О решении задачи распознавания действий, производимых над объектом, на основе показаний акселерометра

Н. В. Куриленко

Решение задачи распознавания движений, производимых над объектом, имеет обширное количество приложений в робототехнике, производстве мобильных устройств и мобильных приложений. Она является примером типовой задачи классификации и во многих случаях может быть решена сравнительно простым анализом показаний приборов: акселерометра, гироскопа, магнитометра и других. Однако в случаях, когда графики показаний приборов для различных жестов визуально неотличимы и поверхностный анализ не даёт содержательных результатов, приходится пользоваться методами машинного обучения. В статье рассматривается одна из таких задач и освещается тема общей методологии решения задач классификации.

Ключевые слова: машинное обучение, анализ данных, распознавание жестов, Скрытые Марковские Модели, акселерометр

Введение

В последнее десятилетие наблюдается стремительное развитие области машинного обучения и анализа данных. Этот рост обусловлен, в первую очередь, всеобщей компьютеризацией и появлению доступа к огромным массивам данных[9]. Основную часть задач области составляют типовые задачи классификации. На текущий момент уже получены содержательные

результаты в таких вопросах как распознавание объектов на фотографиях, распознавание речи, анализ предпочтений пользователей, детекция спама и многих других.

Однако, несмотря на эти и другие достижения, до сих не сформулировано общей методологии решения подобных задач. В большинстве случаев, нам просто приходится ждать появления принципиально нового алгоритма машинного обучения, который решит данную задачу, а до тех пор – рутинно совершенствовать работу разнородных вариаций известных к тому моменту алгоритмов.

В данной статье мы сделаем шаг в сторону разработки методологии. А именно, рассмотрим два основополагающих направления при работе с задачами области.

В первом случае мы пытаемся формализовать задачу, разработать модель и на её основе сформировать решение. Очевидно, что такой подход является наиболее естественным, однако во многих случаях сделать это очень трудно. К примеру, при решении задач области машинного зрения нам приходится моделировать распространение света в среде, решать задачу восстановления трёхмерного тела по плоским проекциям, оптимизировать полученные алгоритмы и т.д. Но, тем не менее, на сегодняшний день этим путём не реализовано сколь либо действующее решение общей задачи распознавания образов (хотя некоторые частные случаи были успешно решены).

Во втором случае нам безразличны какие-либо знания о самой задаче, важна лишь общая постановка. Заметим, что задачи классификации обладают одной важной особенностью - мы можем сравнительно быстро проверить точность предложенного решения. Это означает, что мы можем просто обратиться к базе уже готовых решений (для других задач) и протестировать использованные алгоритмы на нашей задаче. С учётом того, что разные задачи классификации на фундаментальном уровне могут обладать схожей структурой, во многих случаях мы способны получить качественные результаты за довольно короткое время, и потому данный метод широко используется на практике.

Постановка задачи

В работе предлагается решение задачи классификации движений производимых над физическим объектом небольших размеров. При решении задачи мы будем использовать оба метода и покажем, в каких случае стоит придерживаться первого направления, а в каких случаях - второго. Как мы увидим далее, для качественного решения поставленной задачи нам потребуются оба способа.

Проверка теоретической базы осуществлялась на устройстве, представляющей собой плату с датчиком, установленную в пластмассовый корпус, распечатанный на трёхмерном принтере. Корпус имеет форму шара диаметром полтора дециметра, усеченного снизу для создания устойчивого основания.

При постановке любой подобной задачи мы определяем некоторый набор типовых действий, которые, как мы ожидаем, могут быть произведены над объектом. Назовём их базовыми действиями. К примеру, в статье о задаче классификации активностей пользователей «умных браслетов» [7], базовыми действиями является набор движений: «стоит», «идёт», «бежит», «поднимается по лестнице», «спускается по лестнице», «приседает», «пользуется пылесосом», «чистит зубы». В статье о задаче классификации жестов, производимых игровым контроллером Wii [8], под базовыми действиями понимается набор жестов: «квадрат», «круг», «раскачивание», «символ Z», «теннисный мах». Иными словами, набор движений выбирается, исходя из контекста задачи.

