МОДЕЛИРОВАНИЕ ИНТУИЦИИ ПОСРЕДСТВОМ НЕЙРОННЫХ СЕТЕЙ

MODELING INTUITION BY NEURAL NETWORKS

Sun Xilong

Summary. At present, neural networks are an exact means of automation, however, such accuracy depends on the degree of "training" of the network and the degree of training of its "intuition". In this article, we will consider modeling the "intuition" of a neural network in order to create networks with a high degree of training for quick decision-making regarding various phenomena.

Keywords: neural networks, intuition, machine learning, programming.

Сунь Силун
Аспирант, Кубанский государственный университет
sasha.7s@yandex.ru

Аннотация. В настоящее время нейронные сети представляют собой точное средство автоматизации, однако такая точность зависит от степени «обученности» сети и степени обученности ее «интуиции». В настоящей статье будет рассмотрено моделирование «интуиции» нейронной сети с целью создания высокой степени обученности сети для быстрого принятия решений относительно того или иного явления.

Ключевые слова: нейронные сети, интуиция, машинное обучение, программирование.

Искусственные нейронные сети (ИНС) имитируют мозг человека при обработке входных сигналов и преобразуют их в выходные сигналы. Она обеспечивает мощный алгоритм моделирования, который учитывает нелинейность между характеристическими переменными и выходными сигналами. ИНС — это своего рода метод непараметрического моделирования, который подходит для сложного явления, когда исследователи не знают основных функций [6]. Другими словами, ИНС может извлекать уроки из данных без конкретных предположений относительно функций.

Основная идея всех нейронных сетей заключается в следующем: каждый нейрон в нейронной сети принимает *решение*. Необходимо понять, как нейронная система принимает решение, тем самым развивая «искусственную интуицию».

Допустим, вы пытаетесь решить, стоит ли носить шляпу сегодня. Есть ряд факторов, которые повлияют на ваше решение, и, возможно, наиболее важными из них являются:

- Солнечно ли на улице?
- Нужно ли носить шляпу?
- Подойдет ли шляпа моему костюму?

Для простоты предположим, что это единственные три фактора, которые учитываются при принятии этого решения. Забудем на секунду о нейронных сетях и просто попробуем создать «программу, принимающую решение», чтобы помочь нам ответить на этот вопрос.

Во-первых, становится понятно, что каждый вопрос имеет определенный уровень важности, и поэтому нам нужно использовать эту относительную важность каждого вопроса вместе с соответствующим ответом на каждый вопрос, чтобы принять решение.

Во-вторых, нам потребуется некоторый компонент, который интерпретирует каждый (да или нет) ответ вместе с его важностью для получения окончательного ответа. Звучит достаточно просто, чтобы составить уравнение, верно? Давай сделаем это. Мы просто решаем, насколько важен каждый фактор, и умножаем эту важность (или «переменную») на ответ на вопрос (который может быть 0 или 1):

$$3a + 5b + 2c > 6$$

Числа 3, 5 и 2 являются «переменными» вопросов a, b и с соответственно. a, b и с сами могут быть либо «0» (ответом на вопрос был «нет»), либо «1» (ответом на вопрос было «да»). Если приведенное выше уравнение верно, то решение состоит в том, чтобы носить шляпу, и если оно неверно, решение не носить шляпу. Уравнение говорит, что мы будем носить шляпу, только если сумма переменных, умноженная на *факторы*, больше некоторого *порогового значения*. Пороговым значением было выбрано «6». Если вы думаете об этом, это означает, что, если шляпы нет (b = 0), независимо от того, каковы другие ответы, шляпа не будет надета. То есть,

$$3a + 2c > 6$$

Никогда не верно, так как a и c только 0 или 1. Это имеет смысл — наша простая модель решения говорит нам не носить шляпу, если ее вовсе нет. Таким образом, веса 3, 5 и 2, а также пороговое значение 6 кажутся хорошим выбором для нашего простого лица, принимающего решения «я должен носить шляпу». Это также озна-

