Компьютерная Математика

Математические модели в биологии и медицине

Рассматриваются некоторые технологические процессы разработки нейросетевых моделей диагностики в экспертной системе Гомеопат. Приводятся известные подходы к построению таких моделей.

УДК 681.3

Л.А. КАТЕРИНИЧ

О НЕКОТОРЫХ ФОРМАЛЬНЫХ МОДЕЛЯХ РАЗРАБОТКИ НЕЙРОСЕТЕВЫХ АЛГОРИТМОВ В СИСТЕМЕ ГОМЕОПАТ

Введение. Известно, что автоматизация отдельных технологических процессов разработки программ, осуществляемая в операционных средах, интегрированных средах программирования и специализированных интегрированных средах, значительно повышает производительность, качество и надежность соответствующего программного обеспечения. Этим в значительной степени объясняется научно-практическая заинтересованность специалистов в разработке унифицированных механизмов, моделей, языков и методологий создания интегрированных инструментальных средств поддержки технологических процессов на всех этапах жизненного цикла программ – от проектирования, отладки, документирования и реализации до внедрения и усовершенствования.

Задача автоматизации в такой классической постановке сохраняет свою актуальность и при создании прикладных программных систем нового поколения, в частности, интеллектуальных информационных систем с нейросетевыми и нечеткими моделями представления и обработки данных [1–3].

В данной работе рассматриваются некоторые технологические процессы разработки нейросетевых моделей диагностики в экспертной системе Гомеопат [4, 5] учитывая возможность их автоматизации.

Приводятся известные подходы [6] к построению таких моделей, в частности, путем описания последних оптимизационными схемами.

Классификация. Первым и наиболее распространенным примером задач диагностики является классификация без учителя, общая постановка которой состоит в следующем. Задан набор объектов, каждому объекту сопоставлен вектор значений признаков. Требуется разбить эти объекты на классы эквивалентности.

Отнесение объекта к классу проводится путем его сравнения с типичными элементами разных классов и выбора ближайшего. Простейшая мера близости объектов – квадрат евклидового расстояния между векторами значений их признаков (чем меньше расстояние, тем ближе объекты). Соответствующее определение признаков типичного объекта – среднее арифметическое значение признаков по выборке, представляющей класс. Другая мера близости, естественно возникающая при обработке сигналов, изображений и т.п. – квадрат коэффициента корреляции (чем он больше, тем ближе объекты). Возможны и иные варианты – все зависит от задачи.

Если число классов m заранее определено, то формально задача классификации без учителя в общем случае ставится следующим образом.

Пусть $\{x^p\}$ – векторы значений признаков для рассматриваемых объектов и в пространстве таких векторов определена мера их близости $\rho\{x, y\}$. Для определенности считаем, что чем ближе объекты, тем меньше ρ. С каждым классом связываем его типичный объект. Далее называем его ядром класса. Требуется определить набор из m ядер $y^1, y^2, ..., y^m$ и разбиение $\{x^p\}$ на классы:

$$\{x^p\} = Y_1 \cup Y_2 \cup ... \cup Y_m$$

минимизирующее следующий критерий

$$Q = \sum_{i=1}^{m} D_i \to \min,$$

где для каждого (i-го) класса D_i – сумма расстояний от принадлежащих ему точек выборки до ядра класса:

$$D_i = \sum_{x^p \in Y_i} \rho(x^p, y^i).$$

Минимум Q берется по всем возможным положениям ядер y^i и всем разбиениям $\{x^p\}$на m классов Y_i.

Если число классов заранее не определено, то полезен критерий слияния классов: классы Y_i и Y_j сливаются, если их ядра ближе, чем среднее расстояние от элемента класса до ядра в одном из них.

Использовать критерий слияния классов можно так: сначала принимаем гипотезу о достаточном числе классов, строим их, минимизируя Q, затем некоторые Y_i объединяем, повторяем минимизацию Q с новым числом классов и т. д.

Существует много эвристических алгоритмов классификации без учителя, основанных на использовании мер близости между объектами. Каждый из них

имеет свою область применения, а наиболее распространенным недостатком является отсутствие четкой формализации задачи: совершается переход от идеи кластеризации прямо к алгоритму, в результате неизвестно, что ищется (но что-то в любом случае находится, иногда – неплохо).

Сетевые алгоритмы классификации без учителя строятся на основе итерационного метода динамических ядер.

1. Задается выборка предобработанных векторов данных $\{x^p\}$ (пространство векторов данных обозначается E; каждому классу векторов соответствует некоторое ядро a; пространство ядер обозначается A).

2. Для каждых $x \in E$ и $a \in A$ определяется мера близости $d(x, a)$.