В данной статье мы рассмотрим самые простые движения, связанные с изменением положения устройства в пространстве, то есть поступательное движение и вращение (для простоты «поступательное движение» будем именовать «толкание»). Также, дабы усложнить задачу, мы выделим в отдельную категорию два типовых движения: стук по корпусу и поднятие устройства. Под «поднятием устройства» мы понимаем движение, в ходе которого устройство берут в руки. Под «стуком» понимается серия из нескольких слабых ударов по корпусу. В качестве ограничения считаем, что действия должны распознаваться в состоянии

покоя, когда единственной внешней силой, воздействующей на акселерометр, является сила земного притяжения.

Таким образом, получаем четыре базовых движений:

$$D = \{\text{«стук»},\text{«толчок»},\text{«вращение»},\text{«поднятие»}\} = \\ = \{d_1,d_2,d_3,d_4\}.$$

Замечание: существует большое количество движений, которые являются некой комбинацией четырёх базовых движений. Вместо того чтобы просто повернуть или толкнуть устройство, можно протащить его по сложной кривой, произведя таким образов оба действия сразу. В статье мы не будем фокусироваться на этих «сложных» движениях. В общем случае, смешанные движения должны выделяться в отдельную категорию, либо же классификатор должен уметь распознавать «доминирующее» действие.

Выбор акселерометра

Дабы не столкнуться с техническими ограничениями, нам требуется плата с достаточно быстрым процессором и чувствительными датчиками. Поскольку сегодня большинство процессоров способны в течение одной секунды выполнять сотни тысяч операций сложения, умножения и сравнения, основным вопросом становится выбор акселерометра. Во-первых, для выбранных действий нам необходим акселерометр, способный распознавать силу в две-четыре g. Во-вторых, частота дачи показаний должна быть достаточно высокой, чтобы распознавать стук по корпусу, так как это самое непродолжительное из всех базовых движений. В сущности, движение d_1 – это кратковременный микротолчок. При записи этого движения с помощью приборов высокой точности, можно заметить, что в среднем время соприкосновения с корпусом длится приблизительно 10-20 миллисекунд. Таким образом, минимальная частота показаний акселерометра составляет приблизительно 50 показаний в секунду.

Модель акселерометра, используемая нами для измерения показаний, — LSM303DTR — способна на распознавание силы величиной в шестнадцать g. Частота показаний непостоянная и

составляет 100±5 показаний в секунду, то есть примерно одно показание каждые 10 миллисекунд. Все показания представляет собой тройки чисел: каждое число соответствует координате x, y или z и отражает величину проекции силы, действующей на датчик, на соответствующую ось (показание в момент t будем обозначать как $A(t) = (A_x(t), A_y(t), A_z(t))$). В условиях абсолютной невесомости показаниями являются тройки нулей. Акселерометр имеет собственные единицы измерения вектора силы: изменению длины проекции на величину g соответствует изменение координаты $A(t)$ приблизительно на 1 000 000 единиц (далее будем именовать их условными единицами).

Методология решения

Очевидно, что положение датчика после установки на плату влияет на конечное решение: одному и тому же движению будут соответствовать различные показания от датчиков, установленных в разных положениях. Потому нашей целью будет поиск классификатора, инвариантного относительно положения датчика на плате. Кроме того, решение не должно ограничиваться параметрами устройства и должно обобщаться на другие, более чувствительные модели акселерометра.

Условно задачу можно разделить на две подзадачи: создание классификатора движений (теоретическая часть) и оптимизация классификатора для установки на устройство (полученный классификатор должен работать быстро).

Обратим внимание на задачу классификации. Заметим, что все описываемые движения ограничены по времени (приблизительно двух секунд будет достаточно, чтобы распознать каждое из них). За это время датчик присылает конечное число показаний, область значений которых также ограничена. Это означает, что, вообще говоря, мы имеем дело с конечным набором возможных показаний датчика. Каждому элементу этого набора мы должны сопоставить одно из четырёх базовых движений или ни одного. Пусть A - область значений показаний датчика для каждой из трёх координат, а n – количество показаний, пришед-

ших за ограничивающий промежуток времени. Тогда описанное отображение задаёт функцию:

$$F : (A^3)^n \to D \cup \{«\emptyset»\}$$

Наша задача найти эту функцию или достаточно близкую к ней.

