Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего профессионального образования
**«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»**
**Институт математики и компьютерных наук
Кафедра алгебры и дискретной математики**

О фильтрации событий в физике высоких энергий с помощью нейросетей с инверсией градиента.

Квалификационная работа
на степень бакалавра наук
по направлению
«Математика, прикладная математика»
студента группы МТ-401
Киселева Антона Ивановича

Научный руководитель
Клепинин Александр Владимирович,
кандидат физико-математических наук,
доцент кафедры алгебры и
дискретной математики ИМКН УрФУ

Екатеринбург
2016

Реферат

Глубокие нейросетевые архитектуры показывают превосходные результаты при наличии большого количества размеченных данных. В случае их отсутствия доменная адаптация помогает при наличии большого количества данных похожей природы, но имеющих смещенное распределение. Одной из таких задач является фильтрация событий, полученных детекторами Большого Адронного коллайдера. Для решения подобных задач была предложена новая архитектура для проведения доменной адаптации. Ее суть заключается в создании нейросети с двумя выходами — один из них определяет класс, а другой домен. Классифицирующие домен слои подключены к остальной сети при помощи слоя обращения градиента. При проведении процедуры обучения при помощи метода обратного распространения ошибки этот слой негативно влияет на определение домена, таким образом нейросеть выделяет неспецифические для конкретных доменов признаки.

Целью этой работы является исследование применимости архитектуры нейросети со слоем обращения градиента в задаче детектирования распада частиц $\tau^- \to \mu^+\mu^-\mu^-$.

В процессе работы была построена нейросеть, которая успешно обучается и выделяет признаки, позволяющие выделить сигнал, но не позволяющие по ним отделить реальные данные от синтетических.

Ключевые слова: доменная адаптация, глубокие нейронные сети.

Объект исследования — доменная адаптация глубоких нейронных сетей.

Цель работы — рассмотреть применение доменной адаптации нейронных сетей в задаче фильтрации событий, разработать методику применения таких сетей для создания неразличающего домены нейросетевого классификатора.

Результаты работы: проведены эксперименты по применению данного вида нейросетей; изучено влияние изменения параметров классификатора на получаемый результат; сделан вывод насчет практической применимости доменной адаптации в данной задаче.

Содержание

Введение

В современном мире каждый день создается невероятное количество информации. С одной стороны, люди общаются друг с другом через социальные сети и создают таким образом миллионы текстов. С другой стороны, есть миллионы устройств, которые собирают данные — термометры, приборы учета, видеокамеры и прочее. Анализ этих данных является крайне популярной на сегодняшний день темой. Она актуальна как в исследовательской среде, так и среди бизнеса.

Анализ данных требует использования различных методов машинного обучения, таких как машины опорных векторов или решающих деревьев. Для большинства из этих методов наличие *большого объема* данных позволяет строить более точные модели.

Одной из важных задач машинного обучения является задача классификации. К сожалению, для многих задач, сводящихся к задаче классификации невозможно получить достаточно большой объем данных, для получения хорошей модели. Это может быть связано как с тем, что для получения большого количества данных требуется затратить множество ресурсов, так и тем, что генерирующие их процессы случаются довольно редко. Помимо этого, может быть такая ситуация, что достоверность данных сомнительна.

Одной из таких задач является поиск лептонных распадов, не сохраняющих лептонный аромат. В Стандартной модели — общепринятой на данный момент модели физики элементарных частиц — считается, что аромат лептонов является физической характеристикой, которая сохраняется при распадах. В некоторых моделях физики элементарных частиц ароматы лептонов таким свойством не обладают. Успешное нахождение такого рода распадов откроет новые горизонты для физики элементарных частиц. Одним из таких распадов является распад $\tau^- \to \mu^+\mu^-\mu^-$, который рассмотрен в работе. Формальное определение задачи дано в параграфе 3.

Одной из наиболее точных моделей, использующейся в задаче классификации, является глубокая нейронная сеть. Мы описываем ее устройство в параграфе 1. Во многих задачах классификации применение глубоких нейронных сетей с архитектурой прямого распространения сигнала дает хорошие результаты. К сожалению, на данный момент эти задачи ограничиваются лишь такими, для которых имеется большое количество размеченных данных для обучения. В то же время, для некоторых задач с малым количеством размеченных данных возможно получить достаточный для обучения глубоких нейросетевых архитектур объем данных, распределение которого однако является смещенным относительно тех данных,

для которых ожидается использование модели.

