

ЮГОЗАПАДЕН УНИВЕРСИТЕТ "НЕОФИТ РИЛСКИ"

Невронни мрежи

Иван Тренчев, Петър Миланов,

Невена Пенчева, Иван Мирчев

Работен вариант

Невронни мрежи – Въведение и Процеси на обучение

1.1. Същност на невронните връзки

Изследванията по изкуствените невронни връзки са свързани с това, че способа на обработването на информацията от човешкият мозък коренно се отличава от методите, които се използват от цифровите компютри.

Мозъкът представлява много сложен не линеен успореден компютър (система за обработка на информация). Той е способен да организира свои структури компоненти наречени неврони, така че те биха могли да изпълняват конкретни задачи (разпознаване на образи, обработване на сигналите на органите за чувство, моторни функции), много пъти по-бързо и от най-бързите съвременни компютри.

Пример за решение на такава задача от мозъка може да бъде обработката на информация от *обикновеното зрение* (human vision). Във функциите на зрителната система влизат създаване на представа за околната среда в такъв вид, който обезпечава възможността на взаимодействие (interact) в този свят. Още по-точно последователно изпълнява редица задачи за разпознаване (например разпознаване на познато лице в непозната обкръжение). За това нещо отиват около 100 – 200 мили секунди. В същото време за изпълнението на аналогични задачи даже с по-малка сложност изпълнявани от компютъра може да отнеме няколко дни.

Друг пример може да бъде *локатора* (sonar) на прилепите, който представлява система на активна ехолокация.

Понятието развитие на невроните е свързано с понятието *пластичност* (plasticity) на мозъка – способността на настройване на нервната система към обкръжаващата ни среда. Пластичността на мозъка играе най-важна роля в работата на невроните в качеството на единица обработка на информация в човешкият мозък. Аналогично в изкуствените невронни мрежи работата се провежда в изкуствени неврони. В двата случая *невронната мрежа* (neural network) представлява машина за моделиране на способ за обработване на мозъка за конкретна задача. Тази мрежа обикновено се реализира с помощта на електронни компоненти или се моделира от програмен цифров компютър. За да се получи висока производителност, невронните мрежи използват множество взаимосвързани елементарни клетки на изчисление – невроните. По този начин може да се формулира следното определение за невронни мрежи произлизашо от ролята на адаптивната машина.

Невронни мрежи – това е огромен разпределен успореден процесор, състоящ се от елементарни единици за обработка на информацията получил експериментални стойности, които се ползват за следваща обработка. Невронната мрежа е сходна с мозъка от две гледни точки:

- Знанията постъпват в невронната мрежа от околната среда и се използват в процеса на обучение;

- За натрупване на знания се прилагат връзки между невроните – наричайки ги синаптично тегло.

Процедурите използвани за процеса на *обучение* (learning) се наричат *алгоритми за обучение* (learning algorithm). Тази процедура изгражда в определен ред синаптическо тегло на невронната мрежа обезпечаващо необходимите структури на връзките между невроните. Изменението на синаптическото тегло представлява традиционен метод за настройване на невронната мрежа. Този подход е много близък до теорията на линейните адаптивни филтри, която се използва от човека. Невронните мрежи могат да изменят собствената си топология. Това се обуславя от факта, че невроните в човешкият мозък постоянно умират, а нови синаптически връзки постоянно се създават. В литературата невронните мрежи често ги наричат нервнокомпютърни мрежи (connectionist network) за връзка на успоредно разпределени процесори.

Предимство на невронните мрежи

Своята сила невронните мрежи черпят от разклонената обработка на информацията и от способността за самообучение за създаване на обобщение. Под термина *обобщение* (generalization) се разбира способността за получаване на обоснован резултат на основата на данни, които не са се срещали в процеса на обучение. Тези свойства позволяват на невронните мрежи да решават сложни (мащабни) задачи, които за даденият момент се считат трудно разрешими. Обаче напрактика автономната работа на невронните мрежи не може да обезпечава готови решения. Необходимо е те да се интегрират в сложни системи. В частност комплексна задача може да се раздели на последователности (относително прости), част от които може да се решават от невронната мрежа. За създаването на компютрна архитектура, която да бъде способна да имитира човешкият мозък (ако е възможна изобщо) ще тряба да се измине дълъг и труден път.

Използването на невронните мрежи обезпечава следните полезни свойства на системата:

- 1) *Нелинейност* (nonlinearity) – изкуствените неврони могат да бъдат линейни и нелинейни. Невронните мрежи, построени от съединени нелинейни неврони се явяват нелинейни. Тази нелинейност е от особен вид, тъй като е *разпределена* (distributed) по мрежата. Нелинейността е много важно свойство особено ако физическият механизъм отговарящ за формиране на входният сигнал също е нелинеен (например човешката реч).
- 2) *Отразяване на входната информация в изходна* (input-output mapping) – една от популярните парадигми се явява *обучение с учител* (supervised learning). Това включва изменение на синаптическите тегла на основа на избор на маркирани *учебни примери* (training sample). Всеки пример се състои от входен сигнал и съответен на него *желан отговор* (desired response). От това множество случаи образи се избира пример, а невронната мрежа модифицира синаптическите тегла за минимизация на разклонение на желания изходен сигнал и формулираната мрежа съгласно избран статистически критерии. При това собствено се модифицират *свободни параметри* (free parameters) на невронната мрежа. По-рано използваните примери в последствие могат да се прилагат отново, но вече в друг ред. Това обучение се провежда до тогава

докато изменението на синаптическите тегла не стане незначително. По този начин невронната мрежа се обучава на примери съставяйки таблици на съответният вход и изход за конкретна задача. Такъв подход ни кара да си спомним *непараметрическо статистическо обучение* (nonparametric statistical inference). Това направление в статистиката има работа с оценките, които не са свързани с конкретни модели или от биологична гледна точка, с обучението с нула. Тук термина „непараметрически“ се използва за акцентиране на това, че от начало не съществува никакъв предопределен статистически модел за входни данни. За пример ще разгледаме задачата за *класификация на образите* (pattern classification). В нея се изиска да се съотнесе входния сигнал, представляващ физически обект или събитие с някои предопределени категории (класове). При непараметричният подход при решаването на тази задача трябва да се оцени рамка за решение в пространството на входния сигнал на основата на избора на примери. При това не се използва никакъв вероятностен модел на разпределение. Аналогичен подход се прилага в параметричното обучение с учител. Това още веднъж подчертава успоредността между отразяването на входните сигнали в изходни, осъществено от невронната мрежа и непараметрическото статистическо обучение.

- 3) *Адаптивност* (adaptivity) – невронните мрежи имат способността да адаптират своите синаптически тегла към околната среда. В частност невронните мрежи обучени да работят в една среда могат лесно да бъдат пробудени да работят в условия с незначителни колебания на параметрите на средата. При работа в нестационарна (nonstationary) среда (където статистически се изменя с течение на времето) могат да бъдат създадени невронни мрежи изменящи синаптическото си тегло в реално време. За да се използват всички достойнства на адаптивността основните параметри на системата трябва да са достатъчно стабилни, за да работят неочаквани външни въздействия и да бъдат достатъчно гъвкави обезпечавайки реакцията на съществено изменение на средата. Тази задача обикновено се нарича *Дileма на стабилност–пластичност* (stability-plasticity dilemma).
- 4) *Очевидност на отговора* (evidential response). В контекса на задачите за класификация на образи може да се разработи невронна мрежа, събираща информация не само за определен конкретен клас, но и за увеличение на достоверността на взетото решение. Тази информация може да се използва за изключване на съмнителни решения, което повишава продуктивността на невронната мрежа.
- 5) *Контекстна информация* (contextual information) – Знанията се представят в самата структура на невронната мрежа с помощта на нейното състояние на активност. Всеки неврон от невронната мрежа може да бъде подложен на влияние от всички останали неврони. Като следствие съществуват невронни мрежи свързани с конкретна информация.
- 6) *Отказоустойчивост* (fault tolerance) – Невронните мрежи облечени във форма на електроника са потенциално отказоустойчиви, т.е. при неблагоприятни условия тяхната производителност пада незначително. Например ако е повреден някой от невроните или неговата връзка, извлечането на запаметената информация се затруднява. Очтайки разпределителната характеристика на съхранение на информацията в невронните мрежи може да се потвърди, че само сериозни повреди в структурата на невронните мрежи съществено ще повлияе на нейната работоспособност. Затова понижаването на качеството на работа на невронната

мрежа протича бавно. Незначителното повреждане на структурата никога не оказва катастрофални последствия.

- 7) *Мащабируемост* (VLSI Implementability) – Успоредната структура на невронните мрежи потенциално ускорява решението на някои задачи и обезпечава мащабността на невронните мрежи в рамките на технологията VLSI (very-large-scale-integrated). Едно от предимствата ѝ е възможността да се представи достатъчно сложно поведение чрез йерархичния структура.
- 8) *Еднообразие на анализа и проектирането* (Uniformity of analysis and design). Невронните мрежи са универсален механичъм за обработка на информация, т.е. едно и също проектно решение на невронната мрежа може да се използва в много различни области. Това свойство се проявява по няколко начина:
 - невроните в тази ли друга форма са стандартни съставни части на всяка невронна мрежа;
 - тази общност позволява да се използват едни и същи теории и алгоритми за обучение в различни невронни мрежови приложения;
 - модулната мрежа може да бъде построена на основата на интеграция на цели модули.
- 9) *Аналогия с невробиологията* (Neurobiological analogy). Построяването на невронни връзки се определя чрез аналогията с човешкият мозък, който се явява живо доказателство за това, че отказоустойчиви успоредни изчисления не само физически са реализирани, но и се явяват бърз и мощен инструмент за решаване на различни задачи. Невробиологията разглежда изкуствените невронни връзки като средство за моделиране на физически явления. От друга страна инженерите постоянно се опитват да подчертават на невробиолозите нови идеи, произлизящи от традиционните електронни схеми. Със следните примери ще посочим тези две различни гледни точки:
 - Работата на моделите на линейните системи с вестибуло-окулярен рефлекс се сравнява с моделите на *рекурентни невронни връзки*. *Вестибуло-окулярен рефлекс* или рефлекса VOR (vestibule-ocular reflex) се явява съставна част в очнодвигателната система. Неговата задача е да обезпечава стабилност на визуалният образ при завъртане на главата за сметка на завъртане на очите. Процеса VOR се реализира чрез премоторни неврони във вестибуларният център, който получава и обработените сигнали за завъртане на главата от вестибуларните сензорни бутони и предава резултата на моторните неврони на окото. Механизмът VOR е подходящ за моделиране както на входни (завъртане на главата), така и на изходни (завъртане на очите) сигнали, които се описват много точно. Освен това много прост рефлекс, а невронно физическите свойства реализации неговите неврони са добре описани. С появата на невронните модели ситуацията коренно се променя. Рекурентните модели VOR (програми използващи алгоритми за рекурентно обучение в реално време) позволяват да се възпроизведат и описват много статични, динамични, нелинейни и разпределени аспекти за обработката на сигналите при реализирането на рефлекса VOR и в частност на вестибуларният център.
 - *Ретината* на окото представлява матрица от микроскопични рецептори на външната лицева страна на очната ябълка. Нейната задача е да преобразува оптическите сигнали в невронно изображение предавано по оптически нерви в различните цетрове за анализ. Вземайки под внимание синтетическата

организация на ретината – това е сложна задача. Във всяка ретина преобразуването на изображения от оптически в невронни протича през три стадия:

- Снемане на фотокопия от слоя на невронните рецептори.
- Предаване на сформираният сигнал (реакция на светлина) от химически синапси на слоя на *биполярните клетки* (bipolar cell).
- Предаване на тези сигнали (чрез помощта на химически синапси) на изходните неврони.

На двета последни стадия (при предаване на информацията от рецептора на биполярните рецептори и от последните – на изходните неврони) в операцията участват специални неврони с латерално спиране в това число и т. нар. *хоризонтални клетки* (horizontal cell). Тяхната задача е да преобразуват сигнала между различните синаптически слоеве. Съществуват *центробежни елементи* обезпечаващи предаването на сигнала от вътрешния синаптически слой на външния. Някой изследователи създават електронни микро схеми имитиращи структурата на зеницата. Тези електронни чипове ги наричат *невронноморфни контури* (neuromorphic integrated circuit). Невронно морфните сензори представляват матрица от фоторецептори свързани със съответстващи елементи – рисунка (пиксели). Те имитират зеницата в такъв смисъл, че могат да се адаптират към изменението на осветеността идентифицирайки контурите в движение. Невробиологичният модел въплътен в нервно морфните контури има още едно преимущество – дава надежда за това, че физическото разбиране на нерво биологичните структури може да окаже съществено влияние в областта на електрониката и технологиите VLSI.

1.2. Човешки мозък

Нервната система може да се разглежда на три степени, в чиито център стои мозъка (brain), представен от мрежа от неврони (нервонни нерви) (nerve net). Той получава информация, анализира я и взема съответното решение. На фиг. 1.1. са показани два избора на стрелки. Стрелките отляво надясно показват право предаване на сигнали на информация в системата, а насочено отдясно наляво – ответната реакция на системата. *Рецептори* (receptor) преобразуват сигнала от тялото и околната среда в електрически импулси предавани в невронната мрежа (мозъка). *Ефектори* (effector) преобразуват електронните импулси генериирани в невронната мрежа (мозъка) в изходни сигнали.

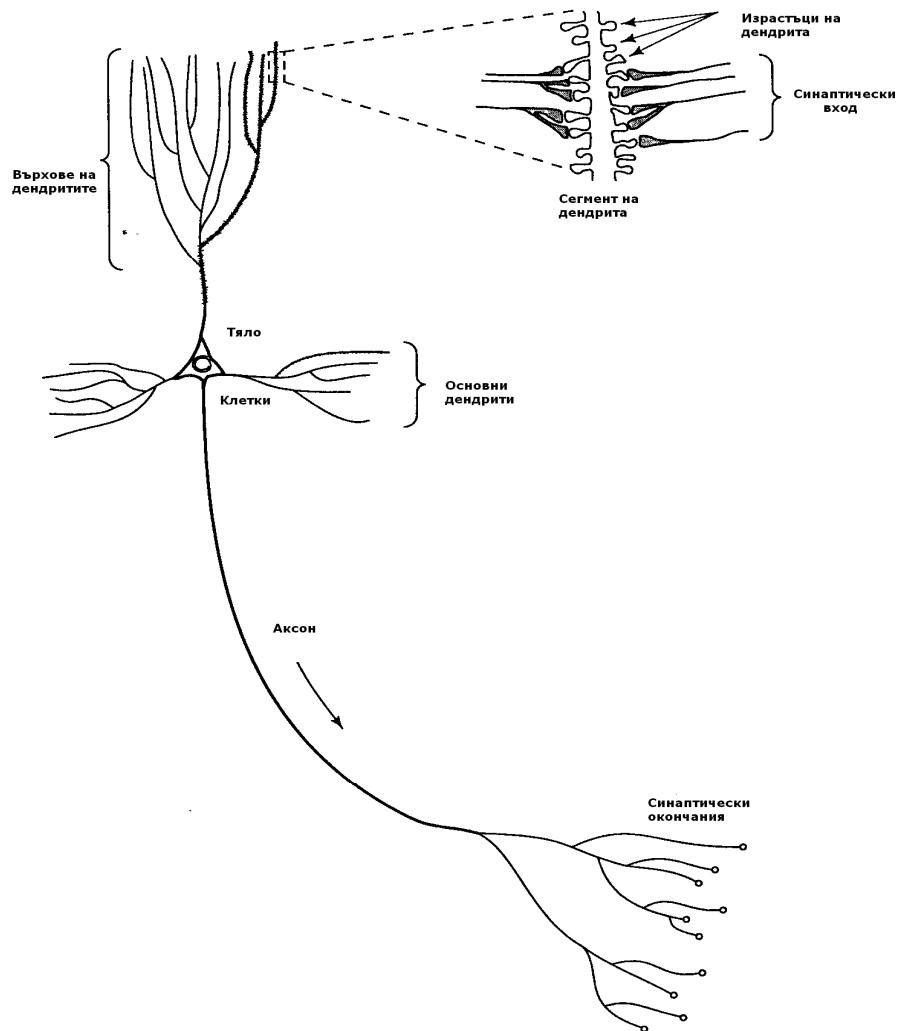


Фиг.1.1. Блок диаграма на невронната система.

Синапсът (synapses) представлява елементарна структура и функционални единици, които предават импулсите между невроните. Най-разпространен тип синапси са *химическите* (chemical synapses), които работят по следният начин – пред синаптическия процес формира *предаваема субстанция* (transmitter substance), която

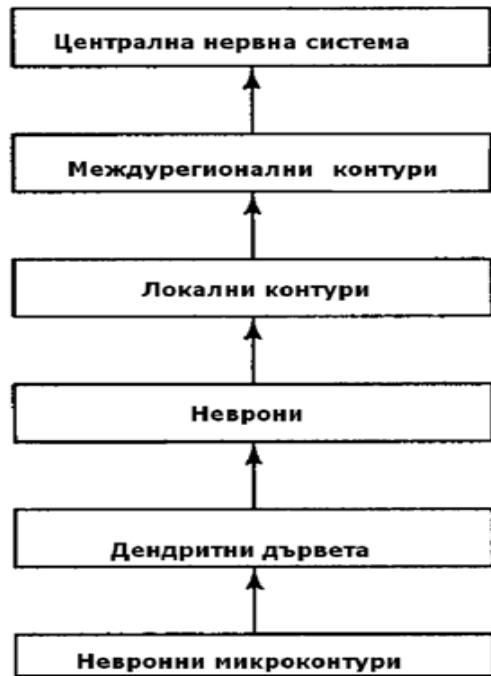
по метода на дифузията се предава по синаптическите съединения между невроните и влияе на постсинаптическият процес. Така синапсите преобразуват предсинаптическият електрически сигнал в химически, а след това в постсинаптически електрически сигнал. В електротехническата терминология това може да се нарече невзаимен *четириполюсник*. (nonreciprocal two-port device). В традиционните описания на невронните организации синапсите представляват прости съединения, които могат да предават *възбуждане* (excitation) или спиране (inhibition) (но не и двете едновременно) между невроните.

По-рано учените са обсъждали пластичността на невронната система като адаптация към условията на околната среда. В мозъка на възрастният човек под пластичност се разбират два механизма – създаване на нови синаптически връзки между невроните за сметка на съществуващите модификации. Аксони (axon) (линии на предаване) и дендрити (зоны на приемане) представляват двата типа елементи на клетки, които се различават даже на морфологично ниво. Аксоните имат по–гладка повърхност, по–тънки граници и голяма дължина. Дендритите (получили са наименованието си от приликата им с дърветата) имат нервна повърхност с много краища. Съществува огромно множество форми и размери на невроните в зависимост от това в коя част на мозъка се намират. На фиг. 1.2. е показана *пирамидална клетка* (pyramidal cell) – най-разпространеният тип невронна кора на главния мозък. Както всички неврони пирамидалните клетки получават сигнали от дендритите. Пирамидалните клетки могат да получават десет хиляди синаптически сигнала и да ги проектират на хиляди други клетки. Изходящите сигнали на большинството неврони се преобразуват в последователност на кратки електрически импулси. Тези импулси се наричат *потенциали на действие* (action potential) или *изхвърляния* (spike) идтайки от теленеврона и предавайки се чрез другите неврони с постоянна скорост и амплитуда. Причина за използваният потенциал на действие за взаимодействието на невроните се състои в самата физическа природа на аксона. Аксон на неврона има голяма дължина и малка дебелина, което се изразява с неговото голямо електрическо съпротивление и обем. Тези две характеристики са разпределени по аксона. Така аксона може да бъде моделиран като линия на електро предаване с използване на *уравнен кабел* (cable equation). Анализ на това уравнение показва, че подаването на напрежение в единият край на аксона експоненциално намалява с разстоянието достигайки до другия край на съвсем малки стойности на напрежението.



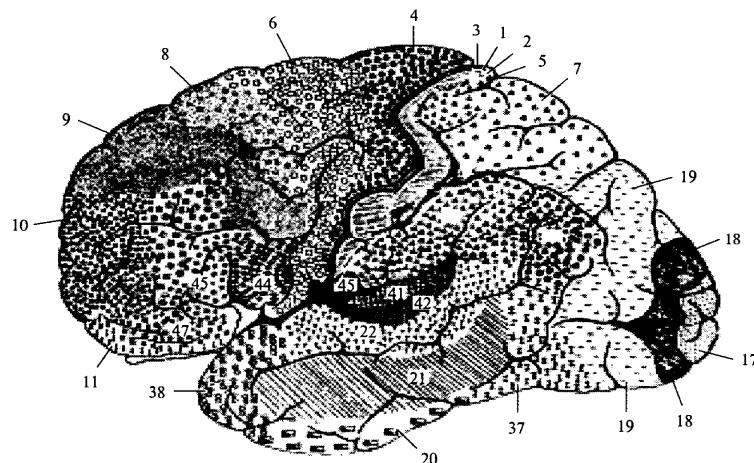
Фиг.1.2. Пирамидална клетка

В човешкият мозък съществуват крупно мащабни и малко мащабни анатомически структури. Тези високи и ниски нива отговарят за изпълнението на различни функции. На фиг. 1.3. е показана йерархия на нивата на организация на мозъка, съставена на основата на анализа на отделни области на мозъка. *Синапсите* (synapse) са най-ниското ниво – ниво молекула и иони. Следващите нива са невронните микроконтури, дендритни дървета и завършва с неврони. Под *невронна микроструктура* (neural microcircuit) се разбира избор на синапси, организирани в шаблонна взаимовръзка изпълняваща определени функции. Невронната микроструктура може да се сравни с електронен чип, съставен от множество транзистори. Минималният размер на микро контурите се измерва в микрони, а скоростта в милисекунди. Невронните микроконтури се групират в *дендритни субблокове* (dendritic subunit) съответстващи на *дендритните дървета* (dendritic tree) на отделните неврони. Теглото на неврона има размери 100 микрона и съдържа няколко дендритни субблока. На следващото ниво по сложност се намират *локалните вериги* (local circuit) (с размери около 1 mm), съставени от неврони с еднакви или сходни характеристики. Тези избори на неврони изпълняват функционални характеристики за отделни области на мозъка. След тях в йерархията следват *междурегионалните вериги* (interregional circuit), състоящи се от траектории, стълбове, топографски карти, обединяващи се в няколко области и намиращи се в различни части на мозъка.



Фиг.1.3.Структурна организация на мозъка

Топографските карти (topographic map) са предназначени за отговор за постъпващата от сензорите информация. Те често се организират във вид на таблици – визуални, звукови, суматосензорни карти. Съхраняват се в стек съхраняващ пространствена конфигурация на конкретни точки на възбуждане. На фиг. 1.4. е показана цитоархитектурна карта на церебралната кора на мозъка. На тази фигура ясно се вижда, че различните сензорни сигнали (моторни, соматически, визуални и т. н.) се отразяват на съответните области на церебралната кора със съхранен ред. На заключителното ниво на сложност топографските карти и различните междурегионални вериги се свързват едни с други и образуват централната нервна система (central nervous system).



Фиг.1.4. Цитоархитектурна карта на церебралната кора на мозъка

1.3. Модел на невроните

Неврона представлява единица за обработка на информацията в невронните мрежи. На блок схемата на фиг. 1.5. е показан модел (model) на неврон лежащ в основата на изкуствените невронни мрежи. В този модел можем да отличим три основни елемента:

- 1) Избор на *синаапси* (synapse) или *връзки* (connecting link), всяка от които се характеризира със своето *тегло* (weight) или *сила* (strength). В частност сигнала x_j на входа на синаапса j свързан с неврона k се умножава по теглото w_{kj} . Важно е да се обрне внимание на това в какъв ред са указаны индексите на синаптическото тегло w_{kj} . Първият индекс се отнася към разглежданния неврон, а втория към входният край на синаапса, с който е свързано даденото тегло. За разлика от синаапсите на мозъка синаптическото тегло на изкуственият неврон може да има както положителна, така и отрицателна стойност.
- 2) *Суматор* (adder) – натрупва входните сигнали претегляни относно съответният синаапсов неврон. Тази операция може да се опише като *линейна комбинация*.
- 3) *Активираща функция* (Activation function) – ограничава амплитудата на изходния сигнал на неврона. Тази функция се нарича още *функция на свиване* (squashing function). Нормалният диапазон на амплитудата на изхода на неврона се намира в интервала $[0,1]$ или $[-1,1]$.

В модела на неврона показан на фиг. 1.5. е включен и *прагов елемент* (bias), който се означава със символа b_k . Тази величина отразява увеличението или намалението на входния сигнал, подаван на активиращата функция. В математически вид функционирането на неврона k може да се опише със следните две уравнения:

$$u_k = \sum_{j=1}^m w_{kj} x_j, \quad (1.1)$$

$$y_k = \varphi(u_k + b_k) \quad (1.2)$$

Където x_1, x_2, \dots, x_m — входни сигнали; $w_{k1}, w_{k2}, \dots, w_{km}$ - синаптически тегла на неврона k ; u_k - линейна комбинация от входни въздействия (linear combiner output); y_k - изходен сигнал на неврона; $\varphi(\cdot)$ - активна функция; b_k - праг. Използването на прага b_k обезпечава ефекта на афинното преобразуване (affine transformation) на изходния линеен суматор u_k . В модела, показан на фиг.1.5, постсинаптическият потенциал се изчислява чрез:

$$v_k = u_k + b_k. \quad (1.3)$$

В зависимост от това дали прага b_k приема положителни или отрицателни стойности, *индукционното локално поле* (induced local field) или *потенциала на активност* (activation potential) v_k на невроните се измена така както е показано на фиг. 1.6. Прата b_k се явява външен параметър на изкуственият неврон k . Това може да

се види след преобразуването на формули 1.1, 1.2. и 1.3., които се свеждат до следващия вид:

$$v_k = \sum_{j=0}^m w_{kj} x_j, \quad (1.4)$$

$$y_k = \varphi(v_k) \quad (1.5)$$

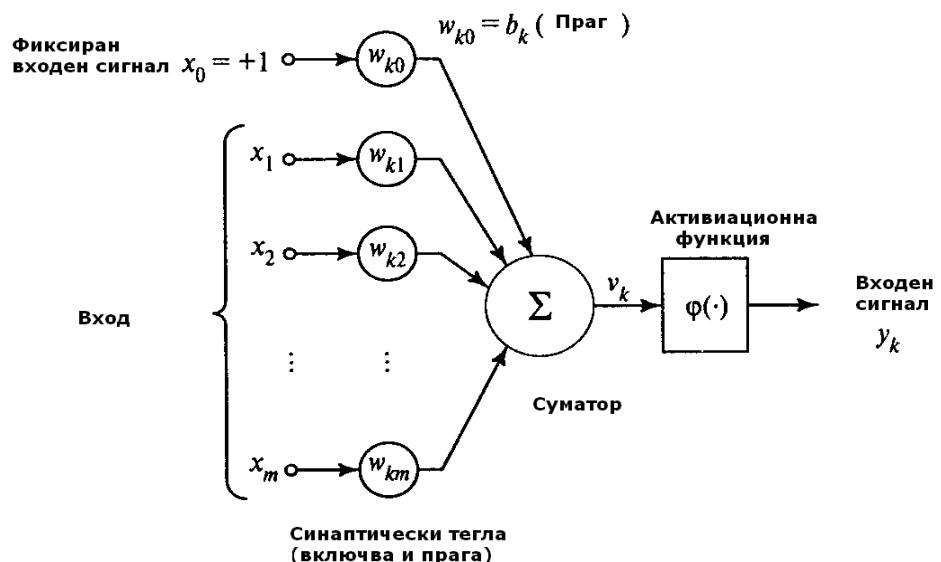
В израза 1.4. се добавя нов синапс. Неговият входен сигнал е равен:

$$x_0 = +1, \quad (1.6)$$

А неговото тегло е равно на:

$$w_{k0} = b_k. \quad (1.7)$$

От фиг. 1.7. се вижда, че в резултат на въведението на праг се добавя нов входен сигнал с фиксирана величина $+1$, а новото синаптическо тегло е равно на праговата стойност b_k . Въпреки че фиг. 1.5 и 1.7. външно не си приличат те математически са еквивалентни.



Фиг.1.7. Още един нелинеен модел на неврона

Видове активни функции

Активните функции представени чрез формула $\varphi(v)$ определят изходният сигнал на неврона в зависимост от индукционното локално поле v . Могат да се отличат три основни функции на активация:

- 1) *Функция единичен скок* или *прагова функция* (threshold function) – този тип функция е показана на фиг. 1.8 а). и се описва от следващият израз:

$$\varphi(v) = \begin{cases} 1, & \text{ако } v \geq 0; \\ 0, & \text{ако } v < 0; \end{cases} \quad (1.8)$$

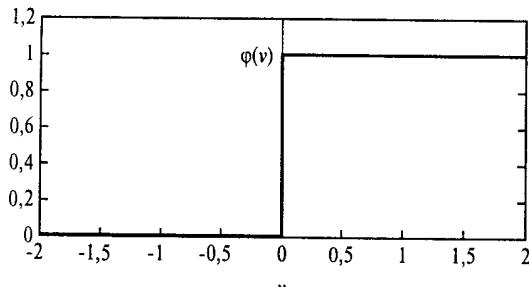
В техническата литература тази форма на функцията единичен скок обикновено се нарича *функция на Хевсайт* (Heaviside function). Съответният изходен сигнал на неврона k на тази функция може да бъде представен чрез:

$$y_k = \begin{cases} 1, & \text{ако } v_k \geq 0; \\ 0, & \text{ако } v_k < 0; \end{cases} \quad (1.9)$$

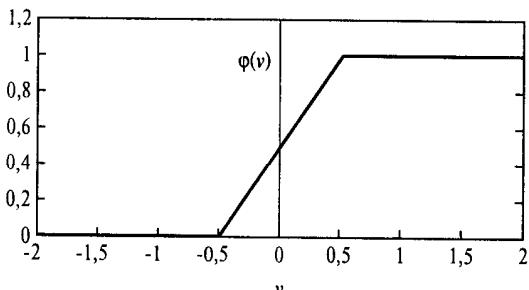
Където v_k – индукционно локално поле на неврона, т.е.

$$v_k = \sum_{j=1}^m w_{kj} x_j + b_k. \quad (1.10)$$

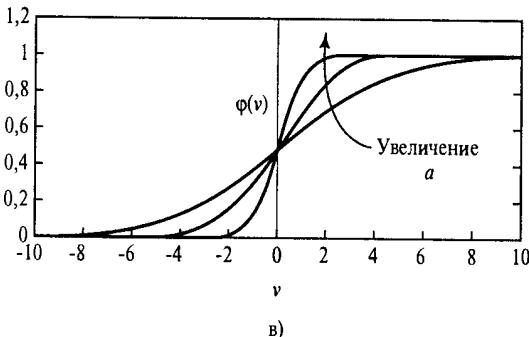
Този модел в литературата се нарича *модел на Мак – Калюка – Питца* (McCalloch-Pitts model). В този модел входният сигнал на неврона приема стойност 1 ако индукционното локално поле на този показател не е отрицателен и 0 – в противен случай.



a)



б)



в)

Фиг.1.8. Вид активационна функция: а) функция единичен скок, б)допирателно-линейна функция, в) сигмоидална функция за различни стойности на параметъра a .

2) *Допирателна-линейна функция* (piecewise-linear function). Допирателната-линейна функция показана на фиг. 1.8 б. се описва от следващият израз.

$$\phi(v) = \begin{cases} 1, & v \geq +\frac{1}{2}; \\ |v|, & +\frac{1}{2} > v > -\frac{1}{2}; \\ 0, & v \leq -\frac{1}{2}, \end{cases} \quad (1.11)$$

Където коефициента на усилване в линейните области на оператора предполага еднакви стойности. Тази активна функция може да се разглежда като *апроксимационен (approximation) нелинеен усилвател*. Следващите два варианта могат да се смятат за особена форма на допирателна линейна функция.

- Ако линейната област на оператора не достига прага на насищане той се превръща в линеен суматор.

- Ако коефициента на усилване на линейната област се приеме за безкрайно голям то допирателната – линейна функция се изражда в прагова.
- 3) *Сигмоидална функция* (sigmoid function) – графиката на сигмоидалната функция напомня буквата S. Тя е една от най-разпространените функции използвани за създаване на изкуствени невронни мрежи. Тази бързонарастваща функция поддържа баланс между линейно и нелинейно поведение. Пример за сигмоидална функция може да бъде *логическата функция* (logistic function), изразена чрез

$$\varphi(v) = \frac{1}{1 + \exp(-av)}, \quad (1.12)$$

Където a е параметър на наклонена сигмоидална функция. Изменяйки този параметър можем да построим функция с различна кривина (фиг. 1.8, в.). Първата графика съответства на величината на параметъра равна на $a/4$. В границите когато параметъра a на наклона достига до безкрайност сигмаидалната функция се изражда в прагова. Ако праговата функция може да приема стойности 0 и 1, то сигмоидалната функция приема безкрайно множество от стойности в интервала от 0 до 1. При това сигмоидалната функция се явява диференцируема, а в същото време праговата – не. Областта от стойностите на функциите на активация определени от формули (1.8), (1.11) и (1.12) представлява отрязък от 0 до +1. Понякога се изисква функцията за активация да има област от стойности от -1 до +1. В такива случай функцията е симетрична относно началото на координатната система. Това значи, че функцията на активация се явява нечетна функция на индукционното локално поле. В частност праговата функция може да се отредели от израза:

$$\varphi(v) = \begin{cases} 1, & \text{ако } v > 0; \\ 0, & \text{ако } v = 0; \\ -1, & \text{ако } v < 0, \end{cases} \quad (1.13)$$

Тази функция се нарича *сигнум*. В даденият случай сигмоидалната функция ще бъде във формата на *хиперболически тангенс*

$$\varphi(v) = \tanh(v). \quad (1.14)$$

Стохастически модел на неврона

Моделът на неврона, показан на фигура 1.7. се явява детерминантен. Това значи, че преобразуването на входен сигнал в изходен е точно определен за всяка стойност на изходният сигнал. В някой литератури по-добре са използвани стохастическите невронни модели, които функцията на активация има вероятност на интерпретация. В подобни модели неврона може да се намира в едно от двете състояния +1 и -1. Обозначаваме състоянието на неврона с x , а *вероятностната активация на неврона*

(probability of firing) – функцията $P(v)$, където v – индукционно локално поле на неврона. Тогава

$$x = \begin{cases} +1, & \text{с вероятност } P(v); \\ -1, & \text{с вероятност } 1 - P(v). \end{cases}$$

Вероятността $P(v)$ описва сигмоидална функция в следващия вид:

$$P(v) = \frac{1}{1 + \exp(-v/T)}, \quad (1.15)$$

Където T е аналог за *температура* (temperature) използвана за определяне на нивото на шума и по този начин на степента на неопределеност на превключванията. Те не описват физически температурата на невронната мрежа, независимо дали е биологическа или изкуствена. Параметърът T управлява термалните функции, представляващи ефект от синаптическият шум. Ако параметърът T се стреми към 0, то стохастическия неврон описан с формула 1.15. приема детерминирована форма (без включване на шума) на Мак – Колюка – Питца.

1.4. Представяне на невронните мрежи с помош на насочен граф

Блок-диаграмите представени на фиг. 1.5. и 1.7. показват функционално описание на различни елементи от които е съставен модела на изкуственият неврон.

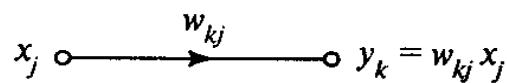
Граф за предаване (или прохождане) (signal-flow) на сигнала – това са насочени връзки (links) (или разклонения (branches)), съединяващи определени точки (възли). С всеки възел j са свързани сигнали \mathbf{x}_j . Обикновено насочването на връзките започва в някой възел j и завършва в друг възел k . С нея са свързани някой *предаващи функции* (transfer function), определящи зависимостта на сигнала y_k от възела k от сигнала \mathbf{x}_j към възела j . Преминаването на сигнала по различните честоти на графа се подчинява на три правила:

Правило 1: Насоченото преминаване на сигнала от всяка мрежа се определя от насочени стрелки. Могат да се определят два типа връзки:

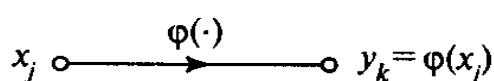
- *синаптически връзки* (synaptic link) – поведението им се определя от *линейното* (linear) съотношение вход-изход, а сигналите на възела \mathbf{x}_j се умножават по синаптическите тегла w_{kj} и в резултат се получава сигнал във възела y_k . Те са показани на фиг. 1.9, a.
- *активиращи връзки* (activation link) – поведението им се определя от *нелинейното* (nonlinear) съотношение вход-изход. Те са показани на фиг. 1.9b, където $\Phi(\cdot)$ е нелинейна функция на активация.

Правило 2: Сигналите на възела са равни на алгебричната сума на сигналите постъпващи на неговия вход. На фиг. 1.9, в. е демонстрирано това правило за случай на *синаптическа сходимост* (synaptic convergence).

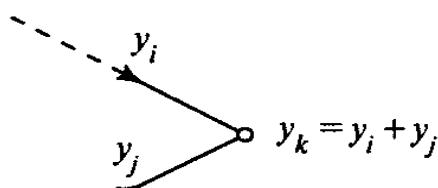
Правило 3: Сигналите от дадения възел се предават по всички изходни връзки без отчитане на предаваната функция на изходните връзки. На фиг. 1.9, г е показано това правило за *синаптически дивергенции* (synaptic divergence) или разходимости.



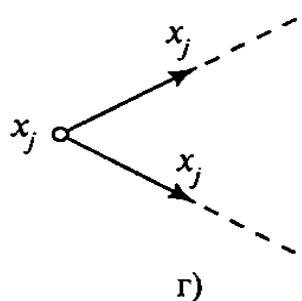
a)



б)



в)



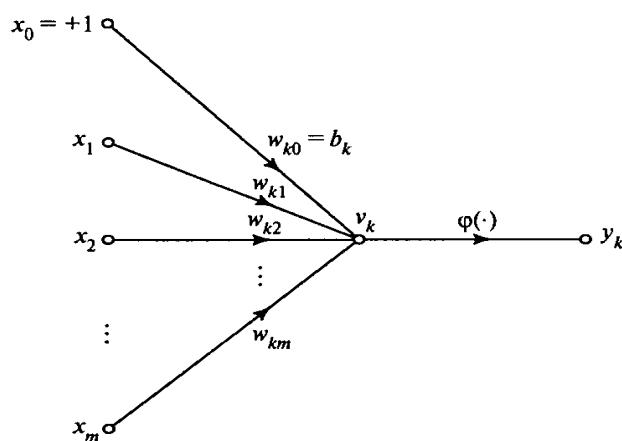
г)

Фиг. 1.9. Основни правила за построяние на граф за предаване на сигнали

На фиг. 1.10. е показан пример за граф на предаване на сигнала. Това е модела на неврона съответстващ на блоковата диаграма от фиг. 1.7. Входният сигнал приема стойност $x_0 = +1$, а съответното синаптическо тегло $w_{k0} = b_k$, където b_k – праг на неврона k .

Невронна мрежа – това е насочен граф, състоящ се от възли съединяващи синаптическите и активиращи връзки, които се характеризират със следните четири свойства:

1. Всеки неврон представлява множество от линейни синаптически връзки, като външният праг е възможна нелинейна връзка на активация. Прагът представляващ входните синаптически връзки се счита за равен на +1.
2. Синаптическите невронни мрежи се използват за претегляне на съответстващите им входни сигнали.
3. Претеглената сума на входните сигнали определя индукционно локално поле за всеки отделен неврон.
4. Активиращата връзка модифицира индукционното локално поле на неврона свързвано с изходният сигнал.

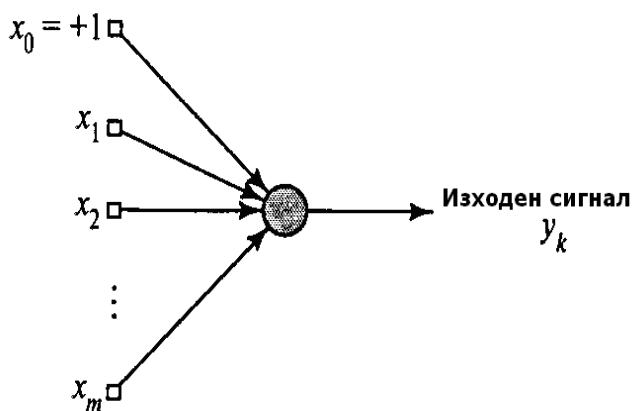


Фиг.1.10. Граф за предаване на сигнала за един неврон

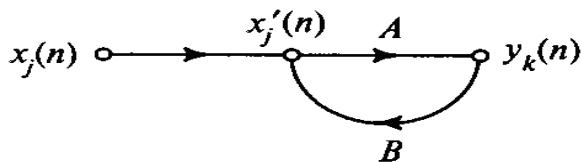
Насоченият граф се определя от показаният по-горе способ, който се явява *пълен* (complete). Това означава, че той описва не само протичането на сигнала между невроните, но и предаването на сигнал в самия неврон. Ако е необходимо да се опише само преминаването на сигнала между невроните, то може да се използва съкратена форма на този граф, изпускаща детайлите на предаване на сигнала в неврона. Такъв насочен граф се нарича *частично пълен* (partially complete) и се характеризира със следните свойства:

- Входният сигнал на графа *формира източника* (source node) на възли или входните елементи.
- Всеки неврон се представя от един възел наречен *изчислителен* (computation node).
- *Линии на предаване на сигнала* (communication links), свързващи възлите – източници и изчислителните възли на графа нямат тегло. Те просто определят направлението на протичането на сигнала на графа.

Частично пълен насочен граф определен по посоченият по-горе способ се нарича *архитектурен* (architectural graph). Той описва структурата на невронните мрежи. Прост случай на граф състоящ се от един неврон с m входа и фиксиран праг е равен на +1 е показан на фиг. 1.11.



Фиг.1.11. Архитектурен граф на неврона



Фиг.1.12. Граф за предаване на сигнала в система с една обратна връзка

Представяне на граф за предаване на сигнала в система с една обратна връзка чрез три графични представления на невронните мрежи:

1. Блокова диаграма описваща функциите на невронната мрежа.
2. Граф за протичане на сигнала обезпечаващ пълно описание на предаването на сигнала по невронната мрежа.
3. Архитектура на граф описваща структурата на невронните мрежи.

1.5. Обратна връзка

Обратната връзка (feedback) е понятие характерно за динамични системи, в които изходният сигнал от някой елементи на системата указва влияние на входният сигнал на този елемент. На фиг. 1.12. е показан граф за протичане на сигнала в *система с една обратна връзка* (single-loop feedback system). В нея входният сигнал $x_j(n)$, вътрешният сигнал $x'_j(n)$ се явяват функции на дискретната променлива n . Предполага се, че тази система е линейна и съдържа пряка и обратна връзка, която се характеризира с операторите A и B съответно. В частност, изходния сигнал по прекия канал частично определя стойността на канала за обратна връзка. На фиг. 1.12. е показана тази зависимост

$$y_k(n) = A[x'_j(n)], \quad (1.16)$$

$$x'_j(n) = x_j(n) + B[y_j(n)], \quad (1.17)$$

където квадратните скоби обозначават операторите А и В. Изключваме променливата $x'_j(n)$ в изразите (1.16.) и (1.17.) и получаваме:

$$y_k(n) = \frac{A}{1 - AB} [x_j(n)]. \quad (1.18)$$

Израза $A/(1 - AB)$ се нарича *оператор за затворен кръг* (closed-loop operator) на системата, а израза AB – *оператор за отворен кръг* (open-loop operator). В общият случай операторът за отвореният кръг не притежава свойството комутативност, т.e. $BA \neq AB$.

За пример ще разгледаме система с една обратна връзка показана на фиг. 1.13. В нея предполагаме, че оператора А има фиксирано тегло w , а В се явява *оператор за единична задръжка* (unit-delay operator) z^{-1} , който задържа изходният сигнал по отношение на входния на една стъпка дискретизация. Изхода на оператора за затворения кръг се изразява чрез:

$$\frac{A}{1 - AB} = \frac{w}{1 - wz^{-1}} = w(1 - wz^{-1})^{-1}.$$

Използваният бином представлява израза $(1 - wz^{-1})^{-1}$, като оператора за затвореният кръг може да бъде записан във вида:

$$\frac{A}{1 - AB} = w \sum_{l=0}^{\infty} w^l z^{-l}. \quad (1.19)$$

Замествайки израза (1.19.) в (1.18.) получаваме

$$y_k(n) = w \sum_{l=0}^{\infty} w^l z^{-l} [x_j(n)]. \quad (1.20)$$

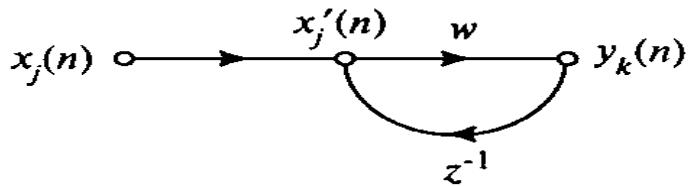
Тук квадратните скоби отново обозначават факта, че z^{-1} е оператор, за който следва

$$z^{-l} [x_j(n)] = x_j(n - l), \quad (1.21)$$

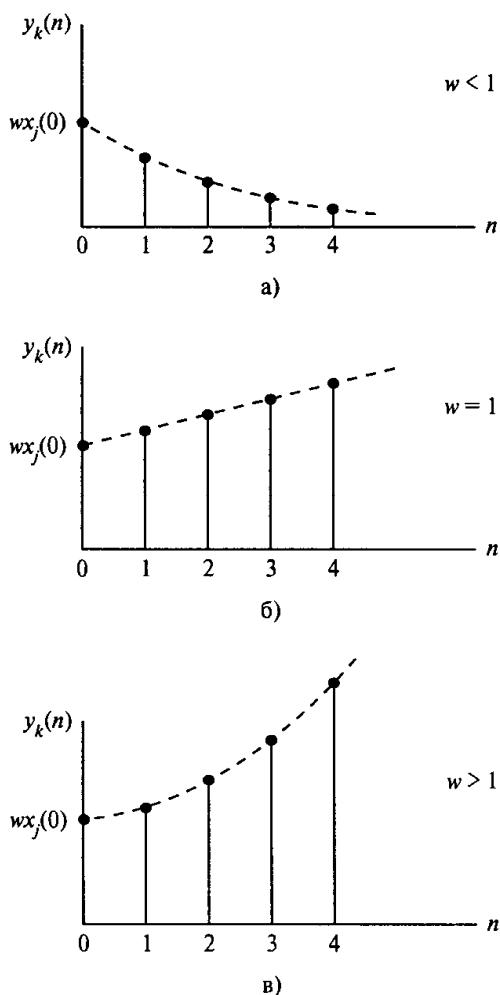
Където $x_j(n - l)$ е входният сигнал на задържане на l единица дискретизация. Следователно изходният сигнал $y_k(n)$ може да се представя като безкрайно претеглена сума на текущи и предишни стойности на входният сигнал $x_j(n)$.

$$y_k(n) = \sum_{l=0}^{\infty} w^{l+1} x_j(n - l). \quad (1.22)$$

На фиг. 1.14. ясно се вижда, че динамичната реакция на системата се определя от теглото w . Може да се разглеждат два случая:



Фиг.1.13. Граф за протичане на сигнал за филтър с безкрайна импулсна характеристика (IIR-filter – infinite-duration impulse response filter) от първи ред



Фиг.1.14. Реакция на системата показана на фиг.1.13. за три различни тегла на w .

(а) Устойчива система, (б) линеен случай, (в) експоненциална разходимост.

- $|w| < 1$. В даденият случай изходният сигнал $y_k(n)$ е експоненциално сходящ (vergent). Това значи, че системата е устойива (stable). Реакцията на фиг.1.14, а е за положителни стойности на w .
- $|w| \geq 1$. В дадения случай изходният сигнал $y_k(n)$ е разходящ (divergent). Това значи, че системата е неустойива (unstable). Ако $|w| = 1$, разходимостта е линейна (фиг. 1.14, б); ако $|w| > 1$ - експоненциална (фиг.1.14, в).

Случая $|w| < 1$ съответства на система с безкрайна памет, в смъсъл че входа на системата зависи от състоянието на системата в безкрайно далекото минало.

1.6. Архитектура на мрежите

Еднослойна мрежа за пряко разпространение

В многослойните мрежи неврона се разполага по слоеве. В най-простиият случай в такава мрежа съществува *входен слой* (input layer) на възлите на източника, информацията се предава на *изходният слой* (output layer) на невроните (изчислителният възел) но не и обратно. Такава мрежа се нарича мрежа за *пряко разпространение* (feed-forward) или *ациклична* (acyclic) мрежа. На фиг. 1.15. е показата структура на такава мрежа за случай на четири възела във всеки слой (входен и изходен). Такива невронни мрежи се наричат *еднослойни* (single-layer network), при това единственият слой е от изчислителни елементи на невроните.

Многослойна мрежа за пряко разпространение

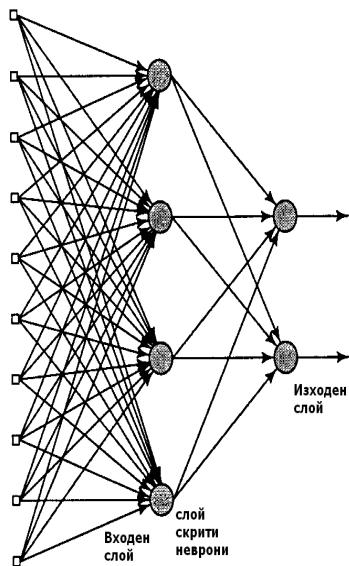
Друг клас невронни мрежи за пряко разпространение се характеризират с наличието на един или няколко *скрити слоя* (hidden layer), възлите се наричат *скрити неврони* (hidden neuron) или *скрити елементи* (hidden unit).

Възлите източници на входният слой на мрежата формират съответните елементи на шаблонна активация (входен вектор), които съставят входният сигнал, постъпващ от неврона (изчислителни) елементи на втория слой (т.е. първият скрит слой). Изходният сигнал на втория слой се използва в качеството на входен сигнал на третия слой и т.н. Избора на изходни сигнали на невроните на последният слой от мрежата определят крайния отговор на мрежата на даденият входен образ оформящ възли от източника на входният (първият) слой на мрежата (фи. 1.16.), наречен *мрежа 10-4-2*, така както тя има 10 входни, 4 скрити и 2 изходни неврона. В общиия случай мрежата на право разпространение с m входа, h_1 неврони на първият скрит слой, h_2 неврони на втория скрит слой и q неврона на изходният слой се нарича мрежа $m - h_1 - h_2 - q$.

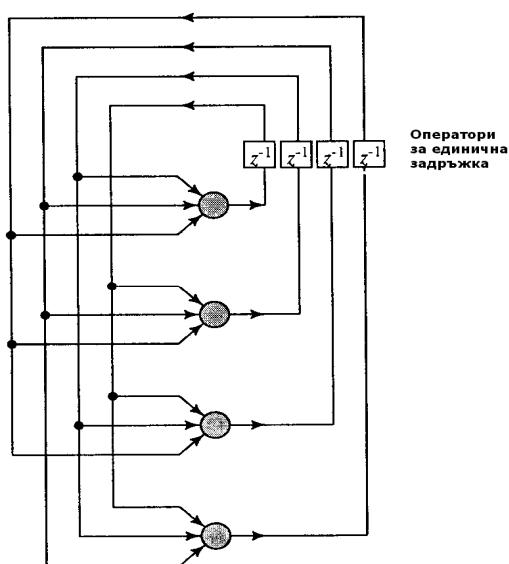
Рекурентни мрежи

Рекурентните невронни (recurrent network) мрежи се отличават от мрежите за пряко разпространение с наличието на поне една *обратна връзка* (feedback loop). Пример за рекурентна мрежа може да бъде мрежа, кочто се състои от единствен слой

неврони, всеки от който изпраща своя изходен сигнал на входа на всички останали невронни слоеве. Архитектура на такава невронна мрежа е показана на фиг. 1.17.

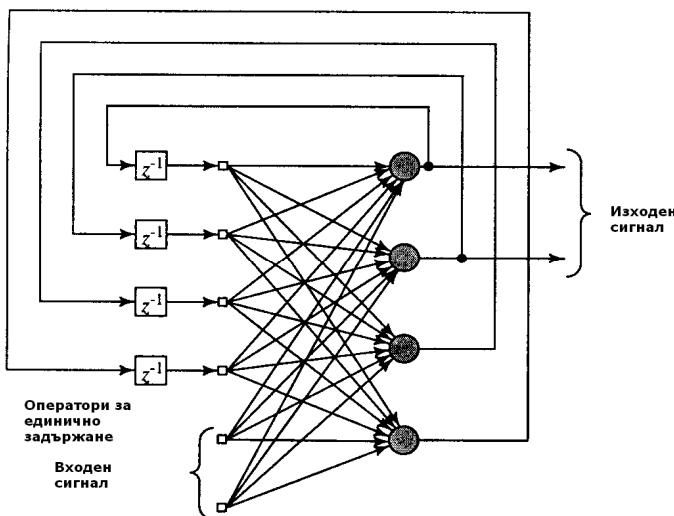


Фиг.1.16. Напълно свързана мрежа на право разпространение с един скрит и един изходен слой



Фиг.1.17. Рекурентна мрежа без скрити неврони и обратна връзка на невроните сами със себе си

На фиг. 1.18. е показън друг клас невронни мрежи – със скрити неврони.



Фиг.1.18. Рекурентна мрежа със скрити неврони

Наличието на обратни връзки в мрежата показана на фиг. 1.17. и 1.18. указва непосредствено влияние на способността на такива мрежи към обучение и на тяхната производителност. Обратната връзка поддържа използването на елементи на единично задържане z^{-1} , което води до нелинейно динамично поведение, ако в мрежата се съдържат нелинейни неврони.

1.7. Представяне на знания

Под *представяне на знания* (knowledge representation) се разбира каква информация е необходимо да се съхрани и как тази информация се представя физически за следващо използване.

Основната задача на невронните мрежи е най-доброто обучение обучение на моделите на околната среда за решаването на поставена задача. Знанията за света включват два типа информация:

- Известно състояние за заобикалящата ни действителност, представляват имащите в наличност достоверни факти. Тази информация се нарича *априорна* (prior).
- Наблюдаването на околната среда (измервания), получени с помощта на сензори, адаптиращи се към съответните условия, в които е длъжна да функционира дадена невронна мрежа. Такива измервания в значителна степен са шумни, което може да доведе до грешки. Във всички случаи измерванията получени по този способ формират множество от операции, примери от които се използват за обучение на невронни мрежи.

Примерите могат да бъдат *маркирани* (labeled) и *немаркирани* (unlabeled). В маркираните примери *входният сигнал* (input signal) съответства на *желания отговор* (desired response). Немаркираните примери се състоят от няколко реализации на един входен сигнал.

Множеството от двойки на сигнала вход-изход, всяка от които се състои от входен сигнал и съответните му желани изходи се нарича *обучаващи данни* (training

data) или *обучаващи избори* (training sample). За пример ще разгледаме задачата за *разпознаване на цифри* (digit recognition problem). В тази задача входния сигнал представлява сам по себе си матрица, състояща се от черни и бели точки. Желаният резултат на мрежата се явява конкретна цифра, изображението на която се подава в качеството на входен сигнал. Обучаващите избори се състоят от множество числа от ръкописните цифри, което отразява ситуацията, която може да възникне в реалния свят. При наличие на такива избори на примери на невронните мрежи се създават по следния начин:

- Избират се съответните невронно-мрежови архитектури, в които размера на входния слой съответства на количеството на пикселите на рисунка, а изходните слоеве съдържат 10 неврона съответстващи на цифрите. След това се извършва настройка на тегловите конфигурации на мрежата на основата на обучаващо множество. Този режим на работа на мрежата се нарича *обучение*.
- Ефективността на обучаващата мрежа се проявява (реституира) на множество примери отличаващи се от използваните при обучението. На входа на мрежата се подава изображение, за което е известен целеви изход на мрежата. Ефективността на обучението на мрежата се проверява по пътя на сравнение на резултатите от разпознаването с реалните цифри. Този етап на работа на невронните мрежи се нарича *обобщение* (generalization).

Наборът от данни, използвани при обучение на мрежата е длъжен да съдържа както положителни, така и отрицателни примери. Например в задачата за пасивната ехо-локация, положителните примери включват сигнали, отразени от интересувани обект (например подводница). Върху реалната среда на отразяването на радара, влияят и морските обекти, случайно попаднали в зоната на сигнала. За да се изясни вероятността от невронна трактовка на сигнала в множеството от примери се добавят сигнали, получени при присъствието на търсения обект.

Въпросите за представянето на знания в невронните мрежи са доста сложни. Още по-добре могат да се видят в следващите 4 правила:

Правило1: Сходни входни сигнали от сходни класове трябва да формират единно представяне в невронната мрежа. Те са длъжни да бъдат класифицирани като принадлежащи към една категория.

