $a^l, z^l \in \mathbb{R}^{m_l \times D}$.

Представим функционал качества $J(W, b)$ в виде

$$J(W, b) = \tilde{J}(W, b) + \frac{\lambda}{2} \sum_{l=1}^{L-1} ||W^l||_2^2,$$

где

$$\tilde{J}(W, b) = \frac{1}{2N} \sum_{i=1}^{N} ||X_i - \text{net}(X_i)||^2 + \beta \sum_{j=1}^{H} \left(\rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \right).$$

Градиент второго слагаемого в данном представлении по параметрам W^i, b^i вычисляется легко.

$$\frac{\partial}{\partial W^i} \left(\frac{\lambda}{2} \sum_{l=1}^{L-1} ||W^l||_2^2 \right) = \lambda W^i,$$

$$\frac{\partial}{\partial b^i} \left(\frac{\lambda}{2} \sum_{l=1}^{L-1} ||W^l||_2^2 \right) = 0.$$

Для вычисления градиента $\tilde{J}(W, b)$ применяется так называемый метод обратного распространения ошибки. Рассмотрим граф вычисления функции $J(W, b)$ по параметрам W^i и b^i. Этот граф приведен на рис. 2.

Подсчитаем для всех $l = 1, \ldots, L$ производную

$$\delta^l = \frac{\partial \tilde{J}(W, b)}{\partial z^l}.$$

Из графа вычислений легко видеть, что

$$\delta^L = \frac{\partial \tilde{J}(W, b)}{\partial z^L} = \frac{\partial \tilde{J}(W, b)}{\partial a^L} \frac{\partial a^L}{\partial z^L} = \frac{\partial \tilde{J}(W, b)}{\partial a^l} * f'(z^l) = \frac{a^L - X^T}{N} * f'(z^L).$$

Здесь $A * B$ — поэлементное умножение матриц.

Поясним последнее равенство. Дифференциал отображения $a^l(z^l)$ — отображение пространства матриц $\mathbb{R}^{m_l \times N}$ в себя. По правилам дифференцирования сложной функции производная $\frac{\partial \tilde{J}(W,b)}{\partial z^l}$ может быть получена как образ матрицы $\frac{\partial \tilde{J}(W,b)}{\partial a^l}$ при этом

