Глава
6

Нейросетевая оптимизация

Комбинаторная оптимизация и NP-полные задачи. Сеть Хопфилда решает задачу коммивояжера . Метод иммитации отжига. Оптимизация и сети Кохонена. Растущие нейронные сети. Другие "биологические" методы.

> 🕮 В Смеральдине, городе на воде, сеть каналов накладывается и пересекается с сетью улиц. Чтобы добраться от одного места к другому, всегда можно выбрать между сухопутной дорогой и лодкой, но поскольку в Смеральдине самый короткий путь пролегает не по прямой линии, а по зигзагообразной, ...
> **Итало Кальвино.** ***Незримые города***

Сети минимизирующие энергию, рассмотренные в предыдущей главе, при релаксации к одному из своих стационарных состояний решают, по существу, оптимизационную задачу - поиск минимума определенной функции своего состояния - энергии. Следовательно, и ассоциативную выборку информации, и выявление прототипов можно сформулировать как частный случай задачи оптимизации. В целом же, оптимизационные задачи представляют собой широкий класс задач, часто встречающихся на практике, в частности, в экономике и бизнесе. В этой главе мы покажем как нейросети можно приспособить к решению таких задач на примере очень важного класса задач *комбинаторной оптимизации*. Такие задачи, кроме прочего, позволят нам познакомиться с новыми методами оптимизации, отличающимися от градиентных методов, лежащих в основе обучения методом backpropagation.

Комбинаторная оптимизация и задача коммивояжера

В задачах комбинаторной оптимизации требуется найти наилучшее из конечного, но обычно очень большого числа возможных решений. Если задача характеризуется характерным числом элементов (*размерностью* задачи), то типичное число возможных решений, из которых предстоит сделать выбор, растет экспоненциально - как a^N или еще скорее - как $N!$ (напомним, что согласно известной формуле Стирлинга $N! \cong \left(N/e\right)^N$ для достаточно больших N).

Это свойство делает простой метод перебора всех вариантов, в принципе гарантирующий решение при конечном числе альтернатив, чрезвычайно неэффективным, т.к. такое решение требует *экспоненциально* большого времени. Эффективными же признаются решения,

гарантирующие получение ответа за *полиномиальное* время, растущее как полином с ростом размерности задачи, т.е. как N^a.[1]

🗁 Различие между полиномиальными и экспоненциальными алгоритмами восходит к фон Нейману (von Neumann, 1953)

Задачи, допускающие гарантированное нахождение оптимума целевой функции за полиномиальное время, образуют класс **P**. Этот класс является подклассом более обширного класса **NP** задач, в которых за полиномиальное время можно всего лишь оценить значение целевой функции для конкретной конфигурации, что, естественно, гораздо проще, чем выбрать наилучшую из всех конфигураций. До сих пор в точности не известно, совпадают ли эти два класса, или нет. Эта проблема, $\mathbf{P} \neq \mathbf{NP}$, о которую сломано уже немало математических копий. Если бы эти классы совпадали, для любой задачи комбинаторной оптимизации, *точное* можно было бы *гарантированно* найти за полиномиальное время. В такой "подарок судьбы" никто не верит, и практически разрешимыми считаются задачи, допускающие полиномиальное решение хотя бы для *типичных* (а не *наихудших*) случаев. Такова, например, общая задача линейного программирования.

Для более трудных задач достаточно было бы и более слабого условия - нахождения *субоптимальных* решений, локальных минимумов целевой функции, не слишком сильно отличающихся от абсолютного минимума. Нейросетевые решения как раз и представляют собой параллельные алгоритмы, быстро находящие субоптимальные решения оптимизационных задач, минимизируя *целевую функцию* в процессе своего функционирования или обучения.

Самые трудные задачи класса **NP** называют **NP**-полными. Это название объясняется тем, что если бы удалось доказать, что существует полиномиальное решение такой задачи, то такое решение существовало бы и для *любой другой* задачи класса **NP**, т.е. классы **P** и **NP** совпадали бы. Поскольку, как уже говорилось, такой сценарий крайне маловероятен, именно для таких проблем наиболее важен поиск субоптимальных решений.

Все **NP**-полные задачи одинаково сложны (поскольку все они сводятся друг к другу за полиномиальное время), и методы решения любой из них можно применять также и к другим задачам комбинаторной оптимизации. Поэтому нам в этой главе достаточно сосредоточиться на одной такой задаче. Исторически наиболее исследованной и популярной задачей такого рода (своего рода "мушкой дрозофилой" комбинаторной оптимизации), которая используется для сравнения различных алгоритмов, стала *задача коммивояжера*.

В классической постановке, коммивояжер должен объехать N городов по замкнутому маршруту, посетив каждый из них лишь однажды, таким образом, чтобы полная длина его маршрута была минимальной. Если решать задачу коммивояжера “в лоб” - перебором всех замкнутых путей, связывающих N городов, то придется проверить все $(N-1)!/2$ возможных маршрутов. Будучи **NP**-полной, задача коммивояжера не имеет практически реализуемого точного решения. На примере этой задачи мы ниже и рассмотрим различные методы ее приближенного решения с помощью нейросетей.

[1] Практически при больших размерностях осуществимы все-таки не всякие полиномиальные решения, а лишь полиномы низших степеней.

Оптимизация и сеть Хопфилда

В 1985г. Хопфилд и Танк предложили использовать минимизирующие энергию нейронные сети для решения задач оптимизации (Hopfield & Tank, 1985). В качестве примера они, естественно, рассмотрели задачу коммивояжера.

Для решения этой задачи с помощью нейронной сети Хопфилда нужно закодировать маршрут активностью нейронов и так подобрать связи между ними, чтобы энергия сети оказалась связанной с полной длиной маршрута. Хопфилд и Танк предложили для этого следующий способ.

