Определение местоположения по сигналам акселерометра

Макаров М. В.

Мы рассматриваем задача определения местоположения человека по данным акселерометра телефона и другим вспомогательным данным. Уже существуют некоторые методы решения данной задачи [1]. В данной статье акцент делается на использовании дополнительной информации, например сигналов гироскопа или магнетометра, для повышения точности.

1 Введение

В данной работе рассматривается задача определения местоположения человека по данным акселерометра его телефона. Данная задача актуальна как часть более общей проблемы определения местоположения. Поскольку акселерометры энергоэффективны и не требуют для работы наличие внешних устройств, таких как спутник или радиоточка, точные методы решения этой задачи востребованы.

В силу того, что акселерометры, использующиеся в мобильных устройствах, неточны, н наивное решение поставленной задачи путём двойного интегрирования даёт путь, значительно отклоняющийся от истинной траектории. В [1] эта проблема решается использованием информации о том, где находится телефон во время перемещения человека.

Местоположение также можно определить, используя данные других датчиков, таких как магнитометр [2] и гироскоп [3].

В данной работе сравниваются различные модели для регресии скорректированных векторов скорости, для последующего определения траектории согласно схеме, используемой в [1]. Используются данные акселерометра и гироскопа.

2 Постановка задачи

Данные с датчиков представляются в виде временного ряда

$$\mathbf{s} = \{(\mathbf{c}(t), \mathbf{r}(t)) | t \in T\} \in \mathbb{R}^{6\times T} = \mathbb{X}, \qquad (*)$$

составленного из показаний акселерометра и гироскопа по трём координатам, где $T = \{t_i | i \in \mathcal{I}\}$, $\mathcal{I} = \{1, \ldots, m\}$ — множество моментов в которые проводились измерения. Аналогично, ряд истинных положений объекта имеет вид $\mathbf{y} \in \mathbb{R}^{3\times T} = \mathbb{Y}$. Обозначим через $\mathbf{X} = \{\mathbf{s}_i | i \in \mathcal{I}\}$ матрицу всех рядов выборки, а через $\mathbf{y} = \{\mathbf{y}_i | i \in \mathcal{I}\}$ — матрицу положений объекта в соответствующие моменты времени.

Предполагается, что в момент времени t_1 базис системы отсчёта объекта совпадает с системой отсчёта, относительно которой происходят измерения перемещения, погрешности измерений $\mathbf{c}_i(t), \mathbf{r}_i(t), \mathbf{b}_i(t)$ независимы и имеют нулевое матожидание, каждый элемент выборки был получен при фиксированном расположении смартфона на теле человека — в сумке, в руке, на теле или на ноге — соответственно классы $P = \{0, 1, 2, 3\}$.

Требуется найти такую модель

$$\mathbf{f} : \mathbb{X} \to \mathbb{Y}, \quad \mathbf{f}(\mathbf{s}) = \hat{\mathbf{y}}$$

что среднеквадратичная функция ошибки

$$S(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{|I|} \sum_{i\in\mathcal{I}} \sum_{t\in T} ||\mathbf{y}_i(t) - \hat{\mathbf{y}}_i(t)||^2$$

минимальна.

В данной работе рассматриваются каскадные модели следующего вида: на первом уровне модель $g : \mathbb{X} \to P, g(\mathbf{s}) = p$ определяет, к какому классу расположения относится данная траектория. После чего модель $\mathbf{f}_p(\mathbf{s}) = \hat{\mathbf{v}}$ даёт оценку истинных векторов скорости. Наконец, скорости $\tilde{\mathbf{v}}$ интегрируются для получения оценки $\hat{\mathbf{y}}$.

3 Общая структура модели

Для получения матрицы признаков X предварительно из данных удаляются высокочастотные шумы с помощью сглаживания Гаусса. Полученные линейные и угловые скорости уже преобразуются в вектор признаков.

В качестве модели рассматривается каскадная регрессия, состоящая из классификатора положения датчиков и семейства регрессоров на каждый из классов, как в [1].

По матрице признаков X классификатор определяет, какому расположению датчиков (в руке, на ноге, на поясе, в сумке) соответствует данное описание, т.е. решает задачу многоклассовой классификации с 4 классами.

