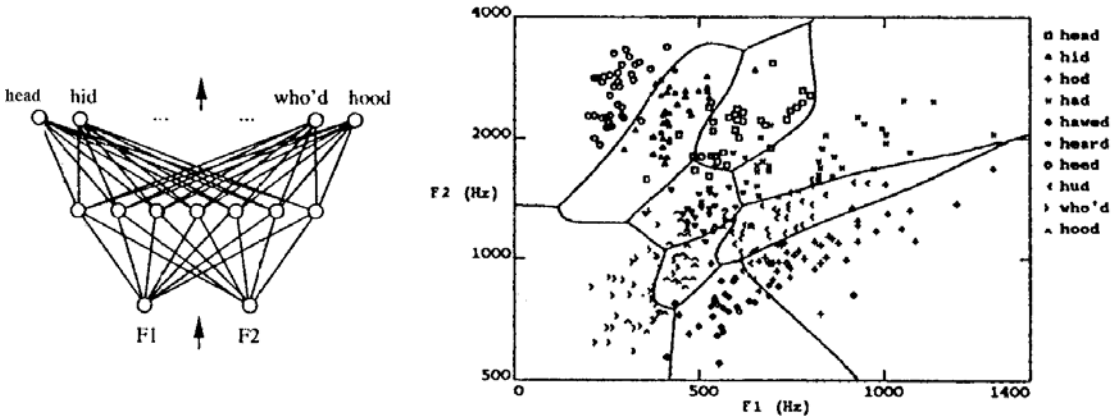
Лекция 8. Многослойни невронни мрежи

В предишната лекция беше показано, че един персептрон може да изразява само линейни повърхнини на решения. В тази лекция ще разгледаме един вид многослойни невронни мрежи, които са способни на изразяват голямо разнообразие от нелинейни повърхнини на решения и ще използват за своето обучение алгоритъм за градиентното спускане, подобен на този, дискутиран в предишната лекция. На Фиг. 8-1 са представени една типична многослойна мрежа и съответната повърхнина на решение. Мрежата е предназначена за решаване на задачата за разпознаване на речта, която, в конкретен случай, включва разграничаване на 10 възможни фонеми, произнасяни в контекста "h_d" (т.е. "hid", "had", "head" и т.н.). Входният речеви сигнал е представен от два числени параметъра, получени чрез спектралния анализ на звука, което позволява лесно да визуализира повърхнината на решение в двумерното пространство на примери. Както е показано на фигурата, тази многослойна мрежа е в състояние да представя силно нелинейни повърхнини на решение, които са значително по-изразителни от линейни повърхнини на решения, съответстващи на един единствен линеен възел.

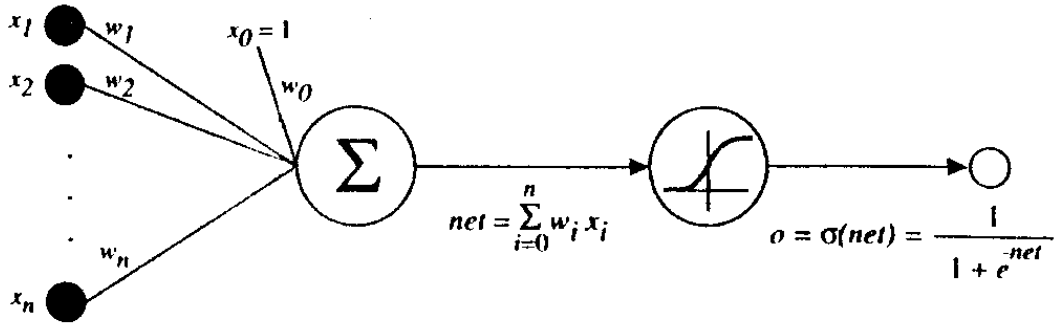


Фиг. 8-1. Области на решения на многослойна невронна мрежа

8.1. Диференцируем прагов възел

Кой тип възли ще използваме като базис за конструиране на многослойни мрежи? Първото впечатление е, че би било естествено да изберем линейните възли, разгледани в предишната лекция, за които имаме изведено правило за обучение. Обаче, комбиниране на линейните възли в няколко слоя ще произвежда само линейни функции, а на нас е за предпочитане да имаме мрежи, способни да представят силно нелинейни функции. Вторият възможен избор е персептрони, но техния не непрекъснатият праг прави ги недиференцируеми и, следователно, неподходящи за градиентното спускане. Това, което ние е нужно, е възел, чийто изход е някоя нелинейна функция на своите входове, но този изход да е също така

диференцируема функция на входовете си. Едно възможно решение и *сигмоиден възел* – възел, който много прилича на персептрон, но се базира върху една гладка, диференцируема прагова функция.



Фиг. 8-2. Сигмоиден прагов възел

Сигмоидният възел е показан на Фиг. 8-2. Както и персептронът, сигмоидният възел отначало изчислява линейната комбинация на своите входове, а след това прилага към резултата определен праг. Обаче в случая на сигмоидния възел праговият изход е непрекъсната функция на своя вход. По-точно, сигмоидният възел изчислява своя изход *o* като: $o = \sigma(\vec{w} \cdot \vec{x})$, където

$$\sigma(y) = \frac{1}{1+e^{-y}} \quad (8.1)$$

σ често наричат *сигмоидна* или *логистична* функция. Обърнете внимание, че нейния изход варира между 0 и 1, монотонно увеличавайки се с увеличаване на нейния вход (виж графика на праговата функция от Фиг. 8-2). Тъй като сигмоидната функция изобразява много голяма входна област в много малък диапазон от изходните стойности, тя често се нарича *смачкващата функция* на възела. Сигмоидната функция има полезното свойство, че нейната производна лесно се изразява в термините на нейния изход, в частност $\frac{d\sigma(y)}{dy} = \sigma(y) \cdot (1 - \sigma(y))$. Понякога вместо σ се използват други диференцируеми функции с лесно изчислими производни. Например, термът e^{-y} от сигмоидната функция понякога се заменя с $e^{-k \cdot y}$, където k е някаква положителна константа, която определя стръмността на прага. Понякога вместо сигмоидната функция се използва и функцията $\tanh = \frac{e^{y} - e^{-y}}{e^{y} + e^{-y}}$.

