УДК 004.855

КОРРЕКЦИЯ ОШИБОК ИЗМЕРЕНИЙ РАСХОДА И ПЛОТНОСТИ КОРИОЛИСОВЫМ РАСХОДОМЕРОМ ПРИ НАЛИЧИИ ВОЗДУХА С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

О.Л. Ибряева, В.В. Барабанов

В данной статье рассмотрены проблемы создания механизма коррекции ошибки кориолисового расходомера в двухфазной среде. Рассмотрены аналогичные работы по создания нейронных сетей для коррекции ошибки с разными архитектурами и свойствами сетей. Описана совершенно новая архитектура сети в данной области и применены механизмы K-валидации (для определения ошибки сети в условиях малого объема обучающей выборки) и дропаута (для исключения переобучения сети).

Ключевые слова: кориолисовый расходомер, нейронные сети, K-валидация, дропаут.

Одним из перспективных направлений исследования работы по созданию многофазных расходомеров является попытка использования нейронных сетей для компенсации ошибок, вызванных многофазной средой. Такие попытки успешно предпринимались, например, еще в работах [1–2]. Однако с тех пор появилось множество новых методов и средств проектирования нейронных сетей, что требует переосмысления ранних результатов и дает возможность найти лучшую архитектуру сети, чем предложенные ранее.

В статье [1] используются следующие типы нейронных сетей: многослойный персептрон и сети радиальных базисных функций. Структура сети радиальных базисных функций состоит из двух слоев, первый содержит радиальные нейроны с параметрами выбранных радиальных базисных функций, второй – линейные нейроны, осуществляющие простую линейную суперпозицию базисных функций. Обучение такой сети проводится быстрее. Однако основным недостатком РБФ сетей является то, что для каждого образца обучающей выборки требуется свой нейрон. Поэтому РБФ сети менее пригодны для онлайн-реализации, но используются для быстрой оценки новых данных и выбора «внутренних» параметров расходомера, влияющих на точность его измерений. После проведенных исследований авторы статьи [1] отказались от дальнейшего использования РБФ-сети в пользу обычной персептронной сети.

Веса нейронов сети многослойного персептрона находятся в процессе обучения с учителем, когда нейронной сети многократно предъявляются образцы обучающей выборки – входы и соответствующие им выходы. В [1] использовалась обычная сеть прямого распространения с одним

скрытым слоем. Обучение сети проводилось с помощью пакета Neural Network в Matlab, с использованием метода сопряженных градиентов. Наилучшей структурой сети была признана структура 4-9-1. В качестве четырех входных параметров были выбраны: температура, коэффициент затухания, доля газа и уровень расхода. Выходным сигналом являлась ошибка измерений.

Далее эта сеть была реализована в цифровом приемопередатчике для работы в реальном времени. В результате ошибка измерений с уровня 20 % была уменьшена до уровня около 2 % .

В статье [2] авторы использовали нейронные сети при создании кориолисового расходомера, способного работать в трехфазном потоке. Используя по-прежнему сеть прямого распространения, они ввели в рассмотрение следующие входные признаки: GVF (Gas Volume Fraction) – процент газа в газо-жидкостной смеси и падение плотности dd (density drop).

dd – процентное соотношение между истинной плотностью жидкости ρ (в отсутствии воздуха) и ее наблюдаемым в эксперименте значением ρ_{obs}:

$$dd = \frac{\rho - \rho_{obs}}{\rho} \cdot 100\ \%.$$

Параметр dd, как и GVF, показывает наличие воздуха в газо-жидкостной смеси, однако в отличие от GVF, истинное значение которого недоступно в реальном времени, может быть вычислен до коррекции значений расхода и плотности.

Для работы нейронной сети используются следующие входные параметры [3]: наблюдаемое значение расхода MFR_{obs} (observed mass flow rate) и падение плотности dd. Выходом являются ошибки измерения расхода MFR_{err} и плотности D_{err}.