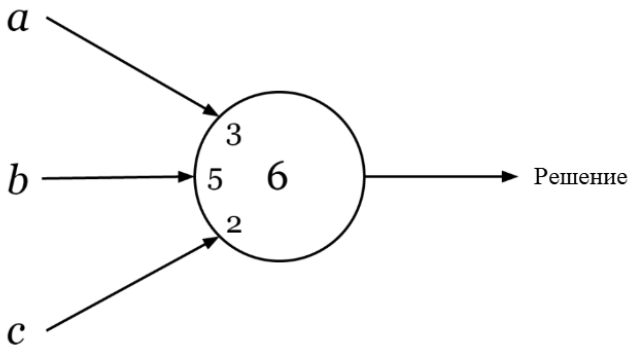

Рис. 1. Нейрон, который обрабатывает 3 фактора: a, b, c, с соответствующими весами важности 3, 5, 2 и с порогом принятия решения 6.

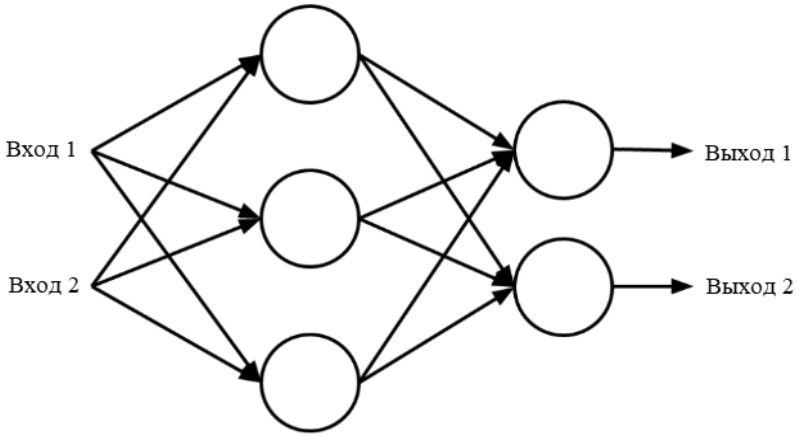

Рис. 2. Простая нейронная сеть с 2 входами и 2 выходами.

чает, что до тех пор, пока есть шляпа, на улице солнечно (a = 1) ИЛИ шляпа подходит для костюма (c = 1). Этого достаточно, чтобы надеть шляпу

То есть,

5 + 3> 6 и 5 + 2> 6

Оба верны. Можно заметить, что, регулируя переменную каждого фактора и пороговое значение и добавляя больше факторов, мы можем настроить «программу, принимающую решение» таким образом, чтобы приблизительно моделировать любой процесс принятия решений.

Попробуем изобразить приведенное выше уравнение в «нейрон-форму»:

Нейрон имеет 3 входных соединения (факторы) и 1 выходное соединение (решение). Каждое входное соединение имеет коэффициент, который преобразует важность этого соединения. Если важность этого соединения низкая, то оно не окажет большого влияния на решение. Если оно высокое, решение будет сильно зависеть именно от этого соединения.

В результате получился работающий нейрон, который «взвешивает» входные данные и принимает решения. Что, если вывод (наше решение) был подан на вход другого нейрона? Этот нейрон будет использовать решение шляпе, чтобы принять *более абстрактное* решение. А что, если входы a, b и c сами являются выходами других нейронов, которые вычисляют решения более низкого уровня? Становится понятно, что нейронные сети можно интерпретировать как сети, которые вычисляют *решения о решениях,* начиная от простых исходных данных и заканчивая более сложными «мета-решениями» [5].

Ниже приведена схема простой нейронной сети, которая, по существу, имеет 3 уровня абстракции:

В качестве примера, вышеуказанные входы могут быть 2 инфракрасными датчиками расстояния, а выходы могут контролировать управление включением/выключение переключателя для 2 -х двигателей, которые приводят в движении колеса робота.