3. Для каждого набора из k ядер $a_1, ..., a_k$ и любого разбиения $\{x^p\}$ на k классов $\{x^p\} = P_1 \cup P_2 \cup ... \cup P_k$ определяется критерий качества

$$D = D(a_1, a_2, ..., a_k, P_1, P_2, ... P_k) = \sum_{i=1}^{k} \sum_{x \in P_i} d(x, a_i).$$

Требуется найти набор $a_1, ..., a_k$ и разбиение $\{x^p\} = P_1 \cup P_2 \cup ... \cup P_k$, минимизирующие D.

Последний шаг алгоритма разбивается на два.

3.1. Для фиксированного набора ядер $a_1, ..., a_k$ ищется минимизирующее критерий качества D разбиение $\{x^p\} = P_1 \cup P_2 \cup ... \cup P_k$; оно дается решающим правилом: $x \in P_i$, если $d(x, a_i) < d(x, a_j)$ при $i \neq j$, в том случае, когда для x минимум $d(x, a)$ достигается при нескольких значениях i, выбор между ними может быть сделан произвольно.

3.2. Для каждого P_i ($i = 1, ..., k$), полученного на шаге 3.1, ищется $a_i \in A$, минимизируещее критерий качества (т. е. слагаемое в D для данного i –

$$D_i = \sum_{x \in P_i} d(x, a_i)).$$

Начальные значения $a_1, ..., a_k$, $\{x^p\} = P_1 \cup P_2 \cup ... \cup P_k$ выбираются произвольно, либо по какому-нибудь эвристическому правилу.

На каждом шаге алгоритма уменьшается критерий качества D, отсюда следует сходимость алгоритма – после конечного числа шагов разбиение $\{x^p\} = P_1 \cup P_2 \cup ... \cup P_k$ уже не меняется.

Если ядру a_i сопоставляется элемент сети, вычисляющий по входному сигналу x функцию $d(x, a_i)$, то решающее правило для классификации состоит в следующем: элемент x принадлежит классу P_i, если выходной сигнал i-го элемента $d(x, a_i)$ меньше всех остальных.

Единственная вычислительная сложность в алгоритме может состоять в поиске ядра по классу, т. е. в поиске $a \in A$, минимизирующего

$$D_i = \sum_{x \in P_i} d(x, a).$$

В связи с этим, в большинстве конкретных реализаций метода мера близости d выбирается такой, чтобы легко можно было найти a, минимизирующее D для данного P .

В простейшем случае пространство ядер A совпадает с пространством векторов x, а мера близости $d(x, a)$ – положительно определенная квадратичная форма от $x - a$, например, квадрат евклидового расстояния или другая положительно определенная квадратичная форма. Тогда ядро a_i, минимизирующее D_i, является центром тяжести класса P_i:

$$a_i = \frac{1}{|P_i|} \sum_{x \in P_i} x ,$$

где $|P_i|$ – число элементов в P_i.

В этом случае также упрощается и решающее правило, разделяющее классы. Пусть $d(x, a) = (x - a, x - a)$ – билинейная форма (если d – квадрат евклидового расстояния между x и a, то билинейная форма – обычное скалярное произведение). В силу билинейности

$$d(x, a) = (x - a, x - a) = (x, x) - 2(x, a) + (a, a).$$

Чтобы сравнить $d(x, a_i)$ для разных i и найти среди них минимальное, достаточно вычислить линейную неоднородную функцию от x:

$$d_1(x, a_i) = (a_i, a_i) - 2(x, a_i).$$

Минимальное значение $d(x, a_i)$ достигается при том же i, что и минимум $d_1(x, a_i)$, поэтому решающее правило реализуется с помощью k сумматоров, вычисляющих $d(x, a)$ и интерпретатора, выбирающего сумматор с минимальным выходным сигналом. Номер этого сумматора и есть номер класса, к которому относится x.

В описанных простейших случаях, когда ядро класса точно определяется как среднее арифметическое (или нормированное среднее арифметическое) элементов класса, а решающее правило основано на сравнении выходных сигналов линейных адаптивных сумматоров, нейронную сеть, реализующую метод динамических ядер, называют сетью Кохонена. В определении ядер a для сетей Кохонена входят суммы

$$\sum_{x \in P} x .$$

Это позволяет накапливать новые динамические ядра, обрабатывая по одному примеру и пересчитывая a_i после появления в P_i нового примера. Сходимость при такой модификации, однако, ухудшается.

Базовый способ использования полученных классификаторов состоит в следующем: для вектора данных x^j и каждого ядра a_i вычисляется $y_i = d(x, a_i)$ (условимся считать, что правильному ядру отвечает максимум d, изменяя, если надо, знак d); строка ответов y_i преобразуется в строку, где только один элемент, соответствующий максимальному y_i, равен 1, остальные – нули. Эта строка и является результатом функционирования сети. По ней может быть определен номер класса (номер места, на котором стоит 1) и другие показатели.