После этого для каждого класса на тренировочной выборке обучается свой регрессор, который выдает скорости движения пешехода для каждого временного блока. Полученные скорости содержат ошибку, связанную с неточностями инерционных датчиков, поэтому далее для этих скоростей находится смещение (по предположению низко-частотное) из следующей задачи минимизации:

$$\min_{\{x_I^1, x^5 1_I, \dots\}} V_{bias} = \min_{\{x_I^1, x^5 1_I, \dots\}} \sum_{f \in F_2} \|v_C^F - v_R^f\| + \lambda \sum_{f \in F_1} \|x_I^f\|^2,$$

$$v_C^f = R_{SW}^f \sum_{f'=1}^{f} R_{WI}^{f'} (a_I^{f'} + x_I^{f'}),$$

где f - единица блока выборки, F - блок выборки, v_C^F - скорректированное значение скорости, v_R^f - предсказанное значение скорости, I - система координат устройства, W - глобальная система координат, S - IMU-стабилизированная система координат, R_{AB} - матрица перехода из системы координат B в систему координат A.

Также для каждого класса создается регрессор, предсказывающий угловые скорости пешехода в каждом временном блоке.

На контрольной выборке для классификатора и каждого регрессора подбираются оптимальные значения гиперпараметров.

По полученным значениям скоростей восстанавливается траектория пешехода.

Для эксперимента используется часть данных, собранных в статье, описывающей алгоритм RIDI [1]. Эти данные были получены с помощью инерционных датчиков, расположенных в смартфонах в нескольких случаях: когда смартфон располагался на поясе, в руке, на ноге или в сумке. Выборки содержат различные траектории длиной в 100 минут и частотой сигнала 200 Гц. В качестве объекта рассматривается положение в определенный момент времени i. Признаками объекта являются угловые скорости и линейные ускорения в стабилизированной системе координат датчиков в моменты времени $i - window\_size, \dots, i$, где $window\_size$ - размер окна (равен 200). Целевыми переменными являются метки классов, характеризующие то, в каком положении находился смартфон при получении определенных данных, а также скорости в данный момент времени i, которые вычисляются

через координаты пешехода и прошедшее время. По полученным данным после уточнения скоростей с помощью оптимизации функции V_{bias} строится предсказанная траектория пешехода.

Цель эксперимента: подобрать такие модели и их параметры, что предсказанная траектория пешехода будет наиболее близкой к истинной.

Формально алгоритм описывается следующим образом:

Вход: $X, Y_{class}, Y, X_{test}$

1: $initialize\ classifier\_options$
2: $classifier = Classifier(classifier\_options)$;
3: $classifier.fit(X, Y_{class})$
4: **для** $cls\ in\ classes$:
5: $\quad initialize\ regressor\_cls\_optons$
6: $\quad regressor\_cls = Regressor(regressor\_cls\_optons)$
7: $\quad regressor\_cls.fit(X[X[ind] \in cls], Y)[Y[ind] \in cls])$
8: $Y_{test-class} = classifier.predict(X_{test})$
9: **для** $cls\ in\ classes$:
10: $\quad Velocity\_cls = regressor\_cls.predict(X_{test}[Velocity\_class[ind] == cls]$
11: $\quad x_I^1, x^5 1_I, \cdots = \underset{\{x_I^1, x^5 1_I, \dots\}}{\arg\min}\ V_{bias}\_cls$
12: $\quad Velocity\_cls = R_{SW}^f \sum_{f'=1}^{f} R_{WI}^{f'}(a_I^{f'} + x_I^{f'})$
13: $\quad Trajectory\_cls$ recovery depending on $Velocity\_cls$
14: **return** $Full\_trajectory$

4 Исследование моделей

Исследования проводились на малой выборке, состоящей из 4 траекторий в обучающей и 4 траекторий в тестовой, причём данные в тестовой выборке были сняты с других людей. Они были разбиты на выборки для подзадач размером 10183 и 9317 соответственно.

Для моделей производился подбор гиперпараметров с помощью кросс-валидации по тренировочной выборке, а также измерялось их качество на тестовой выборке.