В простом примере с шляпой можно довольно легко менять переменные, но как мы можем выбрать переменные и входную информацию, чтобы, скажем, робот мог следить за движением вещей? И как мы узнаем, сколько нейронов нам нужно, чтобы решить эту проблему? Можем ли мы решить это с помощью всего 1 нейрона, может быть, 2? Или нам нужно 20? И как мы их организуем? В слоях? Модулях? Такие методы, как «обратное распространение» и (совсем недавно) «нейроэволюция», эффективно используются для ответа на некоторые из этих тревожных вопросов.

ИНС работает очень похоже на человеческий мозг. По своей структуре мозг человека состоит из нейронов, и в головном мозге человека насчитывается около 85 миллиардов нейронов. Дендриты нейрона получают входные сигналы от стимуляции окружающей среды или восходящих нейронов. Сигнал обрабатывается в теле ячейки и передается по аксону на выходной нейрон. Выходной сигнал может быть получен нисходящими нейронами или функциональными органами, такими как мышцы, для осуществления реакции. Единственный искусственный нейрон работает аналогичным образом. Функциональные переменные, также известные как предикторы, входные переменные и ковариаты, являются входными сигналами, которые предоставляют информацию для распознавания образов. Каждая переменная «взвешивается» в соответствии с ее важностью. Эта работа выполняется дендритами в биологической нервной системе. Обработанные сигналы суммируются и обрабатываются функцией активации. Процедура обработки сигнала может быть математически выражена как [4]:

$$y(x) = \Phi(\sum_{i=1}^{n} w_i \cdot x_i)$$

где y — выходной сигнал, Φ () — функция активации, x — входные переменные и w — важность, присвоенная каждой входной переменной. Предположим, что есть n входных переменных. Чтобы лучше понять ИНС, ее можно сравнить с регрессионной моделью. Каждая входная переменная аналогична предикторам регрессионной модели. Важность является фактическим коэффициентом для каждого предиктора.

В топологии ИНС входные узлы получают переменные функции из необработанных данных, а выходной узел применяет функцию активации к объединенной информации из входных узлов. Узлы расположены в слоях [3]. Например, все входные узлы составляют один слой. Если сеть содержит только входные и выходные узлы, она называется однослойной сетью или единицами пропускаемого уровня. Такая сеть обычно используется для простой классификации, в которой шаблон результатов является линейно отделимым. Более сложную задачу следует выполнять с многослойной сетью, которая добавляет один или несколько скрытых слоев в однослойную сеть.

Обучение ИНС можно легко выполнить с помощью функции nnet (). Определенное количество информация разделена на обучающие и тестовые наборы. Поскольку наблюдения в данных расположены случайным образом, первые 700 случаев используются в качестве обучающего набора, а остальные 300 используются в качестве тестового набора.

```
> train<-data[1:700,]
> test<-data[701:1000,]
```

До настоящего времени данные хорошо подготовлены для обучения ИНС. В большинстве ситуаций переменные функции должны быть обработаны перед передачей в функцию nnet ().

```
> annmod<-nnet(train[,-1],train[,1],size=6)
```

Приведенный выше код создал объект класса «nnet» и сохранен как ANNmod. Первый аргумент функции nnet () — это фрейм данных переменных функции, а второй аргумент — вектор переменной ответа. Аргумент размера определяет количество единиц в скрытом слое.

Функция визуализации ИНС не включена в пакет nnet. К счастью, есть функция plot.nnet (), которая очень сильна при построении нейронной сети. Это также позволяет настраивать график с различными переменными. Чтобы установить данный пакет в рабочее пространство, необходимо установить пакет devtools. Он содержит пакет разработки инструментов для R.

```
> install.packages("devtools")
> library(devtools)
> source_url('https://gist.githubusercontent.com/
Peque/41a9e20d6687f2f3108d/raw/85e14f3a292e126f14
54864427e3a189c2fe33f3/nnet_plot_update.r')
```

Функция «plot.nnet ()» работает следующим образом.

```
> plot.nnet(annmod, alpha.val = 0.5, pos.col='green',neg.
col='red')
```

