Применение нейронных сетей для задач классификации

Решение задачи классификации является одним из важнейших применений нейронных сетей.

Задача классификации представляет собой задачу отнесения образца к одному из нескольких попарно не пересекающихся множеств. Примером таких задач может быть, например, задача определения кредитоспособности клиента банка, медицинские задачи, в которых необходимо определить, например, исход заболевания, решение задач управления портфелем ценных бумаг (продать купить или "придержать" акции в зависимости от ситуации на рынке), задача определения жизнеспособных и склонных к банкротству фирм.

Цель классификации

При решении задач классификации необходимо отнести имеющиеся *статические образцы* (характеристики ситуации на рынке, данные медосмотра, информация о клиенте) к *определенным классам.* Возможно несколько способов представления данных. Наиболее распространенным является способ, при котором образец представляется вектором. Компоненты этого вектора представляют собой различные характеристики образца, которые влияют на принятие решения о том, к какому классу можно отнести данный образец. Например, для медицинских задач в качестве компонентов этого вектора могут быть данные из медицинской карты больного. Таким образом, на основании некоторой информации о примере, необходимо определить, к какому классу его можно отнести. Классификатор таким образом относит объект к одному из классов в соответствии с определенным разбиением N-мерного пространства, которое называется *пространством входов,* и размерность этого пространства является количеством компонент вектора.

Прежде всего, нужно определить уровень сложности системы. В реальных задачах часто возникает ситуация, когда количество образцов ограничено, что осложняет определение сложности задачи. Возможно выделить три основных уровня сложности. Первый (самый простой) – когда классы можно разделить прямыми линиями (или гиперплоскостями, если пространство входов имеет размерность больше двух) – так называемая *линейная разделимость*. Во втором случае классы невозможно разделить линиями (плоскостями), но их возможно отделить с помощью более сложного деления – *нелинейная разделимость*. В третьем случае классы пересекаются и можно говорить только о *вероятностной разделимости*.

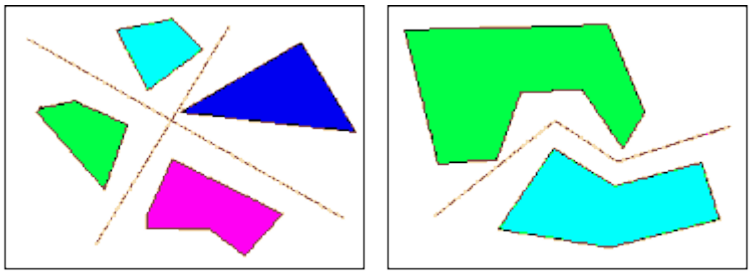

Рис. 1. Линейно и нелинейно разделимые классы

В идеальном варианте после предварительной обработки мы должны получить линейно разделимую задачу, так как после этого значительно упрощается

построение классификатора. К сожалению, при решении реальных задач мы имеем ограниченное количество образцов, на основании которых и производится построение классификатора. При этом мы не можем провести такую предобработку данных, при которой будет достигнута линейная разделимость образцов.

Использование нейронных сетей в качестве классификатора

Сети с прямой связью являются универсальным средством аппроксимации функций, что позволяет их использовать в решении задач классификации. Как правило, нейронные сети оказываются наиболее эффективным способом классификации, потому что генерируют фактически большое число регрессионных моделей (которые используются в решении задач классификации статистическими методами).

К сожалению, в применении нейронных сетей в практических задачах возникает ряд проблем. Во-первых, заранее не известно, какой сложности (размера) может потребоваться сеть для достаточно точной реализации отображения. Эта сложность может оказаться чрезмерно высокой, что потребует сложной архитектуры сетей. Так Минский в своей работе "Персептроны" доказал, что простейшие однослойные нейронные сети способны решать только линейно разделимые задачи. Это ограничение преодолимо при использовании многослойных нейронных сетей. В общем виде можно сказать, что в сети с одним скрытым слоем вектор, соответствующий входному образцу, преобразуется скрытым слоем в некоторое новое пространство, которое может иметь другую размерность, а затем гиперплоскости, соответствующие нейронам выходного слоя, разделяют его на классы. Таким образом сеть распознает не только характеристики исходных данных, но и "характеристики характеристик", сформированные скрытым слоем.