Сделать это можно двумя способами: либо исследовать саму функцию, либо пытаться сымитировать её и отталкиваться от полученных результатов. Первый способ предполагает построение физической модели и/или выделение и сбор специфических свойств функции. К примеру, мы сразу можем заключить, что при поднятии устройства, мы можем ожидать, что в момент поднятия вектор внешней силы значительно увеличиться вдоль направления действия гравитации, и что это должно быть отражено в классификаторе в явном или неявном виде. Во втором случае стоит определить класс алгоритмов, к которым относиться функция, собрать обучающую и тестовую выборку, вычислить параметры итогового алгоритма и проверить решение на тестовой выборке.

Вторая подзадача является классической задачей оптимизации. Будем считать, что для успешного решения распознавание движения должно осуществляться с задержкой не более 0,5 секунд.

Выделение гравитации

Первым шагом к решению задачи служит исключение гравитации из показаний акселерометра. Поскольку мы хотим создать классификатор, инвариантный относительно ориентации датчика, удаление гравитации так или иначе должно быть задействовано в алгоритме, обрабатывающим сигнал. Заметим также, что в случае любого из действий d_{1-3}, устройство двигается только горизонтально (то есть перпендикулярно силе тяжести), а в случае последнего действия – горизонтально и вертикально. Это также означает, что нам будет полезно знать направление гравитации.

Для большинства акселерометров показания датчиков в состоянии покоя представляют собой небольшой шум, сфокуси-

рованный вокруг некоторой точки (рис. 1). Чтобы избавиться от шума и найти координаты этой точки, можно взять среднее значение на некотором временном интервале. Однако, если в течение этого интервала над устройством будет произведено какое-либо действие, результаты подсчётов будут неверны. Чтобы избежать этого, вместо среднего значения на интервале будем считать скользящее среднее до тех пор, пока полученное значение не будет постоянным в течение некоторого интервала времени. Значение скользящего среднего в момент t обозначим через $G_w(t)$, где $w = x, y, z$. Координаты полученного значения вектора гравитации обозначим через $gravity_w$, а скорректированные показания датчика - через $\hat{A}_w(t)$.

Получаем процедуру:

while $|G_w(t) - G_w(t-T)| - |G_w(t-T) - G_w(t-2T)| - |G_w(t-2T) - G_w(t-3T)| > C$: **do**
$\quad G_w(t) = G_w(t-1)\alpha - A_w(t)(1-\alpha)$
end
$gravity_w = G_w(t)$
$\hat{A}_w(t) = A_w(t) - gravity_w$
$(w = x, y, z)$

Параметры α, T, C подбираются вручную. К примеру, в случае с α: если параметр близок к нулю, то скользящее среднее становится слишком чувствительным к шуму, а если близок к единице – слишком медленно возвращается в состояние покоя после каких-либо действий над устройством. Поведение скользящего среднего для $\alpha = 0.95$ показано на рис.2. Алгоритм работает менее секунды и слабо чувствителен к шуму. Таким образом, примем значение α равное 0.95. Путём аналогичных рассуждений получим значения для остальных параметров: T = 20 итераций, C = 5 000 условных единиц.

После получения значения вектора гравитации, этот вектор не стоит оставлять постоянным, так как алгоритм будет работать некорректно в случае, когда устройство ставится на наклонную плоскость (в этом случае измениться направление действия силы тяжести). Таким образом, после распознавания

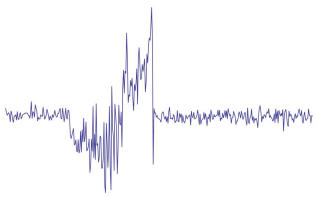

Рис. 1. Пример показаний датчика

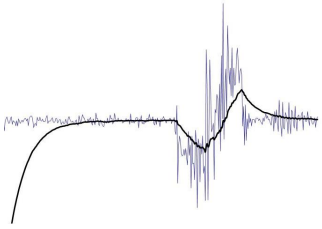

Рис. 2. Скользящее среднее с параметром $\alpha = 0.95$

какого-либо движения классификатор должен заново пересчитывать значение гравитации. В большинстве случаем этот процесс занимает не более секунды.