Важным примером таких задач являются задачи, для которых возможно получить множество размеченных синтетических данных, однако довольно часто их распределение будет отличным от реального. Обучение классификатора в условиях наличия смещения между тестовыми данными и данными для обучения называется доменной адаптацией.

Мы рассматриваем предложенную в [1] архитектуру нейронной сети в параграфе 2. Там же мы описываем реализацию этого метода на языке программирования Python.

В рамках исследования была построена нейросетевая архитектура прямого распространения сигнала, включающая в себя слой обращения градиента. Она описана в параграфе 4. Было исследовано влияние различных параметров модели на получаемый в итоге результат. Результаты проведенных экспериментов описаны в параграфе 5.

1. Теоретический материал

В этом параграфе дается формальное определение задачи классификации, а так же описывается один из методов ее решения — нейронная сеть.

1.1. Задача классификации

Предполагаем, что модель получает на вход объекты $x \in X$, которым соответствуют метки классов $y \in Y$. Предполагается, что $Y = 1, 2, ..., L$ — конечное множество размера L. $S(x, y), T(x, y)$ — двумерные случайные величины на $X \otimes Y$ будем называть *исходным* и *целевым* распределениями (или *доменами*) соответственно. Далее считаем что между ними существует некоторый сдвиг — они в целом схожи, но различны.

Предполагаем, что нам даны $x_1, x_2, ..., x_N$ — обучающая выборка, выделенная как из исходного, так и из целевого доменов.

Также нам известна бинарная переменная d_i, которая равна единице для объектов выборки из $S(x)$ и нулю для $T(x)$. Ее мы будем называть *меткой домена*. При $d_i = 0$ нам известна метка класса y_i, иначе же нет. Именно ее нам необходимо предсказывать для тестовых данных.

Отметим существование такого явления, как *переобучение*. Это явление, при котором частота ошибок на тестовой выборке выше, чем на обучающей. Оно возникает при использовании черезмерно сложных моделях. Методы борьбы с ней называются *регуляризацией*.

1.2. Устройство нейронной сети

Перед тем, как описать устройство нейронной сети, введем понятие *нейрона*. Под нейроном мы имеем в виду функцию $h_{w,b}(x) : \mathbb{R}^n \to \mathbb{R}$:

$$h_{w,b}(x) = f(w^T x) = f(\sum_i w_i x_i + b)$$

- $f : \mathbb{R} \to \mathbb{R}$ — функция активации нейрона;
- $w_i \in \mathbb{R}$ — весовые коэффициенты нейрона.

Таким образом один слой в нейронной сети можно описать как функцию $h_{W,B} : R^n \to R^m$ из m определенных выше функций, где W — матрица весов слоя.

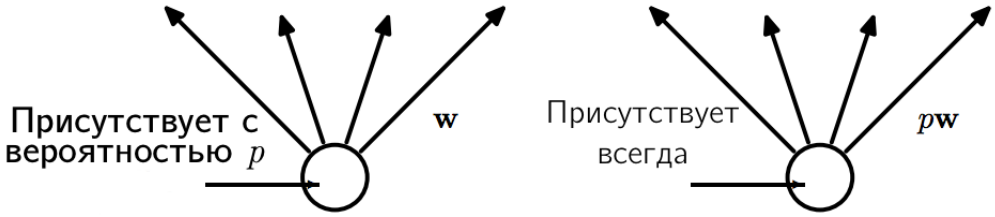
Нейронная сеть представляет собой сеть из нескольких слоев. Нейронные сети с архитектурой прямого распространения сигнала состоят из трех частей: входного слоя, скрытого слоя и выходного слоя.

1.3. Обучение нейронной сети

Приведем краткое описание используемого далее алгоритма обучения нейронной сети — стохастического градиентного спуска.

1.4. Слой Dropout

В работе [7] был предложен dropout-слой как средство регуляризации. Его суть заключается в следующем: во время обучения нейрон такого слоя присутствует в модели с фиксированной вероятностью p; во время же использования модели он присутствует всегда, однако его вес умножается на p.

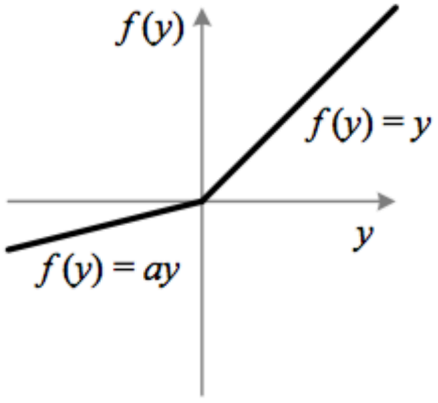


1.5. Функция активации PReLU

В статье [8] предлагается кусочно-линейная функция активации, под названием PReLU. Она определяется следующим образом:

$$f(y_i) = \begin{cases} y_i, & \text{при } y_i >= 0 \\ a_i y_y, & \text{при } y_i < 0 \end{cases} \tag{1}$$

Заметим, что параметр a_i уникален для каждой координаты вектора y. Эти коэффициенты близки к нулю и подбираются в процессе обучения.