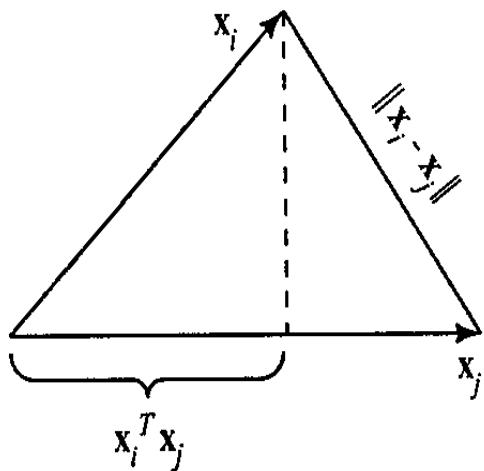
Съществуват множество подходи за определяне степента на сходство на входните сигнали. Обикновено степента на подобие се определя на основата на *Евклидовото разстояние* (Euclidian distance). За примера предполагаме, че съществува вектор \mathbf{x}_i с размерност m .

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \dots, x_{im}]^T.$$

Всички елементи са естествени числа, обозначението T показва, че матрицата е *транспонирана* (transposition). Вектора \mathbf{x}_i , определя някоя точка в m -мерното *Евклидово пространство*. Евклидовото разстояние между двойките m -мерни вектори \mathbf{x}_i и \mathbf{x}_j се изчислява чрез:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \left[\sum_{k=1}^m (x_{ik} - x_{jk})^2 \right]^{1/2}, \quad (1.23)$$

Където \mathbf{x}_{ik} и \mathbf{x}_{jk} – к-елементи на векторите \mathbf{x}_i и \mathbf{x}_j съответно. От това следва, че степента на сходство между изходните сигнали, представена от векторите \mathbf{x}_i и \mathbf{x}_j , се явява величина обратна на Евклидовото разстояние между тях $d(\mathbf{x}_i, \mathbf{x}_j)$.



Фиг.1.19. Илюстрация на взаимовръзките между скаларното произведение и Евклидовото разстояние

Колкото са по-близо един до друг отделните елементи на векторите \mathbf{x}_i и \mathbf{x}_j е по-малко Евклидовото разстояние $d(\mathbf{x}_i, \mathbf{x}_j)$ и е по-близко сходството между векторите \mathbf{x}_i и \mathbf{x}_j . Правило 1, констатира, че ако векторите \mathbf{x}_i и \mathbf{x}_j са сходни, то те трябва да се отнесът към една категория (клас).

Още един подход за определяне степента на сходство се основава на идеята за *скаларно произведение* (inner product) на матрици, взето от алгебрата. Ако векторите \mathbf{x}_i и \mathbf{x}_j имат еднакви стойности, скаларното произведение $\mathbf{x}_i^T \mathbf{x}_j$ определя следващия израз:

$$(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j = \sum_{k=1}^m x_{ik} x_{jk}. \quad (1.24)$$

Резултатът от делението на скаларното произведение $(\mathbf{x}_i, \mathbf{x}_j)$ на $\|\mathbf{x}_i\| \|\mathbf{x}_j\|$ е равен на косинуса на вътрешния ъгъл между векторите \mathbf{x}_i и \mathbf{x}_j .

Тези два метода за измерване на сходството са тясно свързани един с друг (фиг.1.19). Евклидовото разстояние $\|\mathbf{x}_i - \mathbf{x}_j\|$ между векторите \mathbf{x}_i и \mathbf{x}_j е свързано с проекцията на вектора \mathbf{x}_i върху вектора \mathbf{x}_j . На фиг.1.19. се вижда, че колкото по-

малко е Евклидовото разстояние $\|\mathbf{x}_i - \mathbf{x}_j\|$ токъто по-голямо е скаларното произведение $\mathbf{x}_i^T \mathbf{x}_j$.

За да формализираме това съотношение нормираме векторите \mathbf{x}_i и \mathbf{x}_j . При това тяхната дължина е равна на единица: $\|\mathbf{x}_i\| = \|\mathbf{x}_j\| = 1$.

Използвайки израз (1.23), записваме:

$$d^2(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j) = 2 - 2\mathbf{x}_i^T \mathbf{x}_j. \quad (1.25)$$

От израза (1.25) се вижда, че минимизация на Евклидовото разстояние $\|\mathbf{x}_i - \mathbf{x}_j\|$ и максимизация на скаларното произведение $\mathbf{x}_i^T \mathbf{x}_j$ водят до увеличаване на сходството между векторите \mathbf{x}_i и \mathbf{x}_j .

Тук Евклидовото разстояние и скаларното произведение са описани в детерминистични термини. За примера предполагаме, че различието между две множества данни се изразява в различието между векторите и тяхното математическо очакване:

$$\boldsymbol{\mu}_i = E[\mathbf{x}_i], \quad (1.26)$$

Където E – статистически по оператор на математическо очакване. Векторът $\boldsymbol{\mu}_j$ се определя чрез аналогичен способ. За измереното разстояние между двете множества може да се използва *разстоянието на Махаланобис* (Mahalanobis distance), което се обозначава чрез d_{ij} . Квадратът на тази величина се определя от следващата формула:

$$d_{ij}^2 = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_j), \quad (1.27)$$

Където $\boldsymbol{\Sigma}^{-1}$ – обратна матрица за квадратната матрица $\boldsymbol{\Sigma}$. Предполагаме, че матрицата на ковариациите между двете множества е една и съща, т.е.

$$\boldsymbol{\Sigma} = E[(\mathbf{x}_i - \boldsymbol{\mu}_i)(\mathbf{x}_i - \boldsymbol{\mu}_i)^T] = E[(\mathbf{x}_j - \boldsymbol{\mu}_j)(\mathbf{x}_j - \boldsymbol{\mu}_j^T)]. \quad (1.28)$$

В частният случай, когато $\mathbf{x}_i = \mathbf{x}_j$, $\boldsymbol{\mu}_i = \boldsymbol{\mu}_j = \boldsymbol{\mu}$ и $\boldsymbol{\Sigma} = \mathbf{I}$, където \mathbf{I} – единична матрица, разстоянието на Махаланобис се изражда в Евклидово разстояние между вектора \mathbf{x}_i и вектора на математическото очакване $\boldsymbol{\mu}$.

Правило 2: Елементите отнесени към различни класове трябва да имат в мрежата колкото може повече отличаващи представления.

Това правило е противоположно на първото.

Правило 3: Ако някое свойство има важно значение, то за неговото представяне в мрежата е необходимо използването на голямо количество неврони.

За пример може да се разгледа задачата за откриване на обект от радара (например самолет) при наличие на намеса (например облаци, дървета и т.н.). Ефективността на такава радарна система се измерва от две величини:

- *Вероятност за откриване* (probability of detection) – вероятността, че при наличие на обекта системата ще го открие;
- *Вероятност за невярна тревога* (probability of false alarm) – вероятността, че системата поределя наличие на обекта, когато той реално несъществува.

В съответствие с *критерия на Нейман-Пирсон* (Neuman-Pearson criterion) вероятността за откриване трябва да бъде максимална, а вероятност за невярно откриване не трябва да превъзхожда никоя зададена стойност.

Правило 4: В структурата на невронната мрежа трябва да има априорна информация и инварианти, което опростява архитектурната мрежа в процеса на обучение.

Това правило играе особена роля, т.к. правилната конфигурация на мрежата обезпечава нейната специализация, което е много важно по следните причини:

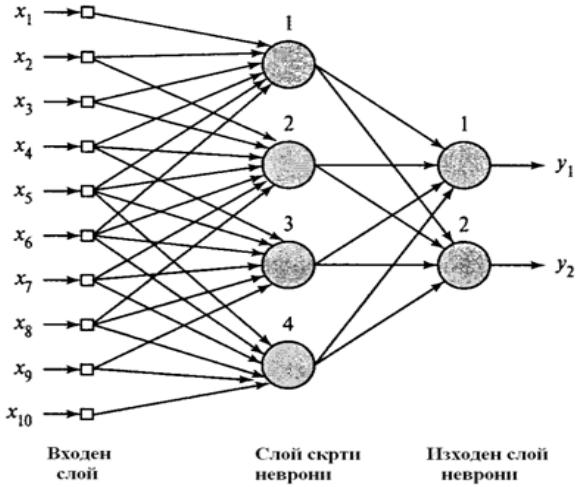
- a) Биологичната мрежа, обезпечаваща обработката на зрителната и слуховата информация е силно специализирана.
- b) Невронната мрежа със специални структури, включва значително малко количество свободни параметри, които е нужно да се настройват към пълносвързаната мрежа. Следователно за обучението на специализираните мрежи трябват по-малко данни. При това на обучението се изразходва малко време, и такава мрежа има добри обобщени способности.
- c) Специализираните мрежи имат голяма пропускателна способност.
- d) Цената на създаване на специализирани невронни мрежи се намалява, т.е. размера ѝ е съществено по-малък от размера на пълносвързаните мрежи.

Как да вмъкнем априорна информация в структурата на невронните мрежи

За изпълнението на правило 4 е необходимо да се разбере как да се разработи специална структура в която е вмъкната априорна информация. Можем да използваме информация между следващите два примера:

- a) Ограничаване на мрежовата архитектура с помощта на локални връзки наречени рецепторни полета.
- b) Ограничение на избора на синаптическо тегло за сметка на съвместното използване на теглото

Тези два примера обезпечават едно важно предимство – значително съкращават количеството свободни параметри в мрежата. За примера разглеждаме не пълно свързана мрежа за пряко разпространение показана на фиг. 1.20.



Фиг.1.20. Пример за мрежа с реципрочни области и съвместно използване на тегла. Всички четири скрити неврона за реализацията на синаптическите връзки съвместно използват едно и също множество от тегла.

Тази мрежа има ограничена архитектура. Първите шест възела на източника образуват рецепторно поле скрития неврон с номер 1 и така за всички останали скрити неврона на мрежата. Индукционното локално поле скрито за неврона j може да бъде описано със следният израз

$$v_j = \sum_{i=1}^6 w_i x_{i+j-1}, \quad j = 1, 2, 3, 4, \quad (1.29)$$

където $\{w_i\}_{i=1}^6$ определя един и същи набор от тегла съвместно използвани от четирите скрити неврона, а x_k е сигнал получен от възела на източника с номер $k = i + j - 1$. Мрежата за пряко разпространение с локални връзки и съвместно използвани тегла се нарича suma от извивки (convolution network).

Как да вмъкнем инварианти в структурата на невронната мрежа

Ще разгледаме следните физически явления:

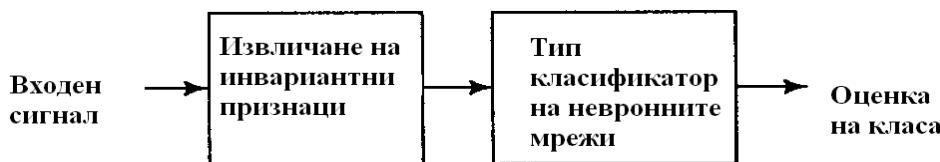
- Ако изследваният обект се върти то съответният образ се изменя, изменя се и неговият образ възприеман от наблюдателя.
- В кохерентният радар обезпечаващ информацията от амплитудата и фазата на източника на околната среда, ехото от движещият се обект се измества по честотата. Това е свързано с ефекта на Доплер, който възниква при радиално (въртеливо) движение на наблюдавания обект относно радара.
- Диктор може да произнася думи, както с тих глас, така и с висок глас, както бавно така и бързо говорейки.

Диапазона на трансформация (transformation) на наблюдаваният обект - едно от основните изисквания при разпознаване на образи е създаването на такъв класификатор, който е инвариантен към тази трансформация. С други думи резултатът

на класификация не трябва да указва влияние върху трансформацията на входния сигнал постъпваща от обекта за наблюдение.

Съществуват три приома на обезпечавана на инвариантност на невронната мрежа класифицирани към подобни трансформации:

1. *Структура на инвариантност* (invariance by structure) – инвариантността може да бъде внесена в невронните мрежки чрез съответната структуризация, в частност синаптическите връзки между отделните невронни мрежки се строят така че трансформационните версии на един и същи сигнал да показват един и същ изходен сигнал. Ще разгледаме невронна мрежова класификация на входен сигнал, която трябва да е инвариантна по отношение на плоското въртене на изображението относно неговият център. Структурната инвариантност на мрежата относно въртенето може да се изрази така: Нека w_{ji} синаптическо тегло на неврона j свързано с пиксела i на входното изображение. Ако условието $w_{ji} = w_{jk}$ се изпълнява за всички пиксели j лежащи на равно разстояние от центъра на изображението невронната мрежа ще бъде инвариантна спрямо въртенето. За да обезпечим инвариантността относно въртенето е нужно да се дублира синаптическото тегло w_{ji} на всички пиксели равно отдалечени от центъра на изображението. Недостатъка на структурата на инвариантността е това, че количеството синаптически връзки на изображението даже на среден размер ще бъде много голямо.

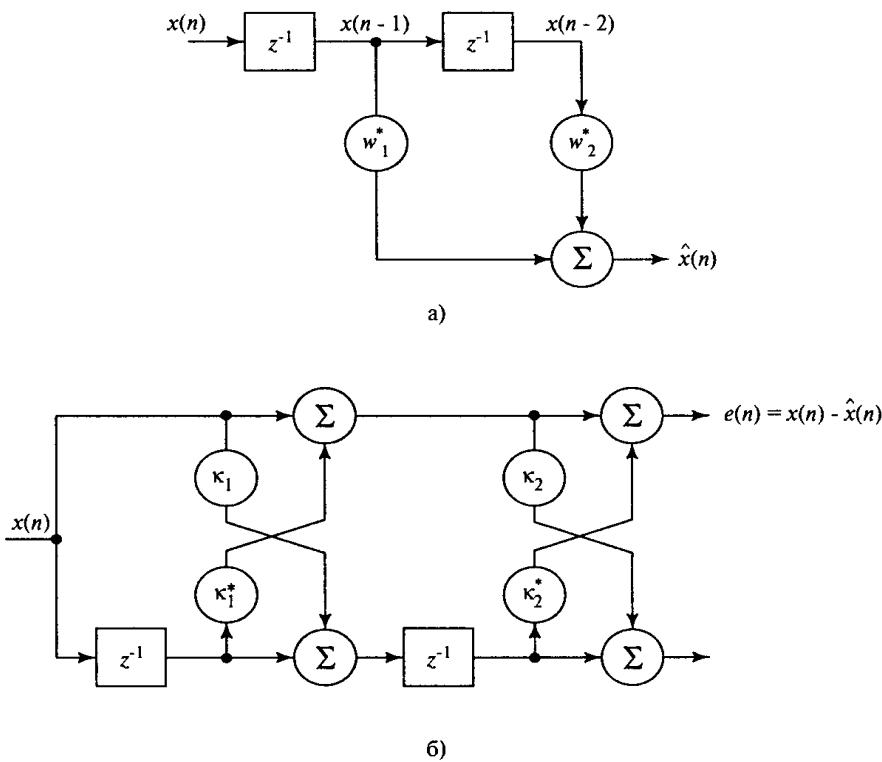


Фиг.1.21. Диаграма на система, използваща пространството от инвариантни признаки

2. *Инвариантност на обучението* (invariance by training). Невронните мрежи имат естествената способност за класификация на образите. Тя може да се използва за обезпечение на инвариантност на мрежите към трансформация. Мрежата се обучава от множество примери на един и същи обект, при това всеки пример от обекта се подава в няколко изменени вида (например снимка с различни страни). От техническа гледна точка инвариантността на обучението има два съществени недостатъка. Първият е ако невронната мрежа е била научена да разпознава трансформации на обекта от някой клас, съвсем не е задължително тя да има инвариантност по отношение на трансформациите на обектите от друг клас. Вторият – такова обучение е много енергоемко, особено при голява размерност на пространството на признаците.
3. *Използване на инвариантни признаки* (invariant feature space). Третият метод на създаване на инвариантно невронно мрежово класификатори е показан на фиг. 1.21. Основава се на предположението, че от входния сигнал може да се отделят информативни признаци, които описват самото съществуване на информацията, съдържаща се в набора от данни и при това инвариантни към

трансформацията на входният сигнал. При използването на такива признания в невронните мрежи не е нужно да се съхранява излишен обек информация, описващи трансформацията на обекта. При използването на инвариантните признания различията между различните екземпляри на един и същи обект могат да се получат само от случайни фактори, такива като шум. Използването на пространството на инвариантност на признанията има три премиства:

- Намалява се количеството на признанията, които се подават в невронната мрежа.
- Намаляват се изискванията към структурата на мрежата
- Гарантира се инвариантност на всички обекти по отношение на известната информация.



Фиг.1.22. Модел на авторегресия от втора степен: (а)Модел на филтъра на линията на задържане;(б)Модел на мрежи с филтър, с * е посочено комплексното разпределение

Ще илюстрираме идеята за пространството на инвариантните признания, разглеждайки пример с радар, използван от авиодиспечърите, като входния сигнал може да съдържа информация, постъпваща от самолет, ято птици и други природни явления. Сигнала на радара, отразяващ различните цели, има различни спектрални характеристики. Експерименталните изследвания показват, че сигнала на такъв радар може да бъде моделиран с помощта на *авторегресивен процесор (AR-процесор)* от *средна степен* (autoregressive process of moderate order). AR-процесора представлява особен вид регресивен модел, описващ следващия израз:

$$x(n) = \sum_{i=1}^M a_i^* x(n-i) + e(n), \quad (1.30)$$

Където $\{a_i\}_{i=1}^M$ - коефициент (coefficient) на авторегресия; M – степен на модела (model order); $x(n)$ - входен сигнал (input signal); $e(n)$ - грешка (error), представляваща бял шум. Моделът описан чрез формула (1.30), представлява филтър на линията на задържане с изходи (tapped-delay-line filter), показан на фиг.1.22, а за $M = 2$. Аналогично може да се представи и решетъчния филтър (lattice filter), показан на фиг.1.22, б, коефициент наречен коефициент на отражение (reflection coefficient). Между коефициентите на авторегресия и коефициентите на отражение съществува еднозначно съответствие. В двета модела се предполага, че входния сигнал $x(n)$ е комплексна величина (както в случая с кохерентния радар) и се отнася за този радар, за който коефициентите на авторегресия и коефициентите на отражение са комплексни. Звездичката в израза (1.30) и фиг.(1.22) обозначава комплексно спрягане. Данните от кохерентния радар могат да се опишат с множество коефициенти на авторегресия или съответстващото им множество на коефициентите на отражение. Последните имат определено предимство в плана за сложността на изчисление. За него съществува ефективен алгоритъм за получаване на резултати непосредствено от входните данни. Разделението на признаките се усложнява от този факт, че движещите се обекти се характеризират с променлива Доплерова честота, зависеща от скоростта на обекта относно радара и създава изкривяване в спектъра на коефициента на отразяване по който се определят признаките. Чрез използването на инвариантност на Доплер (Doppler invariance) се избягват тези усложнения. Ъгълът на фазата на първия коефициент на отражение се приема за равен на Доплеровата честота на сигнала на радара. За всеки коефициент се изпълнява нормализация относно Доплеровата честота. За това се взема ново множество от коефициенти на отражение κ_m свързани с множеството изходни коефициенти на отражение κ_m с израза

$$\kappa'_m = \kappa_m e^{-jm\theta}, \quad m = 1, 2, \dots, M, \quad (1.31)$$

Където Θ - фазов ъгъл на първия коефициент на отражение. Операцията описана в израза (1.31) се нарича хетеродинироване (heterodyning). Избора на инвариантните на изместване по Доплер признаки (Doppler invariant radar feature) се представя от нормализираните коефициенти на отражение $\kappa'_1, \kappa'_2, \dots, \kappa'_m$, където κ'_1 - единствен коефициент на това множество с веществено значение. Сигналите от ехото на самолет и от ехото на повърхността на земята се отличават по малкото изместване по Доплер. Следователно класификатора на радара трябва да съдържа постпроцесор (фиг.1.23) – той обработва резултатите от класификацията с цел да идентифицира класа на Земята. Препроцесора (preprocessor) показан на фиг.1.23. обезпечава инвариантност на признаките по отношение на изместването по Доплер, а в същото време постпроцесора използва изместването по Доплер за разделяне на обектите „самолет“ и „земя“ в изходния сигнал.



Фиг. 1.23. Инвариантно към изместване по Доплеровия класификатор на сигнала на радара

Още по-добър пример за ехолокация на невронните мрежи е биологичната система на ехолокацията на прилепите. Много от тях използват сигнали с *честота на моделиране* (frequency modulation), или *FM-сигнали* (frequency modulated signal) за създаване на акустична картина на обкръжаващото пространство. Постоянната честота на този сигнал се изменя във времето. В частност, прилепът с помощта на устата изпуска крайни FM-сигнали, а органите на слуха ги използват в качеството на приемане на ехото. Ехото от интересуващата цел се предава на слуховия апарат като активност на невроните, отговарящи за различни акустични параметри. В слуховият апарат на прилепа информацията се предава по три основни характеристики:

- *Честота на ехосигнала* (echo frequency). Кодира се честотата на картина от ушния охлюв. Тя се съхранява по целия път на сигнала по слуховия апарат и се отделя от отделни неврони настроени на отделна честота.
- *Амплитуда на ехосигнала* (echo amplitude). Тя се кодира от други неврони, имащи различни динамични характеристики. Те отделят температурната характеристика и количеството на отговорите дошли в резултат като сигнал от един зададен въпрос.
- *Забавяне на ехосигнала* (echo delay). Кодира се чрез невронни изчисления (основани на взаимна корелация).

Двете основни характеристики, които се използват за формиране на изображения са *спектър* (spectrum) за дадена форма на обекта и *забавяне* (delay). Прилепът формира „форма“ на обекта в термините на временно получени отразен сигнал от различни отразяващи повърхности на обекта. Тази честотна информация, съдържаща се в спектъра на ехосигнала се преобразува в оценка на *временна структура* (time structure) на обекта. Този процес се състои от временни и честотно-временни преобразования, в резултат на които се формира обща задръжка за възпроизвеждане обект.

1.8. Изкуствен интелект и невронни мрежи

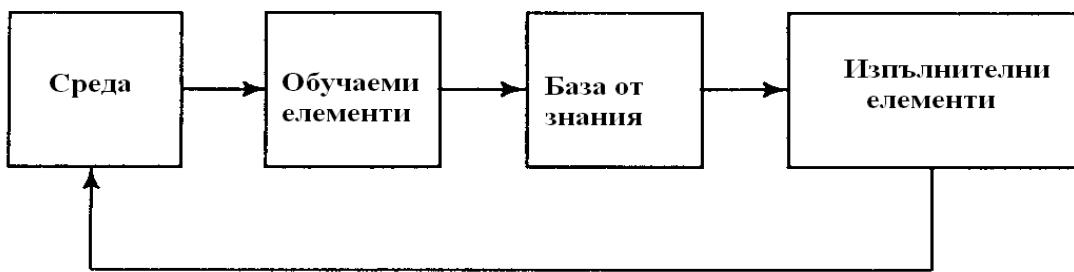
Основната задача на изкуствения интелект (artificial intelligence – AI) е разработване на парадигма или алгоритъм, обезпечаващи конкретни решения на конгнитивни задачи, свойствени за човешкия мозък.

Системите на изкуствения интелект трябва да обезпечават решението на следните три задачи: придобиване на знания, използване на натрупаните знания за решение на проблеми и извлечане на знания от опита. Системите на изкуствения интелект реализират три ключови функции: представяне, разсъждение и обучение. (фиг.1.24).



Фиг.1.24. Трите ключови функции на системите с изкуствен интелект

1. *Представяне* (representation). Една от отличителните черти на системите с изкуствен интелект е използване на *символен език* (symbol structure) за представянето на общи знания от предметната област и конкретни знания за способите за решаване на задачи. Символите се формират във вече известни термини. Това прави символното представяне относително просто и понятно за човека. Символните системи на изкуствения интелект са пригодни за човеко-машинно общуване. Терминът „знание”, използван за създаване на системи с изкуствен интелект се явява друго название на данните. Знанията могат да имат процедурен и декларативен характер. В *декларативния* (declarative) характер са представени знания за статистически събрани факти. При това съществуват относително малък обем от процедури, използвани за манипулиране на тези факти. Характерна особеност на декларативното представяне се явява това, че в очите на човека то има смисъл само по себе си, независимо от използването му в системите с изкуствен интелект. В *процедурното* (procedure) представяне знанията са внедрени в процедури, функциониращи независимо от смисъла на самите знания. В повечето области се изискват едновременно двата типа представяне на знания.
2. *Разсъждение* (reasoning). Под разсъждение обикновено се разбира способността за решаване на задачи. За да се нарича системата разумна, тя трябва да удоволетворява следните условия:
 - Да описва и решава широк спектър от задачи.
 - Да разбира *явна* (explicit) и *неявна* (implicit) информация.
 - Да има механизъм за *управление* (control), определящ операциите, използвани за решение на отделните задачи.



Фиг. 1.25. Прост модел за обучение на машина

Решението на задача може да се разглежда като задача за *търсене* (searching problem). В процеса на търсене се използват *правила* (rules), *данни* (data) и *управляващи взаимодействия* (control). Правилата действат на областта на данните, а управляващите взаимодействия се определят от правилата. За пример ще разгледаме задачата за намиране на най-кратък път от един град до друг град. При това всеки град трябва да се посети само един път. В тази задача множеството от данни се състои от всички възможни маршрути и техните стойности, представени във формата на претеглен граф. Правилата определят пътя на движение от единния град до другия, а модулите за управление решават кога какви правила да се прилагат. В много практически задачи (например в медицинското диагностициране) натрупаните знания се явяват непълни или неточни. В такива ситуации се използват *вероятностни разсъждения* (probabilistic reasoning), позволяващи системите с изкуствен интелект да работят в условия на неопределеност.

3. *Обучение* (learning). В простия модел на машинно обучение (фиг.1.25), информацията за *обучавания елемент* (learning element) използва получената информация за модернизиране на *базата знания* (knowledge base), знанията за всеки *функционален елемент* (performance element), а след това се използват за изпълнение на поставената задача. Информацията постъпваща от външната среда е несъвършена, затова обучавания предмет в началото не знае как да запълни пропуските или да отстрани несъществуващите детайли. Машината действа на налучковане и след това получава сигнал за *обратна връзка* (feedback) от функционалните елементи. Механизмът за обратна връзка позволява на системата да провери хипотезите и да ги прегледа при необходимост.

Машините за обучение може да включват два съвършено различни способа за обработка на информация: *индуктивен* (inductive) и *дедуктивен* (deductive). При индуктивния обработката на информация на общи шаблони и правила се създава на основа на практическия опит и потока от данни. При дедуктивния обработката на информация за определени конкретни факти се изпълнява по общите правила. Обучението на основата на подобие предствалява индуктивен процес, а доказателството на теореми – дедуктивен, т.к. то се основава на вече известни аксиоми и доказани теореми. В обучението на основата на обяснение се използва както дедуктивен, така и индуктивен метод.

Възникващите по време на обучението усложнения и натрупания опит са довели до създаването на различни методи и алгоритми на попълване без знания. В частност ако в дадена предметна област работят опитни професионалисти по-просто е да се получи обобщен опит, вместо да се дублира техния експериментален път, който те са получили в процеса на натрупване. Тази идея се прилага в *приложните системи* (expert system).

Възниква въпросът: как сравнително когнитивни модели на невронната мрежа са символните системи на изкуствения интелект? За това сравнение проблемът ще бъде разделен на три части: ниво на обяснение, стил на обработка и структурно представяне.

- *Ниво на обяснение* (explanation level). Класическите системи за изкуствен интелект са основани на символното представяне. Изкуственият интелект изучава ментални представления, в които познанието се осъществява като *последователна обработка* (sequential processing) на символната информация. В

центъра на вниманието на невронната мрежа се намират модели на успоредна разпределена обработка (parallel distributed processing, PDP). В тези модели се предполага, че обработката на информацията протича за сметка на взаимодействието на голямо количество неврони, всеки от които предава сигнали на възбудждане и възприемане на други невронни мрежи. Освен това в теорията за невронните мрежи се отделя голямо внимание на невробиологичното описание на процеса на познание.

- *Стил на обработка* (processing style). В класическите системи за изкуствен интелект обработката протича *последователно* (sequential), както е в традиционното програмиране. Ако нивото на изпълнение на действия не е строго определено (например, при сканиране на правила и факти в експерименталните системи), операциите така или иначе се изпълняват стъпка по стъпка. Такава последователност на обработка се обяснява с последователната природа на естествените езици и логическите заключения. Такава е структурата на машината на фон Нойман.

За разлика от тях, концепцията за обработка на информацията в невронните мрежи протича на принципа на *паралелизма* (parallelism), който се явява източник на тяхната повърхност. Благодарение на него, успоредността може да бъде масова (стотици хиляди неврони), което придава на невронните мрежи особена форма на работоспособност. За изчислителните разпределения между множеството неврони практически не е важно, че състоянието на невронната мрежа се отличава от очакваното. Зашумен или непълен входен сигнал може да се разпознае еднакво; повредената мрежа може да изпълнява своите функции на удовлетворително ниво, обучението не трябва да бъде съвършено. Производителността на мрежите в пределите на някакъв диапазон се снижава достатъчно бавно. Това може допълнително да повиши работоспособността на мрежите, представени като собствена група от неврони.

- *Структурно представяне* (representational structure). В системи с изкуствен интелект в качеството на модели се използва езика на мислене, затова символното представяне има квази-лингвистична структура. Подобно на фразите на обикновения език, изразите на системите с изкуствен интелект по правило са сложни и се съставят по пътя на систематизация на прости символи. Отчитайки огромното количество на символите, новите смислови изречения се строят на основата на композиции на символични изрази и аналогии между синтактичните структури и семантики.

От друга страна в невронните мрежи природата и структурата на представяне се явяват ключови проблеми. Невронните мрежи не удовлетворяват два основни критерия на процеса на познание: *природа на мисловно представяне* (mental representation) и *мисловни процеси* (mental process). В съответствие с това, следващите свойства са присъщи за системите с изкуствен интелект, но не са присъщи на невронните мрежи.

- a) Мисловното представяне се характеризира с комбинирана избирателна структура и комбинирана семантика;
- b) Мисловните процеси се характеризират с чувството към комбинирано структурно представяне, с което те работят.

Такива образи от символни модели на изкуствения интелект са формални системи, основани на използването на езика на алгоритмите и представянето на знания по принципа „отгоре надолу“ (top-down), а в невронните мрежи – това успоредно разпределение на процесори, има естествената способност към обучение и работи на принципа „отдолу нагоре“ (bottom-up). При решаването на когнитивни задачи е целесъобразно създаването на *структурни модели на основата на връзки* (structured connectionist models) или *хибридни системи* (hybrid system), обединяващи двета подхода. Това ще обезпечи свойствата на адаптивност, работоспособност и единообразие присъщи на невронната мрежа, с представянията, умозаключенията и универсалността на системите с изкуствен интелект. За реализацията на този подход са разработни методи методи на изваждане на правила от обучени на невронни мрежи. Тези резултати не само използват интегрирането на невронни мрежи с интелигентни машини, но и обезпечават решението на следните задачи:

- a) Верификация на невронно-мрежовите коефициенти в програмните системи. За това вътрешното състояние на невронните мрежи се превежда във форма позната на потребителите.
- b) Подобрявайки обобщаващата способност на невронните мрежи за сметка на откривнето на областите на входното пространство, недостатъчно пълно представени в обучаващото множество, а също определяне на условия, при които обобщението е невъзможно.
- c) Откриване на структурната зависимост на множеството входни данни.
- d) Интеграция на символния и конекционисткия подход при разработването на интелектуални машини.
- e) Обезпечаване на безопасност на системата, за която тя се явява критична.

Глава II: Процеси на обучение

1. Въведение

Най-важното свойство на невронните мрежи е способността им да се обучават (learn) на основата на данните от околната среда и в резултат на обучението да повишат своята производителност. Повишаването на производителността протича в течение на времето в съответствие с определени правила. Обучението на невронните мрежи протича посредством интерактивен процес на корекция на синаптическите тегла и прагове. В идеалният случай невронните мрежи получават знания от околната среда на всяка итерация в процеса на обучение.

С понятието обучение се асоциират много видове действия, затова е много сложно да се определи еднозначно. Процеса на обучение зависи от неговата гледна точка.

Обучение – това е процес, в който свободните параметри на невронната мрежа се настройват посредством моделирана среда в която е построена тази мрежа. Типа на обучението се определя от способа на построяване на тези параметри.

Това определение за процеса на обучение предполага следната последователност на събития:

1. В невронната мрежа постъпват стимули от външната среда.
2. В резултат на което се изменят свободните параметри на невронната мрежа.
3. След изменение на външната структура на невронната мрежа тя отговаря на възбуддането вече по друг начин.

Тези правила се наричат *алгоритъм за обучение* (learning algorithm). Съществува набор от средства представени от множеството на алгоритъма на обучение, всяко от които има своето достойнство. Алгоритмите за обучение се отличават един от друг по способността на нарастване на синаптическите тегла на неврона и връзките на обучените неврони с външната среда. В този контекст се определят *парадигмите на обучение* (learning paradigm) свързани с моделите на околната среда, в която функционират данните на невронната мрежа.

2. Обучение основано на корекция на грешките

За да илюстрираме първото правило на обучението ще разгледаме прост случай на неврона k – единственият изчислителен възел на изходния слой на невронната мрежа за пряко разпространение (фигура 2.1a.).

Неврона k работи под управлението на *вектора сигнал* $x(n)$ произвеждащ един или няколко скрити слоя на невроните, които получават информацията от входният вектор (възбуддане) предава на началните възли (на входният слой) на невронните мрежи. Под n се разбира дискретно време (или) номер на стъпката на интерактивният процес на настройване на синаптическите тегла на неврона k . *Изходният сигнал* (output signal) на неврона k се обозначава с $y_k(n)$. Този сигнал се явява единствен на изхода на невронните мрежи. Той ще бъда сравняван с *желания изход* (desired response), обозначен с $d_k(n)$. В резултат се получава *сигнал за грешка* (error signal) $e_k(n)$.

$$e_k(n) = d_k(n) - y_k(n). \quad (2.1)$$

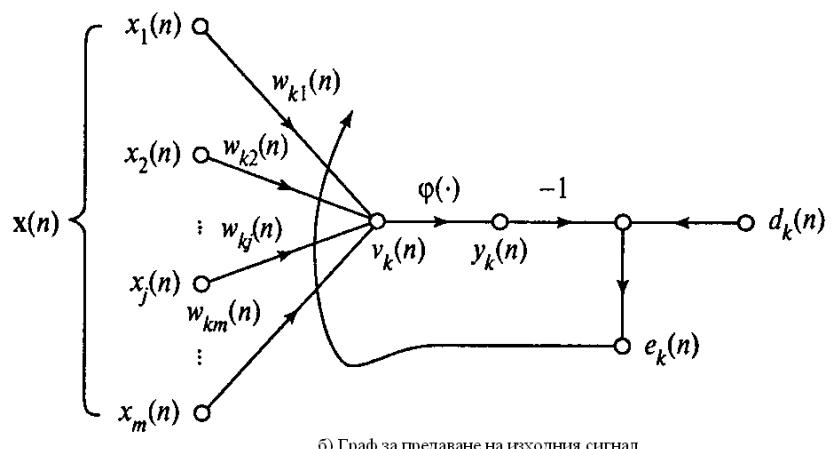
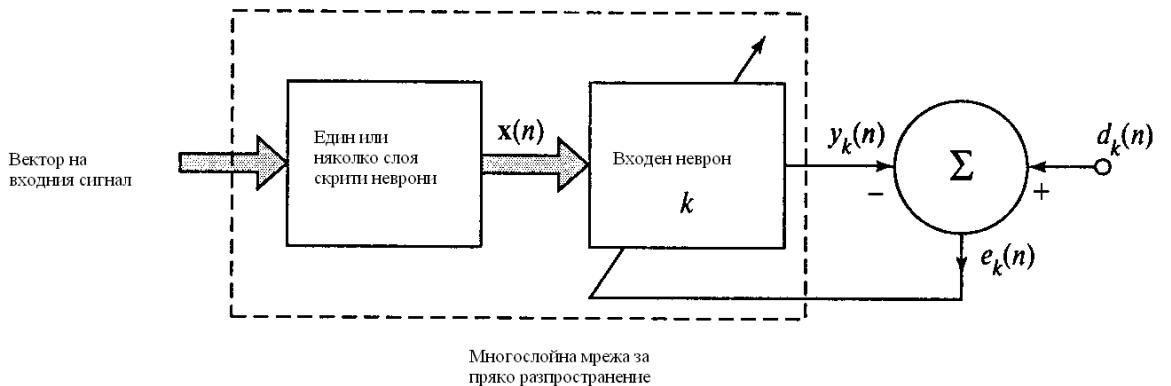
Сигналът за грешка инициализира *механизмът за управление* (control mechanism), чиято задача е да използва последователностите на корекциите на синаптическите тегла на неврона k . Тези изменения са насочени на стъпково приближаване на изходният сигнал $u_k(n)$ към желания $d_k(n)$. Тази цел се постига за сметка на минимизация на *функциите от стойности* (instantaneous value of the error energy) или *индекса на производителност* (performance index) $E(n)$, определен в термините на сигналите за грешки изглежда така:

$$E(n) = \frac{1}{2} e_k^2(n), \quad (2.2)$$

Където $E(n)$ е *текущата стойност на грешки на енергията* (instantaneous value of the error energy). Стъпковото коректиране на синаптическите тегла на неврона k се продължава докато системата не постигне *устойчиво състояние* (steady state) (такова при което синаптическите тегла практически се стабилизират). В тази точка процеса на обучение се спира.

Процесът, описан по горе се нарича *обучение, основано на корекция на грешките*. Минимизацията на функциите от стойности $E(n)$ се изпълнява по т.нар делта – правило, или правило на Видро – Хоф. Обозначаваме с $w_{kj}(n)$ текущата стойност на синаптическите тегла w_{kj} на неврона k съответстващи на елементите $x_j(n)$ от вектора $x(n)$ на стъпка на дискретизация n . В съответствие с делта-правилото на изменение се прилага към синаптическото тегло на на тази стъпка на дискретизация изразена чрез формулата $\Delta w_{kj}(n)$ се пресмята по следния начин:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n), \quad (2.3)$$



Фиг.2.1. Обучение основано на корекция на грешките където η е някая положителна константа определяща *скоростта на обучение* (rate of learning) използвана при прехода от една стъпка към друга. От формула 2.3. се вижда, че тази константа може да се нарече *параметър на скоростта на обучение* (learning rate parameter). Делта правилото може да се определи по следният начин: корекцията която приема синаптическото тегло на неврона е пропорционална на произведението от сигналите от грешките на входният сигнал и изходния.

Определеното по този начин правило дава възможност за *пряко измерване* (direct measure) на сигнала за грешка. За обезпечаване на такова измерване е нужно да настъпи желания резултат от някой външен източник непосредствено достъпен за неврона k , т.е. неврона k трябва да е *видим* (visible) за външния свят (фиг. 2.1,a). На фигурата се вижда, че обучението основано на корекция на грешки по своята природа е *локално* (local). То показва, че корекцията на синаптическите тегла по делта-правилото може да бъде локализирано в отделния неврон k .

Изчислявайки величината на изменение на синаптическото тегло $\Delta w_{kj}(n)$ можем да определим нова стойност на следващата стъпка на дискретизация

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n). \quad (2.4)$$

Тези образи $w_{kj}(n)$ и $w_{kj}(n + 1)$ могат да се определят като стара и нова стойност на синаптическото тегло w_{kj} . В математически вид може да бъде записано:

$$w_{kj}(n) = z^{-1}[w_{kj}(n + 1)], \quad (2.5)$$

Където z^{-1} е *оператор за единично задържане* (unit delay operator). Нарича се още *елемент на паметта* (storage element).

На фиг. 2.1,б е представен граф на протичане на сигнала в процеса на обучението обоснован на корекция на грешките за отделен неврон k . Входният сигнал x_k и индукционното локално поле v_k на неврона k се представят във вид на *предсинаптически* (presynaptic) и *постсинаптически* (postsynaptic) сигнали на j -те синапси на неврона k . На фигурата се вижда, че обучението основано на корекция на грешките е пример за затворена система с *обратна връзка* (closed-loop feedback).

2.3. Обучение на основата на памет

При *обучението основано на паметта* (memory-based learning) всички предни опити са се натрупали в голямо хранилище за правилно класифициране на примери от вида вход-изход $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, където \mathbf{x}_i – входен вектор, а d_i съответстващия на него желан изходен сигнал. Не ограничаваме общността и може да предположим, че изходният сигнал се явява скаларен. Например задачата за бинарно разпознаване на образи или класификация на два класа (хипотези) \mathbf{C}_1 и \mathbf{C}_2 . В този пример желаният резултат на системата d_i приема стойности 0 (или -1) за класа \mathbf{C}_1 и стойност +1 за класа \mathbf{C}_2 . Ако е необходимо да се класифицират някой неизвестен вектор \mathbf{x}_{test} от база данни се избира изход, съответстващ на входният сигнал близък до \mathbf{x}_{test} .

Алгоритъмът на обучението основано на паметта включва две характеристики:

- Критерии използван за определяне стойностите на вектора \mathbf{x}_{test} .
- Правило за обучение, прилагано към примера за изчисляване на вектора

В простия алгоритъм обучението основано на памет се нарича *правило на близкият съсед* (nearest neighbor rule), като в изчисленията се включва пример близък до тествания

$$\mathbf{x}'_N \in \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} \quad (2.6)$$

се счита близък със съседният вектор \mathbf{x}_{test} , ако е изпълнено условието:

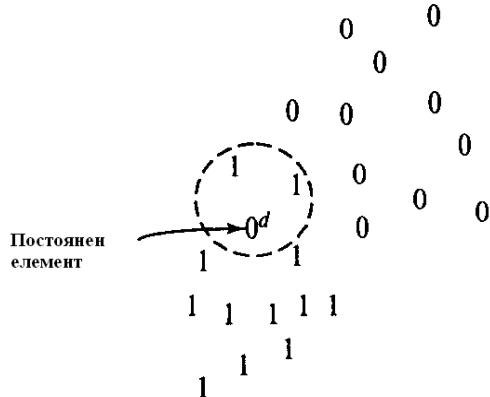
$$\min_i d(\mathbf{x}_i, \mathbf{x}_{test}) = d(\mathbf{x}'_N, \mathbf{x}_{test}), \quad (2.7)$$

Където $d(\mathbf{x}_i, \mathbf{x}_{test})$ е Евклидово разстояние между векторите \mathbf{x}_i и \mathbf{x}_{test} . Класът към който се отнасят близкият съсед се счита също за клас на тестваният вектор \mathbf{x}_{test} . Това правило не зависи от разпределенията използвани при генерирането на примери за обучение.

При анализа на правилото се използват две предположения:

- Класификацията на примерите (\mathbf{x}_i, d_i) са *независими и равномерно разпределени* (independently and identically distributed) в съответствие с съвместното разпределение на примера (\mathbf{x}, d) .
- Размерността на обучаващото множество N е безкрайна величина.

При тези две предположения вероятността от грешки на класифициране при използване на правилото на близкия съсед превишава два пъти *Байсовската вероятност за грешки* (Bayes probability error) (Байсовската вероятност за грешки е минималната вероятност за грешки на множеството на всички правила приети за решение).



Фиг. 2.2. Областта заета от окръжността съдържа две точки, принадлежащи към клас 1, и една принадлежаща към класа 0. Точка d съответства на тестования вектор \mathbf{x}_{test} . При $k = 3$ класификатора на k -близкия съсед ще отнесе точка d към клас 1, въпреки че тя може да е по-близка към „избора”, отнасящ се към класа 0.

Вариации на класификация основани на близкия съсед се явява *класификатор на k -близкия съсед* (k -nearest neighbor classifier). Той се описва по следният начин: Намираме k -класифициран съсед близък на вектора \mathbf{x}_{test} , където k е някое цяло число.

Вектора \mathbf{x}_{test} отнасяме към този клас (хипотеза) който по често от другите среща k – близкия съсед на тестования вектор. Така класифицирането основано на k -близкият съсед работи подобно на устройството на осреднения. Например може да се отчете единично „изхвърляне”, както е показано на фиг. 2.2. за $k = 3$ изхвърляне (outlier). Това изхвърляне е наблюдение, което се отличава от номиналния модел.

2.4. Обучение на Хеб

Постулат на обучението на Хеб (Hebb's postulate of learning):

Ако всеки аксон на клетката А намиращ се на достатъчно близко разстояние до клетката В и постоянно или периодично участва и се възбуджа се наблюдава процес на метаболически изменения на един или на два неврона, изразяващ се в това, че ефективността на неврона А нараства когато един от възбудените неврони В нараства.

Хеб предложил наблюдението да е в основата на асоциативното обучение. Според него това би довело до постоянна модификация на шаблоните на активност на пространствено – разпределените „ансамбли от невронни клетки”. Това може да се потвърди в невробиологичен контекст и може да се перефразира в следващото правило, състоящо се от две части:

- Ако два неврона от двете страни на синапса (съединение) се активизират едновременно (синхронно), то силата на това съединение нараства.
- Ако два неврона от двете страни на синапса се активизират асинхронно то такъв синапс отслабва или въобще не съществува.

Функциониращият по такъв начин синапс се нарича *синапс на Хеб* (Hebbian synapse). Синапса на Хеб използва зависещи от времето във висша степен локални механизми за взаимодействие, изменящи ефективността на синаптическите съединения

в зависимост от корелациите между предсинаптическа и постсинаптическа активност. От това определение могат да се извадят четири свойства (ключови механизми), характеризиращи синапса на Хеб:

1. *Зависимост от времето* (time-dependent mechanism). Синапсът на Хеб зависи от точното време на възникване на предсинаптическия и постсинаптическия сигнал.
2. *Локалност* (local mechanism). По своята природа синапсът се явява възел за предаване на данни, в който информационните сигнали (представляващи текуща активност на предсинаптическите и постсинаптическите елементи) се намират в *пространствено – временна близост* (spatiotemporal). Тази локална информация се използва от синапса на Хеб за обновяване на локалната синаптическа модификация характерна за дадения входен сигнал.
3. *Интерактивност* (interactive mechanism). Измененията в синапса на Хеб се определят от сигнала на двата негови края, т.е. формата на обучение по Хеб зависи от степента на взаимодействие с подсинаптическите и предсинаптическите сигнали. Такава зависимост или интерактивност може да носи детерминиран или статистически характер.
4. *Корелация* (correlational mechanism). Една от интерпретациите на обучението на Хеб се състои в това, че условиято за изменение на ефективността на синаптическите връзки се явява зависимост между предсинаптическите и постсинаптическите сигнали. В съответствие с това за модификацията на синапса е необходимо да се обезпечат постсинаптическите и предсинаптическите сигнали. По тази причина синапса на Хеб се нарича *конюнктивен синапс* (conjunctional synapse). Друга интерпретация на обучението на Хеб използва характерния за синапса на Хеб механизъм на взаимодействие в статическите термини. В частност синаптическите изменения се определят от корелация на предсинаптическите и постсинаптическите сигнали във времето. Отчитайки това понякога синапса на Хеб се нарича *корелационен синапс* (correlational synapse). Самата корелация се явява основа на обучение.

Усиливане и отслабване на синаптическите връзки

Синапса на Хеб не отчита допълнителни процеси, които могат да отслабят връзките между два неврона. Можем да обобщим концепцията за модификацията на връзките на Хеб, използваш факта, че положителната корелационна функция води до усиливане на синаптическите връзки, а отрицателната корелация отсъства или отслабва. Отслабването на синаптическите връзки може да има и не интерактивен характер, в частност условията на взаимодействие на отслабването на синаптическите връзки може да носи случаен характер и не зависи от предсинаптическите и постсинаптическите сигнали.

Систематизиратки тези определения могат да се въведат следващите модели за модификация на асинаптическите връзки: *хебовски* (Hebbian), *антихебовски* (anti-Hebbian) и *нехебовски* (non-Hebian). Следвайки тази схема можем да кажем, че синаптическите връзки по модела на Хеб се усилват при положителни корелации на предсинаптическите ескаи постсинаптически сигнали и отслабват при противен случай. И в двата случая на двата модела изменението на синаптическата ефективност се основава на зависещи от времето локални и интерактивни по своята природа механизми. В този смисъл анти хебовският модел е хебовски по природа, но не и по

функционалност. Нехебовският модел въобще не е свързан с механизма модификация предложена от Хеб.

Математически модели на пердложения от Хеб механизъм за модификация на синаптическите връзки

За да опишем обучението на Хеб в математически термини разглеждаме синаптическото тегло w_{kj} на неврона k с предсинаптическите и постсинаптическите сигнали x_j и y_k . Изменението на синаптическото тегло w_{kj} в момент от време n може да се изрази чрез:

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n)), \quad (2.8)$$

Където $F(\cdot, \cdot)$ е някоя функция зависеща от предсинаптическите и постсинаптическите сигнали. Сигналите $x_j(n)$ и $y_k(n)$ често се приемат без оглед на размера.

Хипотеза на Хеб

Проста форма на обучението на Хеб:

$$\Delta w_{kj}(n) = \eta y_k(n)x_j(n), \quad (2.9)$$

Където η е положителна константа, определяща *скоростта на обучение* (rate of learning). Изразът ясно подчертава корелационната природа на синапса на Хеб. Нарича се още *правило за умножение на активности* (activity product rule). Фиг. 2.3. илюстрира графично формула (2.9). Тя представя зависимостта на изменението на $\Delta w_{kj}(n)$ от входния сигнал (предсинаптическа активност) $y_k(n)$.

Хипотеза на ковариациите

Един от способите преодолял ограниченията на Хеб е използването на *хипотезата на ковариациите* (covariance hypothesis). Съгласно тази хипотеза, предсинаптическите и постсинаптическите сигнали в формула (2.9) се заменят със техните сигнали със средни стойности за дадено време.

$$\Delta w_{kj} = \eta(x_j - \bar{x})(y_k - \bar{y}), \quad (2.10)$$

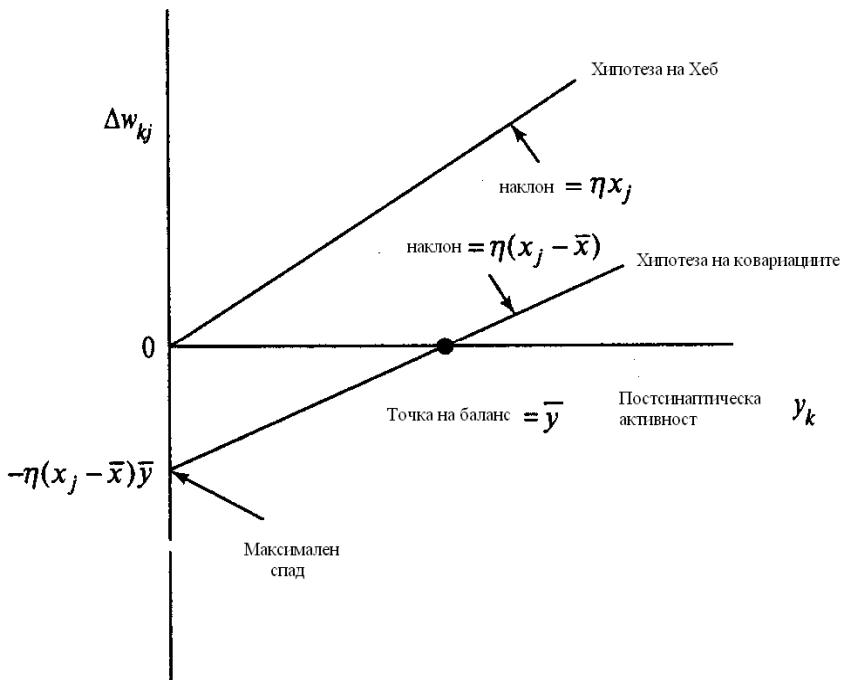
Където η параметър на скоростта на обучение. Средните стойности на x и y съдържат предсинаптически и постсинаптически сигнали, определящи знака на синаптическата модификация. В частност хипотезата на ковариациите обезпечава следното:

- Сходимостта е нетривиално състояние, което се постига при $x_k = \bar{x}$ и $y_j = \bar{y}$.
- Прогнозирани *усилвания* (potentiation) и *отслабвания* (depression) на синаптически връзки.

На фиг. 2.3. е показана разликата между хипотезата на Хеб и хипотезата на ковариациите. В този случай величината зависи от Δw_{kj} от y_k и се явява линейна относно пресичането с оста y_k за хипотезата на Хеб в началото на координатната система, а за хипотезата на ковариациите – в точки $y_j = \bar{y}$.

При внимателно изследване на израз 2.10. може да се видят следните изводи:

- Синаптическите тегла се усилват при високо ниво на предсинаптическите и постсинаптическите сигнали т.e. в този случай, когато се удовлетворяват следните условия $x_j > \bar{x}$ и $y_k > \bar{y}$.



Фиг.2.3. Илюстрация на хипотезата на Хеб и на хипотезата на ковариациите

- Синаптическите тегла отслабват при следните ситуации:
 - предсинаптическата активност ($x_j > \bar{x}$) не предизвика съществена постсинаптическа активност ($y_k < \bar{y}$)
 - постсинаптическата активност ($y_k > \bar{y}$) възниква при отсъствие на съществена предсинаптическа активност ($x_j < \bar{x}$).

Такова поведение може да се разглежда като форма на временна конкуренция между входни модели.

Съществува строго физиологично доказателство за реализиране на принципите на обучение на Хеб в областта на мозъка наречено *хипокампус* (hippocampus). Тази област играе важна роля в някои аспекти на обучението и запомнянето.

2.4. Конкурентно обучение

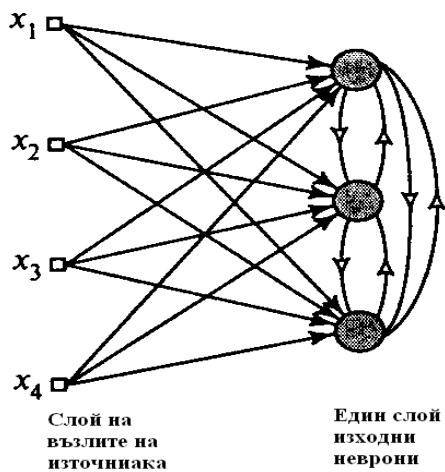
Както следва от самото наименование, в *конкурентното обучение* (competitive learning), изходните неврони на невронните мрежи се конкурират помежду си за правото да бъдат активни. Ако невронната мрежа е основана на обучението на Хеб, едновременно във възбудено състояние могат да се намират няколко неврона, а в конкурентните мрежи във всеки един момент от време може да е активен само един неврон. От това свойство следва, че конкурентното обучение е удобно за изучаване на статистическите свойства използвани в задачите за класификация на образи.

Правилото на конкурентното обучение се основава на три основни елемента:

- Множеството еднакви неврони със случаен разпределение синаптически тегла водят до различни реакции на невроните на един и същ входен сигнал.
- Пределна стойност (limit) на някой неврон

- Механизъм позволяващ на невроните да се *конкурират* за правото на отговор на дадено подмножество на входния сигнал и определят единственият активен изходен неврон (или по един неврон на група). Неврона победител в това съревнование се нарича неврон-победител, а принципа на конкурентното обучение формулира лозунга „победителя взема всичко”.

Така на всеки отделен неврон от мрежата съответстват група от близки образи, при това невроните се превръщат в детектор на признаките на различните класове от входни образи. Най-простата невронна мрежа с конкурентно обучение съдържа единствен слой изходни неврони, всеки от които е съединен с входни възли. В такива случаи могат да съществуват обратни връзки между невроните (фиг. 2.4.). В подобна архитектура обратната връзка обезпечава латерално спиране, когато всеки неврон се стреми да спре свързаните с него неврони.



Фиг. 2.4. Архитектура на граф на проста мрежа на конкурентно обучение с прави (възбудждащи) връзки от входните възли на невроните и обратни (задържащи) връзки между невроните. (Последните са обозначени на рисунката с незашрихованни стрелки).

Преките синаптически връзки, показани на фиг.2.4, се явяват *възбудждащи* (excitatory).

За да победи дадения неврон k в конкурентната борба за индукционното локално поле v_k за зададения входен образ x трябва да бъде максимален в средата на цялата невронна мрежа. Тогава изходният сигнал y_k на неврона победител k преминава в стойност 0. Това може да се запише така:

$$y_k = \begin{cases} 1, & \text{ако } v_k > v_j \text{ за всяко } j, j \neq k, \\ 0 & \text{в останалите случаи} \end{cases} \quad (2.11)$$

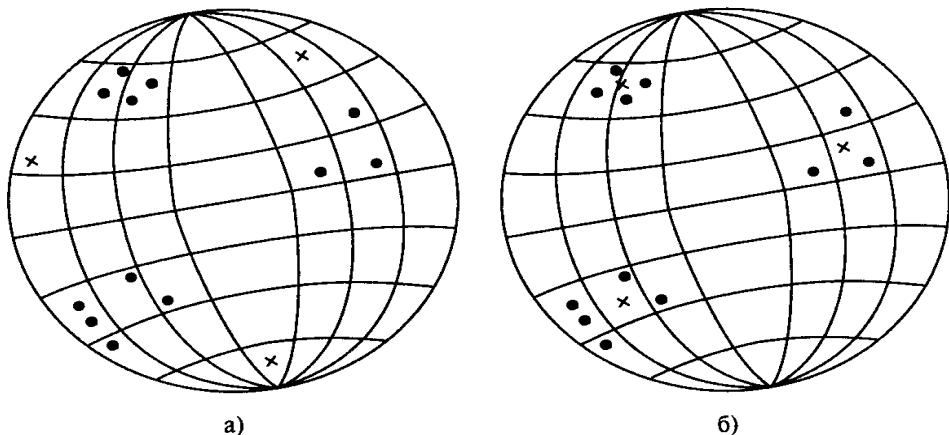
Където индукционното локално поле v_k представлява свободно възбудждане на неврона k от всеки входен сигнал и сигнал за обратна връзка.