Рассмотрим сеть, состоящую из $N \times N$ бинарных нейронов, состояния которых мы обозначим $v_{i\alpha} \in \{0, 1\} \quad (i = 1,\ldots,N; \alpha = 1,\ldots,N)$, где индекс i кодирует город, а индекс α - номер города в маршруте (см. Рисунок 1). Если обозначить через d_{ij} расстояние между i-м и j-м городами, решение задачи коммивояжера сводится к минимизации целевой функции

$$L(\nu) = \frac{1}{2}\sum_{i,j,\alpha}^{i\neq j} d_{ij} v_{i\alpha}\left(v_{j\,\alpha-1} + v_{j\,\alpha+1}\right)$$

при дополнительных условиях

$$\sum_i \nu_{i\alpha} = 1 \quad (\forall\alpha) \text{ и } \sum_\alpha \nu_{i\alpha} = 1 \quad (\forall i),$$

первое из которых говорит о том, что любой город в маршруте встречается лишь однажды, а второе - что маршрут проходит через каждый город.

Общий подход к ограничениям в задачах оптимизации состоит в том, что в итоговый функционал, подлежащий минимизации, включаются штрафные члены, увеличивающие целевую функцию при отклонении от накладываемых ограничений. В данном случае в качестве энергии состояния сети можно выбрать функционал

$$E(\nu) = \frac{1}{2}\sum_{i,j,\alpha}^{i\neq j} d_{ij} v_{i\alpha}\left(v_{j\,\alpha-1} + v_{j\,\alpha+1}\right) + \frac{\gamma}{2}\left[\sum_\alpha\left(\sum_i \nu_{i\alpha} - 1\right)^2 + \sum_i\left(\sum_\alpha \nu_{i\alpha} - 1\right)^2\right],$$

где т.н. *множитель Лагранжа* γ регулирует строгость соблюдения дополнительных условий в конечном решении.

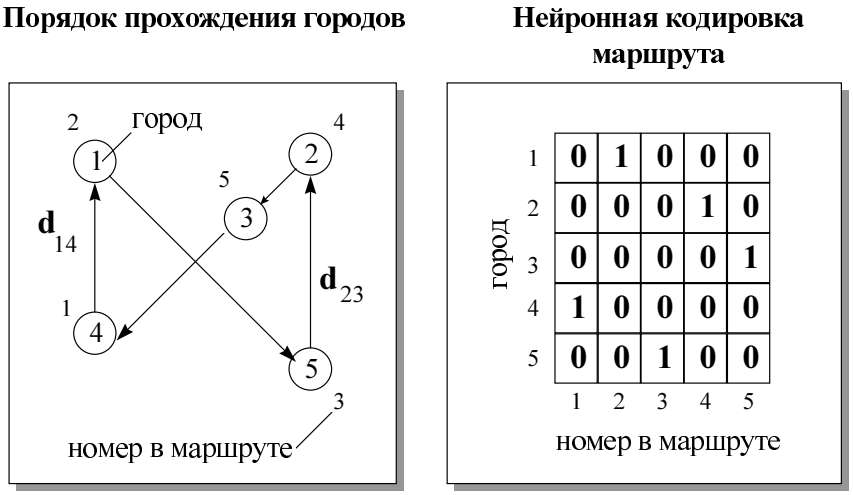

Рисунок 1. Слева - один из возможных маршрутов коммивояжера в случае задачи с 5 городами. Справа - кодировка этого маршрута состояниями 25 бинарных нейронов.

Осмысленному решению будет соответствовать стационарное состояние сети, в котором лишь N нейронов сети будут активными ($v_{i\alpha} = 1$) и в каждом столбце и в каждой строке матрицы $||v_{i\alpha}||$ будет находиться один и только один единичный элемент.

Величина множителя Лагранжа γ регулирует "торг" между поиском маршрута минимальной протяженности и осмысленностью вида самого маршрута. Частное решение, соответствующее локальному минимуму функционала E, может быть осмысленным (второе слагаемое обращаются на нем в ноль), но первое слагаемое (длина маршрута) для него, возможно будет слишком велико. Наоборот, длина маршрута может быть достаточно мала, но одно из оставшихся слагаемых будет ненулевым и маршрут окажется не интерпретируем или недостаточен (например, проходит не через все города).

После того, как минимизируемая целевая функция для задачи коммивояжера построена, можно определить, какие связи в нейронной сети Хопфилда следует выбрать, так чтобы функционал энергии состояния в ней совпал с этой функцией. Для этого достаточно приравнять выражение для $E(\nu)$ к энергии рекуррентной сети:

$$E(\nu) = -\frac{1}{2}\sum_{i,j}^{i\neq j}\sum_{\alpha,\beta}^{\alpha\neq\beta} w_{i\alpha\, j\beta} v_{i\alpha} v_{j\beta} + \sum_{i,\alpha} \vartheta_{i\alpha} v_{i\alpha} .$$

Таким образом находятся значения синаптических связей в сети:

$$w_{i\alpha\, j\beta} = -d_{ij}\left(\delta_{\alpha-1\,\beta} + \delta_{\alpha+1\,\beta}\right) - \gamma\delta_{\alpha\beta} - \gamma\delta_{ij}$$

и значений порогов нейронов $\vartheta_{i\alpha} = -\gamma$. Общее число весов в сети - порядка N^3.

Сети Поттса. Значительного продвижения в эффективности нейросетевой оптимизации можно добиться выбрав более сложный тип нейронов - т.н. *Поттсовские нейронны* - для более естественного представления условий задачи в терминах нейросети (Gilsen et al., 1989). Поттсовский нейроны принимают одно из N значений, что можно описать N-вектором $(0,...,1,...0)$, в котором единица помечает принимаемое им значение. Если при решении задачи коммивояжера сопоставить таким нейронам города, а их состояния соотнести с номером города в туре, то условие посещения города лишь однажды будет гарантировано автоматически.

После того как сеть построена, можно, стартуя со случайного начального состояния, проследить ее эволюцию к стационарной конфигурации, которая может дать если не оптимальное, то по крайней мере хорошее решение задачи. К сожалению, в описанном виде сеть чаще всего "застревает" в локальном минимуме относительно далеком от оптимума.