Классификатор движения d_1

Поведение акселерометра

Следующим шагом к построению классификатора является написание алгоритма, отделяющим действие d_1– удара по корпусу – от остальных действий и случайного шума. Как уже было отмечено в 1.1., для решения задачи нам понадобится как прямой анализ искомой функции F, так и методы машинного обучения. Здесь мы воспользуемся первым методом.

Современные акселерометры являются микроэлектромеханическими системами (МЭМС), и распознавание силы, воздействующей на прибор, производится за счёт измерения уровня деформации внутренних компонент (рис. 3).

Как уже было сказано в 1.2., d_1 отражается в показаниях датчика в виде кратковременных отклонений вектора внешней силы. Время продолжительности скачка — 10 - 20 миллисе-

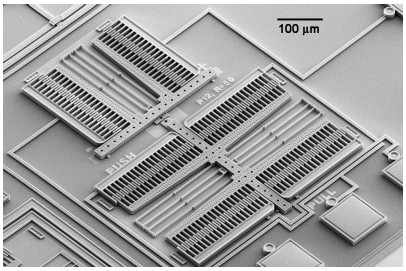

Рис. 3. Микроснимок компоненты МЭМС для измерения ускорения

кунд. Кроме того, можем предположить, что в зависимости от силы удара по корпусу, длина вектор отклонения варьируется от $0.01g$ до одной g. Для нашей модели датчика это соответствует изменению показаний в интервале от 100 000 до 1 000 000 условных единиц. Также заметим, что в виду эластичности компонент, ответственных за измерение силы, в течение некоторого времени после скачка показания будут возвращаться в исходную точку наподобие пружины, возвращающейся в исходное положение после натяжения. То есть на графике мы будем наблюдать затухающие колебания с небольшим шумом.

Именно такую картину мы увидим, если запишем данное действие (рис. 4).

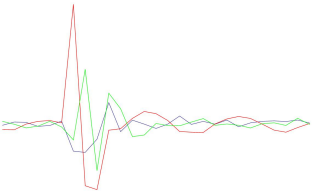

Рис. 4. График движения d_1

По аналогии с уже выявленными ограничениями мы можем взглянуть на графики и выделить дополнительные:

- До удара по корпусу устройство находится в состоянии покоя.

- Прирост вектора внешней силы происходит стремительно; величина прироста по одной из координат составляет минимум 150 000 условных единиц, иначе выход из состояния покоя соответствует другому действию или является случайным шумом.

- Затухающие колебания продолжаются 30 – 90 миллисекунд.

- Критерием затухания является любое из двух событий: вектор силы приближается к состоянию покоя на 100 000 условных единиц, длина вектора приращения силы становится меньше 100 000 условных единиц.

- По окончанию действия устройство находится в состоянии покоя ещё 50 миллисекунд.

Классификатор, построенный на основе этих ограничений, показал достаточно высокую точность (99.09%) при проверке на тестовой выборке.

Общая методология решения заключается в итеративном поиске подобных ограничений, конструировании классификатора на их основе и программы проверке на тестовом наборе движений. Собранные данные должны помогать (хоть и не точно) определять прообраз базовых движений.

Итоговый классификатор

В ходе решения была записана выборка, включающая 87 движений d_1, 111 движений d_2, 128 движений d_3 и 115 движений d_4. Построенный классификатор успешно распознал все 87 движений d_1, но при этом дал 3 ложноположительных результатов для движений d_2 и 1 ложноположительный результат для движений d_3. В пункте 4 мы модифицируем классификатор, чтобы исключить эти неточности.

Замечание: к сожалению, даже в случае максимально возможной точности классификатора, он не может корректно распознавать движение d_1 в реальной жизни. К примеру, если

устройство стоит на столе, то обычный шум производимый другими предметами (удар по столу) также может быть распознан удар по корпусу. Оба движения производят одинаковые колебания, действующие на датчик. Отличить их возможно только при анализе данных с других датчиков, например, микрофона в корпусе.