Подготовка исходных данных

Для построения классификатора необходимо определить, какие параметры влияют на принятие решения о том, к какому классу принадлежит образец. При этом могут возникнуть две проблемы. Во-первых, если количество параметров мало, то может возникнуть ситуация, при которой один и тот же набор исходных данных соответствует примерам, находящимся в разных классах. Тогда невозможно обучить нейронную сеть, и система не будет корректно работать (невозможно найти минимум, который соответствует такому набору исходных данных). *Исходные данные обязательно должны быть непротиворечивы*. Для решения этой проблемы необходимо увеличить размерность пространства признаков (количество компонент входного вектора, соответствующего образцу). Но при увеличении размерности пространства признаков может возникнуть ситуация, когда число примеров может стать недостаточным для обучения сети, и она вместо обобщения просто запомнит примеры из обучающей выборки и не сможет корректно функционировать. Таким образом, при определении признаков необходимо найти компромисс с их количеством.

Далее необходимо определить способ представления входных данных для нейронной сети, т.е. определить способ нормирования. Нормировка необходима, поскольку нейронные сети работают с данными, представленными числами в диапазоне 0..1, а исходные данные могут иметь произвольный диапазон или вообще быть нечисловыми данными. При этом возможны различные способы,

начиная от простого линейного преобразования в требуемый диапазон и заканчивая многомерным анализом параметров и нелинейной нормировкой в зависимости от влияния параметров друг на друга.

Кодирование выходных значений

Задача классификации при наличии двух классов может быть решена на сети с одним нейроном в выходном слое, который может принимать одно из двух значений 0 или 1, в зависимости от того, к какому классу принадлежит образец. При наличии нескольких классов возникает проблема, связанная с представлением этих данных для выхода сети. Наиболее простым способом представления выходных данных в таком случае является вектор, компоненты которого соответствуют различным номерам классов. При этом i-я компонента вектора соответствует i-му классу. Все остальные компоненты при этом устанавливаются в 0. Тогда, например, второму классу будет соответствовать 1 на 2 выходе сети и 0 на остальных. При интерпретации результата обычно считается, что номер класса определяется номером выхода сети, на котором появилось максимальное значение. Например, если в сети с тремя выходами мы имеем вектор выходных значений (0.2,0.6,0.4), то мы видим, что максимальное значение имеет вторая компонента вектора, значит класс, к которому относится этот пример, – 2. При таком способе кодирования иногда вводится также понятие *уверенности* сети в том, что пример относится к этому классу. Наиболее простой способ определения уверенности заключается в определении разности между максимальным значением выхода и значением другого выхода, которое является ближайшим к максимальному. Например, для рассмотренного выше примера уверенность сети в том, что пример относится ко второму классу, определится как разность между второй и третьей компонентой вектора и равна 0.6-0.4=0.2. Соответственно чем выше уверенность, тем больше вероятность того, что сеть дала правильный ответ. Этот метод кодирования является самым простым, но не всегда самым оптимальным способом представления данных.

Известны и другие способы. Например, выходной вектор представляет собой номер кластера, записанный в двоичной форме. Тогда при наличии 8 классов нам потребуется вектор из 3 элементов, и, скажем, 3 классу будет соответствовать вектор 011. Но при этом в случае получения неверного значения на одном из выходов мы можем получить неверную классификацию (неверный номер кластера), поэтому имеет смысл увеличить расстояние между двумя кластерами за счет использования кодирования выхода по коду Хемминга, который повысит надежность классификации.