Для улучшения ситуации Хопфилд и Танк предложили использовать сети с непрерывными (аналоговыми) нейронами, принимающими любые значения в интервале $x_{i\alpha} \in [0, 1]$.[2] В качестве тестовых они использовали задачи с 10 и 30 городами. В первом случае сеть в 20 попытках 16 раз эволюционировала к состояниям, описывающим осмысленный маршрут и в 10 случаях давала один из двух возможных оптимальных маршрутов. Поскольку для задачи с N городами полное число всевозможных маршрутов равно $N!/2N$ (делитель $2N$ возникает вследствие инвариантности маршрута относительно циклического сдвига и обращения направления движения), то в задаче с 10 городами оно составляет 181440. Таким образом, выигрыш при использовании сети, по сравнению со случайным выбором составляет 10^5. В случае задачи с 30 городами полное число маршрутов приблизительно равно 4.4×10^{30}. Экономия, даваемая сетью, составила в этом случае 10^{22}. В дальнейшем было показано, что использование сети Кохонена дает лучшие результаты при решении той же задачи. Однако, поскольку на практике (в робототехнике, при проведении стыковки космических аппаратов, в автоматической навигации) необходимо быстро находить хорошее, но не обязательно лучшее решение, то при электронной реализации аналоговая сеть Хопфилда дает исключительно эффективное решение задач оптимизации.

В дальнейшем разные исследователи выявили и другие особенности описанного подхода. Было показано, что недостатком оригинальной схемы Хопфилда и Танка является то, что простейшая сеть Хопфилда имеет тенденцию включать в маршрут ближайшие друг к другу города. Это происходит из-за того, что в определяющую длину маршрута часть функции Ляпунова входят парные произведения состояний нейронов сети. В результате, с увеличением числа городов маршрут, предлагаемый сетью, как правило, распадается на локально оптимальные участки, соединение которых, однако, далеко от оптимального. Ситуацию можно улучшить, если стимулировать сеть находить, например, локально наилучшие тройки городов. Для этого основная часть функции Ляпунова может быть представлена в виде

$$\sum_{i}^{i} \sum_{k}^{k} \sum_{\alpha} (d_i + d_k) x_{i\alpha} x_{\alpha+1} x_{k\alpha+} \quad .$$

Однако, сети, динамика которых направляется такой функцией Ляпунова, должны состоять из более сложных нейронов, нелинейно суммирующих внешние воздействия - нейронов *высокого порядка* (в данном случае - второго):

[2] Состояния аналоговых нейронов мы обозначаем латинскими буквами, тогда как состояния бинарных нейронов - греческими.

$$h_i = -\vartheta_i + \sum_j w_{ij} x_j + \sum_{j,k} w_{ijk} x_j x_k .$$

Купер показал, что использование таких сетей значительно улучшает результаты поиска оптимального решения. Так для $N = 10$ такая сеть вдвое чаще находит оптимальное решение, чем обычная сеть Хопфилда. Повышение порядка сети приводит к дальнейшему увеличению улучшению найденных сетью решений.

Отметим в заключение, что мы упомянули только о небольшой части разработанных к настоящему времени способов улучшения свойств минимизирующих энергию нейронных сетей при решения задач оптимизации.

Имитация отжига

В предыдущем разделе мы заметили, что переход от бинарных нейронов к аналоговым значительно улучшил свойства решения. Аналогичного эффекта можно добиться используя по-прежнему бинарные нейроны, но заменив детерминистскую динамику стохастической, характеризуемой некоторой эффективной температурой T. При этом *среднее* значение состояния нейрона также будет лежать в допустимом интервале $\left[0, 1\right]$.

Положительная роль температуры заключается в том, что шум позволяет системе покидать локальные минимумы энергии и двигаться в сторону более глубоких энергетических минимумов. Соответствующий (не нейросетевой) алгоритм оптимизации был предложен в 1953 г. и получил название *имитации отжига* (Metropolis et al., 1953). Этот термин происходит от названия способа выжигания дефектов в кристаллической решетке. Атомы, занимающие в ней неправильное место, при низкой температуре не могут сместиться в нужное положение - им не хватает кинетической энергии для преодоления потенциального барьера. При этом система в целом находится в состоянии локального энергетического минимума. Для выхода из него металл нагревают до высокой температуре, а затем медленно охлаждают, позволяя атомам занять правильные положения в решетке, соответствующее глобальному минимуму энергии.

Субоптимальное решение некоторой задачи оптимизации, например, задачи коммивояжера, также может рассматриваться как решение в котором имеются дефекты - неправильные части маршрута. Лин и Кернигэн (Lin & Kernigan, 1973) ввели элементарные операции изменения текущего решения, такие как *перенос* (часть маршрута вырезается и вставляется в другое место) и *обращение* (выбирается фрагмент маршрута и порядок прохождения городов в нем меняется на обратный). При применении одной из этих операций происходит изменение маршрута с M на M', и значение минимизируемого функционала меняется на $\Delta E = E(M') - E(M)$. В соответствии с принципами термодинамики, это изменение принимается с вероятностью

$$\Pr\{M \to M'\} = \begin{cases} 1, & \Delta E \le 0 \\ \exp(-\Delta E / T), & \Delta E > 0 \end{cases}$$

где T - эффективная температура. Таким образом в методе отжига с некоторой вероятностью допускается переход системы в состояния с более высокой энергией. Эта вероятность тем выше, чем выше эффективная температура. Поиск минимума начинается с некоторого начального маршрута при высоком значении температуры. По мере эволюции состояния системы эта температура медленно снижается (для примера - на 5% после осуществления

$100N$ элементарных операций изменения маршрута). Поиск продолжается до тех пор, пока система не захватывается энергетическим минимумом, из которого она уже не может выйти за счет тепловых флуктуаций. Многочисленные исследования показали, что метод имитации отжига является очень эффективным способом получения решений близких к оптимальному и часто служит эталоном сравнения для нейросетевых подходов. Заметим, однако, что при реализации "в железе" нейросетевой подход все равно оказывается вне конкуренции по скорости получения решения.

Метод эластичной сети

Иной подход к решению задачи коммивояжера использовали в 1987 году Дурбин и Уиллшоу (Durbin & Willshaw, 1987). Хотя они явно и не использовали в своей работе понятия искусственной нейронной сети, но в качестве отправной точки упоминали об аналогии с механизмами установления упорядоченных нейронных связей. Исследователи предложили рассматривать каждый из маршрутов коммивояжера как отображение окружности на плоскость, так что в каждый город на плоскости отображается некоторая точка этой окружности. При этом требуется, чтобы соседние точки на окружности отображались в точки, по возможности ближайшие и на плоскости. Алгоритм стартует с помещения на плоскость небольшой окружности (кольца), которая неравномерно расширяясь проходит практически около всех городов и, в конечном итоге, определяет искомый маршрут. Каждая точка расширяющегося кольца движется под действием двух сил: первая перемещает ее в сторону ближайшего города, а вторая смещает в сторону ее соседей на кольце так, чтобы уменьшить его длину. По мере расширения такой эластичной сети, каждый город оказывается ассоциирован с определенным участком кольца.