8.2. Алгоритъм BACKPROPAGATION

Алгоритмът BACKPROPAGATION (т.е. *алгоритъм с обратно разпространение на грешката*) научава теглата на една многослойна мрежа при зададената нейна

топология – фиксираното множество от възли и техните взаимовръзки Той използва градиентното спускане, опитвайки се да минимизира квадратичната грешка между реалните стойностите на изходите на мрежата и техните целеви стойности. Тъй като разглеждаме мрежи с няколко изходни възли, а не само с един такъв, както беше преди, ще започнем с предефиниране на E като сума от изходните възли на мрежата:

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in изходи} (t_{kd} - o_{kd})^2 \quad (8.2)$$

където *изходи* е множеството от изходните възли на мрежата, а t_{kd} и o_{kd} са целевите и изходни стойности, асоциирани с k-ия изходен възел и d-я обучаващ пример.

Проблемът за обучение, стоящ пред алгоритъма, е претърсването на голямо пространство от хипотези, определено от всички възможни тегла на всички възли в мрежата. Ситуацията може да се визуализира в термините на повърхнината на грешката, подобна на тази, използвана при линейните възли. На тази диаграма грешката трябва да бъде заместена с нашата дефиниция на E, а другите размерности на пространството ще съответстват на всички тегла, асоциирани със всички възли на мрежата. Както и в случая на обучение на един единствен възел, градиентното спускане може да се използва за намиране на хипотезата, минимизираща E.

Една от главните разлики за случая с многослойни мрежи е, че повърхнината на грешката сега може да има *няколко* локални минимума в отличие от параболичната повърхност на грешката при един единствен възел, която има само един минимум. За съжаление, това означава, че градиентното спускане гарантира сходимостта само към някой локален минимум, а не задължително – към глобалния. Не зависимо от това, практиката показва, че алгоритмът BACKPROPAGATION дава прекрасни резултати при решаване на множество реални задачи.

Алгоритмът BACKPROPAGATION е представен в Таблица 8-1. Както е описан, алгоритмът се прилага към двуслойна мрежа *с разпространение на активация в права посока* (feed-forward – от входните възли – през скритите – към изходните), съдържаща сигмоидни възли, като възлите от всяко ниво са съединени с всички възли от предишното ниво. Описаният алгоритъм е един инкрементален или стохастичен вариант на градиентното спускане. Използваните означения са същите, като и в предишната лекция със следните разширения:

- Индексът (например някое цяло число) се назначава за всеки възел в мрежата, като "възелът" е или вход в мрежата, или някой от изходи на мрежата.
- x_{ji} означава вход от възела i към възела j, а w_{ji} означава съответното тегло.
- δ_n означава грешката, асоциирана с възела n. Тя играе роля, аналогична на величина *(t – o)* в предишното описание на делта правилото и е равна $\delta_n = -\frac{\partial E}{\partial net_n}$, където net_n е мрежа с изходен възел n.

Алгоритмът започва работата с конструиране на мрежата с желания брой скрити и изходни възли и с инициализацията на теглата на цялата мрежа с малки случайни стойности. При зададената структура на мрежата основният цикъл на алгоритъма повтаря итерациите по обучаващите примери. За всеки обучаващ пример се изчисляват грешката на изхода на мрежата и градиентът по отношение на тази грешка, след което се обновяват всички теглата на мрежата. Тази стъпка на градиентното спускане се повтаря (често хиляди пъти, използвайки многократно едни и същи обучаващи примери), докато мрежата не започне да работи приемливо добре.

BACKPROPAGATION(*обучаващи_примери*, η, $n_{вход}$, $n_{изход}$, $n_{скрити}$)

Всеки обучаващ пример е двойката $<\vec{x},\vec{t}>$, *където* $\vec{x}$ *е вектор на входните стойности на мрежата, а* $\vec{t}$ *е вектор на целевите стойности на изходите на мрежата.*

η е скорост на обучение (например 0.5), $n_{вход}$ *е брой на входните възли,* $n_{изход}$ *е брой на изходни възли, а* $n_{скрити}$ *е брой на скритите възли в мрежата.*

Входът от възела i към възела j се означава с x_{ji}, *а неговото тегло - с* w_{ji}.

- Създай мрежа с разпространение на активация в права посока с $n_{вход}$ входните възли, $n_{изход}$ изходните възли и $n_{скрити}$ скритите възли.
- Инициализирай теглата на цялата мрежа с малки случайни стойности (например, между –0.05 и 0.05).
- **Докато** не бъдат изпълнени условията за спиране **Направи**
  - За всеки $<\vec{x},\vec{t}>$ от *обучаващи_примери* **Направи**

    Разпространение на входове напред по мрежата:

    1. Подай на вход на мрежата примерът $\vec{x}$ и изчисли изходът o_u за всеки възел u в мрежата.

    Обратно разпространение на грешката по мрежата:

    2. За всеки изходен възел k изчисли неговата грешка δ_k

       $\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$ (*)

    3. За всеки скрит възел h изчисли неговата грешка δ_h

       $\delta_h \leftarrow o_h(1-o_h)\cdot \sum_{k\in изходи} w_{kh}\delta_k$ (**)

    4. Обнови всяко тегло на мрежата w_{ji}

       $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$, *където*

       $\Delta w_{ji} = \eta \delta_j x_{ji}$ (***)

Таблица 8-1. Версията на алгоритъма BACKPROPAGATION във вид на стохастическо градиентно спускане за мрежите с разпространение на активация в права посока, съдържащи два слоя от сигмоидни възли.