2. Исследуемый метод

Ниже мы опишем предложенную в [1] архитектуру. Она представима в виде сети из трех отдельных сегментов.

Первый из сегментов будем называть *экстрактором признаков* и обозначать G_d. Он занимается тем, что по исходному входу $x \in X$ возвращает вектор $f \in \mathbb{R}^D$. Если обозначить за θ_f параметры всех слоев экстрактора, то можно отметить что $f = G_d(x, \theta_d)$.

Второй сегмент G_y затем продолжает первый и называется *классификатором класса*. Он по выданным G_f признакам f возвращает метку y. Его параметры — θ_y. Другими словами, $y = G_y(f, \theta_y)$.

Аналогично третий сегмент G_d продолжает экстрактор признаков и возвращает метку домена d и имеет параметры θ_d: $d = G_d(f, \theta_d)$. Его мы будем далее называть *классификатором домена*.

При процедуре обучения нашей целью является проведение *доменной адаптации* — получение признаков f, для которых распределения $S(f) = \{G_f(x; \theta_f) \mid x \sim S(x)\}$ и $T(f) = \{G_f(x; \theta_f) \mid x \sim T(x)\}$ будут схожи. Именно с этим и справляется предложенный в [1] метод.

Данный метод рассматривает следующий функционал:

$$\begin{gathered} E(\theta_f, \theta_y, \theta_d) = \\ \sum_{\substack{i=1..N \\ d_i=0}} L_y(G_y(G_f(x_i; \theta_i); \theta_y), y_i) - \lambda \sum_{i=1..N} L_d(G_d(G_f(x_i; \theta_i); \theta_d), y_i) = \\ \sum_{\substack{i=1..N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1..N} L_d^i(\theta_f, \theta_d) \end{gathered} \tag{2}$$

- $L_y(\,\cdot\,,\,\cdot\,)$ — функция потерь классификатора класса;
- $L_d(\,\cdot\,,\,\cdot\,)$ — функция потерь классификатора домена;

Далее, производится поиск параметров $\hat{\theta}_f, \hat{\theta}_y, \hat{\theta}_d$, образующих седловую точку функционала:

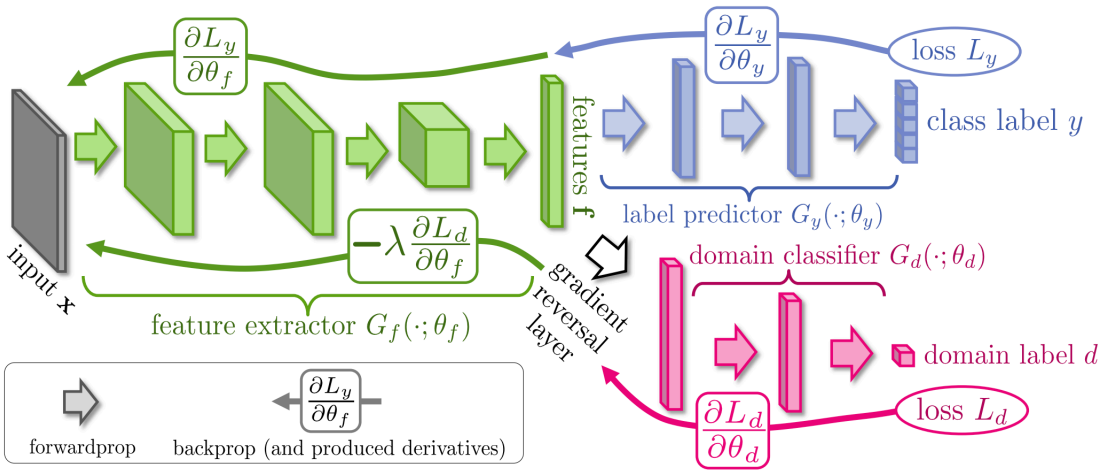
$$(\hat{\theta}_f, \hat{\theta}_d) = \arg\min_{\theta_f, \theta_d} E(\theta_f, \theta_y, \hat{\theta}_d)$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)$$

При таких параметрах с одной стороны будет достигаться минимум функции потерь для классификатора класса, а с другой максимум для функции потерь классификатора домена. Это обеспечит признаки f как

оптимально различающими классы, так и независимыми от домена — другими словами, произойдет доменная адаптация.