Мы в наших экспериментах использовали такие же входные признаки и выходные значения. Конечно, возможно усложнение данной архитектуры сети за счет введения, например, дополнительных входных признаков, таких как температура газо-жидкостной смеси, абсолютное давление на входе в массовый расходомер Кориолиса, измеренное датчиком давления и т.д. Мы оставляем это исследование на будущее.

Рис. 1 и 2 показывают зависимость ошибки измерения расхода MFR_{err} и плотности D_{err} от параметра dd (принимающего значения от 0 до 25 %) для разных уровней расхода MFR. Данные, по которым построены данные линии, были использованы для обучения нейронной сети. Как можно видеть, объем обучающей выборки составляет всего 122 эксперимента. Такой относительно небольшой объем сразу ограничивает нас в выборе архитектуры сети – она, разумеется, не должна быть излишне сложной и емкой. В противном случае сеть «запомнит» предъявляемые ей в процессе обучения данные и потеряет способность к обобщению и качественному реше-

нию данной задачи регрессии для новых входных значений, не участвовавших в процессе обучения.

Малый объем обучающей выборки приводит также к необходимости использовать метод так называемой К-кросс валидации (K fold cross validation, перекрестная проверка) [4] для оценки того, как точно будет работать на практике наша сеть.

Хорошо известно, что нейронная сеть на обучающем наборе данных выдает меньшую ошибку, чем на данных, которые она не видела в процессе обучения. Для адекватной оценки погрешности модели принято делить выборку на обучающую и тестовую и оценивать точность обученной модели на тестовых данных. Однако в случае малого объема данных обучающая и тестовая выборка могут оказаться «разнородными», неравномерно разделенными и точность на тестовых данных не будет отражать реальную точность обученной модели. Особенно эта проблема актуальна в случае, когда размер тренировочного набора невелик, или когда число параметров в модели велико. Поэтому мы не будем использовать сложную модель. А для оценки точности применим «золотой стандарт» в задачах машинного обучения – метод К-кросс валидации, суть которого заключается в следующем.

Исходный набор данных разбивается на K одинаковых по размеру блока. Из K блоков один оставляется для тестирования модели, а остающиеся K−1 блока используются как тренировочный набор. Процесс повторяется K раз, и каждый из блоков используется один раз как тестовый набор. Получаются K результатов, по одному на каждый блок, они усредняются и дают одну оценку. Все наблюдения используются и для тренировки, и для тестирования модели, и каждое наблюдение используется для тестирования в точности один раз. Часто используется кросс-валидация на 10 блоках, но каких-то определенных рекомендаций по выбору числа блоков нет. Перекрестная проверка часто не используется для оценки моделей глубокого обучения из-за больших вычислительных затрат. При K = 10 десять моделей должны быть построены и оценены, что значительно увеличивает время оценки модели.

Еще одним современным средством в конструировании нейронной сети, который мы будем использовать является дропаут (dropout, исключение), предложенный в 2014 году «отцом искусственного интеллекта» Джеффри Хинтоном и его коллегами [5].

Дропаут – метод регуляризации искусственных нейронных сетей, предназначенный для предотвращения переобучения сети. Суть метода заключается в том, что в процессе обучения выбирается слой, из которого случайным образом выбрасывается определённое количество нейронов (например 30 %), которые выключаются из дальнейших вычислений. Такой приём улучшает эффективность обучения и качество результата.