Однако вероятность распознавания подобных «ложных» движений мала и в большинстве случаев они не следуют друг за другом. Чтобы свести эту вероятность к достаточно низкому уровню, принято решение изменить классификатор. Алгоритм должен распознавать как минимум два удара по корпусу в течение 2 секунд, иначе движение считается нераспознанным.

Реализация классификатора производилась на языке Python. Код алгоритм описан в приложении А.

Распознавание трёх других действий

Распознаватель шума

Перейдём к классификации оставшихся трёх действий. Ключевым отличием этой задачи является то, что графики движений d_2 и d_3 визуально практически неотличимы и нет возможности разделить эти действия, воспользовавшись методологией, изложенной в предыдущей главе. Однако эти методы могут помочь отделить три исследуемых действия от случайного шума. Очень сильный стук, слабое покачивание и другие не интересующие нас действия также должны быть отделены.

На основании тех же рассуждений, что были изложены в 3.1., выделим следующие особенности d_2, d_3 и d_4:

- Действие начинается со смещения вектора внешней силы на величину большую, чем 60 000 условных единиц.
- В течение любого интервала из 50 миллисекунд энергия сигнала должна превышать 150 000 условных единиц.
- Действие длится не более 2 секунд.
- По окончанию действия устройство остаётся в состоянии покоя как минимум одну десятую секунды.

Реализация на языке Python классификатора на основе описанных выше особенностей изложена в приложении Б.

Замечание: в предыдущем параграфе мы построили классификатор для первого движения. В виду того, что алгоритм работает в очень малых временных интервалах, в некоторых случаях он может распознать одно из трёх последних движений как удар по корпусу. Мы получили 4 ложноположительных результата такого рода. Чтобы избежать подобных ошибок, мы позволим классификатору давать показания только в случае, если классификатор для трёх последних движений не распознал шума. Таким образом, мы имеет два классификатора, один из которых имеет больший приоритет, в сравнение с другим.

Проверка сдвоенного классификатора на тестовой выборке показала, что он верно распознаёт последние три действия, но даёт 5 ложноположительных результатов для движения d_1 (движение классифицируется как d_2). Это означает, что в тестовой выборке присутствовали 5 ударов по корпусу, настолько сильных, что их показания были ложно приняты за толкание устройства. Такими ошибками на данных элементах выборки можно пренебречь.

Выбор модели

В предыдущем параграфе мы отделили действия d_2, d_3 и d_4 от остальных. Теперь необходимо отделить сами действия друг от друга. Как уже говорилось ранее, в виду визуальной неотличимости графиков показаний d_2 и d_3, мы будем решать задачу методами машинного обучения. Перейдём к первому шагу – выбору модели.

Решение подобных задач уже производилось в других исследования, к примеру, [5], [6], [7]. Ключевое отличие этой задачи от основной массы задач области анализа данных – непостоянная длина вектора у элементов обучающей и тестовой выборки. То есть любое движение, будь то поворот устройства или толкание, могут быть выполнены за разное время и, тем не менее, находиться в одном классе.

Существует не так много моделей, гибких к длине входных данных. Два самых распространенны класса моделей, обладающих подобными свойствами: Рекуррентные Нейронные Сети (RNN) и Скрытые Марковские модели (HMM). Приложения RNN являются довольно востребованной областью исследований, однако существенных результатов в решении интересующих нас задач пока не достигнуто. В то же время модель HMM широко распространена в решение многих задачи, в особенности, в задачах распознавания речи [1], [2]. Что же касается нашей задачи, то пример работы HMM можно найти, к примеру, в [5], [6], [8]. Далее в статье мы будем придерживаться данной модели.

Существует множество вариантов Марковских моделей. Мы будем фокусироваться на классической модели, рассмотренной в [3]. Марковская модель есть набор $\lambda = \{A, B, \pi\}$, где A - матрица $q \times q$, B - матрица $q \times k$, π – вектор длины q (где q - число состояний модели, k - количество символов в выходном алфавите), причём $\sum_{j=1}^{q} a_{ij} = 1$, $\sum_{j=1}^{q} b_{ij} = 1$ и $\sum_{i=1}^{q} \pi_i = 1$. При запуске модели начальное состояние выбирается с помощью дискретного вероятностного распределения на основе вектора π. Если выбрано состояние i, то модель генерирует выходной сигнал v, затем переходит в следующее состояние – с помощью распределения на основе i-ой строки B и i-ой строки A соответственно. Переходы между состояниями повторяется, пока не генерируется T символов или слово x длины T.