Правилото за обновяване на теглата чрез метода на градиентното спускане (***) в Таблица 8-1 е подобно на обучаващото делта правило за линейните възли. Както и делта правилото, то обновява всяко тегло пропорционално на скоростта на обучение η, входната стойност x_{ji}, към която теглото е приложено, и изхода на дадения възел. Единствената разлика е, че грешката $(t - o)$ в делта правилото е заместена с по-сложен терм – δ_j. Точната форма на δ_j може лесно да бъде изведена. За да разберем нея на интуитивно ниво, отначало да разгледаме δ_k за всеки *изходен възел* на мрежата k (ф-ла (*)). δ_k е просто познатата ни от делта правилото стойност $(t_k - o_k)$ умножена по фактор $o_k (1 - o_k)$, което е производната от сигмоидната функция. Стойността δ_h за всеки *скрит възел* h на мрежата има подобна форма (ф-ла (**)), обаче тъй като обучаващите примери предоставят целевите стойности t_k само за изходите на мрежата, никакви целеви стойности на изходите на скрити възли директно не са достъпни. Вместо това, грешките на скритите възли δ_h се изчисляват чрез сумиране на грешките δ_k за всеки изходен възел, повлиян от h, претеглени чрез теглото на връзката w_{kh} между скрития възел h и изходния възел k. Това тегло характеризира степента, в която скритият възел h "е отговорен" за грешката в изходния възел k.

Описаният в Таблица 8-1 алгоритъм инкрементално обновява теглата, следвайки представянето на всеки обучаващ пример. Това съответства на стохастическата апроксимация към метода за градиентното спускане. За да получим истинския градиент на E, трябва да сумираме стойностите $\delta_j x_{ji}$ по всички обучаващи примери преди промяната на стойностите на теглата.

Основният цикъл на обновяване на теглата в алгоритъма може да се повтаря хиляди пъти за едно типично приложение. За прекратяване на тази процедура могат да се използват множество различни условия за спиране. Например, може да бъдат избран определен фиксиран брой итерации в цикъла, или когато грешката върху обучаващите примери падне под избрания праг, или когато грешката върху определено отделно множество от тестови примери започва да отговаря на избрания критерий. Изборът на критерия за спиране е много важен, тъй като прекалено малко итерации могат да доведат до това, че грешката няма да спадне достатъчно, а прекалено много итерации могат да доведат до преспецификацията на хипотезата към обучаващите данни.

8.2.1. Добавяне на инерция

Съществуват множество различни вариации на този алгоритъм. Една от най-често използваните е такава промяна на правилото за обновяване на теглата в уравнение (***) на алгоритъма, при която обновяването на теглото на n-та итерация частично зависи от обновяването, изпълнено на (n – 1)-ва итерация:

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \Delta w_{ji}(n-1) \quad (8.3)$$

Тука $\Delta w_{ji}(n)$ е обновяване на теглото, изпълнявано на n-та итерация от основният цикъл, а $0 \leq \alpha < 1$ е константа, наречена *инерция*. Както виждате, първият терм на новото правило е същия, както и в старото; новият терм е втория, така наречения

инерционен терм. За да се разбере ефекта на този терм, нека да представим, че траекторията на градиентното спускане е аналогична на тази, която има една (безинерционна) топка, търкаляща се надолу по повърхнината на грешката. Ефектът от α се състои в добавянето на инерцията, която запазва топка в състоянието на търкаляне в същата посока при преход от една итерация в другата. Понякога, това може да има ефектът на запазването на състоянието на търкалящата топка през малки локални минимуми на повърхността на грешката, или през плоски области от тази повърхнина, където топката би спряла, ако нямаше инерция. Инерцията също така има ефект на постепенно увеличаване на размера на стъпката при търсене в областите, където градиентът не се променя, като по този начин ускорява сходимостта.

8.2.2. Обучение в произволни ациклични мрежи

Дефиницията на алгоритъма BACKPROPAGATION от Таблица 8.1. приложима само към двуслойни мрежи. Обаче, алгоритмът може да бъде лесно обобщен върху мрежи с разпространение на активация в права посока с произволен брой слоеве. Правилото за обновяване на теглата (***) се запазва, а се променя процедурата за изчисляване на стойностите на δ. В общия случай δ_r за възела r от нивото m се изчислява на следващото по дълбочина ниво $m + 1$, съгласно формулата:

$$\delta_r = o_r(1-o_r)\cdot \sum_{s\in ниво_{m+1}} w_{sr}\delta_s \quad (8.4)$$

Както се вижда, тази стъпка е същата като стъпка 3 на алгоритъма от таблица 8.1, затова можем да кажем, че тази стъпка може да бъде повторена за произволен брой скрити слоеве в мрежата.

Алгоритмът сравнително лесно може да бъде обобщен за произволен насочен ацикличен граф, независимо от това, дали възлите на мрежата са подредени в еднакви слоеве, както сме подразбирали по-рано, или не. Ако не, правилото за изчисляване на δ за всеки вътрешен възел (т.е. възел, който не е изход) се задава от формулата:

$$\delta_r = o_r(1-o_r)\cdot \sum_{s\in Наследници(r)} w_{sr}\delta_s \quad (8.5)$$

където *Наследници(r)* са множество от възли в мрежата, входове на които непосредствено включват изхода на възела *r*.

8.3. Сходимостта на алгоритъма BACKPROPAGATION и локални минимуми

И така, алгоритмът BACKPROPAGATION имплементира метод за търсене в пространство на възможни тегла на мрежата чрез градиентното спускане, итеративно намалявайки грешката E между целевите стойностите на обучаващите примери и изходите на мрежата. Тъй като повърхнината на грешката за

многослойни мрежи може да съдържа множество локални минимуми, градиентното спускане може да попадне в един от тях.

Не зависимо от липсата на осигурена сходимост към глобалния минимум на грешката, алгоритмът BACKPROPAGATION е един високо ефективен практически метод за апроксимация на функции. В множество от различни реални приложения проблемът с локалния минимум не се показа като толкова сериозен, както може да се очаква. За да разберем на интуитивно ниво защо е така, да представим, че мрежите с голямо количество тегла съответстват на повърхнини на грешката в пространства с много големи размерности (по една размерност на тегло). Когато градиентното спускане попада в някой локален минимум по отношение на едно от тези тегла, той не е задължително да е локален минимум по отношение на други тегла. Фактически, колко повече има тегла в мрежата, толкова повече ще има размерности, осигуряващи на градиентното спускане "пътища за избягване" от локалния минимум само по едно от теглата.