Другой подход состоит в разбиении задачи с k классами на k*(k-1)/2 подзадач с двумя классами (2 на 2 кодирование) каждая. Под подзадачей в данном случае понимается то, что сеть определяет наличие одной из компонент вектора. Т.е. исходный вектор разбивается на группы по два компонента в каждой таким образом, чтобы в них вошли все возможные комбинации компонент выходного вектора. Число этих групп можно определить как количество неупорядоченных выборок по два из исходных компонент. Из комбинаторики

$$A_k^n = \frac{k!}{n!(k-n)!} = \frac{k!}{2!(k-2)!} = \frac{k(k-1)}{2}$$

Тогда, например, для задачи с четырьмя классами мы имеем 6 выходов (подзадач) распределенных следующим образом:

N подзадачи(выхода) КомпонентыВыхода

1	1-2
2	1-3
3	1-4
4	2-3
5	2-4
6	3-4

Где 1 на выходе говорит о наличии одной из компонент. Тогда мы можем перейти к номеру класса по результату расчета сетью следующим образом: определяем, какие комбинации получили единичное (точнее близкое к единице) значение выхода (т.е. какие подзадачи у нас активировались), и считаем, что номер класса будет тот, который вошел в наибольшее количество активированных подзадач (см. таблицу).

N класса	**Акт. Выходы**
1	1,2,3
2	1,4,5
3	2,4,6
4	3,5,6

Это кодирование во многих задачах дает лучший результат, чем классический способ кодирование.

Выбор объема сети

Правильный выбор объема сети имеет большое значение. Построить небольшую и качественную модель часто бывает просто невозможно, а большая модель будет просто запоминать примеры из обучающей выборки и не производить аппроксимацию, что, естественно, приведет к некорректной работе классификатора. Существуют два основных подхода к построению сети – конструктивный и деструктивный. При первом из них вначале берется сеть минимального размера, и постепенно увеличивают ее до достижения требуемой точности. При этом на каждом шаге ее заново обучают. Также существует так называемый метод каскадной корреляции, при котором после окончания эпохи происходит корректировка архитектуры сети с целью минимизации ошибки. При деструктивном подходе вначале берется сеть завышенного объема, и затем из нее удаляются узлы и связи, мало влияющие на решение. При этом полезно помнить следующее правило: *число примеров в обучающем множестве должно быть больше числа настраиваемых весов*. Иначе вместо обобщения сеть просто запомнит данные и утратит способность к классификации – результат будет неопределен для примеров, которые не вошли в обучающую выборку.

Выбор архитектуры сети

При выборе архитектуры сети обычно опробуется несколько конфигураций с различным количеством элементов. При этом основным показателем является объем обучающего множества и обобщающая способность сети. Обычно используется алгоритм обучения Back Propagation (обратного распространения) с подтверждающим множеством.

Алгоритм построения классификатора на основе нейронных сетей

1. Работа с данными
    - Составить базу данных из примеров, характерных для данной задачи
    - Разбить всю совокупность данных на два множества: обучающее и тестовое (возможно разбиение на 3 множества: обучающее, тестовое и подтверждающее).
2. Предварительная обработка
    - Выбрать систему признаков, характерных для данной задачи, и преобразовать данные соответствующим образом для подачи на вход сети (нормировка, стандартизация и т.д.). В результате желательно получить линейно отделяемое пространство множества образцов.
    - Выбрать систему кодирования выходных значений (классическое кодирование, 2 на 2 кодирование и т.д.)
3. Конструирование, обучение и оценка качества сети
    - Выбрать топологию сети: количество слоев, число нейронов в слоях и т.д.
    - Выбрать функцию активации нейронов (например "сигмоида")
    - Выбрать алгоритм обучения сети
    - Оценить качество работы сети на основе подтверждающего множества или другому критерию, оптимизировать архитектуру (уменьшение весов, прореживание пространства признаков)
    - Остановится на варианте сети, который обеспечивает наилучшую способность к обобщению и оценить качество работы по тестовому множеству
4. Использование и диагностика
    - Выяснить степень влияния различных факторов на принимаемое решение (эвристический подход).
    - Убедится, что сеть дает требуемую точность классификации (число неправильно распознанных примеров мало)
    - При необходимости вернутся на этап 2, изменив способ представления образцов или изменив базу данных.
    - Практически использовать сеть для решения задачи.

Для того, чтобы построить качественный классификатор, необходимо иметь качественные данные. Никакой из методов построения классификаторов, основанный на нейронных сетях или статистический, никогда не даст классификатор нужного качества, если имеющийся набор примеров не будет достаточно полным и представительным для той задачи, с которой придется работать системе.