Слову x можно сопоставить вероятность $P(x|\lambda)$, с которой оно может быть сгенерировано данной моделью. Простейший классификатор на основе Марковских Моделей будет работать следующим образом:

1) Для каждого движения высчитывается отдельная Марковская Модель.

2) После получения данных от датчика подсчитывается вероятность того, что данное действия было сгенерировано той или иной моделью.

3) Действие классифицируется как d_2, d_3 или d_4, если вероятность того, что слово сгенерировано соответствующей Марковской Моделью, наивысшая.

Векторная квантизация

Прежде чем переходить к построению Марковской Модели для каждого действия, необходимо определить преобразования показаний датчика в выходные символы. Специфика модели заключается в том, все входные данные должны быть сжаты и представлены в некотором алфавите. Процесс сжатия называется векторной квантизацией. В идеальном случае, задачу можно было бы решить и без сжатия, однако тогда нам понадобится обучающая выборка значительных размеров. Запись такой выборки заняла бы непосильно много времени и, кроме того, мы столкнулись бы с большим объёмом вычислений в работе обучающего алгоритма. Поэтому мы должны определить эффективный вариант сжатия данных (т.е. информации, получаемой от датчика за одну итерацию, - координат $\hat{A}_x$, $\hat{A}_y$ и $\hat{A}_z$ в момент t), который сохранит принадлежность действий к своему классу (т.е. хэш-функцию на $(A^3)^n$ без коллизий). Очевидно, что такой способ сжатия существует, так как для большинства точек $(A^3)^n$ значения функции F в некоторой их окрестности будут одинаковыми.

При решении схожей задачи в [5], [8], использовался следующий метод сжатия:

1) Вначале фиксируется число символов k в алфавите.

2) Затем на сфере с центром в нуле равномерно распределяется k точек.

3) Для каждого показания $(\hat{A}_x, \hat{A}_y, \hat{A}_z)$ алгоритм возвращает номер ближайшей точки на сфере.

Модифицируем этот алгоритм следующим образом:

1) Фиксируется число символов k в алфавите.

2) Затем на сфере с центром в точке ноль равномерно распределяется k точек.

3) Сопоставляем каждому показанию $(\hat{A}_x, \hat{A}_y, \hat{A}_z)$ ближайшую точку на сфере (с номером $k' \in \{1, ..., k\}$).

4) Если длина вектора больше 55 000 условных единиц, то возвращается число $2k' + 1$; если длина меньше 20 000 условных единиц – возвращается число 1; иначе возвращается $k' + 1$. Здесь 20 000 и 55 000 - это приблизительно 0.33- и 0.66-квантили выборочного распределения всех троек $(\hat{A}_x, \hat{A}_y, \hat{A}_z)$, содержащихся в тестовой выборке.

Для реализации на языке Python нам понадобиться алгоритм, который принимает натуральное число N и возвращает координаты N точек, равномерно распределённых на сфере. Алгоритм равномерного распределения точек был описан в [5]. Однако данную задачу можно решить обычным методом спуска:

- Расставляем точки, используя некоторое фиксированное распределение.
- Немного меняем (случайно) положение точек, пытаясь минимизировать сумму расстояний между точками, пока эта величина не станет постоянной в течение большого количества попыток.

Реализация алгоритма на языке Python описана в приложении В.

Алгоритм Баума-Уэлша

Получив обучающую выборку с помощью алгоритма, описанного в 4.1., мы можем перейти к построению Марковских Моделей для каждого действия. Для этой цели будем использовать классический алгоритм Баум-Вэлша [3]. Качественное описание алгоритма дано в [4]. Кратко опишем процедуру.