Первым аргументом функции является объект «nnet». Функциональность соединений определяется аргументом «alpha.val», значение которого находится в диапазоне от 0 до 1. Цвет положительных соединений определяется аргументом «pos.col», в данном случае зеленый. Точно так же отрицательные связи изображены красным цветом. Рисунок 3 — график ИНС, показывающий узлы и соединения сети. Есть два входных узла с именами I1 и I2, которые передают информацию из переменных функций x1 и x2. Важность присваивается каждому из соединений между входными узлами и скрытыми узлами. Зеленый и красный цвета представляют положительные и отрицательные переменные. B1 — это смещение, применяе-

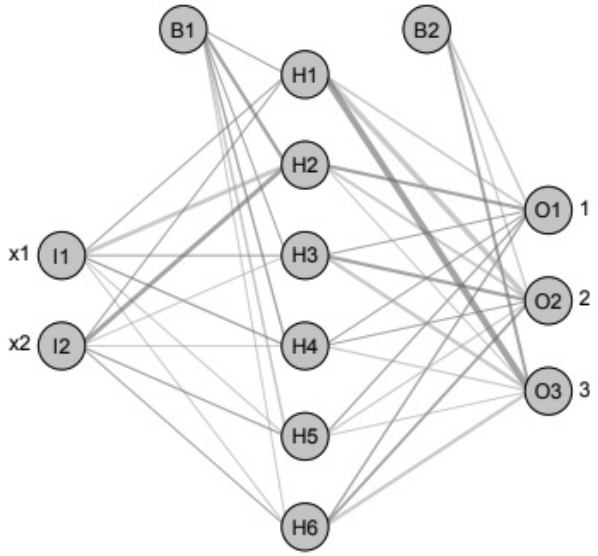

Рис. 3. График искусственных нейронных сетей, показывающий узлы и соединения сети. Есть два входных узла с именами I1 и I2, которые передают информацию из переменных функций x1 и x2. Зеленый и красный цвета представляют положительные и отрицательные переменные. B1 — это смещение, применяемое к скрытым нейронам.

мое к скрытым нейронам. Наконец, сигналы передаются на выходные нейроны, которые обозначены от O1 до O3.

Обучение ИНС — это только первый шаг. Более важной работой является прогнозирование будущих наблюдений с помощью обученной модели. Как и прогнозирование с другими моделями, функция predict() хорошо работает с ИНС [2]. Первый аргумент — это объект «nnet». Второй аргумент — это фрейм данных, содержащий переменные объекта. Поскольку переменная ответа в нашем примере является факторной переменной с тремя уровнями, тип вывода — «class».

```
> pred<-predict(annmod,test[,-1],type="class")
> table(test[,1],pred)
 pred
```

	1	2	3
1	44	12	11
2	7	163	13
3	12	17	21

Чтобы проверить точность прогнозирования модели ИНС, я создал матрицу путаницы, как показано в выходных данных функции table (). Диагональ матрицы показывает правильно классифицированные числа. Кроме того, классификация модели ИНС может быть оценена с использованием precisionCal. Функция precisionCal () позволяет легко вычислить ряд средней точности, изменяя количество единиц в скрытом слое. В следующем синтаксисе функции передается число 30, которое, в свою очередь, возвращает ряд средних погрешностей для ИНС с числом скрытых единиц измерения в диапазоне от 1 до 30.

```
> accuracyCal<-function(N) {
 accuracy<-1
 for (x in 1:N) {
 annmod<-nnet(y~., data=train,
size=x,trace=FALSE,maxit=200)
 pred<-predict(annmod,test[,-1],type="class")
 table<- table(test[,1],pred)
 if (ncol(table)==3) {
 table<-table
 }
 else {
 table<-cbind(table,c(0,0,0))
 }
 tp1<-table[1,1]
 tp2<-table[2,2]
 tp3<-table[3,3]
 tn1<-table[2,2]+table[2,3]+table[3,2]+table[3,3]
 tn2<-table[1,1]+table[1,3]+table[3,1]+table[3,3]
 tn3<-table[1,1]+table[1,2]+table[2,1]+table[2,2]
 fn1<-table[1,2]+table[1,3]
 fn2<-table[2,1]+table[2,3]
 fn3<-table[3,1]+table[3,2]
 fp1<-table[2,1]+table[3,1]
 fp2<-table[1,2]+table[3,2]
 fp3<-table[1,3]+table[2,3]
 accuracy<-c(accuracy, (((tp1+tn1)/
(tp1+fn1+fp1+tn1))+((tp2+tn2)/
(tp2+fn2+fp2+tn2))+((tp3+tn3)/(tp3+fn3+fp3+tn3)))/3)
 }
 return(accuracy[-1])
 }
```