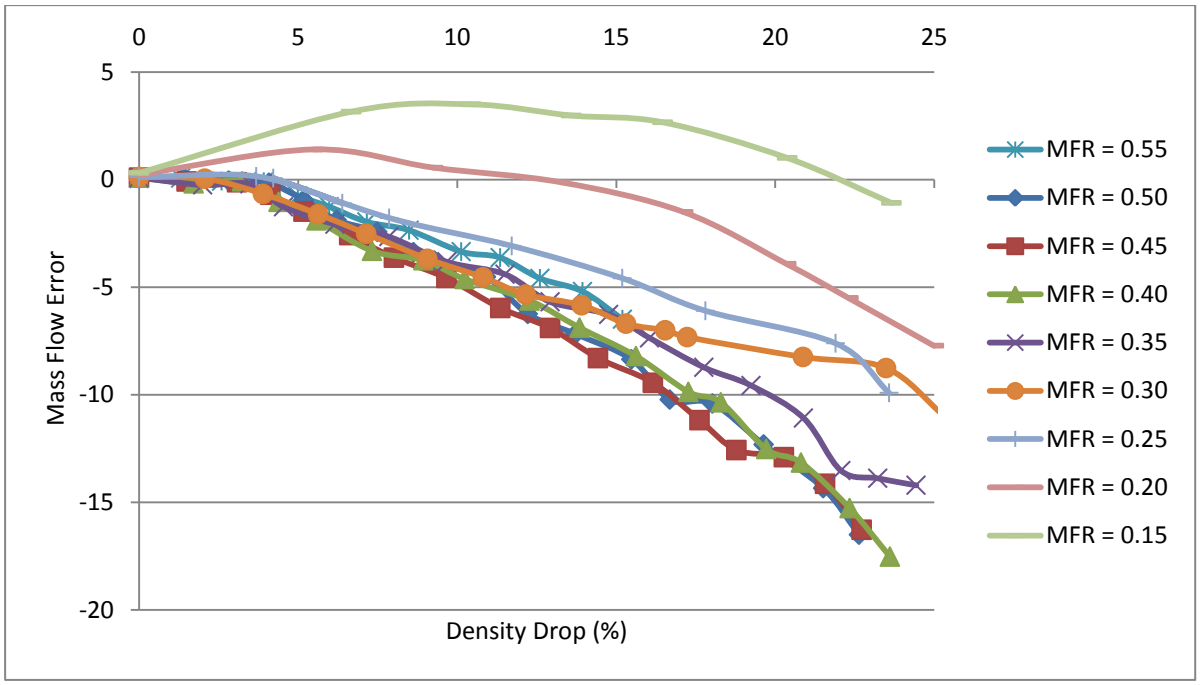

Рис. 1. Зависимость ошибки измерения расхода $\boldsymbol{MFR_{err}}$ от параметра dd