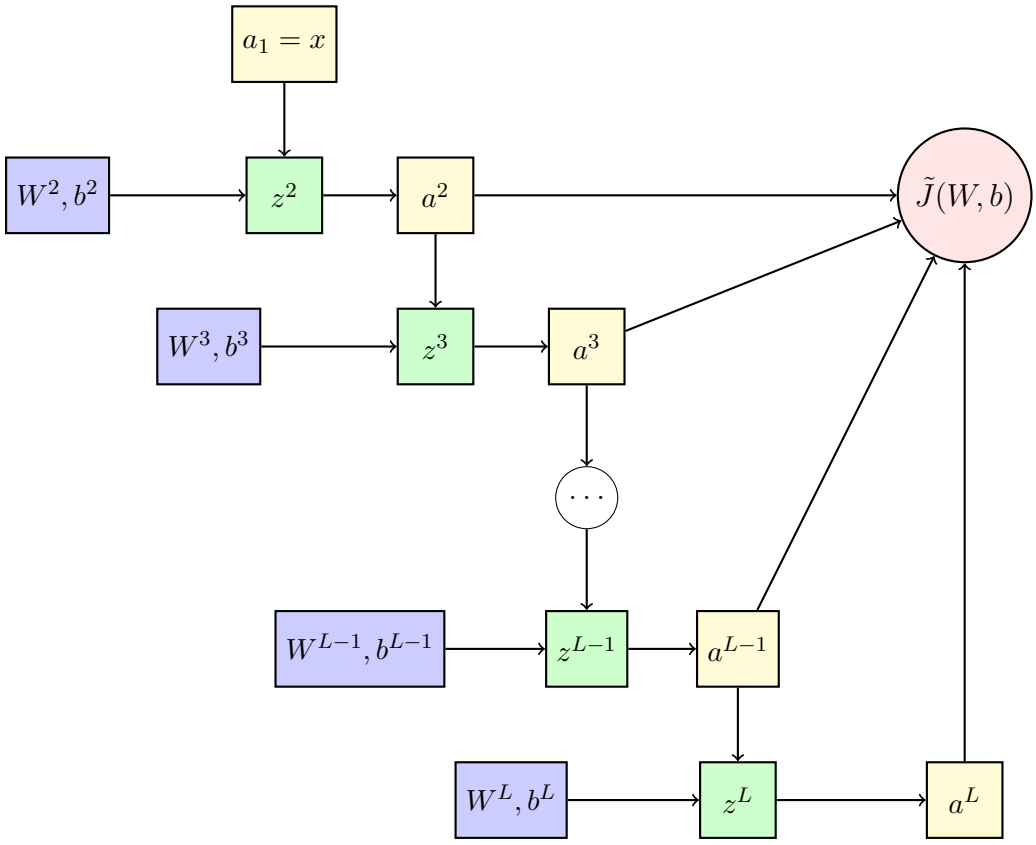


Рис. 2: Граф вычисления функционала качества

отображении. Рассмотрим теперь это отображение более подробно.

$$f(z^l+\Delta)-f(z^l) = \begin{pmatrix} f(z^l_{11}+\Delta_{11})-f(z^l_{11}) & \dots & f(z^l_{1N}+\Delta_{1N})-f(z^l_{1N}) \\ \dots & \dots & \dots \\ f(z^l_{m_l1}+\Delta_{m_l1})-f(z^l_{m_l1}) & \dots & f(z^l_{m_lN}+\Delta_{m_lN})-f(z^l_{m_lNpdflatex}) \end{pmatrix} =$$

$$= \begin{pmatrix} f'(z^l_{11})\Delta_{11}+o(\Delta_{11}) & \dots & f'(z^l_{1N})\Delta_{1N})+o(\Delta_{1N}) \\ \dots & \dots & \dots \\ f'(z^l_{m_l1})\Delta_{m_l1}+o(\Delta_{m_l1}) & \dots & f'(z^l_{m_lN})\Delta_{m_lN}+o(\Delta_{m_lN}) \end{pmatrix} = f'(z) * \Delta + o(\Delta)$$

Рассмотрим теперь произвольный слой $l \in \{2, \dots, L-1\}$. Из графа вычислений видно, что $\tilde{J}$ зависит от z^l только через a^l. В свою очередь от a^l $\tilde{J}$ зависит через z^{l+1} и напрямую. Поэтому можем записать

$$\delta^l = \frac{\partial \tilde{J}(W,b)}{\partial z^l} = \frac{\partial \tilde{J}(W,b)}{\partial a^l}\frac{\partial a^l}{\partial z^l} = \frac{\partial \tilde{J}(W,b)}{\partial a^l} * f'(z^l).$$

Вычислим $\frac{\partial \tilde{J}(W,b)}{\partial a^l}$.

$$\frac{\partial \tilde{J}(W,b)}{\partial a^l} = \frac{\partial \tilde{J}(W,b)}{\partial z^{l+1}}\frac{\partial z^{l+1}}{\partial a^l} + \frac{\partial}{\partial a^l}\left(\beta \sum_{j=1}^{H}\left(\rho \log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j}\right)\right) =$$

$$= (W^{l+1})^T \delta^{l+1} + \beta \sum_{j=1}^{H} \left(\left(-\frac{\rho}{\hat{\rho}_j} + \frac{1-\rho}{1-\hat{\rho}_j} \right) \frac{\partial \hat{\rho}_j}{\partial a^l} \right) =$$

$$= (W^{l+1})^T \delta^{l+1} \oplus \frac{\beta}{N} \left(-\frac{\rho}{\hat{\rho}^l} + \frac{1-\rho}{1-\hat{\rho}^l} \right),$$

где $A \oplus b$ означает матрицу, получаемую прибавлением вектора b ко всем строкам матрицы A, а $\hat{\rho}^l$ — вектор из всех $\hat{\rho}_j$ соответствующих нейронам слоя l. Доказательство последнего перехода аналогично рассуждениям, проведенным при вычислении δ^L.