Нека w_{kj} е синаптическото тегло на връзката на входния възел j с неврона k . Предполагаме, че синаптическите тегла на всички неврони са фиксирани (положителни), при това:

$$\sum_j w_{kj} = 1 \quad \text{за всяко } k. \quad (2.12)$$

Тогава обучението на този неврон се състои в изместване на синаптическите тегла от неактивните към активните входни възли. Ако неврона не формира отговор на

конкретен входен образ, то той не се обучава. Ако някой неврон побеждава в конкурентната борба, то теглото за връзка на този неврон равномерно се разпределя между неговите активни входни възли, а връзките с неактивните входни възли.



Фиг.2.5. Геометрична интерпретация на процеса на конкурентно обучение.

Точките представляват входни вектори, а кръсчетата – вектори на синаптическите тегла на три изходни неврона в изходен (а) и крайно състояние на мрежата (б).

Съгласно *правилото за конкурентно обучение* (competitive learning rule) изменението Δw_{kj} на синаптическите тегла w_{kj} се определя от следния израз:

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & \text{ако неврона } k \text{ побеждава в саревнованието} \\ 0, & \text{ако неврона } k \text{ побеждава в саревнованието} \end{cases} \quad (2.13)$$

Където η е параметър на скоростта на обучение. Това правило отразява изместването на вектора на синаптическите тегла w_k на победилия неврона k в страната на входния образ x .

За илюстриране на съществуващото конкурентно обучение можем да използваме геометричната аналогия (фиг. 2.5.). Предполага се, че всички входни образи имат някаква постоянна Евклидова норма. Така те могат да се изброят във вид на точки в N-мерна единична сфера, където N е количеството на входните възли. N се явява размерност на векторите на синаптическото тегло w_k . Предполага се, че всички невронни мрежи имат същата Евклидова норма, т.е.

$$\sum_j w_{kj}^2 = 1 \quad \text{за всяко } k. \quad (2.14)$$

Ако синаптическите тегла са правилно мащабирани те формират набор от вектори, които се проектират на същата N-мерна единична сфера. На фиг. 2.5, а могат да се отделят три естествени групи кълстери от точки, представляващи входни образи. На фигурата е показано и вероятността на началното състояние на мрежата, отбелзано с кръстчета до началото на обучението. На фиг. 2.5, б е показано крайно състояние на мрежата получено като резултат на конкурентното обучение, което синаптическите тегла на всеки изходен неврон са изместени към центъра на тежестта на съответните кълстери.

От примера се вижда способността на невронните мрежи да решават задачи за кълстерилизация в процеса на конкурентното обучение. Устойчивите решения на задачите за входни образи са дължни да формират различни групи от вектори, иначе

мрежата може да стане неустойчива, тъй като в отговор на зададен входен образ ще се формират отзуци от различни изходни неврони.

Обучение на Болцман

Правилото на обучението на Болцман представлява стохастически алгоритъм на обучение основан на идеята на стохастически механизми. Невронните мрежи създадени на основата на обучението на Болцман се наричат *машина на Болцман* (Boltzmann machine).

В машините на Болцман всички неврони представляват рекурентни структури работещи с бинарни сигнали. Това значи, че те могат да се намират във включено (съответстващо на стойност +1) или изключено (съответстващо на стойност -1) състояние. Такава машина характеризира функцията от енергията E , която се определя от конкретното състояние на отделен неврон съставящ машината. Това може да се опише по формула

$$E = -\frac{1}{2} \sum_j \sum_{k(j \neq k)} w_{kj} x_k x_j, \quad (2.15)$$

Където x_j е състояние на неврона j ; w_{kj} - синаптическо тегло на връзките на невроните j и k . При условие, че $j \neq k$, то в мрежата невроните нямат обратна връзка със себе си. Работата на машината се състои в случаен избор на някой неврон (предполагаме, k -ти) на определена стъпка в процеса на обучение и привеждане на този неврон от състояние x_k , в състояние $-x_k$ при никаква температура T с вероятност

$$P(x_k \rightarrow -x_k) = \frac{1}{1 + \exp(-\Delta E_k/T)}, \quad (2.16)$$

Където ΔE_k - изменение на енергията на машината (т.е. изменение на енергията на функцията) получена от преходното състояние. Под температурата T се разбира не физическата температура, а псевдотемпература (pseudotemperature). При многократно прилагане на това правило машината достига *термално равновесие* (thermal equilibrium).

Невронната машина на Болцман може да бъде раздели на две функционални групи: *видими* (visible) и *невидими* (hidden). Видимите неврони реализират интерфейс между мрежата и средата с нейното функциониране, а скритите работят независимо от външната среда. Ще разгледаме двата режима на функциониране на такава мрежа:

- *Сковано състояние* (clamped condition), в което всички видими неврони се намират в състояние предопределено от външната среда.
- *Свободно състояние* (free-running condition), в които всички неврони (както видими, така и скрити) могат свободно да функционират.

Означаваме с ρ_{kj}^+ корелацията между състоянията на невроните j и k в сковано състояние. Аналогично ρ_{kj}^- означаваме корелация (correlation) между състоянията на невроните j и k , когато мрежата се намира в свободно състояние. Тези две корелации се осредняват по всички възможни състояния на машината, намираща се в условия на термално равновесие. Съгласно *правилото на обучение по Болцман* (Boltzmann learning rule), изменението Δw_{kj} на синаптическите тегла w_{kj} и връзките между невроните j и k се определят от следващият израз

$$\Delta w_{kj} = \eta(\rho_{kj}^+ - \rho_{kj}^-), \quad j \neq k, \quad (2.17)$$

Където η е параметър на скоростта на обучение. Стойностите на ρ_{kj}^+ и ρ_{kj}^- се изменят в диапазона от -1 до $+1$.

2.7. Задачи за присвояване на коефициента на доверие

При изучаване на алгоритмите на обучение на разпределени системи е полезно да се запознаем с концепцията *присвояване на коефициента на доверие* (credit assignment). По същество тази задача за присвояване на коефициента на доверие или на недоверие за всички резултати получени от някоя обучаема машина. (Задачите за *присвояване на коефициент на доверие* се наричат още *задачи за натоварване* (loading problem), т.е. разпределение на множеството обучаващи данни по свободните параметри на мрежата).

В много от случаите зависимостта на изходите от вътрешните решения се определя от последователността на действията изчислявани от обучаемата машина. С други думи вътрешните решения влияят на изпълнението на определени действия, след което именно тези действия, а не самите решения непосредствено определят общите резултати. В такава ситуация можем да изпълним декомпозиция на задачите за присвояване коефициента на доверие на две други подзадачи:

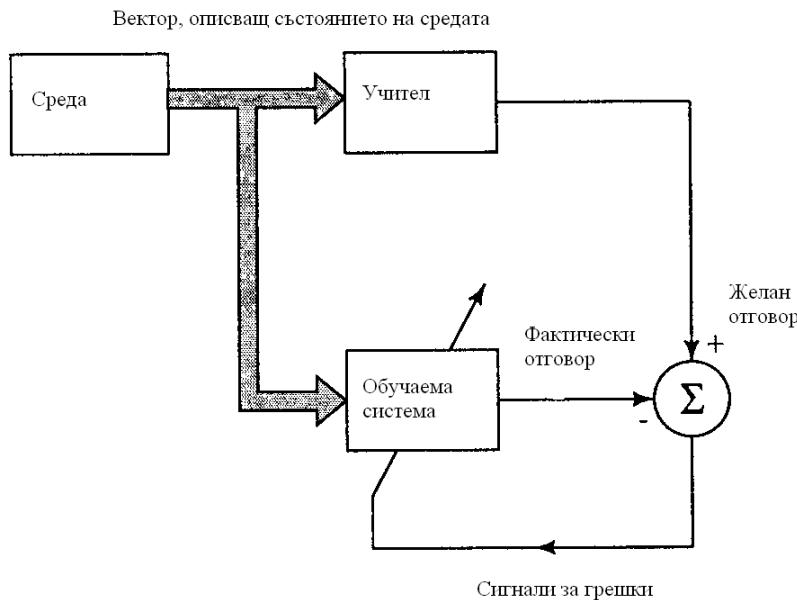
1. Присвояване на коефициентите на доверие от резултатите на действие. Тази задача се нарича *временна задача за присвояване на коефициентите на доверие* (temporal credit assignment problem). В нея се определя помеждутък от време, в течение на което реално се изпълняват действия, които отпуска кредит на доверие.
2. Присвояването на коефициените на доверие от действията за вътрешни решения. Това се нарича *структурна задача за присвояване на коефициентите на доверие* (structural credit-assignment problem). В нея коефициентите на доверие се наричат *вътрешни структури* (internal structures) на действията генериирани от системата.

Структурната задача за присвояване на коефициените на доверие има смисъл в контекста на многокомпонентните обучаеми машини. Когато е необходимо да се определи точното поведение на някой елементи от системата е нужно да се определи на каква степен, за да се повиши общата производителност на системата. От друга страна временната задача за присвояване коефициентите на доверие се поставя в този случай когато обучаемата машина изпълнява достатъчно много действия, довеждащи до някакъв резултат и е нужно да се определи кои от тези действия носят отговорност за резултата. Съчетаването на временната и структурната задача за присвояване на коефициените на доверие изискват усложняване на поведението на разпределителната обучаема машина.

Например задачата за присвояване на коефициените на доверие въниква в този случай когато обучението на основата на корекция на грешките се прилага към многослойните невронни мрежи за пряко разпространение. Действието на всеки скрит и изходен неврон от тази мрежа важи за формирането на правилен резултат в дадена област. Това значи, че за решението на поставената задача е необходимо да се задават определени форми на обучение на всички неврони. В този контекст се връщаме към фиг. 2.1, a. където изходният неврон k е видим за външния свят, а желания изходен сигнал може да се носи непосредственото към този неврон.

2.8. Обучение с учител

Ще разгледаме парадигмата за обучение на невронната мрежа. Ще започнем с *парадигмата обучение с учител* (supervised learning). На фиг. 2.6 е показана блокдиаграма илюстрираща тази форма на обучение. Концептуално участието на учителя може да се разглежда като наличие на знания за околната среда представени във вида *вход-изход*. При това самата околнна среда е неизвестна за самата невронна мрежа. Предполагаме че на учителя и обучаемата мрежа се подава обучаващ вектор от околната среда. На основата на запомнетите по-рано знания учителя може да сформира и да придае на обучаваната невронна мрежа желаният отговор съответстващ на дадения входен вектор. Този желан резултат представлява оптимални действия които трябва да изпълнява невронната мрежа. Параметрите на мрежата се коригират с отчитане на обучаващият вектор и сигнала за грешки. *Сигнала за грешки* (error signal) е разликата между желания сигнал и текущия отговор на невронната мрежа. Корекцията на параметрите се извършва стъпка по стъпка с цел *имитация* (emulation) на мрежата от неврони на поведението на учителя. По този начин в процеса на обучение знанията на учителя се предават в мрежата максимално бързо. След обучението с учител, учителя може да се изключи и да позволи на невронната мрежа да работи със средата самостоятелно.



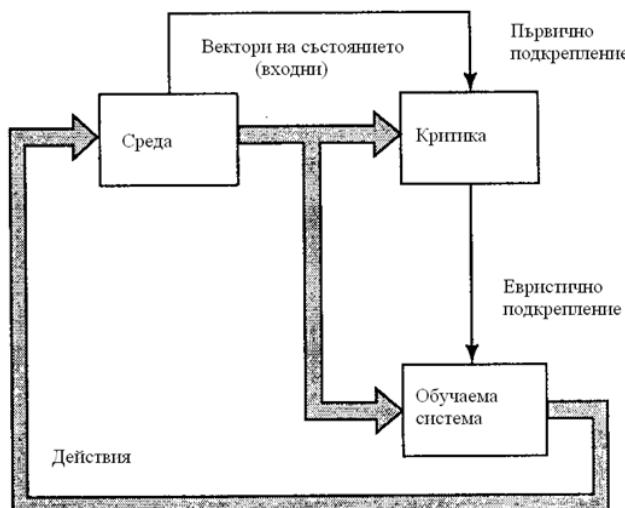
Фиг.2.6. Блок диаграмма на обучението с учител

Описаната форма на обучение е затворена система с обратна връзка, която на включва в себе си околната среда. Производителността на такава система може да се изясни в термините на средноквадратична грешка или сума от квадратите на грешките на обучаващата извадка, представена във вид на функция на свободните параметри на системата. За такава функция можем да посочим многомерна *повърхност за грешки* (error surface) в координатите на свободните параметри. При това реалната повърхност на грешките се *усреднява* (averaged) по всички възможни примери представени във вид на двойки „вход–изход“. Всяко конкретно действие на системата с учител е представено в една точка на повърхността на грешките. За повишаване на производителността на системата във времето стойността на грешките трябва да е известна към страната на максимума на повърхността на грешки. Този максимум може да бъде както локален така и глобален. Това може да се направи ако системата притежава полезна информация за градиента на повърхността на грешки, съответстващ

на текущото поведение на системата. Градиента на повърхността за грешки във всяка точка е вектор определящ най-бързото спускане на тази повърхност. В случая на обучението с учител по примери се изчислява *моментната оценка* (instantaneous estimate) на вектора на градиента, в който входния вектор се счита за функция на времето. При използването на резултатите на такава оценка преместването на точки по повърхността за грешки има вид на „случайно блуждаене“. Въпреки това при използването на съответният алгоритъм за минимизиране на функцията на стойността адекватният избор на обучаващ пример във формата „вход-изход“ е достатъчно време за обучение на системата за обучение с учител, което може да решава такива задачи като класификация на образи и апроксимация на функцията.

2.9 Обучение без учител

Обучението без учител (learning without a teacher) е алтернатива на парадигмата на обучението с учител. Самото название подчертава отсъствието на ръководител, контролиращ процеса на настройване на тегловите коефициенти. При използването на такъв подход не съществуват маркирани примери, по които се провежда обучението. В тази алтернативна парадигма можем да отделим два метода.



Фиг.2.7. Блок диаграма за обучение с подкрепление

Обучение с подкрепа, или невродинамично програмиране

За *обучението с подкрепа* (reinforcement learning) формирането на отразени входни сигнали в изходни се изпълнява в процеса на взаимодействие с външната среда, с цел минимизиране на скаларният индекс на производителността. На фиг. 2.7. е показано блокдиаграма на една от формите на системата за обучение с подкрепа включваща блок „критика“, който преобразува *първичния сигнал за подкрепа* (primary reinforcement signal), получен от външната среда в сигнал с по-високо качество наречен *евристичен сигнал за подкрепа* (heuristic reinforcement signal). Двата сигнала са скаларни.

Такава система предполага *обучение с отложена подкрепа* (delayed reinforcement). Това значи, че системата получава от външната среда последователност от сигнали на възбудждане (т.е. вектори на състоянията), които довеждат до генериране на евристичен сигнал на подкрепа. Целта на обучението е минимизация на *функцията на стойността* на прехода, определен като математическо очакване на комулативната

стойност на *действията*, предприети в разстояние от няколко стъпки, а не просто текущи стойности. Може да се окаже, че някой от предприетите по-рано в дадената последователност действия са били определящи за формирането на общото поведение на цялата система. Функцията на *обучаемата машина* (learning machine), съставлява вторият компонент на системата, определя тези действия и формира техния основен сигнал за обратна връзка насочен към външната среда.

Практически реализацията на обучението с отложена подкрепа се усложнява по две причини:

- Не съществува учител, формиращ желания отговор на всяка стъпка на обучението.
- Наличието на задръжка при формирането на първичния сигнал за подкрепа изиска решаването на *временна задача за присвояване на коефициентите на доверие* (temporal credit assignment). Това означава, че обучаваната машина е длъжна да присвоява коефициенти на доверие и недоверие на действията, изпълнени на всички стъпки, довеждащи до краен резултат, в същото време както първичния сигнал за подкрепа се оформя само на основата на крайният резултат.

Системата за обучение с отложена подкрепа е много привлекателна. Тя съставя базис от системи, взаимодействащи с външната среда, развиващи по този начин способността за самостоятелно решение на възникващите задачи на основата само на собствените резултати на взаимодействие със средата.

Обучението с подкрепа е тясно свързано с *динамичното програмиране* (dynamic programming) по методологията на Белман. Динамичното програмиране реализира математическият формализъм за последователно вземане на решения. Премествайки обучението с подкрепа в предметната област на динамичното програмиране можем да вземем всички резултати от последното.

Обучение без учител

Обучението без учител (unsupervised) (или *обучение на основата на самоорганизация* (selforganized)) се осъществява без намесата на външен учител или коректор контролиращ процеса на обучение (фиг.2.8). Съществува само *независима от задачата мярка за качеството* (task-independent) за представяне, на което е нужно да се научи невронната мрежа, и свободните параметри на мрежата се оптимизират по отношение на тази мярка.



Фиг.2.8. Блок диаграма на обучението без учител

След обучение на мрежата на статистически закономерности на входния сигнал, тя е способна да формира вътрешно представени кодирани признания на входните данни и по този начин автоматично да създава нови класове. За обучение без учител можем да използваме правилото за конкурентното обучение. Можем да използваме невронната мрежа, състояща се от два слоя входен и изходен. Входния слой получава достъпни данни. Изходният слой се състои от неврони, конкуриращи се за право на отговор на

признаците съдържащи се във входните данни. В простия случай невронната мрежа действа на принципа „победителя получава всичко”.

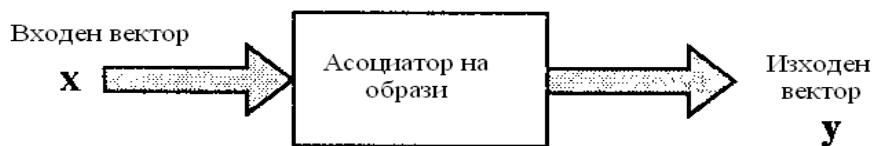
2.10. Задачи за обучението

Избора на конкретния алгоритъм за обучение зависи от задачите решението на които следва да обуви невронната мрежа. В този контекст могат да се отделят шест основни задачи, за решението на които се прилагат невронните мрежи.

Асоциативна памет

Асоциативната памет (associative memory) е разпределена памет, която се обучава на основата на осоциации, подобно на мозъка на живите същества. Асоциативността се счита за основа характеристика на човешкият мозък от времето на Аристотел. В следствие на това всички модели за знания в качеството на една от основните операции в тази или друга форма използват асоциативна памет.

Съществуват два типа асоциативни памети: *автоасоциативна* (autoassociation) и *хетероасоциативна* (heteroassociation). При решаването на задача от автоасоциативната памет в невронната мрежа се запомнят предадените й образи (вектори). След това в тази мрежа последователно се подават не пълни описания или зашумени представяния, съхранявани в паметта на изходния образ и се поставя задачата за разпознаване на конкретния образ. Хетероасоциативната памет се отличава от автоасоциативната по това че произведен избор на изходни образи съответства на друг произведен избор от изходни сигнали. За настройване на невронната мрежа за решаване на задачите за автоасоциативната памет се използва обучение без учител, а в хетероасоциативните памети обучение с учител.



Фиг.2.9. Диаграма „вход-изход” за мрежи на асоциативните образи

Нека \mathbf{x}_k е *ключов образ* (key pattern) (вектор) приеман за решаване на задачите на асоциативната памет, а \mathbf{y}_k - *запомнен образ* (memorized pattern) (вектор). Отношението за асоциация на образите може да се опише чрез:

$$\mathbf{x}_k \rightarrow \mathbf{y}_k, k = 1, 2, \dots, q, \quad (2.18)$$

Където q е количеството на съхранените в мрежата образи. Ключовият образ \mathbf{x}_k се явява в ролята на стимул, който не само определя местоположението в запомнения образ \mathbf{y}_k , но и съдържа ключа за извличането му.

В асоциативната памет $\mathbf{y}_k = \mathbf{x}_k$. Това значи, че пространствата на входните и изходните данни на мрежата са дължни да имат еднакви размери. В хетероасоциативните памети $\mathbf{y}_k \neq \mathbf{x}_k$. Това значи, че размерността на пространството на изходните вектори може да се отличава от размерността на пространството на входните вектори (но може и да съвпадат с нея).

В работата на асоциативната памет има 2 фази:

- *Фаза на запомняне* (storage phase), съответства на процеса на обучение на мрежата в съответствие с формула (2.18).

- *Фаза на възстановяване* (recall phase), съответства на извлечането на запомнения образ в отговор на представения в мрежата в зашумена или изкривена версия на ключа.

Нека стимула (входния сигнал) x представлява зашумена или изкривена версия на ключовия образ x_j . Този стимул поражда отговор (изходния сигнал) y (фиг.2.9). В идеалния случай $y = y_j$, където y_j е запомнен образ асоцииран с ключа x_j . Ако при $x = x_j$, изхода на мрежата е $y \neq y_j$, то асоциативната памет при възстановяване на образа е направила грешка.

Количеството q на образите се съхранява в асоциативната памет и се явява непосредствена мярка за *обема на паметта* (storage capacity) на мрежата. При построяване на асоциативната памет е желателно максимално да се увеличава нейния обем (информационния обем на асоциативната памет се измерва в процеса от общото количество неврони N , използвани за създаването на мрежата).

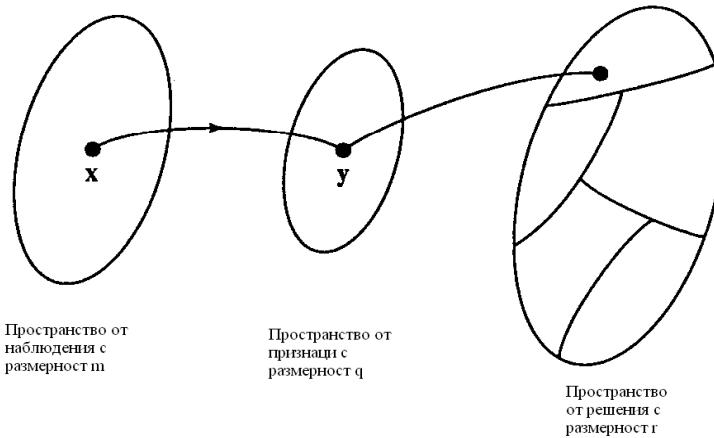
Разпознаване на образи

Разпознаването на образи формално се определя като процес, в който полученият образ/сигнал трябва да се отнесе към един от предопределените класове (категории). За да могат невронните мрежи да решават задачи за разпознаване на образи, отначало е необходимо тяхното обучение, подавайки последователно входни образи заедно с категориите към които тези образи принадлежат. След обучението на мрежите на входа се подава от по-рано отделен образ, който принадлежи към избраната по-рано категория, както и множество образи, използвани при обучението. Благодарение на информацията получена от даденото обучение, мрежата ще може да отнесе представения образ към конкретен клас. Разпознаването на образи се използва от невронните мрежи статистически, при това образа се представя от отделни точки в многомерно *пространство от решения*. Всяко пространство от решения се разделя на отделни области, всяка от които се асоциира с определен клас. Границата на тези области се формира в процеса на обучение. Пространството на тези граници се изпълнява статистически на основата на дисперсия, присъща на данните от дадения клас. Като цяло машините за разпознаване на образи, създадени на основата на невронните мрежи може да се разделят на два типа:

- Системата се състои от две части : мрежа за *извлечение на признания* (future extraction) (без учител) и мрежа за класификация (classification) (с учител) (фиг.2.10, а). Такъв метод съответства на традиционния подход към статистическото разпознаване на образи. В концептуалните термини обаразът се представя като избор от m наблюдения, всяко от които може да се разглежда като точка x в m -мерно *пространство от наблюдения* (данни) (observation (data) space). Извличането на признания се описва с помощта на преобразование, което протича в точка x , помеждутъчна точка y в q -мерно пространство от признания, където $q < m$ (Фиг.2.10, б). Това преобразование може да се разглежда като операция за намаляване на размерността (свиване на данните), опростяваща задачата за класификация. Самата класификация се описва като преобразование, което изобразява точка y в един от нейните класове от r -мерното пространство от решения (където r е количество разделени класове).



a)



б)

Фиг. 2.10. Илюстрация на класическият подход за разпознаване на образи

- Системата се проектира като единна многослойна мрежа за пряко разпространение. Използва се един от алгоритмите за обучение с учител. При този подход задачата за извличане на признаки се използва от изчислителните възли на скрития слой на мрежата.

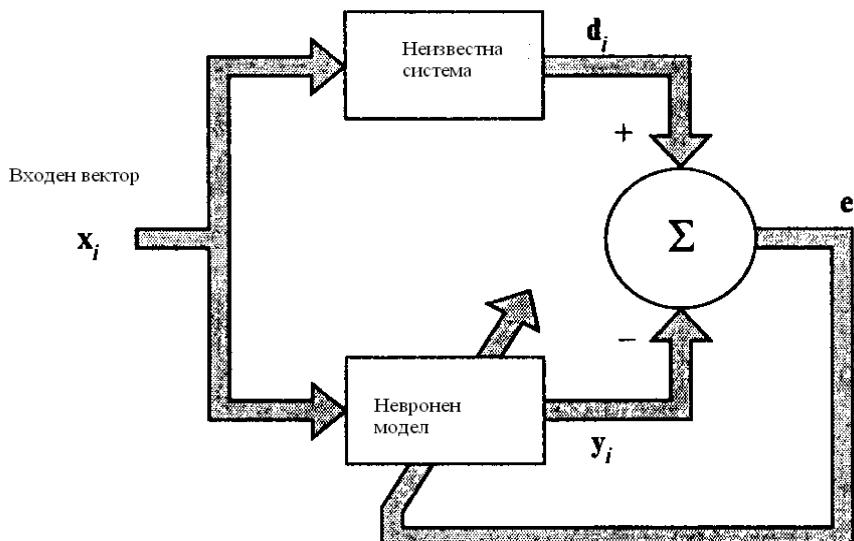
Аproxимация на функции

Третата задача за обучение е апроксимация на функции. Ще разгледаме нелинейно изображение от вида „вход-изход”, зададено чрез следната формула:

$$\mathbf{d} = \mathbf{f}(\mathbf{x}), \quad (2.19)$$

Където вектора \mathbf{x} – вход, а вектора \mathbf{d} – изход. Векторната функция $\mathbf{f}(.)$ се счита за неизвестна. За да попълним множеството от стойности на $\mathbf{f}(.)$, се предоставят множество от маркирани примери:

$$\mathcal{T} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^N. \quad (2.20)$$



Фиг.2.11. Блок-диаграма за решаване на задачите от идентификация на системата

Към структурата на невронните мрежи и апроксимацията на неизвестната функция $f(\cdot)$, се предявяват следните изисквания: функцията $F(\cdot)$, описваща отражението на входния сигнал в изходен трябва да бъде достатъчно близка до функцията $f(\cdot)$, в смисъла на Евклидовата норма на множеството на всички входни вектори x , т.e.

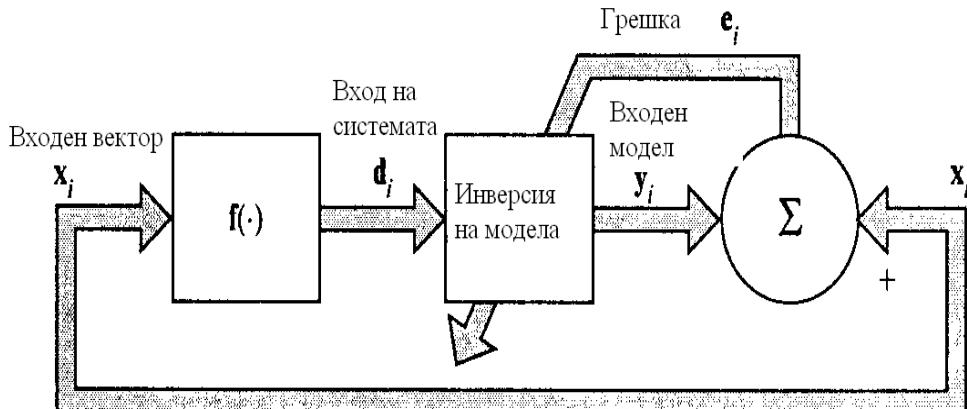
$$\|F(x) - f(x)\| < \epsilon \quad \text{За всеки вектор } x, \quad (2.21)$$

Където ϵ е някое малко положително число. Ако количеството N от елементи на обучаващото множество е достатъчно голямо и в мрежата има достатъчно свободни параметри, то грешките на апроксимацията ϵ може да станат достатъчно малки.

Описаната задача за апроксимация се явява отличен пример на задача за обучение с учител. Нека x_i играе ролята на входен вектор, а d_i - ролята на изходен вектор. Така задачата за обучение с учител се свежда към задача за апроксимация.

Способността на невронните мрежи да апроксимират неизвестни отражения на входното пространство в изходно може да се използва за решение на следните задачи:

- *Идентификация на системата* (system Identification). Нека формула (2.19) описва съотношението между входа и изхода в неизвестна система с няколко входа и изхода (MIMO-система) без памет. Терминът „без памет“ означава инвариантност на системата във времето. Тогава множеството от маркирани примери (2.20) може да се използва за обучение на невронни мрежи, представляващи модел за тази система. Нека y_i е изход на невронната мрежа, съответстващ на входния вектор x_i . Разликата между желания отговор d_i и изхода на мрежата y_i съставят вектора на сигнала за грешки e_i (фиг.2.11), използван за корекция на свободните параметри на мрежата с цел минимизация на средноквадратичните грешки – сумата от квадратите на разстоянията между изходите на неизвестните системи и невронните мрежи в статистически смисъл.



Фиг.2.12. Блок диаграма за моделиране на инверсии

- *Инверсия на системата* (inverse system). Предполагаме, че съществува някоя система ММО без памет, за която преобразуването на входните пространства в изходни се описва чрез формула (2.19). Трябва да построим инверсна система, която в отговор на вектор \mathbf{d} генерира вектор \mathbf{x} . Инверсната система може да бъде описана по следния начин:

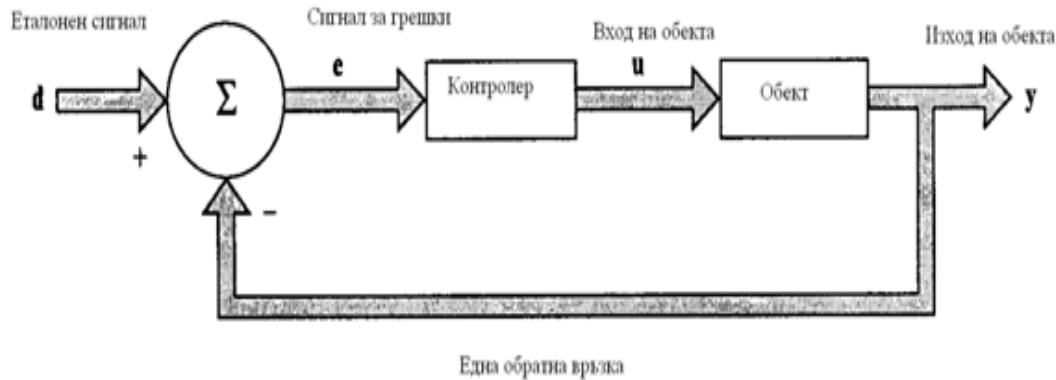
$$\mathbf{x} = \mathbf{f}^{-1}(\mathbf{d}), \quad (2.22)$$

Където вектор-функцията $\mathbf{f}^{-1}(\cdot)$ се явява инверсия на функцията $\mathbf{f}(\cdot)$. Функцията $\mathbf{f}^{-1}(\cdot)$ не се явява обратна на функцията $\mathbf{f}(\cdot)$. Индексът -1 се използва като индикатор на инверсията. В много ситуации на практика функцията $\mathbf{f}(\cdot)$ може да бъде достатъчно сложна, което прави практически невъзможно формалното извеждане на обратната функция $\mathbf{f}^{-1}(\cdot)$. На основата на множество маркирани примери (2.20) може да се построи нронна мрежа, апроксимираща обратната функция $\mathbf{f}^{-1}(\cdot)$, с помоща на схемата описана на фиг.2.12. В дадената ситуация ролите на векторите \mathbf{x}_i и \mathbf{d}_i се разменят: вектора \mathbf{d}_i се използва като входен сигнал, а вектора \mathbf{x}_i - като желан отговор. Нека векторът за сигнала на грешки се определя от разликата между вектора \mathbf{x}_i и изхода на нронната мрежа \mathbf{y}_i , получен в отговор на \mathbf{d}_i . Както и в задачите за идентификация на системата, векторите на сигналите за грешки се използват за корекция на свободните параметри на нронната мрежа с цел минимизация на сумата от квадратите като разлика между изходите на неизвестната инверсна система и нронните мрежи в статистически смисъл.

Управление

Управлението на *предприятие* (plant) е още една задача за обучение, което може да се реши с използването на нронните мрежи. Под термина „предприятие“ се разбира процес или критична част от системата, подлежаща на управление. Контекста на задачата за управление на мозъка се явява живо доказателство за създаване на обобщена система за управление, използваща предимствата на успоредното разпределение на изчисленията и едновременно управляваща хиляди функционални механизми. Такава система се явява нелинейна, може да обработва шум и да оптимизира своята работа в дългосрочен план.

Ще разгледаме *системата за управление с обратна връзка* (feedback control system), показана на фиг.2.13. В този случай системата използва единствена обратна връзка, обхващаща целия обект за управление (т.е. изхода на обекта е свързан с неговия изход). Изхода на обекта на управление y се изчислява от *еталонния сигнал* (reference signal) d , приет от външния източник. Така полученият сигнал за грешки e се обработва от *невроконтролер* (neurocontroller) за настройка на свободните параметри. Основната задача на контролера е да поддържа такъв входен вектор за обекта, за който изходният сигнал y съответства на еталонната стойност d . В задачата за контролера влизат инвертиране на образния вход – изход обект за управление.



Заместваме във фиг. 2.13. сигнала за грешки e разпространен чрез невроконтролер, преди да достигне обекта на управление. Следователно за настройката на свободния параметър на обекта за управление в съответствие с алгоритъма на обучение основан на корекция на грешките е необходимо използването на матрицата на Якоби:

$$\mathbf{J} = \left\{ \frac{\partial y_k}{\partial u_j} \right\}, \quad (2.23)$$

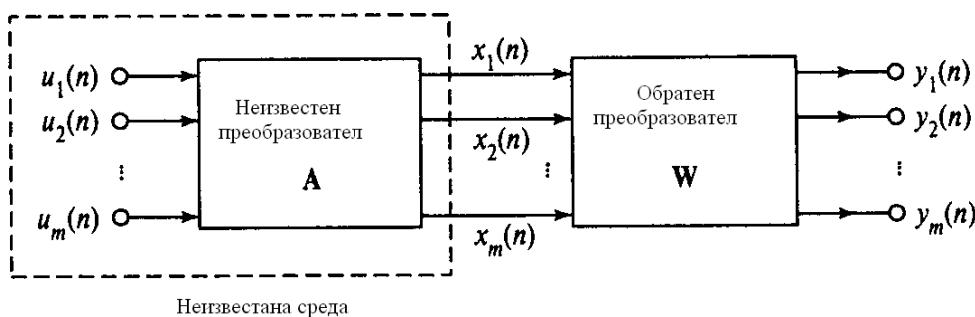
Където y_k елемент на входния сигнал y на обекта за управление; u_j елемент на вектора на входния обект u . Частните производни $\partial y_k / \partial u_j$ за различни k и j зависят от работата на точките на обекта за управление, и следователно са неизвестни. За техните оценки се използват два подхода:

- *Непряко обучение* (indirect learning). С използването на текущите измервания на входа и изхода на обекта за управление се построява модел на невронната мрежа, възпроизвеждащ тази зависимост. Той се използва за оценка на матрицата на Якоби \mathbf{J} . Частните производни съответстващи на тази матрица се използват в алгоритъма за обучение основан на корекция на грешките за настройване на свободните параметри на невроконтролера.
- *Пряко обучение* (direct learning). Значите на частните производни $\partial y_k / \partial u_j$ в общият случай са известни и остават постоянни в някой динамични диапазони на стойности за обекта на управление. Следователно частните производни могат да се апроксимират по значите им. Абсолютните стойности на частните производни се задават от разпределение представено в свободните параметри на невроконтролера.

Филтрация

Под терминът *филтър* (filter) се разбира устройство или алгоритъм за извличане на полезна информация от зашумени данни. Шума може да визникне по много причини. Например данните могат да бъдат измерени с грешки, или смущенията могат да възникват при предаване на информационния сигнал през зашумена линия на въръщата. На полезните сигнали може да бъде наложен друг сигнал постъпващ от околната среда. Филтрите могат да се прилагат за решение на три основни задачи за обработка на информация:

1. *Филтрация* (filtering), т.е. извличане на полезна информация в дискретен момент от време n измерени до момента на време n включително.
2. *Изглаждане* (smoothing) – тази задача се отличава от филтрацията по това че информацията за полезните сигнали в момент от време n не трябва, за това за извличане на тази информация могат да се използват данни получени по-късно. В изглаждането при формиране на резултата присъства *закъснение* (delay). Така както при изглаждането можем да използваме данни получени не само до момента от време n , но и след него, то този процес в статистически смисъл е по-точен от филтрацията.
3. *Прогнозиране* (prediction) – целта на процеса е получаване на прогноза за относителното състояние за обекта на управление в някой момент от време $n + n_0$, където $n_0 > 0$ на основата на данни получени до момента n (включително).



Фиг. 2.14. Блок диаграма на сляпо разделяне на източници

Задачите за филтрация се явяват *задачи за разпознаване на гласа на събеседника на шумна вечеринка* (cocktail party problem). Ние притежаваме способността да се концентрираме на гласа на събеседника независимо от разногласния шум донасящ се до групата посетители на вечеринката, думи които ние не разпознаваме. Вероятност за решението на такава задача се използва някаква форма на предсъзнателен и превантивен анализ. В контекста (изкуствени) на невронните мрежи аналогична задача за филтрация възниква при *сяло разпределение на сигналите* (blind signal separation). За да се сформира задачата за сляпо разпределение на сигналите се представя неизвестен източник като неизвестен сигнал $\{s_i(n)\}_{i=1}^m$. Тези сигнали линейно се смесват с неизвестни сензори и се получава вектор за наблюдение с размерност $m \times 1$ (фиг. 2.14).

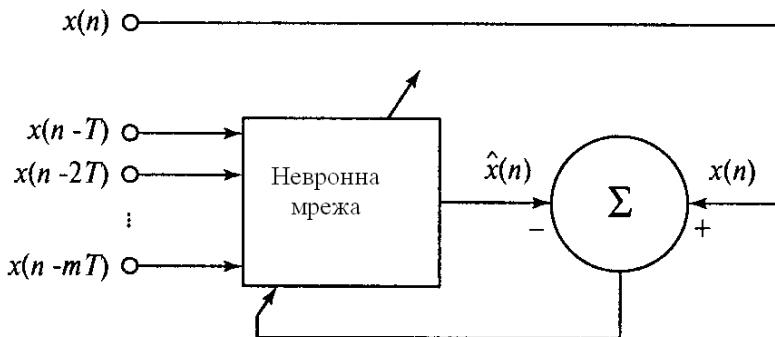
$$\mathbf{x}(n) = \mathbf{A}\mathbf{u}(n), \quad (2.24)$$

Където

$$\mathbf{u}(n) = [u_1(n), u_2(n), \dots, u_m(n)]^T, \quad (2.25)$$

$$\mathbf{x}(n) = [x_1(n), x_2(n), \dots, x_m(n)]^T, \quad (2.26)$$

Където A е неизвестна несингулярна матрица на смесване на размености $m \times m$. Векторът на наблюдение $\mathbf{x}(n)$ е известен. Трябва да се възстанови изходния сигнал $u_1(n), u_2(n), \dots, u_m(n)$ по метода за обучение с учител.



Фиг 2.15. Блок диаграмма на нелинейното прогнозиране

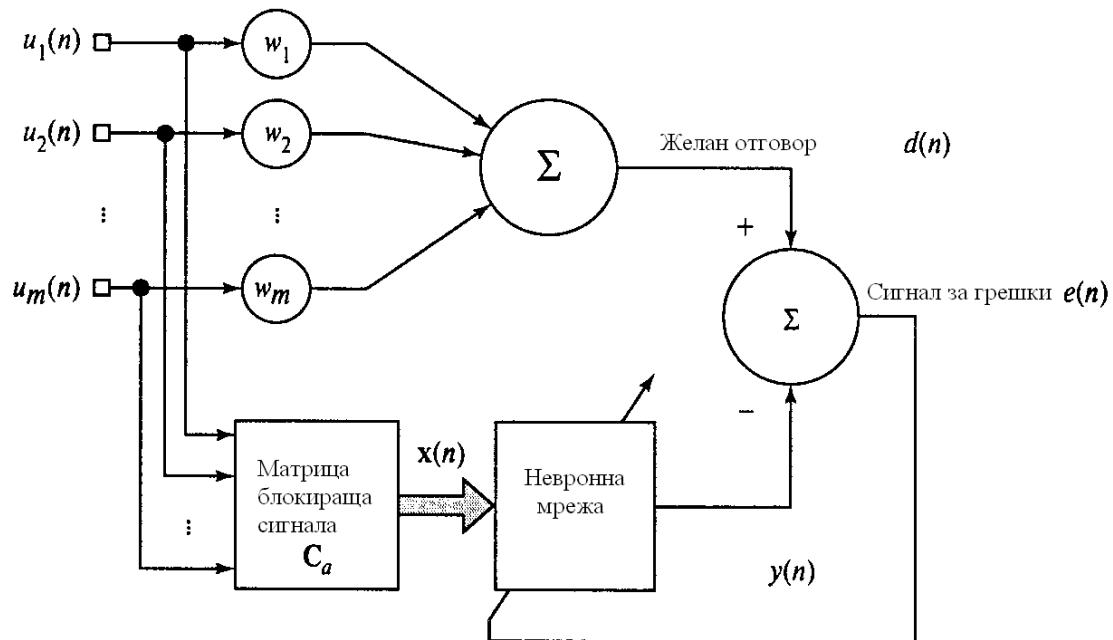
Връщаме се към задачата за прогнозиране в която се изисква да се предскаже текущата стойност на процеса $x(n)$ по предишна информация за него взета в дискретен момент от време и представена чрез стойностите $x(n - T), x(n - 2T), \dots, x(n - mT)$, където T е периодично снемане на сигнала, а m е степен на прогнозиране. Задачата може да се реши с помощта на обучение основано на грешки без учител, тъй като обучаващите примери се получават от самия процес (фиг. 2.15.). При това $x(n)$ е в ролята на желан отговор. Обозначаваме с $\hat{x}(n)$ резултатът от прогнозата на една стъпка напред генерирана от невронната мрежа в момент от време n . Сигналът за грешки се определя като разлика между $x(n)$ и $\hat{x}(n)$. Той се използва за настройка на свободните параметри на мрежата. При тези допускания прогнозите може да се разглеждат като форма за *построяване на модел* (model building) в този смисъл, че грешките на прогнозиране в статистически смисъл на невронните мрежи могат да работят в качеството на модели имитиращи физическия процес, обезпечаващ генерирането на данните. Този процес се явява *нелинеен*, невронните мрежи се явяват мощно средство за решаване на задачи за прогнозиране, така както нелинейната обработка на сигнала предполага само архитектурата на невронните мрежи. Единствено изключение е изходния слой неврони: ако динамичният диапазон на времето $\{x(n)\}$ е неизвестен, във изходния слой е целесъобразно да се използва линейна обработка на елементи.

Формиране на диаграми на направление

Това е *пространствен* (spatial) случай на филтрация и се използва за разделяне на пространствени признаки на полезен сигнал и шум. За формирането на диаграми на направление се използва *специално устройство*.

Задачите за формиране на диаграми на направление се решават с използването на невронни мрежи, създадени на основата на знания на слуховия апарат на човека. И решаване на задачите за ехо-локация в коровите слоеве на слуховата система на прилепите. Системата за ехо-локация на прилепите отфильтрира влиянието на околната среда. Прилепа използва кратки ултразвукови сигнали с честотна модулация, а след това улавя отразените сигнали с помощта на слуховия апарат, фокусирайки вниманието върху уловената плячка (насекоми). Ушите на прилепа изпълняват някакъв вид пространствена филтрация, резултатите от която се използват от слуховата система за

извличане на сигнала, съдържащ интересуващият клас от обекти (attentional selectivity).



Фиг.2.16. Блок диаграмма обобщаяща системата

Построяването на диаграмите на направление се използва в радарните системи. Задачата се свежда към отслабване на тректорията на целта в условията на шум и интерференции. Задачата се усложнява от следните два фактора:

- Полезния сигнал се формира в неизвестно направление.
 - Не съществува априорна информация от наложените сигнали в резултат на интерференция.

Блок диаграмата показана на фиг.2.16. решава тази задача. Тази система сортира следващите компоненти:

- *Масив от антитела на елементите* (array of antenna elements), който сваля снигла от дискретни точки на пространството;
 - *Линеен суматор* (linear combiner), определя множество от фиксирани тегла $\{w_i\}_{i=1}^m$ на изхода на които се формира желания отговор. Работата на този линеен суматор е подобна на пространствения филтър, характеризиращ се с диаграми на изльчване (графика на амплитудата на изходния сигнал на антената в зависимост от ъгъла на постъпване на сигнала в полярни координати).
 - *Матрица на блокиране на сигнала* (signal-blocking matrix) \mathbf{C}_a , функционира, когато се появява никаква блокировка влияеща върху сортирането на сигналите, като чрез диаграмата изльчения пространствен филтър, представлява линейния суматор.
 - *Невронни мрежи* с нарастването на параметрите, предназначени за адаптация към статистическите вариации влияе на сигнала.

Настройката на свободните параметри на невронните мрежи се извършва с помощта на алгоритъм за обучение на основата на корекция на грешките, определени като разликата между изходния линеен суматор $d(n)$ и фактическите изходни сигнали $y(n)$ на невронните мрежи. Този образ обобщава системата като GSLC (generalized sidelobe canceller), която работи под управлението на линейния суматор, играещ ролята на учител. Както и при обучението с учител линейния суматор се намира в контура на

обратна връзка на невронните мрежи. Този клас обучаеми системи се наричат още неврокомпютри, предназначени за настройка на внимания.

Във фундаментален смисъл всички тези шест задачи за обучение се явяват проблеми за обучение построени на основата на примери. При отсъствието на априорни знания се явяват лошо обусловими (*ill posed*), т.е. възможните решения се определят нееднозначно. Един от способите обезпечава *правилно обусловени* (*well pose*) решения в примерните теории за регуляризация.

2.11. Памети

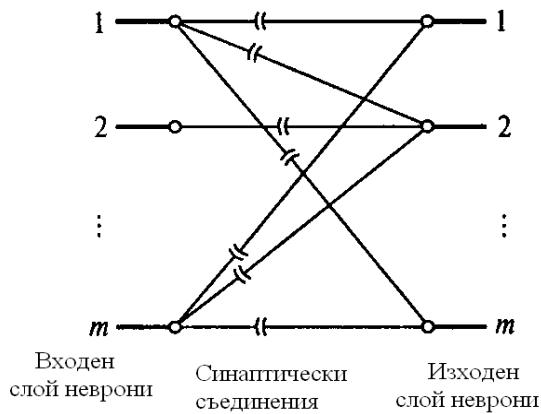
Обсъждането на задачите за обучение, особено задачите за асоциативната памет, съставят задачите на самата *памет* (*memory*). В контекста на невробиологията под памет се разбира относителното продължение на временната деформация на структурата на неврона, оказваща влияние на организма. Без тези деформации паметта не съществува. В паметта предварително трябва да се съберат определени модели на поведение. Това се осъществява чрез *процеса на обучение* (*learning process*). Паметта и обучението са тясно взаимосвързани. При изучаването на някой образ той се съхранява в структурата на мозъка, откъдето може да бъде извлечен в случай на необходимост. Формално паметта може да се раздели на кратковременна и дълговременна в зависимост от срока на възможно съхранение на информацията. *Кратковременната памет* отразява текущото състояние на околната среда. Всяко „ново“ състояние на средата, различно от образа, съдържащ се в краткотрайната памет, води до обновяване на данните в паметта. В *дълготрайната памет* се съхраняват знания за продължително (постоянно) използване.

Асоциативната памет се характеризира със следните особености:

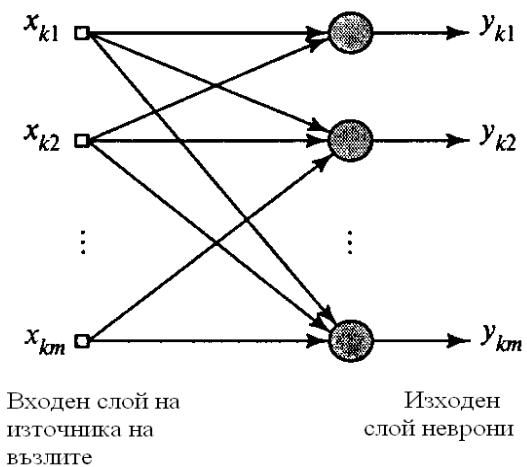
- Тази памет е разпределена;
- Стимулите (ключове) и отговорите (съхранени образи) в асоциативната памет представляват сбор на вектори от данни;
- Информацията се запомня с помощта на формиране на пространство от образи на невромрежовата активност на голямо количество неврони;
- Информацията, съдържаща се в стимулите, определя не само мястото, което тя заема, но и адреса за извлечане;
- Не винаги невроните са най-надеждни за изчисляване на елементи и работа в условия на шум, паметта има висока устойчивост към смущения и изкривяване на данни;
- Между отделните образи съхранявани в паметта, могат да съществуват вътрешни взаимовръзки. (В противен случай размера на паметта трябва да е невероятно голям, че да събере всички отделни изолирани образи.) Така има вероятност от грешки при извлечане на информация от паметта.

В *разпределената памет* (*distributed memory*) главни интерес представлява едновременно или почти паралелно функциониране на множество различни неврони при обработката на вътрешни или външни стимули. Невронната активност формира в паметта пространство от образи, съдържащи информация за стимулите. Така паметта изпълнява разпределително отразяване на образи в пространството на входните сигнали, а другите образи в изходното пространство. Някои важни свойства за отразяване на разпределителната памет могат да се видят на примера на идеализирана невронна мрежа, състояща се от два слоя неврони. На фиг. 2.17, a е показана мрежа, която може да се разглежда като *модулен компонент на невронната система* (*model component of nervous system*). Всички неврони от входния слой са съединени с всички неврони на изходния слой. В реалните системи синаптическите връзки между

невроните са сложни. В модела, показан на фиг. 2.17, *a* е представен общият резултат от всички синаптически контакти между дендритните неврони на входния слой и отговарящите аксони на невроните на изходния слой биха се използвали като идеално съединение. Нивото на активност на невроните на входния слой може да укаже влияние на степента на активност на невроните от изходния слой.



а) Компонентен модел на асоциативната памет на невронна система

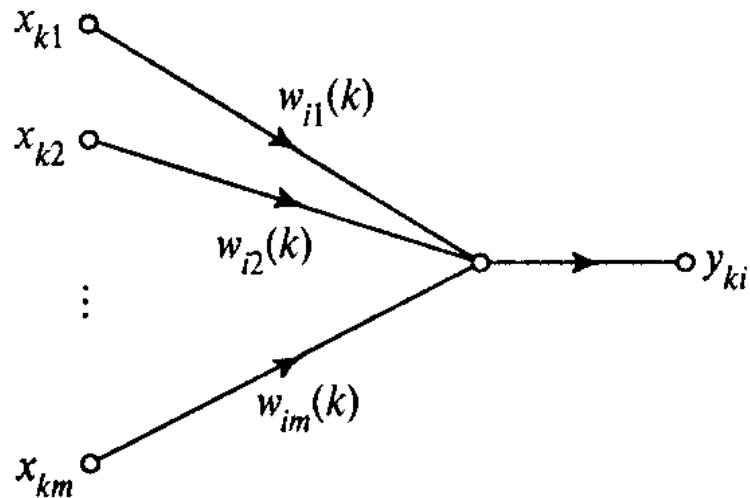


б) Модел на асоциативната памет с изкуствени неврони

Фиг.2.17. Модели на асоциативната памет

Аналогична е ситуацията за изкуствените невронни мрежи, показани на фиг. 2.17, *b*. В дадения случай възлите на източника на входния слой и невроните на изходния слой работят като изчислителни елементи. Синаптическите тегла се интегрират в невроните на изходния слой. Връзките между двата слоя на мрежата представляват сбор от съединения.

Изображенията на фиг.2.17, се считат за *линейни*. В резултат на това се предполага, че всички неврони се представят като линейни суматори на граф за предаване на сигнали. (фиг.2.18). За анализа предполагаме, че във входния слой се предава образа \mathbf{x}_k , а в изходния слой възниква образа \mathbf{y}_k .



Фиг.2.18. Граф за предаване на сигнала на i -я неврон

Ще разгледаме въпроса за обучение основано на асоциациите между образите \mathbf{x}_k и \mathbf{y}_k . Образите \mathbf{x}_k и \mathbf{y}_k са представени като вектори, разгърната форма на които има следния вид:

$$\mathbf{x}_k(n) = [x_{k1}(n), x_{k2}(n), \dots, x_{km}(n)]^T,$$

$$\mathbf{y}_k(n) = [y_{k1}(n), y_{k2}(n), \dots, y_{km}(n)]^T.$$

За удобство предполагаме, че размерността на входните и изходните сигнали съвпада и е равна на m , т.е. размерностите на векторите \mathbf{x}_k и \mathbf{y}_k са еднакви. Нека m се нарича *размерност на мрежата* (network dimensionality) или просто *размерност*. И нека m е равно на количеството възли на източника на входния слой и числото на изчисляване на неврона на изходния слой. В реалните невронни мрежи размерността m може да бъде достатъчно голяма.

Елементите на векторите \mathbf{x}_k и \mathbf{y}_k могат да приемат както положителни, така и отрицателни стойности. В изкуствените невронни мрежи тези допускания се явяват естествени. Такава ситуация е характерна за невронна система, ако разглеждаме пороменлива равна на разликата между фактическото ниво на активност (т.е. степента на възбуждане на неврона) и произволно ненулево ниво на активност.

Отчитайки линейната мрежа, показана на фиг. 2.17, асоциира между ключовия вектор \mathbf{x}_k и запомнения вектор \mathbf{y}_k може да представи в матричен вид:

$$\mathbf{y}_k = \mathbf{W}(k)\mathbf{x}_k, k=1, 2, \dots, q, \quad (2.27)$$

Където $\mathbf{W}(k)$ - матрица на теглата, определяща двойката „вход-изход” ($\mathbf{x}_k, \mathbf{y}_k$). За да опишем матрицата на тегловите коефициенти $\mathbf{W}(k)$, използваме фиг.2.18, на която е представена схема на i -я неврон на изходния слой. Изхода y_{ki} на този неврон се изчислява като претеглена сума на елементите на ключовия образ \mathbf{x}_k по следващата формула:

$$y_{ki} = \sum_{j=1}^m w_{ij}(k) x_{kj}, \quad i = 1, 2, \dots, m, \quad (2.28)$$

Където $w_{ij}(k), j = 1, 2, \dots, m$ - синаптически тегла на неврона i , съответстващ на k -та двойка на асоциативните образи. Използваме матрично представяне на елементите y_{ik} , която може да се запише в следния еквивалентен вид:

$$y_{ki} = [w_{i1}(k), w_{i2}(k), \dots, w_{im}(k)] \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix}, \quad i = 1, 2, \dots, m. \quad (2.29)$$

Векторът-стълб в дясната част на равенството (2.29) представлява ключов вектор \mathbf{x}_k . Заместваме израза (2.29) вектора \mathbf{y}_k с размерности $m \times 1$, получаваме:

$$\begin{bmatrix} y_{k1} \\ y_{k2} \\ \vdots \\ y_{km} \end{bmatrix} = \begin{bmatrix} w_{11}(k) & w_{12}(k) & \dots & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & \dots & w_{2m}(k) \\ \dots & \dots & \ddots & \dots \\ w_{m1}(k) & w_{m2}(k) & \dots & w_{mm}(k) \end{bmatrix} \begin{bmatrix} x_{k1} \\ x_{k2} \\ \vdots \\ x_{km} \end{bmatrix}. \quad (2.30)$$

Съотношението (2.30) описва матричното преобразование (2.27) в разгърнат вид. В частност матрицата $\mathbf{W}(k)$ с размерност $m \times m$ се определя от:

$$\mathbf{W}(k) = \begin{bmatrix} w_{11}(k) & w_{12}(k) & \dots & w_{1m}(k) \\ w_{21}(k) & w_{22}(k) & \dots & w_{2m}(k) \\ \dots & \dots & \ddots & \dots \\ w_{m1}(k) & w_{m2}(k) & \dots & w_{mm}(k) \end{bmatrix}. \quad (2.31)$$

Отделните представления на q двойки на асоциативните образи $\mathbf{x}_k \rightarrow \mathbf{y}_k, k = 1, 2, \dots, q$ формират стойностите на елементите на отделните матрици $\mathbf{W}(1), \mathbf{W}(2), \dots, \mathbf{W}(q)$. Отчитайки факта, че тази асоциация на образите се представя от матрицата на теглата $\mathbf{W}(k)$, *матрицата на паметта* (memory matrix) с размерност $m \times m$ може да се определи като сума от матрици на тегловите кофициенти на цялата асоциация:

$$\mathbf{M} = \sum_{k=1}^q \mathbf{W}(k). \quad (2.32)$$

Матрицата \mathbf{M} определя връзките между входните и изходните слоеве на асоциативните памети. Тя представлява *опит* (experience), натрупан в резултат на подаване на q образа, представени във вида на двойките „вход-изход“. С други думи в \mathbf{M} се съдържат данни за всички двойки „вход-изход“, представени за запис в паметта.