4.1 Метод ближайших соседей

В качестве первого класса моделей рассматривался метод ближайших соседей. Подбирались следующие параметры: число соседей, метрика и функция весов.

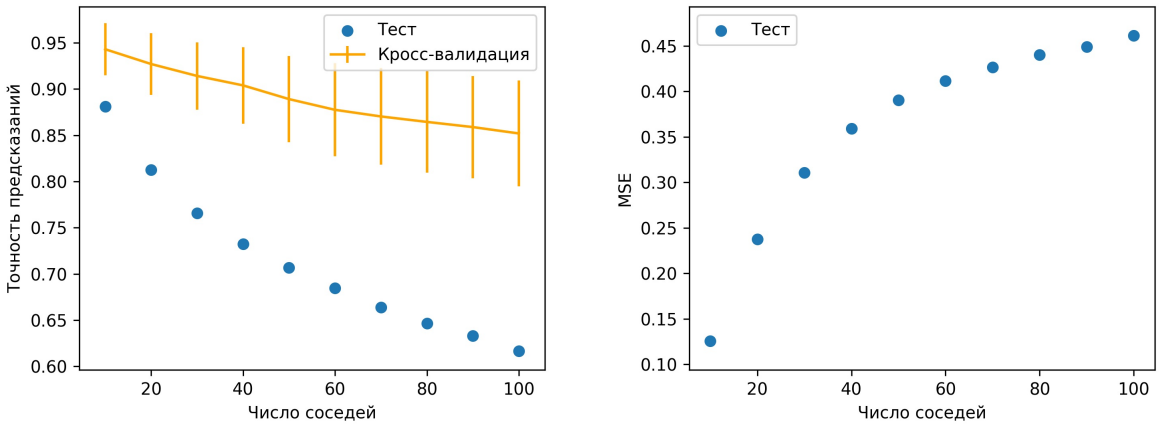


Рис. 1 Точность предсказания класса и ошибка на тесте, kNN

В итоге наилучшей метрикой оказалась $l1$, а наилучшей функцией весов оказалось расстояние.

4.2 Свёрточные нейронные сети

Вторым рассмотренным классом классификаторов были свёрточные нейронные сети. Исторически они использовались для задачи распознавания действий человека (Human Activity Recognition) [4]. Архитектура сети была выбрана, основываясь на [5] [6].

Тип	Слой	Параметры
A1	Свёрточный слой	Выходное число каналов — 12, длина ядра — 7
A2	Average pooling	Длина — 3
B	Полносвязный слой	Число нейронов — 16
C1	Softmax слой	Число классов — 4
C2	Полносвязный слой без активации	Число нейронов — 1

Рассматривались архитектуры следующего вида:

- От 1 до 3 пар слоёв A1, A2.
- От 0 до 2 слоёв B.
- Слой C1 для классификатора или слой C2 для регрессоров.

В качестве функции активации использовалась ReLU.

Каждая из них обучалась 200 эпох на обучающих данных под контролем валидационной выборки. Две наилучшие из них — с 3 свёрточными и 2 полносвязными и с 2 свёрточными и 2 полносвязными слоями. Также хорошие результаты на классификации показала архитектура с одним свёрточным и одним полносвязным слоями.