Окончательно получаем

$$\delta^l = \left((W^{l+1})^T \delta^{l+1} \oplus \frac{\beta}{N} \left(-\frac{\rho}{\hat{\rho}^l} + \frac{1-\rho}{1-\hat{\rho}^l} \right) \right) * f'(z).$$

Перейдем к вычислению производных $\tilde{J}$ по параметрам сети W и b. Из графа вычислений видно, что $\tilde{J}$ зависит от W^l, b^l только через z^l. Таким образом имеем

$$\frac{\partial \tilde{J}(W,b)}{\partial W^l} = \frac{\partial \tilde{J}(W,b)}{\partial z^l} \frac{\partial z^l}{\partial W^l} = a^l \delta^l,$$

$$\frac{\partial \tilde{J}(W,b)}{\partial b^l} = \frac{\partial \tilde{J}(W,b)}{\partial z^l} \frac{\partial z^l}{\partial b^l} = \delta^l.$$

3 Эксперименты

В данном разделе приведены результаты экспериментов по данному заданию.

3.1 Визуализация скрытого слоя автокодировщика

В первом эксперименте производилась визуализация скрытого слоя однослойного автокодировщика. Каждому нейрону скрытого слоя можно поставить в соответствие изображение следующим образом. В первом слое каждый нейрон соответствует одному каналу одного пикселя поданного на вход изображения. Для нейрона скрытого слоя построим новое изображение, значение интенсивности i-го канала j-го пикселя которого равно весу ребра, соединяющего данный нейрон скрытого слоя с соответствующим нейроном первого слоя.

Полученные таким образом изображения являются по сути фильтрами, которые применяет автокодировщик к патчам, чтобы преобразовать их в числа. Фильтры, полученные в ходе экспериментов приведены на рисунке 3.

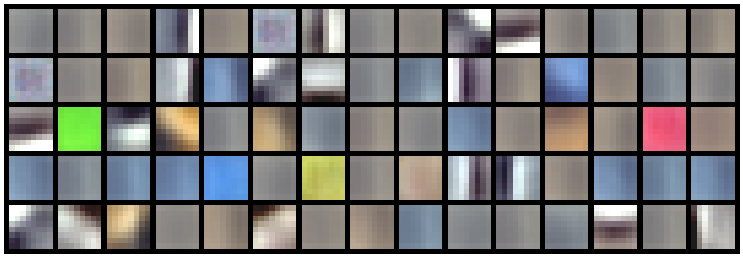
Из изображений фильтров видно, что при значениях параметров $\beta = 3, \lambda = 10^{-4}, \rho = 10^{-2}$ на фильтрах можно заметить полосы под разным наклоном и цветовые переходы. При других значениях параметров фильтры не являются интерпретируемыми. Если требовать от сети слишком большой разреженности или сделать коэффициент регуляризации слишком большим, то фильтры становятся серыми и сильно теряют в информативности. При отсутствии регуляризации фильтры перестают быть интерпретируемыми, но все равно могут давать неплохие признаки.

3.2 Обучение классификаторов на данных сокращенной размерности

Автокодировщик позволяет преобразовать патчи в некоторое сжатое представление. Для этого каждому патчу сопоставляется вектор значений всех нейронов среднего слоя. Представив данные в таком виде, можно обучить на полученных признаках тот или иной классификатор. В данном задании мною использовались логистическая регрессия и случайный лес. С помощью этих методов производилась классификация изображений на 10 классов.

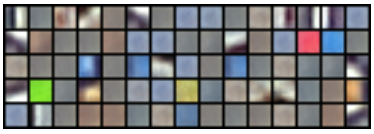
Чтобы получить сжатое представление изображения, нужно разбить его на патчи. Это можно сделать разными способами. В данном задании предлагалось разбивать изображения на патчи с равномерным шагом. Требовалось исследовать зависимость качества классификации от размера шага.

