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

заведующий лабораторией
доктор КАКИХ?! 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.

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 активации.

Глава 2

Подробные сведения об используемой нейронной сети

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

Постановка задачи

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 $E_i(W)$. Мы должны минимизировать значение функции, подобрав аргумент так, чтобы значение стало минимальным.

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

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:

$$f(x_1, \ldots, x_n)$$

$$x_i = x_i(y_1, \ldots, y_m)$$

$$f(y_1, \ldots, y_m) = f(x_1(y_1, \ldots, y_m), \ldots, x_n(y_1, \ldots, y_m))$$

$$\frac{\partial f}{\partial y_i} = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j} \frac{\partial x_j}{\partial y_j}$$

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

Обратное распространение ошибки

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

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

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

$$y_i = f(S_i)$$

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

$$E_k(W) = D_k(y_1(w_{01}, \ldots, w_{mn}), \ldots, y_n(w_{0n}, \ldots, w_{mn}))$$

$$\frac{\partial E_k}{w_{ji}} = \sum_{l=1}^{n} \frac{\partial D_k}{y_l} \frac{\partial y_l}{w_{ji}} = 2(y_i - a_i)f'(S_i)x_j$$

Глава 3

Описание решения задачи с применением нейронной сети

Ну мы взяли нейронную сеть и попробовали решить проблемы калибровки как умеем: запихали как-то входной вектор(желаемая скорость), взяли как-то выходной(какой сигнал к этой скорости приводит), обучили сеть и обосрались: то, как мы калибровали и то, как мы использовали – это разные сценарии. Основная проблема оказалась вот в чём: хуйню засунул – хуйня выпала, мы собирали данные по следующему сценарию: выставляли сигнал, ждали, пока скорость установится, и в экспериментальную выборку добавляли пару (текущий сигнал, текущая установившаяся скорость). Понятно, что в динамике это показывало ужасающий результат. Наиболее заметен он при торможении, кривая скорости падала недостаточно быстро и робот очень сильно переезжал.

Понятно, кажется, для начала нужно было собрать данные, соответствующие

хотя бы сценарию использования. Данные стали собирать так: выставляли сигнал и все скорости, которые регистрировали, записывали в выборку обучения. К сожалению, это ни к чему не привело. Это очевидно, если взглянуть на получившуюся картинку. Множество различных данных для сети оказалось лишь шумом и улучшения не дало. (КАРТИНКА!)

В целом жизнь устроена так: чтобы управлять штуками, нужно постоянно контроллировать, как они отреагировали на прошлое воздействие. Иначе штуки будут делать хуету. Фактически, за штуками нужно следить постоянно, с какой-то периодичностью. Поскольку жизнь вокруг штук тоже может поменяться и прежде годное управляющее воздействие прямо сейчас мчит вас в пропасть. Весь этот процесс называется циклом регуляции. Фактически, устроен он так: мы получаем обратную связь от штуки, вспоминаем, в каком состоянии нам необходимо её поддерживать и на основе обратной связи и наших грязных желаний вырабатываем управление. Возможно, даже учитываем наши предыдущие управляющие воздействия.

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: запихали 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.