Идея метода в следующем: классификатор домена подключается к экстрактору признаков через специальный *слой обращения градиента*, описанный далее. Благодаря своим свойствам, он позволяет при проведении обучения получать такие параметры θ_f, которые максимизируют функцию потерь классификатора домена, но при этом такие θ_d, которые ее минимизируют.



3. Задача

Для исследования метода была выбрана задача из соревнования [4] с сайта `kaggle.com`. Оно проходило с 20 июля по 12 октября 2015 года. Соревнующимся был дан набор данных, содержащий как синтетические, так и реальные данные из эксперимента LHCb Большого Адронного Коллайдера. По этим данным было предложено построить модель, которая сможет выявлять наличие событий, соответствующих распаду $\tau \to 3\mu$. Обнаружение таких событий будет означать нарушение предсказываемого Стандартной моделью свойства сохранения аромата у лептонов, которому данный распад не подчиняется.

Далее мы подробно опишем формат данных и использованные для ранжирования участников соревнования метрики.

3.1. Формат данных

Для анализа участникам состязания были предоставлены четыре файла:

1. `training.csv` — набор размеченных данных для обучения классификатора, целевой переменной является переменная `signal`. Ее значение 1 описывает события сигнала, 0 — фона.

2. `check_agreement.csv` — набор размеченных данных для проверки на согласованость (описана в 3.2.1). Признаки такие же, как и в `training.csv`, однако `signal` отвечает за домен — 1 для синтетических данных, 0 для реальных.

3. `check_correlation.csv` — набор данных для проверки классификатора на коррелированность с массой. Включает дополнительный признак `mass`. Подробней описано в 3.2.2.

4. `test.csv` — набор неразмеченных данных, для которых участникам необходимо предоставить метки классов.

3.2. Процедура проверки решения

3.2.1. Проверка на согласованность

Так как для обучения классификатора участникам были предоставлены как реальные, так и синтетические данные, при создании хорошего классификатора возможно было бы использовать такие признаки, которые были плохо смоделированы во время получения синтетических данных.

Чтобы избежать этого, при проверке решения используется критерий Колмогорова-Смирнова. Он рассчитывается следующим образом:

$$KS = max|F_{simulation} - F_{real}|$$

- $F_{simulation}, F_{real}$ — функции распределения для синтетических и реальных данных соответственно

Большее значение этого критерия будет означать большую зависимость между значениями классификатора на реальных и синтетических данных соответственно. Для принятия проверочной системой решения требовалось, чтобы $KS < 0.09$.

3.2.2. Проверка на корреляцию с массой

Помимо этого, полученный классификатор может в процессе обучения по предоставленным данным научиться восстанавливать массу исходной частицы, распад которой был зафиксирован. Такое поведение вызывает некорректную оценку фона и может привести к ложным обнаружениям сигнала. Таких обнаружений требуется избегать.

Для проверки классификатора на корреляцию с массой исходной частицы применяется критерий Крамера-Мизеса, описанный в [5]. Для его вычисления функция распределения на всем диапазоне масс сравнивается с функцией распределения на некотором интервале. После этого итоговое значение для критерия усредняется по всем интервалам:

$$CvM_{interval} = \int (F_{global} - F_{local})^2 dF_{global},$$

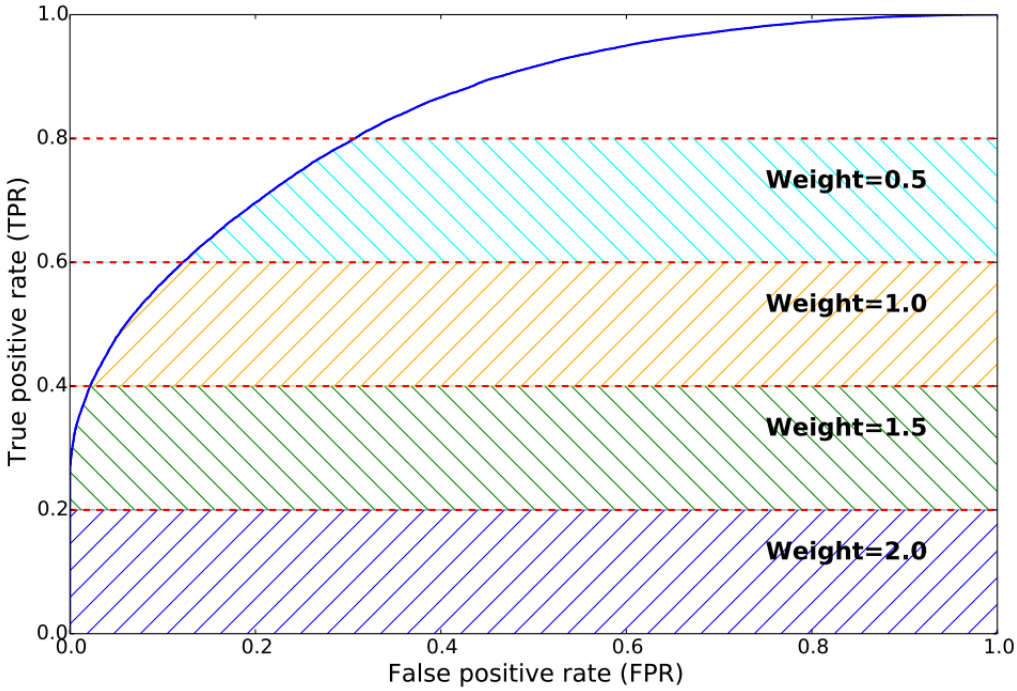
$$CvM =< CvM_{interval} >_{interval}$$

- F_{global}, F_{local} — функции распределения для всех данных и для данных в некотором интервале масс

Чтобы решение засчитывалось, в условии задачи требуется чтобы $CvM < 0.002$.

3.2.3. Метрика качества

Качество полученного классификатора оценивалось при помощи взвешенного показателя AUC. Этот показатель рассчитывается как площадь под ROC-кривой, построенной для результатов классификатора. В данном случае вклад в показатель AUC конкретных сегментов под ROC-кривой показан ниже:


1.0
0.8
0.6
0.4
0.2
0.0
True positive rate (TPR)
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate (FPR)
Weight=0.5
Weight=1.0
Weight=1.5
Weight=2.0

4. Применение метода к задаче

Далее приведена исследованная нейросетевая архитектура: ее диаграмма, ее описание в виде программного кода на языке Python с использованием фреймворка Keras, а так же описаны методы оценки результатов экспериментов.

4.1. Описание использованной нейросетевой архитектуры

Для исследования метода было выбрано следующее строение:

Обучение происходит следующим образом. Сначала нейросеть обучается при $\lambda = 0$ некоторое количество времени. После этого частично обученная сеть дает результат для данных соответствия, а затем дообучивается с их добавлением следующим образом: на вход классификатору домена для данных соотвествия выставляется метка, получанная ранее, а классификатору домена такая, которая уже находится в данных. Таким образом мы стараемся максимально сохранить результат именно классификатора класса, пытаясь изменить веса внутри экстрактора признаков. При этом мы плавно начинаем повышать *lambda*, каждый раз вычисляя метрики. Останавливаемся в тот момент, когда метрика KS не станет удовлетворять условиям соревнования, а именно станет меньше 0.09.

4.2. Особенности программной реализации

Для решения задачи был выбран язык программирования Python, а так же фреймворк для проектирования нейронных сетей Keras. Выбор этих инструментов мотивирован простотой их использования.

Отметим отдельно особенности фреймворка Keras, позволяющие лучше описать нейросетевую модель. В версии 1.0.0 во фреймворке появилась так называемое Functional API. Продемонстрируем его применение:

```
n_extracted_features = 100
lam = 0.2

f = feature_extractor(input.shape[1], n_extracted_features)
l = label_classifier(n_extracted_features)
d = label_classifier(n_extracted_features,
                     name="domain_classifier")

model_input = Input(shape=(f[1],))
features = f(model_input)
label_class = l(features)
```

```
grl = GradientReversal(lam,
                input_shape=(domain_classifier.input_shape[1],))
domain_class = d(grl(f))
m = Model(input=[model_input], output=[label_class, domain_class])
m.compile(loss=['categorical_crossentropy',
   'categorical_crossentropy'], loss_weights=[1, 1],
   metrics=['accuracy', 'accuracy'], optimizer='rmsprop')
```

Как можно заметить, вместо описывания в коде всей нейросети графом, расписывая всех слоев и всех связей между ними, достаточно передать каждому следующему слою предыдущий как аргумент функции. Это заметно упрощает описание модели программным кодом.