Описаната матрица на паметта (2.32) може да бъде записана в рекурсивна форма:

$$\mathbf{M}_k = \mathbf{M}_{k-1} + \mathbf{W}(k), \quad k = 1, 2, \dots, q, \quad (2.33)$$

Където \mathbf{M}_0 - изходната матрица се явява нулева (т.е. всички синаптически тегла напаметта са равни на нула), а окончателната матрица \mathbf{M}_q съвпада с матрицата \mathbf{M} , определена от израза (2.32). В съответствие с рекурсивната формула (2.33) под обозначението \mathbf{M}_{k-1} се разбира матрица, получена на $k-1$ стъпка на асоциация, а под \mathbf{M}_k се разбира обновената матрица, получена в резултат на добавяне на $\mathbf{W}(k)$, получена на основата на k -те двойки вектори.

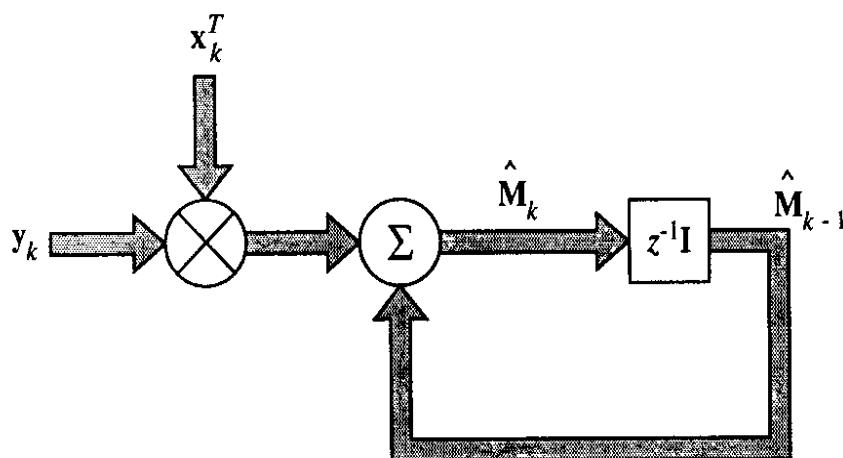
Памет във вид на матрични корелации

Предполагаме, че асоциативната памет (фиг 2.17, б) е била обучена на двойки вектори на входни и изходни сигнали $\mathbf{x}_k \rightarrow \mathbf{y}_k$, в резултат на което се изчислява матрицата на паметта \mathbf{M} . За оценка на матрицата \mathbf{M} въвеждаме обозначението $\hat{\mathbf{M}}$, което в термините за запаметените образи се описва чрез израза:

$$\hat{\mathbf{M}} = \sum_{k=1}^q \mathbf{y}_k \mathbf{x}_k^T. \quad (2.34)$$

Под обозначението $\mathbf{y}_k \mathbf{x}_k^T$ се разбира *външно (матрично) произведение* (outer product) на ключа \mathbf{x}_k и запаметения образ \mathbf{y}_k . Това произведение се явява оценка на матрицата на теглата $\mathbf{W}(k)$, която отразява изходния вектор \mathbf{y}_k , от входния вектор \mathbf{x}_k . Така както векторите \mathbf{x}_k и \mathbf{y}_k имат еднакви размерности, равни на $m \times 1$, матрицата за оценки $\hat{\mathbf{M}}$ ще има размерност $m \times m$. Това се съгласува с размерността на \mathbf{M} , определена от израза (2.32). Сумата в определянето на оценките на $\hat{\mathbf{M}}$ има пряко отношение към матрицата на паметта, определена от отношението (2.32).

Елементите от външното произведение $\mathbf{y}_k \mathbf{x}_k^T$ се обозначават с $y_{ki} x_{kj}$, където x_{kj} е изходен сигнал на възела j на входния слой, а y_{ik} – стойността на неврона на i -я изходен слой. В контекста на синаптическите тегла $w_{ij}(k)$ за k -та асоциация на възела на източника j встъпва в ролята на предсинаптически възел, а неврона i – в качеството на постсинаптически възел. Така „локалните“ процеси на обучение, описани от израза (2.34) може да се разглеждат като *обобщение на постулата на Хеб* (generalization of Hebb's postulate product rule). Нарича се още *правило на външно (матрично) произведение* (outer product rule), използвано за построяване на оценките на $\hat{\mathbf{M}}$. Построяването на такива образи на матрицата на паметта $\hat{\mathbf{M}}$ се нарича *памет във вид на матрични корелации* (correlation matrix memory). Корелация в тази или някоя друга форма, се явява основа на процеса на обучение, разпознаване на асоциации и образи, а също и извлечане на данни от паметта в невронната система на човека.



Фиг.2.19. Представяне на израза (2.38) във вид на граф за предаване на сигнали

Изразът (2.34) може да се запише в следната еквивалентна форма:

$$\hat{\mathbf{M}} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q] \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_q^T \end{bmatrix} = \mathbf{YX}^T, \quad (2.35)$$

Където

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_q], \quad (2.36)$$

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q]. \quad (2.37)$$

Матрицата \mathbf{X} има размерност $m \times q$ и се състои от всички множества от ключови образи, използвани в процеса на обучение. Тя получава названието *матрица на ключовете* (key matrix). Матрицата \mathbf{Y} с размерност $m \times q$ се състои от множеството от запаметени образи. Тя се нарича *матрица на запомняне* (memorized matrix).

Изразът (2.35) може да се запише в следната рекурсивна форма:

$$\hat{\mathbf{M}}_k = \hat{\mathbf{M}}_{k-1} + \mathbf{y}_k \mathbf{x}_k^T, \quad k = 1, 2, \dots, q. \quad (2.38)$$

На фиг. 2.19 е показан граф за предаване на сигнали. Съгласно него и рекурсивната формула (2.38), матрицата $\hat{\mathbf{M}}_{k-1}$ представляваща сбор от оценките на матрицата на паметта, а матрицата $\hat{\mathbf{M}}_k$ - обновената матрица с новите асоциации на образите \mathbf{x}_k и \mathbf{y}_k . Сравнявайки рекурсивните формули (2.33) и (2.38) и чрез несложно заместване, се получава, че външното произведение $\mathbf{y}_k \mathbf{x}_k^T$ представлява сбор от оценките на матрицата от теглата $\mathbf{W}(k)$, съответстващи на k -те асоциативни ключове и запаметените образи \mathbf{x}_k и \mathbf{y}_k .

Извличане от паметта

Фундаменталните задачи, възникващи при използване на асоциативни памети са адресация и извлечане на запаметени образи. Нека на входа на системата на асоциативната памет се подава случаен вектор на възбуждане \mathbf{x}_j , за който трябва да се получи вектора на *отговора* (response):

$$\mathbf{y} = \hat{\mathbf{M}} \mathbf{x}_j. \quad (2.39)$$

От изразите (2.34) и (2.39) получаваме:

$$\mathbf{y} = \sum_{k=1}^m \mathbf{y}_k \mathbf{x}_k^T \mathbf{x}_j = \sum_{k=1}^m (\mathbf{x}_k^T \mathbf{x}_j) \mathbf{y}_k, \quad (2.40)$$

Където $\mathbf{x}_k^T \mathbf{x}_j$ е *скалярно произведение* на \mathbf{x}_k и \mathbf{x}_j . Изразът (2.40) може да се запише във вида:

$$\mathbf{y} = (\mathbf{x}_j^T \mathbf{x}_j) \mathbf{y}_j + \sum_{k=1, k \neq j}^m (\mathbf{x}_k^T \mathbf{x}_j) \mathbf{y}_k. \quad (2.41)$$

Нека ключовите образи са нормализирани, т.е. имат единична дължина (или енергия):

$$E_k = \sum_{l=1}^m x_{kl}^2 = \mathbf{x}_k^T \mathbf{x}_k = 1, \quad k = 1, 2, \dots, q. \quad (2.42)$$

Тогава отговора на паметта на възбуждане (ключовия образ) \mathbf{x}_j може да се опрости:

$$\mathbf{y} = \mathbf{y}_j + \mathbf{v}_j, \quad (2.43)$$

Където

$$\mathbf{v}_j = \sum_{k=1, k \neq j}^m (\mathbf{x}_k^T \mathbf{x}_j) \mathbf{y}_k. \quad (2.44)$$

Първото слагаемо в дясната част на израза (2.34) представлява очаквания отговор \mathbf{y}_j . Така може да се трактова като „сигнал”, съставен от фактическия отговор \mathbf{y} . Вторият вектор \mathbf{v}_j в дясната част на израза представлява шум, който възниква в резултат на смесване на ключовия вектор \mathbf{x}_j с останалите вектори, намиращи се в паметта. Именно векторът на шума \mathbf{v}_j носи отговорност за грешките при извлечане от паметта.

В контекста на линейното пространство на сигналите косинуса на ъгъла между векторите \mathbf{x}_k и \mathbf{x}_j може да се определи като скаларно произведение между вектори, разделено на произведението на Евклидовата норма:

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = \frac{\mathbf{x}_k^T \mathbf{x}_j}{\|\mathbf{x}_k\| \|\mathbf{x}_j\|}. \quad (2.45)$$

$\|\mathbf{x}_k\|$ означава Евклидова норма на вектора \mathbf{x}_k , определен като квадратен корен от енергии:

$$\|\mathbf{x}_k\| = (\mathbf{x}_k^T \mathbf{x}_k)^{1/2} = E_k^{1/2}. \quad (2.46)$$

Изразът (2.45) може да се упрости:

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = \mathbf{x}_k^T \mathbf{x}_j. \quad (2.47)$$

Векторът на шума може да се запише по следния начин:

$$\mathbf{v}_j = \sum_{k=1, k \neq j}^m \cos(\mathbf{x}_k, \mathbf{x}_j) \mathbf{y}_k. \quad (2.48)$$

Тези ключови вектори са *ортогонални* (orthogonal) (т.е. перпендикулярни един на друг в Евклидов смисъл), то

$$\cos(\mathbf{x}_k, \mathbf{x}_j) = 0, \quad k \neq j, \quad (2.49)$$

Следователно векторът на шума е нулев. В този случай вектора на отговора \mathbf{y} е равен на \mathbf{y}_j . Такава памет се счита за *съвършено асоциирана* (associated perfectly), ако ключовите вектори се избират от *ортогонално множество* (orthogonal set), т.e. удоволстворяват следното условие:

$$\mathbf{x}_k^T \mathbf{x}_j = \begin{cases} 1, & k = j, \\ 0, & k \neq j. \end{cases} \quad (2.50)$$

Предполагаме, че всички ключови вектори изпълняват условие (2.50). Как се определя *обемът или запаметващата способност* (storage capacity) на асоциативната памет? С други думи, колко максимално количество образи може да се съхранят в нея? Този фундаментален въпрос е свързан с ранга на матрицата на паметта $\hat{\mathbf{M}}$. Рангът на матрицата се определя от количеството независими стълбове. Това значи, че ако всеки ранг на матрицата с размерност $l \times m$ е равен на r , то се изпълнява съотношението $r \leq \min(l, m)$.

В случая корелационната памет може да бъде представена от матрица на паметта съставена от $m \times m$, където m е размерността на пространството от входните сигнали. Рангът на матрицата \mathbf{M} е ограничен отгоре от числото m . Количество на образите, които могат да бъдат надеждно съхранени в корелационната памет не може да превишава размерността на пространството на входните сигнали.

В реална ситуация количеството на образите, представени на входа на асоциативната памет, практически никога не се явяват ортогонални. Следователно корелационната памет характеризираща израза (2.34) понякога може да дава грешни резултати. Това значи, че асоциативната памет може да разпознава и класифицира образи, които никога по-рано не са виждани. Това нейно свойство може да се разгледа чрез следния набор от ключови образи:

$$\{\mathbf{x}_{\text{key}}\} : \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_q,$$

И съответните му запаметени образи:

$$\{\mathbf{y}_{\text{mem}}\} : \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q.$$

За описание на близостта на ключовите образи в линейното пространство на сигналите въвеждаме понятието *общност* (community) и определяме общността на множеството от образи $\{\mathbf{x}_{\text{key}}\}$ като нисък праг на скаларното произведение $\mathbf{x}_k^T \mathbf{x}_j$ за всеки два вектора от това множество. Нека $\hat{\mathbf{M}}$ е матрица на паметта, построена съгласно формула (2.34) в резултат на обучение на асоциативна памет на избор на ключови обтази $\{\mathbf{x}_{\text{key}}\}$ и съответстващи им на него набор $\{\mathbf{y}_{\text{mem}}\}$. Отговорът \mathbf{y} на тази памет на външно въздействие \mathbf{x}_j от множеството $\{\mathbf{x}_{\text{key}}\}$ може да се опише чрез израза (2.39). Предполагаме, че всички вектори от множеството $\{\mathbf{x}_{\text{key}}\}$ са единични (т.e. вектори с единична енергия). Предполагаме, че

$$\mathbf{x}_k^T \mathbf{x}_j \geq \gamma, \quad k \neq j. \quad (2.51)$$

Ако ниската граница γ е достатъчно голяма, то паметта няма да може да отграничи вектора y от отговора на всеки друг входен вектор от $\{x_{key}\}$. Ако ключовите образи от това множество имат вида:

$$x_j = x_0 + v, \quad (2.52)$$

Където v е някой стохастически вектор, то вероятно паметта ще разпознае вектора x_0 и асоциирания с него вектор y_0 , а не с векторите от фактическите множества на образите, използвани в процеса на обучение. Символно x_0 и y_0 обозначават никога преди това невиждани сигнали. Този феномен може да се нарече *логика на живота*.

2.12. Адаптация

При решаването на задачи често се оказва, че едно от основните измервания на процеса на обучение се оказва пространството, а при други – времето. Биологичният вид, от насекоми до човека имат способността да представят временна структура на опита. Такова представяне позволява на животните да *адаптират* (adapt) своето поведение към временна структура на събития в пространството.

Ако невронната мрежа работи в *стационарна* (stationary) среда (т.е. среда, статистическата характеристика на която не се изменя във времето), тя теоретично може да бъде обучена с най-съществените характеристики на средата с помощта на учител. Например, синаптическите тегла на мрежите могат да се изчислят в процеса на обучение на множество данни, представлящи средата. След това при завършване на процеса на обучение синаптическите тегла на мрежата отразяват статистическата структура на средата, която се счита за неизменна или „замразена“. За извлечането и използването на придобития опит на обучаемата система се основава на тази или друга форма на паметта.

Често околната среда се нарича *нестационарна* (nonstationary). Това значи, че статистическите параметри на входните сигнали, генерирали среда изменяща се във времето. В такъв род ситуации, методите за обучение с учител доказват своята несъстоятелност, така както мрежата няма средства за проследяване на статистическите вариации на средата, с която работи. Затова е необходимо постоянно адаптиране на свободните параметри на мрежата към нарастващите сигнали в режим на реално време, т.е. адаптивните системи трябва да отговарят на всеки следващ сигнал като на нов. Процеса на обучение в адаптивните системи не завършва, когато постъпят нови сигнали за обработка. Такава форма на обучение се нарича *непряка форма на обучение* (continuous) или *обучение на летене* (learning on-the-fly).

За реализацията на непрякото обучение може да се използват *линейни адаптивни филтри* (linear adaptive filter), построени за линейния суматор (т.е. отделният неврон функционира в линеен режим). Не сложни структури (а в много случаи благодарение и на тях) днес се използват широко в такива несходни области като радиолокацията, сейзмологията, връзк и биометрия. Теорията на линейните адаптивни филтри вече е достигнала своето развитие в стадия на зрелост, но това не трябва да се свързва с нелинейни филтри.

При изследването на непрякото обучение и примерите в теорията на невронните мрежи възниква следния въпрос: Как невронните мрежи могат да адаптират своето поведение към измененията на временната структура на входния сигнал в поведенческото пространство? Един от отговорите на този фундаментален въпрос предполага, че изменението на статистическите характеристики на нестационарните процеси протича бавно, затова този процес може да се разглежда като *псевдостационарен*. Примери:

- Синтеза на речевия сигнал може да се разглежда като стационарен процес за интервала от време 10-30 милисекунди.
- Ехото на радара от дъното на океана може да се счита за стационарен процес за интервал от време няколко секунди.
- При дългосрочни прогнози на времето синоптическите данни могат да се разглеждат като стационарни за интервал от време няколко минути.

Използва се свойството псевдостационарност в стохастическите процеси, което може да увеличи срока на ефективност на работата на невронните мрежи за сметка на периодично преучаване (retraining), позволяващо да се отчитат вариациите на входните данни.

Може да се използва по-точен *динамичен* (dynamic) подход. За тази цел използваме следната последователност от действия:

- Избираме достатъчно кратък интервал от време, в който данните могат да се считат за псевдостационарни, и се използват за обучение на мрежи.
- След получаване на новия обучаващ пример, трябва да се премахне най-стария вектор и да се добави нов пример.
- Използваме обновения избор за обучение на мрежата.
- Непрекъснато се повтаря описаната процедура.

Описаният алгоритъм позволява да се придават временни свойства в архитектурата на невронната мрежа се реализират образи на принципа *непрекъснато обучение на поредни във времето примери* (continual learning with time-ordered examples). При използването на такъв подход невронните мрежи може да се считат за *нелинеен адаптивен филтър* (nonlinear adaptive filter), представляващ обобщение на линейните адаптивни филтри.

2.13. Статистическа природа на процеса на обучение

Ще разгледаме еволюцията на вектора на теглата на коефициентите \mathbf{w} , а алгоритъма за обучение на невронните мрежи може да се разглежда като цикличен. Ще оценяваме смо отклонението на целевата функция $f(\mathbf{x})$ и фактическата функция $F(\mathbf{x}, \mathbf{w})$, реализирана в невронните мрежи. Векторът \mathbf{x} е входен сигнал. Неговото отклонение може да се изрази в статистически термини.

Невронните мрежи се разглеждат като една от формите, в които с помощта на процеса на обучение може да се закодират *емпиричните знания* (empirical knowledge) за физическите явления и околната среда. Под термина „емпирически знания“ се разбира някакъв набор от измервания, характеризиращи дадено явление. Пример за стохастическо явление, описано от случайния вектор \mathbf{X} , състоящ се от *независими променливи* (independent variable) и случайния скалар D , представляващ *зависима променлива* (dependent variable). Всеки елемент от случайния вектор \mathbf{X} може да има свой физически смисъл. Предположението, че зависимата променлива D е скаларна е направено с цел да се опрости изложenia материали, без загуба на общности.

Предполагаме, че съществуват N различни случайни вектора \mathbf{X} , обозначени с $\{\mathbf{x}_i\}_{i=1}^N$ и съответните им множество реализации на случайния скалар D , което обозначаваме с $\{d_i\}_{i=1}^N$. Тези реализации (измервания) представляват обучаващата извадка

$$\mathbf{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N. \quad (2.53)$$

Обучението не разполага със знания за функционалната взаимовръзка между \mathbf{X} и D :

$$D = f(\mathbf{X}) + \boldsymbol{\epsilon}, \quad (2.54)$$

Където $f(\cdot)$ е някоя *дeterministic функция* (deterministic) на вектора на аргумента; $\boldsymbol{\epsilon}$ -очаквана грешка (expectational error), предаваща нашето „незнание“ за зависимостта между \mathbf{X} и D . Статистическият модел описан от израза (2.54) се нарича *регресивен* (regression model) (фиг.2.20, а). Очакваната грешка $\boldsymbol{\epsilon}$ се явява случайна величина с нормално разпределение и нулево математическо очакване. Регресивният модел, илюстриран на фиг 2.20, а показва две важни свойства: с

- Средната стойност на очакваната грешка $\boldsymbol{\epsilon}$ за всички реализации на \mathbf{x} е равна на нула, т.е.

$$E[\boldsymbol{\epsilon}|\mathbf{x}] = 0, \quad (2.55)$$

Където E е статистически оператор на математическото очакване. Естествено следствие на това свойство е, че регресивната функция $f(\mathbf{x})$ се явява условен среден модел на изхода D за входния сигнал $\mathbf{X}=\mathbf{x}$:

$$f(\mathbf{x}) = E[D|\mathbf{x}]. \quad (2.56)$$

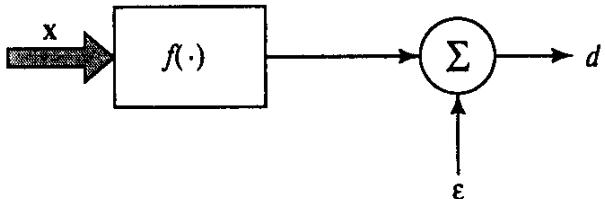
- Очакваната грешка $\boldsymbol{\epsilon}$ не се корелира с функцията на регресията $f(\mathbf{X})$, т.е.

$$E[\boldsymbol{\epsilon}f(\mathbf{X})] = 0. \quad (2.57)$$

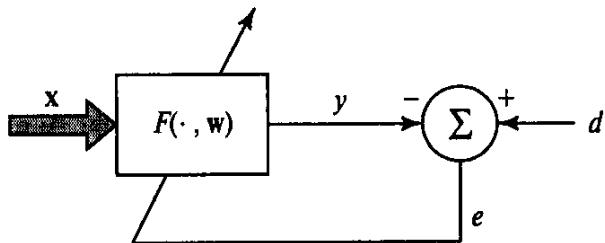
Това свойство е известно като *принцип на ортогоналност* (principle of orthogonality), който гласи, че всяка всяка информация от D , допусната чрез входния сигнал \mathbf{X} , се закодира във функцията на регресията $f(\mathbf{X})$. Равенството (2.57) се свежда до следния вид:

$$E[\boldsymbol{\epsilon}f(\mathbf{X})] = E[E[\boldsymbol{\epsilon}f(\mathbf{X})|\mathbf{x}]] = E[f(\mathbf{X})E[\boldsymbol{\epsilon}|\mathbf{x}]] = E[f(\mathbf{X}) \cdot 0] = 0.$$

Регресивният модел (фиг. 2.20, а) представлява математическо описание на стохастическата среда. В нея векторът \mathbf{X} се използва за описание или предсказване на зависимата променлива D . На фиг 2.20, б е представен съответния „физически“ модел на данните на средата. Това е вторият модел, основан на невронните мрежи, позволяващ закодиране на емпиричните знания, заключени в обучаващата извадка T , с помощта на съответния набор от вектори на синаптическите тегла \mathbf{w} :



a)



б)

Фиг.2.20. Математическо (а) и физическо (б) представяне на невронните мрежи
 $T \rightarrow w$. (2.58)

Така образно невронните мрежи обезпечават апроксимирането на регресионният модел, представен на фиг. 2.20, а. Нека фактическият отклик на невронните мрежи на входния вектор x се обозначава със следните вероятностни променливи:

$$Y = F(\mathbf{X}, \mathbf{w}). \quad (2.59)$$

Където $F(\cdot, \mathbf{w})$ е функция отразяваща входните данни в изходни, реализирана с помощта на невронните мрежи. За избора на данни за множеството T , представено във вида (2.53), вектора на синаптическите тегла може да се изчисли чрез:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (d_i - F(\mathbf{x}_i, \mathbf{w}))^2, \quad (2.60)$$

Където коефициента $1/2$ се въвежда за съвместимост с обучението използвани по-рано. Ако не вземем под внимание коефициента $1/2$, функцията от стойности $E(\mathbf{w})$, описва сумата от квадратите на разстоянията между d и фактическият отговор y на невронните мрежи за всеки пример за обучение от T .

Нека E_T е оператор за усреднение (average operator) по цялата обучаваща извадка T . Променливите, или техните функции, обработвани от оператора за осреднение E_T , обозначаваме с x и d . При това двойката (x, d) представя всеки конкретен обучаващ пример от извадката T . За разлика от оператора за усреднение, статистическото очакване E функционира върху множеството от всички стойности на случайните променливи X и D , подмножество на които се явява T . Разликата между операторите E и E_T ще бъде показана по-долу.

След преобразованията описани във формула (2.58), функциите $F(x, w)$ и $F(x, T)$, се явяват взаимосвързани и изразът (2.60) може да се представи във вида:

$$E(w) = \frac{1}{2} E_T[(d_i - F(x_i, T))^2]. \quad (2.61)$$

Добавяме функцията $f(\mathbf{x})$ с аргументи $(d_i - F(\mathbf{x}_i, T))^2$ и след използване на (2.54) получаваме:

$$d - F(\mathbf{x}, T) = (d - f(\mathbf{x})) + (f(\mathbf{x}) - F(\mathbf{x}, T)) = \epsilon + (f(\mathbf{x}) - F(\mathbf{x}, T)).$$

Поставяме този израз в (2.61) и след разкриване на скобите получаваме следния еквивалентен вид:

$$E(w) = \frac{1}{2} E_T[\epsilon^2] + \frac{1}{2} E_T[(f(\mathbf{x}) - F(\mathbf{x}, T))^2] + E_T[\epsilon(f(\mathbf{x}) - F(\mathbf{x}, T))]. \quad (2.62)$$

Последното слагаемо в първата част на формула (2.62) е равно на нула по две причини.

Очакваната грешка ϵ не корелира с регресивната функция $f(\mathbf{x})$, което се вижда от израза (2.57), интерпретиран в термините на оператора E_T .

Очакваната грешка ϵ отнесена към регресивният модел, изобразен на фиг. 2.20,а в същото време апроксимиращата функция $F(\mathbf{x}, w)$ се отнася към невронния модел показан на фиг.2.20, б.

Следователно изразът (2.62) може да се упрости:

$$E(w) = \frac{1}{2} E_T[\epsilon^2] + \frac{1}{2} E_T[(f(\mathbf{x}) - F(\mathbf{x}, T))^2]. \quad (2.63)$$

Първото слагаемо в първата част на израза (2.63) описва дисперсия на очакваните грешки (регресивни модели) ϵ , изчислени върху обучаващата извадка T . тази *изходна* (intrinsic) грешка не зависи от вектора на теглата w . Тя може да не се отчита, т.к. главната задача е минимизация на функцията на стойностите $E(w)$ относно вектора w . Следва да се отчита, че стойността на вектора на теглата w^* минимизира функцията от стойностите $E(w)$ също така ще минимизира и средното по ансамбли квадратично разстояние между регресивната функция $f(\mathbf{x})$ и функцията на апроксимация $F(\mathbf{x}, w)$. Естественото измерване на ефективността на използване на $F(\mathbf{x}, w)$ за програмирането на желания отговор d се явява следната функция:

$$Lav(f(\mathbf{x}), F(\mathbf{x}, w)) = E_T[(f(\mathbf{x}) - F(\mathbf{x}, T))^2]. \quad (2.64)$$

Този резултат има фундаментално значение, т.к. обезпечава математическата основа за изучаване на зависимостта между изместяване и дисперсия, получени в резултат на използваната $F(\mathbf{x}, w)$ в качеството на апроксимация на функцията $f(\mathbf{x})$.

Дileма между изместяване и дисперсия

От формула (2.56) и квадрата на разстоянието между функциите $F(\mathbf{x}, w)$ и $f(\mathbf{x})$ може да се определи следния вид на формулата:

$$Lav(f(\mathbf{x}), F(\mathbf{x}, w)) = E_T[(E[D|\mathbf{X} = \mathbf{x}] - F(\mathbf{x}, T))^2]. \quad (2.65)$$

Този израз може да се нарече средна стойност на *грешките на оценяване* (estimation error) на регресивната функция $f(\mathbf{x}) = E[D|\mathbf{X}=\mathbf{x}]$ и функцията на приближаване $F(\mathbf{x}, \mathbf{w})$, изчислени върху обучаващата извадка \mathbf{T} . Условно средната стойност $E[D|\mathbf{X}=\mathbf{x}]$ има постоянно математическо очакване върху обучаващата извадка \mathbf{T} . След като добавим средното $E_{\mathbf{T}}[F(\mathbf{x}, \mathbf{T})]$, получаваме:

$$E[D|\mathbf{X}=\mathbf{x}] - F(\mathbf{x}, \mathbf{T}) = (E[D|\mathbf{X}=\mathbf{x}] - E_{\mathbf{T}}[F(\mathbf{x}, \mathbf{T})]) + (E_{\mathbf{T}}[F(\mathbf{x}, \mathbf{T})] - F(\mathbf{x}, \mathbf{T})).$$

След пребразуване и след използване на изразите (2.61) и (2.62), формула (2.65) може да се запише в следния вид:

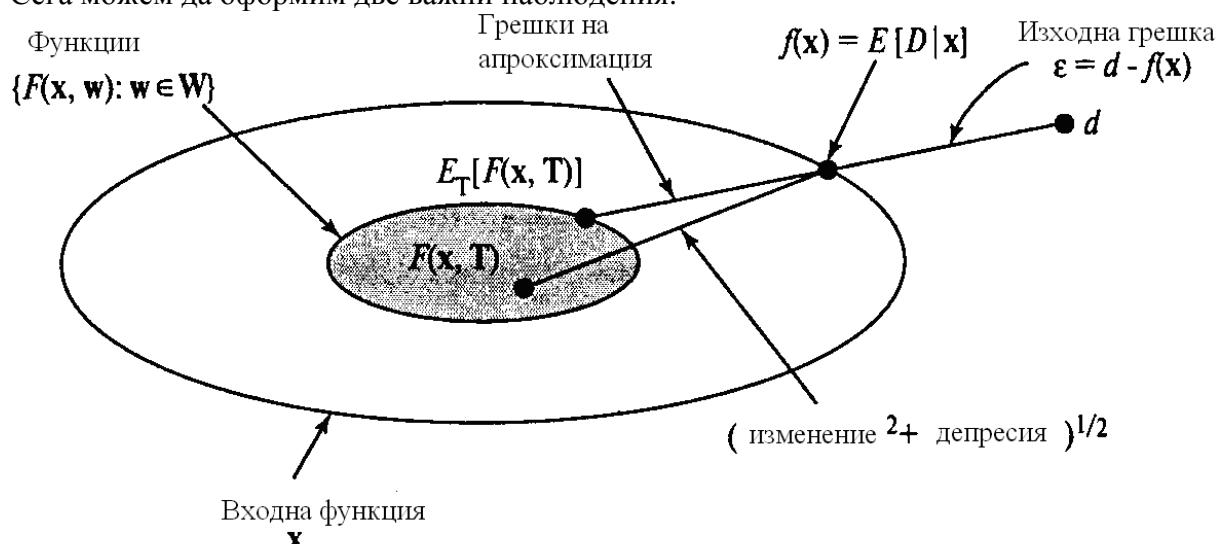
$$L_{av}(f(\mathbf{x}), F(\mathbf{x}, \mathbf{T})) = B^2(\mathbf{w}) + V(\mathbf{w}), \quad (2.66)$$

Където $B(\mathbf{w})$ и $V(\mathbf{w})$ определят следния образ:

$$B(\mathbf{w}) = E_{\mathbf{T}}[F(\mathbf{x}, \mathbf{T})] - E[D|\mathbf{X}=\mathbf{x}], \quad (2.67)$$

$$V(\mathbf{w}) = E_{\mathbf{T}}[(F(\mathbf{x}, \mathbf{T}) - E_{\mathbf{T}}[F(\mathbf{x}, \mathbf{T})])^2]. \quad (2.68)$$

Сега можем да оформим две важни наблюдения:



Фиг.2.21. Различни източници на грешки при решаване на задачи за регресия

- Елементът $B(\mathbf{w})$ описва изместване (bias) на средната стойност на функцията на приближаване $F(\mathbf{x}, \mathbf{T})$ относно функцията на регресия $f(\mathbf{x}) = E[D|\mathbf{X}=\mathbf{x}]$. Този елемент отразява неспособността на невронните мрежи, представляващи функцията $F(\mathbf{x}, \mathbf{w})$ да бъде точно приближавана от регресивната функция $f(\mathbf{x}) = E[D|\mathbf{X}=\mathbf{x}]$. Така образът на элемента $B(\mathbf{w})$ може да се счита за *грешка при приближаване* (approximation error).
- Елементът $V(\mathbf{w})$ представлява дисперсия (variance) на приближаващата функция $F(\mathbf{x}, \mathbf{T})$ на всяко обучаващо множество \mathbf{T} . Това слагаемо отразява нееднаквостта на информацията за регресивната функция $f(\mathbf{x})$, съдържаща се в обучаващото множество \mathbf{T} . Така елементът $V(\mathbf{w})$ може да се счита за *грешка при оценяване* (estimation error).

На фиг. 2.21 са изобразени взаимовръзките между целевата и апроксимиращата функция, нагледно е показано как се грешките на оценяване – изместване и дисперсия. Изместването $B(\mathbf{w})$ и дисперсията $V(\mathbf{w})$ на функцията на априорна апроксимация $F(\mathbf{x}, \mathbf{w}) = F(\mathbf{x}, \mathbf{T})$ трябва да са много малки.

В невронните мрежи обучението на данните на извадки с фиксиран размер с малко изместване се достига голяма депресия. Едновременно може да се намали изместването и дисперсията само в един случай – когато размерът на обучаващото множество е безкрайно голям. Този проблем се нарича *дилема изместване/депресия* (bias/variance dilemma). Следствие на този проблем се явява бавната сходимост на процеса на обучение. Дилемата може да се избегне, ако преднамерено въведем такова изместване, което води дисперсията до отричане или до значително намаляване. Но трябва да се убедим в това, че построеното в системата изместване е приемливо. Например в контекста на класификацията на образите, изместването може да се счита за „приемливо”, ако оказва голямо влияние на средноквадратичните грешки само в случаите на регресия, която не принадлежи към очаквания клас. В общият случай изместването е необходимо да се задава потенциално за всяка предметна област. На практика за достижане на целта се използва *ограничение* (constrained) на сетивната архитектура, която работи по-добре от архитектурата за общо предназначение. В частност ограничението (и изместването) могат да приемат формата на априорни знания, в пространството на архитектурата на невронните мрежи по пътя на *съвместното използване на тегла* (ако няколко синапса на мрежата се намират под управлението на един и същи теглови коефициент за време) и/или създаването на *локални рецептори* (local receptive field), свързани с отделните невронни мрежи.

2.14. Теория на статистическото обучение

В този раздел ще се обърне внимание на *теорията на обучение* (learning theory), свързана с решението на фундаментални въпроси, които управляват обобщените способности на невронните мрежи. Този въпрос ще бъде разгледан в контекста на обучението с учител.

Моделът на обучение с учител се състои от три взаимосвързани компонента.

- *Среда* (environment). Средата е стационарна. Тя е представена от вектора \mathbf{x} с фиксирана, но неизвестна функция на разпределение на вероятности $F_X(\mathbf{x})$.
- *Учител* (teacher). Учителят генерира желаният отговор d за всеки входен вектор \mathbf{x} , получен от външната среда, в съответствие с условната функция на разпределение $F_X(\mathbf{x}|d)$, която е фиксирана, но неизвестна. Желаният отговор d и входният вектор \mathbf{x} са свързани със следващото съотношение:

$$d = f(\mathbf{x}, v), \quad (2.69)$$

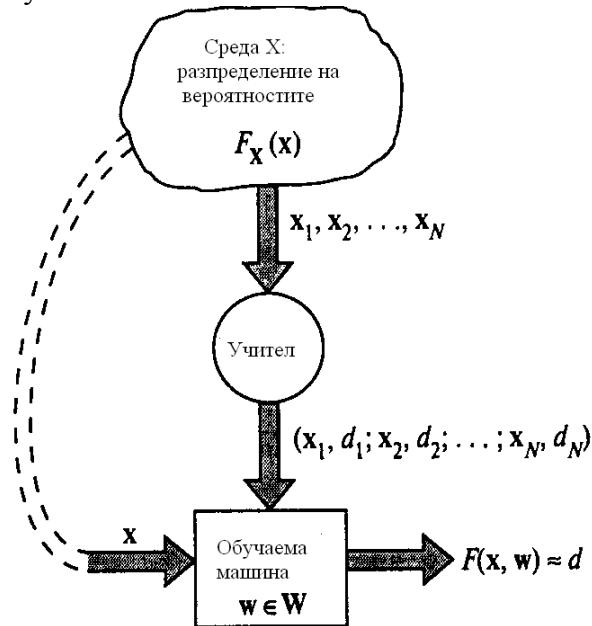
Където v е шум, т.e. отначало се предполага за „зашумени” данни на учителя.

- *Обучаема машина* (learning machine). Обучаемата машина (невронна мрежа) е „способна да реализира множество функции на отражение” вход-изход, описани от съотношението:

$$y = F(\mathbf{x}, \mathbf{w}), \quad (2.70)$$

Където y – фактически отговор, генериран от обучаемата машина в отговор на входния сигнал \mathbf{x} ; \mathbf{w} – избор на свободни параметри (синаптически тегла), избрани от пространството на параметри \mathbf{W} .

Уравненията (2.69) и (2.70) записани в термините на примерите, използвани при обучението.



Фиг. 2.22. Модел на процеса на обучение с учител

Задачата на обучението с учител се състои в избор на конкретна функция $F(\mathbf{x}, \mathbf{w})$, която оптимално (в статистически смисъл) априксимира очаквания отговор d . Избора се основава на множеството N неавтосими, равномерно разпределени примери на обучение, описани от формула (2.53). За удобство използваме следния запис:

$$\mathbf{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N.$$

Всяка двойка се избира от обучаващата машина на множеството \mathbf{T} с някоя обобщена функция на разпределение на вероятностите $F_{X,D}(\mathbf{x}, d)$, която както и другите функции на разпределение е фиксирана, но неизвестна. Принципните възможности на обучението с учител зависят от отговора на следния въпрос: Съдържат ли примерите от множеството $\{(\mathbf{x}_i, d_i)\}$ достатъчно информация за създаване на обучаема машина, която да има добри обобщаващи способности? Отговорът на този въпрос се намира в резултатите получени през 1971г. Пазглеждаме задачата за обучение с учител като *задача за априксимация* (approximation problem), състояща се в намиране на функцията $F(\mathbf{x}, \mathbf{w})$, която приближава желаната функция $f(\mathbf{x})$.

Нека $L(d, F(\mathbf{x}, \mathbf{w}))$ е мярка на загубите или несъответствия между желанияят резултат d , съответният му входен вектор \mathbf{x} , и отговора $F(\mathbf{x}, \mathbf{w})$, генериирани от обучаващата машина. В качеството на мярката $L(d, F(\mathbf{x}, \mathbf{w}))$ се разглежда *квадратичната функция на загубите* (quadratic loss function), определяща квадрата на разстоянието между $d = f(\mathbf{x})$ и априксимацията $F(\mathbf{x}, \mathbf{w})$.

$$L(d, F(\mathbf{x}, \mathbf{w})) = (d - F(\mathbf{x}, \mathbf{w}))^2. \quad (2.71)$$

Квадратичното разстояние в формула (2.64) – то е усреднение на множеството от теглата на всички примери (\mathbf{x}, d) разширени по мярката $L(d, F(\mathbf{x}, \mathbf{w}))$.

Основното свойство на задачите в теориите на статистическото обучение е това, че функцията на загубите $L(d, F(\mathbf{x}, \mathbf{w}))$ не играе особена роля.

Очакваната величина на загубите се определя от *функцията на риска* (risk functional)

$$R(\mathbf{w}) = \int L(d, F(\mathbf{x}, \mathbf{w})) dF_{\mathbf{X}, D}(\mathbf{x}, d), \quad (2.72)$$

Където интегралът се взема по всички възможни стойности на (\mathbf{x}, d) . Целта на обучението с учител се явява минимизация на функцията на риска $R(\mathbf{w})$ в класа на функциите за апроксимация $\{F(\mathbf{x}, \mathbf{w}), \mathbf{w} \in \mathbf{W}\}$. Оценката на функцията на риска се усложнява от това, че обобщаващата функция на разпределение $F_{\mathbf{X}, D}(\mathbf{x}, d)$ е независима. При обучението с учител всичката достъпна информация се съдържа в множеството данни за обучение \mathbf{T} . За да се избегне това математическо усложнение се използва индуктивният принцип на минимизация на емпирическия риск. Той се основава на достъпността на обучаващото множество \mathbf{T} , което е идеално съгласувано с философията на невронните мрежи.

Някои основни определения

Сходимост на вероятностите. Разглеждаме последователността на случайните променливи a_1, a_2, \dots, a_N . Тази последователност се счита *сходяща по вероятност* (converge in probability) към случайната променлива a_0 , за всяко $\sigma > 0$ се изпълнява седното вероятностно съотношение:

$$P(|a_N - a_0| > \sigma) \xrightarrow{P} \text{при } N \rightarrow \infty. \quad (2.73)$$

Долна и горна граница (supremum and infimum). Долната граница на непразното множество от скаларни величини \mathbf{A} ($\sup \mathbf{A}$) се нарича най-малкия скалар x , за който е изпълнено неравенството $x \geq y$ за всяко $y \in \mathbf{A}$. Ако такава скаларна величина несъществува, то считаме, че горната граница на множеството \mathbf{A} се явява безкрайна. Аналогично, долната граница на непразното множество от скалари \mathbf{A} ($\inf \mathbf{A}$) се нарича най-голяма от скаларите x , за който е изпълнено неравенството $x \leq y$ за всяко $y \in \mathbf{A}$. Ако такава скаларна величина не съществува, то се счита, че долната граница на непразното множество \mathbf{A} се явява безкрайна.

Функция на емпиричния риск. За обучаващото множество $\mathbf{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N$ функцията на емпиричният риск се определя в термините на функциите на загубите $L(d, F(\mathbf{x}, \mathbf{w}))$ по следния начин:

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(d_i, F(\mathbf{x}_i, \mathbf{w})). \quad (2.74)$$

Строга последователност (strict consistency). Разглеждаме множеството \mathbf{W} от функции $L(d, F(\mathbf{x}, \mathbf{w}))$, разпределението на които се определя от интегралната функция на разпределение $F_{\mathbf{X}, D}(\mathbf{x}, d)$. Нака $\mathbf{W}(c)$ е непразно подмножество на това множество, така че:

$$\mathbf{W}(c) = \left\{ \mathbf{w} : \int L(d, F(\mathbf{x}, \mathbf{w})) \geq c \right\}, \quad (2.75)$$

Където $c \in (-\infty, +\infty)$. Функцията на емпиричния риск се счита *строго последователна* (strictly consistent), ако за всяко подмножество $\mathbf{W}(c)$ се обезпечава сходимост на вероятностите:

$$\inf_{\mathbf{w} \in \mathbf{W}(c)} R_{\text{emp}}(\mathbf{w}) \xrightarrow{P} \inf_{\mathbf{w} \in \mathbf{W}(c)} R(\mathbf{w}) \text{ при } N \rightarrow \infty. \quad (2.76)$$

Принципи за минимизация на емпирическия риск

Основната идея на принципите на *минимизация на емпиричния риск* (empirical risk minimization) се състои в използването на функционала на емпирическия риск $R_{\text{emp}}(\mathbf{w})$, определен от формула (2.74). Този нов функционал се отразява от функционала $R(\mathbf{w})$, зададен с формула (2.72) в два аспекта:

- Той не зависи от неизвестната функция на разпределение $F_{X,D}(\mathbf{x}, d)$.
- Теоретично той може да се минимизира по вектора на теглата на коефициентите \mathbf{w} .

Нека \mathbf{w}_{emp} и $F(\mathbf{x}, \mathbf{w}_{\text{emp}})$ – вектор на теглата и съответстващото на него отражение, което минимизира функционала на емпиричния риск $R_{\text{emp}}(\mathbf{w})$, определен по формула (2.74). Аналогично, нека \mathbf{w}_o и $F(\mathbf{x}, \mathbf{w}_o)$ – вектор на тегловите коефициенти и отражени, минимизиращи фактическия функционал на риска $R(\mathbf{w})$, зададен с формула (2.72). Векторите \mathbf{w}_{emp} и \mathbf{w}_o принадлежат на пространството на теглата \mathbf{W} . Трябва да се намери условие, при което апроксимиращото отражение $F(\mathbf{x}, \mathbf{w}_{\text{emp}})$ да е достатъчно „близко“ до фактическото отражение $F(\mathbf{x}, \mathbf{w}_o)$ (в качеството на мярка за близостта ще използваме разликата между $R_{\text{emp}}(\mathbf{w})$ и $R(\mathbf{w})$).

За някое фиксирано $\mathbf{w} = \mathbf{w}^*$ функционала на риска $R(\mathbf{w}^*)$ определя *математическо очакване* на случайните променливи, определени от съотношението:

$$Z_{\mathbf{w}^*} = L(d, F(\mathbf{x}, \mathbf{w}^*)). \quad (2.77)$$

За разлика от него функционала на емпиричния риск $R_{\text{emp}}(\mathbf{w}^*)$ обезпечава *емпирична аритметична средна стойност* (empirical (arithmetic) mean) на случайната променлива $Z_{\mathbf{w}^*}$. Съгласно *законът на големите числа* (law of large numbers), който е една от основните теореми на теорията на вероятностите, за обучаващото множество \mathbf{T} с безкрайно голям размер N на емпиричната средна случайна променлива $Z_{\mathbf{w}^*}$ в общият случай се свежда към очакваната стойност. Това наблюдение обезпечава теоретичната база за използваната функция на емпиричния риск $R_{\text{emp}}(\mathbf{w})$ вместо функцията на риска $R(\mathbf{w})$. Този факт, че емпиричната средна променлива $Z_{\mathbf{w}^*}$ е сходяща към очакваната стойност, не означава, че векторът на тегловите коефициенти \mathbf{w}_{emp} минимизира функционала на емпиричния риск $R_{\text{emp}}(\mathbf{w})$ и ще минимизира и функционала на риска $R(\mathbf{w})$.

Това изследване приблизително може да се удоволетвори, при прилагане на следния подход. Ако функционала на емпиричния риск $R_{\text{emp}}(\mathbf{w})$ апроксимира изходния функционал на риска $R(\mathbf{w})$ равномерно по \mathbf{w} с точност ϵ , то минимума на $R_{\text{emp}}(\mathbf{w})$ ще се намира не повече от 2ϵ от минимума на $R(\mathbf{w})$. Това означава, че е необходимо изпълнението на следващото условие. За някое $\mathbf{w} \in \mathbf{W}$ и $\epsilon > 0$ трябва да е изпълнено вероятностното съотношение:

$$P\left(\sup_{\mathbf{w}} |R(\mathbf{w}) - R_{\text{emp}}(\mathbf{w})| > \varepsilon\right) \rightarrow 0 \text{ при } N \rightarrow \infty. \quad (2.78)$$

Ако се изпълнява условие (2.78), то може да се твърди, че *векторът на теглата w на средно емпиричният риск е равномерно сходящ към своята очаквана стойност*. Така ако на всяка зададена точност ε и някое положително α се изпълнява неравенството:

$$P\left(\sup_{\mathbf{w}} |R(\mathbf{w}) - R_{\text{emp}}(\mathbf{w})| > \varepsilon\right) < \alpha, \quad (2.79)$$

Изпълнява се и следното неравенство:

$$P(R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) > 2\varepsilon) < \alpha. \quad (2.80)$$

С други думи, ако е изпълнено условие (2.79), то вероятността $(1 - \alpha)$ е решение на $F(\mathbf{x}, \mathbf{w}_{\text{emp}})$, минимизираща функционала на емпиричния риск $R_{\text{emp}}(\mathbf{w})$, обезпечаващ разлика от фактическият риск $R(\mathbf{w}_{\text{emp}})$ от минималният възможен фактически риск на величина, не по-голяма от 2ε . Това значи, че при изпълнение на (2.79) с вероятност $(1 - \alpha)$, едновременно се изпълняват следващите две неравенства:

$$R(\mathbf{w}_{\text{emp}}) - R_{\text{emp}}(\mathbf{w}_{\text{emp}}) < \varepsilon, \quad (2.81)$$

$$R_{\text{emp}}(\mathbf{w}_o) - R(\mathbf{w}_o) < \varepsilon. \quad (2.82)$$

Тези две отношения определят разликата между функционалите на фактическият и емпиричния риск в точка $\mathbf{w} = \mathbf{w}_{\text{emp}}$ и $\mathbf{w} = \mathbf{w}_o$. Отчитайки, че \mathbf{w}_{emp} и \mathbf{w}_o се явяват точки на минимум на функционалите $R_{\text{emp}}(\mathbf{w})$ и $R(\mathbf{w})$, може да се направи извода, че:

$$R_{\text{emp}}(\mathbf{w}_{\text{emp}}) \leq R_{\text{emp}}(\mathbf{w}_o). \quad (2.83)$$

От неравенствата (2.81) и (2.82) и вземането под внимание на неравенство (2.83) може да се запише:

$$R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) < 2\varepsilon. \quad (2.84)$$

Тъй като неравенства (2.81) и (2.82) се изпълняват едновременно с вероятност $(1 - \alpha)$, то с такава вероятност се изпълнява и неравенство (2.84). Може да се твърди, че с вероятност α ще бъде изпълнено неравенството

$$R(\mathbf{w}_{\text{emp}}) - R(\mathbf{w}_o) > 2\varepsilon,$$

Можем да формулираме *принцип на минимизация на емпиричния риск* (principle of empirical risk minimization), състоящ се от три части:

- Вместо функционала на риска $R(\mathbf{w})$ се използва функцията на емпиричния риск

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(d_i, F(\mathbf{x}_i, \mathbf{w}))$$

$$F(\mathbf{x}, \mathbf{w}) = \begin{cases} 0 & \text{для } \mathbf{x} \in \mathbf{L}_0, \\ 1 & \text{для } \mathbf{x} \in \mathbf{L}_1. \end{cases} \quad (2.87)$$

На

базата на множеството от примери за обучение $(\mathbf{x}_i, d_i), i = 1, 2, \dots, N$.

- Нека \mathbf{w}_{emp} е вектор на тегловите коефициенти, минимизиращи функционала на емпиричния риск $R_{\text{emp}}(\mathbf{w})$ в пространството от теглата \mathbf{W} . Тогава $R(\mathbf{w}_{\text{emp}})$ е сходящ по вероятност към минималната възможна стойност на фактическия риск $R(\mathbf{w})$, $\mathbf{w} \in \mathbf{W}$. При увеличаване на количествата N на примерите на обучение до безкрайност, функционала на емпирическия риск $R_{\text{emp}}(\mathbf{w})$ е равномерно сходящ към функционала на фактическия риск $R(\mathbf{w})$.
- Равномерната сходимост, определена от

$$P\left(\sup_{\mathbf{w} \in \mathbf{W}} |R(\mathbf{w}) - R_{\text{emp}}(\mathbf{w})| > \varepsilon\right) \rightarrow 0 \text{ при } N \rightarrow \infty,$$

се явява необходимо и достатъчно условие за непротиворечивост на принципа на минимизация на емпирическия риск.

За физическата интерпретация на този важен принцип провеждаме следното наблюдение. За обучаващата машина всички апроксимиращи функции са равни. С повишаване на обучението правоспособността на тези функции на апроксимация, които не противоречат на обучаващото множество $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ нараства. С увеличаване на количеството на използваните при обучението примери, и повишаването на „плътността“ на входното пространство, точката на минимума на функционала на емпирическия риск $R_{\text{emp}}(\mathbf{w})$ е сходяща по вероятност с точката на минимум на функционала на фактическия риск $R(\mathbf{w})$.

VC – измерване

Теорията на равномерната сходимост на функционала на емпиричния риск $R_{\text{emp}}(\mathbf{w})$ към функционала на фактическия риск $R(\mathbf{w})$ включва ограничение на скоростта на сходимост, която е основана на важни параметри, получили названието *измерения на Вапник-Червоненкис* (Vapnik-Chervonenkis dimension) (или просто *VC-измерване*). То се явява мярка за обем или изчисляване на мощността на семейства функции за класификация, реализирани от обучаемите машини.

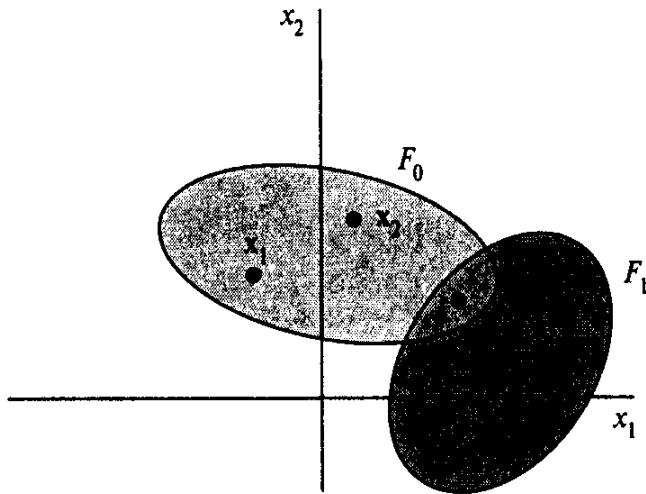
За да опишем VC-измерването ще разгледаме задачата за двоична класификация на образи, състоящи се от две стойности $d \in \{0, 1\}$. За обучението на правилата на приемане на решения или функциите на двоична класификация ще използват терминът *дихотомия* (dichotomy). Нека F е множество от дихотомии, реализирани от обучаващата машина, т.е.

$$F = \{F(\mathbf{x}, \mathbf{w}) : \mathbf{w} \in \mathbf{W}, F : \mathbb{R}^m \mathbf{W} \rightarrow \{0, 1\}\}. \quad (2.85)$$

Нека L е множество, съдържащо N точки на m -мерното пространство X на входните вектори:

$$L = \{\mathbf{x}_i \in X; i = 1, 2, \dots, N\}. \quad (2.86)$$

Дихотомията, реализирана от обучаващата машина, разбива множеството L на две непресичащи се подмножества L_0 и L_1 , такива че:



Фиг. 2.23. Диаграма за пример 2.1.

$$F(\mathbf{x}, \mathbf{w}) = \begin{cases} 0 & \text{для } \mathbf{x} \in \mathbf{L}_0, \\ 1 & \text{для } \mathbf{x} \in \mathbf{L}_1. \end{cases} \quad (2.87)$$

Нека $\Delta_F(\mathbf{L})$ е количество от различни дихотомии, реализирани от обучаващата машина; $\Delta_F(l)$ е максимум на $\Delta_F(\mathbf{L})$ на множеството от теглата \mathbf{L} , за което $|\mathbf{L}| = l$ където $|\mathbf{L}|$ е количество от елементи в \mathbf{L} . Ансамбълът на дихотомии \mathbf{F} се явява разбиване на множеството \mathbf{L} , ако $\Delta_F(\mathbf{L})=2^{|\mathbf{L}|}$, т.е. ако всички възможни дихотомии в \mathbf{L} могат да бъдат реализирани с функцията F . При това $\Delta_F(l)$ се нарича *функция на растежа* (growth function).

Пример 2.1.

На фиг. 2.23. е показано двумерно входно пространство \mathbf{X} , състоящо се от 4 точки $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$. Показаните на фигурата решения на функциите F_0 и F_1 съответства на класа (хипотези) 0 и 1. На фиг.2.23 се вижда, че функцията F_0 индуцира дихотомия

$$\mathbf{D}_0 = \{\mathbf{G}_0 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4\}, \mathbf{G}_1 = \{\mathbf{x}_3\}\}.$$

От друга страна функцията F_1 описва дихотомията

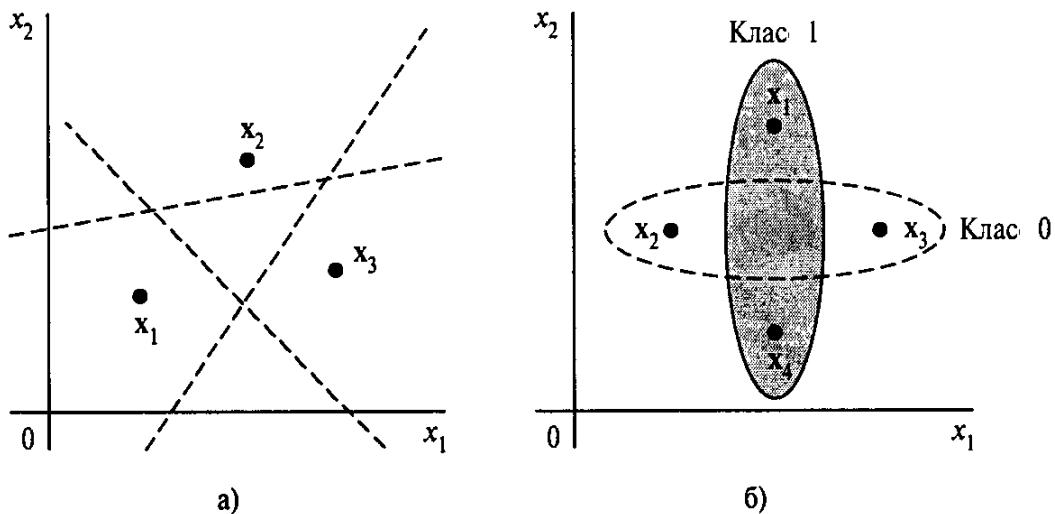
$$\mathbf{D}_1 = \{\mathbf{G}_0 = \{\mathbf{x}_1, \mathbf{x}_2\}, \mathbf{G}_1 = \{\mathbf{x}_3, \mathbf{x}_4\}\}.$$

Множеството \mathbf{G} се състои от 4 точки, с мощност $|\mathbf{G}| = 4$. Следователно

$$\Delta_F(\mathbf{G}) = 2^4 = 16.$$

Връщаме се към обсъждания ансамбъл дихотомии \mathbf{F} , описан от формула (2.85) и съответстващ на множеството от точки \mathbf{L} , зададено от формула (2.86), VC-измерването може да се определи така:

VC-измерването F се нарича мощност на най-голямото множество L, което се разбива от F.