Вначале все города оказывают приблизительно одинаковое влияние на каждую точку маршрута. В последующем, большие расстояния становятся менее влиятельными и каждый город становится более специфичным для ближайших к нему точек кольца. Такое постепенное увеличение специфичности, которое, конечно, напоминает уже знакомый нам метод обучения сети Кохонена, контролируется значением некоторого эффективного радиуса R. Если обозначить через $\mathbf{X}_i$ вектор, определяющий положение i-го города на плоскости, а $\mathbf{Y}_j$ - координату j-й точки на кольце, то закон изменения последний имеет вид

$$\Delta\mathbf{Y}_j = \alpha\sum_j W_j \cdot\left(\mathbf{X} - \mathbf{Y}_j\right) + \beta R\cdot\left(\mathbf{Y}_{j+1} - 2\mathbf{Y}_j + \mathbf{Y}_{j-1}\right),$$

где параметры α, β определяют относительное воздействие на точку $\mathbf{Y}_j$ описанных выше двух сил. Коэффициенты W_{ij}, определяющие воздействие i-го города на j-ю точку кольца, являются функцией расстояния $\left|\mathbf{X}_i - \mathbf{Y}_j\right|$ и параметра R. Эти коэффициенты нормированы так, что полное воздействие каждого из городов оказывается одинаковым:

$$W_{ij} = \Phi\left(\left|\mathbf{X}_i - \mathbf{Y}_j\right|, R\right)\Big/\sum\nolimits_k \Phi\left(\left|\mathbf{X}_i - \mathbf{Y}_k\right|, R\right),$$

где $\Phi\left(d, R\right)$ - положительная, ограниченная и убывающая функция d, приближающаяся к нулю при $d > R$ Если в качестве этой функции выбрать распределение Гаусса $\exp\left(-d^2/2R^2\right)$, то можно определить функцию Ляпунова

$$E = -\alpha R \sum_i \log \sum_j \Phi(|\mathbf{X}_i - \mathbf{Y}_j|, R) + \beta \sum_j |\mathbf{Y}_{j+1} - \mathbf{Y}_j|^2 ,$$

которая минимизируется в ходе динамического изменения параметров кольца.

Дурбин и Уиллшоу показали, что для задачи с 30 городами, рассмотренной Хопфилдом и Танком, метод эластичной сети генерирует наикратчайший маршрут примерно за 1000 итераций. Для 100 городов найденный этим методом маршрут лишь на 1% превосходил оптимальный.

Оптимизация с помощью сети Кохонена.

Успех применения метода эластичной сети для решении задачи коммивояжера был оценен Фаватой и Уолкером, понявшими, что в нем, по сути, используется отображение двумерного распределения городов на одномерный кольцевого маршрута (Favata & Walker, 1991). Поскольку в наиболее общем виде такой подход был сформулирован Кохоненом, то использование его самоорганизующихся карт для оптимизации оказалось вполне естественным. Сеть Кохонена позволяет обеспечить выполнение условия, которому должен удовлетворять хороший маршрут в задаче коммивояжера: близкие города на плоскости должны быть отображены на близкие в одномерном маршруте.

Алгоритм решения задачи следует из оригинальной схемы Кохонена, в которую вносятся лишь небольшие изменения. Используется сеть, состоящая из двух одномерных слоев нейронов (т.е. содержащая лишь один слой синаптических весов). Входной слой состоит из трех нейронов, а выходной - из N (по числу городов). Каждый нейрон входного слоя связан с каждым выходным нейроном. Все связи вначале инициируются случайными значениями. Для каждого города входной 3-мерный вектор формируется из двух его координат на плоскости, а третья компонента вектора представляет из себя нормирующий параметр, вычисляемый так, чтобы все входные вектора имели одинаковую Евклидову длину и никакие два вектора не были бы коллинеарны. Это эквивалентно рассмотрению двумерных координат городов, как проекций трехмерных векторов, лежащих на сфере. Обозначим через $\mathbf{w}^j$ 3-мерный вектор синаптических связей, связывающих *j*-й выходной нейрон с входными нейронами. Если $\mathbf{c}^i$ - трехмерный входной вектор, определяющий *i* -й город, то активация *j-го* выходного нейрона при подаче $\mathbf{c}^i$ на вход определяется скалярным произведением ($\mathbf{c}^i, \mathbf{w}$). Выходной нейрон, для которого это произведение максимально, называется *образом* города.

Алгоритм формирования маршрута формулируется следующим образом. Выбираются значения для параметра усиления α и радиуса взаимодействия r. Следующий цикл выполняется вплоть до выполнения условия $\alpha \le 0$.

1) Выбирается случайный город c.

2) Определяется номер образа города в выходном слое - i_c.

3) Векторы связей $\mathbf{w}$, соединяющих нейрон i_c, и всех его 2r близлежащих соседей справа и слева: $j = i_c - r,\quad i_c - r + 1, \ldots, i_c, \ldots, i_c + r - 1, i_c + r$ модифицируются следующим образом:

$$\mathbf{w}^j(t+1) = \frac{\mathbf{w}^j(t) + \mathbf{c}^i \alpha(t)}{||\mathbf{w}^j(t) + \mathbf{c}^i \alpha(t)||},$$

где $||\mathbf{x}||$ - Евклидова норма вектора $\mathbf{x}$. Для устранения концевых эффектов слой выходных нейронов считается кольцевым, так что N-й нейрон примыкает к первому.

4) Радиус взаимодействия постепенно уменьшается согласно некоторому правилу (например, вначале можно положить $r = 0.1N$, затем за первые 10% циклов снизить его до значения 1, которое далее поддерживается постоянным).

5) Параметр усиления α постепенно снижается на небольшую величину (например, в экспериментах Фавата и Уолкера он линейно уменьшался до нуля).