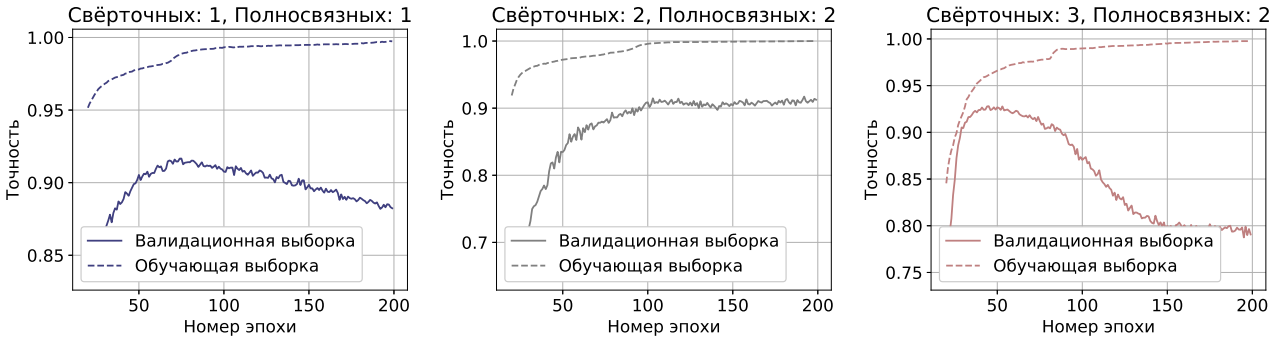


Рис. 2 Точность предсказания, CNN

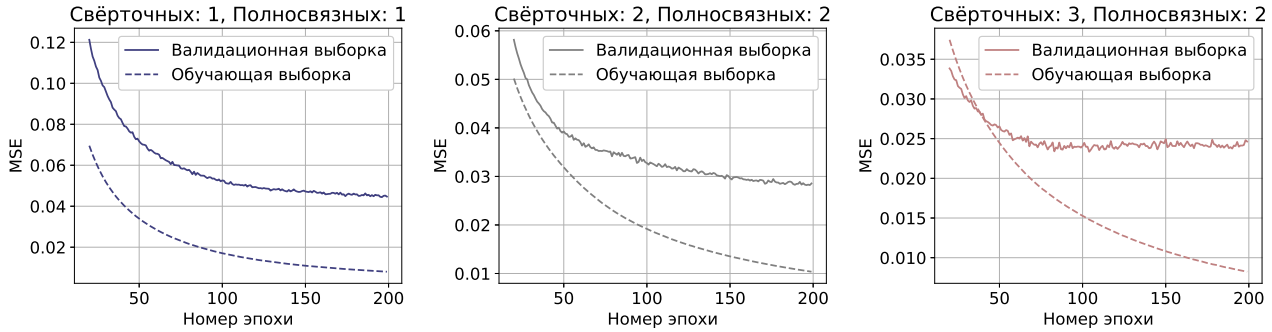


Рис. 3 Ошибка регрессии, усреднённая по классам и каналам, CNN

Наилучшую точность и наименьшую ошибку регрессии показала архитектура с 3 свёрточными и 2 полносвязными слоями — 92.8% и 0.028 соответственно.

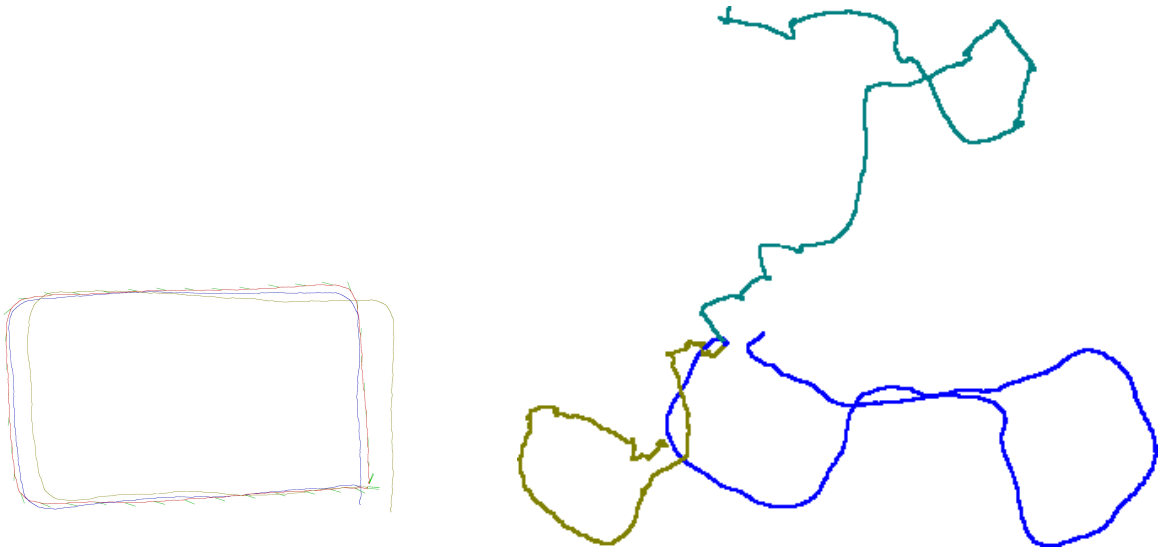
5 Увеличенная выборка

Также были проведены эксперименты на увеличенной выборке размером 49 и 25 траекторий. Они были разбиты на выборки для подзадач размером ? и ? соответственно.