Фиг.2.24. Две двумерни разпределения за пример 2.2

VC-измерването F (обозначава се с $\text{Vcdim}(F)$) се явява най-голямата стойност на N , за която $\Delta_F(N) = 2^N$. VC-измерването F на множеството от функциите на класификация $\{F(\mathbf{x}, \mathbf{w}) : \mathbf{w} \in \mathbf{W}\}$ е максималното число от образи, на които машината може да бъде обучена без грешка за всяка възможна бинарна маркировка на функциите на класификация.

Пример 2.2.

Разглеждаме просто решаващо правило в m -мерното пространство X на входните вектори, описано по следващия начин:

$$F : y = \phi(\mathbf{w}^T \mathbf{x} + b), \quad (2.88)$$

Където \mathbf{x} е m -мерен вектор на теглата; b – праг. Функцията на активация ϕ се явява прагова, т.e.

$$\phi(v) = \begin{cases} 1, & v \geq 0, \\ 0, & v < 0. \end{cases}$$

VC-измерването F решаващо правилото, определено по формула (2.88), определя съотношението:

$$\text{VCdim}(F) = m + 1. \quad (2.89)$$

На фиг.2.24 е илюстриран този резултат за двумерно входно пространство (т.e. $m = 2$). На фиг.2.24, а са показани три точки x_1 , x_2 и x_3 , а така три възможни варианта разделят тези точки. На фиг.2.24, б са изобразени четири точки x_1 , x_2 , x_3 и x_4 . Точките x_2 и x_3 се отнасят към клас 0, а точките x_1 и x_4 към клас 1. От фигурата се вижда, че точки x_2 и x_3 се отделят от точките x_1 и x_4 чрез една линия. Така VC-измерването F решаващо правилата описани по формула (2.88), при $m = 2$ равно на 3, съответстват на формула (2.89).

Пример 2.3.

Нека VC-измерването F се явява мярка за обем на множеството от функции на класификация (индикатори), то обучаващата машина с много числови свободни параметри ще бъде VC-измерване и обратното. Ще направим контра пример, опровергаващ това твърдение.

Разглеждаме еднопараметрово семейство от функции-индикатори

$$f(x, a) = \text{sgn}(\sin(ax)), a \in \mathfrak{R},$$

Където $\text{sgn}(\cdot)$ е функция на изчисляване на знака на аргумента. Предполагаме, за някое число N , за което е нужно да се намерят N точки, за които е необходимо да се построи разбиването. Това се удовлетворява от функцията $f(x, a)$, където

$$x_i = 10^{-i}, i = 1, 2, \dots, N.$$

Така се разделят точките на данните на два класа, определени последователно

$$d_1, d_2, \dots, d_N, d_i \in \{-1, 1\},$$

Параметъра a се определя от формулата

$$a = \pi \left(1 + \sum_{i=1}^N \frac{(1 - d_i)10^i}{2} \right).$$

От това можем да заключим, че VC-измерването на семейства от функции-индикатори $f(x, a)$ с единствен свободен параметър a е равно на безкрайност.

Важност на VC-измерването и неговата оценка

VC-измерването се явява сбор от *комбинирани понятия* (combination concept) и не е свързано с геометричното понятие измерване. То играе централна роля в теорията на статистическото обучение. VC-измерването е важно и от конструктивна гледна точка. Образно количеството примери, необходими за обучение на системата от данни на някой клас е строго пропорционално на изменението на VC-измерването на този клас.

В някои случаи VC-измерването определя свободните параметри на невроните мрежи. Границата на VC-измерването често се установява много лесно. Интерес представляват следните два резултата:

Нека N – произволна невронна мрежа с пряко разпространение, състояща се от неврони с прагова функция на активиране (Хевсайд)

$$\phi(v) = \begin{cases} 1, & v \geq 0, \\ 0, & v < 0. \end{cases}$$

VC-измерването на мрежата N съставя $O(W \log W)$, където W – общото количество на свободните параметри на мрежата.

Нека N -произволна многослойна невронна мрежа за пряко разпространение, състояща се от неврони със сигмоидеална функция на активиране.

$$\phi = \frac{1}{1 + \exp(-v)}.$$

Където VC-измерването на мрежата N е равно на $O(W^2)$, W – общото количество свободни параметри на мрежата.

Изводът е, че VC-измерването на мрежи, състоящи се от два типа неврони (с линейни и прагови функции на активиране) е пропорционално на W^2 . Резултатът отнасящ се до невронните мрежи със сигмоидеална функция на активиране е получен чрез двойна апроксимация. Първо невронът с прагова функция на активиране може да апроксимира възли със сигмоидеална функция и много синаптически тегла. Второ, линейните невронни мрежи апроксимират неврони със сигмоидеална функция и малки синаптически тегла.

Многослойните мрежи за пряко разпространение се наричат *ограничено VC-измерване*.

Конструктиви, независещи от разпределението на обобщаващата способност на невронните мрежи

Ще разгледаме особен тип задача за двоична класификация на образи, в които очакваният отговор се определя от множеството $d = \{0, 1\}$. Функцията на загубите може да приема следващите две стойности:

$$L(d, F(\mathbf{x}, \mathbf{w})) = \begin{cases} 0, & \text{ако } (\mathbf{x}, \mathbf{w}) = d, \\ 1, & \text{ако } F(\mathbf{x}, \mathbf{w}) \neq d. \end{cases} \quad (2.90)$$

При тези условия функционала на риска $R(\mathbf{w})$ и емпиричния риск $R_{\text{emp}}(\mathbf{w})$, определени от формули (2.72) и (2.74) съответно, могат да имат следната интерпретация.

- Функционала на риска $R(\mathbf{w})$ - вероятността от грешки на класификация (probability of classification error), обозначени с $P(\mathbf{w})$.
- Функционала на емпиричния риск $R_{\text{emp}}(\mathbf{w})$ - грешки при обучение (training error) (т.е. честото появяване на грешки в процеса на обучение), обозначено с $v(\mathbf{w})$.

Съгласно закона за големите числа (law of large numbers) емпирическата честота на възникване на каквото и да било събития почти винаги е сходяща към фактическата вероятност за същите събития при количеството на опитите, стремящо се към безкрайност (предполага се, че всички опити са независими и еднакво разпределени). Този резултат овori, че някой вектор \mathbf{w} , независещ от обучаващото множество и за някоя точност $\epsilon > 0$ се изпълнява неравенството:

$$P(|P(\mathbf{w}) - v(\mathbf{w})| > \epsilon) \rightarrow 0 \text{ при } N \rightarrow \infty, \quad (2.91)$$

Където N е размер на множеството на обучение. Но изпълняването на условие (2.91) не означава, че минимизацията на грешките на обучение $v(\mathbf{w})$ при използването на някои правила на класификация (т.е. данните на векторите на теглата \mathbf{w}) водят до минимизация на вероятностите на грешки на класификация $P(\mathbf{w})$. За съществено големия размер на N на обучаващото множество, близостта между $P(\mathbf{w})$ и $v(\mathbf{w})$ следва от следващото

$$P\left(\sup_{\mathbf{w}} |P(\mathbf{w}) - v(\mathbf{w})| > \epsilon\right) \rightarrow 0 \text{ при } N \rightarrow \infty. \quad (2.92)$$

В дадения случай се говори за *равномерна сходимост на честотата на грешките на обучение с вероятности $v(\mathbf{w})=P(\mathbf{w})$* .

Понятието VC-измерване натрупва ограничения на скоростта на равномерни сходимости. В частност, за множеството от функциите на класификация с VC-измерване равно на h , се използва следното неравенство:

$$P\left(\sup_{\mathbf{w}} |P(\mathbf{w}) - v(\mathbf{w})| > \epsilon\right) < \left(\frac{2eN}{h}\right)^h \exp(-\epsilon^2 N), \quad (2.93)$$

Където N е размерът на обучаващото множество; e – основа на натурален логаритъм. За да се достигне равномерна сходимост трябва да се постигнат малки стойности на първата част на неравенство (2.93) за N . За това може да послужи множителя

$\exp(-\epsilon^2 N)$, множителят експоненциално намалява с нарастването на N . Оставащият множител $(2eN/h)^h$ представлява граница на растежа на функцията $\Delta_F(l)$ за семействата от функции $F=\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathbf{W}\}$ при $l \geq h \geq 1$. Този резултат описва лемата на Сауер (Sauer's lemma). Ограничавайки бързия растеж на функцията може да се обезпечи сходимост на първата част на неравенството към 0 при N стремящо се към безкрайност. Това се удовлетворява, ако VC-измерването h не се явява безкрайно голямо. С други думи краят на VC-измерването се явява необходим и достатъчно условие за равномерна сходимост на принципа на минимиза VC-измерване на емпиричния риск. Ако изходното пространство X на крайното множество, то семейството на дихотомиите F има крайно VC-измерване по X . Обратното твърдение не винаги е вярно.

Нека α е вероятност на събитието

$$\sup_{\mathbf{w}} |P(\mathbf{w}) - v(\mathbf{w})| \geq \epsilon.$$

Тогава с вероятност $(1 - \alpha)$ може да се твърди, че всеки вектор на тегловите коефициенти $\mathbf{w} \in \mathbf{W}$ удовлетворява следващото неравенство

$$P(\mathbf{w}) < v(\mathbf{w}) + \epsilon. \quad (2.94)$$

Използвайки неравенство (2.93) и определената вероятност α , можем да запишем:

$$\alpha = \left(\frac{2eN}{h} \right)^h \exp(-\epsilon^2 N). \quad (2.95)$$

Нека $\epsilon_0(N, h, \alpha)$ е някоя стойност на ϵ , удовлетворяваща съотношението (2.95).

Тогава получаваме следния важен резултат

$$\epsilon_0(N, h, \alpha) = \sqrt{\frac{h}{N} \left[\log \left(\frac{2N}{h} \right) + 1 \right] - \frac{1}{N} \log \alpha}. \quad (2.96)$$

Величината $\epsilon_0(N, h, \alpha)$ се нарича *доверителен интервал* (confidence interval). Тази стойност зависи от размера на N , VC-измерването h и вероятността α .

Описаният израз (2.93) при $\epsilon = \epsilon_0(N, h, \alpha)$ достига в най-лошия случай вероятност $P(\mathbf{w})=1/2$, но не за малки стойности на $P(\mathbf{w})$, които са интересни при решаване на практически задачи. За малки стойности на $P(\mathbf{w})$ е важно ограничението, което може да се получи в резултат на някои модификации на неравенство (2.93).

$$P \left(\sup_{\mathbf{w}} \frac{|P(\mathbf{w}) - v(\mathbf{w})|}{\sqrt{P(\mathbf{w})}} > \epsilon \right) < \left(\frac{2eN}{h} \right)^h \exp \left(\frac{-\epsilon^2 N}{4} \right). \quad (2.97)$$

В литературата са представени различни видове ограничения (2.97), зависещи от конкретната форма на неравенствата, използвани за получаването им. Ако неравенството (2.97) има вероятност $(1 - \alpha)$ едновременно за всяко $\mathbf{w} \in \mathbf{W}$ се изпълнява съотношението:

$$P(\mathbf{w}) \leq v(\mathbf{w}) + \epsilon_1(N, h, \alpha, v), \quad (2.98)$$

Където $\epsilon_1(N, h, \alpha, v)$ е новия доверителен интервал, определен в термините на предния разглеждан доверителен интервал $\epsilon_0(N, h, \alpha)$ представен по следния начин:

$$\epsilon_1(N, h, \alpha, v) = 2\epsilon_0^2(N, h, \alpha) \left(1 + \sqrt{1 + \frac{v(\mathbf{w})}{\epsilon_0^2(N, h, \alpha)}} \right). \quad (2.99)$$

Този доверителен интервал зависи от грешките на обучение $v(\mathbf{w})$. При $v(\mathbf{w}) = 0$ приема следващия упростен вид

$$\epsilon_1(N, h, \alpha, 0) = 4\epsilon_0^2(N, h, \alpha). \quad (2.100)$$

Могат да се определят две ограничения на скоростта на равномерна сходимост.

1. В общия случай скоростта на равномерната сходимост удовлетворява следващото неравенство

$$P(\mathbf{w}) \leq v(\mathbf{w}) + \epsilon_1(N, h, \alpha, v),$$

Където $\epsilon_1(N, h, \alpha, v)$ се определя по формула (2.99).

2. При малки (близки до нула) стойности на грешките на обучение $v(\mathbf{w})$ се използва неравенството

$$P(\mathbf{w}) \leq v(\mathbf{w}) + 4\epsilon_0^2(N, h, \alpha).$$

3. При големи стойности на грешките на обучение $v(\mathbf{w})$ близки към единица се изпълнява следното ограничение

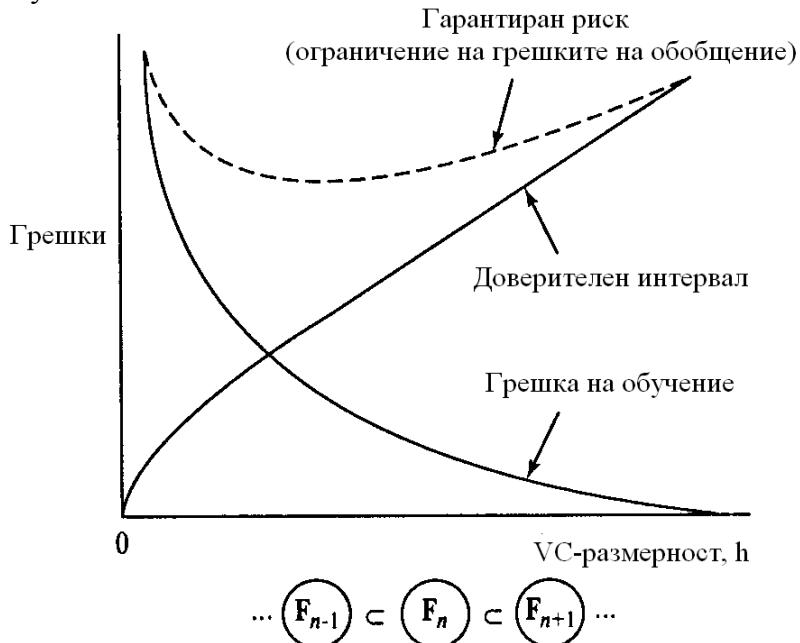
$$P(\mathbf{w}) \leq v(\mathbf{w}) + \epsilon_0(N, h, \alpha).$$

Минимизация на структурния рисък

Под *грешки на обучение* (training error) се разбира честота на направените от машината грешки в течение на обучението на определения вектор на тегалата \mathbf{w} . Аналогично под *грешки на обобщение* (generalization error) се разбира честотата на допуснатите от машината грешки при нейното тестване. Предполагаме, че тестовите данни принадлежат към това семейство, както и данните за обучение. Тези две величини обозначаваме с $v_{train}(\mathbf{w})$ и $v_{gene}(\mathbf{w})$ съответно. Знаем, че $v_{train}(\mathbf{w})$ е същата величина, която в предния раздел беше записана като $v(\mathbf{w})$. Обозначаваме със символа h VC-измерването на семейството от функции на класификация $\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathbf{W}\}$ по отношение на пространството на входните сигнали \mathbf{X} . Тогава в теорията на скоростта на равномерна сходимост може да се твърди, че с вероятност $(1 - \alpha)$ за количеството примери на обучение $N > h$, едновременно за всички функции на класификация $F(\mathbf{x}, \mathbf{w})$ грешката на обобщение $v_{gene}(\mathbf{w})$, има малки стойности на *гарантирания рисък* (guaranteed risk), определена като сума от двойки на конкуриращи се величини

$$v_{guarant}(\mathbf{w}) = v_{train}(\mathbf{w}) + \epsilon_1(N, h, \alpha, v_{train}), \quad (2.101)$$

Където доверителният интервал $\epsilon_1(N, h, \alpha, v_{\text{train}})$ се определя по формула (2.99). За фиксираните числа N на примерите за грешките на обучение, обучението монотонно намалява при увеличаване на VC-измерването h , а доверителният интервал монотонно се увеличава



Фиг. 2.25. Взаимовръзка между грешките на обучение, доверителния интервал и гарантирания риск

Следователно както гарантирани рискове, така и грешките на обобщение имат точка на минимум. Общият случай на това твърдение е показан на фиг. 2.25. До момента достигнатата точка на минимум на задачите за обучение се явява *периодична* (overdetermined) в този смисъл, че обемът на машината h е много малък, за да вмести целия обем на детайлите на обучението. След преминаване на точката на минимум, задачата за обучение се явява *недоопределена* (underdetermined), т.е. обемът на машината е много голям за такъв обем от данни на обучение.

При този начин на решение на задачите за обучение с учител е необходимо обезпечаване на максимална ефективност на обобщение за сметка на преведените в съответствие с обема на машината за допустимото количество данни за обучение. *Методът на минимизация на структурния риск* (method of structural risk minimization) обезпечава индуктивната процедура за постигане на тази цел, в която VC-измерването на обучаващата машина се разглежда като *управляваща* променлива. Ще разгледаме ансамбъла за класификация на образите $\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathbf{W}\}$ и ще определим вложена структура, състояща се от n подобни машини:

$$F_k = \{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in \mathbf{W}_k\}, k = 1, 2, \dots, n, \quad (2.102)$$

Така, че (см. Фиг. 2.25)

$$F_1 \subset F_2 \subset \dots \subset F_n, \quad (2.103)$$

Където символът \subset означава „съдържащи в”. Съответното VC-измерване на отделните класификатори на образите удовлетворява следното условие:

$$h_1 \leq h_2 \leq \dots \leq h_n. \quad (2.104)$$

Затова говорим, че VC-измерването на всички класификатори е крайно. Тогава методът на минимизация на структурния риск може да се изложи по следния начин:

- Минимизация на емпиричния риск (т.е. грешката на обучение) за всеки от класификаторите.
- Определя се класификатор F^* , който има най-малък гарантиран риск. Тази конкретна машина обезпечава компромиса между грешките на обучение (т.е. качество на апроксимирани данни за обучение) и доверителния интервал (т.е. сложност на функцията за апроксимация), които се конкурират помежду си.

Нашата цел е да намерим такава невронна структура, в която намаляването на VC-измерването се достига за сметка на минималната възможност на увеличаване на грешките на обучение.

Принципът на минимизация на структурния риск може да бъде реализиран чрез множество от различни способи. Например, VC-измерването h може да се изменя за сметка на изменянето на количеството скрити неврони. В качеството на примери се разглеждат ансамбли от пълносвързани многослойни мрежи за пряко разпространение, в които количеството на невроните в един от скритите слоеве monotонно нараства. В съответствие с принципа на минимизация на структурния риск наличната мрежа в това множество може да бъде такава, в която гарантирания риск да бъде минимален.

VC-измерването се явява основно понятие не само за принципа на минимизация на структурния риск, но и на моделите за обучение, получили названието *вероятностно-коректни в смисъла на апроксимация* (probably approximatly correct - PAC).

2.15. Вероятностно-коректни в смисъл на апроксимация на моделите на моделите за обучение

Вероятностно-коректни в смисъла на апроксимация (probably approximatly correct) на модела за обучение (PAC) може да се представи като една вероятностна „рамка“ (или среда) за процеса на обучение и обобщение в системата на двоичната класификация. Тя е тясно свързана с принципите на обучение с учител.

Ще определим термините, свързани със средата \mathbf{X} . Множеството от елементите на \mathbf{X} се нарича *понятия* (concept), а всеки нанор на неговите подмножества – *классе понятия* (concept class). Пример на понятията се нарича всеки обект отъ предметната област, заедно с черта от неговия клас. Ако примера се отнася към даденото понятие той се нарича *положителен пример* (positive example). Ако не се отнася към даденото понятие, той се нарича *отрицателен пример* (negative example). Понятия, за които се превеждат примери се наричат *целеви* (traget concept). Последователността от данни за обучение с дължина N на понятията може да се определи по следния начин:

$$\mathbf{T} = \{(\mathbf{x}_i, c(\mathbf{x}_i))\}_{i=1}^N. \quad (2.105)$$

В тази последователност може да се съдържат и повтарящи се примери. Примерите $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ избрани от средата \mathbf{X} са случайни образи, в съответствие с някои фиксиранни, но неизвестни разпределени вероятности. За отношение (2.105) заслужават внимание следните въпроси:

- Целевото понятие $c(\mathbf{x}_i)$ се разглежда като функция на отражение на \mathbf{X} в множеството $\{0, 1\}$. При това предполагаме, че функцията $c(\mathbf{x}_i)$ е неизвестна.
- Предполагаме, че примерите са статистически независими. Това значи, че функциите на плътност са съвместими с вероятностите на два различни примера \mathbf{x}_i и \mathbf{x}_j равни на произведението на съответните функции на плътностна вероятност.

В контекста на терминологията, която използвахме по-рано, среда \mathbf{X} съответства на пространството на входните сигнали на невронните мрежи, а целевото понятие – на очаквания отговор на мрежите.

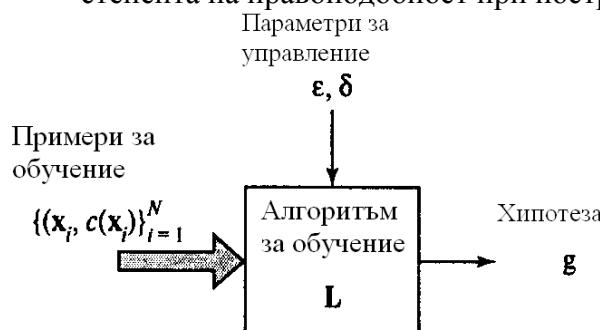
Наборът от понятия, породени от средата \mathbf{X} се наричат *пространство от понятия* \mathbf{B} . Например, пространството от понятия може да съдържа фрази от типа „буква А”, „буква Б” и т.н. В тези понятия може да бъдат закодирани различни спосobi при формирането на множествата от положителни и отрицателни примери. Обучението с учител използва друго множество от понятия. Обучаемата машина представлява множество от функции, които съответстват на определено състояние. Например, машината може да е предназначена за разпознаване на „буква А”, „буква Б” и т.н. Множеството от всички функции (т.е. понятия), определя обучаема машина наречена *пространство от хипотези* (*hypothesis space*) \mathbf{G} . Това пространство може да съвпада или да не съвпада с пространството от понятия \mathbf{B} . При определени гледни точки понятията и хипотезите се явяват аналогии на функцията $f(\mathbf{x})$ и апроксимиращата функция $F(\mathbf{x}, \mathbf{w})$, с която оперирахме в предния раздел.

Предшолагаме, че съществува някое целево понятие $c(\mathbf{x}) \in \mathbf{B}$, приемащо стойности 0 и 1. С това понятие трябва да обучим невронната мрежа с помощта на нейната настройка на множеството от данни T , определено с израза (2.105). Нека $g(\mathbf{x}) \in \mathbf{G}$ е хипотеза, съответстваща на отразения вход на изхода, формиран в резултат на проведеното обучение. Един от способите за постигане на успех в обучението се явява измерената степен на близост на хипотезите $g(\mathbf{x})$ към целевата концепция $c(\mathbf{x})$. Естествено, винаги съществуват грешки, обезпечаващи различните величини. Тези грешки се явяват следствие на това, че опитваме да обучим невронната мрежа на някои функции на основата на ограничена информация за тях. Вероятността за грешки в обучението се определя от израза

$$v_{\text{train}} = P(\mathbf{x} \in \mathbf{X}: g(\mathbf{x}) \neq c(\mathbf{x})). \quad (2.106)$$

Разпределението на вероятностите в този пример трябва да бъде такова, както при формирането на примерите. Целта на обучението РАС е минимизация на стойностите на v_{train} . Предметната област достъпна за алгоритъма на обучение се определя от размера N на обучаващото множество T . Алгоритъмът на обучение има два параметъра на обучение:

- *Параметър на грешките* (error parameter) $\epsilon \in (0, 1]$. Този параметър дава величината на грешките, при които апроксимирането на целевото понятие $c(\mathbf{x})$ на хипотезата $g(\mathbf{x})$ се счита за удовлетворително.
- *Параметър за доверие* (confidence parameter) $\delta \in (0, 1]$. Този параметър задава степента на правоподобност при построяване на „хипотези” на апроксимация.



Фиг.2.26. Блок диаграма илюстрираща моделът на обучение РАС

Можем формално да определим моделът за обучение РАС.

Нека \mathbf{B} – клас от понятия за средата X . Счита се, че класа \mathbf{B} се явява РАС-обучение, ако съществува алгоритъм L , имащ следните свойства. За всяко целево понятие $c \in \mathbf{B}$, на всяко разпределение на вероятностите на X и за всяко $0 < \epsilon < 1/2$ и $0 < \delta < 1/2$ при използвания алгоритъм L за множествата от примери на обучение $T = \{(x_i, c(x_i))\}_{i=1}^N$ с вероятност не по-лоша от $(1 - \delta)$ резултатът от алгоритъма за обучение L ще бъде хипотезата g с грешки при обучение $v_{train} < \epsilon$. Тази вероятност се получава за всички подмножества на множеството T и при всички вътрешни рандомизации, които може да съществуват в алгоритъма за обучение L . При това размерът на обучаващото множество N не трябва да надвишава стойността на никоя от функциите от δ и ϵ .

С други думи, ако размерът N на обучаващото множество T е достатъчно голям, то съществува вероятност, че в резултат на обучението на мрежата от този избор на примери на отражение на входа и изхода, реализирани от мрежата, ще бъдат „приблизително коректни“.

Сложност на обучаващото множество

При използването на теорията РАС-обучението на практика възниква въпрос за *сложността на обучаващото множество* (sample complexity). Той може да се формулира по следния начин: колко случайни параметъра е нужно да предостави алгоритъма за обучение, за да бъде неговата информация достатъчна за „изучаване“ на неизвестното понятие c , избрано от класа понятия \mathbf{B} , или колко голямо трябва да бъде обучаващото множество T ?

Въпросът за сложността на обучаващото множество е тясно свързан с VC-измерването. Въвеждаме понятието *съгласуваност* (consistence). Нека $T = \{(x_i, d_i)\}_{i=1}^N$ е множество от маркирани примери, в които теглото $x_i \in \mathbf{X}$ и теглото $d_i \in \{0, 1\}$. Тогава понятието c се нарича съгласувано с набора от примери T (и наборът T се нарича съгласуван с c), ако за всяко $1 \leq i \leq N$ се изпълнява равенството $c(x_i) = d_i$. В концепцията на РАС-обучението критично се явява не размерът на множеството изчисления на невронните мрежи на функциите на отражение на входа в изход, а VC-измерването на мрежите.

Разглеждаме невронна мрежа с крайно VC-измерване $h \geq 1$.

- 1) Всеки състоятелен алгоритъм за обучение на тази невронна мрежа е алгоритъм за обучение на РАС.
- 2) Съществува такава константа K , която с достатъчна точност на обучаващото множество T за всеки алгоритъм се явява

$$N = \frac{K}{\epsilon} \left(h \log \left(\frac{1}{\epsilon} \right) + \log \left(\frac{1}{\delta} \right) \right), \quad (2.107)$$

Където ϵ - параметър на грешките; δ - параметър на доверие.

Този резултат може да се използва в обучението с учител, независимо от типа на алгоритъма за обучение и разпределението на вероятностите на маркираните примери. При сравнението на този резултат, получен на основата на изчисленията на VC-измерването, с експериментални данни се получават различни стойности. Това не е удивително за ткаова разъгласувано отразяване на *независимото от разпределението и пессимистичния характер* (distribution-free, worst-case) на теоретичната оценка.

Изчислителна сложност

Един от въпросите, на който трябва да се обърне внимание при разглеждането на концепциите на обучението PAC е изчислителната сложност. Този въпрос касае изчислителната ефективност на алгоритъма за обучение. Понятието *изчислителна сложност* (computational complexity) е свързано с пессимистичните оценки във времето, необходими за обучението на невронните мрежи (обучаема машина) на множеството маркирани примери с мощност N .

На практика работата на алгоритъма зависи от скоростта на използваните величини. От теоретична гледна точка е необходимо такова определяне на времето за обучение, което няма да зависи от конкретните устройства, използвани за обработка на информацията. Времето за обучение (и съответната изчислителна сложност) се измерва в термините на количеството операции (сложност, умножение и съхранение), необходими за използваните изчисления.

При изчисляване на сложността на алгоритъма за обучение е необходимо да се знае, как тя зависи от размерността на примерите на обучение m (т.е. от размера на входния слой на обучение на невронните мрежи). В този контекст алгоритъмът се счита за изчислително *ефективен* (efficient), ако времето за работа е пропорционално на $O(m^r)$, където $r \geq 1$. В този случай говорим, че времето за обучение полиноминално зависи от m (polynomially with m), а самият алгоритъм се нарича *алгоритъм с полиноминално време на изпълнение* (polynomial time algorithm). Задачите за обучение, основани на алгоритъма с полиноминално време на изпълнение се считат за „прости”.

Още един параметър, на който трябва да се обърне внимание е параметъра за грешки ϵ . При сложността на обучаващите множества, параметъра за грешки се явява фиксиран, но произволен; а при оценката на изчислителната сложност на алгоритъма за обучение е необходимо да се знае, как тя зависи от този параметър. Интуитивно следва, че при използване на параметъра ϵ задачата за обучение се усложнява. Следователно трябва да се достигне до някое състояние, което обезпечава вероятностно-коректния в смисъл на апроксимация изход. За обезпечаване на ефективността на изчисленията съответстващото състояние се достига за полиноминално време $1/\epsilon$.

Обединявайки тези разсъждения можем да сформираме следното твърдение:
Алгоритъмът за обучение се явява изчислително ефективен по параметъра за грешки ϵ , размерността на примерите за обучение m и размерът на обучаващото множество N , ако времето за изпълнение се явява полиноминално по N , и съществува такава стойност $N_0(\delta, \epsilon)$, достатъчна за PAC-обучението, при която алгоритъмът се явява полиноминален по m и ϵ^{-1} .

2.2 Мрежови архитектури

В предишната секция ние обсъдихме свойствата на основните обработващи възли в изкуствените невронни мрежи. В тази секция се наблюга върху вида на връзките между възлите и предаваните данни:

В зависимост от вида на връзките между невроните разграничаване главно:

- *Прави мрежи*, в които данните се предават строго от входните към изходните възли. Мрежата може да е многослойна, но не съществуват никакви обратни връзки, т.е. няма връзки, излизящи от изходна невроните от един слой и завършващи на входа на неврони от същия или предишни слоеве.
- *Рекурентни мрежи*, които включват и обратни връзки. Противоположно на правите мрежи динамичните свойства на тези структури са важни. В някои случаи, активиращите стойности на възлите са в процес на релаксация, при който мрежата преминава в стабилно състояние, в което не настъпват повече значителни промени. В някои приложения, промените на активността на изходните възли са значими и динамичното поведение определя изхода на мрежата.

Класически пример за права мрежа е **перцептронът** и **adaline**. Примери за рекурентни мрежи са представени от Андерсън (Anderson), Кохонен (Kohonen) и Хопфийлд (Hopfield).

2.3 Обучение на изкуствените невронни мрежи

Невронните мрежи могат да бъдат конфигурирани така че приложението на множество от входни данни (било то пряко или чрез процеси на релаксация) да доведе до желани изходни стойности. Съществуват различни методи за установяване на силата на връзките. Един от тях е да се положат теглата явно, като се използват априорни знания. Друг начин е да се обучи НМ, като ѝ се представят обучаващи примери и теглата се нагласят според някакво правило на обучение.

2.3.1 Обучаващи парадигми

Процесите на обучение могат да бъдат класифицирани в две различни категории. Те са:

- *Обучение с учител* или *Асоциативно обучение*, при което мрежата се обучава, като ѝ се предоставят входният и съответстващия изходен образец. Тези входни изходни двойки могат да се представят от външен учител или от система, която включва мрежата (самоучител).
- *Обучение без учител* или *самоорганизация*, при което изходните неврони се обучават да отговарят на кълстери от входните образци. Тези парадигми предполагат, че системата открива най-характерните черти на входната популация. За разлика от обучението с учител, априори няма никакво множество от категории, към които да се класифицират входните образи; по-скоро системата сама трябва да си изработи свое собствено представяне на входните стимули.

2.3.2 Модифициране на свързаност

Двете обучаващи парадигми описани по горе, изискват настройване на теглата на връзките между невроните според някакво модифициращи правило. Всъщност всички обучаващи правила за модели на този тип могат да бъдат разглеждани като вариант на обучаващото правило на Хеб, предложено от Хеб в неговата класическа книга „Organization of Behaviour“ (Организация на характера на изменение) през 1949 г. Основната идея се състои в това, че ако два възела j и k са едновременно активни, връзката между тях трябва да бъде усилена. Ако j получава вход от k , най-опростения вериант на правилото на Хеб е теглото ω_{jk} да се промени чрез формулата:

$$\Delta\omega_{jk} = \gamma y_j y_k, \quad (2.7)$$

където γ е положителен коефициент на пропорционалност, представящ степента на обучение. Друго общо правило използва не действителната активност на възел k , а разликата между действителната и желаната активност за нагласяване на теглата:

$$\Delta\omega_{jk} = \gamma y_j (d_k - y_k), \quad (2.7)$$

където d_k е желаната активност, получена чрез учител. Това правило се нарича *правило на Уидроу-Хоф* (*Windrow-Hoff rule*) или *делта правило*.

През последните години бяха публикувани различни негови варианти (някои от тях доста екзотични).

2.4 Означения и терминология

През дългогодишните изследвания по различни научни направления се изнамериха огромен брой термини приложими в областта на невронните мрежи. От наша гледна точка като информатики, ни е позволено да се придържаме към терминологично подмножество, което по-малко съответства на терминологията вдъхновена от биологията, докато не възникнат конфликти. Използваните от нас конвенции се представени по-долу.

2.4.1 Означения

Ще използваме следните означения в нашите формули. Да отбележим, че не всички символи са сmisлени за повечето мрежи, и че в някои случаи долните или горните индекси могат да бъдат пропуснати (т.е. не винаги p е необходимо) или добавени (т.е. от друга страна векторите могат, както нотацията по-долу, да имат индекси), където е необходимо. Векторите са означени с удебелен ненаклонен шрифт:

j, k, \dots възел j, k, \dots ;

i входен възел;

h скрит възел;

o изходен възел;

χ^p p -ти вектор входен-образец;

x_j^p j -тия елемент на p -тия вектор входен-образец;

s^p входът на множеството от неврони, когато вектора входен-образец p е ограничен (т.е., представлящ за мрежата); често: входът на мрежата е ограничен чрез вектора входен-образец p

d^p желания изход на мрежата, когато вектора входен-образец p е подаден на входа на мрежата;

d_j^p j -тия елемент на желания изход на мрежата, когато вектора входен-образец p е подаден на входа на мрежата;

y^p стойност на актививност на мрежата, когато вектора входен-образец p е подаден на входа на мрежата;

y_j^p активиращата стойност на елемента j на мрежата, когато вектора входен-образец p е подаден на входа на мрежата;

- W матрица на теглата на връзките;
- w_j теглата на всички връзки влизащи във възел j ;
- ω_{jk} теглото на връзката от възел j във възел k ;
- Φ_j функция на активност или предавателна функция асоциирана с възел j ;
- γ_{jk} степента на обучени свързана с тегло ω_{jk} ;
- θ отклонение на възлите;
- θ_j отклонение на входа на възел j ;
- U_j прага на възел j в Φ_j ;
- E^p грешката в изхода на мрежата, когато вектора входен-образец p е вход;
- ϵ енергия на мрежата;

2.4.2 Терминология

Изходът в сравнение с активността на възлите. Тъй като няма необходимост това да се прави в противен случай, ние разглеждаме изхода и активиращата стойност на възела като едно и също нещо. Така че, изходът на всеки неврон се равнява на неговата стойност на активност.

Отклонение, отместване, праг. Всички тези термини се отнасят за една и съща константа (т.е., независима от входа на мрежа, но определена от правилото за правило), която е вход на възела. Те могат да се използват равнозначно, макар, че последните два термина са често представят като свойства на функцията на активност. Освен това, обикновено този външен вход е реализиран (и може да бъде записан) като тегло на възел със стойност на активност равна на 1.

Брой на слоевете. В правата мрежа, входовете не извършват изчисление и следователно тяхното ниво не е от значение. Такава мрежа с едно входно ниво, едно скрито ниво и едно изходно ниво се причислява към двусловните мрежи. Тази конвенция е широко известна макар, че все още не се използва повсеместно.

Представяне в сравнение с обучение. Когато някой използва невронна мрежа трябва да различи два проблема които влияят върху работата на системата. Първият е *силата на представяне* на мрежа, вторият е *обучаващия алгоритъм*.

Силата на представяне на невронната мрежа се отнася за способността на невронна мрежа да представи желана функция. Тъй като невронната мрежа е построена чрез набор от стандартни функции, в повечето случаи мрежа само ще бъде *приближена* до желаната функция, и дори при оптималното множество от тегла грешката при апроксимацията няма да бъде нулева.

Вторият проблем е обучаващия алгоритъм. Ако в една мрежата съществува оптимален набор от тегла, то има ли процедура, която (итеративно) да намира този набор от тегла?

Глъвка 2 Перцептрон и Adaline

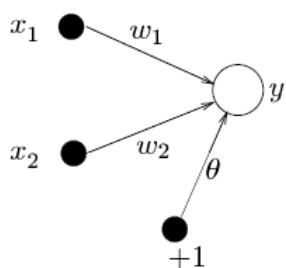
Тук ще разгледаме еднослойни невронни мрежи, ще включим някои от класическите подходи при изчисляването на невроните и задачи свързани с тяхното обучение. В първата част на това изложение ще обсъдим представителната мощност на мрежите от един слой и техните обучаващи алгоритми и ще дадем примери за използване на мрежите. Във втората част изложението ще обсъдим представителните ограничения на еднослойните мрежи.

В първата част ще бъдат описани двата „классически“ подхода: *Перцептронът*, предложен от Франк Розенблат (Frank Rosenblatt) през 50-те години на XX век и простата мрежа Adaline, представена в началото на 60-те години на XX век от Уидроу (Widrow) и Хоф (Hoff)

Перцептронът представлява мрежа, при която обработващите елементи са организирани в слоеве с насочени връзки между невроните от един слой и тези от следващия.

3.1 Мрежи с прагови функции на активност

Еднослойните мрежи съдържат един или повече изходни неврони o , всеки от които е свързан с тегловия коефициент ω_o . В най-простия случай мрежата има само два входа и един изход, като рисунката на *фиг. 3.1* (нарочно пропускаме индекса o). Входът на неврона е сумата от теглата на входовете плюс отклонението. Изходът на мрежата се



Фигура 3.1: Мрежа от един слой с един изход и два входа

формира чрез активността на изходния неврон, която е функция от входовете:

$$y = F \left(\sum_{i=1}^2 \omega_i x_i + \theta \right), \quad (3.1)$$

Тази функция определя състоянието на активност на неврона. Активиращата функция F може да бъде линейна, таке че имаме линейна или нелинейна мрежа. В тази секция ще разгледаме праговата (или sgn) функция:

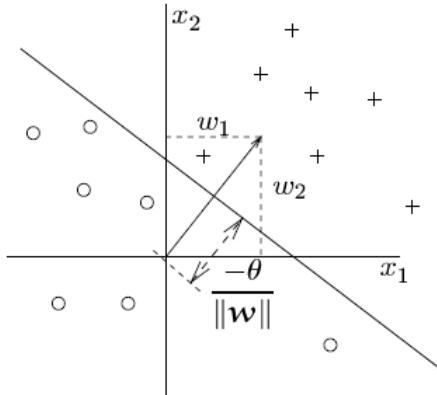
$$F(s) = \begin{cases} 1 & \text{ако } s > 0 \\ -1 & \text{в противен случай} \end{cases}. \quad (3.2)$$

Изходът на мрежата, който е или 1 или -1 , зависи от входа. Сега мрежата може да се използва за решаването на *классическата задача*: може ли да се реши дали входния образец се отнася към един от двата класа. Ако общия вход е положителен, моделът ще се отнесе към класа $+1$, ако общия вход е отрицателен – към класа -1 . В този случай разделителя между двата класа е права линия, зададена чрез равенството:

$$\omega_1 x_1 + \omega_2 x_2 + \theta = 0. \quad (3.3)$$

Еднослойните мрежите представляват *линейно отделима функция*.

Геометричното представяне на линейните прагови невронни мрежи е даден на *фиг. 3.2*.



Фигура 3.2: Геометрично представяне на функцията на отделимост и на теглата.

Равенството (3.3) може да се запише като

$$x_2 = -\frac{\omega_1}{\omega_2} x_1 - \frac{\theta}{\omega_2}. \quad (3.4)$$

и виждаме че теглата определят наклона на правата, а отклонението определя „отместването”, т.е. колко далече е правата от началната точка. Да обърнем внимание

на факта, че теглата могат да бъдат отбелязани във входното пространство: тегловия вектор винаги е перпендикулярен на функцията на отделимост.

След като е представена силата на връзките на еднослойните мрежи с линейно прагови възли, преминаваме към втория проблем: как да обучим теглата и влиянието им върху мрежата? Ще опишем два метода на обучение на тези видове мрежи: обучаващата процедура на „перцепtronът“ и „делта“ или „LMS“ правилото. Двата метода итеративно определят теглата. Обучаващото правило е представено за мрежата. За всяко тегло новата стойност се изчислява, чрез добавяне на корекции към старата стойност. Прагът се актуализира по същия начин:

$$\omega_i(t+1) = \omega_i(t) + \Delta \omega_i(t). \quad (3.5)$$

$$\theta(t+1) = \theta(t) + \Delta \theta(t). \quad (3.6)$$

Сега задачата за обучението може да се формулирана по следния начина: в какъв ред да се изчислят $\Delta \omega_i(t)$ и $\Delta \theta(t)$, за да се класифицират правилно моделите на обучение?

3.2 Обучаващо правило на перцепtron и теорема на сходство

Да предположим, че имаме множество от обучаващи образци съгласно входния вектор χ и очакваме изхода $d(\chi)$. За класифициращата задача $d(\chi)$ обикновено е $+1$ или -1 . Обучаващото правило на перцептрона е много просто и може да бъде изразено както следва:

1. Започва се с произволно избрани тегла за връзките;
2. Избира се входен вектор χ от множество от трениращи образци;
3. Ако $y \neq d(\chi)$ (перцепtronът дава некоректни отговори), модифицират се всички връзки ω_i съгласно: $\Delta \omega_i = d(\chi)x_i$;
4. Преминава се към стъпка 2.

Трябва да отбележим, че процедурата е много сходна с правилото на Хеб; разликата се състои само в това, че когато мрежата отговаря правилно, теглата на връзките не се модифицират. Освен това модифициране на теглата, трябва също да се промени и прага θ . Това θ се разглежда като връзката с тегло ω_0 , между изходния неврон и „фиктивния“

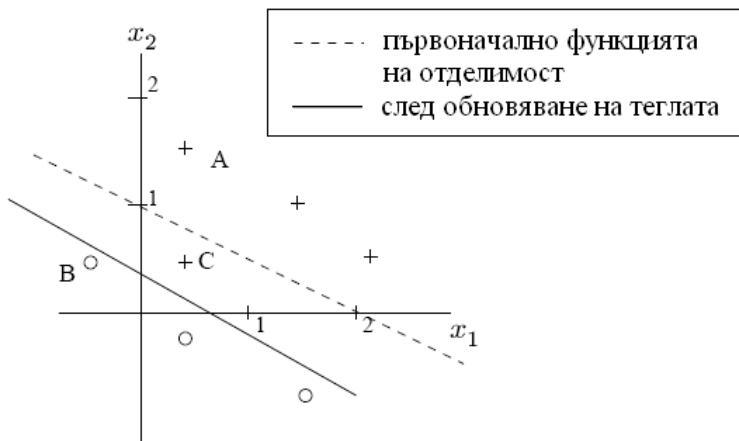
предикатен възел, който винаги е: $x_0 = 1$. За даден обучаващо правило на перцептрон, което съответства на условията посочени по-горе, този праг се променя съгласно:

$$\square \theta = \begin{cases} 0 & \text{ако перцептронът отговаря коректно;} \\ d(x) & \text{в противен случай.} \end{cases} . \quad (3.7)$$

3.2.1 Пример за обучаващо правило на перцептрон

Перцептронът се инициализиран със следните тегла: $\omega_1 = 1$, $\omega_2 = 2$, $\theta = -2$. Обучаващото правило на перцептрана е използвано за правилно обучение на отделящата функция на набора от образи, представени на *фиг. 3.3*. Първия образ А, със стойности $\chi = (0.5, 1.5)$ и целева стойност на мрежата е $d(\chi) = +1$. Чрез равенство (3.1) може да се пресметне, че изхода на мрежата е $+1$, така че теглата не се регулират. Същото се отнася и за точка В със стойност $\chi = (-0.5, 0.5)$ и целевата стойност $d(\chi) = -1$; изхода на мрежата е отрицателен, така че не се променя. При представянето на точка С със стойности $\chi = (0.5, 0.5)$ изходът на мрежата ще бъде -1 , докато целевата функция $d(\chi) = +1$. Съгласно обучаващото правило на перцептрана, теглата се променят по следния начин: $\square \omega_1 = 0.5$, $\square \omega_2 = 0.5$, $\square \theta = 1$. Новите тегла сега са: $\omega_1 = 1.5$, $\omega_2 = 2.5$, $\theta = -1$, и образа С е правилно класифициран.

Във *фиг. 3.3* е представена функцията на отделимост преди и след обновяване на теглата.



Фигура 3.3: Функция на отделимост преди и след обновяване на теглата

3.2.2 Теорема на сходимост

За обучаващото правило на перцептрана съществува теорема на сходимост, чиито изложение е следното:

Теорема 1. Ако съществува множество от тегла на връзките w^* , което е способно да изпълнява трансформацията $y = d(\chi)$, обучаващото правило на перцепtronът ще бъде приближение към някое решение (което може да е или да не е същото като w^*) в краен брой стъпки за всеки начален избор на тегла.

Доказателство: Даден е фактът, че дължината на векторите w^* не е от значение (в следствие на оператора sgn) и ще вземе $\|w^*\|=1$. Тъй като w^* е правилно решение, стойността $|w^* \cdot \chi|$, където не е отбелязано точка или вътрешния продукт, тогава ще е по-голямо от 0 или: съществува $\delta > 0$, такова че $|w^* \cdot \chi| > \delta$ за всички входове χ ¹. Сега да дефинираме $\cos \alpha \equiv w \cdot w^* / \|w\|$. Когато съгласно обучаващото правило на перцептрана, теглата на връзките се променят чрез зададения вход χ , знаем че $\Delta w = d(\chi) \chi$ и след промяната теглото е $w' = w + \square w$. От това следва, че:

$$\begin{aligned} w' \cdot w &= w \cdot w^* + d(\chi) \cdot w^* \cdot \chi \\ &= w \cdot w^* + \text{sgn}(w^* \cdot \chi) \cdot w^* \cdot \chi \\ &> w \cdot w^* + \delta \end{aligned}$$

$$\begin{aligned} \|w'\|^2 &= \|w + d(\chi) \cdot \chi\|^2 \\ &= w^2 + 2d(\chi) \cdot w \cdot \chi + \\ &< w^2 + \chi^2 && (\text{защото } d(\chi) = -\text{sgn}[w \cdot \chi]) \\ &= w^2 + M \end{aligned}$$

След t промени получаваме:

$$\begin{aligned} w(t) \cdot w^* &> w \cdot w^* + t\delta \\ \|w(t)\|^2 &< w^2 + tM \end{aligned}$$

така че

$$\begin{aligned} \cos \alpha(t) &= \frac{w^* \cdot w(t)}{\|w(t)\|} \\ &= \frac{w^* \cdot w + t\delta}{\sqrt{w^2 + tM}} \end{aligned}$$

¹ Технически не е необходимо всяко ω^* да бъде истина; всъщност за всяко ω^* , получено без класификация би могло $\omega^* \cdot \chi$ да бъде равно на 0 (виж дефиницията на Φ). Обаче, за всяка ненулева величина може да бъде намерено друго ω^* .

От това следва, че

$$\lim_{t \rightarrow \infty} \cos \alpha(t) = \lim_{t \rightarrow \infty} \frac{\delta}{\sqrt{M}} \sqrt{t} = \infty,$$

докато по дефиниция $\cos \alpha \leq 1$!

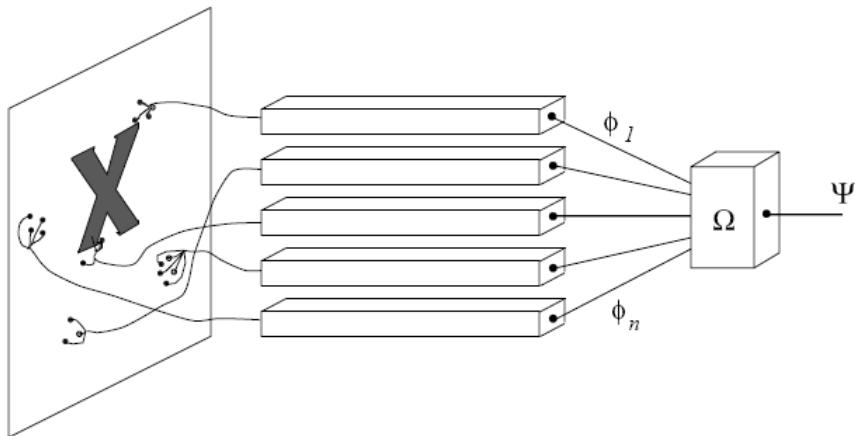
Заключението е, че трябва t_{max} да е ограничено отгоре за t . Системата променя тези връзки само стойностите ограничени във времето. С други думи: след максимално t_{max} за правилното изменение на теглата на перцептрана е изпълнението на последователността на стъпките. t_{max} ще бъде достигнато, когато $\cos \alpha = 1$. Ако започнем с връзки $w = 0$,

$$t_{max} = \frac{M}{\delta^2} \quad (3.8)$$

□

3.2.3 Първоначалния перцепtron

Перцептранът, предложен от Розенблант е до голяма степен по комплексен от еднослойните мрежи с прагова функция на активност. Неговата най-проста форма се състои от N -елемента на входния слой („ретина“) с поддръжка в слой на M „асоциативни“, „маскирани“ или „предикатни“ възли ϕ_h , и един изходен възел. Целта на работата на перцептрана е да научи зададената му трансформация $d : \{-1,1\}^N \rightarrow \{-1,1\}$ използвайки обучаващите образци съответно с вход χ и с изход $y = d(\chi)$. В оригиналната дефиниция, действието на предикатните възли може да бъде всяка функция ϕ_h на входния слой χ , но само обучаващата процедура регулира връзките на изходния слой. Причината за това е, че не е намерена никаква рецепта за регулирането на връзката между χ и ϕ_h . В зависимост от функцията ϕ_h , перцептраните могат да бъдат групирани в различни *семейства*. Мински и Пепърт описват броя и свойствата на тези семейства. Изходния възел на перцептрана е линеен прагов елемент. Розенблат през 1959 година доказва забележителна теорема относно обучението на перцептрана и в началото на 60-те години на XX век перцептраните предизвикват голяма доза оптимизъм и интерес. Първоначалната еуфория по-късно обаче е заменена от отчаяние, след публикуваната през 1969 година книга „Перцептрони“ на Мински и Пепърт. В нея те анализират задълбочено перцептрана и доказват че има строги ограничения, които перцептраните трябва да спазват.



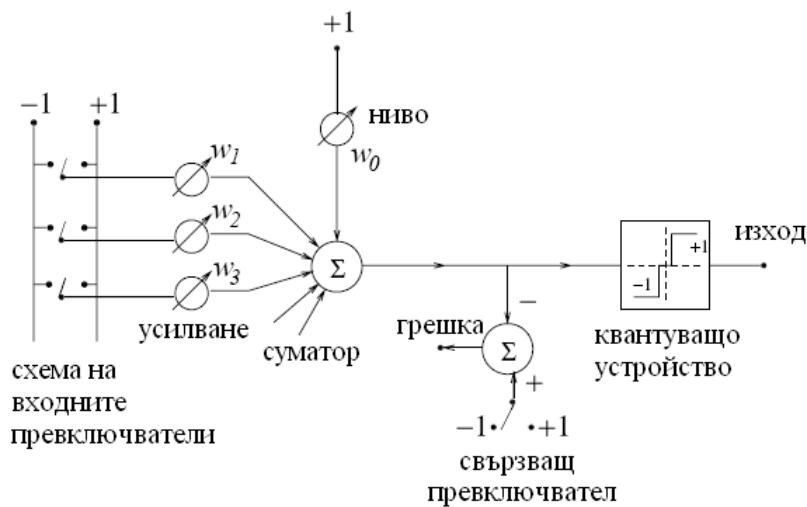
Фигура 3.4: Перцепtron

3.3 Адаптивен линеен елемент (The adaptive linear element – Adaline)

Важно обобщение на обучаващия алгоритъм на перцептрана е представено от Уиндоу и Хоф като „най-малка средноквадратична” (Least Mean Square - LMS) обучаваща процедура, и е позната още като *делта правило*. Основната функционална разлика с обучаващото правило на перцептрана е начина на изход от системата използван в обучаващото правило. Обучаващото правило на перцептрана използва изходът на праговата функция (или -1 , или $+1$) за обучение. Делта правилото използва изхода на мрежата без по-нататъшно отразяване на изходните стойности -1 , или $+1$.

Обучаващото правило се прилага при „адаптивния линеен елемент”, познат като *Adaline*², разработен от Уидроу и Хоф. В простата физическа реализация (фиг. 3.5), това устройство се състои от потенциометър (от набор управлявани резистори) свързани към ел. верига, която сумира текущите напрежения на входните сигнали. Обикновено централния блок, суматорът, следван също така от квантуващото устройство, което изпълнява или -1 , или $+1$, в зависимост от поляритета на сумата.

² ADALINE първоначално е означавало ADaptive LInear NEuron (Адаптивен линеен неврон), но когато изкуствените неврони ставали все по-малко и по-малко известни, този акроним се променил на ADaptive LINear Element (Адаптивен линеен елемент).



Фигура 3.5: Адаптивен линеен елемент (Adaline)

Въпреки че тук се разглежда пример на адаптивния процес в случай когато има само един изход, той може да бъде определен за система с много паралелни изходи и непосредствено изпълнен чрез множество възли от гореуказания вид.

Ако електрическата проводимост на входовете са означени с ω_i , $i = 0, 1, \dots, n$ и входните и изходните сигнали чрез x_i и y , респективно, в такъв случай изходът на централния блок се дефинира чрез

$$y = \sum_{i=1}^n \omega_i x_i + \theta \quad (3.9)$$

където $\theta \equiv \omega_0$. Целта на това устройство е да получи дадена стойност $y = d^p$ на негови изход когато на входа е приложено множество от стойности x_i^p , $i = 1, 2, \dots, n$.

Задачата се определя от коефициентите ω_i , $i = 1, 2, \dots, n$, по такъв начин входно-изходния резултат да е правилен за голяма брой произволно избрани сигнали. Ако не е възможно точно отразяване, средната стойност на грешката трябва да бъде намалена, например в значението на най-малки квадрати. Адаптивната операция означава, че съществува механизъм, чрез който ω_i може да бъде регулирано, обикновено итеративно, за достигане на коректните стойности. За Adaline, Уидроу въвежда делта правило за регулиране на теглата. Това правило ще бъде обсъдено в секция 3.4.

3.4 Мрежа с линейна активираща функция: делта правило

За еднослойната мрежа с изходен възел с линейна функция на активност на изхода се задава просто чрез

$$y = \sum_j \omega_j x_j + \theta \quad (3.10)$$

Такава проста изчислителна мрежа е способна да представи линейната зависимост между стойностите на изходните и на входните възли. Чрез праговата стойност на изхода, класификаторът може да бъдете построена (например както при Adaline на Уидроу), но тук се концентрираме върху линейната зависимост и използването на мрежата за задача на апроксимиране на функцията. В максималното измерение на входното пространство мрежата представлява (хипер)равнина и така ще бъде ясно, че множеството от изходни възли може да бъде дефинирано.

Да предположим, че искаме да тренираме мрежа, която е станала хиперравнина, което е възможно за множеството от обучаващи образци, състоящи се от входните стойности x^p и желаните (или целевите) изходни стойности d^p . За всеки даден входен образец, изходът на мрежата се определя от целевата стойност d^p чрез $(d^p - y^p)$, където y^p е текущия изход на схемата. Сега делта правилото използва оценъчна функция или функция на грешката основана на тези различия при регулиране на теглата.

