Отчет по задаче на определение четности числа (№3)

Лебедев Андрей, группа 324

Содержание архива

1. Программа parity_1_layer.py
2. Программа parity_2_layers.py
3. Данный отчет

Постановка задачи

1. Построить сеть для определения четности двузначного числа (от 0 до 63). Число представляется в его двоичное виде. Надо написать нейросеть, которая, обучившись на нескольких векторах, правильно классифицирует все остальные.
   Программа должна сначала обучаться на нескольких векторах, а потом проверять, правильно ли вычисляется четность всех чисел от 0 до 63.
2. Сколько слоев достаточно, необходимо, оптимально? Почему?
3. Сколько векторов и итераций достаточно для обучения?
4. Улучшится ли сеть с двумя нейронами?

Вопросы 1 и 3

Для решения задачи я написал две программы, реализующие обучение нейросетей с одним и двумя слоями. Каждая из программ автоматически проверяет и демонстрирует ответы на вопрос 3. То есть обучение нейросети выведено в отдельную функцию, у которой параметрами являются количество векторов в обучающей выборке и количество итераций корректировки весов. Далее запускается обучение при различных параметрах и вычисляется количество ошибок.
Каждая часть кода прокомментирована в необходимой мере, за основу взят алгоритм из программы с семинара и адаптирован под собственное восприятие, некоторые шаги разбиты на несколько.

Программа с обучением нейросети с одним слоем проверяет ее на количестве обучающих векторов от 4 до 14 и при каждом количестве обучающих векторов проводит от 100 до 50000 итераций корректировки весов. Количество ошибок считается при проверке на четность чисел от 0 до 64. Результат выполнения программы:

Count of training vectors: 4
Count of iterations: 100
Count of errors: 19

Count of training vectors: 4
Count of iterations: 10000
Count of errors: 15

Count of training vectors: 4
Count of iterations: 1000
Count of errors: 15

Count of training vectors: 4
Count of iterations: 50000
Count of errors: 15

Count of training vectors: 5
Count of iterations: 100
Count of errors: 9

Count of training vectors: 5
Count of iterations: 10000
Count of errors: 8

Count of training vectors: 5
Count of iterations: 1000
Count of errors: 8

Count of training vectors: 5
Count of iterations: 50000
Count of errors: 8

Count of training vectors: 6
Count of iterations: 100
Count of errors: 9

Count of training vectors: 6
Count of iterations: 10000
Count of errors: 7

Count of training vectors: 6
Count of iterations: 1000
Count of errors: 7

Count of training vectors: 6
Count of iterations: 50000
Count of errors: 7

Count of training vectors: 7
Count of iterations: 100
Count of errors: 7

Count of training vectors: 7
Count of iterations: 10000
Count of errors: 7

Count of training vectors: 7
Count of iterations: 1000
Count of errors: 7

Count of training vectors: 7
Count of iterations: 50000
Count of errors: 7

Count of training vectors: 8
Count of iterations: 100

Count of errors: 9

Count of training vectors: 8
Count of iterations: 1000
Count of errors: 7

Count of training vectors: 8
Count of iterations: 10000
Count of errors: 7

Count of training vectors: 8
Count of iterations: 50000
Count of errors: 7

Count of training vectors: 9
Count of iterations: 100
Count of errors: 5

Count of training vectors: 9
Count of iterations: 1000
Count of errors: 5

Count of training vectors: 9
Count of iterations: 10000
Count of errors: 5

Count of training vectors: 9
Count of iterations: 50000
Count of errors: 5

Count of training vectors: 10
Count of iterations: 100
Count of errors: 4

Count of training vectors: 10
Count of iterations: 1000
Count of errors: 5

Count of training vectors: 10
Count of iterations: 10000
Count of errors: 5

Count of training vectors: 10
Count of iterations: 50000
Count of errors: 5

Count of training vectors: 11
Count of iterations: 100
Count of errors: 4

Count of training vectors: 11
Count of iterations: 1000
Count of errors: 5

Count of training vectors: 11
Count of iterations: 10000
Count of errors: 5

Count of training vectors: 11
Count of iterations: 50000
Count of errors: 5

Count of training vectors: 12
Count of iterations: 100
Count of errors: 3

Count of training vectors: 12
Count of iterations: 1000
Count of errors: 3

Count of training vectors: 12
Count of iterations: 10000
Count of errors: 3

Count of training vectors: 12
Count of iterations: 50000
Count of errors: 3

Count of training vectors: 13
Count of iterations: 100
Count of errors: 3

Count of training vectors: 13
Count of iterations: 10000
Count of errors: 3

Count of training vectors: 13
Count of iterations: 1000
Count of errors: 3

Count of training vectors: 13
Count of iterations: 50000
Count of errors: 3

Count of training vectors: 14
Count of iterations: 100
Count of errors: 3

Count of training vectors: 14
Count of iterations: 10000
Count of errors: 3

Count of training vectors: 14
Count of iterations: 1000
Count of errors: 3

Count of training vectors: 14
Count of iterations: 50000
Count of errors: 3

Эти данные позволяют нам сделать вывод о том, что объем обучающих данных напрямую влияет на качество нейронной сети. Так, начиная с 12 векторов удалось сократить количество ошибок нейронной сети до 3 из 64. В то же время количество итераций неочевидно влияет на качество. Разница между 100 и 50000 итерациями заметна только на маленьком количестве векторов (4 и 5). **Оптимальными значениями можем считать 12 векторов и 100 итераций.**

Далее приведу результаты выполнения 2-й программы, где обучалась нейросеть со скрытым слоем из двух нейронов. Нейросеть проверялась на количестве векторов от 3 до 7 и количестве итераций от 100 до 10000.

Count of training vectors: 3
Count of iterations: 100
Count of errors: 32

Count of training vectors: 3
Count of iterations: 5000
Count of errors: 0

Count of training vectors: 3
Count of iterations: 1000
Count of errors: 0

Count of training vectors: 3
Count of iterations: 10000
Count of errors: 0

Count of training vectors: 4
Count of iterations: 100
Count of errors: 4

Count of training vectors: 4
Count of iterations: 5000
Count of errors: 0

Count of training vectors: 4
Count of iterations: 1000
Count of errors: 0

Count of training vectors: 4
Count of iterations: 10000
Count of errors: 0

Count of training vectors: 5
Count of iterations: 100
Count of errors: 11

Count of training vectors: 5
Count of iterations: 5000
Count of errors: 0

Count of training vectors: 5
Count of iterations: 1000
Count of errors: 2

Count of training vectors: 5
Count of iterations: 10000
Count of errors: 0

Count of training vectors: 6
Count of iterations: 100
Count of errors: 32

Count of training vectors: 6
Count of iterations: 5000
Count of errors: 0

Count of training vectors: 6
Count of iterations: 1000
Count of errors: 0

Count of training vectors: 6
Count of iterations: 10000
Count of errors: 0

Count of training vectors: 7
Count of iterations: 100
Count of errors: 2

Count of training vectors: 7
Count of iterations: 5000
Count of errors: 0

Count of training vectors: 7
Count of iterations: 1000
Count of errors: 0

Count of training vectors: 7
Count of iterations: 10000
Count of errors: 0

В данном случае качество модели значительно выше. Также при наличии скрытого слоя качество зависит уже скорее от количества итераций, нежели от количества векторов. **Оптимальными значениями для второй нейронной сети можно считать 3 обучающих вектора и 1000 итераций.**

Вопрос 2

Из сравнения результатов выполнения первой и второй программ можем сделать вывод, что наличие скрытого слоя сильно повышает качество модели, причем количество ошибок снижается до 0 при достаточно малых значениях обучающих векторов и итераций. Делать большее количество слоев бессмысленно, ведь мы уже **при двух слоях получили идеальный результат**, значит, увеличение слоев только приведет к лишним затратам на вычислительные мощности.

Вопрос 4

Как было сказано выше, вторая нейросеть содержала 2 нейрона. В ходе эсперимента увеличение этого количества до 3-4 не повышало качество результата. А вот уменьшение размера скрытого слоя до 1 нейрона приводит к тому, что при всех рассмотренных мной вариантах количества итераций и размера обучающей выборки нейросеть дает один и тот же ответ как для четных, так и для нечетных чисел.

Выводы

В ходе работы мной был подробно изучен вопрос обучения простейшей нейросети, использующей алгоритм обратного распространения ошибки и метод градиентного спуска. В итоге удалось написать две нейронные сети: с наличием скрытого слоя из двух нейронов и без. Экспериментально я получил оптимальные значения количества итераций и объема обучающей выборки. Также стало очевидно, что добавление скрытого слоя ощутимо улучшает качество модели, сводя количество ошибок к нулю, в то время как без дополнительного слоя так и не удалось получить на 100% достоверные ответы.