Количественные и качественные характеристики нейронных сетей.

Далее исследуются два вопроса, встающие перед каждым исследователем, решившим использовать нейронные сети: «Сколько нейронов необходимо для решения задачи?» и «Какой должна быть структура нейронной сети?». Объединяя эти два вопроса, мы получаем третий: «Как сделать работу нейронной сети понятной для пользователя (логически прозрачной) и какие выгоды может принести такое понимание?».

При ответе на вопрос «Сколько нейронов нужно использовать?» существует две противоположные точки зрения. Одна из них утверждает, что чем больше нейронов использовать, тем более надежная получится сеть.

Вторая точка зрения опирается на такое «эмпирическое» правило: чем больше подгоночных параметров, тем хуже аппроксимация функции в тех областях, где ее значения были заранее неизвестны.

Подводя итог анализу двух крайних позиций, можно сказать следующее: нужно выбирать число нейронов большим, чем необходимо, но не намного. Это можно осуществить путем удвоения числа нейронов в сети после каждой неудачной попытки обучения. Однако существует более надежный способ оценки минимального числа нейронов – использование процедуры контрастирования [7]. Кроме того, процедура контрастирования позволяет ответить и на второй вопрос: какой должна быть структура сети.

Рассмотрим процедуру контрастирования основанную на оценке показателей чувствительности. Приведем два наиболее широко используемых способа вычисления показателей чувствительности.

Контрастирование на основе оценки. Рассмотрим сеть, правильно решающую все примеры обучающего множества. Обозначим w_p, $p = 1,\ldots,n$ веса всех связей. При обратном функционировании сети по принципу двойственности или методу обратного распространения ошибки сеть вычисляет вектор градиента функции оценки H по весам связей

$$\mathrm{Grad}(H) = \left\{\partial H / \partial w_p\right\}_{p\,=\,1,\,\ldots\,,\,n}.$$

Пусть w^0 – текущий набор весов связей, а оценка текущего примера равна H^0. Тогда в линейном приближении можно записать функцию оценки в точке w как

$$H(w) = H^0 + \sum_{p=1}^{n} \frac{\partial H}{\partial w_p}(w_p - w_p^0).$$

Используя это приближение можно оценить изменение оценки при замене w_p^0 на w_p^* как

$$\chi(p,q) = \left|\frac{\partial H}{\partial w_p}\right| \cdot \left|w_p^* - w_p^0\right|,$$

где q – номер примера обучающего множества, для которого были вычислены оценка и градиент. Величину $\chi(p,q)$ называют показателем чувствительности к замене w_p на w_p^* для примера q. Далее вычисляется показатель чувствительности, не зависящий от номера примера. Для этого можно воспользоваться любой нормой. Обычно используется равномерная норма (максимум модуля):

$$\chi(p) = \max_q \chi(p,q).$$

Умея вычислять показатели чувствительности, можно приступать к процедуре контрастирования. Приведем простейший вариант этой процедуры.

1. Вычисляем показатели чувствительности.
2. Находим минимальный среди показателей чувствительности – χ_{p^*}.
3. Заменяем соответствующий этому показателю чувствительности вес $w_{p^*}^0$ на w_p^*, и исключаем его из процедуры обучения.
4. Предъявляем сети все примеры обучающего множества. Если сеть не допустила ни одной ошибки, то переходим ко второму шагу процедуры.
5. Пытаемся обучить отконтрастированную сеть. Если сеть обучилась безошибочному решению задачи, то переходим к первому шагу процедуры, в противном случае переходим к шестому шагу.
6. Восстанавливаем сеть в состояние до последнего выполнения третьего шага. Если в ходе выполнения шагов со второго по пятый был отконтрастирован хотя бы один вес, (число обучаемых весов изменилось), то переходим к первому шагу. Если ни один вес не был отконтрастирован, то получена минимальная сеть.

Контрастирование без ухудшения. Пусть нам дана только обученная нейронная сеть и обучающее множество. Допустим, что вид функции оценки и процедура обучения нейронной сети неизвестны. В этом случае так же возможно контрастирование сети. Предположим, что данная сеть идеально решает задачу. Тогда нам необходимо так отконтрастировать веса связей, чтобы выходные сигналы сети при решении всех задач изменились не более чем на заданную величину. В этом случае контрастирование весов осуществляется понейронно. На входе каждого нейрона стоит адаптивный сумматор, который суммирует входные сигналы нейрона, умноженные на соответствующие веса связей. Для нейрона наименее чувствительным будет тот вес, который при решении примера даст наименьший вклад в сумму. Обозначив x_p^q входные сигналы рассматриваемого нейрона при решении q-го примера, получаем формулу для показателя чувствительности весов:

$$\chi(p,q) = \left| \left(w_p - w_p^* \right) x_p^q \right|.$$

Аналогично получаем

$$\chi(p) = \left| \left(w_p - w_p^* \right) \right| \max_q \left| x_p^q \right|.$$

В самой процедуре контрастирования есть только одно отличие – вместо проверки на наличие ошибок при предъявлении всех примеров проверяется, что новые выходные сигналы сети отличаются от первоначальных не более чем на заданную величину.