Конкретный вид законов изменения радиуса взаимодействия и параметра усиления, как правило, не имеет большого значения.

После завершения процесса обучения, положение города в маршруте определится положением его образа в кольцевом выходном слое. Иногда случается, что два или большее число городов отображаются на один и тот же выходной нейрон. Подобная ситуация может интерпретироваться так, что локальное упорядочивание этих городов не имеет значения и требует только локальной оптимизации части маршрута. При нескольких десятках городов такая оптимизация может скорректировать его длину на величину до 25%. Для сотен городов она, как правило, не улучшает результат и поэтому не используется.

Эксперименты Фаваты и Уолкера, проведенные для задачи коммивояжера с 30 городами дали лучшие результаты, чем полученные с помощью сети Хопфилда (см. таблицу).

Таблица 1. Сравнение результатов решения задачи коммивояжера с 30 городами

	сеть Хопфилда	сеть Кохонена
Длина маршрута	< 7	< 5.73
Средняя длина маршрута	> 6	4.77
Наименьшая длина маршрута	5.07	4.26

Однако для большего числа городов сеть Кохонена все же в среднем дает более длинные маршруты, чем метод имитации отжига (примерно на 5%). При практическом применении нейросетевых подходов к решению задач оптимизации, однако, главное значение имеет не столько близость решения к глобальному оптимуму, сколько эффективность его получения. В этом смысле сеть Кохонена значительно эффективнее имитации отжига. Однако, и ее использование, как и в случае использования других лучших методов оптимизации, требует вычислительных затрат, растущих не медленне, чем N^2. Ниже мы опишем нейросетевой подход, в котором они растут линейно с размерностьюзадачи.

Растущие нейронные сети

Эффективное практическое применение нейронных сетей для оптимизации возможно, если вычислительные затраты у соответствующей модели не слишком быстро растут с ростом размерности задачи. Так, для задачи коммивояжера затраты при эмуляции сети Хопфилда на последовательном компьютере растут как N^5, т.к. каждый из N нейронов имеет порядка N^3 синаптических весов.[3] Эвристический подход Лина и Кернигана масштабирует вычислительные затраты как $N^{2.2}$. Фритцке и Вильке предложили нейросетевую систему очень близкую к сети Кохонена, для которой затраты даже при ее эмуляции на последовательном компьютере растут лишь линейно с размерностью задачи (Fritzke & Wilke, 1991).

Предложенная ими модель относится к классу растущих нейронных сетей. Такие сети по-своему решают задачу адаптации своей структуры к требованиям решаемой задачи. Вспомним многослойные персептроны, для которых количества скрытых слоев и нейронов в них часто выбираются методом проб и ошибок. Как уже отмечалось в связи с этим, имеются два подхода к адаптивному выбору архитектуры нейросетей. В первом подходе заведомо избыточная нейросеть прореживается до нужной степени сложности. Растущие сети, напротив, стартуют с очень простых и небольших структур, которые разрастаются и усложняются по мере необходимости.

Фритцке и Вильке разработали целый класс самоорганизующихся (и обучаемых с учителем) сетей с изменяющейся структурой, такие как *Растущие Клеточные Структуры, Растущий Нейронный Газ* и *Растущие Сетки*. Первые и были использованы ими для решения задачи коммивояжера (и других задач комбинаторной оптимизации).

Растущая клеточная структура для задачи коммивояжера представляет из себя вначале кольцо из трех ячеек нейронов. Каждый нейрон характеризуется двумерным вектором $\mathbf{w}_i$, определяющим его положение на плоскости. Каждому нейрону в кольце приписывается также своя величина погрешности ε_i, которая вначале полагается равной нулю. Дальнейшая последовательность действий включает две следующие основные элементарные операции: *смещение* и *добавление нового нейрона.*

Смещение (см. Рисунок 2)

- выбирается случайный город $\boldsymbol{c}$
- определяется нейрон-победитель $\mathbf{w}_*$, ближайший к этому городу
- положение нейрона $\mathbf{w}_*$ и его двух ближайших в кольце соседей смещается в сторону города $\boldsymbol{c}$ на определенную долю расстояния до него.

Эти операции очень близки к используемым в модели Кохонена. Различие состоит в том, что в последней радиус, в котором определяется соседство и параметр адаптации уменьшаются со временем.

[3] Параллельный же аналоговый вариант сети Хопфилда находит решение за конечное число шагов, практически не зависящее от N.

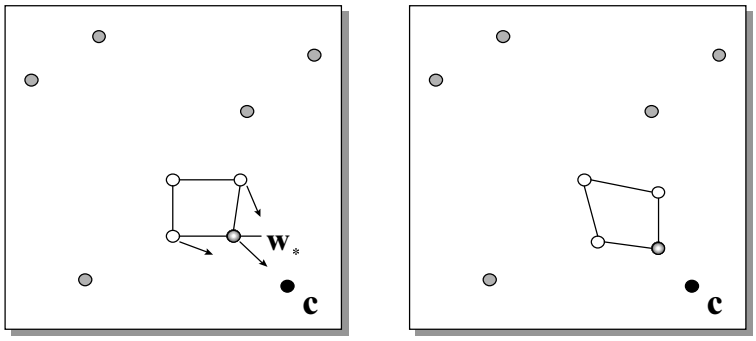

Рисунок 2. Процедура смещения перемещает нейрон-победитель и его ближайших соседей в сторону случайно выбранного города

Добавление нового нейрона.

Со временем после нескольких циклов смещений накапливается информация, на основании которой принимается решение о месте, в котором должен быть добавлен новый нейрон. Каждый раз, когда для случайно выбранного города $\mathbf{c}$ определяется ближайший к нему нейрон $\mathbf{w}_*$, локальная ошибка для последнего ε_{i_*} получает приращение $||\mathbf{w}_* - \mathbf{c}||$. Большое значение этой ошибки служит указанием на то, что соответствующий нейрон лежит в области, где отношение *<число нейронов>/<число городов>* невелико. Именно в таких областях следует добавлять новые нейроны, поскольку для получения правильного осмысленного маршрута около каждого города должен находиться свой ближайший нейрон. Маршрут и определяется путем перехода вдоль кольца к нейрону, являющимся ближайшим к некоторому городу. Алгоритм поиска оптимального маршрута, использующий две описанные операции, формулируется следующим образом

0) Инициализация: генерируется кольцевая структура, состоящая из трех нейронов, имеющих случайное положение на плоскости.