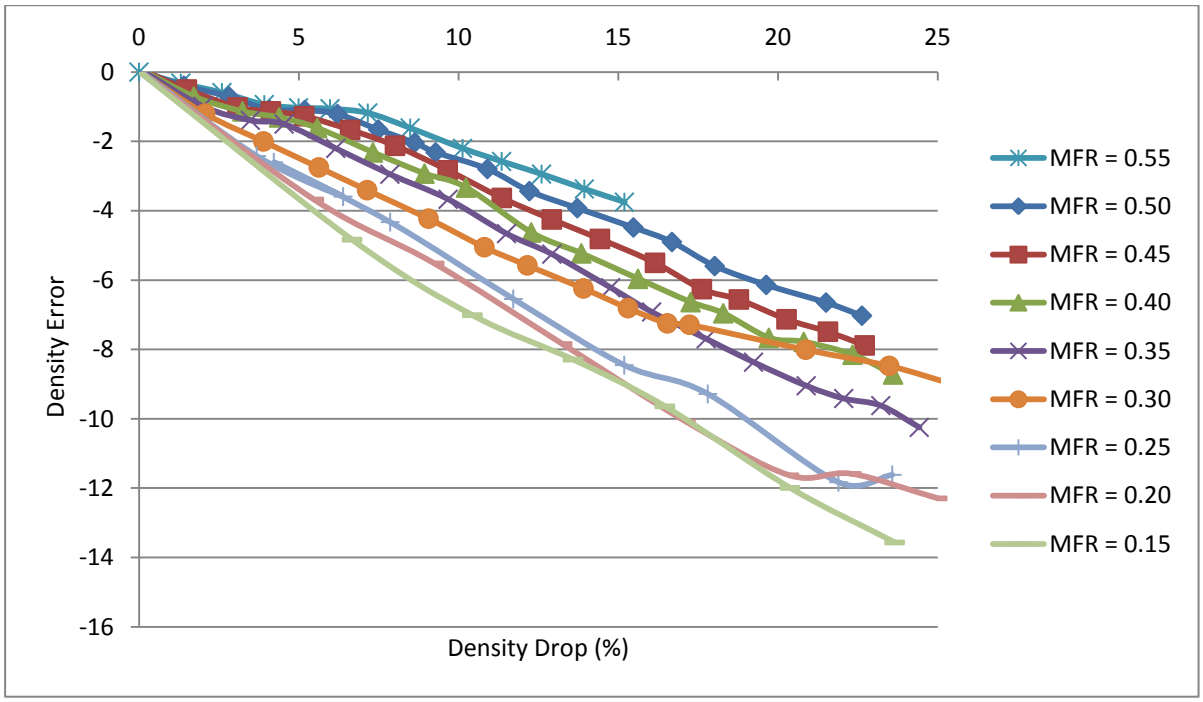

Рис. 2. Зависимость ошибки измерения плотности $\boldsymbol{D_{err}}$ от параметра dd

Обучение нейронной сети обычно производят стохастическим градиентным спуском, случайно выбирая по одному объекту из обучающей выборки. Dropout-регуляризация заключается в том, что при выборе очередного объекта изменяется структура сети: каждая вершина выбрасывается с

некоторой вероятностью p. По такой прореженной сети делается обратное распространение ошибки, для оставшихся весов делается градиентный шаг, после чего все выброшенные вершины возвращаются в нейронную сеть. Таким образом, на каждом шаге стохастического градиента мы настраиваем одну из возможных 2^N архитектур сети, где под архитектурой мы понимаем структуру связей между вершинами, а через N обозначаем суммарное число вершин. Таким образом, обученную с помощью dropout-регуляризации нейронную сеть можно рассматривать как результат усреднения 2^N сетей. Дропаут является, пожалуй, самым распространенным современным приемом регуляризации нейронной сети.

И наконец, последнее, на чем мы бы хотели заострить внимание перед переходом непосредственно к обучению сетей, это выбор *функции активации* (activation function), которая привносит в сеть **нелинейность.** В последние годы большую популярность приобрела функция активации под названием «выпрямитель» (rectifier, по аналогии с однополупериодным выпрямителем в электротехнике). Нейроны с данной функцией активации называются ReLU (rectified linear unit). ReLU имеет следующую формулу:

$$f(x) = max(0, x),$$

и реализует простой пороговый переход в нуле. В отличии от таких «классических» функций активации, как сигмоида и гиперболический тангенс, он не подвержен проблеме «насыщения». Кроме того, вычисление сигмоиды и гиперболического тангенса требует выполнения ресурсоемких операций, таких как возведение в степень, в то время как ReLU может быть реализован с помощью простого порогового преобразования матрицы активаций в нуле. Эти свойства обуславливают существенное повышение скорости сходимости стохастического градиентного спуска (в некоторых случаях до 6 раз [6]) по сравнению с сигмоидой и гиперболическим тангенсом.

Перейдем теперь к обучению нейронной сети. Эксперименты проводились с помощью высокоуровневой библиотеки Keras для работы с нейронными сетями, на языке Python. Пример кода, задающего модель нейронной сети, приведен на рис. 3. Здесь реализована сеть прямого распространения с одним скрытым слоем из 32 нейронов и дропаутом с p=0,5. В качестве функции потерь используется MSE (mean squared error) – сумма квадратов отклонений правильных значений входа сети от предсказанных ею. В качестве метрики (величины, характеризующей качество обучения) мы выбрали MAE (mean absolute error) – модуль разности между правильным и полученным сетью значением. Наш выбор является довольно стандартным при решении задач регрессии. В качестве функции активации используется ReLU. В качестве «оптимизатора», т.е. метода обучения используется современный вариант градиентного спуска Adam [7].