Обобщенная процедура обучения нейронных сетей. Математически процесс обучения можно описать следующим образом.

В процес се функционирования нейронная сеть формирует выходной сигнал Y в соответствии с входным сигналом X, реализуя некоторую функцию $Y = G(X)$.

Пусть решением некоторой задачи является функция $Y = F(X)$, заданная парами входных-выходных данных (X_1, Y_1), (X_2, Y_2), …, (X_n, Y_n), для которых $Y_k = F(X_k)$ $(k = 1, \ldots, n)$.

Обучение состоит в поиске функции G близкой к F в смысле некоторой функции ошибки E.

Если выбраны множество обучающих примеров – пар (X_k,Y_k) $(k = 1, \ldots, n)$ и способ вычисления функции ошибки E, то обучение нейронной сети превращается в задачу многомерной оптимизации, имеющую очень большую размерность. В общем случае обучение нейронной сети это многоэкстремальная невыпуклая задача оптимизации.

Для решения таких задач могут быть использованы следующие алгоритмы:

– алгоритм локальной оптимизации с вычислением частных производных первого порядка;

– алгоритм локальной оптимизации с вычислением частных производных первого и второго порядка;

– стохастические алгоритмы оптимизации;

– алгоритмы глобальной оптимизации.

Выводы. Таким образом, процесс построения нейросетевых моделей решения задач диагностики в системе Гомеопат с учетом рассмотренных методов авоматизации отдельных его стадий выглядит довольно просто и вроде бы не вызывает проблем – необходимо просто реализовывать описанные методы построения нейронных сетей. Точнее если сначала строим нейронную сеть и обучаем ее решению задачи классификации. Потом приводим нейронную сеть к логически прозрачному виду с помощью процедуры контрастирования так, чтобы полученную сетью способность можно было “прочитать”. В более сложных случаях применяются общие методы обучения нейронных сетей на основе многомерных оптимизационных схем.

Л.О. Катеринич

ПРО ДЕЯКІ ФОРМАЛЬНІ МОДЕЛІ РОЗРОБКИ НЕЙРОМЕРЕЖЕВИХ АЛГОРИТМІВ У СИСТЕМІ ГОМЕОПАТ

Розглядаються деякі технологічні процеси розробки нейромережевих моделей діагностики в експертній системі Гомеопат. Наводяться відомі підходи до побудови таких моделей.

L.O. Katerynych

ON SOME FORMAL MODELS OF CREATION OF NEURAL NETWORK ALGORITHMS IN "GOMEOPAT" SYSTEM

Some technological processes of neural network diagnostic model creation in "Gomeopat" diagnostic expert system are considered. The known approaches to construction of such models are given.

1. *Рутковская Д.*, *Пилиньский М.*, *Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы. – М.: Горячая линия – Телеком, 2006. – 452 с.
2. *Круглов В.В.*, *Дли М.И.*, *Голунов Р.Ю.* Нечеткая логика и искусственные нейронные сети: Учеб. пособие. – М.: Издательство физико-математической литературы, 2001. – 224 с.
3. *Jacek Leski.* Systemy neuronowo-rozmyte. – Warszawa: Wydawnictwo Naukowo-Techniczne, 2008. – 686 s.
4. *Катеринич Л., Провотар А.* Синтез нейронных сетей на основе информационных гранул // International Book Series «Information Science and Computing: Advanced Research in Artificial Intelligence». – Sofia, 2008. – V1. – P. 179–182.
5. *Катеринич Л., Провотар А.* Диагностирование на нейронных сетях в системе Гомеопат // XIII-th International Conference: Knowledge Dialogue Solution. – Sofia, 2007. – V 1. – P. 64–68.
6. *Горбань А.Н., Россиев Д.А.* Нейронные сети на персональном компьютере. – Новосибирск: Наука, 1996. – 276 с.
7. *Горбань А.Н.* Обучение нейронных сетей. – М.: СП «ParaGraph», 1990. – 160 с.

Получено 15.12.2009

Об авторе:

Катеринич Лариса Александровна,
асистент кафедры информационных систем факультета кибернетики
Киевского национального университета имени Тараса Шевченко.
e-mail katerynich@unicyb.kiev.ua