Пусть нам дана обучающая выборка для отдельно взятого действия. Пусть она состоит из L элементов, каждый длиной T_l,

где $l \in \{1, \ldots, L\}$. Первым делом сжимаем каждый элемент методом, описанным в предыдущем параграфе. Теперь наша задача: для заданного наперёд количества состояний q, найти такую Марковскую Модель, что вероятность $P(x|\lambda)$ будем максимальна. Для этого возьмём случайную Модель $\{A,\ B,\ (1, 0, ..., 0)\}$ и будем изменять её по следующему правилу:

$A = (a_{ij})$; $B = (b_{ij})$;
($\alpha^l, \beta^l, \bar{\alpha}^l, \bar{\beta}^l, c^l$ - вспомогательные переменные)

$\alpha^l = 0_{q,T_l}$; $\beta^l = 0_{q,T_l}$; $\bar{\alpha}^l = 0_{q,T_l}$; $\bar{\beta}^l = 0_{q,T_l}$; $c^l = 0_{T_l}$;
for $l \in \{1, ..., L\}$: **do**
  $\alpha_1^l(i) = I_1(i) b_i(x_1^l)$; $c_1^l = \frac{1}{\sum_{i=1}^q \alpha_1^l(i)}$; $\bar{\alpha}_1^l(j) = c_1^l \alpha_1^l(j)$;
  for $t \in \{1, ..., T_l - 1\}$: **do**
    $\alpha_{t+1}^l(j) = b_j(x_{t+1}^l) \sum_{i=1}^q \bar{\alpha}_t^l(i) a_{ij}$;
    $c_{t+1}^l = \frac{1}{\sum_{i=1}^q \bar{\alpha}_{t+1}^l(j)}$;
    $\bar{\alpha}_{t+1}^l(j) = c_{t+1}^l \alpha_{t+1}^l(j)$;
  end
end
for $l \in \{1, ..., L\}$: **do**
  $\beta_T^l(i) = 1$; $\bar{\beta}_T^l(j) = c_T^l \beta_T^l(j)$;
  for $t \in \{1, ..., T_l - 1\}$: **do**
    $\beta_t^l(j) = \sum_{i=1}^q a_{ji} b_i(x_{t+1}^l) \bar{\beta}_{t+1}^l(i)$;
    $\bar{\beta}_t^l(j) = c_t^l \beta_t^l(j)$;
  end
end

$$a_{ij} = \frac{\sum_{l=1}^{L} \sum_{t=1}^{T_l - 1} \bar{\alpha} t^l(i) a_{ij} b_j(x_{t+1}^l) \bar{\beta}_{t+1}^l(i)}{\sum_{l=1}^{L} \sum_{t=1}^{T_l - 1} \bar{\alpha}_t^l(i) \bar{\beta}_{t+1}^l(i) / c_t^l}$$

$$b_j(k) = \frac{\sum_{l=1}^{L} \sum_{t=1, x_t^l = v_k}^{T_l - 1} \bar{\alpha}_t^l(i) \bar{\beta}_{t+1}^l(i) / c_t^l}{\sum_{l=1}^{L} \sum_{t=1}^{T_l - 1} \bar{\alpha}_t^l(i) \bar{\beta}_{t+1}^l(i) / c_t^l}$$

$$log(P(x|\lambda)) = -\sum_{l=1}^{L} \sum_{t=1}^{T_l} log(c_t^l)$$

Каждый раз мы получаем новые матрицы A и B такие, что вероятность $P(x|\lambda)$ становиться выше. Производя изменения по этим формулам, мы доходим до некоторого локального макси-

мума. Далее повторяем ту же операцию для других случайных матриц в поисках более высоких точек максимума.

Замечание: изменяя матрицы по описанному выше алгоритму, иногда можно получить сходимость к нулю некоторых их элементов, хотя сами формулы составлены так, что этого не может произойти. В случае компьютерной реализации, это означает, что данный элемент, достаточно близкий к нулю, будет заменён на ноль, а это в свою очередь приведёт к ошибке в вычислениях (на следующей итерации будет присутствовать деление на ноль). Потому после изменения матриц, мы должны заменить такие элементы на некоторую заранее установленную константу. Реализация замены на языке Python описана в приложении Г.

Получив Марковские Модели для каждого действия, мы, наконец, можем построить классификатор. Основу составляет подсчёт $P(x|\lambda)$ с помощью алгоритма «прохода вперёд» [3]. Частично эта процедура задействована в самом алгоритме Баума-Вэлша.