Другият аспект, който е свързан с проблема за локалните минимуми, е начинът по който се променят теглата на мрежата при нарастване на итерациите по обучаващите примери. Обърнете внимание, че теглата се инициализират с много малки стойности близки до нулата, така че при ранните стъпки на градиентното спускане мрежата представлява една много гладка функция, която приблизително линейна относително своите входове. Това е така, тъй като сигмоидната прагова функция сама по себе си е приблизително линейна, когато теглата са близки към нула. Само след като теглата са имали достатъчно време за нарастване, те достигат точката, където вече могат да представляват силно нелинейни функции. Може да се очаква, че повечето локални минимуми съществуват в области от пространството на теглата, които представят сложни нелинейни функции. Обаче към това време, когато теглата достигнат тези точки, можем да се надаваме, че те вече са достатъчно близки към глобалния минимум, така, че дори локалните минимуми в тази област са приемливи.

Не зависимо от тези разглеждания, поведението на метода за градиентното спускане върху сложни повърхнини на грешката, представяни от ИНМ, все още е слабо разбрано и не съществуват никакви методи за предсказване със сигурност, кога локалните минимуми могат да предизвикат затруднения. Често използваните евристики за избягване на проблема с локалните минимуми са:

- Добавяне на инерционен терм към правилото за обновяване на теглата, как това е показано в уравнение (8.3). Инерцията може понякога да изкара процедурата на градиентното спускане от плитки локални минимуми (макар, че по принцип, тя също така може и да изкара процедурата и от плитък глобален минимум, за да го вкара в друг локален!).
- Използване на стохастичното градиентно спускане вместо истинския градиент. Както вече обсъждахме преди, стохастичната апроксимация към градиентното спускане позволява ефективно спускане по различни повърхнини на грешката за всеки обучаващ пример, разчитайки на средната от тях, за да апроксимира градиента по отношение на пълното обучаващо множество.

- Обучение на различни мрежи с едни и същи данни, но инициализацията на всяка от мрежите да се прави с различни случайни тегла. Ако при различни обучения бъдат достигнати различни локални минимуми, се избира мрежата с най-доброто поведение върху отделно тестово множество. Друг подход е да бъдат запазени всички мрежи, които да се третират като "ансамбъл" от мрежите, чийто изход е (възможно претеглено) средно на изходите на отделни мрежи.

8.4. Изразителната сила на мрежи с разпространение на активация в права посока

Какво множество от функции може да бъде представено чрез мрежи с разпространение на активация в права посока? Отговорът зависи от тяхната ширина и дълбочина. Макар, че още не е известно достатъчно по въпроса, какви класове функции могат да бъдат описани от какви типове мрежи, са получени следните три общи резултата:

- *Булеви функции*. Всяка булева функция може да бъде представена абсолютно точно чрез някаква двуслойна мрежа, макар че в най-лошия случай броят на необходимите скрити възли нараства експоненциално по отношение на броя на входове на мрежата. За да видим, как това може да се направи, да разгледаме следната обща схема за представяне на произволна булева функция: за всеки възможен входен вектор се създава един отделен скрит възел, като неговите тегла се установяват по такъв начин, че възелът се активира тогава и само тогава, когато този специфичен вектор се подава на входа на мрежата. Това създава един скрит слой, който винаги има само един активен възел. Изходният възел се имплементира като порт ИЛИ, който се активира точно за желани входни шаблони.
- *Непрекъснати функции*. Всяка ограничена непрекъсната функция може да бъде апроксимирана с произволно малка грешка от някоя двуслойна мрежа (Cybenko 1989). В този случай теоремата се прилага към мрежите, които използват сигмоидните възли в скрития слой и (безпрагови) линейните възли на изходния слой. Броят на необходимите скрити възли зависи от функцията, която се апроксимира.
- *Произволни функции*. Всяка функция може да бъде апроксимирана с произволна точност от някоя мрежа с три слоя (Cybenko 1988). Отново, изходният слой се състои от линейни възли, а двата скрити слоя използват сигмоидни възли, като броят на необходимите възли във всеки слой в общия случай е неизвестен.

Тези резултати показват, че мрежите с ограничена дълбочина осигуряват едно много изразително пространство от хипотези за алгоритъма BACKPROPAGATION. Обаче важно е да се помни, че тегловните вектори на мрежата, достижими от началните стойности на теглата чрез метода за градиентното спускане, могат да не съдържат всички възможни тегловни вектори.

8.5. Търсене в пространство на хипотези и индуктивното пристрастие

Интересно е да се сравни търсенето в пространство от хипотези, извършвано от BACKPROPAGATION, с това, извършвано от други алгоритми за самообучение. За BACKPROPAGATION всяко възможно назначаване на теглата в мрежата представлява една синтактично различна хипотеза, която по принцип може да бъде разгледана от алгоритъма. С други думи, пространството на хипотези е n-мерното Евклидово пространство от *n* тегла на мрежата. Това пространство е *непрекъснато*, в отличие от пространствата на хипотези при обучение с класификационни дървета и други методи, базирани на дискретното представяне. Този факт, заедно с факта, че E е диференцируема функция по отношение на непрекъснатите параметри на хипотези, води до добре дефиниран градиент на грешката, който осигурява една много полезна структура за организация на търсене на най-добрата хипотеза. Тази структура доста се различава от наредбата от-общото-към-частното, използвана за организацията на търсене в символните алгоритми за обучение, или от наредбата от-простото-към-сложното върху класификационни дървета, използвана от ID3 и C4.5.

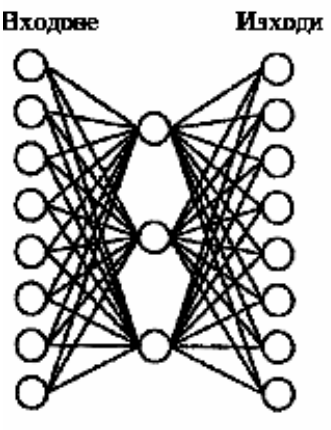
Какво е индуктивното пристрастие, чрез което BACKPROPAGATION обобщава извън наблюдаваните данни? Трудно е точно да характеризираме това пристрастие, тъй като то зависи от взаимодействие между търсене чрез градиентното спускане и начина, по който пространството на теглата обхваща пространство от представими функции. Обаче то може грубо да се характеризира като *гладка интерполация между точки от данни*. При зададени два положителни обучаващи примера без никакви отрицателни примери между тях, BACKPROPAGATION има тенденция да класифицира всички междинни точки също като положителни.