1) Осуществляется фиксированное число n_d шагов распространения. На каждом шаге пересчитывается значение локальной ошибки ε_{i_*}.

2) Определяется “наихудшее” звено в кольце, связывающее два нейрона i_1 и i_2, для которых сумма $\varepsilon_{i_1} + \varepsilon_{i_1}$ максимально. Новый нейрон вставляется в середину звена связывающего нейроны i и i_2, и его ошибка инициализируется величиной $\varepsilon_{inew} = \frac{1}{3}\varepsilon_{i_1} + \frac{1}{3}\varepsilon_{i_2}$. В то же время значения ошибок для нейронов i и i_2 уменьшается таким образом, чтобы суммарная ошибка сохранилась: $\varepsilon_{i_1} = \frac{2}{3}\varepsilon_{i_1}$, $\varepsilon_{i_2} = \frac{2}{3}\varepsilon_{i_2}$

3) Если для любых двух городов ближайшие к ним нейроны различны между собой, то маршрут найден. В противоположном случае возвращаемся к шагу 1.

Очевидно, что решение задачи может быть найдено не ранее того, как число нейронов в кольце достигнет числа городов N. В действительности для его достижения требуется сеть с $2N$-$3N$ нейронами. Исходя из этого эмпирического наблюдения, согласно которому число итераций имеет порядок $O(N)$, можно оценить общую сложность алгоритма. На шаге 1 требуется инспекция всех нейронов для поиска ближайшего к данному городу. Она производится n_d раз и, поскольку это число постоянно, полное число инспекций также имеет порядок $O(N)$. На шаге 2 необходимо проверить каждое звено цепи, чтобы найти то, которому соответствует максимальная суммарная ошибка концевых нейронов. Поскольку число звеньев равно числу нейронов, то число действий опять имеет порядок $O(N)$. На шаге 3 для каждого города необходимо найти ближайший нейрон, что, как минимум, требует $O(N)$ действий. Таким образом, так как шаги 1-3 требуют по меньшей мере $O(N)$ операций, а цикл повторяется $O(N)$ раз, то временная сложность алгоритма как минимум равна $O(N^2)$. Пространственная сложность алгоритма составляет $O(N)$, так как необходимо резервировать память для N городов, $O(N)$ нейронов и некоторых локальных переменных.

Для улучшения квадратичной временной сложности описанного алгоритма Фритцке и Вильке модифицировали шаги 1-3. Они учли, что согласно численным экспериментам вначале кольцевая структура нейронов быстро распределяется по всей области размещения городов, и затем с ростом числа нейронов изменения приобретают локальный характер. Такое поведение натолкнуло их на идею заменить глобальный поиск нейрона-победителя на шаге 1 приближенной локальной процедурой. А именно: для каждого города запоминается тот нейрон, который наиболее часто оказывался к нему ближайшим, и если город выбран вновь, то поиск ближайшего к нему нейрона ограничивается *этим* нейроном и его ближайшими по кольцу соседями вплоть до порядка k. Поскольку k есть константа, то сложность поиска оказывается в этом случае $O(1)$.

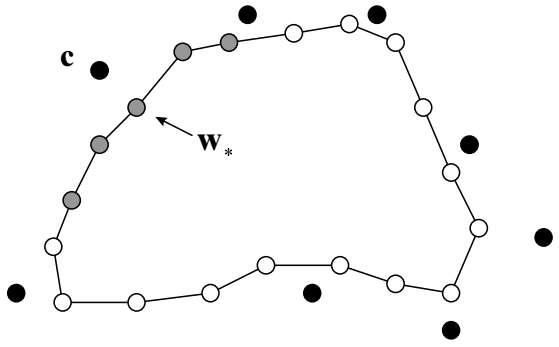

Рисунок 3. Локальный поиск наилучшего нейрона: $\mathbf{w}_*$ -предыдущий нейрон ; ближайшие его соседи вплоть до 2 порядка являются кандидатами в победители на следующем шаге.

Для устранения на шаге 2 линейного поиска звена с максимальной ошибкой используется тот факт, что таким звеном является то, которое связывает нейроны, часто становящиеся победитями.

Третий шаг тоже можно модифицировать: если некоторый нейрон несколько раз оказывается ближайшим для данного города, значит для этого города структура кольца уже стабилизировалась и нейрон “приклеивается” к данному пункту маршрута. Это означает, что он совмещается со своим городом и больше уже не двигается. Город же удаляется из списка

городов, разыгрываемых на шаге распределения. Когда этот список становится пустым процесс поиска маршрута заканчивается.

Таким образом, каждый шаг в цикле теперь требует постоянное число операций и временная сложность всего алгоритма становится порядка $O(N)$.

Описанный эффективный нейросетевой подход (FLEXMAP) был протестирован на разных распределениях городов числом до 200 и неизменно находил маршруты, отличающиеся не более чем на 9% от оптимального.

Нейросетевая оптимизация и другие "*биологические* "методы

Преимущества и недостатки нейросетевой оптимизации познаются в сравнении с другими развитыми в настоящее время методами. Из методов, которые иногда дают аналогичные, а порой и лучшие результаты, отметим генетические и эволюционные алгоритмы (Fogel, 1993), а также метод муравьиных колоний (Dorigo & Gambardella, 1996).

В этом разделе мы очень кратко остановимся на них, поскольку эти подходы, так же как и нейросети, используют ясные и плодотворные биологические аналогии. Кроме того, генетические алгоритмы широко используются и для обучения нейронных сетей самих по себе, поскольку обучение нейросетей связано с минимизацией функционала ошибки.

