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 компьютерный наук
Кафедра алгебры и дискретной математики

Применение нейронных сетей для калибровки оборудования на примере двух задач робототехники.

Выпускная квалификационная
работа на степень бакалавра
по направлению
Математика и компьютерные науки
студента группы МК-410502
Штех Геннадия Петровича

Научный руководитель
кандидат физико-математических наук
Окуловский Юрий Сергеевич

Допустить к защите:

_____

«_____» _____ 2015 г.

Екатеринбург
2015

Оглавление

Глава 1

О чем диплом?

Не редко при конструировнии подвижных платформ применяются двигатели, управляемые постоянным током. Характерной особенностью этих двигателей является их дешевизна и долговечность. Скорость вращения вала двигателя постоянного тока обычно регистрируется с помощью дополнительных устройств. В рассмотренном случае это были энкодеры. Не будем вдаваться в принцип их работы, стоит отметить лишь тот факт, что для текущей задачи их точности хватает с запасом. Главная сложность при их использовании – это сложность управления непосредственно скоростью вращения вала. Потому что фактически управлять можно только напряжением на контактах двигателя, что хорошо связано только с моментом силы на валу. Однако при использовании двигателей для перемещения необходимо хорошо контролировать именно скорость. Поэтому остро стоит проблема определения подходящего сигнала для достижения необходимой скорости вращения.

Итак задачу можно формализовать следующим образом: нужно построить функцию, по желаемой скорости и по, возможно, необходимым дополнительным параметрам, вычисляющую напряжение на контактах двигателя, приводящее скорость к желаемой. Для простоты будем называть напряжение на контактах двигателей сигналом.

Поскольку есть возможность собрать экспериментальные данные об искомой зависимости, подобную задачу восстановления неизвестной функции возможно решить методами регрессии. К качестве используемого метода выберем нейронную сеть. А именно – простую многослойную нейронную сеть, обучающуюся с учителем, биполярная сигмоида в качестве функции актива-

ции.

Глава 2

Подробные сведения об используемой нейронной сети

Рассматривать обучение нейронной сети будем как задачу оптимизации функции нескольких переменных. Для этого введем некоторые определения:

2.1 Постановка задачи минимизации

Дано:

$\mathcal{X} = (X_1, \ldots, X_k)$	входные вектора, $X_i \in \mathbb{R}^n$
$\mathcal{A} = (A_1, \ldots, A_k)$	правильные выходные вектора, $A_i \in \mathbb{R}^m$
$(\mathcal{X}, \mathcal{A})$	обучающая выборка
W	вектор весов нейронной сети
$N(W, X)$	функция, соответствующая нейронной сети
$Y = N(W, X)$	ответ нейронной сети, $Y \in \mathbb{R}^m$
$D(Y, A) = \sum_{j=1}^{m} (Y[j] - A[j])^2$	функция ошибки
$D_i(Y) = D(Y, A_i)$	функция ошибки на i-ом примере
$E_i(W) = D_i(N(W, X_i))$	ошибка сети на i-ом примере
$E(W) = \sum_{i=1}^{k} E_i(W)$	ошибка сети на всей обучающей выборке

Найти:

вектор W такой, что $E(W) \to \min$ (обучение на всей выборке)

вектор W такой, что $E_i(W) \to \min$ (обучение на одном примере)

Обучение нейронной сети свелось к минимизации вектор-функции $E_i(W)$.

2.2 Одномерный случай градиентного спуска

Для начала рассмотрим минимизацию функции одной переменной алгоритмом градиентного спуска, затем перейдем к минимизации функции многих переменных.

Начнем с того, что случайным образом выберем точку. Изучим значение производной в этой точке. Напомним, что производная функции в точке — это число, которое показывает нам, возрастает функция или убывает, то есть если функция возрастает, то производная положительная, если убывает — отрицательная. Из выбранной точки сделаем шаг, равный значению производной, в направлении, обратном направлению значения производной в данной точке, получим следующую точку, для которой проделаем все в точности то же, что и для выбранной изначальной точки. На каждом таком шаге значение функции будет убывать, а значит, в конце она достигнет минимума. Таким образом получаем **алгоритм градиентного спуска:**

1. Инициализировать x_1 случайным значением из $\mathbb{R}$
2. $i := 1$, i — номер итерации
3. $x_{i+1} = x_i - \varepsilon f'(x_i)$, $-\varepsilon f'(x_i)$ — направление обратной производной, $-\varepsilon$ обеспечивает малый шаг
4. $i++$