Функцията на грешката, означена чрез наименованието най-малкото средно квадратично отклонение, е сумата от квадратите на грешките. Това е общата грешка E , дефиниран по следния начин:

$$E = \sum_p E^p = \frac{1}{2} \sum_p (d^p - y^p)^2, \quad (3.11)$$

където индекса p се изменя над множеството от входни образи и E^p представлява грешката на образца p . Процедурата за най-малко средно квадратично търси стойности на всички тегла, които минимизират функцията на грешката чрез метода наречен *спускане по градиентта*. Идеята се състои в това да се направи промяна в теглата пропорционални на отрицателната стойност на производната на грешката като измерена на текущия образец по отношение на всяко тегло:

$$\square_p \omega_j = -\gamma \frac{\partial E^p}{\partial \omega_j}, \quad (3.12)$$

където γ е константа на пропорционалността. Производната е

$$\frac{\partial E^p}{\partial \omega_j} = \frac{\partial E^p}{\partial y^p} \frac{\partial y^p}{\partial \omega_j}. \quad (3.13)$$

Поради линейността на възлите (равенство (3.10)),

$$\frac{\partial y^p}{\partial \omega_j} = x_j \quad (3.14)$$

и

$$\frac{\partial E^p}{\partial y^p} = - (d^p - y^p) \quad (3.15)$$

такова, че

$$\square_p \omega_j = y \delta^p x_j \quad (3.16)$$

където $\delta^p = d^p - y^p$ е разликата между очаквания и действителния изход за образец p .

Делта правилото променя подходящо теглата за очакваните и действителните изходи за който и да е поляритет и за непрекъснатите и двоични входни и изходни възли.

3.5 Задачата за изключващото или (Exclusive-OR)

В предишните секции обсъдихме два обучаващи алгоритъма за еднослойните мрежи, но не разглеждахме ограниченията при *представянето* на тези мрежи.

x_0	x_1	d
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

Таблица 3.1: Таблица на истинност на XOR

Един от по-обезкуражителните резултати на Мински (Minsky) и Пепърт (Papert) показва, че еднослойния перцепtron не може да пресметне някои прости функции и най-простия пример за това е XOR (изключващото или). Таблица 3.1 показва желаната връзка между входовете и изходите за тази функция.

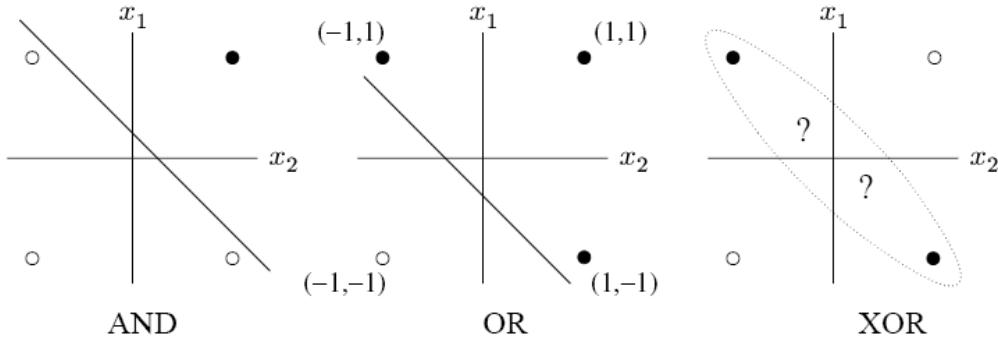
В проста изчислителна мрежа с два входа и един изход, изобразена на фиг. 3.1, входната мрежа е равна на:

$$s = \omega_1 x_1 + \omega_2 x_2 + \theta \quad (3.17)$$

Според равенство (3.1), изходът на перцептрона е нула, когато s е отрицателно или равно на едно, когато s е положително. Във *фиг. 3.6* е дадено геометричното представяне на входната област от стойности. За константата θ , изходът на перцептрона е равен на едно от едната страна на разделящата прива, която се дефинира чрез:

$$\omega_1 x_1 + \omega_2 x_2 = -\theta \quad (3.18)$$

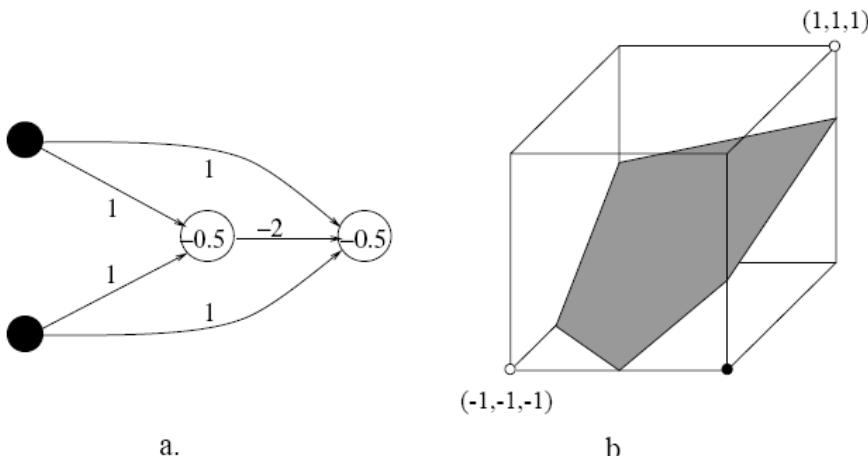
и равно на нула от другата страна на тази линия.



Фигура 3.6: Геометрично представяне на входното пространство

За да се убедим, че такова решение не може да бъде намерено, ще разгледаме *фиг. 3.6*. Входното пространство се състои от четири точки: две тъмни кръгчета $(1, -1)$ и $(-1, 1)$ не могат да бъдат разделени от право ребро от двете светли кръгчета на $(-1, -1)$ и $(1, 1)$. Очевидно е задаването на въпроса: Как може този проблем да бъде преодолян? Мински и Пепърт доказват в тяхната книга, че при двоични входове, всяка промяна може да бъде изпълнена чрез добавяне на слой от твърдения, които са свързани с всички входове.

Специфичната XOR задача (изключващо или) ще покажем геометрично, че при въвеждането на *скрити възли*, по този начин мрежата се разширява към *многослоен перцепtron*, задачата може да бъде решена. *Фиг. 3.7a* показва, че четирите входни точки сега са поставени в 3-мерно пространство, дефинирано чрез две входни точки плюс един



Фигура 3.7: Решение на задачата XOR

- a) Перцептронът на фиг. 3.1 с допълнителен скрит възел. С означените стойности на теглата ω_{ij} (до свързващите линии) и прагове θ_i (в кръгчетата) този перцепtron решава задачата XOR.
- b) Той е осъществен чрез начертаването на четири точки от фиг. 3.6 върху четирите точки тук; очевидно разделянето сега (чрез линейно разнообразие) в необходимите групи е възможно

скрит възел. Сега тези четири точки лесно се разделят чрез линейно многообразие (равнина) в две групи, както искахме. Този прост пример демонстрира, че добавянето на скрити възли увеличава класа от проблеми, които са разрешими за правите мрежи като перцептрона. Обаче, чрез това пораждане на базови архитектури, се получават сериозни загуби: не съществуват други обучаващи правила за определяне на оптимални тегла.

3.6 Многослойните перцептрони могат да правят всичко

В предишната секция показвахме, че чрез добавянето на допълнителни скрити възли, XOR задачата може да бъде решена. За бинарни възли, някои може да докаже че тази архитектура е способна да изпълнява всяка трансформация при правилни връзки и тегла. Най-примитивен е следния.

За дадена трансформация $y = d(x)$, можем да разделим множеството то всички възможни вектори в два класа:

$$X^+ = \{\chi | d(\chi) = 1\} \quad \text{и} \quad X^- = \{\chi | d(\chi) = -1\} \quad (3.19)$$

Докато има N входа, общия брой от възможни входни вектори χ е 2^N . За всяко $\chi^p \in X^+$ скрития възел h , чиито активация y_h е 1, може да бъде определен, тогава и само тогава, когато специфичния образец p е представен на входа: можем да изберем неговите тегла ω_{ih} равни на специфичния образец χ^p и отклонението θ_h равно на $1 - N$, така че

$$y_h^p = \operatorname{sgn}\left(\sum_i \omega_{ih} x_i^p - N + \frac{1}{2}\right) \quad (3.20)$$

е равно на 1 само при $\chi^p = w_h$. По същия начин, теглата на изходният неврон могат да бъдат избрани така че изходът да бъде единствен, щом като за някой от M предикатни неврони е изпълнено:

$$y_0^p = \operatorname{sgn}\left(\sum_{h=1}^M y_h + M - \frac{1}{2}\right) \quad (3.21)$$

Този перцептрон ще даде $y_0 = 1$ само ако $\chi \in X^+$: това изпълнява желаното означение. Проблемът е големия набор от предикатни възли, който е равен на броя на образците в X^+ , чиито максимална стойност е 2^N . Разбира се може да се приложи същия трик за X^- , и винаги ще вземаме минималния брой от предпазени възли, който е максимум 2^{N-1} . По елегантно доказателство е дадено в книгата „Перцептрон“ на Мински и Пепърт, но същността е че при сложни трансформации броят на желаните възли в скрития слой е експонента на N .

3.7 Заключение

В този раздел бяха представени еднослойните прави мрежи за класифициране на задачите и за изпълнението на задачите за апроксимиране. Беше обсъдена представителната сила на еднослойните прави мрежи и бяха представени два обучаващи алгоритъма за намирането на оптималните тегла. Простите мрежи представени тук, имат своите предимства и недостатъци. Недостатъкът е ограничената представителна сила: само линейните класификатори могат да бъдат построени или, в случай на апроксимацията на функция, само линейни функции могат да са представени. Преимущество, обаче, се състои в това че поради линейността на системата,

обучаващия алгоритъм ще клони към оптималното решение. Този случай не се отнася за повечето нелинейни системи например като, многослойните мрежи.

Глава 7

Committee Machines (комитетни машини)

Въведение

В предишните три глави описахме три различни подхода за обучение чрез надзор. MLP-то тренирано чрез обратно разпространителния алгоритъм, обсъден в Глава 4, се позовава на формата на глобална оптимизация за своя дизайн. RBF мрежата, обсъдена в Глава 5 се позовава на локална оптимизация за своя дизайн. Поддържащата векторна машина, дискутирана в Глава 6 експлоатира димензионната теория - VC за своя дизайн. В тази глава ще обсъждаме друг клас от методи за решаване на задачи чрез надзорно обучение. Подходът използван тук е базиран на често използвани инженерен принцип: *разделяй и владей*.

Според принципа на „*разделяй и владей*”, сложна изчислителна задача се решава като се разделя на няколко прости изчислителни задачи и след това се комбинират решенията на тези задачи. В контролираното обучение, компютърната простота се постига разпространявайки задачата за обучение между няколко експерта, които от своя страна разделят входното пространство в серия от субпространства. Комбинацията от експерти представлява комитетна машина (committee machine). По принцип, това са предпазни знанията, придобити от експертите, за да достигне до цялостно решение, което по предположение е недостижимо, но възможно ако всеки от тях действа по отделно. Идеята за комитетната(сборната) машина може да се проследи назад до Нилсън (1965); структурата на мрежата се състои от слой от елементарни перцептрони последвани от гласувани перцептрони във втори слой.

Комитетните машини са универсални апроксиматори. Те могат да се разделят в две по-големи категории:

1. Статични структури.

В този клас отговорността на няколко предиктора(експерта) са комбинирани с помощта на механизъм, който не включва входния сигнал, оттам и наименованието „статичен”. Тази категория включва следните методи:

- Групово усредняване, където изходите на различни предиктори(експерти) са линейно комбинирани, за да се достигне пълна мощност.
- Увеличително, когато един слаб алгоритъм се превръща в такъв с произволно висока точност.

2. Динамични структури.

В този клас входния сигнал е пряко ангажиран със задействащия механизъм, който интегрира резултатите от индивидуални експерти в една обща продукция, оттам и наименованието „динамичен”.

Тук също ще споменем два вида динамични структури:

- Смес от експерти, в която отделните отговори на експертите не са линейно комбинирани по значение на едно-изходна мрежа.
- Йерархична смес от експерти, в която индивидуалните отговори на експертите са нелинеарно комбинирани с помощта на няколко мрежи, подредени по йерархичен начин.

В микса от експерти, принципа на „разделяй и владей” е приложен само веднъж, докато в йерархичния микс от експерти, той се прилага няколко пъти, в резултат на съответстващия им брой йерархични нива.

Сместа от експерти и йерархичната смес от експерти може да се разглежда като пример на модулни мрежи. Официална дефиниция на понятието за модулност е (Osherson и др. 1990.):

Една невронна мрежа се казва, че е модулна ако изчисленията осъществени от нея могат да се разделят на два или повече модула(подсистеми), които работят за отделни входове, без да общуват помежду си. Резултатите от модулите се опростяват от интегрираща единица, която не позволява да се връща информация обратно към тях. И по-специално, интегрираната единица (1) решава как резултатите от модулите трябва да се комбинира, за да формират крайният резултат на системата, а (2) решава кои модули да изучават избрани модели.

Това определение на модулността изключва статичния клас комитетни машини, тъй като няма интегрирана единица на изхода, която да има „вземаща-решения” роля.

Организация на Главата

Тази Глава е организирана в две части. Класът на статичните структури е включен в първата част, включващи разделите от 7.2 до 7.5. Раздел 7.2 разглежда метода на общото усредняване, следван от компютърен експеримент в раздел 7.3 . Раздел 7.4 разглежда техниката са подсилване, следвана от компютърен експеримент в раздел 7.5. Класът на динамичните структури е разгледан във втората част на главата, обхващаща разделите от 7.6 до 7.13. Конкретно, Раздел 7.6 разглежда микса от експерти(ME) под формата на асоциативен Гаузионен(Gaussian) смесен модел. Раздел 7.7 разглежда по-общия случай, а именно прекият микс от експерти(Hierarchical mixture of experts-HME). Последният модел е тясно свързан със стандартните дърводвидни решения. След това раздел 7.8 описва стандартно „дърводвидния” начин за вземане на решения. А в 7.8 се описва как „дърводвидния” начин за вземане на решения може да бъде използван за решаване на проблема при селективния модел(т.е. броят на стробиращите и експертните мрежи) за HME. В раздел 7.9 се дефинират някои от последвалите вероятности, които ни помагат в разработването на алгоритми за обучение на модела HME. В раздел 7.10 се полагат основите за решаване на проблема с параметричната оценка с цел формулиране на функцията за вероятността за модела HME. Раздел 7.11

представя преглед на стратегии за учене. Това е последвано от подробно обсъждане на така наречения алгоритъм EM в раздел 7.12 и неговото приложение към НМЕ модела в раздел 7.13. Главата завършва с някои заключителни бележки в раздел 7.14.

7.2 КОЛЕКТИВНО ОСРЕДНЯВАНЕ

Фигура 7.1 показва различен брой обучени невронни мрежи(т.е. експерти), които имат общ вход и чиито индивидуални резултати са комбинирани по някакъв начин, така че да се достигне пълна мощност у. За да се опрости представянето, резултатите на експертите се приемат за скаларно-оценени. Тази техника е обединена под един общ метод – метод1. Мотивацията за използването му се състои от две части:

- Ако комбинацията от експерти във Фигура 7.1 бъде заменена от една невронна мрежа, ние ще имаме мрежа със съответно по-голям брой регулируеми параметри. Времето за обучение а такава голяма мрежа може да бъде по-дълъг, отколкото при случай на набор от специалисти, преминали обучението паралелно.
- Рискът от препълване на паметта се увеличава, когато броят на регулируеми параметри е по-голям в сравнение с кардиналния брой на „обучаващите“ данни.

Във всеки случай, при използването на комитетна машина както е показано на фиг.7.1, очакването е, че различно обучаваните експерти се доближават до различни локални минимуми на грешка и цялостното изпълнение е подобрено чрез комбиниране на резултатите по някакъв начин. Да разгледаме първо случая с едно невронна мрежа, която е била обучена за даден набор от данни. Нека с x означим един входен вектор, неизползван до сега и нека с d обозначим съответният желан отговор(представляващ етикета class или цифрово обозначение); съответно нека x и d представляват реализация на произволен вектор x и произволна променлива d . Нека с $F(x)$ обозначим входно-изходната функция, реализирана от мрежата. След това въз основа на материала за дилемата – отклонение/вариации, разгледана в Глава 2, може да се разложи до средна квадратна грешка между $F(x)$ и условното очакване $E[D|X=x]$ в своето отклонение и компонентно вариране, както следва:

$$E_d [(F(x) - E [D | X = x])^2] = B_d (F(x)) + V_d (F(x)) \quad (7.1)$$

Където $B_d (F(x))$ е отклонението на квадрат :

$$B_d (F(x)) = (E_d [F(x)] - E [D | X = x])^2 \quad (7.2)$$

а $V_d (F(x))$ е вариацията:

$$V_d (F(x)) = E_d [(F(x) - E_d F(x))^2] \quad (7.3)$$

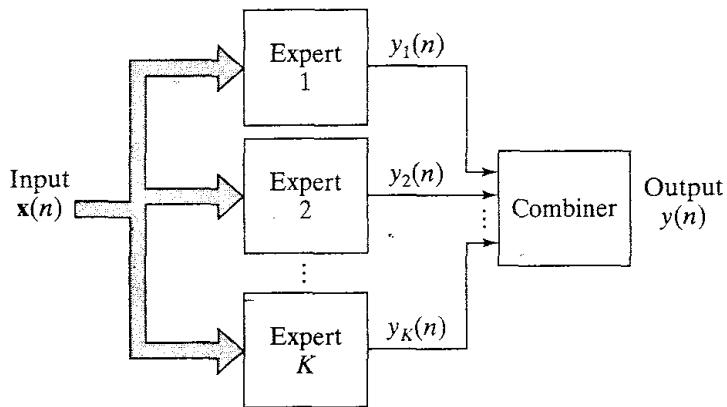


FIGURE 7.1 Block diagram of a committee machine based on ensemble-averaging.

Очакването E_d се взема от пространството D , определено като пространството обхващащо разпределението на всички обучителни комплекти (т.е. входните и изходните цели) и разпределението на всички начални условия.

Съществуват различни начини за индивидуално обучение на експертни мрежи на фиг. 7.1 и също така различни начина за комбиниране на крайните им резултати. Във връзка с представения пример ще обсъдим ситуацията, при която експертните мрежи имат идентична конфигурация, но започващи с различни начални условия. За комбинатор на изхода на машината на фиг. 7.1 ще използваме прост осредняващ обобщител (*ensemble-averger*)². Нека с φ обозначим пространството на всички начални условия. Нека $F_1(x)$ означава средната стойност на входно-изходните функции на експертните мрежи на фиг. 7.1 с определен брой първоначални условия. Аналогично чрез уравнението (7.1) може да напишем:

$$E \varphi [(F_1(X) - E [D|X=x^2])] = B \varphi (F(x)) + V \varphi (F(x)) \quad (7.4)$$

Където $B \varphi (F(x))$ е квадратното отклонение дефинирано над пространството φ :

$$B \varphi (F(x)) = (E \varphi [F_1(X)] - E [D|X=x])^2 \quad (7.5)$$

И $V \varphi (F(x))$ е съответната вариация:

$$V \varphi (F(x)) = E \varphi [(F_1(X) - E \varphi [F(x)])^2] \quad (7.6)$$

Очакваното $E \varphi$ е взето над пространството φ .

От дефиницията за пространството D , може да го разглеждаме като продукт на пространството на първоначалните условия φ и останалото пространство отбелязано с D' . Следователно отново можем да напишем по аналогичен начин чрез уравнение 7.1:

$$E_{D'} [(F_1(X) - E [D|X=x^2])] = B_{D'} (F_1(x)) + V_{D'} (F_1(x)) \quad (7.7)$$

Където $B_{D'} (F(x))$ е квадратното отклонение определено над оставащото пространство D' :

$$B_{D'}(F_1(x)) = (E_{D'}[F_1(x)] - E[D|X=x])^2 \quad (7.8)$$

И $V_{D'}(F_1(x))$ е съответната вариация:

$$V_{D'}(F_1(x)) = E_{D'}[(F_1(x) - E_{D'}[F_1(x)])^2] \quad (7.9)$$

От дефинициите за пространствата D , φ и D' ясно се вижда:

$$E_{D'}[F_1(x)] = E_D[F(x)] \quad (7.10)$$

От там следва, че уравнение (7.8) може да се запише в еквивалентната форма:

$$B_{D'}(F_1(x)) = (E_{D'}[F(x)] - E[D|X=x])^2 = B_D(F(x)) \quad (7.11)$$

Следващото, което ще вземем предвид е вариацията $V_{D'}(F_1(x))$ от (7.9). Тъй като отклонението на случайна променлива е равна на средноаритметичната стойност на квадрат на тази променлива минус нейното отклонение на квадрат, може да запишем:

$$\begin{aligned} V_{D'}(F_1(x)) &= E_{D'}[(F_1(x))^2] - (E_{D'}[F_1(x)])^2 \\ &= E_{D'}[(F_1(x))^2] - (E_D[F(x)])^2 \end{aligned} \quad (7.12)$$

Където в последния ред използваме Ур. (7.10). По същия начин можем да предефинираме уравнение (7.3) в еквивалентна форма:

$$V_{D'}(F_1(x)) = E_D[(F(x))^2] - (E_D[F(x)])^2 \quad (7.13)$$

Забележете, че средноаритметичната стойност на $F(x)$ над цялото пространство D е предвидена да бъде равна или по-голяма от средната стойност на квадрата на общата функция $F_1(x)$ над останалото пространство D' . Това е:

$$E_D[F(x)]^2 \geq E_{D'}[(F_1(x))^2]$$

Според това уравнение и уравнения (7.12) и (7.13) веднага следва, че:

$$V_{D'}(F_1(x)) \leq V_D(F(x)) \quad (7.14)$$

Така от уравнения (7.11) и (7.14) ще извадим два извода (Naftaly и др., 1997.):

1. Отклонението на средно-съставната функция $F_1(x)$, което се отнася до машината на фиг. 7.1 е точно същото като това на функцията $F(x)$, отнасящо се до единична невронна мрежа.
2. Вариациите на средната съставна функция $F_1(x)$ са по-малко от тези на функция $F(x)$.

Тези теоретични констатации говорят за обучаваща стратегия за намаляване на общото ниво на грешки, произведени от комитетната машина поради различните начални условия (Naftaly и др., 1997): съвместимите експерти на машината са нарочно тренирани над нормата, а използването им е основано на следните основания. Доколкото отделни експерти се концентрират към отклонението, толкова се намаля

„цената“ на вариациите. Впоследствие обаче, вариацията се намалява с общото усредняване на експерти спрямо първоначалните условия, оставящи отклонението непроменено.

7.3 КОМПЮТЪРЕН ЕКСПЕРИМЕНТ 1

В този компютърен експеримент на общият среден метод, ние преразглеждаме моделно класификационния проблем, разгледан в предходните три глави. Проблемът се отнася до класирането на две припокриващи се Gaussian дистрибуции. Двете дистрибуции имат различни средни вектори и различни отклонения. Статистиката на дистрибуция 1 (клас ζ_1) е

$$\mu_1 = [0, 0]^T$$

$$\sigma_1^2 = 1$$

Статистиката на дистрибуция 2 (клас ζ_2) е

$$\mu_2 = [2, 0]^T$$

$$\sigma_2^2 = 4$$

Разпръснатите площи за тези две дистрибуции са показани на фиг. 4.13 . Двата класа се приема, че са equiprobable. Разходите за недоброто класифициране се приема, че са равни, а точните класификации се приема за 0. На тази основа Бейс класификатора постига вероятността за правилна класификация $p_c = 81.51$ процента. Детайлите по това изчисление са представени в Глава 4.

В компютърният експеримент, описан в Глава 4 бяхме в състояние да постигнем вероятност за правилна класификация близо 80% с помощта на многопластов перцептор с два скрити неврона и трениран чрез обратно размножаване алгоритъм. В този експеримент ние изучавахме една машина композирана както следва:

- Десет експерта.
- Всеки експерт, съставен от многослойни перцептрони и два скрити неврона. Всички експерти са лично обучени с помощта на обратно-размножителния логаритъм. Параметрите използвани в алгоритъма бяха

Learning-rate параметър, $\eta = 0.1$

Временна константа $\alpha = 0.5$

Размерът на обучителната проба бе 500 модела. Всички експерти бяха обучени по еднакви данни, но инициализирани по различен начин. По-специално, първоначалната стойност на синаптичните тегла и праговете са били уловени случайно от равномерното разпределение в самия интервал [-1,1].

Таблица 7.1 представя резюме на класификационно изпълнение от 10-те експерта, обучени за 500 модела, използвайки тест комплекта. Вероятността от правилни класификации получени само чрез взимането на средно-аритметичната величина от 10 резултата, представени в таблица 7.1 е $P_{c,av} = 79.37$ процента. От друга страна, използвайки ensemble-averaging метода, който е просто сумиране на индивидуалните резултати от 10 експерта, а след това изчисляване на вероятността за правилно класиране се получи резултат: $P_{c,ens} = 80.27$ процента. Този резултат представя едно одобрение от 0.9 процента над $P_{c,av}$. Подобренето на $P_{c,ens}$ над $P_{c,av}$ се запазва във всички проучвания на експеримента. Класификационните резултати бяха всичките изчислени с помощта на 32,000 тест модела.

TABLE 7.1 Classification Performances of Individual Experts Used in a Committee Machine

Expert	Correct classification percentage
Net 1	80.65
Net 2	76.91
Net 3	80.06
Net 4	80.47
Net 5	80.44
Net 6	76.89
Net 7	80.55
Net 8	80.47
Net 9	76.91
Net 10	80.38

Обобщавайки резултатите от този експеримент, можем да кажем: Класификационното изпълнение е подобрено чрез трениране на отделните многослойни перцептрони(експерти), като първо обединява техните индивидуални цифрови изходи за изготвяне на цялостната продукция на машината и след това се взима решението.

7.4 BOOSTING (Стимулиране, повишаване на ефективността)

Както бе споменато във въведението, буутстването(boosting) или повишаването е друг метод, който принадлежи на „статичният” клас на комитетните машини. Boosting метода е доста различен от метода с общото усредняване. В комитетна машина, базирана на общото усредняване всички експерти в машината са обучени по едни и същи данни; могат да се различават един от друг при избора на началните условия, използвани в мрежовото обучение. От друга страна, в boosting машината експертите са обучени по данни с напълно различни дистрибуции; това е генерален метод, който може да се използва за подобряване на работата на всеки обучаващ алгоритъм.

Boosting може да се реализира по три коренно различни начина:

1. **Стимулиране чрез филтриране(Boosting by filtering).** Този подход включва филтриране на обучителните примери чрез различни версии на слаб обучаващ алгоритъм. Това предполага наличие на голям(на теория-безкраен) източник на примери, като примерите бъдат или изхвърлени или запазени по време на обучението. Едно от предимствата на този подход е, че той изисква по-малко памет в сравнение с другите два подхода.
2. **Стимулиране чрез вземане на проба(Boosting by subsampling).** Този втори подход работи с обучаваща извадка от фиксиран размер. Примерите са променени, съгласно определено вероятностно разпределение по време на обучението. Грешката се изчислява по отношение на фиксираната обучителна проба.
3. **Стимулиране на препретеглянето(Boosting by reweighting).** Този трети подход също работи с фиксирана обучителна проба, но предполага, че слабия обучителен алгоритъм може да получава „претеглени” примери. Грешката се изчислява по отношение на „претеглените” примери.

В този раздел ще опишем два различни стимулиращи алгоритъма. Един алгоритъм по Schapire (1990), принадлежи към подход 1. Другия, известен като AdaBoost по Freund и Schapire (1996a, 1996b), принадлежи към подход 2.

Boosting by filtering(Стимулиране чрез филтриране).

Първоначалната идея за стимулиране описана в Schapire (1990) се корени в свободното разпределение или може би в приблизително верния (PAC) модел на обучение. От дискусията за PAC обучението описана в Глава 2, ние си спомняме, че концепцията е Булева функция в някои случаи на домейн, който съдържа кодировки на всички обекти, представляващи интерес. В PAC обучението, една обучаваща машина се опитва да идентифицира непозната двуместна концепция на базата на случайно избрани примери на концепцията. По-конкретно целта на обучаващата машина е да намери хипотеза или прогнозиращо правило с допустимост за грешка е, при произволно малки положителни

стойности на Е и това трябва да се отнася еднакво за всички входни дистрибуции. Това е и причината РАС обучаващия модел да е посочен като силен обучителен модел. Тъй като примерите са от случаен характер, то е вероятно обучаващата машина да не е в състояние да научи нещо за неизвестно съдържание, тъй като примерите са силно непредставителни. За това ние изискваме обучаващия модел да успее само в намиране на добро приближаване до непознатото съдържание с вероятност $1-\delta$, където δ е малко положително число.

В даден вариант на РАС изучаващият модел, наричан *слаб изучаващ модел*, изискването за изучаване на неизвестна концепция е намалено драстично. Обучаващата машина сега трябва да намери хипотеза с коефициент за грешка само малко по-малко от $\frac{1}{2}$. Когато хипотеза предполага бинарен етиケット в напълно случаен характер на всеки пример, може да бъде правилна или не с еднаква вероятност. Това означава, че тя постига коефициент за грешка точно $\frac{1}{2}$. Оттук следва, че един слаб изучаващ модел трябва да изпълнява само по-леки/слаби модели отколкото тези на случаен принцип. Понятието за слабото изучаване е въведено от Kearns и Valiant(1989), което постави хипотетичния повишителен (boosting) проблем, който е залегнал в следният въпрос:

Дали понятията силно и слабо обучение са еквивалентни? (*Are the notions of strong and weak learning equivalent?*)

С други думи дали всеки концептуален клас, който се изучава като слаб може да се изучава и като силен? На този въпрос, който е може би изненадващ беше отговорено положително от Schapire(1990г.). Доказателствата представени там от него са конструктивни. Конкретно, създаден е алгоритъм за директно конвертиране на слаб изучаващ модел в силен изучаващ модел. Това е постигнато като е модифицирано разпределението на примери по такъв начин, че един силен изучаващ модел е изграден около слаб.

В повишаване чрез филтриране, комисионната машина се състои от три експерта или под-хипотези. Алгоритъмът за тяхното обучение се нарича повишителен алгоритъм. Тримата експерти са условно наречени „първи”, „втори” и „трети”. Тримата експерти са индивидуално обучени, както следва:

1. Първият експерт е обучаван по комплект, състоящ се от N_1 примери.
2. Обученият първи експерт се използва за филтриране на друг набор от примери, като се прилагат по следния начин:
 - „Хвърляне на справедлива монета”, това въщност симулира случаен избор.
 - Ако резултата е ези, пропускат се новите модели през първия експерт и се отхвърлят правилно класифицирани модели, докато един модел е определен погрешно за отрицателен. Този модел се добавя към пакета за обучение за втория модел.

- Ако резултата е тура, прави точно обратното. Конкретно, прекарване на нови модели през първият експерт и изхвърляне на неправилно класифицираните, докато даден модел се класифицира правилно. Този правилно класифициран модел се добавя към тренировъчния сет на втория експерт.
- Продължи този процес, до достигането на N_1 примери, филтрирани от първия експерт. Този сет от филтрирани примери представлява обучаващия сет на втория експерт.

Следвайки тази „монетно-обръщаща“ процедура се гарантира, че ако първият експерт е тестван на втория сет от примери ще има коефициент за грешка $\frac{1}{2}$. С други думи, вторият сет от N_1 примери, свободен за обучение на втория експерт има дистрибуция, напълно различна от първия сет от N_1 примера, използвани за обучение на първия експерт. По този начин, вторият експерт е принуден да научи дистрибуция, различна от тази, която е научил първия експерт.

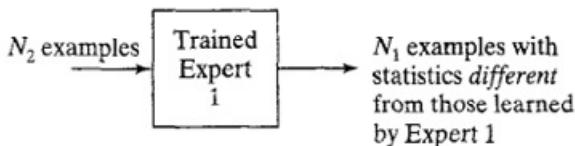
3. Веднъж научен вторият експерт по обичайния начин, трети тренировъчен сет е формиран за третия експерт, следвайки следния маниер:

- Пропусни нов модел през двата – първият и вторият експерт. Ако двата експерта са единодушни в тяхното решение, отхвърлете този модел. От друга страна ако те го отхвърлят, модела се добавя към тренировъчния сет за третия експерт.
- Продължете с процеса, докато примерите са филтрирани съвместно от първия и втория експерт. Този сет от съвместно филтрирани примери представлява тренировъчния сет за третия експерт.

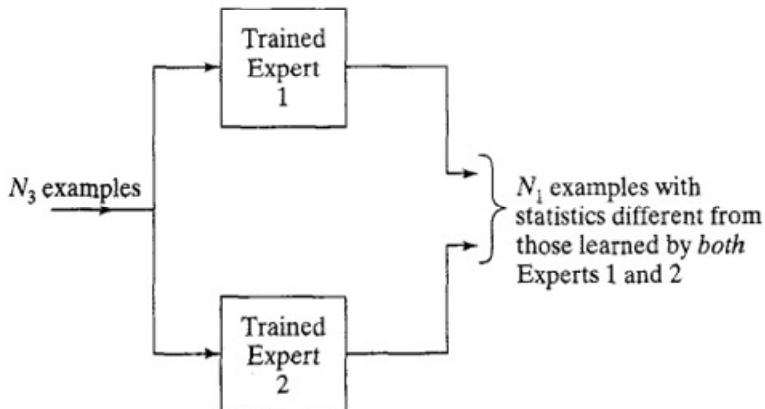
Третият експерт се тренира по обичайния начин и обучението на цялата машина приключи по този начин.

Третият етап на филтриращата процедура е илюстриран на Фиг.7.2

Нека с N_2 означим броя на примерите, които трябва да бъдат филтрирани с първия експерт, за да придобием тренировъчният сет от N_1 примера за втория експерт. Забележете, че N_1 е фиксирано, а N_2 зависи от обобщеният процент за грешка на първия експерт. Нека с N_3 отбележим броя на примерите, които трябва да бъдат общо филтрирани от първия и втория експерт, за да получим тренировъчният сет на N_1 примерите за третия експерт. С N_1 примерите трябва също да бъде трениран и първия експерт. Общийят размер на всички данни, необходими за обучение на цялата комисионана машина е $N_4 = N_1 + N_2 + N_3$. Въпреки това, изчислителната стойност се основава на $3N_1$ примера, защото N_1 е броят на примерите, които всъщност са използвани за трениране на всеки един от трите експерта. За това можем да кажем, че стимулиращият алгоритъм описан тук е наистина „умен“ в смисъл, че комитетната машина изисква голям набор от примери за своето действие, но само част от тези данни са използвани за същностната тренировка.



(a) Filtering of examples performed by Expert 1



(b) Filtering of examples performed by Experts 2 and 3

FIGURE 7.2 Illustration of boosting by filtering.

В теоретичният извод за стимулиращият алгоритъм първоначално представен в Schapire (1990), опростено гласуване е било използвано за оценяване на дейността на машината на тестовите модели, невиждано до сега. Конкретно, тест модел е представен на комитетната машина. Ако първият и вторият експерт са единодушни в решението си този клас се използва. Иначе, класа открыт от третия експерт се използва. Въпреки това, в експериментална работа представена в Drucker et al. (1993, 1994) е установено, че чрез допълването на съответните резултати от тримата експерти се получава по-добър резултат, отколкото при гласуване. Например в проблема за оптичното разпознаване на символи – optical character recognition(OCR), съответната операция е само да се добави „цифра 0“ към резултатите на трите експерта, а същото се отнася и за другите девет цифрови резултати.

Да предположим, че тримата експерти (т.е. суб-хипотези) имат процент на грешка от $\epsilon < \frac{1}{2}$ по отношение на дистрибуцията, по която те са индивидуално обучени, т.е. те всички са слаби обучаващи модели. В Schapire (1990) е доказано, че общото ниво на грешките на комитетната машина е ограничена от

$$g(\epsilon) = 3\epsilon^2 - 2\epsilon^3 \quad (7.15)$$

Границата $g(\epsilon)$ е представена спрямо ϵ на Фиг. 7.3. От тази фигура виждаме, че границата е значително по-малка от оригиналния коефициент за грешка ϵ . Прилагайки стимулиращият алгоритъм рекурсивно, новото на грешки може да се направи условно малко. С други думи, един слаб обучаващ модел, който се представя само малко по-добре от случайния избор, е превърнат в силен обучаващ модел.

Именно в този смисъл можем да кажем, че силните и слабите обучителни възможности са наистина равностойни.

AdaBoost

Практическо ограничение на стимулирация алгоритъм е, че често изисква голяма тренировъчна извадка. Това ограничение може да бъде преодоляно по друг стимулиращ алгоритъм наречен AdaBoost (Freund and Schapire, 1996a, 1996b), който принадлежи към повторното вземане на проби. Пробовата „рамка“ на AdaBoost е естествената рамка на партидното обучение; най-важното е, че позволява на тренировъчните данни да бъдат преизползвани.

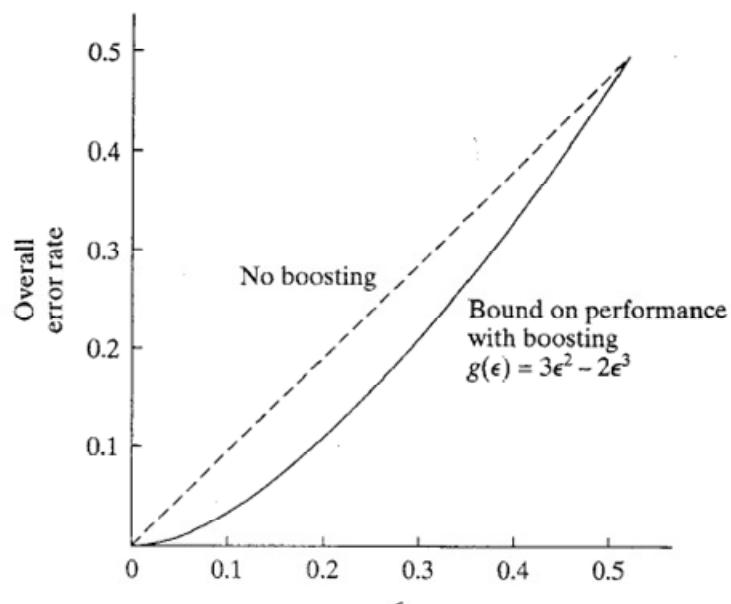


FIGURE 7.3 Graph of Eq. (7.15) for boosting by filtering.

Както и при *boosting by filtering* алгоритъма, AdaBoost има достъп до слаб обучаващ модел. Целта на новият алгоритъм е да намери окончателна картова функция или хипотеза с нисък процент за грешка, релативен на дадена дистрибуция D над етикираните примери за обучение. Той се различава от други алгоритми за повишаване в две отношения:

- AdaBoost се настройва адаптивно към грешките на слабата хипотеза, върната от слабия обучаващ модел, откъдето идва и името на алгоритъма.
- Ограничението за изпълнението на AdaBoost зависи само от изпълнението на слабия обучаващ модел за тези дистрибуции, които са действително генериирани по време на учебния процес.

AdaBoost функционира по следния начин. На повторение n стимулирацият алгоритъм предоставя слабия модел на обучение с дистрибуция \mathcal{D}_n над тренировъчния пример

\mathcal{T} . В отговор слабия обучаващ модел изчислява хипотеза $\mathcal{F}_n : \mathbf{X} \rightarrow Y$, която правилно класифицира част от обучителните примери. Грешката се измерва по отношение на дистрибуцията \mathcal{D}_n . Процесът продължава за T повторения, а накрая стимулиращата машина съчетава хипотези $\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_T$ в една финална хипотеза \mathcal{F}_{fin}

За да изчислим (1) дистрибуцията \mathcal{D}_n на повторението n и (2) крайната хипотеза \mathcal{F}_{fin} , най-опростената процедура е обобщена и използвана в Таблица 7.2. Първоначалната дистрибуция \mathcal{D}_1 е постоянна над тренировъчния пример \mathcal{T} , както следва

$$\mathcal{D}_1(i) = \frac{1}{n} \quad \text{for all } i$$

Давайки на дистрибуцията \mathcal{D}_n и слабата хипотеза \mathcal{F}_n през повторение n на алгоритъма, следващата дистрибуция \mathcal{D}_{n+1} е изчислена, умножавайки стойността на пример i с някакво число $\beta_n \in [0, 1]$ ако \mathcal{F}_n класифицира изходящия вектор \mathbf{x}_i правилно; иначе стойността остава непроменена. Тогава стойностите са пре-нормализирани, разделяйки ги с нормализиращата константа Z_n . Ефекта е, че „лесните“ примери в обучаващия сет \mathcal{T} , които са правилно класифицирани от много от предишните слаби хипотези получават по-ниски стойности, а на „силните“ примери, които често са определени погрешно за отрицателни се дават по-големи стойности. Така AdaBoost алгоритъма фокусира най-голямата стойност на тези примери, върху тези, които изглежда са най-трудните за него да бъдат класифицирани.

Що се отнася до крайната хипотеза \mathcal{F}_{fin} е изчислена като претеглен глас (т.e. претеглен линеен праг) на тези слаби хипотези $\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_T$. Това означава, за даден изходящ вектор \mathbf{x} , крайната хипотеза \mathcal{F}_{fin} изчислява етикета d , който увеличава сумата от теглата от слабите хипотези, прогнозирайки този етикет. Стойността на хипотезата \mathcal{F}_n е дефинирана да бъде $\log(1/\beta_n)$, така че по-голяма стойност да се дава на хипотеза с по-малка грешка.

Важна теоретична стойност на AdaBoost е заложена в следващата теорема (Freund and Schapire, 1996a):

Нека предположим, че когато слабият обучаващ модел бъде „извикан“ от AdaBoost генерира хипотези с грешки $\epsilon_1, \epsilon_2, \dots, \epsilon_T$, където грешката ϵ_n на итерация n на AdaBoost алгоритъма се определя от

$$\epsilon_n = \sum_{i: \mathcal{F}_n(\mathbf{x}_i) \neq d_i} \mathcal{D}_n(i)$$

Да приемем, че $\epsilon_n \leq \frac{1}{2}$ и нека $\gamma_n = \frac{1}{2} - \epsilon_n$. Тогава следващата горна граница се придържа към грешката на финалната хипотеза:

$$\frac{1}{N} |\{i : \mathcal{F}_{\text{fin}}(\mathbf{x}_i) \neq d_i\}| \leq \prod_{n=1}^T \sqrt{1 - 4\gamma_n^2} \leq \exp\left(-2 \sum_{n=1}^T \gamma_n^2\right) \quad (7.16)$$

TABLE 7.2 Summary of AdaBoost

<i>Input:</i>	Training sample $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ Distribution \mathcal{D} over the N labeled examples Weak learning model Integer T specifying the number of iterations of the algorithm
<i>Initialization:</i>	Set $\mathcal{D}_1(i) = 1/N$ for all i
<i>Computation:</i>	Do the following for $n = 1, 2, \dots, T$:
1.	Call the weak learning model, providing it with the distribution \mathcal{D}_n .
2.	Get back hypothesis $\mathcal{F}_n : \mathbf{X} \rightarrow Y$
3.	Calculate the error of hypothesis \mathcal{F}_n :
	$\epsilon_n = \sum_{i: \mathcal{F}_n(\mathbf{x}_i) \neq d_i} \mathcal{D}_n(i)$
4.	Set $\beta_n = \frac{\epsilon_n}{1 - \epsilon_n}$
5.	Update the distribution \mathcal{D}_n :
	$\mathcal{D}_{n+1}(i) = \frac{\mathcal{D}_n(i)}{Z_n} \times \begin{cases} \beta_n & \text{if } \mathcal{F}_n(\mathbf{x}_i) = d_i \\ 1 & \text{otherwise} \end{cases}$
	where Z_n is a normalization constant (chosen so that $\mathcal{D}_{n+1}(i)$ is a probability distribution).
<i>Output:</i>	The final hypothesis is
	$\mathcal{F}_n(\mathbf{x}) = \arg \max_{d \in \mathcal{D}} \sum_{n: \mathcal{F}_n(\mathbf{x})=d} \log \frac{1}{\beta_n}$

Тази теорема показва, че ако слабите хипотези, конструирани от слаб обучаващ модел последователно имат грешка повече от $\frac{1}{2}$, след това обучаващата грешка на крайната хипотеза \mathcal{F}_{fin} спадне до 0 показателно бързо. Това обаче не означава, че повсеместната грешка на експерименталните данни е задължително да бъде малка. Експерименти, представени в Freund and Schapire (1996a) показват две неща. На първо място, теоретичната граница на обучителната грешка обикновено е слаба. На второ място, обобщаващата грешка обикновено е много по-добра от това, което теорията предлага.

Таблица 7.2 представя резюме на AdaBoost за двукомпонентни класификационни проблеми. Когато броят на възможните класове (етикиети) е $M > 2$, boosting проблема става по-сложен, тъй като вероятността за случайно познаване дава правилната

стойност $1/M$, която сега е по-малка от $1/2$. За да бъде възможно за boost-то да използва някоя хипотеза, която е по-добра от случайното познаване, в такава ситуация ние трябва по никакъв начин да променим алгоритъма и дефиницията на това какво значи „слаб обучаващ“ алгоритъм. Начини да се използва тази промяна са описани в Freund and Schapire (1997) и Schapire (1997).

Грешка при изпълнение

Експериментите с AdaBoost докладвани в Breiman (1996b) показват, че когато обучителната грешка и тест-грешката се нанасят като функция на броя на увеличаващите се повторения, често откриваме, че тест-грешката продължава да намалява, след като обучителната грешка е вече намаляла до нула.

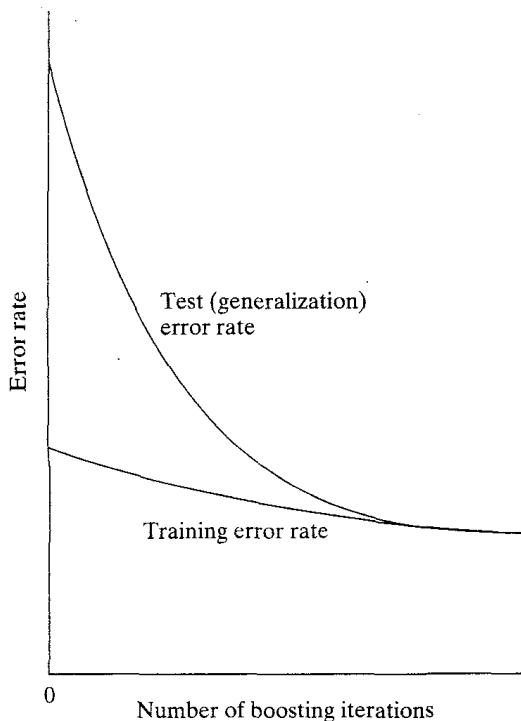


FIGURE 7.4 Conceptualized error performance of the AdaBoost algorithm.

Този феномен е илюстриран на Фиг.7.4. Подобен резултат е бил докладван по-рано в Drucker et al. (1994) за *boosting by filtering*. Феноменът показан на фиг. 7.4 е много изненадващ, с оглед на това, което знаем за обобщеното изпълнение на една невронна мрежа. От глава 4 си спомняме, че в случай на многослойни перцептрони тренирани с back-propagation алгоритъма, грешката на тест данните намалява, достига минимум и след това се увеличава в резултат на overfitting; виж фиг. 4.20. Поведението, показано на Фиг.7.4 е изключително различно, тъй като мрежите стават все по-сложни чрез увеличаване на обучението, а генерализационната грешка продължава да се понижава. Такова явление изглежда в противоречие с Occam's razor, който гласи, че една обучаваща машина трябва да бъде възможно най-проста, за да се постигнат добри генерализационни резултати.

В Schapire et al. (1997) се дава обяснение за това явление, тъй като се отнася за AdaBoost. Основната идея за анализа, представен в него е, че в оценката на генерализационната грешка, произведена от boosting машината, не само обучителната грешка трябва да се вземе предвид, но също и конфиденциалността на класификацията. В анализа представен там е показана връзката между boosting и support вектор машините; support вектор машините са разгледани в предишната глава. По-конкретно, класификационната граница например се определя като разлика между теглото, възложени на правилния етикет, спадащи към този пример и максималното тегло на всеки един неточен етикет. От това определение е лесно да се види, че границата е число в интервала $[-1, 1]$ и че един пример е коректно класифициран ако и само неговата граница е положителна. Така Schapire et al. Показва, че феномена описан във Фиг. 7.4 е наистина свързан с разпределението на границите на тренировъчните примери по отношение на генерираната гласувана класификационна грешка. Трябва отново да се подчертая, че граничният анализ представен в Schapire et al. (1997) е специфично за AdaBoost и не се прилага към други boosting алгоритми.

7.5 КОМПЮТЪРНИ ЕКСПЕРИМЕНТ II

В този експеримент ще опознаем *boosting by filtering* алгоритъма за решаване на доста трудна задача за класификационния модел. Класификационният проблем е двуизмерен, включващ региони с неизпъкващи решения, както е показано на Фиг. 7.5. Един клас от модели се състои от информационни точки, лежащи в отбелязания \mathcal{C}_1 регион, а другия клас от модели се състои от информационни точки във региона \mathcal{C}_2 . Изискването е да се изработи комисионна машина, която решава дали един тестови модел принадлежи към клас \mathcal{C}_1 или към клас \mathcal{C}_2 .

Комисионната машина, използвана за решаването на този проблем се състои от три експерта. Всеки експерт се състои от 2-5-2 многослойни перцептрони, които имат два входни възела, пет скрити неврона и два изходни неврона. Алгоритъмът Back-propagation е използван за извършване на обучение. Фиг. 7.6 показва разпръснатите парцели на информацията, използвана за трениране на тези три експерта.

Информацията показана на Фиг. 7.6a е използвана, за да се тренира експерт1.

Информацията, показана на Фиг. 7.6b е филтрирана от експерт1, след като обучението му е свършило; този набор от данни е бил използван за обучението на експерт2.

Информацията, показана на Фиг. 7.6c е филтрирана от комбинирано действие на обучените експерти 1 и 2; този набор от данни е бил използван за обучението на експерт3. Размерът на обучителната проба на всеки експерт бе $N_1 \approx 1000$ модела.

Разглеждайки тези 3 фигури наблюдаваме следното:

- Обучителните данни за експерт 1 на Фиг. 7.6a са равномерно разпределени.
- Обучителните данни за експерт 2 на Фиг. 7.6b са изложени концентрирано на точки от данни в области, етикирани А и В, които привидно за трудни за

класификация на първия експерт. Броят на точките с данни в тези два региона се равнява на броя на правилно класифицираните точки.

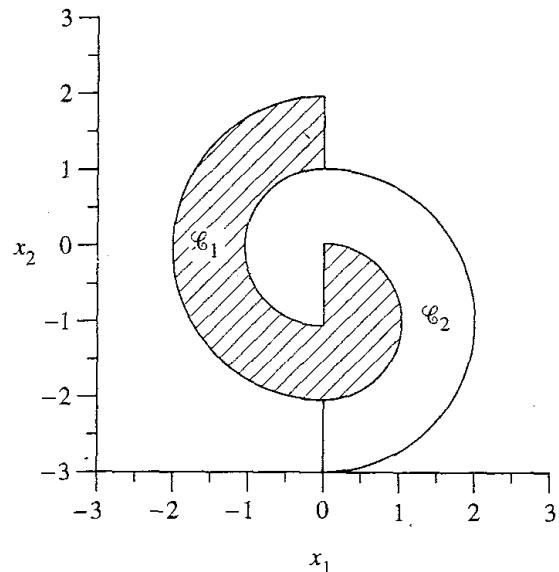


FIGURE 7.5 Pattern configurations for experiment on boosting.

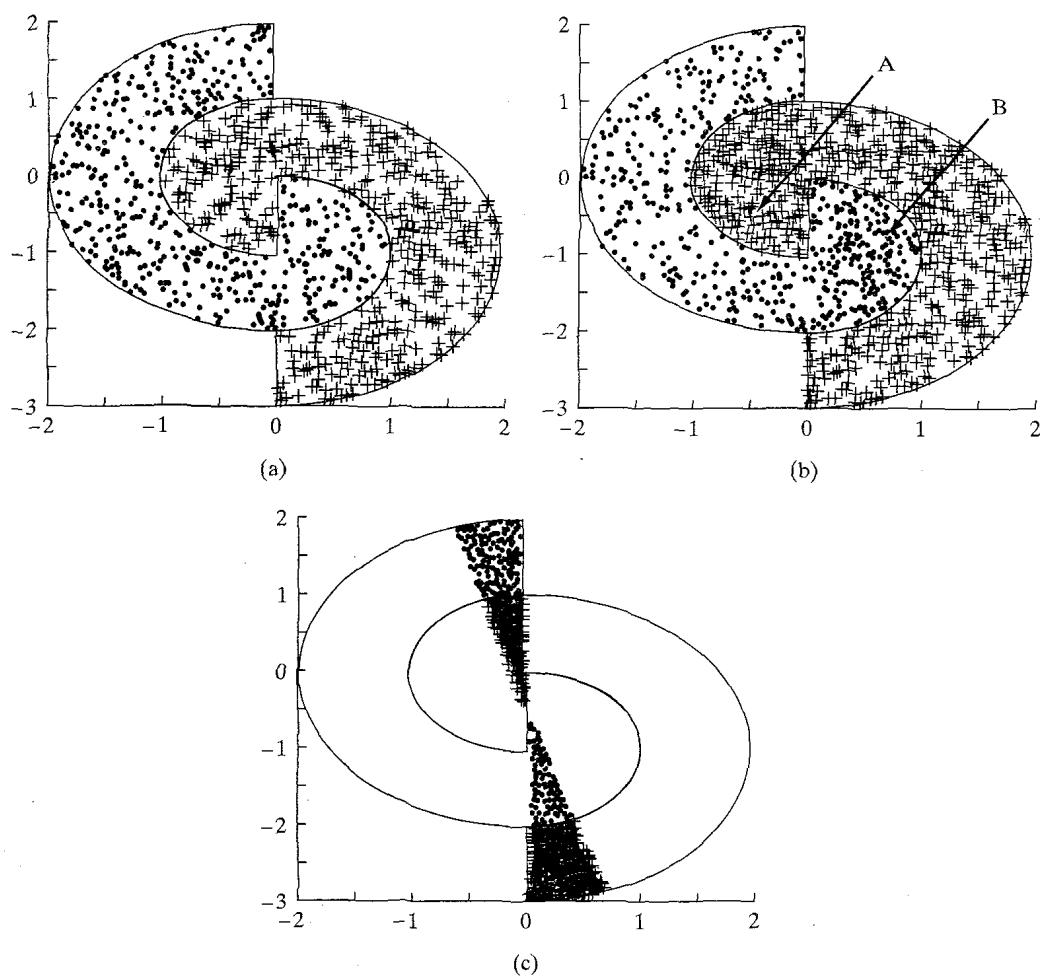


FIGURE 7.6 Scatter plots for expert training in computer experiment on boosting:
(a) Expert 1. (b) Expert 2. (c) Expert 3.

- Обучителните данни за експерт 3 на Фиг.7.6с проявяват още по-голяма концентрация на точките с данни, които са трудни и за двата експерта 1 и 2 за класифициране.