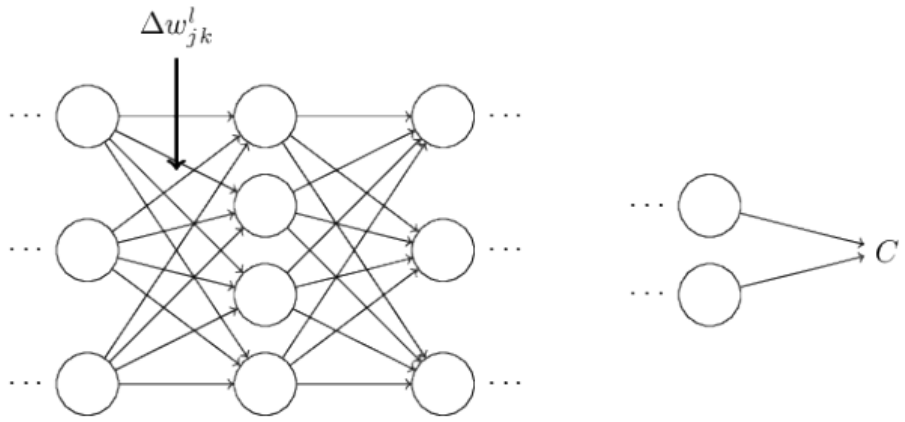

Рис. 4

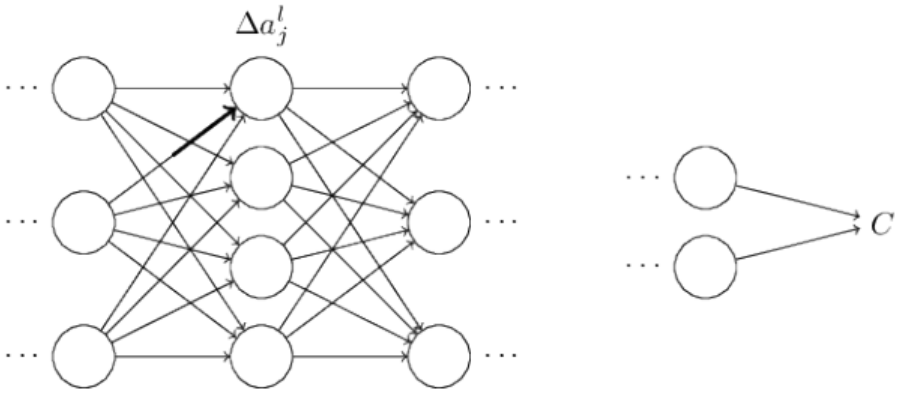

Рис. 5

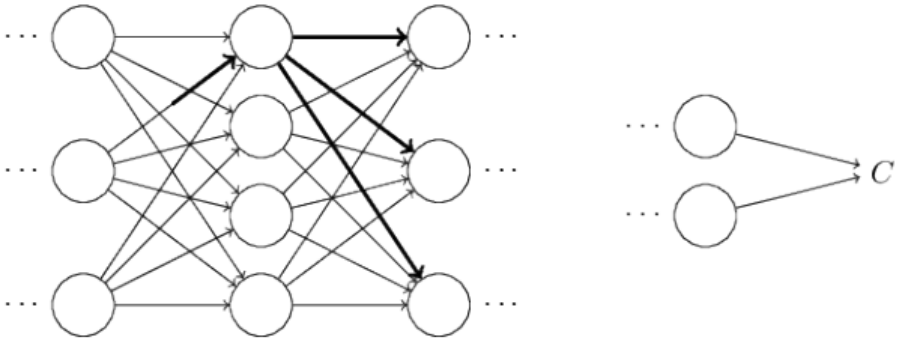

Рис.6

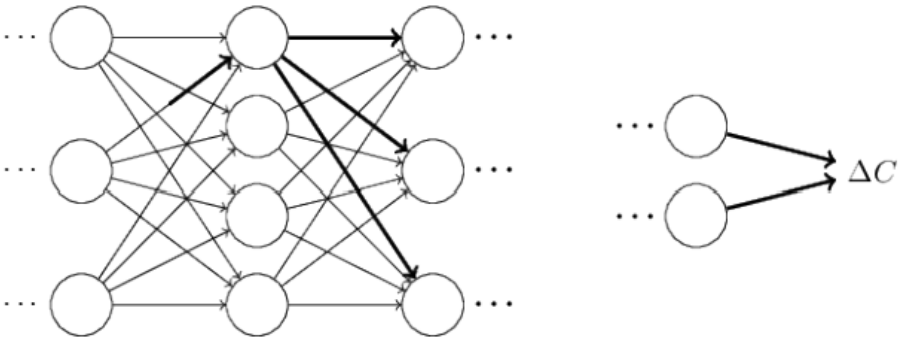

Рис.7

Обратное распространение — это основной механизм обучения нейронных сетей. Это своего рода программа, сообщающая сети, допустила ли она ошибку, когда сделала прогноз [1].

Распространение — это передача чего-либо (свет, звук, движение или информация) в определенном направлении или через определенную среду. Когда анализируется обратное распространение в машинном обучении, необходимо говорить о передаче информации, и эта информация относится к ошибке, вызванной нейронной сетью, когда она делает предположение о данных.

Чтобы улучшить «интуицию» алгоритма, необходимо представить, что мы внесли небольшое изменение «Δw^l_{jk}» в некоторый вес в сети, «w^l_{jk}» (рис. 4).

Это изменение в весе приведет к изменению активации выхода из соответствующего нейрона (рис. 5).

Это, в свою очередь, приведет к изменению всех активаций в следующем слое (рис. 6).

Эти изменения, в свою очередь, будут вызывать изменения на следующем слое, а затем на следующем и т.д. Вплоть до изменения конечного слоя, а затем сети в целом, формируя «интуицию» (рис. 7)