5. if $f(x_i) - f(x_{i+1}) > c$ goto 3

2.3 Многомерный случай градиентного спуска

В случае нескольких переменных шаги будем делать в направлении, обратном направлению градиента, который является неким аналогом производной для многомерного случая. Пусть $f(x_1, \ldots, x_n)$ — функция $n-$переменных, $f(x_1, \ldots, x_n)$:$\mathbb{R}^n \to \mathbb{R}$. Частная производная по $i-$ой переменной:

$$\frac{\partial f}{\partial x_i}(x_1, \ldots, x_n) =$$

$$\lim_{\varepsilon \to 0}[f(x_1, \ldots, x_i + \varepsilon, \ldots, x_n) - -f(x_1, x_2, \ldots, x_i, \ldots, x_n)]/\varepsilon$$

Частная производная по $i-$ой переменной — это тоже функция $\frac{\partial f}{\partial x_i} : \mathbb{R}^n \to \mathbb{R}$.

Градиент функции:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)$$

$$\nabla f : \mathbb{R}^n \to \mathbb{R}^n$$

Производная сложной функции:

$$f(x_1, \ldots, x_n)x_i = x_i(y_1, \ldots, y_m)(y_1, \ldots, y_m) = f(x_1(y_1, \ldots, y_m), \ldots, x_n(y_1, \ldots, y_m))\frac{\partial f}{\partial y_i}$$

Алгоритм градиентного спуска для многомерного случая:

1. Инициализировать W_1 случайным значением из $\mathbb{R}^n$
2. $i := 1$
3. $W_{i+1} = W_i - \varepsilon \nabla f(W_i)$

4. $i++$

5. if $f(W_i) - f(W_{i+1}) > c$ goto 3

Посчитаем градиент функции ошибки, с помощью которого мы минимизируем функцию ошибки и тем самым подберем те веса, на которых она минимальна, а минимальна она тогда, когда выход практически совпадает с правильным ответом, то есть подберем веса так, чтобы нейронная сеть научилась давать выход, совпадающий с ответом.

2.4 Обратное распространение ошибки

Научимся считать градиент ошибки по весам нейронной сети в случае, если она состоит из нескольких слоев. Этот алгоритм называется алгоритмом обратного распространения ошибки, который обеспечил возможность практического применения нейронных сетей, так как нейронные сети из одного слоя могут решать маленькое количество задач, а из нескольких слоев — большое. Алгоритм градиентного спуска — алгоритм минимизации любой произвольной функции с помощью градиента, это численный алгоритм, который использует значение градиента, алгоритм обратного распространения ошибки — алгоритм вычисления градиента для того конкретного случая, когда функция является нейронной сетью.

Функция ошибки в общем случае:

$$D_k(y_1, ..., y_n) = (y_1 - a_1)^2 + ... + (y_n - a_n)^2.$$

Производная функции ошибки по y_i:

$$\frac{\partial D_k}{\partial y_i} = 2(y_i - a_i).$$

Взвешанная сумма $i-$ого нейрона:

$$S_i = \sum_{j=0}^{m} x_j w_{ji}.$$

Функция активации соответствующего S_i:

$$y_i = f(S_i).$$

Производная функции активации по x_j:

$$\frac{\partial y_i}{\partial x_j} = f'(S_i) w_{ji}.$$

Производная ошибки по входам нейронной сети:

$$\frac{\partial D_k}{\partial x_i} = \sum_{i=0}^{n} \frac{\partial D_k}{\partial y_i} \frac{\partial y_i}{\partial x_j} = 2 \sum_{i=0}^{n} (y_i - a_i) f'(S_i) w_{ji}$$

Функция вычисляется на основании своих переменных, каждая из которых вносит определенный вклад в эту функцию, которая и является частной производной по этой переменной.

Глава 3

Описание прямого решения задачи с применением нейронной сети

3.1 Модель

За входной вектор возьмем желаемую скорость(Spd), за выходной вектор возьмём сигнал(Sig), приводящий к этой скорости.

$$Sig \in \mathbb{SIG}; Spd \in \mathbb{SPD}$$

$$\mathbb{SIG} = \{Sig \in \mathbb{N} | -256 < Sig < 256\}; \mathbb{SPD} = \mathbb{R}$$

3.2 Описание сценария для сбора экспериментальных данных

Соберём экспериментальные данные по следующему сценарию: будем произвольным образом менять сигнал (Sig_l)(l – значит last, прошлый сигнал), ждать установления скорости вращения(Spd_c)(c – значит current, текущая скорость), записывать пару (Sig_l, Spd_c) в экспериментальную выборку и вновь менять сигнал.