$\alpha^\sigma = 0_{q,T}$; $\bar{\alpha}^\sigma = 0_{q,T}$; $c^\sigma = 0_T$;
$\alpha_1^\sigma(i) = I_1(i) b_i^\sigma(x_1)$; $c_1^\sigma = 1/\sum_{i=1}^q \alpha_1^\sigma(i)$; $\bar{\alpha}_1^\sigma(j) = c_1^\sigma \alpha_1^\sigma(j)$;
while *movement continues* **do**
  $\alpha_{t+1}^\sigma(j) = b_j^\sigma(x_{t+1}) \sum_{i=1}^q \bar{\alpha}_t^\sigma(i) a_{ij}^\sigma$;
end
$log(P(x|\lambda^\sigma)) = -\sum_{t=1}^T log(c_t^\sigma)$;

где, σ – индекс, обозначающий принадлежность модели к определенному действию, $\sigma \in \{d_2, d_3, d_4\}$.

Как только алгоритм, построенный в 4.1., регистрирует действие, показания датчика проходят через векторную квантизацию. Затем, с помощью алгоритма «прохода вперёд» мы вычисляем вероятность того, что данное действие было сгенерировано той или иной Марковской Моделью. Вероятности пересчитываются с каждым новым показанием датчика, пока действие не закончится. Классификатор возвращает то действие, которое

с большей вероятностью было сгенерировано соответствующей Марковской Моделью.

Результат

Тестирование полученного классификатора производилось с помощью метода перекрёстной проверки для различных значений мощности алфавита k и количества состояний q. Из общей выборки (111 движений d_2, 128 движений d_3 и 115 движений d_4) случайно отбирались 90 движений каждого вида: 45 движений – для обучающей выборки, 45 – для тестовой. В тестах для $k = 13$ и $q = 3$ точность работы классификатора составляла приблизительно 95% (пример - таблица 1).

Движение \ Результат	d_2	d_3	d_4
d_2	44	-	1
d_3	-	43	2
d_4	-	-	45

Таблица 1. Пример работы классификатора

Таким образом, метод скрытых Марковских Моделей решает задачу по разделению последних трёх движений с приемлемой точностью.

Выводы

В статье рассмотрены два важных метода решения задач классификации: прямое построение модели и методы машинного обучения. Показано, что на любом этапе решении задач оба метода должны рассматриваться равноправно. Проблема отсутствия чёткой методологии решения задач классификации также остаётся открытой.

Список литературы

[1] Gernot A. Fink Markov Models for Pattern Recognition: From Theory to Applications - Berlin: Springer, 2008.

[2] Daniel Jurafsky, James H. Martin Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition - Englewood Cliffs, New Jersey: Prentice Hall, 2009.

[3] Lawrence R. Rabiner A tutorial on hidden markov models and selected applications in speech recognition. // In Proceedings of the IEEE - 1989. V. 77. - P. 257-286.

[4] Dawei Shen Some mathematics for HMM - Massachusetts Institute of Technology, 2008.

[5] Marco Klingmann Accelerometer-Based Gesture Recognition with the iPhone - Goldsmiths University of London, 2009.

[6] Juha Kela, Panu Korpipää, Jani Mäntyjärvi , Sanna Kallio, Giuseppe Savino, Luca Jozzo, Sergio Di Marca Accelerometer-based gesture control for a design environment // Personal and Ubiquitous Computing - 2006. V. 10 (5) - P. 285–299

[7] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, Michael L. Littman Activity Recognition from Accelerometer Data // Proceeding IAAI'05 Proceedings of the 17th conference on Innovative applications of artificial intelligence - 2005. V. 3 - P. 1541-1546.

[8] Thomas Schlömer, Benjamin Poppinga, Niels Henze, Susanne Boll1 Gesture Recognition with a Wii Controller // Proceedings of the 2nd international conference on Tangible and embedded interaction - 2008. - P. 11-14.

[9] Viktor Mayer-Schonberger, Kenneth Cukier: Big Data: A Revolution That Will Transform How We Live, Work and Think - London: Hachette UK, 2013.