Генетические алгоритмы

Эти алгоритмы могут использоваться для поиска экстремума нелинейных функций с множественными локальными минимумами. Они имитируют адаптацию живых организмов к внешним условиям в ходе эволюции. Точнее, они моделируют эволюцию целых *популяций* организмов и поэтому требуют достаточно больших ресурсов памяти и высокой скорости вычислительных систем. Важным достоинством их является то, что они не накладывают никаких требований на вид минимизируемой функции (например, дифференцируемость). Поэтому их можно применять в случаях, когда градиентные методы не применимы.[4]

Генетические алгоритмы используют соответствующую терминологию, конфигурации системы называют *хромосомами*, над которой можно производить операции *кроссинговера* и *мутации*. Хромосома является основной информационной единицей, кодирующей переменную, относительно которой ищется оптимум. Обычно она представляет собой битовую строку, хотя компоненты этой строки могут иметь и более общий вид (для задачи коммивояжера компоненты хромосом представляют собой последовательность номеров городов в данном маршруте, например (145321)). Каждая компонента хромосомы называется *геном*. Выбор удачного представления для хромосомы, или же кодировка искомого решения, могут значительно облегчить нахождение решения.

Обучение происходит в популяции хромосом, к которым на каждом шаге эволюции применяются две основные операции. При мутациях в хромосоме случайным образом выбираются и изменяются ее компоненты (гены). При кроссинговере две хромосомы A и B разрезаются на две части в случайно выбранной *одной точке* $A=(A_1, A_2)$ и $B=(B_1, B_2)$ и обмениваются ими, давая две новые хромосомы: $A'=(A_1, B_2)$ и $B'=(B_1, A_2)$ (см. Рисунок 4).

[4] Например, для обучения нейросетей с бинарными весами

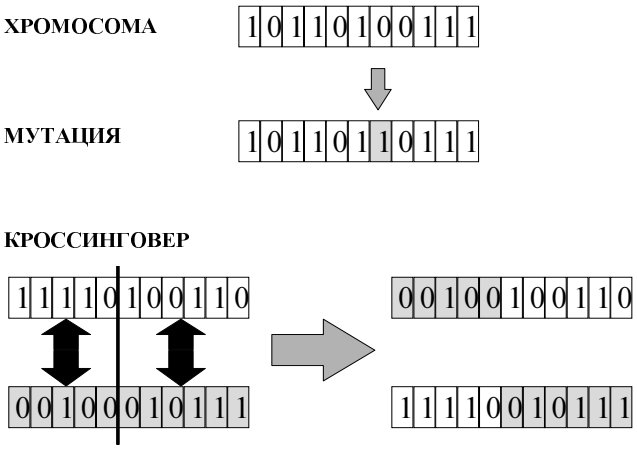

Рисунок 4. Представление искомого решения в виде битовой строки - хромосомы (вверху). Операции мутации и кроссинговера (внизу)

После каждого шага эволюции - *генерации*, на котором мутируют и подвергаются кроссинговеру все хромосомы, для каждой из новых хромосом вычисляется значение целевого функционала, которое достигается на кодируемых ими решениях. Чем меньше это значение для данной хромосомы, тем с большей вероятностью она отбираются для кроссинговера. В ходе эволюции усредненное по популяции значение функционала будет уменьшаться, и после завершения процесса (проведения заданного числа генераций) хромосома с минимальным его значаением выбирается в качестве приближенного решения поставленной задачи. Можно значительно улучшить свойства генетического алгоритма если после порождения новой генерации *N* хромосом предварительно объединить ее с предыдущей популяцией и выбрать из *2N* полученных хромосом *N* наилучших. Опыт показывает, что генетические алгоритмы особенно эффективны при поиске глобального оптимума, поскольку они осуществляют поиск в широком пространстве решений. Если закодировать в виде хромосом значения весов и порогов нейронной сети заданной архитектуры и использовать в роли минимизируемой функции функционал ошибки, то генетические алгоритмы можно использовать для обучения этой нейронной сети. Очевидно, что для этой же цели можно использовать и описанный ранее метод иммитации отжига.

Метод муравьиных колоний

Энтомологи установили, что муравьи способны быстро находить кратчайший путь от муравейника к источнику пищи. Более того, они могут адаптироваться к изменяющимся условиям , находя новый кратчайший путь. Рассмотрим Рисунок 5: муравьи движутся по прямой, соединяющей муравейник с местом, в котором находится пища. При движении муравей метит свой путь специальными веществами - *феромонами*, и эта информация используется другими муравьями для выбора пути. А именно, муравьи предпочитают тропки наиболее обогащенные феромонами. Это элементарное правило поведения муравьев и определяет их способность находить новые пути, если старый оказывается перерезанным преградой. Действительно, достигнув этой преграды, муравьи уже не смогут продолжить свой путь и с равной вероятностью будут обходить ее справа и слева. То же самое будет происходить и на обратной стороне преграды. Однако, те муравьи, которые случайно выберут кратчайший путь (налево от преграды и направо - на обратном пути), будут быстрее проходить свой путь и он с большей скоростью станет обогащаться феромонами. Поэтому следующие муравьи будут предпочитать именно

этот наикратчайший путь, метя его и далее. Очевидная положительная обратная связь быстро приведет к тому, что кратчайший путь станет единственным маршрутом движения насекомых.

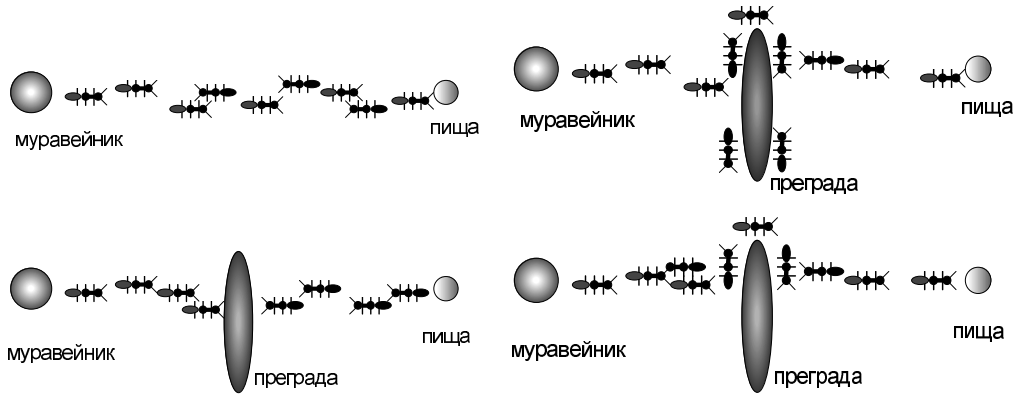

Рисунок 5. Муравьи находят новый кратчайший новый путь (сверху от преграды) который быстрее обогащается феромонами.