Фактически мы получили выборку отображения из пространства сигналов в пространство скоростей.

$$Sig \in \mathbb{SIG}; Spd \in \mathbb{SPD}$$

$$F(Sig_l) = Spd_c$$

Но нам нужна обратная функция, по желаемой скорости $(Spd_n)(n$ – next, следующая) возвращающая необходимый текущий сигнал (Sig_c).

$$Sig \in \mathbb{SIG}; Spd \in \mathbb{SPD}$$

$$F^{-1}(Spd_n) = Sig_c$$

Чтобы регрессировать обратную функцию, надо лишь перевернуть пару так, как там нужно.

$$(Sig_l, Spd_c) \to (Spd_c, Sig_l)$$

В том числе, конечно, необходимо проверить наличие обратной данной функции: исходная функция должна быть монотонной. Монотонность функции очевидна из графика, на котором отображены экспериментальные данные в виде точек.

КАРТИНКА С ДАННЫМИ Особенность в районе нуля обусловлена наличием в системе трения и чтобы "сорвать"систему из состояния покоя требуется достаточно высокий момент вращения, который требует достаточно высокого сигнала на двигателях.

3.3 Результат подхода

КАРТИНКА РЕЗУЛЬТИРУЮЩЕЙ ЛИНИИ На практике такой подход показывает себя плохо: поскольку сценарий обучения не соответствует сценарию использования. При управлении платформой скорость двигателей меняется достаточно часто(до двадцати раз в секунду) в достаточно широком диапазоне. Но функция регрессии спроектирована так, что желаемая скорость

достигается через некоторое неопределённое время после выставления сигнала. Экспериментально установлено, что в среднем это несколько секунд. При динамическом управлении платформой это неприемлемый результат. Особенно заметна эта особенность при торможении.

КАРТИНКА ЛИНИИ ТОРМОЖЕНИЯ Более того, видно, что особенность вокруг нуля нейронная сеть не выучила. Эту особенность удалось выучить только разбиением пространства на 2: $Sig > 0$ и $Sig <= 0$ с независимым обучением двух сетей на этих пространствах.

3.4 Попытки модификации подхода к формированию экспериментальной выборки

Для приведения в соответствие сценария использования и сценария обучения была сделана модификация сценария сбора данных. Изменение: не ждать установления скорости вала и добавлять в выборку все пары (Sig_l, Spd_c). Этот путь тоже не приводит к нужному результату, поскольку выборка становится зашумлённой и обилие информации нейронная сеть воспринимает как шум. В результате использование этой функции регрессии для управления двигателем ни в каких сценариях не даёт приемлимого результата. **КАРТИНКА ТУЧИ ТОЧЕК ХАОС КРОВЬ КИШКИ!!!11**

3.5 Вывод

Прямой подход не даёт пригодного к использованию результата. Главным образом из-за того, что сеть не имеет достаточной информации для выдачи сигнала, подходящего к текущей ситуации. Например ситуации ускорения, торможения и равномерного движения требуют различного подхода к фор-

мированию сигнала: завышение, занижения и непосредственное соотвествие. В том числе у сети нет временных рамок достижения скорости по сигналу, что даёт непредсказуемое время приведения системы в необходимое состояние. Таким образом мы получили список недостатков:

- Отсутствие временных рамок
- Отсутствие контекста применения сигнала
  - Режим езды: ускорение, торможение, равномерное движение
  - Предыдущий выставленный сигнал индуцирует в катушке двигателя ток, который тоже нужно преодолеть
- Сложная особенность в районе нуля

Все перечисленные проблемы решаются рассмотрением двигателя в терминах управляемых систем. Для управления двигателем в терминах системы введём понятие регулятора и цикла регуляции.

Глава 4

Инверсный нейрорегулятор

4.1 Описание принципа устройства регулятора

Регулятор – это сущность, которая способна вычислять управляющее воздействие для поддержания управляемой системы в указанном стостоянии. Мы будем использовать термин регулятор в следующем ключе: регулятор – это функция, принимающая три вектора:

1. $\mathbb{S}_n, \mathbb{S}_n \in \mathbb{STATES}$, вектор, описывающий состояние, которое необходимо поддерживать
2. $\mathbb{S}_c, \mathbb{S}_c \in \mathbb{STATES}$, вектор, описывающий текущее состояние управляемой системы
3. $\mathbb{C}_l, \mathbb{C}_l \in \mathbb{CONTROLS}$, вектор, описывающий прошлые управляющие воздействия, оказанные на систему

и возвращающая управляющее воздействие, которое приближает(не отдаляет) систему от целевого состояния $\mathbb{C}_c, \mathbb{C}_c \in \mathbb{CONTROLS}$.