Фигурите 7.7а, 7.7б и 7.7с изобразяват крайните граници, формирани от експерти 1,2 и 3, съответно. Фигура 7.7д показва крайната граница, формирана от комбинираните действия на трите експерта, която се получава от простото сумиране на индивидуалните резултати. Имайте предвид, че разликата между крайните региони от Фигури 7.7а и 7.7б, отнасящи се до експерти 1 и 2, определя разпределението на точките с данни на Фиг. 7.6с, използвани на експерт 3. Вероятностите за правилно класифициране за трите експерта на експериментални данни са:

Expert 1: 75.15 percent

Expert 2: 71.44 percent

Expert 3: 68.90 percent

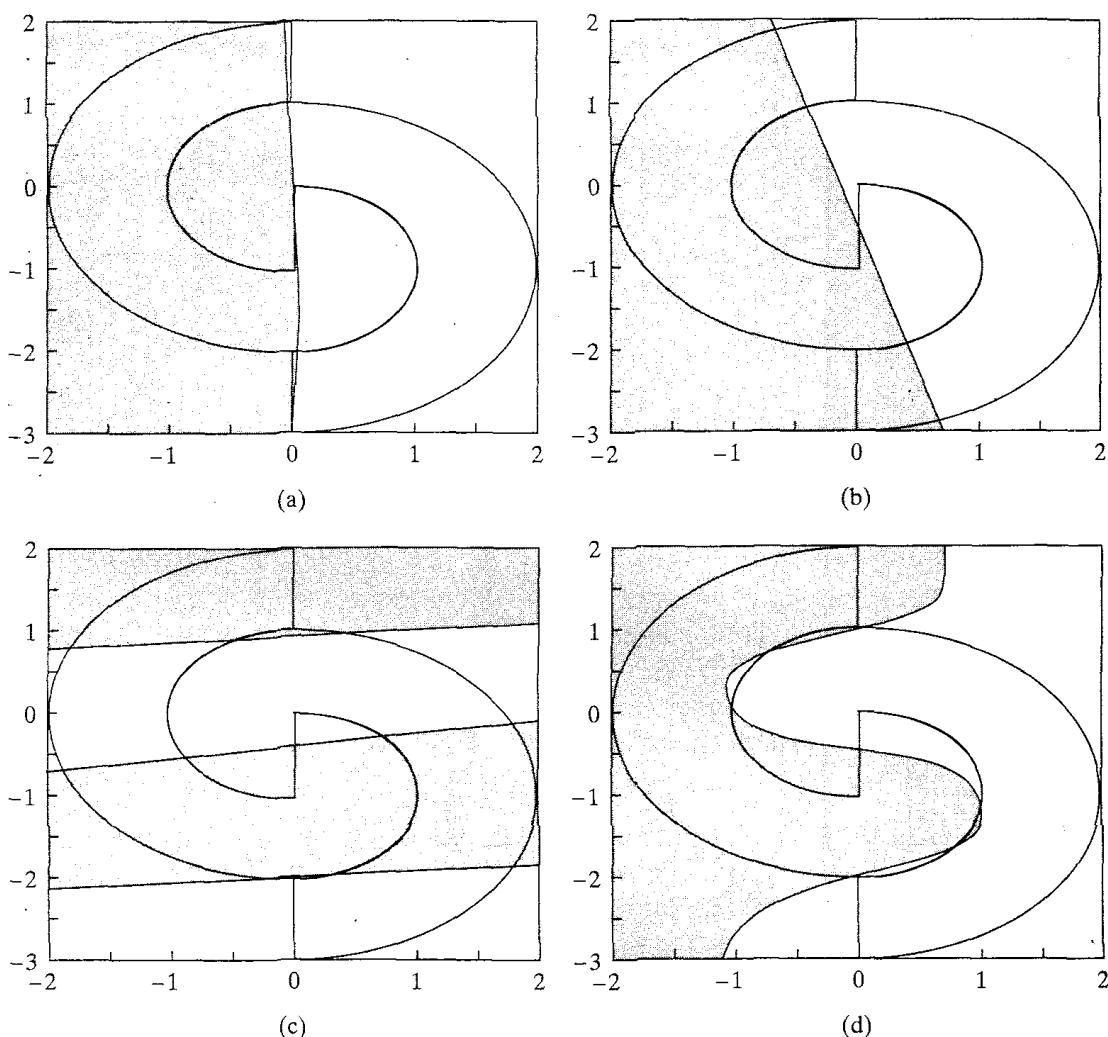


FIGURE 7.7 Decision boundaries formed by the different experts in the boosting experiment. (a) Expert 1. (b) Expert 2. (c) Expert 3. (d) Entire committee machine.

Общата вероятност за правилното класифициране за цялата комисионна машина е 91,79%, която е изчислена, използвайки 32,000 модела за тест-датата. Общото гранично решение, построено от повишаващият алгоритъм за трите експерта, показано на фиг. 7.7d е още едно доказателство за добро представяне на класификация.

7.6 Асоциативен Гаузионен смесен модел (ASSOCIATIVE GAUSSIAN MIXTURE MODEL)

Във втората част на главата, като се започне с този раздел, ще се изучава втори клас на комитетните машини, а именно динамичните структури. Понятието „динамична“ тук се използва в смисъл, че интегрирането на знанията, придобити от експертите се извършва под действието на входния сигнал.

За да започне дискусията взимаме предвид модулна мрежа, в която процеса на обучение продължава, сливайки самостоятелно организирания и контролирам начин на обучение по един непрекъснат начин. Експертите технически надзоряват обучението и така техните индивидуални резултати се комбинират, за да получат желания отговор. По този начин обаче експертите също така извършват самоорганизация на учението, т.е. те се самоорганизират, за да се получи добро разделяне на входното пространство, така че всеки експерт да се справи с моделирането на собственото си подпространство и като цялостна група да моделират входното пространство добре. В обучителната схема, която описахме току що съществува отправна точка от схемите разгледани в предходните три глави, от които се приема един специфичен модел за генериране на данни за обучение.

Вероятностен Генеративен Модел(Probabilistic Generative Model)

За да коригираме идеи, да разгледаме регресивния проблем, при който един регресор \mathbf{x} произвежда отговор, обозначен със случайна величина \mathbf{D} ; реализация на тази случайна величина се означава с d . Без загуба на общоприложимостта, приехме скаларна форма на регресия, просто да се опрости представянето. Но – просто, предполага се, че образуването на отговора d се ръководи от следния вероятностен модел(Jordan and Jacobs, 1995):

1. Един входен вектор x се взима на случаен принцип от някое предварително разпределение.
2. Едно специално правило, така нареченото **kth** правило е избрано в съответствие с условната вероятност $P(k|x, \mathbf{a}^{(0)})$, дадено x и някой параметричен вектор $\mathbf{a}^{(0)}$.
3. За правило k , $k = 1, 2, \dots, K$, моделният отговор d е линеарен в x , с добавена грешка ϵ_k , моделирана като Гаузионна разпределителна случайна величина с нулева средна стойност и единица вариация:

$$E[\epsilon_k] = 0 \quad \text{за всяко } k \quad (7.17)$$

и

$$\text{Var} [\epsilon_k] = 1 \quad \text{за всяко } k \quad (7.18)$$

Под точка 3, допуснатата единица вариация е направена само за дидактическа простота. По принцип, всеки експерт може да има различна изходна вариация, която може да се научи от обучителните данни.

Вероятностното поколение на D се определя от условната вероятност $P(D = d | \mathbf{x}, \mathbf{w}_k^{(0)})$, дадено \mathbf{x} и някой параметричен вектор $\mathbf{w}_k^{(0)}$, за $k = 1, 2, \dots, K$. не се изиска вероятностния генеративен модел, който описахме току-що да е в пряка връзка с физическата реалност. Вместо това, ние просто изискваме вероятностните решения, въплътени в него да представляват абстрактен модел, който с нарастваща точност да посочва мястото на условната стойност на отговор d , на нелинеен колектор, който свързва входния вектор в смислена продукция (Jordan, 1994).

Според този модел, отговорът D може да се получи по K различни начина, съответстващи на K избора с обозначение k . По този начин, условната вероятност за генериране на отговор $D = d$, даден входен вектор \mathbf{x} се равнява на :

$$P(D = d | \mathbf{x}, \theta^{(0)}) = \sum_{k=1}^K P(D = d | \mathbf{x}, \mathbf{w}_k^{(0)}) P(k | \mathbf{x}, \mathbf{a}^{(0)}) \quad (7.19)$$

Където $\theta^{(0)}$ е *генеративния моделен параметричен вектор*, обозначаващ комбинацията $\mathbf{a}^{(0)}$ и $\{\mathbf{w}_k^{(0)}\}_{k=1}^K$. Горният индекс 0 в $\mathbf{a}^{(0)}$ и $\mathbf{w}_k^{(0)}$ има за цел да разграничи генеративните моделни параметри от тези, които са *смесица* от експертни модели. Сега ще разгледаме последните.

Модел – Смес от Експерти (Mixture of Experts Model)

Да разгледаме мрежовата конфигурация на фигура 7.8, която показва Модела – Смес от Експерти (ME)⁴. По-конкретно, състои се от K контролиращи модула, наречени мрежа от експерти или само експерти, както и интегрирана единица наречена стробираща мрежа, която изпълнява функцията на посредник между експертни мрежи. Тук се приема, че различните експерти работят най-добре в различни региони на входното пространство в съответствие с описания вероятностен генеративен модел, откъдето произтича и необходимостта от входна мрежа.

С регресивният проблем, който се приема, че е скаларен, всеки мрежов експерт трябва да се състои от линеарен филтър. Фиг. 7.9 показва поточния сигнал на един неврон, съдържащ експерт k . По този начин изходният продукт на експерт k е вътрешния продукт на входния вектор \mathbf{x} и синаптичния тегловен вектор \mathbf{w}_k на този неврон, както е показано:

$$y_k = \mathbf{w}_k^T \mathbf{x}, \quad k = 1, 2, \dots, K \quad (7.20)$$

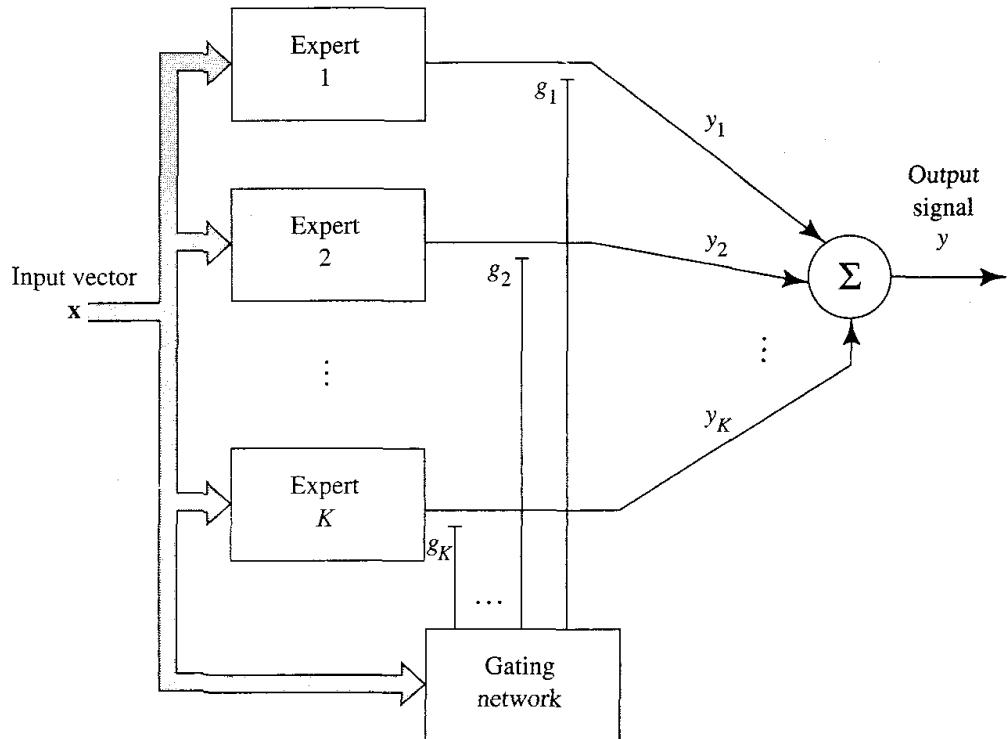


FIGURE 7.8 Block diagram of the ME model; the scalar outputs of the experts are mediated by a gating network.

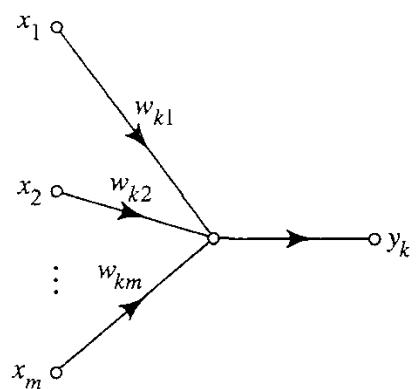


FIGURE 7.9 Signal-flow graph of a single linear neuron constituting expert k .

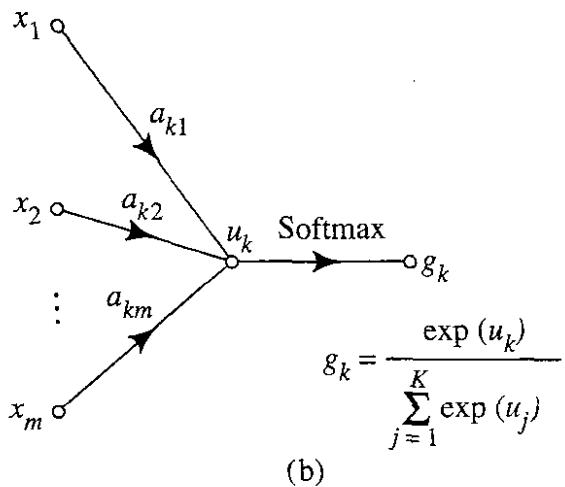
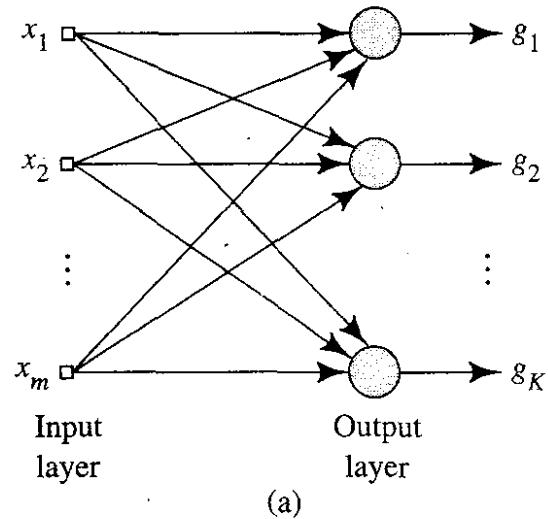


FIGURE 7.10 (a) Single layer of softmax neurons for the gating network. (b) Signal-flow graph of a softmax neuron.

Входната мрежа се състои от единствен слой K неутрони, като всеки неутрон е свързан с конкретен експерт. Фигура 7.10а показва архитектурна графика на входяща мрежа, а Фиг. 7.10б показва графиката на сигналния поток на неврон k в тази мрежа. За разлика от експертите, невроните на входящата мрежа са нелинеарни, с тяхната активационна функция, дефинирана от

$$g_k = \frac{\exp(u_k)}{\sum_{j=1}^K \exp(u_j)}, \quad k = 1, 2, \dots, K \quad (7.21)$$

където u_k е вътрешният продукт на входния вектор \mathbf{x} и синаптичният тегловен вектор \mathbf{a}_k ; това е,

$$\mathbf{u}_k = \mathbf{a}_k^T \mathbf{x}, \quad k = 1, 2, \dots, K \quad (7.22)$$

„Нормализираната” експоненциална трансформация на уравнение 7.21 може да се разглежда като многовходово обобщение на логистичната функция. Тя запазва ранговата подредба на входните стойности и е диференцируемо обобщение на „победителят-взема-всичко” операция на избор на максимална стойност. По тази

причина, активационната функция на уравнение 7.21 е наричана *softmax*(Bridle, 1990a). Забележете, че линеарната зависимост на u_k на входа x прави изхода на входната мрежа нелинеарна функция на x .

За една вероятностна интерпретация на ролята на входната мрежа, може да я възприемем като „*класификатор*”, който превръща входния вектор x в полиномна вероятност, така че различните експерти да отговарят на желания отговор(Jordan and Jacobs, 1995).

Най – важното е, че използването на softmax като активационна функция за входящата мрежа ни гарантира, че тези вероятности ще отговарят на следните изисквания:

$$0 \leq g_k \leq 1 \quad \text{за всяко } k \quad (7.23)$$

и

$$\sum_{k=1}^K g_k = 1 \quad (7.24)$$

Нека с y_k означим продукцията на експерта k th в отговор на входния вектор x . Цялостното производство на МЕ модела е:

$$y = \sum_{k=1}^K g_k y_k \quad (7.25)$$

където, както посочихме по-рано, g_k е нелинеарна функция на x . Като се има предвид, че правилото k на вероятностния модел е избрано и входа е x , индивидуален продукт y_k се разглежда като условна стойност на случайната величина D , както се вижда от

$$\begin{aligned} E[D | \mathbf{x}, k] &= y_k \\ &= \mathbf{w}_k^T \mathbf{x}, \quad k = 1, 2, \dots, K \end{aligned} \quad (7.26)$$

С μ_k обозначаващо условното значение на D , можем да запишем

$$\mu_k = y_k, \quad k = 1, 2, \dots, K \quad (7.27)$$

Вариацията на D е същата като тази на грешка ϵ_k . С това, включвайки използването на уравнение 7.18, можем да напишем

$$\text{var}[D | \mathbf{x}, k] = 1, \quad k = 1, 2, \dots, K \quad (7.28)$$

Вероятностната честотна функция на D , дадения вектор x и предвид факта, че правилото k th на вероятностния генеративен модел (т.е. експерт k) е избрано, можем да напишем:

$$f_D(d | \mathbf{x}, k, \Theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(d - \mu_k)^2\right), \quad k = 1, 2, \dots, K \quad (7.29)$$

където $\boldsymbol{\theta}$ е параметричен вектор, сочещ параметрите на входната мрежа и тези на експертите в модела МЕ. Вероятностната честотна функция на D , дадения вектор x е микса на вероятностни честотни функции $\{f_D(d | \mathbf{x}, k, \boldsymbol{\theta})\}_{k=1}^K$, със смесените параметри, които са мулти номинални вероятности, определени от входната мрежа. От тук следва

$$\begin{aligned} f_D(d | \mathbf{x}, \boldsymbol{\theta}) &= \sum_{k=1}^K g_k f_D(d | \mathbf{x}, k, \boldsymbol{\theta}) \\ &= \frac{1}{\sqrt{2\pi}} \sum_{k=1}^K g_k \exp\left(-\frac{1}{2}(d - y_k)^2\right) \end{aligned} \quad (7.30)$$

Вероятностната дистрибуция на Уравнение 7.30 се нарича Асоциативен Гаузионен Смесен Модел (Gaussian mixture model). Негов неасоциативен „колега“ е традиционния Гаузионен Смесен Модел (Gaussian mixture model (Titterington et al., 1985; McLachlan and Basford, 1988), който е описан накратко в Глава 5. Един асоциативен модел се различава от неасоциативен по това, че условните средства μ_k и смесените параметри g_k не са фиксираны, а напротив, те всички са функции на x . Поради тази причина Асоциативен Гаузионен Смесен Модел на уравнение 7.30 може да се разглежда като обобщение на традиционния Гаузионен Смесен Модел.

Важните аспекти на модела МЕ, показан на Фиг. 7.8, като се предполага, че е правилно настроен по време на обучението, са:

1. Продукцията y_k на k th експерта предоставя оценка на условната стойност на случайна величина, представляваща желан отговор D , дадено x и правило k на вероятностния генеративен модел.
2. Продукцията g_k на гейтинг-мрежата определя полиномната вероятност, така че продукцията на експерт k отговаря на стойността $D = d$, въз основа на познанията, натрупани самостоятелно от x .

Работейки с вероятностната дистрибуция на уравнение 7.30 и дадения тренировъчен пример $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, проблема е да се научат условните значения $\mu_k = y_k$ и смесените параметри g_k , $k = 1, 2, \dots, K$ по оптимален начин, така че $f_D(d | \mathbf{x}, \boldsymbol{\theta})$ да дава добра оценка на основната функция за нормална честотна вероятност на средата, отговорна за генерирането на данните за обучение.

Пример 7.1 Regression Surface(Регресия на повърхността)

Взимайки предвид МЕ модела с два експерта и гейтинг мрежа с два изхода, обозначени с g_1 и g_2 . Изхода g_1 се определя (вижте ур-е 7.21)

$$\begin{aligned} g_1 &= \frac{\exp(u_1)}{\exp(u_1) + \exp(u_2)} \\ &= \frac{1}{1 + \exp(-(u_1 - u_2))} \end{aligned} \quad (7.31)$$

Нека с a_1 и a_2 обозначим двата тегловни вектора на мрежата. Тогава пишем

$$u_k = \mathbf{x}^T \mathbf{a}_k, \quad k = 1, 2$$

И за това презписваме уравнение 7.31, както следва:

$$g_1 = \frac{1}{1 + \exp(-\mathbf{x}^T(\mathbf{a}_1 - \mathbf{a}_2))} \quad (7.32)$$

Другият изход е:

$$\begin{aligned} g_2 &= 1 - g_1 \\ &= \frac{1}{1 + \exp(-\mathbf{x}^T(\mathbf{a}_2 - \mathbf{a}_1))} \end{aligned}$$

Така и g_1 и g_2 са под формата на логистична функция, но с разлика. Ориентацията на g_1 се определя от посоката на векторната разлика $(a_1 - a_2)$, а пък ориентацията на g_2 се определя от посоката на векторната разлика $(a_2 - a_1)$, която е негативна за входа g_1 . По ръба дефиниран от $a_2 = a_1$, ние имаме $g_1 = g_2 = 1/2$, така и двата експерта допринасят еднакво за производството на модела МЕ. Далече от ръба(кривата), единия от двата експерта поема доминираща роля.

7.7 Модел - Йерархична смес от експерти(HIERARCHICAL MIXTURE OF EXPERTS MODEL)

Моделът МЕ на Фиг. 7.8 работи, като разделя входното пространство на различни подпространства, с единствена гейтинг мрежа, отговаряща за разпространението на информация(събрана от обучаващите данни) на различни експерти. Йерархичната смес от експерти (HME) – моделът, илюстриран на Фиг. 7.11 се явява естествено продължение на модела МЕ. Илюстрацията е за HME модел, състоящ се от четири експерта. Архитектурата на модела HME е като дърво, на което гейтинг мрежите стоят на различни разклонения на дървото, а експертите се намират на листата на дървото. Моделът HME се различава от модела МЕ по това, че входното пространство е

разделено на вложени набори от подпространства, с информацията, комбинирана и разпространена сред експертите, под контрола на няколко стробиращи мрежи, подредени по йерархичен начин.

XME моделът на Фиг. 7.11 има две нива на йерархия или два слоя на входяща мрежа. Като продължим по принципа на „разделяй и владей” по начин, подобен на показания ние ще можем да построим HME модел с произволен брой йерархични нива. Забележете, че в съответствие с конвенцията, описана във Фиг. 7.11 номерацията на стробиращите нива започва от изходния възел на дървото.

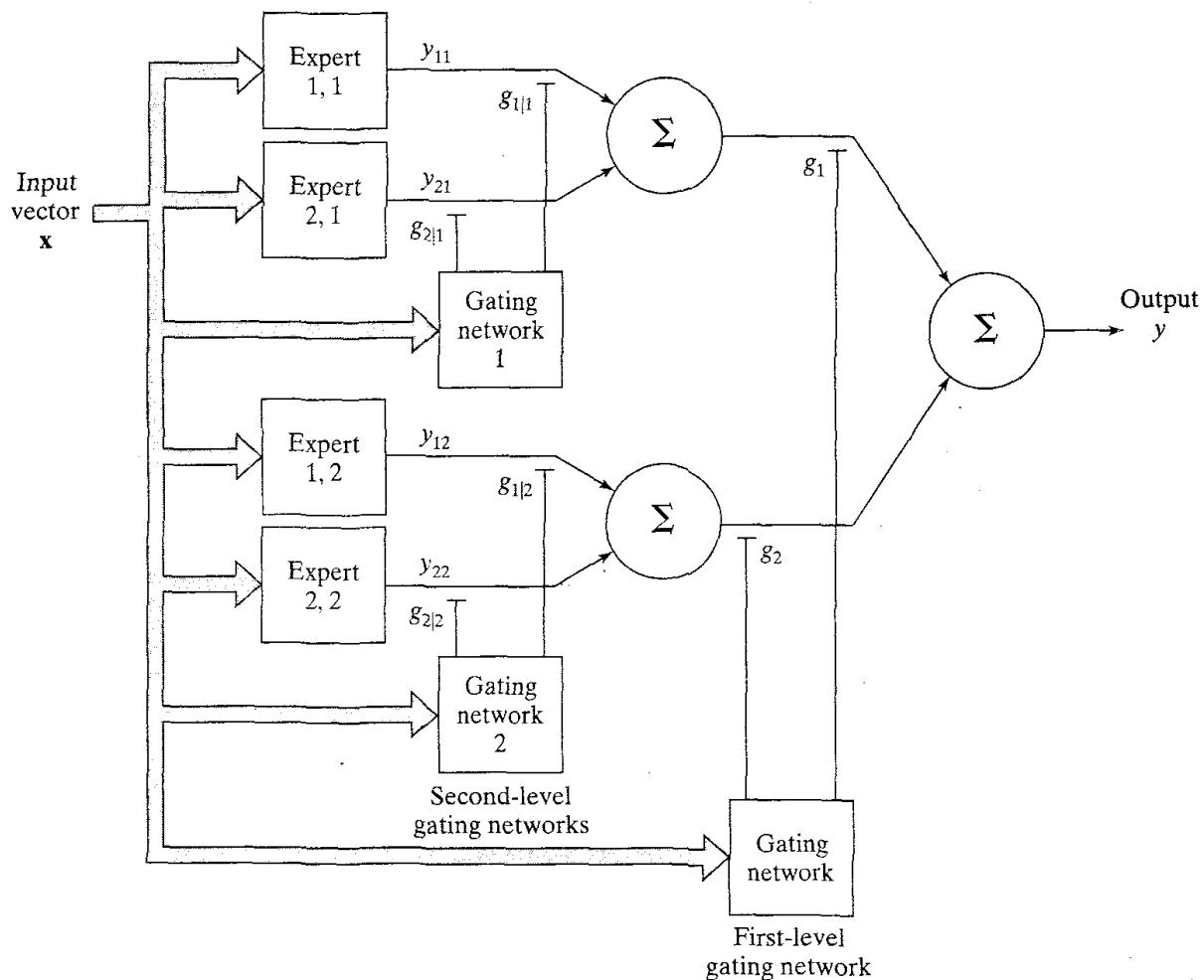


FIGURE 7.11 Hierarchical mixture of experts (HME) illustrated for two levels of hierarchy.

Формулировката на HME модела на фиг. 7.11 може да се разглежда по два начина (Jordan, 1994):

1. Моделът HME е продукт на стратегията „Разделяй и Владей”. Ако вярваме, че това е добра стратегия, да разделим входното пространство на региони, то е също толкова добра и стратегията да се разделят всеки от тези региони в подрегионы. Можем да продължим рекурсивно по този начин, докато не достигнем до етап, когато комплексността на приблизителната повърхност е

подходяща за комплексността на „местните” тренировъчни данни. НМЕ моделът следва да изпълнява най-малко, както и често по-добре от МЕ моделът поради следната причина: По-високо ниво на гейтинг мрежа в НМЕ модела ефективно комбинира информация и я разпределя сред експертите в същото поддърво, контролирано от гейтинг мрежата. Следователно всеки параметър в поддървото разделя силата с други параметри, съдържащи се в същото поддърво, което ще допринесе за възможно подобряване на цялостното изпълнение на модела НМЕ.

2. Моделът НМЕ дърво с „меки решения”. Според тази втора гледна точка, микса от експерти се състои само от едно ниво изводно дърво, понякога разглеждайки го по-скоро като изведен пън. По-общо, моделът НМЕ се разглежда като възможна рамка за изводното дърво с изходно разклонение на НМЕ модела, препращащ към корена на самото дърво. Методологията на едно стандартно изводно дърво е конструкция, която води до сложни(например да-не) решения в различни региони на входното пространство. С това се различава от дърветата с „меки решения”, извършвани от модела НМЕ. Следователно, моделът НМЕ стандартното изводно дърво по две причини:
 - Едно трудно решение неизбежно води до загуба на информацията, докато лекото решение се опитва да я запази. Например едно леко бинарно решение предава информацията за разстоянието от „границата на решението”(т.е. точката, в която решението е 0.5), докато едно трудно решение не може. По тази причина ние можем да кажем, че за разлика от стандартното изводно дърво, НМЕ модела спазва правилото за запазването на информация. Това емпирично правило посочва, че информацията, съдържаща се на входния сигнал трябва да бъде запазена по един изчислително ефективен начин, докато системата е готова за окончателно вземане на решения или параметрична оценка (Haykin, 1996).
 - Стандартните изводни дървета страдат от „алчностен” проблем. Ако веднъж е взето едно решение в такова дърво, то се замразява и никога повече не се променя. Моделът НМЕ намалява този проблем, защото решенията направени чрез това дърво постоянно се променят. За разлика от изводното дърво, в НМЕ модела е възможно възстановяване от слабо решение някъде по протежението на дървото.

Втората гледна точка, която е, че едно „меко” изводно дърво е предпочитания начин да се мисли за един НМЕ модел. С НМЕ модела, погледнат като вероятна основа за едно изводно дърво, ни позволява да изчислим вероятността за всеки даден набор от данни и да повишаме максимално тази вероятност, по отношение на параметрите, които определят разпределението между различни райони във входното пространство. По този начин, ако се базираме върху това, което вече знаем за стандартните изводни дървета, ние можем да имаме едно практическо разрешение за проблема – модел на подбор, който се обсъжда в следващия раздел.

7.8 Модел за подбор чрез стандартно дърво на решения

Както при всяка друга невронна мрежа, едно задоволително решение на параметричния оценъчен проблем зависи от подбора на подходящ модел. В случая на НМЕ модела, селекцията на модел включва избора на брой и разположение на възлите-решения в дървото. Едно практически решение за точно този модел на селекция проблем, е да се стартира стандартен алгоритъм на стандартно изводно дърво, върху обучителни данни и да се вземе дървото, като инициализираща стъпка за обучаващия алгоритъм, използван за определяне на параметрите на НМЕ модела(Jordan, 1994).

Моделът НМЕ има ясни сходства със стандартните изводни дървета, като например с класификационното и регресивното дърво(classification and regression tree (CART)) според Breiman et al.,(1984). Фигура 7.12 показва един пример на CART, където

пространството на вписаните данни \mathcal{X} е многократно разпределено от бинарна серия, разделяща се на терминални възли. Сравнявайки фиг. 7.12 с фиг.7.11, лесно можем да видим следните прилики между CART и НМЕ:

- Правилата за избор на разделителите на междинни(т.е. нетерминални) възли на CART играят аналогична роля, като входовите мрежи в НМЕ модела.
- Терминалните възли в CART играят аналогична роля като мрежата от експерти в НМЕ модела.

Като се започне с CART за класификационния или регресивния проблем на интереси, се възползваме от дискретния характер на CART за осигуряване на ефективно търсене между алтернативни дървета. С помощта на едно дърво, избрано за инициализираща стъпка в обучаващия алгоритъм за оценяване на параметрите, ние се възползваме от базовата непрекъсната вероятност на НМЕ модела, за да се получат по-добри(меки) оценки на желания отговор.

Алгоритъмът CART

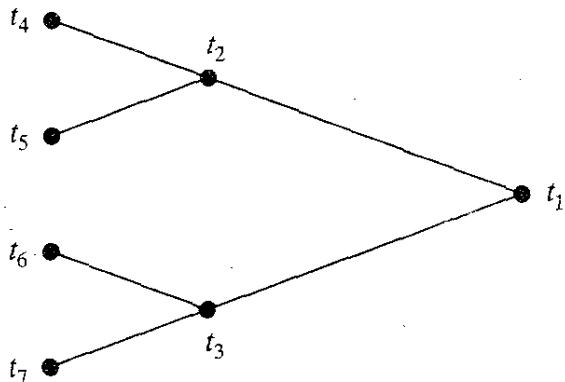
Въз основа на това, което току-що бе казано, може да се изведи кратко описание на CART алгоритъма. Описанието се представя в контекст на регресия. Започвайки с обучителни данни $\{(x_i, d_i)\}_{i=1}^N$, можем да използваме CART, за да построим бинарно дърво T за най-малка квадратна регресия, действайки както следва (Breiman et al., 1984):

1. **Избор на разделителя.** Нека един възел t обозначава едно подмножество на текущото дърво T . Нека с $\bar{d}(t)$ обозначим средната стойност на d_i за всички случаи (x_i, d_i) , попадащи под t , което е :

$$\bar{d}(t) = \frac{1}{N(t)} \sum_{x_i \in t} d_i \quad (7.33)$$

Фиг.7.12 Бинарно дърво на решенията, описано както следва:

- Възли t_2 и t_3 са наследници на възела t_1 .
- Възли t_4 и t_5 са наследници на възела t_2 ; по същия начин t_6 и t_7 са наследници на t_3 .



където сумата е над всички d_i , така че $x_i \in t$ и $N(t)$ е цялостния брой на случаи в t .

Дефинираме

$$\mathcal{E}(t) = \frac{1}{N} \sum_{x_i \in t} (d_i - \bar{d}(t))^2 \quad (7.34)$$

И

$$\mathcal{E}(T) = \sum_{t \in T} \mathcal{E}(t) \quad (7.35)$$

За възел t сумата $\sum_{x_i \in t} (d_i - \bar{d}(t))^2$ представлява „сумата от квадратите в рамките на възела”, т.е. това са общо квадратните отклонения на всички d_i в t от тяхната средна стойност $\bar{d}(t)$. Сумирайки тези отклонения над $t \in T$ дава цялостната сума от възли на квадрат и разделяйки го на N дава средната стойност.

Даден е набор от разделители S на текущо t в T , най-добрия разделител s^* е този разделител в S , който е с най-голямо понижение $\xi(T)$. За да бъдем по-точни, нека предположим, че за всеки разделител s на възел t в t_L (нов възел отляво на t) и t_R (още един нов възел, но отдясно на t), тогава

$$\Delta \mathcal{E}(s, t) = \mathcal{E}(T) - \mathcal{E}(t_L) - \mathcal{E}(t_R) \quad (7.36)$$

Най-добрия разделител s^* тогава взимаме да е точния разделител, за който имаме:

$$\Delta \mathcal{E}(s^*, t) = \max_{s \in S} \Delta \mathcal{E}(t, s) \quad (7.37)$$

Регресивно дърво, конструирано по този начин, има за цел да увеличи максимално намаляването в $\xi(T)$.

- 2. Определяне на терминален възел.** Един възел t се казва, че е терминален ако е изпълнено следното условие:

$$\max_{s \in S} \Delta \mathcal{E}(s, t) < \beta \quad (7.38)$$

където β е описано като праг(*threshold*).

- 3. Най-малкото квадратно оценяване на параметрите на терминалните възли.** Нека с t означим терминален възел в крайното бинарно дърво T и нека $X(t)$ е матрица, съставена от $x_i \in t$. Нека $d(t)$ е кореспондирящият вектор, съставен от всичките d_i в t . Дефинираме

$$w(t) = X^+(t)d(t) \quad (7.39)$$

Където X^+ е псевдо инверсията на матрицата $X(t)$. Използването на $w(t)$ ни дава най-малките квадратни оценки на $d(t)$ на изхода на терминалния възел t . С помощта на стойностите, изчислени от уравнение 7.39, разпределително-селекционния проблем е решен като търсим сумата от най-малката сума от квадратите на грешките, по отношение на регресивната повърхност, а не по отношение на средната стойност.

Използването на CART за инициализиране на НМЕ модела.

Да предположим, че алгоритъма CART се прилага по отношение на набор от данни за обучение, което води до бинарно изводно дърво на този проблем. Може да опишем разделител, получен чрез CART като мултидимензионална повърхност, определена от

$$\mathbf{a}^T \mathbf{x} + b = 0$$

където \mathbf{x} е входния вектор, \mathbf{a} е параметричен вектор и b е системна грешка. Нека вземем предвид следната ситуация в модел НМЕ. От пример 7.1 виждаме, че регресивната повърхност, получена от входната мрежа в двоично дърво може да се изрази като:

$$g = \frac{1}{1 + \exp(-(\mathbf{a}^T \mathbf{x} + b))} \quad (7.40)$$

което определя разделител, особено когато $g = 1/2$. Нека тегловния вектор(разликата) \mathbf{a} за точно тази мрежа да се запише по следния начин:

$$\mathbf{a} = \|\mathbf{a}\| \cdot \frac{\mathbf{a}}{\|\mathbf{a}\|} \quad (7.41)$$

Където $\|\mathbf{a}\|$ означава дължината(т.е. Евклидовата норма) на \mathbf{a} , и $\mathbf{a}/\|\mathbf{a}\|$ е нормализиран вектор. Използвайки Уравнение 7.41 в 7.40 можем да презапишем един параметризиран разделител на мрежата като :

$$g = \frac{1}{1 + \exp\left(-\|\mathbf{a}\| \left(\left(\frac{\mathbf{a}}{\|\mathbf{a}\|}\right)^T \mathbf{x} + \frac{b}{\|\mathbf{a}\|}\right)\right)} \quad (7.42)$$

Където виждаме, че $\mathbf{a}/\|\mathbf{a}\|$ определя **посоката** на разделителя , а $\|\mathbf{a}\|$ определя остротата на разделителя. От дискусията, разгледана в Глава 2, забелязахме, че дължината на вектора действа ефективно като реципрочна на температурата. Важната част, на която трябва да се обърне внимание в Ур-е 7.42 е, че входната мрежа, съставена от линеарен филтър, последвана от софт-макс(softmax) формата на нелинейност е в състояние да имитира разцепление в стила на CART. Освен това, имаме допълнителна степен на свобода, а именно, дължината на вектора \mathbf{a} параметъра \mathbf{a} . В едно стандартно изводно дърво, този допълнителен параметър е без значение, тъй като прага (т.е. трудното решение) се използва за създаване на разделител. От друга страна, дължината на \mathbf{a} има голямо значение за разделителната способност, произведена от мрежата в HME модела. Специално, за синаптичното векторно тегло на \mathbf{a} за фиксирана посока, може да кажем:

- Когато \mathbf{a} е дълго(т.е. температурата е ниска), разделителя(разпределението) е остър. И
- Когато \mathbf{a} е късо(т.е. температурата е висока) разделителя(разпределението) е мека.

Ако, в границата имаме $\|\mathbf{a}\| = 0$, разделителя изчезва и $g = 1/2$ и от двете страни на изчезналия(фиктивния) разделител. Ефекта от това, да се сложи $\|\mathbf{a}\| = 0$ е еквивалентно на откъсване на нетерминалните връзки от дървото, защото мрежата повече не се разделя. В най-екстремният случай, когато $\|\mathbf{a}\|$ е малко(т.е. температурата е висока) на всяка нетерминална връзка, целия HME модел се държи като единствена връзка; т.е. HME е сведен до линейно регресивен модел(вземайки предвид линейните експерти). Когато синаптичните векторни тегла на гейтинг мрежите започва да се развиват на дължина, HME започва да се разделя, като по този начин разширява възможността за свобода, предоставена на модела. По този начин може да се инициализира HME модела, както следва:

1. Прилагайки CART модела към обучителните данни.
2. Задайте синаптичните тегловни вектори на експертите в модела HME да са равни на най-малките квадратни стойности на параметричните вектори, на съответните терминални възли на бинарните дървета, в резултат на прилагането на CART.

3. За гейтинг мрежите:

- a) Задаваме синаптичните тегловни вектори да сочат в посоки, които са взаимно перпендикулярни със съответните разделители в бинарното дърво, получени от CART и
- b) Задаваме дължините(т.е. Евклидовите норми) на синаптичните тегловни вектори да са равни на малки произволни вектори.

Предварителни и Следствени Вероятности

Полиномните вероятности g_k и g_{jk} , отнасящи се до първото и второто ниво на мрежите, съответно, може да се разглеждат като предварителни(*priori*) вероятности, в смисъл, че техните стойности са изключително зависими от входния(стимулирация) вектор \mathbf{x} . По съответен начин може да се определи следствената(*posteriori*) вероятност h_{jk} и h_k , чиито стойности зависят както от входния вектор \mathbf{x} , така и от отговорите на експертите в \mathbf{x} . Последния набор от вероятности е полезен при разработването на учебните алгоритми за НМЕ моделите.

Позовавайки се на модела НМЕ на Фиг. 7.11, определяме **a** постериорните вероятности в нетерминални възли на дървото като (Jordan and Jacobs, 1994):

$$h_k = \frac{g_k \sum_{j=1}^2 g_{j|k} \exp\left(-\frac{1}{2} (d - y_{jk})^2\right)}{\sum_{k=1}^2 g_k \sum_{j=1}^2 g_{j|k} \exp\left(-\frac{1}{2} (d - y_{jk})^2\right)} \quad (7.43)$$

и

$$h_{j|k} = \frac{g_{j|k} \exp\left(-\frac{1}{2} (d - y_{jk})^2\right)}{\sum_{j=1}^2 g_{j|k} \exp\left(-\frac{1}{2} (d - y_{jk})^2\right)} \quad (7.44)$$

Продукта на h_{jk} и h_k определя съвместната **a** постериорна вероятност, която експерта (j , k) произвежда продукция y_{jk} , която съвпада с желания отговор d , като е дадено

$$\begin{aligned} h_{jk} &= h_k h_{j|k} \\ &= \frac{g_k g_{j|k} \exp\left(-\frac{1}{2} (d - y_{jk})^2\right)}{\sum_{k=1}^2 g_k \sum_{j=1}^2 g_{j|k} \exp\left(-\frac{1}{2} (d - y_{jk})^2\right)} \end{aligned} \quad (7.45)$$

Вероятността h_{jk} удовлетворява следните две условия

$$0 \leq h_{jk} \leq 1 \quad \text{за всяко } (j, k) \quad (7.46)$$

и

$$\sum_{j=1}^2 \sum_{k=1}^2 h_{jk} = 1 \quad (7.47)$$

Изводът на Уравнение (7.47) е, че кредитите се разпределят между експертите на конкурентна основа. Освен това, забелязваме от Ур. (7.45), че колкото по-близко y_{jk} е до d , толкова по-вероятно е експерта (j, k) да получава кредит за резултатите, които съвпадат с d , което интуитивно е задоволително.

Една от характерните черти на НМЕ модела, която заслужава специално внимание е рекурсивността в изчисленията, участващи в пресмятането на постериорната вероятност. При проверка на Уравненията (7.43) и (7.44) виждаме, че знаменателя в Ур-е (7.44) се появява в числителя на h_{jk} в Ур-е (7.43). В един НМЕ модел ние искаме да изчислим **a** постериорната вероятност за всеки нетерминален възел в дървото. Това е мястото, където рекурсивността е изключително важна. По – конкретно, изчисляването на последователните вероятности на всички нетерминални възли в дървото се постига с едно минаване, както е описано тук:

- При преминаване през дървото към коренния възел, ниво по ниво, **a** постериорната вероятност на всеки нетерминален възел на дървото се получава просто чрез комбиниране на постериорните вероятности на неговите „деца”.

7.10 Максимална изчислителна вероятност (Maximum Likelihood Estimation)

Обръщайки се към въпроса за параметричната оценка на НМЕ модела, първо забелязваме, че неговото възможно тълкуване е по-различно от това на модела МЕ. С НМЕ модела, формулиран като бинарно дърво се приема, че околната среда, отговорна за генерирането на данните включва вложена поредица от меки(двоични, бинарни) решения, приключващи в регресия на входния вектор x върху изхода d . В частност приемаме, че във вероятностния генеративен модел на НМЕ, решенията са моделирани като случайни величини (Jordan and Jacobs, 1994). За всеки вход x интерпретираме $g_j(x, \theta_i^0)$ като полиномна вероятност, асоциирана с първото решение и $g_{ji}(x, \theta_{ji}^0)$ като условна полиномна дистрибуция, свързана с второто решение. Както и преди, горния индекс 0 означава действителни стойности на генеративните моделни параметри. Изводите(решенията) оформят едно изводно дърво. Както и при модела МЕ, softmax се използва за активационна функция на гейтинг мрежите през модела НМЕ. По-специално, активацията g_k на k т изходния неврон в най-високото ниво на мрежата се определя от

$$g_k = \frac{\exp(u_k)}{\exp(u_1) + \exp(u_2)}, \quad k = 1, 2 \quad (7.48)$$

Където u_k е определената сума от входа, приложена към този неврон. По подобен начин за активиране на j th изходния неврон в k th гейтинг мрежата във второто ниво на юерархията се определя от

$$g_{j|k} = \frac{\exp(u_{jk})}{\exp(u_{1k}) + \exp(u_{2k})}, \quad (j, k) = 1, 2 \quad (7.49)$$

Където u_{jk} представлява определената сума от входовете, приложена по отношение на този неврон.

В интерес на презентацията ще работим с модел НМЕ само с две нива на юерархия(т.е. два слоя на гейтинг мрежи), както е показано на Фиг. 7.11. Както и при МЕ модела, всеки един от експертите в модела НМЕ се предполага, че се състои от един слой от линейни неврони. Нека с y_{jk} обозначим продукцията на експерта (j, k) . Така можем да изразим общия изход на НМЕ модела като

$$y = \sum_{k=1}^2 g_k \sum_{j=1}^2 g_{j|k} y_{jk} \quad (7.50)$$

Следвайки процедура, подобна на описаната за модела МЕ в Раздел 7.6, може да се формулира функция за вероятностна настеност на случайна величина D , представляваща желания отговор за НМЕ модела на Фиг.7.11, като се има предвид входа x , както следва:

$$f_D(d | \mathbf{x}, \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^2 g_k \sum_{j=1}^2 g_{j|k} \exp\left(-\frac{1}{2}(d - y_{jk})^2\right) \quad (7.51)$$

По този начин, за даден набор от данни за обучение , Ур-е (7.51) дефинира модел за извършеното разпространение на данните. Векторът $\boldsymbol{\theta}$ обхваща всички синаптични тегла, участващи в квалификацията на гейтинг и експертните мрежите на модела НМЕ.

Посочената вероятностна функция, обозначена с $l(\boldsymbol{\theta})$ се дава на функцията за нормална вероятностна плътност $f_D(d | \mathbf{x}, \boldsymbol{\theta})$, разглеждана като функция на параметричния вектор $\boldsymbol{\theta}$. Така пишем:

$$l(\boldsymbol{\theta}) = f_D(d | \mathbf{x}, \boldsymbol{\theta}) \quad (7.52)$$

Въпреки, че условната съставна вероятностна плътностна функция и вероятностната функция имат точно същата формула, изключително важно е да оценим физическите разлики между тях. В случая на $f_D(d | \mathbf{x}, \boldsymbol{\theta})$, входния вектор \mathbf{x} и параметричния вектор $\boldsymbol{\theta}$ са фиксираны, но желания отговор d е променлив. Въпреки това, в случая на вероятностната функция $l(\boldsymbol{\theta})$, \mathbf{x} и d са фиксираны, но $\boldsymbol{\theta}$ е променлив.

На практика откриваме, че е по-удобно да се работи с натуранният алгоритъм на вероятностната функция, а не с самата вероятностна функция. Използвайки $L(\boldsymbol{\theta})$ за обозначаване на входната вероятностна функция пишем:

$$L(\boldsymbol{\theta}) = \log[l(\boldsymbol{\theta})] = \log[f_D(d | \mathbf{x}, \boldsymbol{\theta})] \quad (7.53)$$

Логаритъмът на $l(\boldsymbol{\theta})$ е монотонна трансформация на $l(\boldsymbol{\theta})$. Това означава, че всеки път, когато $l(\boldsymbol{\theta})$ се увеличава, неговия $L(\boldsymbol{\theta})$ логаритъм също се увеличава. Тъй като $l(\boldsymbol{\theta})$ е формула за условно вероятностна плътностна функция, никога не може да стане отрицателна. Поради това следва, че няма проблеми при оценката на $L(\boldsymbol{\theta})$.

Следователно стойност 0 на параметричният вектор $\boldsymbol{\theta}$ може да бъде получена като решение на вероятностното уравнение

$$\frac{\partial}{\partial \boldsymbol{\theta}} l(\boldsymbol{\theta}) = \mathbf{0}$$

Или еквивалентно логаритично-подобно уравнение

$$\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}) = \mathbf{0} \quad (7.54)$$

Терминът „максимална оценка на вероятността“ с желаното асимптотично свойство обикновено се отнася до корена на вероятностното уравнение, който в глобален мащаб максимизира вероятностната функция $l(\boldsymbol{\theta})$. Оценката $\hat{\boldsymbol{\theta}}$ използваната в практиката, може всъщност да бъде локален максимум, но не и глобален максимум. Във всеки случай, максималната вероятност за оценка, според Фишер (1925г.) се основава на сравнително простата идея:

Различни популации генерираат различни преби на данни и всяка дадена дата преба е по-вероятно да дойде от някоя популация, отколкото от някои други.

По-специфично, неизвестният параметричен вектор $\boldsymbol{\theta}$ се изчислява по неговата най-правдоподобна стойност, т.е. входния вектор \mathbf{x} . С други думи, максималното близко изчисление $\hat{\boldsymbol{\theta}}$ е тази стойност на параметричния вектор $\boldsymbol{\theta}$, за която условна вероятностна наситена функция $f_D(d | \mathbf{x}, \boldsymbol{\theta})$ е максимална.

7.11 Стратегии за изучаване на НМЕ модела

Вероятностното описание на НМЕ модела щ раздел 7.10 ни доведе до логаритмичната функция $L(\boldsymbol{\theta})$, за целта objective функцията да бъде максимилизирана. Ключовият въпрос е как да постигнем това максимилизиране. Както и с всеки оптимизационен

проблем, няма уникален подход за максимизация на $L(\theta)$. Напротив, разполагаме с няколко подхода, два от които са обобщени тук (Jacobs and Jordan, 1991; Jordan and Jacobs, 1994,):

1. Стохастичният градиентен подход. Този подход води до он-лайн алгоритъм за максимиране на $L(\theta)$. Това е формулировка за НМЕ модел на две нива, както е изобразено на фиг. 7.11, фокусирана върху формули за следните съставки:
 - Градиент-вектора $\delta L/\delta \mathbf{w}_{jk}$ за вектора на синаптичните тегла в експерт (j,k) .
 - Градиент-вектора $\delta L/\delta \mathbf{a}_k$ за вектора на синаптичните тегла в изходния неврон k на най-високо ниво в гейтинг мрежа.
 - Градиент-вектора $\delta L/\delta \mathbf{a}_{jk}$ за вектора на синаптичните тегла в изходния неврон на второто ниво на мрежата, асоцииран с експертите (j,k) .

От значение е да се отбележи, че (вижте Плоблем 7.9):

$$\frac{\partial L}{\partial \mathbf{w}_{jk}} = h_{j|k}(n) h_k(n)(d(n) - y_{jk}(n))\mathbf{x}(n) \quad (7.55)$$

$$\frac{\partial L}{\partial \mathbf{a}_k} = (h_k(n) - g_k(n))\mathbf{x}(n) \quad (7.56)$$

$$\frac{\partial L}{\partial \mathbf{a}_{jk}} = h_k(n)(h_{j|k}(n) - g_{j|k}(n))\mathbf{x}(n) \quad (7.57)$$

Уравнение (7.55) заявява, че по време на процеса на обучение, синаптичните тегла на експерт (j,k) се коригират, за да се коригира също и грешката между изхода y_{jk} и желания отговор d , пропорционално на това експерт (j,k) произвежда една последваща вероятност h_{jk} , която съвпада с d . Уравнение (7.56) гласи, че синаптичните тегла на изходния неврон k в най-високо ниво на мрежата са коригирани, така че да притиснат априорната вероятност $g_k(n)$ да се движи към съответните апостериорни вероятности $h_k(n)$. Уравнение (7.57) посочва, че синаптичните тегла на изходния неврон на второто ниво на мрежата, асоциирани с експерт (j,k) са коригирани, за да коригират грешката м/у априорната вероятност g_{jk} и косерподиращата апостериорна вероятност в пропорционална апостериорна вероятност $h_k(n)$.

Според Уравненията (7.55) и (7.57) синаптичното тегло на модела НМЕ се актуализира след представянето на всеки модел(стимул). Чрез сумиране на градиент векторите, показани тук над п, ние можем да формулираме партидна версия на градиентно покачващия алгоритъм за максимизиране на лог-вероятностната функция $L(\theta)$.

2. *Очаквателно-Максимизационен Подход (Expectation-maximization approach).* Очаквателно-максимизационният алгоритъм (EM), според Dempster. (1977), предоставя една повтаряща се процедура за изчисляване на максималната прогноза за вероятността в ситуации, с изключение където има липсващи данни, проблема на максимално вероятностната оценка ще има ясно значение.

Алгоритъмът Ем получава името си от факта, че на всяка итерация на алгоритъма има две стъпки:

- i. *Очаквателна стъпка (Expectation step)* или *E-стъпка*, която включва наблюдателни данни, които включват непълни данни за проблем и текущата стойност на параметричния вектор за производство на данни, така че да постулираме увеличението или т.нар. Пълен набор от данни.
- ii. *Максимационната стъпка (Maximization step)* или още *M-стъпката*, която се състои от установяването на нова оценка на параметър вектора чрез максимизиране на лог-вероятността функцията на пълен набор от данни, произведени в Е-стъпката.

По този начин, като се започне с подходяща стойност за параметричният вектор, Е-и М-стъпка се повтарят на алтернативна основа до постигане на конвергенция.

Ситуациите, в които се прилага алгоритъм ЕМ включват не само тези, които съдържат естествено непълни данни, но също така и различни други ситуации, при които непълнотата на данните изобщо не е очевидна или натурална спрямо проблема на интереси. Всъщност, изчисляването на максималната оценка за вероятността често е в голяма степен улеснено чрез изкуствено формулиране на непълни данни проблеми. Това се прави, защото ЕМ алгоритъма е в състояние да се възползва от намаляването на сложността на максималната оценка на вероятността, като се има предвид пълните данни (McLachlan and Krishnan, 1997). НМЕ Моделът е един такъв пример за приложение. В този случай, липсващите данни под формата на определени променливи индикатори са изкуствено въведени в НМЕ модела, за да се улесни максимално оценката на вероятността от неизвестен вектор параметър, както е описано в раздел 7.12.

Важна характеристика на НМЕ модела, независимо дали е създаден чрез стохастичния градиентен подход или алгоритъма ЕМ, се състои от две части:

- i. Всяка гейтинг мрежа непрекъснато изчислява постериорната вероятност за всяка данна точка в обучителния набор от данни.
- ii. Корекциите, които се прилагат към синаптичните тегла на експерта и стробиращите мрежи в модела, са функции на постериорна вероятност, като по този начин се изчислява и апостериорната вероятност.

Съответно, ако експертна мрежа падне в йерархията на дървото не успее да свърши добра работа с пасването на данните за обучение в локалното обкръжение, регресивната повърхност на леяковата мрежа, намираща се по-нагоре по дървото ще бъде преместена наоколо. Това движение, от своя страна помага на експертите на следващата итерация на учебния алгоритъм, за да съвпаднат данните по-добре, като

изместват суб-пространството, в което се предполага, че трябва да извършват своето пасване на данните. Това е процеса, чрезкойто НМЕ модела е в състояние постоянно да подобрява ненаситността на проблема, присъщ на стандартно дърво на решения като КАРТ (CART).

7.12 ЕМ АЛГОРИТЪМ.

Алгоритъмът на ЕМ е забележителен от части поради простотата и всеобщността на основната теория и от части, поради широката гама от приложения, които попадат под неговия чадър. В този раздел ние представяме описание на алгоритъма ЕМ в общ смисъл. Продължаваме, за да разгледаме прилагането му към параметричният оценителен проблем в НМЕ модела в следващата секция.

Нека вектор \mathbf{z} обозначава липсващи или не наблюдавани данни. Нека \mathbf{r} обозначава пълен данни вектор, направен от някои наблюдавани данни \mathbf{d} и липсваща данни вектор \mathbf{z} . Поради това има две данни пространства R и D , да се разглежда с мапиране от R към D като *много-към-едно*. Вместо това, вместо спазване на пълен набор от данни вектор \mathbf{r} , всъщност сме само в състояние да наблюдаваме непълния данни вектор $\mathbf{d} = d(\mathbf{r})$ в D .

Нека $f_c(\mathbf{r}|\boldsymbol{\theta})$ обозначава условното pdf на \mathbf{r} и даден параметричен вектор $\boldsymbol{\theta}$. От тук следва, че условното pdf на случайната променлива D , с дадено $\boldsymbol{\theta}$ е дефинирано от

$$f_D(d|\boldsymbol{\theta}) = \int_{\mathcal{R}(d)} f_c(\mathbf{r}|\boldsymbol{\theta}) d\mathbf{r} \quad (7.58)$$

Където $\mathcal{R}(d)$ е подпространство на R определено от $d = d(\mathbf{r})$. ЕМ алгоритъмът е насочен към намирането на стойност на $\boldsymbol{\theta}$, която *максимизира непълната-данна лог-подобна функция*

$$L(\boldsymbol{\theta}) = \log f_D(d|\boldsymbol{\theta})$$

Този проблем, както и да е, е решен индиректно, работейки итеративно с *пълната-данна лог-подобна функция*

$$L_c(\boldsymbol{\theta}) = \log f_c(\mathbf{r}|\boldsymbol{\theta}) \quad (7.59)$$

която е случайната променлива, защото липсващия данни вектор \mathbf{z} е неизвестен.

За да бъдем по-специфични, нека $\hat{\boldsymbol{\theta}}(n)$ бележи стойността на параметричният вектор $\boldsymbol{\theta}$ на итерация n на ЕМ алгоритъма. В Е-стъпката на тази итерация, изчисляваме очакването

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) = E[L_c(\boldsymbol{\theta})] \quad (7.60)$$

Където очакването е извършено по отношение на $\hat{\theta}(n)$. В M-стъпката на тази итерация ние максимизираме $Q(\hat{\theta}, \hat{\theta}(n))$ по отношение на θ над параметричното (тегловно) пространство W , и така намираме обновената параметрична приблизителна оценка $\hat{\theta}(n + 1)$, както е показано

$$\hat{\theta}(n + 1) = \arg \max_{\theta} Q(\theta, \hat{\theta}(n)) \quad (7.61)$$

Алгоритъмът е стартиран с някаква първоначална стойност $\hat{\theta}(0)$ на параметричния вектор θ . E-стъпката и M-стъпката тогава се повтарят алтернативно, в съответствие с Уравнение (7.60) и (7.61), респективно, докато разликата между $L(\hat{\theta}(n + 1))$ и $L(\hat{\theta}(n))$ падне до някоя случайно малка величина; в тази точка изчислението се спира.

Забележете, че сле итерация на EM алгоритъма непълната-дата лог-вероятностна функция не е спаднала, както се вижда (вижте проблем 7.10)

$$L(\hat{\theta}(n + 1)) \geq L(\hat{\theta}(n)) \quad \text{for } n = 0, 1, 2, \dots, \quad (7.62)$$

Равенството обикновено значи, че сме на неподвижна точка на лог-вероятностната функция.