Изменение «ΔC» связано с изменением «Δw^l_{jk}» по уравнению:

$$\Delta C \approx \frac{\partial C}{\partial w^l_{jk}} \Delta w^l_{jk}.$$

Это говорит о том, что возможный подход к вычислению «$\partial C / \partial w^l_{jk}$» заключается в тщательном отслеживании того, как небольшое изменение в w^l_{jk} распространяется, чтобы вызвать небольшое изменение в C.

ЛИТЕРАТУРА

1. Basheer, Imad & Hajmeer, M.N. (2001). Artificial Neural Networks: Fundamentals, Computing, Design, and Application. Journal of microbiological methods. 43. 3–31. 10.1016/S0167–7012(00)00201–3.
2. Buscema, Massimo. (1998). Back Propagation Neural Networks. Substance use & misuse. 33. 233–70. 10.3109/10826089809115863.
3. O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.
4. Schmidhuber, Juergen. (2014). Deep Learning in Neural Networks: An Overview. Neural Networks. 61. 10.1016/j.neunet.2014.09.003.
5. Горбачевская Елена Николаевна Классификация нейронных сетей // Вестник ВУиТ. 2012. № 2 (19).
6. Фаустова К. И. Нейронные сети: применение сегодня и перспективы развития // Территория науки. 2017. № 4.

Журнал «Современная наука: актуальные проблемы теории и практики»