В результате получим отображение, которое будем называть функцией регулятора.

$$F : \mathbb{STATES} \times \mathbb{STATES} \times \mathbb{CONTROLS} \rightarrow \mathbb{CONTROLS}$$

$$F(\mathbb{S}_n, \mathbb{S}_c, \mathbb{C}_l) = \mathbb{C}_c$$

4.2 Описание принципа устройства цикла регуляции

Предположим, у нас есть функция-регулятор. Цикл регуляции является бесконечным(не имеет выхода), каждый установленный промежуток времени δt снимаются показания состояния управляемой системы $\mathbb{S}_c$, формируется входная матрица для функции-регулятора $(\mathbb{S}_n, \mathbb{S}_c, \mathbb{C}_l)$, передаётся функции регулятору $F(\mathbb{S}_n, \mathbb{S}_c, \mathbb{C}_l)$, получаем управляющее воздействие $\mathbb{C}_c$, применяем его к управляемой системе, продолжаем собирать данные о состоянии управляемой системы, пока через δt не нужно будет опять начинать цикл.

Важной здесь является идея периодической корректировки управляющего воздействия: даже если состояние, в котором нужно поддерживать управляемую систему не изменится(не поступит новых команд), могут изменится параметры внешней среды, которые, если оставить их без реакции, могут сместить управляемую систему в нежелательное состояние. В том числе, периодичность полезна в том случае, если за одно управляющее воздействие невозможно привести систему в желаемое состояние (например из-за ограничений в максимальном ускорении, или из-за принципиальной неспособности контролируемого объекта менять состояния с требуемой скоростью). Как раз второе преимущество мы и будем эксплуатировать в дальнейшем.

4.3 Модель данных для случая управления двигателем

Зададим пространства состояний и управляющих воздействий для рассматриваемого случая.

$$\mathbb{CONTROLS} = \mathbb{SIG};\ \mathbb{STATES} = \mathbb{SPD}$$

То есть функция регулятор принимает вектор-строку в качестве аргумента и возвращает скаляр в качестве результата.

$$F : \mathbb{SPD} \times \mathbb{SPD} \times \mathbb{SIG} \rightarrow \mathbb{SIG}$$

$$F(Spd_n, Spd_c, Sig_l) = Sig_c$$

4.4 Нейронная сеть в качестве функции-регулятора

Для построения цикла регуляции нам нужна функция-регулятор. Применим ту же нейронную сеть для регрессии. Для проведения регрессии нужно собрать экспериментальную выборку...

Для улучшения качества обучения применили фичи: лавина, замыкание по нулю, среднее отклонение.

4.5 Результат

Штука хорошо тормозит, штука плавно разгоняется, штука наращивает мощность, чтобы что-то переехать. То есть ведёт себя как приличный регулятор.

4.6 Вывод

Параметры сети не особо имеют значение. Фичи решают.

Глава 5

Калибровка инфракрасных дальномеров

5.1 Постановка задачи

5.2 Описание реальных характеристик дальномеров

5.3 Модель данных

5.4 Процесс обучения сети

5.5 Сглаживание оригинала

5.6 Результат

5.7 Вывод

В голову пришла мысль о том, что чтобы управлять динамической системой недостаточно просто построить регрессионную модель её отзывов. Оказалось, что наш двигатель – это классическая динамическая система, требующая построения регулятора для работы. Тут мы малясь загрустили, ибо ПИД-регулятор если брать обобщенный, то там объебаться можно с коеффициентами, а если по уму его строить, то е нас такого ума нет.

Ее мы взяли нейронную сеть и попробовали решить проблемы как умеем: за-

пихали входной вектор(прошлый сигнал, прошлая скорость, текущий сигнал), взяли выходной(текущая скорость), собрали данных по сценарию(сценарий сцукорегулятора). Обучили сеть, а она сука поехала и стали мы счастливые очень. Фактически нейронная сеть стала инверсным нейрорегулятором, а не просто регрессионной калибровкой.

Глава 6

Литература

1. Омату С., Халид М. Юсоф Р. *Нейроуправление и его приложения* // Издательство "ИПРЖР", 2000.

2. Галушкин А.И. *Нейронные сети. Основы теории* // Издательство "Горячая Линия - Телеком", 2012.

3. Хайкин С. *Нейронные сети. Полный курс* // Издательство "Вильямс", 2006.

4. Уоссермен Ф. *Нейрокомпьютерная техника: Теория и практика* // Издательство "Мир", 2006.