8.6. Представяне на скритите слоеве

Едно интересно свойство на алгоритъма BACKPROPAGATION е способността му да открива полезни междинни представяния на скритите слоеве вътре в мрежата. Тъй като обучаващите примери ограничават само входове и изходи на мрежата, процедурата за нагласяването на теглата е свободна да установява тегла, които дефинират какво представяне на скритите възли е най-ефективно за минимизиране на квадрата на грешката E. Това води до дефиниране в скрития слой на *нови признаци*, които не са зададени явно във входното представяне, но които улавят най-релевантни за научаване на целевата функция свойства на входните примери.

Да разгледаме, като пример, мрежата, показа на Фиг. 8-3. Тука осем входа на мрежата са съединени с три скрити възела, които на свой ред са съединени с осем изходни възела. Тази структура кара трите скрити възела да представят осемте входни стойности по такъв начин, че улавяйки техни релевантни признаци, това

ново представяне (на скритите възли) може да се ползва от изходните възли за изчисляване на коректни целеви стойности.



Вход		Скрити стойности				Изход
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Фиг. 8-3. Наученото представяне на скрития слой

Мрежата, представена на Фиг. 8-3, е предназначена за научаване на една проста целева функция на идентичност $f(\vec{x}) = \vec{x}$, където $\vec{x}$ е вектор, съдържащ седем нули и една единица. Мрежата трябва да се научи да репродуцира осемте входа на съответните осем изхода. Макар, че това е една лесна функция, мрежата, в този случай, е ограничена да използва само три скрити възела. По този начин, важната информация от всички осем входни възела трябва да бъде уловена само от три обучени скрити възела.

При прилагането на алгоритъма BACKPROPAGATION към тази задача с използване на осемте възможни вектори като обучаващи примери, той успешно научава целевата функция. Какво представяне на скритите възли се създава от метода на градиентното спускане, използвано от алгоритъма BACKPROPAGATION? Чрез изследване на стойностите на скритите възли, генерирани от обучената мрежа за всеки от осемте възможни входни вектора, е лесно да се разбере, че наученото кодиране е просто познатото стандартно двоично кодиране на осемте стойности с използване на три бита (например, 001, 010, ..., 111). Точните стойности на скритите възли за едно типично пускане на алгоритъма са показани на Фиг. 8-3.

Способността на многослойните мрежи автоматично да откриват полезно представяне на скритите възли е ключова характеристика на обучение с ИНМ. В отличието от методи са самообучение, които са ограничени да използват само предварително определени признаци, предоставени от човека, мрежите дават една важна степен на гъвкавост, позволяваща на системата да изобретява признаци, които не са представени явно от нейния конструктор – човека. Естествено, тези изобретени признаци трябва да бъдат изчислени като функциите на сигмоидните възли на входовете в мрежата. Колко повече нива са използвани в мрежата, толкова по-сложни признаци могат да бъдат открити.

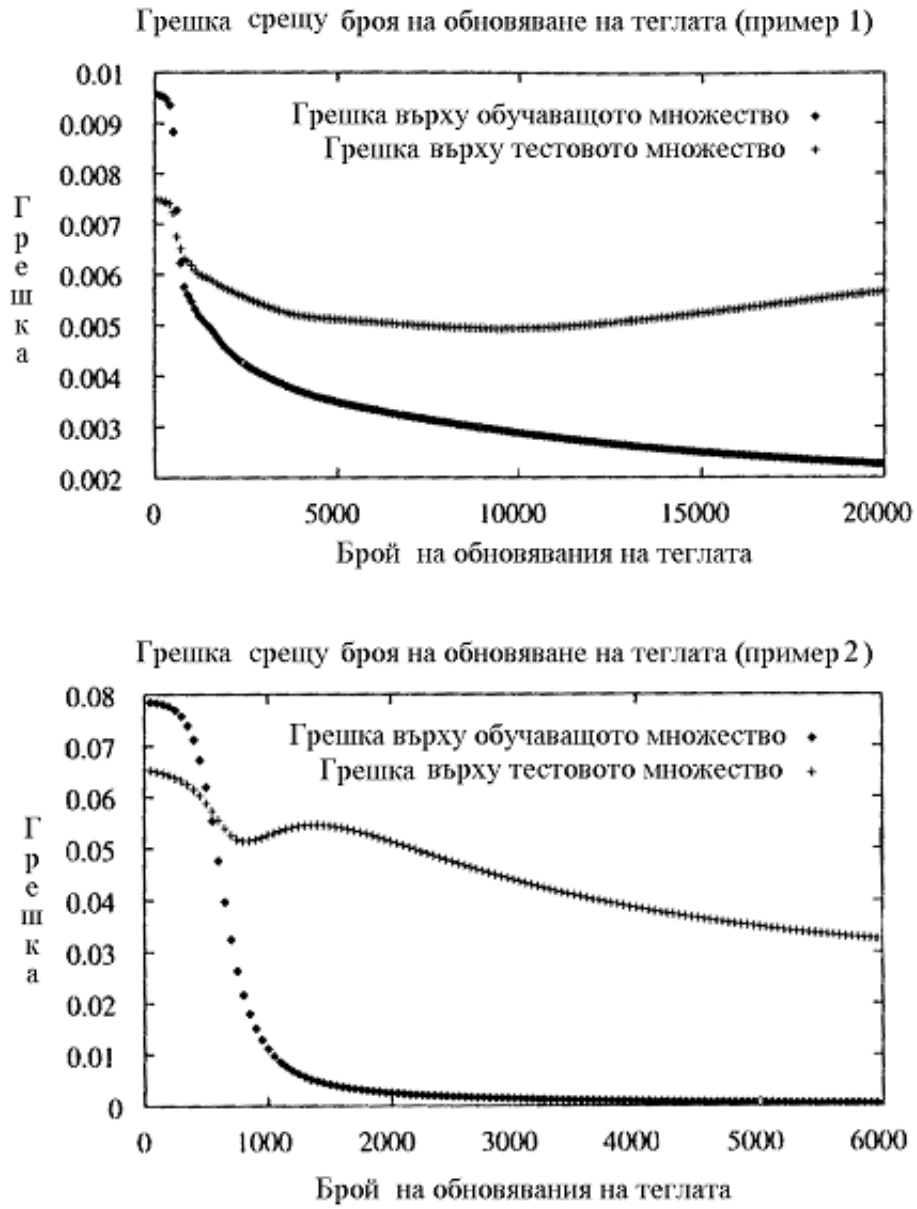
8.7. Обобщение, преспецифициране и критерий за спиране