Структура регрессионной свёрточной сети была усложнена, так как в результате предыдущего эксперимента выяснилось, что её разрешающей способности недостаточно. Её архитектура выглядит так:

- 3 слоя A1.
- Слой A2.
- 4 слоя B.
- Слой C2.

Также модели сравнивались с предобученным SVM-каскадом из [1].



(*а*) Траектория из тестовой выборки (*б*) Траектория из собственных измерений

Рис. 4 Красный — истинная траектория, синий — SVM, жёлтый — CNN, зелёный — KNN

Таблица 1 Точность классификации и среднеквадратичная ошибка регрессии скорости на тестовой выборке, увеличенная выборка

Модель	Точность	MSE	Размер, в мб	
			классификатор	регрессоры
SVM	0.379	0.085	30	130
kNN	0.372	0.027	260	260
CNN	0.381	0.013	0.006	0.8

Результаты можно видеть на таблице 1. Заметим, что точность моделей на тестовой выборке совпадает. Скорее всего, это указывает на недостаточное число данных. Также примечательно, что свёрточные нейронные сети способны достигнуть качества не хуже, чем SVM, при этом требуя значительно меньше памяти.

На рисунке 4 изображены примеры регрессий. По ним видно, что хотя иногда CNN работает хорошо, на некоторых траекториях он всё же уступает SVM в точности.

6 Выводы

Как и ожидалось, увеличение сложности модели может повысить качество на обоих уровнях каскада, как это видно для регрессии в сравнении SVM и свёрточной сети. Но, к сожалению, это так далёко не всегда, что хорошо демонстрирует метод ближайших соседей. С другой стороны, большая сложность модели также означает, что модель будет требовать больших вычислительных ресурсов, что будет означать её неприменимость для устройств с небольшими вычислительными мощностями.

Кроме того, среднеквадратичная функция потерь при регрессии скорости оказывается нецелесообразной для измерения качества получаемых траекторий.

Литература

[1] Hang Yan, Qi Shan, and Yasutaka Furukawa. Ridi: Robust imu double integration. *CoRR*, abs/1712.09004, 2017.

[2] W. Kang and Y. Han. Smartpdr: Smartphone-based pedestrian dead reckoning for indoor localization. *IEEE Sensors Journal*, 15(5):2906–2916, May 2015.

[3] Jian Kuang, Xiaoji Niu, and Xingeng Chen. Robust pedestrian dead reckoning based on mems-imu for smartphones. *Sensors*, 18(5), 2018.

[4] Y. Chen and Y. Xue. A deep learning approach to human activity recognition based on single accelerometer. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1488–1492, Oct 2015.

[5] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *CoRR*, abs/1809.04356, 2018.

[6] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, Feb 2017.

[7] R. Hostettler and S. Särkkä. Imu and magnetometer modeling for smartphone-based pdr. In *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8, Oct 2016.

[8] Frédéric Evennou and François Marx. Advanced integration of wifi and inertial navigation systems for indoor mobile positioning. *EURASIP J. Adv. Sig. Proc*, 2006, 2006.

[9] Patrick Robertson, Michael Angermann, and Bernhard Krach. Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors. In Sumi Helal, Hans Gellersen, and Sunny Consolvo, editors, *UbiComp 2009: Ubiquitous Computing, 11th International Conference, UbiComp 2009, Orlando, Florida, USA, September 30 - October 3, 2009, Proceedings*, ACM International Conference Proceeding Series, pages 93–96. ACM, 2009.

[10] M. Edel and E. Köppe. An advanced method for pedestrian dead reckoning using blstm-rnns. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–6, Oct 2015.