Сеть имеет два входных параметра и один выходной. Мы будем компактно задавать архитектуру такой сети в виде 2-32(ReLU)-D(0.5)-1.

```
def build_model():
    activate_function = 'relu'
    model = Sequential()
    model.add(Dense(32, activation=activate_function,
                    input_shape=(Data_Set_X.shape[1],)))
    model.add(Dropout(0.5))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model
```

Рис. 3. Пример модели нейронной сети в Keras

Точность работы данной модели оценивалась методом К-кросс валидации с параметром K=5. На рис. 4 показана точность работы сети, характеризуемая параметром MAE, вычисленным на проверочном наборе данных (одна пятая от общего объема данных). Пять линий соответствуют пяти блокам метода К-кросс валидации. Небольшие искажения на концах вызваны используемым нами сглаживанием данных линий по 10 точкам. Обучение сети проводилось в течение 500 эпох. В данном примере корректировалась ошибка измерения расхода MFR.

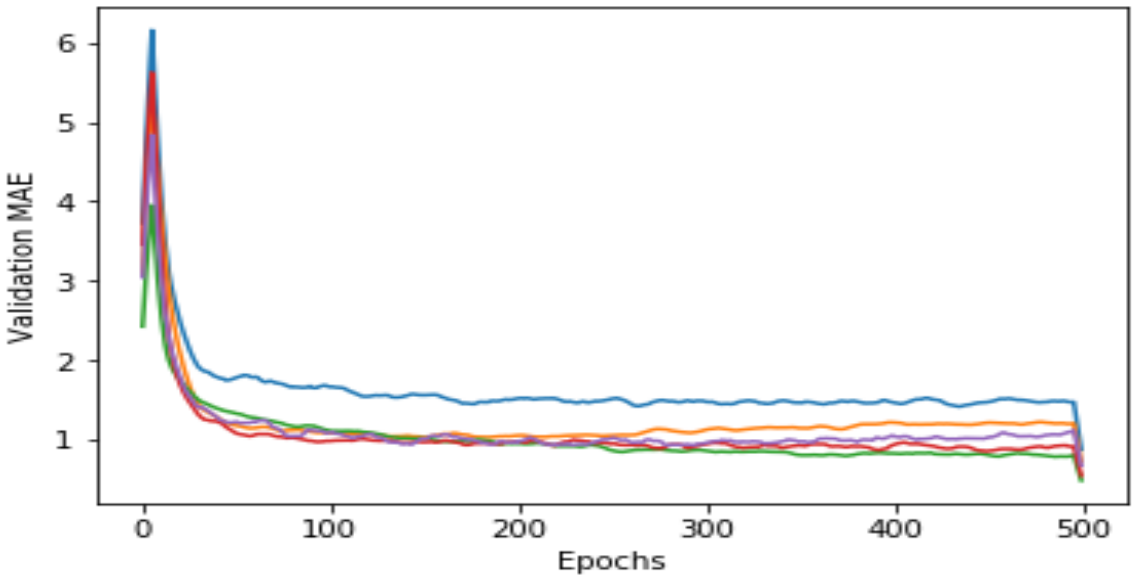

Рис. 4. Ошибка сети на проверочном наборе данных при K=5

Как можно видеть, ошибка MAE сначала снижается с ростом числа эпох обучения, но в некоторых из пяти экспериментах начинает возрастать при дальнейшем росте числа эпох обучения, что означает, что наступило переобучение сети. (Это хорошо заметно, например, для оранжевой линии, но на самом деле явление переобучения наступает здесь в 3 случаях из 5.) Выберем в качестве оптимального числа эпох обучения – 170 эпох, когда

сеть еще не склонна к переобучению, но уже дает хорошую оценку точности модели. В данном случае оценка точности работы сети, даваемая методом перекрестной проверки составила 1,07 % – это наша ошибка в определении расхода MFR.