В описанието на алгоритъма BACKPROPAGATION в Таблица 8-1 условията за спиране на алгоритъма не са указани явно. Какви са подходящи условия за спиране на цикъла за обновяване на теглата? Един очевиден избор е да продължим обучението докато грешката E върху обучаващите примери не падне под някакъв предварително определен праг. На практика това е една доста слаба стратегия, тъй като алгоритмът е чувствителен към преспецифициране към обучаващите примери за сметка на намаляване на класификационната точност върху нови тестови примери.

За да се види опасността от минимизацията на грешката върху обучаващите примери, да разгледаме как тя се променя с броя на итерациите на теглата. Фиг. 8-4. показва тази промяна за два типични приложения на алгоритъма. На горната картинка по-долната от двете линии показва монотонно намаляване на грешката E върху обучаващите примери при нарастване на броя на итерации на градиентното спускане. Горната линия показва грешката E, измерена върху едно независимо тестово множество от примери. Тази линия измерва класификационната (обобщаващата) точност на мрежата – точността, с която мрежата покрива примери, отлични от вече известни обучаващи примери.

Както се вижда, класификационната точност върху тестовите примери отначало намалява, а след това нараства, дори когато грешката върху обучаващите примери продължава да намалява. Причината е преспецификацията на теглата на мрежата (количеството на които е достатъчно голямо), в резултат на което тя започва да се нагажда прекалено точно към прекалено много примери.

Защо тази преспецификация се появява при по-късни итерации, а не в началото? Да припомним, че теглата се инициализират със случайни малки числа. Когато теглата са приблизително еднакви, чрез тях могат да бъдат описани само много гладки повърхнини на решения. При напредването на процеса на обучение някои от теглата започват да нарастват, за да намалят грешката върху обучаващите данни, и сложността на научаваната повърхност на решения започва да нараства. Така ефективната сложност на хипотезите, които могат да бъдат постигнати от алгоритъма, се увеличава с броя на итерациите по настройка на теглата. При достатъчно голям брой на такива итерации алгоритмът е често в състояние да създаде прекалено сложни повърхнини на решения, за да покрие и зашумени обучаващи данни или непредставителни характеристики на определени обучаващи примери. Този проблем с преспецификацията е аналогичен на проблема за преспецификацията при обучение с класификационни дървета.



Фиг. 8-4. Графики на грешката E като функцията на брой на обновяване на теглата за два приложения.

Съществуват няколко техники, адресиращи проблема с преспецификацията на алгоритъма BACKPROPAGATION. Една от тях, известна като *отслабване на теглата,* се състои в намаляване на всяко тегло с определен малък фактор при всяка итерация. Това е еквивалентно на модифициране на дефиницията на *E* чрез включването на определен наказателен терм, съответстващ на общия размер на всички тегла в мрежата. Мотивацията на този подход е да се пазят теглата малки, давайки предпочитания за научаване на по-малко сложни повърхнини на решения.

Един от най-удачните методи за борба с преспецификацията е използване на отделно тестово множество от примери. Алгоритмът следи грешката по отношение на това множество, използвайки обучаващото множество за реализацията на градиентното спускане. При една типична имплементация на този подход се пазят две копия от теглата на мрежата: едно копие е за обучение, а другото – са теглата, при които е постигнато най-доброто до сега поведение на мрежата относително тестовото множество. Когато текущи тегла на мрежата, изчислени на базата на минимизацията на грешка върху обучаващите примери, започват да дават значително по-голяма грешка от мрежата с теглата, даващи най-малката грешка върху тестовото множество, процесът на обучението на мрежата се прекратява и се избират като резултат теглата, осигуряващи най-доброто поведение относително тестовото множество. Например, когато тази процедура се приложи към пример 1 от Фиг. 8-4, финалните тегла на мрежата се получават след 9100 итерации. Втората графика показва, че не винаги е ясно, кога се достига най-малката грешка върху тестовото множество. Виждаме, че тази грешка отначало намалява, след това се увеличава, а след това отново намалява. Трябва да се внимава, за да бъде избегнато грешното решение да бъде спряно обучение след 850 итерации.

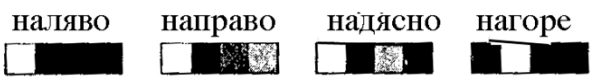
Както винаги, при едно достатъчно голямо количество данни добрите резултати се постигат чрез използване на отделно тестово множество. За съжаление, проблемът с преспецификацията най-много се проявява при малки обучаващи множества. В тези случаи се използва процедурата за k-крос-валидацията, при която наличното множество от обучаващите примери се разбива на k непресичащи се подмножества. След това мрежата се обучава k пъти, всеки път използвайки k – 1 подмножества за обучение и 1 подмножество – за тестване. При всяко от тези k сеанса на обучението на мрежата се определя броят на итерации i, при които се постига най-доброто поведение относно текущото тестово множество. На тази база се изчислява средния брой итерации $\bar{i}$. Финалното пускане на алгоритъма се изпълнява без всякакво тестово множество, използвайки за обучение *всички обучаващи примери*, като обучението се спира след $\bar{i}$ на брой итерации.