При шаге 8 патчи покрывают все пиксели изображения без перекрытий. При шаге

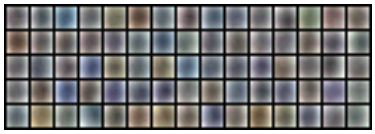


(a) $\beta = 3, \lambda = 10^{-4}, \rho = 10^{-2}$

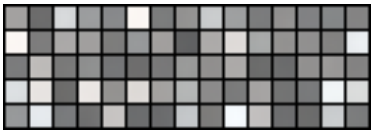
(b) $\beta = 0, \lambda = 10^{-4}$



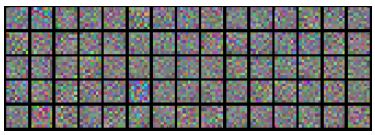
(c) $\beta = 100, \lambda = 10^{-4}, \rho = 10^{-2}$



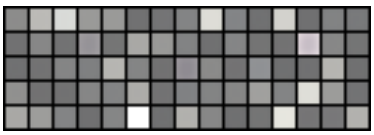
(d) $\beta = 3, \lambda = 10^{-4}, \rho = 10^{-4}$



(e) $\beta = 3, \lambda = 10^{-4}, \rho = 1$



(f) $\beta = 3, \lambda = 0, \rho = 10^{-2}$



(g) $\beta = 3, \lambda = 10^{-2}, \rho = 10^{-2}$

Рис. 3: Фильтры однослойного автокодировщика с 75 скрытыми нейронами для различных значений параметров регуляризации

4 патчи перекрываются. При шагах 16 и 32 патчи не покрывают всего изображения.

Результаты работы методов приводятся в таблице 1. Из таблицы видно, что логистическая регрессия не справляется с большим числом информации, которая может быть кореллированной, что плохо для линейных методов, при размере шага 4. Случайный лес показывает лучший результат при наименьшем шаге — при 4.

Для сравнения те же классификаторы были обучены на значениях интенсивности каждого канала в пикселях исходного изображения. При этом были полученны следующие результаты. Логистическая регрессия: доля правильных ответов 0.32, и случайный лес: доля правильных ответов 0.36.

Шаг	Доля правильных ответов			
	На признаках однослойного автокодировщика		На признаках трехслойного автокодировщика	
	Логистическая регрессия	Случайный Лес	Логистическая регрессия	Случайный Лес
4	0.30	0.38	0.38	0.39
8	0.32	0.38	0.37	0.39
16	0.32	0.36	0.34	0.36
32	0.26	0.32	0.29	0.32

Таблица 1: Результаты работы методов для разных длин шага при разбиении изображения на патчи

3.3 Использование трехслойного автокодировщика для генерации признаков

В данном разделе проводится исследование зависимости качества классификации от струкутуры автокодировщика. А именно, для генерации признаков используется трехслойный автокодировщик со скрытыми слоями размеров 75, 50 и 75. При этом не использовалась регуляризация по разреженности, так как иначе с большой вероятностью до внутреннего слоя не доходит почти никакой информации.

Как и в прошлом пункте, признаки, полученные на среднем слое классификатора использовались для обучения логистической регрессии и случайного леса. Результаты работы методов приведены в таблице 1. Из таблицы видно, что за счет большего, по сравнению с однослойным автокодировщиком, уменьшения размерности, логистическая регрессия смогла показать на шаге 4 результат близкий к лучшим результатам случайного леса. Вообще, результаты работы обоих методов для каждого шага не ухудшились при переходе от однослойного автокодировщика к трехслойному, а для логистической регрессии заметно улучшились.

3.4 Зависимость качества классификации от размера неразмеченной выборки

В данном разделе исследуется зависимость качества итоговой классификации, от размеров неразмеченной выборки патчей, используемой при обучении автокодировщика.

Был проведен следующий эксперимент. Однослойный автокодировщик был обучен на выборках размеров 10^i для $i = 2, 3, 4, 5$. После этого на основе признаков, генерируемых автокодировщиком, обучались логистическая регрессия и случайный лес. Все методы обучались на патчах, сгенерированных с длиной шага 8.

Результаты экспериментов приведены в таблице 2. Из таблицы видно, что качество классификации почти не зависит от размеров выборки, что свидетельствует о том, что даже выборка из 10^2 патчей репрезентативна.

Число патчей при обучении автокодировщика	Доля правильных ответов	
	Логистическая регрессия	Случайный Лес
10^2	0.37	0.39
10^3	0.38	0.39
10^4	0.38	0.39
10^5	0.38	0.39

Таблица 2: Результаты работы методов для разных длин шага при разбиении изображения на патчи

3.5 Использование RLU в качестве функции активации

В данном разделе исследуется влияние использования функции активации RLU(Rectified Linear Unit) вместо сигмоидной функции активации на качество классификации и скорость решения задачи оптимизации.