4.3. Оценка результатов работы

Поскольку в условии задачи явно проверяются показатели AUC, KS и CvM, они и были выбраны для исследования во время эксперимента.

Во время процесса обучения производится нахождение этих показателей. После проведения обучения отрисовывается график, на котором видно их поведение.

Основным показателем качества полученной модели был выбран показатель AUC, который высчитывается при загрузке решения на сайт `kaggle.com`. В соответствии с этим показателем на этой сайте доступен рейтинг участников. Эти результаты для тех решений, которые удовлетворяют проверке на согласованность и корреляцию (указанные в 3.2.1 и 3.2.2).

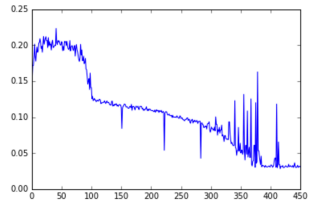
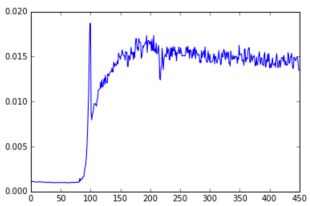
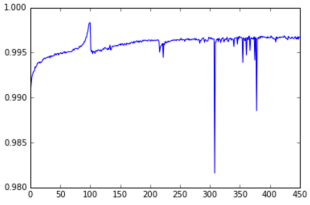
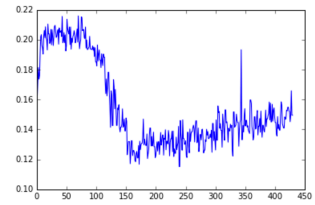
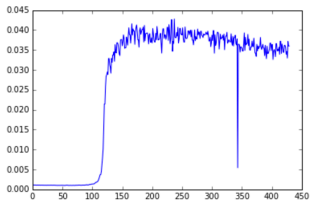
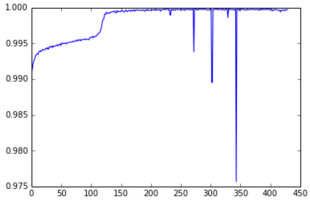
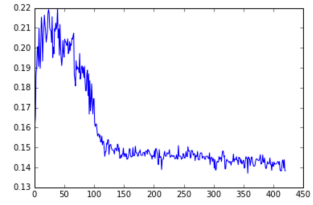
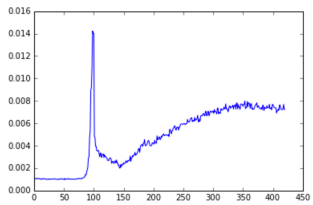
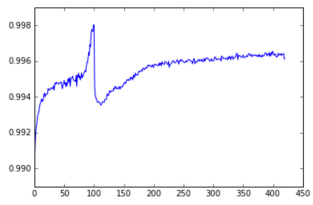
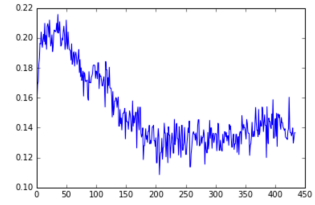
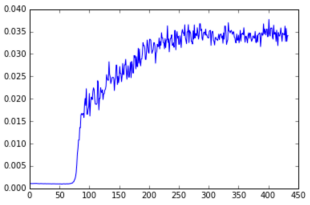
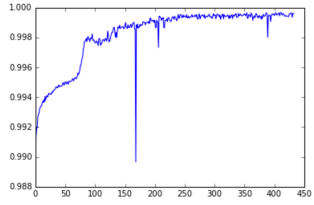
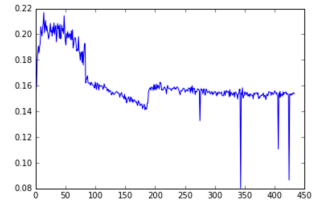
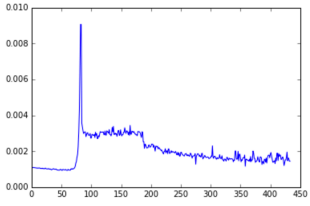
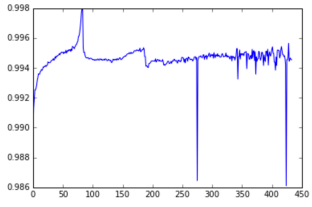
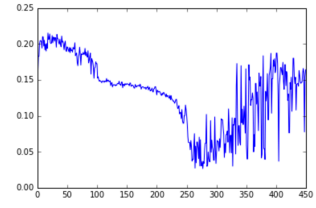
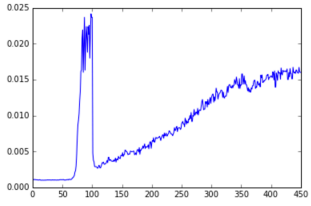
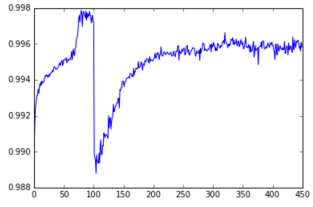
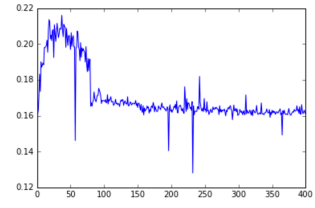
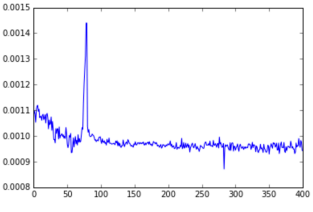
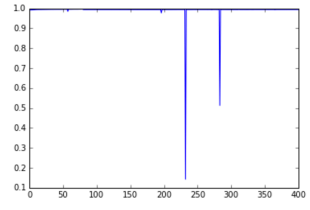
5. Результаты эксперимента