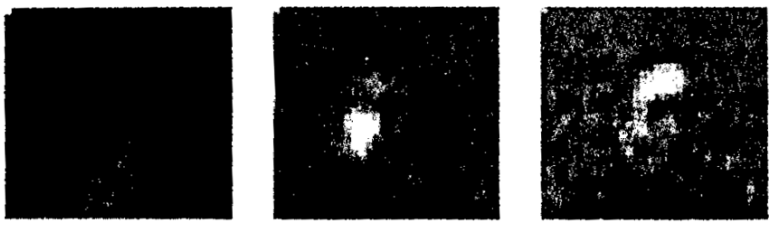
8.8. Пример: разпознаване на лица

За илюстрация на практическите решения, използвани при прилагане на алгоритъма BACKPROPAGATION към решаване на реални задачи, ще разгледаме задачата за разпознаване на лица. Всички образи и кодове на програмата, използвани за създаване на примери, описани в този раздел, както и документация по използването на кодове, се намират в http://www.cs.edu/~tom/mbook.html,.

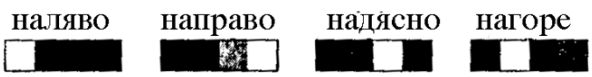
8.8.1. Задачата

Входни изображения с резолюция 30 x 32





Теглата на мрежата след 1 итерация по всички обучаващи примери



Теглата на мрежата след 100 итерации по всички обучаващи примери

Фиг. 8-5. Обучение на невронна мрежа да разпознава положение на лице.

Тази 960 x 3 x 4 мрежа се обучава върху изображения на лица в гамата на сиво (горния ред), за да предсказва, дали човекът гледа наляво, напред, нагоре или надясно. Теглата на обучената мрежа са показани след една итерация по всички обучаващи примери и след сто такива итерации. Всеки изходен възел (наляво, напред, нагоре или надясно) има четири тегла, показани чрез тъмни (отрицателни) и светли (положителни) блокчета. Най-лявото блокче съответства на прага на възела (w_0), а останалите три блокчета отдясно отговарят на теглата на входове от три скрити възела. Показани са теглата от всеки пиксел на изображението към всеки скрит възел, като всяко тегло е показано в позицията, отговаряща на пиксела на изображението.

Задачата за обучение се състои в класификацията на лица на различни хора в различни пози чрез снимките, направени от камера. В експеримента са били

събрани снимките на 20 човека, по приблизително 32 снимки за един човек, с различни изражения на лицата (тъжни, весели, сърдити и т.н.), както и в различни пози (анфас, ляв профил, десен профил, вдигнато лице нагоре и т.н.), а също така със слънчеви очила и без. Както се вижда от примерите на снимки (фигура 8.5), има също и вариации във фона зад лицето, в облеклото и положението на лицето на човека върху снимката. Бяха събрани общо 624 снимки в скалата на сивото с резолюция 120 x 128 пиксела всяка, като всеки пиксел от снимката се описваше с някаква стойност на интензивността на сивото между 0 (черно) и 255 (бяло).

От тези данни могат да се научат множество различни целеви функции. Например, при зададена снимка като вход можем да обучим ИНМ да извежда идентичността на човека, направление, в което е обърнато неговото лице, пол на човека, дали той носи слънчеви очила и т.н. Всички тези целеви функции могат да бъдат научени с висока степен на точност. В настоящия пример ще разгледаме само една конкретна задача: научаване на направление, в което е обърнато лицето на човека (наляво, надясно, право напред или нагоре).

8.8.2. Решения, направени при дизайна на системата

При прилагане на алгоритъма BACKPROPAGATION към някоя конкретна задача, трябва да бъдат взети множество различни решения за неговото конфигуриране. Макар, че не е бил направен опит за оптимален избор за нашата задача, описаните в раздела решения водят до много добро научаване на целевата функция. След обучение върху 260 снимки, класификационната точност върху отделното тестово множество е 90%. За сравнение, точността по премълчаване, която се постига при случайно налучкване на една от четири възможни посоки на лицето, е 25%.

Кодиране на входове

Приемайки, че вход към ИНМ трябва да бъде някое представяне на снимката, едно от ключовите решения е да се избере, как да бъде кодирана тази снимка. Например, можем да подготвим предварително снимката чрез извличане на ръбове, на области с еднаква интензивност или някакви други локални характеристики на снимката, а след това да използваме тези признаци като входове в мрежата. Една от трудностите с подобно решение е, че то ще води до променлив брой на входните признаци (например ръбове) в една снимка, докато ИНМ има фиксиран брой входове. По тази причина направеният в този случай изборът е да кодираме снимката като едно фиксирано множество от 30 x 32 пикселни стойности на интензивността, с по един вход от мрежата за всеки пиксел. Пикселните стойности на интензивността, вариращи от 0 до 255, бяха линейно скалирани в диапазон от 0 до 1, така че входовете на мрежата да приемат стойностите в същия диапазон като и активациите на скритите и изходните възли. Фактически, получената 30 x 32 пиксела снимка представлява резюме на оригиналната 120 x 128 пикселна снимка, направена с груба резолюция, като интензивността на всеки груб пиксел се изчислява като средната стойност на съответните интензивности на пиксели с

висока резолюция. Използването на такива снимки в груба резолюция съществено намалява броя на входове и тегла в мрежата, намалявайки по този начин и изисквания към изчислителните ресурси.

Кодиране на изходи

ИНМ трябва да извежда една от четирите стойности, указващи направление, в което е обърнато лицето. Подобното 4-ри степенно кодиране може да бъде постигнато с използване само на един изход, назначавайки, например, стойности 0.2, 0.4, 0.6 и 0.8 за кодиране на тези възможни направления. Вместо това, ще използваме четири отделни изходни възела, като всеки ще представлява една от четири възможни стойности, като най-голямата стойност ще се използва за предсказване. Тази схема често се нарича *изходното кодиране 1 от n*. Мотивацията за този избор е следната: 1) тя дава повече степени на свобода на мрежата за представяне на целевата функция (т.е. има n пъти повече тегла в изходния слой от възли). 2) при кодирането *1 от n* разликата между изхода с най-голямата стойност и втория по величина изход може да бъде използвана като мярката за убедеността на мрежата в своето предсказване (неопределени класификации могат да се проявяват чрез почти еднакви или съвпадащи стойности).