Функция активации RLU имеет следующую простую формулу.

$$\mathrm{RLU}(x) = \max\{0, x\}$$

График этой функции приведен на рис. 4

Использование этой функции активации позволяет автоматически получить разреженный автокодировщик без дополнительных усилий. Кроме того, эта функция активации порождает простую (кусочно-квадратичную) зависимость между функционалом качества и параметрами сети. Это делает задачу обучение автокодировщика более простой. Наконец, вычисление функционала качества и его градиента становится более простым при использовании RLU вместо сигмоиды.

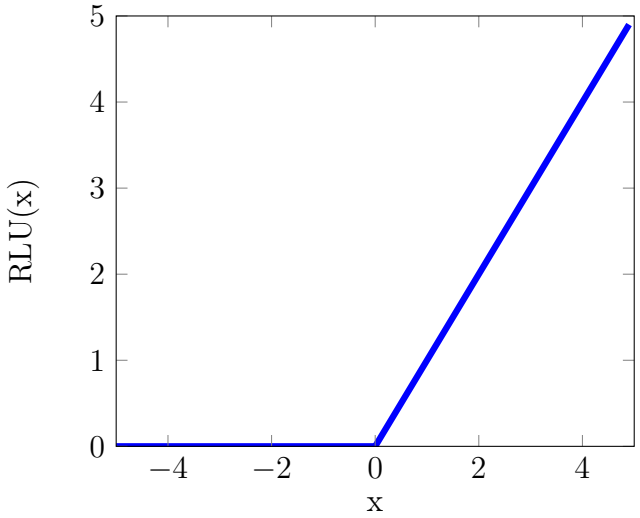


Рис. 4: График RLU(x)

В экспериментах к данному заданию был обучен автокодировщик с функцией активации RLU, после чего на признаках, определяемых автокодировщиком, были обучены логистическая регрессия и случайный лес. Генерация патчей по изображениям производилась с шагом 8. В результате, качество классификации оказалось ниже (доля правильных ответов 0.29 для логистической регрессии и 0.37 для случайного леса), чем в аналогичных экспериментах с сигмоидной функцией активации. Тем не менее, возможно результаты оказались бы другими при более правильном подборе параметров автокодировщика.

Задача обучения автокодировщика с функцией RLU действительно была решена быстрее, чем задача обучения автокодировщика с сигмоидной функцией активации. Для проверки этого факта была запущена оптимизация по параметрам нейросетей из одной и той же случайно сгенерированной начальной точки. При этом в задаче с функцией активации RLU метод сошелся за 4400 итераций, а в задаче с сигмоидной функцией активации после 5000 итераций метод так и не успел сойтись.

4 Выводы

В данной работе мною был реализован разреженный нейросетевой автокодировщик, алгоритм обратного распространения ошибки для подсчета градиента соответствующего функционала качества, функции для визуализации скрытых слоев автокодировщика. Был проведен ряд экспериментов, связанных с использованием кодов, генерируемых автокодировщиком, в качестве признакового описания для изображений. В частности были исследованы влияние размера обучающей выборки для автокодировщика, используемой функции активации, структуры автокодировщика и используемого алгоритма классификации на итоговое качество классификации.

По результатам экспериментов, проведенных в данной работе, можно сделать ряд выводов.

Во-первых, размер неразмеченной выборки, используемой для обучения автокодировшика, можно выбрать не слишком большим. В проведенных экспериментах, результаты для автокодировщика, обученного на 100 патчах, фактически не отличались от результатов для автокодировщика, обученного на 10^5 патчах. Это можно объяснить тем, что выборка из 100 патчей 8×8 с большой вероятностью является репрезентативной.

Во-вторых, при хорошей структуре автокодировщика качество классификации, получаемое для логистической регрессии почти не отличается от качетсва классификации, получаемого для случайного леса.

В-третьих, использование функции активации RLU позволяет упростить задачу обучения автокодировщика, хотя получить на обученном таким образом автокодировщике хорошее качество классификации в поставленных экспериментах не удалось.

В-четвертых, использование многослойного автокодировщика при правильном подборе параметров может позволить улучшить результаты классификации по сравнению с однослойным.