Программа, реализующая выбранную модель, запускалась на компьютере Macbook Pro Late 2013 с процессором 2,4 GHz Intel Core i5 и 8 Гб оперативной памяти. Избранные результаты приведены в таблице ниже.

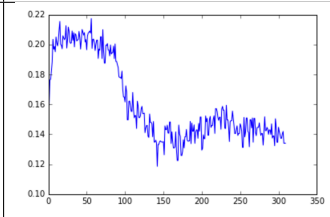
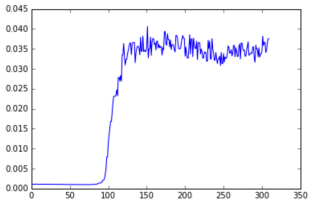
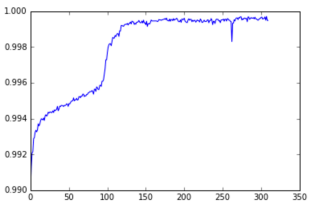
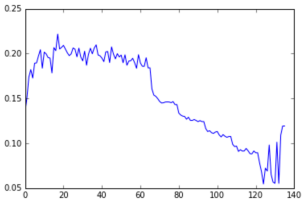
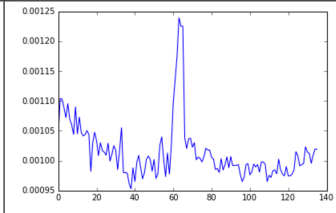
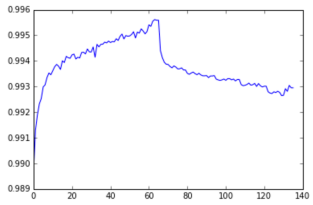
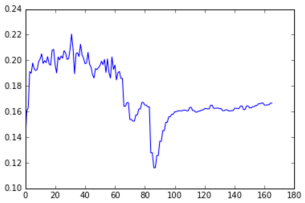
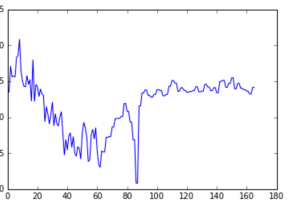
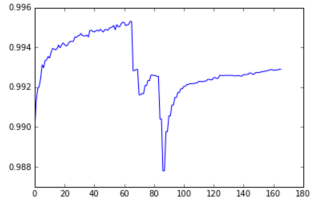
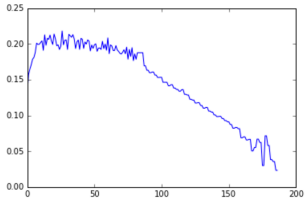
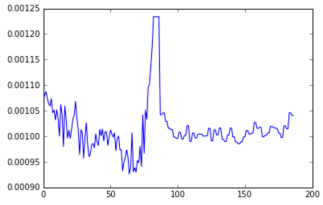
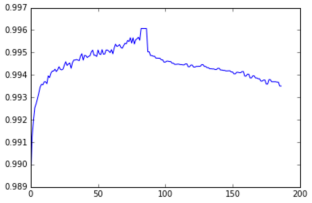
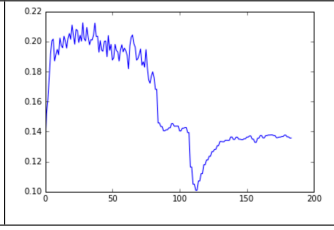
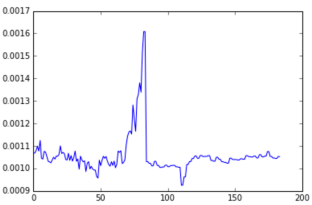
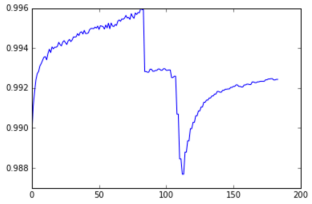
Таблица 1: *Параметры, изменявшиеся во время эксперимента.*