Мы провели ряд аналогичных экспериментов для других структур нейронной сети. Архитектура сети и соответствующие ошибки представлены в табл. 1. Как можно видеть, описанная выше структура сети является наилучшей из всех представленных в этой таблице.

Таблица 1

Выбор сети для коррекции ошибки измерения расхода MFR

№	Структура сети	Ошибка сети, %
1	2-32(ReLU)-D(0.5)-1	1,07
2	2-64(ReLU)-D(0.5)-32(ReLU)-D(0.5)-16(ReLU)-D(0.5)-1	1,22
3	2-32(ReLU)-D(0.5)-16(ReLU)-D(0.5)-1	1,24
4	2-64(sigmoid)-D(0.5)-32(sigmoid)-D(0.5)-16(sigmoid)-D(0.5)-1	1,93
5	2-32(linear)-D(0.5)-32(linear)-D(0.5)-16(linear)-D(0.5)-1	2,01
6	2-32(SeLU)-D(0.5)-1	1,42
7	2-32(SeLU)-D(0.5)-32(SeLU)-D(0.5)-16(SeLU)-D(0.5)-1	1,30

Для создания сети, корректирующей ошибку измерения плотности, мы использовали тот же подход. Варианты архитектуры сети с соответствующими погрешностями приведены в табл. 2. Как и в случае с коррекцией расхода, оптимальной оказывается архитектура 2-32(ReLU)-D(0.5)-1. Ошибка измерения плотности составляет всего 0,37 %.

Таблица 2

Выбор сети для коррекции ошибки измерения плотности

№	Структура сети	Ошибка сети, %
1	2-32(ReLU)-D(0.5)-1	0,37
2	2-64(ReLU)-D(0.5)-32(ReLU)-D(0.5)-16(ReLU)-D(0.5)-1	0,80
3	2-32(ReLU)-D(0.5)-16(ReLU)-D(0.5)-1	0,69
4	2-64(sigmoid)-D(0.5)-32(sigmoid)-D(0.5)-16(sigmoid)-D(0.5)-1	0,70
5	2-32(linear)-D(0.5)-32(linear)-D(0.5)-16(linear)-D(0.5)-1	0,68
6	2-32(SeLU)-D(0.5)-1	0,56
7	2-32(SeLU)-D(0.5)-32(SeLU)-D(0.5)-16(SeLU)-D(0.5)-1	0,52

Таким образом, нейронные сети хорошо проявили себя в решении задачи уменьшения погрешности измерений. Достоинством такого подхода является отсутствие необходимости строить сложные модели устройства для

получения алгоритмов обработки. Однако это, в свою очередь, влечет за собой отсутствие гарантий, что расходомер, настроенный на какую-то определенную среду, продемонстрирует похожие результаты в других условиях.

Библиографический список

1. A neural network to correct mass flow errors caused by two-phase flow in a digital coriolis mass flowmeter / R.P. Liu et al. // Flow Measurement and Instrumentation. – 2001. – Vol. 12, № 1. – P. 53–63.

2. Coriolis mass flow metering for three-phase flow: A case study / M.P. Henry, M. Tombs, M. Zamora, F. Zhou // Flow Measurement and Instrumentation. – 2013. – Vol. 30, April. – P. 112–122.

3. Multiphase Coriolis Flowmeter / Tombs M.S. et al. – US Patent 7, 188, 534. 2007.

4. Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection / R. Kohavi // Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. San Mateo, 1995. – Vol. 2, № 12. – P. 1137–1143.

5. Dropout: A Simple Way to Prevent Neural Networks from Overfitting / G. Hinton et al. // Journal of Machine Learning Research. – 2014. – Vol. 15. – P. 1929−1958.

6. Krizhevsky, A. Imagenet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. Hinton // Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS. – 2012. – P. 1097–1105.

7. Diederik, P. Kihgma. Adam: A method for Stochastic Optimization, 2014 / Diederik P. Kihgma, Jimmy Ba // Proceedings of the 3rd International Conference on Learning Representations (ICLR).

К содержанию