Подобный процесс может осуществляться и в компьютерном мире, населенном Искусственными Муравьями (ИМ). Такие муравьи могут решить и нашу задачу коммивояжера. В этом случае они движутся от города к городу по ребрам соответствующего графа. При этом они выбирают направление движения, используя вероятностную функцию, зависящую как от предыдущих попыток движения по данному ребру, так и от эвристического значения, являющегося функцией длины ребра. ИМ с большей вероятностью будут предпочитать ближайшие города и города, связанные ребрами, наиболее богатыми феромонами. Первоначально N искусственных муравьев размещаются в случайно выбранных городах. В каждый последующий момент времени они перемещаются в соседние города и изменяют концентрацию феромона на своем пути (локальная модификация). После того, как все ИМ завершат движения по замкнутому маршруту, тот из них, который проделал кратчайший путь, добавляет к его звеньям количество феромона, обратно пропорциональное длине этого пути (глобальная модификация). В отличие от живых муравьев, ИМ обладают способностью определять расстояние до соседних городов и помнят, какие города они уже посетили. Оказывается, метод искусственных муравьиных колоний может давать результаты, лучшие чем при использовании имитации отжига, нейронных сетей, и генетических алгоритмов.

Таблица 2. Результаты решения задачи коммивояжера (длина маршрута)

Набор	Муравьи	Отжиг	Эластич. Сети	Сети Кохонена
1	5.86	5.88	5.98	6.06
2	6.05	6.01	6.03	6.25
3	5.57	5.65	5.70	5.83
4	5.70	5.81	5.86	5.87
5	6.17	6.33	6.49	6.70

Напомним вновь, что при электронной или оптической реализации нейросетевой подход находится вне конкуренции в ситуациях, когда необходимо очень быстро находить не обязательно оптимальное, но достаточно хорошее решение.

ЛИТЕРАТУРА

Кук, С. (1982) "Обзор вычислительной сложности". Тьюринговская лекция в: *Лекции лауреатов премии Тьюринга за первые двадцать лет 1966-1985.* Мир. М:1993.

Burke, L.,L.., and Ignizio, J,P. (1992) "Neural networks and operations research: An overview". *Computers and Operations Research*, **19**, 179.

Cichocki, A. and Unbehauen, R. *Neural Networks for Optimization and Signal Processing.* John Wiley & Sons, 1994.

Cooper, B.,S. (1995) "Higher order neural networks - can they help us optimise?" *Proceedings of the Sixth Australian Conference on Neural Networks (ACNN'95)* , 29.

Cooper, B.S. (1996) "A comparison of the number of stable points of oprimisation networks". *Univ. of Adelaide, Report CSSIP TR*, 4/96.

Dorigo, M., and Gambardella, L.,M. "Ant colonies for the traveling salesman problem". *Technical Report, Universte Libre de Bruxelles - TR/IRIDIA/1996-3*.

Durbin, R., and Willshaw, D.(1987) "An analogue approach to the travelling salesman problem using an elastic net method". *Nature*, **326**, 689.

Favata, F. and Walker, R. (1991) "A study of the application of Kohonen-type neural network to the travelling salesman problem". *Biological cybernetics*, **64**, 463.

Fritzke, B. and Wilke, P.(1991) "FLEXMAP - A neural network for the travelling salesman problem with linear time and space complexity". *Proceddings of IJCNN-91*, Singapore, 929.

Fogel, D. (1993) "Applying evolutionary programming to selected traveling salesman problems". *Cybernetics and Systems. An International Journal*, **24**, 27.

Gee, A.,H. (1993) *Problem solving with optimization networks*, PhD thesis, University of Cambridge.

Gislen, L., Soderberg, B., and Peterson, C. (1992) "Complex scheduling with Potts neural networks", *Neural Computation*, **4**, 805.

Gislen,L., Soderberg, B., and Peterson, C. (1989) "Teachers and classes with neural networks", *International Journal of Neural Systems*, **1**, 167.

Hopfield J.,J., & Tank, D.,W. (1985) "Neural computation of decisions in optimization problems", *Biological Cybernetics*, **52**, 141.

Lin, S. & Kernigan, B.,W. (1973) "An effective heuristic algorithm for the travelling-salesman problem", *Operations Research*, **21**, 498.

Looi, C. (1992) "Neural networks methods in combinatorial optimization", *Computers and Operations Research*, **19**, 191.

Metropolis, N., Rosenbluth, A.,W., Rosenbluth, M.,N., Teller,A.,H., Teller, E.,J. (1953) *J.Chem.Phys.* **21**, 1087.

von Neumann, J. (1953) “A certain zero-sum two-person game equivalent to the optimal assignment problem”. *Contributions to the Theory of Games II.*H.W. Kahn and A.W. Tucker, Eds. Princeton Univ. Press, Princeton, NJ.

Ogier, R.,G. and Beyer, D.,A. (1990) “Neural network solution to the link scheduling problem using convex relaxation”. *Proceedings of the 1990 IEEE Global Telecommunications Conference*, Dec.,1371.

Ohlsson,M., Peterson, C., and Soderberg, B. (1993) “Neural networks for optimization problems with inequality constraints - the knapsack problem”. *Neural Computation*, **5**, 331.

Peterson, G. and Soderberg, B. (1989) “A new method for mapping optimization problems onto neural networks”, *International Journal of Neural Systems*, **1**, 3.

Potvin, J-Y. (1993) “The travelling salesman problem - A neural network perspective”. *ORSA Journal of Computing*, **5**, 328.

Smith, K., Palaniswami. M., and Krishnamoorthy, M. (1996) “Traditional heuristic versus Hopfield neural network approaches to car sequencing problem”. *European Journal of Operational Research*.

Vaithyanathan, S., and Ignizio, J.,P. (1992) “A stochastic neural network for resource constrained scheduling”. *Computers and Operations Research*, **19**, 241.

Wilson, G.,V. and Pawley, G.,S. (1988) “On the stability of the Travelling Salesman Problem algorithm of Hopfield and Tank”. *Biological Cybernetics*, **58**, 63.