id	$extracted\_features$	$epochs$	$transfering\_ratio$	$steps$	λ_{min}	λ_{max}	$Dropout$	agreement_data
1	100	400	0.75	50	0	0.5	нет	да
2	100	400	0.80	50	0	0.5	нет	нет
3	100	400	0.75	100	0	0.2	да	да
4	100	400	0.79	50	0	0.5	нет	нет
5	100	400	0.79	50	0	0.5	нет	да
6	100	400	0.75	50	0	0.8	да	да
7	100	400	0.80	1	0.4	0.4	нет	да
8	100	300	0.80	50	0	0.3	нет	нет
9	100	130	0.5	5	0.1	0.5	да	да
10	120	130	0.5	50	1	5	да	да
11	120	130	0.36	50	0.1	0.5	да	да
11	120	130	0.36	50	1	5	да	да

Таблица 2: *Графики поведения метрик 3.2.1, 3.2.2 и 3.2.3*

id	KS	CvM	AUC
1			
2			
3			
4			
5			
6			
7			

Отметим основные факты, полученные из приведенных выше экспериментов:

id	KS	CvM	AUC
8			
9			
10			
11			
12			

- После проведения некоторого числа (зависящего от наличия и величины dropout) эпох нейронная сеть начинает коррелировать с массой — другими словами, она формирует признак, который частично ее восстанавливает из исходных данных. Помимо этого, уменьшению параметра CvM после первоначального обучения способствует лишь только использование данных из проверки на соответствие.

- Без использования agreement_data метрика KS занижается, однако недостаточно сильно для засчитывания решения.

Таким образом, можно обобщить приведенные выше факты и отметить работоспособность данного метода в этой задаче. Однако, его все еще возможно улучшить. Например, можно на полученных в результате обучения этой нейросетевой модели признаках протестировать другие

методы машинного обучения, вроде градиентного бустинга. Помимо этого, имеет смысл получить больше синтетических данных похожей структуры, но другого распада — это может улучшить выделение признаков. Помимо этого, метод стоит попробовать в других задачах — например, в задаче классификации изображений.

Заключение

В работе продемонстрировано применение доменной адаптации в задаче фильтрации событий для детектора частиц коллаборации LHCb Большого Адронного Коллайдера. Были рассмотрены различные приемы обучения нейросетевого классификатора. В результате было получено решение задачи Flavours of Physics на сайте Kaggle, показывающее результат лучше методов, используемых в ЦЕРН.

Полученную нейронную сеть возможно с легкостью адаптировать для решения других задач.

Список литературы

[1] *Yaroslav Ganin, Victor Lempitsky.* Unsupervised Domain Adaptation by Backpropagation
http://arxiv.org/pdf/1409.7495.pdf

[2] *François Chollet.* keras, репозиторий GitHub
https://github.com/fchollet/keras

[3] *Thomas Blake, Marc-Olivier Bettler, Marcin Chrząszcz, Francesco Dettori, Andrey Ustyuzhanin, Tatiana Likhomanenko.* Flavours of Physics: the machine learning challenge for the search of $\tau^- \to \mu^+\mu^-\mu^-$ decays at LHCb
https://kaggle2.blob.core.windows.net/competitions/kaggle/4488/media/lhcb_description_official.pdf

[4] Flavours of Physics: Finding $\tau^- \to \mu^+\mu^-\mu^-$
https://www.kaggle.com/c/flavours-of-physics/

[5] H. Cramér. On the composition of elementary errors — Scandinavian Actuarial Journal, 1928

[6] Alex Minnaar. Deep Learning Basics: Neural Networks, Backpropagation and Stochastic Gradient Descent
URL: http://alexminnaar.com/deep-learning-basics-neural-networks-backpropagation-and-stochastic-gradient-descent.html

[7] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting.
URL: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.
URL: https://arxiv.org/pdf/1502.01852v1.pdf