Следващият избор трябва да бъде направен за начина на кодиране на целевите стойности на четири изходни възела. Един очевиден избор е използването на четири целеви стойности <1, 0, 0, 0> за кодиране на лице, обърнато наляво, <0, 1, 0, 0> - напред, и т.н. Обаче, вместо 0 и 1 ще използваме стойности 0.1 и 0.9, така че <0.9, 0.1, 0.1, 0.1> е целевия изходен вектор за кодиране на лице, обърнато наляво. Причината за този избор е, че сигмоидната функция не може да произвежда изходите 0 и 1 при никакви крайни стойности на тегла. Ако ще опитаме да обучаваме мрежата, за да покрива точно целевите функции 0 и 1, градиентното спускане ще накара теглата да растат без ограничения. От другата страна, стойностите 0.1 и 0.9 са достижими при използването на сигмоидните възли с крайните тегла.

Топологичната структура на мрежа

Както вече знаете, алгоритъм BACKPROPAGATION може да бъде приложен към произволен ацикличен насочен граф от сигмоидни възли. Следователно, следващия избор е колко възли трябва да бъдат включени в мрежата и как те да бъдат свързани. Най-често използваната структура на ИНМ е слоестата структура с преки връзки между всеки възел от един слой с всеки възел от следващия слой. Ние ще използваме тази структура, съставена от два слоя сигмоидните възли (един скрит и един изходен слой). Не е прието да се използва повече от три слоя сигмоидни възли, тъй като 1) времето за обучение на такава мрежа става много дълго и 2) защото мрежата с три слоя от сигмоидните възли вече може да изрази едно много богато множество от целевите функции. Следващия избор е колко скрити възли трябва да използваме. Резултати, показани на Фигура 8.5, са получени при използване само на три скрити възела, постигайки 90% точност върху тестовото

множество. При други експерименти са използвани 30 скрити възела, постигайки точността с два процента по-голяма. Макар, че разликата в точността е доста малка, времето за обучение на втората мрежа е значително по-голямо от първата. Използвайки 260 обучаващи изображения, времето за обучение е било приблизително 1 час за работната станция Sun Spark 5 с 30 скрити възела в сравнение с приблизително 5 минути за 3 скрити възела. Показано е, че за различни приложения е необходим определен минимум от скрити възли, за да бъде научена целевата функция достатъчно точно. Добавянето на повече скрити възли след това не води до значително подобряване на класификационната точност при условие, че за определяне на необходимия брой на итерации във всеки случай се използва крос-валидацията. Ако този метод не се прилага, увеличаването на броя на скритите възли често води до ефекта на преспецифициране към обучаващите данни, т.е. към намаляване на класификационната точност върху тестови примери.

Други параметри на алгоритъма

При експериментите скоростта на обучение η е избрана равна на 0.3, а инерцията α - на 0.3 също. По-малките стойности на двата параметъра водели до приблизително същата класификационна точност, но за значително по-дълго време на обучение. Ако тези стойности са били установявани прекалено големи, обучението пропадаше в опита за сходимост към мрежата с приемливото ниво на грешката върху обучаващото множество. В експериментите е бил използван метод на пълния градиент (в отличие от стохастическата апроксимация на градиентното спускане, описана в таблица 8.1). Теглата на мрежата в изходните възли са били инициализирани чрез случайни малки стойности, обаче теглата на входни възли са инициализирани с нули, тъй като това е давало по-интелигентна визуализация на научените тегла, без значимо влияние върху класификационната точност. Броят на обучаващи итерации е бил избран чрез разделяне на наличните данни на обучаващото и тестовото множества. Градиентното спускане е било използвано за минимизиране на грешката върху обучаващото множество, като след всеки 50 итерациите на спускането поведението на мрежата е било оценявано върху тестовото множество. Финалната мрежа е тази с най-висока точност върху тестовото множество. Докладваната крайна точност (90%) е била измерена върху третото, отделно множество от тестови примери, които по никакъв начин не са били използвани при обучение на мрежата.

8.8.3. Наученото представяне на скритите възли

Да разгледаме по-детайлно структурата на научените 2899 тегла на мрежата (виж Фиг. 8-5). Отначало да разгледаме първите четири правоъгълни блокчета, разположени под изображенията на лицата. Всеки от тях изобразява теглата на един от четири изходни възли на мрежата (кодиращи направления наляво, напред, надясно и нагоре). Четири квадрата във всеки от правоъгълниците указват четири тегла, асоциирани с този изходен възел – теглото w_0, определящо прага на възела (най левия квадрат), и трите тегла, съединяващи трите скритите възли с този

изходен възел. Яркостта на квадратчето съответства на стойността на теглото, като ярко бяло указва на голямо положително тегло, тъмно черно – на голямо отрицателно тегло, а междинното сиво – на междинните стойности на теглата. Например, изходният възел, означен с "нагоре", има праг w_0 близък до нула, едно голямо положително тегло от първия скрит възел и едно голямо отрицателно тегло от вторият скрит възел.

Теглата на скритите възли са показани непосредствено под тези на изходните възли. Напомням, че всеки скрит възел получава на вход всеки от 30 x 32 пиксела на изображение. 30 x 32 тегла, асоциирани с тези изходи, са показани по такъв начин, че всяко тегло е разположено в позиция на съответния пиксел от изображението (с праговото тегло w_0, наложено отгоре в горната лява част на масива).

Стойностите на теглата на мрежата след 100 итерации по градиентното спускане са показани в долната част на фигурата. Обърнете внимание, че най-левият скрит възел има доста различни тегла по сравнение с това, което той имал след една итерация. Другите два скрити възела също се промениха. Кодирането на този краен набор от теглата може до някъде да бъде разбрано. Например, да разгледаме изходният възел, който указва, че човекът гледа надясно. Този възел има голямо положително тегло от втория скрит възел и силно отрицателно тегло от третия скрит възел. Анализирайки теглата на тези два скрити възела, лесно е да се види, че ако лицето на човека е обърнато надясно (нашето ляво), то неговата светла кожа приблизително съвпада по място с силни положителни тегла на този скрит възел, а неговата тъмна коса приблизително съответства на места с отрицателните тегла. Същото изображение кара третият скрит възел да извежда стойността, близка до нула, тъй като в този случай светлото лице има тенденция да отговаря по положение на големи отрицателни тегла.