7.13 Приложение на EM алгоритъмът към НМЕ модела

След като се запознахме с EM алгоритъмът сме готови вече да решим приблизителния параметричен проблем в НМЕ модела, използвайки EM алгоритъма.

Нека $g_k^{(i)}$ и $g_{j|k}^{(i)}$ обозначават (условните) мултиноминални вероятности, асоциирани с решението, взети от първото ниво на мрежата k и от второто (j, k) ниво на НМЕ модела на Фиг. 7.11, респективно, когато действа под пример i на тренировъчните данни.

Тогава от Уравнение (7.51) лесно се вижда, че кореспондиращата на условното pdf на случайната величина D , даден пример \mathbf{x}_i и параметричен вектор θ е дадено

$$f_D(d_i | \mathbf{x}_i, \theta) = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^2 g_k^{(i)} \sum_{j=1}^2 g_{j|k}^{(i)} \exp\left(-\frac{1}{2} (d_i - y_{jk}^{(i)})^2\right) \quad (7.63)$$

Където $y_{jk}^{(i)}$ е изходният продукт на експерт (j, k) в отговор на i -тия пример на тренировъчния сет. Като се вземе в предвид, че всички N примери, съдържащи се в тренировъчния сет са статистически независими, може да формулираме log-likelihood за непълния дат проблем, както следва:

$$L(\boldsymbol{\theta}) = \log \left[\prod_{i=1}^N f_D(d_i | \mathbf{x}_i, \boldsymbol{\theta}) \right] \quad (7.64)$$

Използвайки уравнение (7.63) в (7.64) и игнорирайки константата $-(1/2)\log(2\pi)$, може да запишем

$$L(\boldsymbol{\theta}) = \sum_{i=1}^N \log \left[\sum_{k=1}^2 g_k^{(i)} \sum_{j=1}^2 g_{j|k}^{(i)} \exp \left(-\frac{1}{2}(d_i - y_{jk}^{(i)})^2 \right) \right] \quad (7.65)$$

За да изчислим максималната вероятностна приблизителна оценка на $\boldsymbol{\theta}$, трябва да намерим неподвижна точка(локален или глобален максимум) на $L(\boldsymbol{\theta})$. За съжаление, лог-вероятностната функция $L(\boldsymbol{\theta})$, както е дефинирано е Уравнение (7.65) не се поддава лесно на такъв тип изчисления.

За да преодолеем тази изчислителна трудност, ние изкуствено усилваме наблюдаваните данни $\{d_i\}_{i=1}^N$, включвайки кореспондиращ сет на липсващи данни, според EM алгоритъма. Правим това като въвеждаме *индикации променливи*, които се отнасят до вероятностния модел на НМЕ архитектурата, както следва (Jordan and Jacobs, 1994):

- iii. $z_k^{(i)}$ и $z_{j|k}^{(i)}$ са интерпретирани като етикети, които съответстват на решенията, взети в i -тия модел за вероятност, например в обучиленния сет.
Тези променливи са дефинирани по такъв начин, че само едно от $z_k^{(i)}$ е еднакво с едно и само едно от $z_{j|k}^{(i)}$ е еднакво едно за всички i . И двете $z_k^{(i)}$ и $z_{j|k}^{(i)}$ се третират като статистически независими отделни случаини променливи, с техните респективни очаквания дефинирани от

$$\begin{aligned} E[z_k^{(i)}] &= P[z_k^{(i)} = 1 | \mathbf{x}_i, d_i, \hat{\boldsymbol{\theta}}(n)] \\ &= h_k^{(i)} \end{aligned} \quad (7.66)$$

и

$$\begin{aligned} E[z_{j|k}^{(i)}] &= P[z_{j|k}^{(i)} = 1 | \mathbf{x}_i, d_i, \hat{\boldsymbol{\theta}}(n)] \\ &= h_{j|k}^{(i)} \end{aligned} \quad (7.67)$$

Където $\hat{\boldsymbol{\theta}}(n)$ е приблизителната оценка на параметричния вектор $\boldsymbol{\theta}$ при итерация n на EM алгоритъма.

- iv. $z_{jk}^{(i)} = z_{j|k}^{(i)} z_k^{(i)}$ е интерпретирано като етикета, обозначаващ експерта (j,k) във вероятностния модел за i -тия пример в тренировъчния пример.
Третирано е също като отделна случаина променлива със своето очакване, дефинирана от

$$\begin{aligned}
E[z_{jk}^{(i)}] &= E[z_{j|k}^{(i)} z_k^{(i)}] \\
&= E[z_{j|k}^{(i)}] E[z_k^{(i)}] \\
&= h_{j|k}^{(i)} h_k^{(i)} = h_{jk}^{(i)}
\end{aligned} \tag{7.68}$$

$h_k^{(i)}$, $h_{jk}^{(i)}$ и $h_{j|k}^{(i)}$ в Уравнения (7.66) до (7.68) са апостериорните вероятности, представени в Секция 7.9; горният индекс i беше добавен към тях, за да означи тренировъчния пример във въпроса. Вижте Проблем 7.13 за обяснение на тези три уравнения.

С добавянето на липсващите данни, така определените наблюдаеми данни, максималния вероятностен оценителен проблем значително се опростява. И по-специфично нека $f_c(d_i, z_{jk}^{(i)} | \mathbf{x}_i, \boldsymbol{\theta})$ обозначава условния pdf на цялостните данни, добити от d_i и $z_{jk}^{(i)}$, дадено \mathbf{x}_i и параметричен вектор $\boldsymbol{\theta}$. Тогава пишеме:

$$f_c(d_i, z_{jk}^{(i)} | \mathbf{x}_i, \boldsymbol{\theta}) = \prod_{j=1}^2 \prod_{k=1}^2 (g_k^{(i)} g_{j|k}^{(i)} f_{jk}(d_i)) \tag{7.69}$$

Където $f_{jk}(d_i)$ е условното pdf на d_i , давайки този експерт (j,k) в НМЕ модела е избран; $f_{jk}(i)$ е дадено от Гаузионната дистрибуция

$$f_{jk}(d_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} (d_i - y_{jk}^{(i)})^2\right) \tag{7.70}$$

Забележете, че формулата на Уравнение (7.69) кореспондира с хипотетичен експеримент, съдържащ индициращи променливи, представени от $z_{jk}^{(i)}$, които са ненаблюдавани във физическия смисъл на данните. Във всяко събитие, вероятностната функция за цялостния дата проблем, смятайки за цялостния обучителен сет е дадено

$$\begin{aligned}
L_c(\boldsymbol{\theta}) &= \log \left[\prod_{i=1}^N f_c(d_i, z_{jk}^{(i)} | \mathbf{x}_i, \boldsymbol{\theta}) \right] \\
&= \log \left[\prod_{i=1}^N \prod_{j=1}^2 \prod_{k=1}^2 (g_k^{(i)} g_{j|k}^{(i)} f_{jk}(d_i))^{z_{jk}^{(i)}} \right] \\
&= \sum_{i=1}^N \sum_{j=1}^2 \sum_{k=1}^2 z_{jk}^{(i)} [\log g_k^{(i)} + \log g_{j|k}^{(i)} + \log f_{jk}(d_i)]
\end{aligned} \tag{7.71}$$

Използвайки Ур. (7.70) в (7.71) и игнорирайки константата $-(1/2)\log(2\pi)$ можем да запишем

$$L_c(\boldsymbol{\theta}) = \sum_{i=1}^N \sum_{j=1}^2 \sum_{k=1}^2 z_{jk}^{(i)} \left[\log g_k^{(i)} + \log g_{j|k}^{(i)} - \frac{1}{2} (d_i - y_{jk}^{(i)})^2 \right] \tag{7.72}$$

Сравнявайки Ур.(7.72) с (7.65) ние веднага виждаме изчислителната полза, получена от добавянето на индициращи променливи като липсващия сет от данни към наблюдавания сет от данни: Максималния вероятностен приблизителен проблем е отделен в сет на регресини проблеми за индивидуалните експерти и отделен сет от мултифункционални класификационни проблеми за гейтинг мрежата.

За да продължим с прилагането на ЕМ алгоритъма, първо се позоваваме на Е-стъпката на алгоритъма, като взимаме очакването на пълната вероятностна функция за данни $L_c(\boldsymbol{\theta})$, както е показано

$$\begin{aligned} Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) &= E[L_c(\boldsymbol{\theta})] \\ &= \sum_{i=1}^N \sum_{j=1}^2 \sum_{k=1}^2 E[z_{jk}^{(i)}] \cdot \left(\log g_k^{(i)} + \log g_{j|k}^{(i)} - \frac{1}{2} (d_i - y_{jk}^{(i)})^2 \right) \end{aligned} \quad (7.73)$$

Където очакващият оператор е показан как действа на индициращата променлива $z_{jk}^{(i)}$, като единствена ненаблюдавана променлива. Следователно, използвайки Ур.(7.68) в (7.73) добиваме (Jordan and Jacobs, 1994):

$$Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) = \sum_{i=1}^N \sum_{j=1}^2 \sum_{k=1}^2 h_{jk}^{(i)} \left(\log g_k^{(i)} + \log g_{j|k}^{(i)} - \frac{1}{2} (d_i - y_{jk}^{(i)})^2 \right) \quad (7.74)$$

М-стъпката на алгоритъма изиска максимизиране на $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n))$ по отношение на $\boldsymbol{\theta}$. Параметричният вектор $\boldsymbol{\theta}$ е съставен от два сета от синаптични тегла: единия принадлежащ на гейтинг мрежата, а другия на експертите. От нашата по-ранна дискусия ние отбелязахме следното:

- v. Синаптичните тегла на експертите определени от $y_{jk}^{(i)}$, които включват също и дефиницията на $h_{jk}^{(i)}$. Израза $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n))$ е за това повлиян от експертите само чрез понятието $h_{jk}^{(i)} (d_i - y_{jk}^{(i)})^2$.
- vi. Синаптичните тегла на гейтинг мрежата определят вероятностите $g_{jk}^{(i)}$, $g_{j|k}^{(i)}$ и $h_{jk}^{(i)}$. Израза $Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n))$ е за това повлиян от гейтинг мрежите само чрез понятието $h_{jk}^{(i)} (\log g_k^{(i)} + \log g_{j|k}^{(i)})$.

Съответно М-стъпката на алгоритъма се редуцира до следните три оптимизационни проблема за НМЕ с две нива на йерархия:

$$\mathbf{w}_{jk}(n+1) = \arg \min_{\mathbf{w}_{jk}} \sum_{i=1}^N h_{jk}^{(i)} (d_i - y_{jk}^{(i)})^2 \quad (7.75)$$

$$\mathbf{a}_j(n+1) = \arg \max_{\mathbf{a}_j} \sum_{i=1}^N \sum_{k=1}^2 h_k^{(i)} \log g_k^{(i)} \quad (7.76)$$

И

$$\mathbf{a}_{jk}(n+1) = \arg \max_{\mathbf{a}_{jk}} \sum_{i=1}^N \sum_{l=1}^2 h_l^{(i)} \sum_{m=1}^2 h_m^{(i)} |l \log g_m^{(i)}|_l \quad (7.77)$$

Оптимизацията в Ур.(7.75) до (7.77) е направена с фиксирано h ; h е функция на параметрите, но производните не са взети през h . Забележете също, че всичките стойности от дясно за тези уравнения се отнасят до измерванията, направени в n стъпката.

Оптимизацията в Ур.(7.75), отнасяща се към експертите е тегловен оценителен проблем на най-малките квадрати. Останалите два оптимизации в Ур.(7.76) и (7.77) отнасящи се до гейтинг мрежите са максимални вероятностни приблизителни проблеми. Забележете също, че въпреки че ур-та са формулирани за две нива на йерархия, те могат да бъдат лесно разширени до един случаен брой на нива в йерархията.

7.14 Обобщение и Дискусия

В проучването на моделирането, моделните класификации и регресивните проблеми имаме два екстремни случая за разглеждане:

1. Прости модели, които осигуряват в проблема за интересите, но липсва точност.
2. Сложни модели, които предоставят точни резултати, но липсва проницателност.

Може би е невъзможно да се комбинира простота и точност в един единствен модел. В контекста на дискусията, представени във втората част на тази глава, CART е пример за един прост модел, който използва трудни решения за разделяне на входното пространство в мъдро разделен набор от подпространства, като всяко подпространство има собствен експерт. За съжаление, използването на твърди решения понякога довежда до загуба на информация и по този начин и загуба в производството.

Многослойният перцепtron (MLP), от друга страна, е комплексен модел с вложена форма на нелинейност, пред назначен за запазване на информацията, съдържаща се в обучителните данни. Въпреки това, той използва метода „черна кутия“ за глобално побиране на една функция в данните, като по този начин се губи погледа върху проблема. НМЕ, предсвляващ динамичен вид на комит машината(committee machine) е компромисен модел между тези две крайности, споделящи общи характеристики с CART и MLP:

- vii. Архитектурата на НМЕ е подобна като тази на CART, но се различава от него по по-мекото разделяне на входното пространство.
- viii. НМЕ използва вложена форма на нелинейност, подобна на тази на MLP, не за щелите за входно-изходното очертаване, а по-скоро за разделяне на входното пространство.

В тази глава ние наблягаме на използването на двата инструмента за дизайн на модела НМЕ:

- ix. CART като архитектурна основа за справяне с проблема за избор на модел.
- x. ЕМ алгоритъмът за решаване на проблема за оценяване на параметрите, като итеративно се изчислява максималната вероятностна оценка на параметрите на модела.

За Алгоритъма ЕМ обикновено е гарантирано придвижването му нагоре във вероятността. Ето защо, използвайки CART за инициализиране на ЕМ алгоритъма по начина описан в Секция 7.8, можем да очакваме ЕМ алг-ма да доведе до по-добро обобщено представяне, което ще бъде възможно с първоначалните условия, установени от CART.

ЕМ алгоритъма е важен и фундаментален ако апликацията от интереси е тази на максимално вероятностно оценяване, както при моделирането. Интересна моделна апликация е описана в Jacobs, Jordan и Barto (1991b), където един МЕ модел е обучен да изпълнява КАКВО/КЪДЕ – задачата. В тази задача, модела трябва да определи какво е обекта и къде се намира във видимото пространство. Два експерта бяха използвани при обучението, като всеки един от тях е специализиран за един аспект на задачата. За определен вход, двата експерта генерират изход.. Но мрежата преценява подходящия микс за този вход. Успешният резултат обявен от Jacobs демонстрира, че е възможно за възложена задача да бъде предварително определена, не върху основата на задачата сама по себе си, а по-скоро от съвпадението между изискванията на задачата и изчислителните свойства на модела(Елман и др, 1996).

Завършваме тази дискусия чрез връщане към друг клас на комитетните машини, изуцхавани в първата част на тази глава. И двата модела – МЕ и НМЕ, разчитат на използването на активността на гейтинг мрежата на входния сигнал за сливането на знанията, придобити от различните експерти в модела, а комитетната машина, базирана въз основа на употреба на цялостното усредняване или алтернативно – повишаването(boosting), разчитащо на обучителния алгоритъм, за да направи интеграция, както е обобщено тук:

1. Цялостното усредняване подобрява производствената грешка по един умен начин от комбинираното използване на два ефекта:
 - xi. Намаляване на грешката по един пристрастен начин, надвишаващ индивидуалните експерти в машината.
 - xii. Намаляване на грешката по логически път, като се използват различни начални условия в обучаването на отделни експерти, а след това цялостно осредняване на съответните им изходи.
2. Увеличаващо подобряване на изпълнението на грешка по един собствен гениален начин. В този случай индивидуалните експерти се изисква да работят

малко по-добре от случайното познаване. Слабата способност на обучение на експертите е превърнато в силно, по този начин грешките на машината стават произволно малки. Това забележително преобразуване е постигнато чрез филтриране на разпределението на входните данни по начин, който кара слабите обучаващи модели(т.е. експертите) евентуално да научат цялото разпределение, или променяйки тренировъчните примери според някои вероятностни разпределения като в AdaBoost-а. Предимството на AdaBoost над стимулация чрез филтриране е, че той работи с тренировъчна извадка с фиксиран размер.

Бележки и Референции

1. Цялостнотните осреднителни методи са разгледани в Perrone(1993), където е включена голяма библиография по темата. Други препратки по тази тема са включни при Wolpert(1992) и Hashem (1997).
2. Използването на цялостно усреднителен дизайн на една машина над набор от различни случайни условия е предложено от няколко практици на невронни мрежи. Въпреки това, статистически анализ, представен в Naftaly и др. (1997) и процедурата, описана в него за обучение на комитета машина, проектирана от цялостно усредняване над пространството на първоначалните условия изглежда, че е първо по рода си. В този документ са представени експериментални резултати въз основа на sunspot данни и енергийно-предричащи конкурентни данни. И в двата случая значително намаляване на вариацията се демонстрира чрез усредняване над пространството на първоначалните условия.

Според Naftaly и др. (1997), използването на популярни обучителни ограничения, каквито са тегловното разпадане и ранното спиране, не са препоръчителни при проектирането на комитетна машина от цялостно осредняване над пространството на първоначалните условия.

3. Оновните препоръки на повишаващата теория и експерименталните изследвания, повече или по-малко в хронологичен ред са както следва: Schapire(1990), Drucker и др.(1993,1994), Freund(1995), Breiman(1996b), Freund и Schapire(1996a, 1996b, 1997), Schapire(1997) и Schapire и др.(1997). Първите референции за трите основни подхода за повишаване са, както следва:
 - xiii. Filtering(Филтриране): Schapire (1990)
 - xiv. Resampling: Freund и Schapire(1996a)
 - xv. Reweighting: Freund(1995)
4. Идеята за използване на смес от експерти за реализиране на сложна функция за преобразуване е обсъдена от Jacobs, Jordan, Nowlan и Hinton в тяхните книжа от 1991а. Разработването на този модел бе мотивирано от (1) предложение, описано

в Nowlan(1990), разглеждайки конкурентната адаптация в неконтролирано обучение като опит да се вмести в смес от прости вероятностни дистрибуции(например като Gaussians) в сет от дата точки и (2) идеи, разработени в Ph.D. тезата на Jacobs (1990), използвайки подобна модуларна архитектура, но различна функция на разходите.

5. Максималните вероятностни оценители имат някои желани свойства. При твърде общи условия, следните асимптотични свойства могат да бъдат доказани (Kmenta, 1971):
 - i. *Максималните вероятностни оценители са съвместими.* Нека $L(\boldsymbol{\theta})$ обозначава лог-вероятностната функция, а $\boldsymbol{\theta}_i$ обозначава елемент от параметричния вектор $\boldsymbol{\theta}$. Частичните производни $\delta L / \delta \theta_i$ са наричани резултат. Казваме, че максимално вероятностния оценител е последователен в смисъл, че стойността на $\boldsymbol{\theta}_i$, за който резултата $\delta L / \delta \theta_i$ е идентично нула, клони вероятно към истинската стойност на $\boldsymbol{\theta}_i$, доколкото размера на извадката, използвана при оценителния подход клони към безкрайност.
 - ii. *Максималните вероятностни оценители са асимптотично ефективни.* Това е,

$$\lim_{N \rightarrow \infty} \left\{ \frac{\text{var}[\hat{\theta}_i - \hat{\theta}_i]}{I_{ii}} \right\} = 1 \quad \text{for all } i$$

Където N е размера на извадката, $\hat{\theta}_i$ е максималната вероятностна оценка на $\boldsymbol{\theta}_i$, а I_{ii} е и-тия диагонален елемент на инверсията на *Информационната Матрица на Фишер*. Информационната матрица на Фишер е дефинирана от

$$\mathbf{J} = - \begin{bmatrix} E\left[\frac{\partial^2 L}{\partial \theta_1^2} \right] & E\left[\frac{\partial^2 L}{\partial \theta_1 \partial \theta_2} \right] & \dots & E\left[\frac{\partial^2 L}{\partial \theta_1 \partial \theta_M} \right] \\ E\left[\frac{\partial^2 L}{\partial \theta_2 \partial \theta_1} \right] & E\left[\frac{\partial^2 L}{\partial \theta_2^2} \right] & \dots & E\left[\frac{\partial^2 L}{\partial \theta_2 \partial \theta_M} \right] \\ \vdots & \vdots & & \vdots \\ E\left[\frac{\partial^2 L}{\partial \theta_M \partial \theta_1} \right] & E\left[\frac{\partial^2 L}{\partial \theta_M \partial \theta_2} \right] & \dots & E\left[\frac{\partial^2 L}{\partial \theta_M^2} \right] \end{bmatrix}$$

Където M е размера на параметричният вектор $\boldsymbol{\theta}$.

- iii. *Максималните вероятностни оценители са асимптотично Гаузионни(Gaussian).* Това е, когато размерът на извадката се приближава до безкрайност, всеки елемент на максималната вероятностна оценка $\boldsymbol{\theta}$ приема Гаузионно разпределение.

В практиката ние откриваме, че големите извадкови(т.е. асимптоматичните) свойства на максималната вероятностна оценка се държат по-добре, когато размера на извадката е $N \geq 50$.

6. Трудът на Newcomb (1886г.), вземайки предвид оценяването на параметрите ма смесицата от две различни Гаузионни дистрибуции, се явява като първа референция към ЕМ-типа на процес, обявен в литературата.

Изразът „ЕМ алгоритъм“ беше измислен от Dempster, Laird и Rubin в техният фундаментален труд от 1977г. В този труд, формулировка за ЕМ алгоритъма за изчисление на максималната вероятностна оценка от непълни данни на различни нива беше представена за първи път.

В първият обединен акаунт на теорията, методологията и приложението на ЕМ алгоритъма, неговата история и екзистенция бе представена под формата на книга от McLachlan и Krishnan(1997г.).

7. Под сравнително генерални условия, вероятностните стойности на ЕМ алгоритъма достигат до постоянни стойности. Wu (1983г.) представя един детайлрен акаунт на конвергенционните свойства на ЕМ алгоритъма. Обаче ЕМ алгоритъмът няма винаго да довежда до локален или глобален максимум на жероятностната функция. В Зта глава на тази книга са представени два примера от McLachlan и Krishnan(1997г.), където това не е случая. В единият пример алгоритъма достига до седалищна точка, а в другия достига до локален минимум на вероятностната функция.

8. ЕМ алгоритъма може също да оперира с Байесионно (Bayesian) максимално апостериорно пресмятане, включвайки приорна информация на параметричния вектор; вижте Проблем 7.11. Използвайки правилото на Байес, можем да изразим условната вероятностна наситена функция за параметричния вектор θ , даден сет от наблюдавани x , като

$$f_{\theta}(\theta|x) = \frac{f_x(x|\theta)f_{\theta}(\theta)}{f_x(x)}$$

От тази връзка ние ясноможем да видим, яе максимизирайки апостериорната наситеност $f_{\theta}(\theta|x)$ е еквивалентно на максимизирането на продукта $f_x(x|\theta)f_{\theta}(\theta)$, тъй като $f_x(x)$ е независимо от θ . Вероятностната наситена функция $f_{\theta}(\theta)$ представя приорната информация достъпна за вектор θ . Максимизирайки $f_{\theta}(\theta|x)$ ни предоставя най-вероятната оценка на параметричния вектор θ с дадено x . Две точки за важни да се отбележат в контекста на това оценяване:

- Максимално вероятностно оценяване, представено чрез максимизиране на $f_x(x|\theta)$ по отношение на θ , е редуцирана форма на максимално апостериорно оценяване, редуцирана в смисъл , че то е нищожно от гледна точка на предварителната информация(prior information).

- Употребата на приорна информация е синоним на узаконяване, което (да се върнем за малко към глава 5) кореспондира с едно гладко входно-изходно мапиране.

В Waterhouse(1996), един Байсенов framework за оценяване на параметрите на микс от експертни модели е представен. В Байсеновит подход, описан вътре се достига до един феномен, наречен “overfitting”, което води до оценяване с висока вариация, когато се използва максимален вероятностен извод.

9. Един ефективен алгоритъм, познат като *итеративен „преизчислен“ на най-малките квадрати алгоритъм* (*iteratively reweighted least-squares(IRLS) algorithm*), е способен да разреши максималния жероятностен изчислителен проблем, описан в Уравнения (7.76) и (7.77); за описание на IRLS алгоритъма, погледнете McCullagh и Nelder (1989г.).

Проблеми

Общо усредняване

7.1 Нека разгледаме една комитетна машина, съдържаща K експерта. Входно – изходната функция на k-тия експерт е отбелязана с $F_k(\mathbf{x})$, където \mathbf{x} е входния вектор и $k = 1, 2, \dots, K$. Индивидуалните изходи на експертите са линеарно комбинирани, за да формират общия изход y , дефиниран от

$$y = \sum_{k=1}^K w_k F_k(\mathbf{x})$$

Където w_k е линеарно тегло присвоено на $F_k(\mathbf{x})$. Изискването е да се оцени така w_k , че y да представи оценка за нак-малките квадрати на желания отговор d според x .

Давайки тренировъчен сет от обучителни данни $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ се определя изискваните стойности на w_k -то, за да реши този параметричен оценителен проблем.

Повишаване (Boosting)

7.2 Сравнете изчислителните предимства и недостатъци на *boosting by filtering* и AdaBoost.

7.3 Обикновено, буустването действа най-добре на слаби обучителни модели, това са обучителните модели, които имат релативно нисък генерализационен коефициент на грешка. Да предположим обаче, че ни е даден силен обучителен модел, който е с висок коефициент на грешка. И че ние даден тренировъчен пример с фиксиран размер. Как *boosting by filtering* и AdaBoost ще се справят с тази ситуация.

Микс от Експерти

7.4 Имаме даден пример на линеарна задача

$$F(x_1, x_2, \dots, x_{10}) = \begin{cases} 3x_2 + 2x_3 + x_4 + 3 + \epsilon & \text{if } x_1 = 1 \\ 3x_5 + 2x_6 + x_7 - 3 + \epsilon & \text{if } x_1 = -1 \end{cases}$$

За сравнение, използвани са следните мрежови конфигурации:

- | | |
|----------------------------|--|
| 1. Многослойни перцептрони | ,,10→10→1” мрежа |
| 2. Микс от експерти: | Гейтинг мрежи : 10→2;
Експертни мрежи: 10→1 |

Сравнете изчислителната комплексност на тези две мрежи.

7.5 МЕ модела, описан от условната вероятностна наситена функция в Уравнение (7.30) е базиран на скаларен регресионен модел, в който грешката е Гаузионно разпределена с 0-ва стойност и единична вариация.

- (a) Преформулирайте уравнението за по-генерален случай на МЕ модел, кореспондирайки с множество регресионни модели, за които желаният отговор е вектор с димензия q и грешката е многовариантна Гаузионна дистрибуция с нулево значение и ковариационна матрица Σ .
- (b) Как е МЕ моделът за тази реформулировка различен от МЕ модела, показан на Фиг.7.8 ?

7.6 Намерете стохастичният градиентен алгоритъм за тренирането на модела – микс от експерти.

Йерархичен Микс от Експерти

7.7 (a) постройте блок-диаграма на един НМЕ модел с три нива на йерархия. Дадено е, че начина на решения в дървото е бинарен.

(b) Напишете а-постериорната вероятност за нетерминалните възли на НМЕ, описан в подточка (a). Демонстрирайте рекурсивността на изчисленията, включени в оценяването на тези вероятности.

(c) Формулирайте условната вероятностна наситена функция за НМЕ модела, описан в подточка (a).

7.8 Обсъдете подобностите и разликите между НМЕ модела и радиално-базираните функционни (radial-basis function RBF) мрежи.

7.9 Намерете уравненията, които описват стохастичният градиентен алгоритъм за тренирането на един НМЕ модел с две нива на юерархия. Нека предположим, че дървото е бинарно.

ЕМ Алгоритъма и Приложението му към НМЕ Модела

7.10 Докажете монотонно нарастващата стойност на ЕМ алгоритъма, описан в Ур-е (7.62). За намирането направете следното:

(a) Нека

$$k(\mathbf{r}|d, \boldsymbol{\theta}) = \frac{f_c(\mathbf{r}|\boldsymbol{\theta})}{f_D(d|\boldsymbol{\theta})}$$

Обозначава условната вероятностна наситена функция на увеличенията данни вектор \mathbf{r} , дадено наблюдение d и параметричен вектор $\boldsymbol{\theta}$. Следователно, непълната данна лог-подобна функция може да се опише по следния начин

$$L(\boldsymbol{\theta}) = L_c(\boldsymbol{\theta}) - \log k(\mathbf{r}|d, \boldsymbol{\theta})$$

Където $L_c(\boldsymbol{\theta}) = \log f_c(\mathbf{r}|\boldsymbol{\theta})$ е пълната данна лог-вероятностна функция. Вземайки очакваното от $L(\boldsymbol{\theta})$ с отношение към условната дистрибуция на \mathbf{r} , дадено d , показва че

$$L(\boldsymbol{\theta}) = Q(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) - K(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n))$$

където

$$K(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}(n)) = E[\log k(\mathbf{r}|d, \hat{\boldsymbol{\theta}})]$$

Следователно

$$\begin{aligned} L(\hat{\boldsymbol{\theta}}(n+1)) - L(\hat{\boldsymbol{\theta}}(n)) &= [Q(\hat{\boldsymbol{\theta}}(n+1), \hat{\boldsymbol{\theta}}(n)) - Q(\hat{\boldsymbol{\theta}}(n), \hat{\boldsymbol{\theta}}(n))] \\ &\quad - [K(\hat{\boldsymbol{\theta}}(n+1), \hat{\boldsymbol{\theta}}(n)) - K(\hat{\boldsymbol{\theta}}(n), \hat{\boldsymbol{\theta}}(n))] \end{aligned}$$

(b) Неравенството на Дженсен (*Jensen's inequality*) казва, че ако $f(\cdot)$ е изпъкната функция и u е случайна стойност, тогава

$$E[g(u)] \geq g(E[u])$$

където E е очакваниия оператор; още повече, ако $f(\cdot)$ е строго изпъкната, тогава равенството в тази връзка предполага, че $u = E[u]$ с вероятност 1 (Cover и Thomas, 1991г.). Използвайки Неравенството на Дженсен, показваме че

$$K(\hat{\theta}(n+1), \hat{\theta}(n)) - K(\hat{\theta}(n), \hat{\theta}(n)) \leq 0$$

следователно показва, че Ур-е (7.62) се отнася като $n = 0, 1, 2, \dots$.

7.11 ЕМ модела е лесно модифицируем, за да поеме максималната апостериорна оценка на даден параметричен вектор θ . Използвайки правилото на Байе, модифицирайте Е – стъпката о М – стъпката на ЕМ алгоритъма, за пресмуштането на това оценяване.

7.12 За НМЕ трениран с ЕМ алгоритъм и един MLP трениран с един back-propagation алгоритъм за достигане на подобно ниво на представяне за дадена задача, ние интуитивно ще очакваме изчислителната комплексност на НМЕ да надделее над тази на MLP. Аргументирайте в услуга или против правдоподобността на това твърдение.

7.14 Уравнение (7.75) описва измерените най-малки квадрати за оптимизация на експертната мрежа в НМЕ модела на Фиг.7.11, взимайки под предвид, че желаништ отговор d скаларен. Как е модифицирана тази връзка за случая мултидимензионален желан отговор.

Глава 8

Анализ на основните компоненти

8.1 Въведение

Важна особеност на невронните мрежи е способността да се учат от заобикалящата ги среда и чрез изучаване да подобряват представянето си в някакъв смисъл. В предишните четири глави, фокъсът беше върху алгоритъма за контролното обучение, за който сет от цели бе осигурен от страничен учител. Целите взимаха форма на желано входно-изходно мапиране, които мрежата трябваше да направи приблизителни. В тази глава и следващите три ще изучаваме алгоритми за *самоорганизирано обучение* или без *надзорно обучение*. Целта на такъв алгоритъм е да открие значителни модели или особености във водните данни и да направи откриването без учител. За да направи това, на алгоритъма е дадено набор от правила от локално естество, което му позволява да научи да изчислява входно-изходно мапиране със специфични желани свойства; терминът „локален“ означава, че промените приложени на синаптичното тегло на един неврон е ограничен до най-близкото обкръжение на този неврон. Моделирането на мрежовата структура, използвано за самоорганизирано обучение тенденцира към следването на невро-биологичните структури, клонящи към много по-голямо разширяване, отколкото за обучението под надзор. Това може да не е изненадващо, защото процеса на мрежова организация е базирана на организацията на мозъка.

Структурата на самоорганизираща система може да варира в различни форми. Може, например, да се състои от входен(дата) слой и един изходен (представителен) слой, със напредваща връзка от входа към изхода и латерална връзка между невроните в изходния слой. Друг пример е мрежа с множество слоеве, в която самоорганизацията протича от слой-към-слой основа. И в двата примера обучителният процес се състои от повтарящо се модифициране на синаптичните тегла на всички връзки в системата, в отговор на входния(активиращ) модел и в съчетание с въведените правила, докато не се изгради крайна конфигурация.

Тази глава на самоорганизационна система е насочена към Хебиан обучението(Hebbian learning). Основният фокус е *анализ на основните компоненти*, което е стандартна техника, често използвана за намаляване на данните в статистичен разпознаваем модел и за обработка на данните.

Организация на Главата

Материала в главата е организиран както следва. В Секция 8.2 използваме качествени аргументи, за да опишем основните принципи на самоорганизацията. Това е последвано от материал за запознаване с основните компонентни анализи в Секция 8.3, която е също базисна за самоорганизационната система, обсъждана в останалата част от главата.

С този базисен фонов материал на ръка ще продължим да изучаваме някои специфични самоорганизационни системи. В Секция 8.4 ще опишем прост модел, състоящ се от един неврон, който извежда първия главен компонент в самоорганизационния маниер. В Секция 8.5 описваме една по-сложна самоорганизационна система, под формата на „хранеща-напред” мрежа със единствен слой неврони, която извежда всички главни компоненти, основавайки се на предходния прост модел. Тази процедура е илюстрирана чрез компютърен експеримент изобразяващ кодиране в Секция 8.6. В Секция 8.7 описваме друга самоорганизираща се система за подобна функция; тази система е дори още по-сложна, защото включва и странични връзки.

В Секция 8.8 представяме класификация от алгоритми за анализ на основните компоненти, използвайки невронни мрежи. Това е последвано от Секция 8.9 с класификация на алгоритми за намаляване на данните в адаптивни и партидни методи.

В Секция 8.10 описваме нелинейна форма на анализа на главните компоненти, която се основава на идеята за вътрешно продуктивно ядро, определено според теоремата на Мерцер(Mercer), която е обсъдена в глава бта.

Главата завършва в Секция 8.11 с някои мисли за анализа на основните компоненти.

8.2 Няколко Интуитивни Принципи на Самоорганизацията

Както бе споменато по-рано, самостоятелно организираното (без надзор) обучение се състои от многократно модифициране на синаптичните тегла на невронна мрежа, в отговор на активационните модели и в съответствие с определените правила, докато не се развие окончателна конфигурация. Ключовият въпрос, разбира се е как една полезна конфигурация може да се развие от само-организация. Отговорът се крие най-вече в следното наблюдение(Тюринг, 1952г.):

Глобалният ред може да възникне от местни взаимодействия.

Това наблюдение е от основно значение; то се прилага към мозъка и изкуствените мрежи. И по-специално, много първоначално случайни местни взаимодействия между съседните неврони на мрежата могат да се съединяват в състояние на глобален ред и в

крайна сметка да доведе до съгласувано поведение, под формата на пространствени модели или временни ритми, които са същността на самоорганизацията.

Мрежовата организация се извършва на две различни нива, които си взаимодействват едно с друго чрез обратна връзка. Двете нива са:

- *Дейност(activity)*. Някои дейностни модели са произведени от дадена мрежа, в отговор на входните сигнали.
- *Свързване(connectivity)*. Силата на връзките(синаптичните тегла) на мрежата са модифицирани в отговор на невронните сигнали в дейностните модели, поради синаптичната пластичност.

Обратната връзка между промените в синаптичните тегла и промените в activity моделите трябва да бъде положителна, за да достигне самоорганизация (вместо стабилизация) на мрежата. Съответно, можем да изведем първият принцип на самоорганизация(von der Malsburg, 1990a):

Принцип 1. Модификациите в синаптичните тегла са склонни към само-разширяване(усилване).

Процесът на само-разширяване е ограничен от изискването, че модификациите в синаптичните тегла трябва да се основават на локални сигнали, а именно *пресинаптични и постсинаптични сигнали*. Изискванията за само-подсилване и локалност определят механизма, където силен синапс води до съвпадение на пресинаптичните и постсинаптични сигнали. На свой ред синапса се увеличава в силата от такова съвпадение. Механизмът, описан тук всъщност е преформулиран постулат на Хеб(Hebb) за обучението.

С оглед за стабилизирането на системата трябва да има някаква форма на конкуренция за „ограничените“ ресурси (например, броя на входовете, енергийните ресурси). Конкретно, едно усилване на силата на някои синапси в мрежата трябва да бъде компенсирано с намаляването в други. Съответно само „успелите“ синапси могат да растат, докато по-малко успелите да отслабват и в крайна сметка да изчезнат. Това твърдение ни води до извеждане на втория принцип на само-организация(von der Malsburg, 1990a):

Принцип 2. Ограничението на ресурси води до конкуренция между синапсите и следователно избора на най-енергично растящите синапси(т.е. по-силните) става за сметка на останалите.

Този принцип също е направен възможен от синаптичната пластичност.

За нашето следващо наблюдение ще отбележим, че един синапс сам по себе си не може ефикасно да произведе благоприятни събития. За да направим това, имаме нужда от сътрудничество между набор от синапси, сближавайки се до даден неврон и носейки съвпадащи сигнали, достатъчно силни за активирането на този неврон. От тук извличаме и третия принцип на само-организацията (von der Malsburg, 1990a):

Принцип 3. Модификациите в синаптичните тегла са склонни към сътрудничество.

Наличието на мощен синапс може да подобри пригодността на други синапси, въпреки цялостната конкуренция в мрежата. Тази форма на сътрудничество може да възникне поради синаптичната пластичност, или поради едновременно стимулиране на пресинаптичните неврони, донесени от наличието на подходящи условия във външната среда.

И трите принципа на само-организацията, описани по този начин моментално се отнасят само до самата невронна мрежа. Въпреки това, за самоорганизираното обучение да може да извърши полезно обработваща информацията функция, трябва да има съкращения в активационните модели, доставени на мрежата от околната среда. Въпросът за излишъка се обсъжда в Shannon's framework за теорията на информацията в Глава 10. За сега е достатъчно да постулираме последният принцип за самоорганизация на обучението, както следва (Барлоу, 1989):

Принцип 4. Реда и структурата в активационните модели представят излишна информация, която се придобива чрез невронната мрежа под формата на знания, което е необходима предпоставка за самоорганизация на обучението.

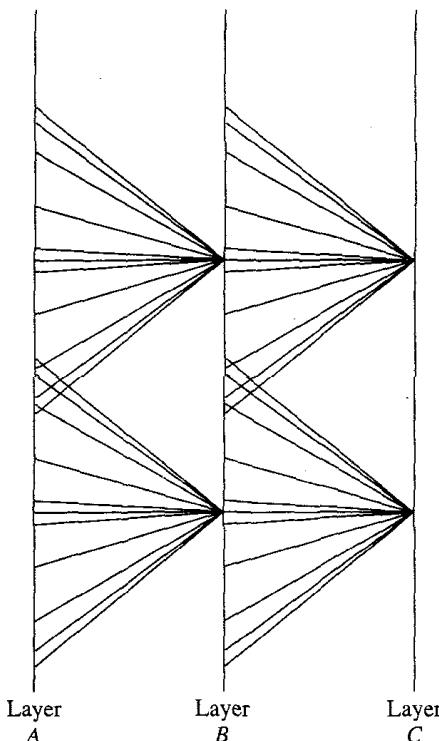


FIGURE 8.1 Layout of modular self-adaptive network.

Част от това знание може да бъде получено от наблюдение на статистическите параметри, както са средната стойност, дисперсията и корелационната матрица на входните данни. Принципите от 1 до 4 на самоорганизация на обучението предоставят невробиологичната основа за адаптивните алгоритми за анализ на основните компоненти, описани в тази глава и за самоорганизираната карта на Кохонен(Kohonen), предоставени в следващата глава. Тези принципи са включени също и в много други самоорганизирани модели, мотивирани от невробиологични съображения. Един такъв модел, който заслужава да бъде споменат е модела за зрителната система на бозайниците на Линскер(Linsker) (Linsker, 1986).

Самоорганизационен Функционен Анализ

Обработката на информацията в зрителната система се извършва на етапи. И по-специално, прости функции като контраст и ъглова ориентация се анализират в ранните етапи на системата, докато по-сложни комплекси характеристики се анализират в по-късните етапи. Фигура 8.1 показва груба структура на модулна мрежа, която прилича на визуална система. В модела на Линскер, невроните на мрежата на фиг.8.1 са организирани в двуизмерни слоеве, с локални препращащи връзки от един слой към следващ. Всеки неврон получава информация от ограничен брой неврони, намиращи се в регион, лежащ над предишния, който представлява възприеманото поле на този неврон. Възприемните области на мрежата играят ключова роля в развитието на синаптичния процес, защото те правят възможно невроните от един слой да отговорят на пространствените корелации на невронните дейности в предишния слой. Направени са две предположения от структурен характер:

1. Позициите на синаптичните връзки са фиксираны за цялостното развитие на невронното развитие, веднъж след като са били избрани.
2. Всеки неврон действа като линеен комбинатор.

Моделът съчетава аспекти на Хеб-подобни синаптични модификации с кооперативно и конкурентно обучение, по такъв начин, че изходите на мрежата оптимално да дискриминира сред цялостта от входове, като самоорганизационната процедура на обучение е на база слой-по-слой. Това означава, че обучителният процес позволява самоорганизационните функционални аналитични свойства на всеки слой да се развият напълно, преди да продължи към следващия слой. В Линскер(1986) с представени симулационни резултати, които са качествено сходни със свойствата, открити в ранните етапи на визуалната обработка при котките имаймуните. Отчитайки изключително сложният характер на зрителната система, наистина е забележително това, че простият модел на Линскер е в състояние да развие подобни анализиращи функционни неврони. Смисълът не е да се предположи, че анализационните функции на неврона в зрителната система на бозайниците се развива точно по начина, описан в модела на Линскер.Напротив, такива структури могат да бъдат произведени от една сравнително прости пластова мрежа, чиито синаптични връзки се развиват в

съответствие с формата за обучение на Хеббиян. Основният ни интерес в тази глава, обаче, е в анализа на основните компоненти и как той може да се извърши чрез самоорганизиращи се системи на базата на Хеббияновото обучение.

8.3 Основен Компонентен Анализ

Често срещан проблем в статистическото моделно разпознаване на образи е избора на функция или извлечането на функция. Функционалният избор се отнася до процес, при който пространството на данните се превръща в характеристично пространство и на теория имат точно същите размери. Трансформацията, обаче, е проектирана по такъв начин, че посочените данни могат да бъдат представени от намален брой „ефективни“ функции и все още да запазват повечето от присъщото информационно съдържание на данните. С други думи данните претърпяват намаляване на размера. За да бъдем по-конкретни нека предположим, че имаме даден m -мерен вектор \mathbf{x} и искаме да го превърнем, използвайки l числа, където $l < m$. Ако просто намалим размера на \mathbf{x} ще предизвикаме следната средно-квадратна грешка, равна на сумата от дисперсията на елементите, елиминирани от \mathbf{x} . Задаваме следният въпрос: Съществува ли обратима линейна трансформация T , такава, че отразяването на $T\mathbf{x}$ е оптимално в смисъла на средната квадратна грешка? Ясно е, че трансформацията T трябва да има свойството, при което някои от нейните компоненти да имат ниска дисперсия. Основният компонентен анализ (известен още като Трансформацията на Karhunen-Loeve в комуникационната теория) максимизира скоростта на намаляване на вариацията и поради това е правилния избор. В тази глава ще изведем алгоритъмът за обучение на Хеббиян, който може да извършва анализ на основните компоненти върху дата вектор на интереси.

Нека \mathbf{X} обозначи m -мерен случаен вектор, представляващ околната среда от интереси. Предполагаме, че случаен вектор \mathbf{X} има за стойност 0:

$$E[\mathbf{X}] = 0$$

Където E е статистическия очакван оператор. Ако \mathbf{X} има ненулева стойност, то ние изваждаме стойността от него, преди да продължим с анализа. Нека с \mathbf{q} да отбележим единичен вектор, също от m -димензията, върху който вектора \mathbf{X} да се прогнозира. Тази прогноза се определя от вътрешният продукт на векторите \mathbf{X} и \mathbf{q} , както е показано

$$A = \mathbf{X}^T \mathbf{q} = \mathbf{q}^T \mathbf{X} \quad (8.1)$$

подлежащо на ограничението

$$\|\mathbf{q}\| = (\mathbf{q}^T \mathbf{q})^{1/2} = 1 \quad (8.2)$$

Проекцията A е случайна променлива със средна стойност и отклонения, свързани със статистиката на случаяния вектор \mathbf{X} . Според допускането, че случаяният вектор \mathbf{X} е нула следва, че средната стойност на проекцията A е също нула:

$$E[A] = \mathbf{q}^T E[\mathbf{X}] = 0$$

За това и дисперсията на A е същата като средната квадратна стойност, следователно можем да запишем

$$\begin{aligned}\sigma^2 &= E[A^2] \\ &= E[(\mathbf{q}^T \mathbf{X})(\mathbf{X}^T \mathbf{q})] \\ &= \mathbf{q}^T E[\mathbf{X} \mathbf{X}^T] \mathbf{q} \\ &= \mathbf{q}^T \mathbf{R} \mathbf{q}\end{aligned}\tag{8.3}$$

Матрицата $m \times m$ \mathbf{R} е съответната матрица на случайния вектор \mathbf{X} , официално определена като очаквания външен продукт на вектора \mathbf{X} сам със себе си, както се вижда от

$$\mathbf{R} = E[\mathbf{X} \mathbf{X}^T]\tag{8.4}$$

Забелязваме, че съответната матрица \mathbf{R} е симетрична, което значи

$$\mathbf{R}^T = \mathbf{R}\tag{8.5}$$

От това свойство следва, че ако a и b са който и да било вектор $m \times 1$, то

$$\mathbf{a}^T \mathbf{R} \mathbf{b} = \mathbf{b}^T \mathbf{R} \mathbf{a}\tag{8.6}$$

От Ур-е (8.3) виждаме, че вариацията σ^2 на проекцията A е функция на единичния вектор \mathbf{q} ; за това можем да запишем

$$\begin{aligned}\psi(\mathbf{q}) &= \sigma^2 \\ &= \mathbf{q}^T \mathbf{R} \mathbf{q}\end{aligned}\tag{8.7}$$

На базата, на което за $\psi(\mathbf{q})$ можем да мислим като за вариационна проба.

Eigenstructure (т.е. Съвкупността от собствени стойности на матрицата) на Анализа на Основните Компоненти

Следващият проблем, който ще вземем под внимание е този за намирането на единичните вектори \mathbf{q} , а заедно с това $\psi(\mathbf{q})$ има екстремално или стационарна стойност (локален максимум или минимум) и е обект на ограничението на Евклидовата норма на \mathbf{q} . Решението на този проблем се крие в „ейген-структурата“ на корелационната матрица \mathbf{R} . Ако \mathbf{q} е единичен вектор, такъв че вариационната проба $\psi(\mathbf{q})$ има екстремална стойност, след това за всяко малко смущение $\delta \mathbf{q}$ на единичния вектор \mathbf{q} ние намираме, че на първия ред в $\delta \mathbf{q}$

$$\psi(\mathbf{q} + \delta\mathbf{q}) = \psi(\mathbf{q}) \quad (8.8)$$

Сега от дефиницията на вариационата проба, дадена в Ур-е(8.7) имаме

$$\begin{aligned}\psi(\mathbf{q} + \delta\mathbf{q}) &= (\mathbf{q} + \delta\mathbf{q})^T \mathbf{R}(\mathbf{q} + \delta\mathbf{q}) \\ &= \mathbf{q}^T \mathbf{R}\mathbf{q} + 2(\delta\mathbf{q})^T \mathbf{R}\mathbf{q} + (\delta\mathbf{q})^T \mathbf{R} \delta\mathbf{q}\end{aligned}$$

Където на втория ред, използвахме Ур-е(8.6). Пренебрегваме термина от втори ред $(\delta\mathbf{q})^T \mathbf{R} \delta\mathbf{q}$ и се позоваваме на определението на ур-е(8.7), за това можем да напишем

$$\begin{aligned}\psi(\mathbf{q} + \delta\mathbf{q}) &= \mathbf{q}^T \mathbf{R}\mathbf{q} + 2(\delta\mathbf{q})^T \mathbf{R}\mathbf{q} \\ &= \psi(\mathbf{q}) + 2(\delta\mathbf{q})^T \mathbf{R}\mathbf{q}\end{aligned} \quad (8.9)$$

Следователно, използването на ур.(8.8) в (8.9) означава, че

$$(\delta\mathbf{q})^T \mathbf{R}\mathbf{q} = 0 \quad (8.10)$$

Просто всякакви смущения $\delta\mathbf{q}$ от \mathbf{q} не са допустими; по-скоро ние сме ограничени да използваме само тези смущения, за които Евклидват норма на смутения вектор $\mathbf{q} + \delta\mathbf{q}$ остава равна на единицата, което е

$$\|\mathbf{q} + \delta\mathbf{q}\| = 1$$

Или еквивалентното

$$(\mathbf{q} + \delta\mathbf{q})^T (\mathbf{q} + \delta\mathbf{q}) = 1$$

Следователно, в светлината на ур.(8.2), изискваме на първия ред

$$(\delta\mathbf{q})^T \mathbf{q} = 0 \quad (8.11)$$

Това означава, че смущенията $\delta\mathbf{q}$ трябва да са ортогонални на \mathbf{q} и следователно е разрешено промяна в посоката само на \mathbf{q} .

Според установената практика, елементите на единичният вектор \mathbf{q} са безмерни във физическия смисъл. За това, ако искаме да комбинираме Ур-я (8.10) и (8.11) трябва да въведем скаларен вектор X в последното уравнение, със същите размери, както и всипсванията в корелационната матрица R . Следователно можем да запишем:

$$(\delta\mathbf{q})^T \mathbf{R}\mathbf{q} - \lambda(\delta\mathbf{q})^T \mathbf{q} = 0$$

Или еквивалентното

$$(\delta\mathbf{q})^T (\mathbf{R}\mathbf{q} - \lambda\mathbf{q}) = 0 \quad (8.12)$$

За да се задържи в това състоянието ур-е (8.12) е необходимо и достатъчно да има

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q} \quad (8.13)$$

Това е Уравнението, което регулира единичните вектори \mathbf{q} , за които вариационната проба $\psi(\mathbf{q})$ има екстремални стойности.

Уравнение (8.13) се разпознава като проблемът за собствената стойност(eigenvalue), често срещан в линейната алгебра (Странг, 1980). Проблемът има нетривиално решение (т.e. $\mathbf{q} \neq \mathbf{0}$) само за специални стойности на X , които се наричат собствени стойности на корелационната матрица \mathbf{R} . Свързаните с \mathbf{q} стойности се наричат собствени вектори eigenvectors. Корелационната матрица се характеризира с реални, неотрицателни собствени стойности. Свързаните собствени вектори са уникални, ако се приеме, че собствените стойности са различни. Нека собствените стойности на m -към- m на матрицата \mathbf{R} се обозначат с $\lambda_1, \lambda_2, \dots, \lambda_m$, а съответните собствени вектори с $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_m$. Тогава пишем

$$\mathbf{R}\mathbf{q}_j = \lambda_j \mathbf{q}_j, \quad j = 1, 2, \dots, m \quad (8.14)$$

Нека съответните собствените стойности да са подредени в низходящ ред:

$$\lambda_1 > \lambda_2 > \dots > \lambda_j > \dots > \lambda_m \quad (8.15)$$

Така че $\lambda_1 = \lambda_{\max}$. Нека собствените вектори бъдат използвани за конструирането на матрица m -към- m :

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_j, \dots, \mathbf{q}_m] \quad (8.16)$$

Тогава може да комбинираме сета от m набора от уравнения, представени в (8.14) в едно уравнение:

$$\mathbf{R}\mathbf{Q} = \mathbf{Q}\Lambda \quad (8.17)$$

Където Λ е диагонална матрица, дефинирана от собствените стойности на матрицата R :

$$\Lambda = \text{diag}[\lambda_1, \lambda_2, \dots, \lambda_j, \dots, \lambda_m] \quad (8.18)$$

Матрицата Q е ортогонална(единна) матрица, в смисъл, че нейните колонни вектори(т.e. собствени вектори на R) отговарят на условията на ортогоналността.

$$\mathbf{q}_i^T \mathbf{q}_j = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases} \quad (8.19)$$

Уравнението (8.19) изисква различни собствени стойности, еквивалентно пишем

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$$

От където може да заключим, че обратната матрица на Q - матрицата е същата като транспонираната ѝ, както е показано

$$\mathbf{Q}^T = \mathbf{Q}^{-1} \quad (8.20)$$

Това означава, че може да пренапишем Ур-то (8.17) във форма, известна като ортогоналното трансформационно сходство:

$$\mathbf{Q}^T \mathbf{R} \mathbf{Q} = \Lambda \quad (8.21)$$

Или в разгърнат вид

$$\mathbf{q}_j^T \mathbf{R} \mathbf{q}_k = \begin{cases} \lambda_j, & k = j \\ 0, & k \neq j \end{cases} \quad (8.22)$$

Ортогоналната сходна(единна) трансформация на ур-е(8.21) трансформира корелационната матрица \mathbf{R} в диагонална матрица на собствени стойности. Корелационната матрица \mathbf{R} може да бъде изразена по отношение на собствените стойности и собствени вектори като:

$$\mathbf{R} = \sum_{i=1}^m \lambda_i \mathbf{q}_i \mathbf{q}_i^T \quad (8.23)$$

Което е посочено като спектрална теорема. Изходната продукция $\mathbf{q}_i \mathbf{q}_i^T$ е от първа степен за всички i .

Уравнения (8.21) и (8.23) са две равностойни представяния на eigend – състава на корелационната матрица \mathbf{R} .

Анализа на основните компоненти и „ейгенд“ – композицията на матрицата \mathbf{R} са въщност едно и също, само гледаме проблема по различен начин. Тази еквивалентност следват Ур-ята (8.7) и (8.23), където виждаме, че вариационната проба и собствените стойности са равни, както се вижда от

$$\psi(\mathbf{q}_j) = \lambda_j, \quad j = 1, 2, \dots, m \quad (8.24)$$

Сега можем да обобщим две важни констатации, които сме направили от ейгенструктурата на анализа на основните компоненти:

- Собствените единични вектори на матрицата \mathbf{R} свързани с нулевия случаен вектор \mathbf{X} , определящ единичните вектори \mathbf{q} , представляващи основните направления, по които вариационните преби $\psi(\mathbf{q}_j)$ получават своите екстремални стойности.
- Свързаните с тях собствени стойности определят екстремалните стойности на вариационните преби $\psi(\mathbf{u}_j)$.

Базово представяне на данни

Нека дата векторът \mathbf{x} обозначава реализация на случайния вектор \mathbf{X} .

С тъм можни решения за единичния вектор \mathbf{q} , откриваме, че са тъм можни и прогнози за дата вектора \mathbf{x} , които да се вземат под внимание. Конкретно, от Ур-е (8.1) отбелоязваме, че

$$a_j = \mathbf{q}_j^T \mathbf{x} = \mathbf{x}^T \mathbf{q}_j, \quad j = 1, 2, \dots, m \quad (8.25)$$

Където a_j са прогнозите на \mathbf{x} върху основните насоки, представени от единичния вектор \mathbf{q}_j . a_j се наричат основни компоненти; те имат едни и същи физически измерения като дата вектора \mathbf{x} . Формулата в Ур-е (8.25) може да се разглежда като един от анализите.

За реконструкция на оригиналният дата вектор \mathbf{x} точно от проекцията a_j процираме, както следва. Първо комбинираме набора от проекции $\{a_j | j = 1, 2, \dots, m\}$ в един единствен вектор, както се вижда

$$\begin{aligned} \mathbf{a} &= [a_1, a_2, \dots, a_m]^T \\ &= [\mathbf{x}^T \mathbf{q}_1, \mathbf{x}^T \mathbf{q}_2, \dots, \mathbf{x}^T \mathbf{q}_m]^T \\ &= \mathbf{Q}^T \mathbf{x} \end{aligned} \quad (8.26)$$

Следващото, което правим е пре-умножване на двете страни на уравнението (8.26) от матрицата \mathbf{Q} , а след това използваме връзката на Ур-е (8.20). Следователно оригиналният дата вектор \mathbf{x} може да бъде реконструиран по следния начин:

$$\begin{aligned} \mathbf{x} &= \mathbf{Q} \mathbf{a} \\ &= \sum_{j=1}^m a_j \mathbf{q}_j \end{aligned} \quad (8.27)$$

Което може да се разглежда като формулата за *Синтез*. В този смисъл, единичните вектори \mathbf{q} представляват основата на дата пространството. И наистина ур.(8.27) е нищо друго освен една координатна трансформация, според която точката \mathbf{x} в дата пространството се трансформира в съответната точка във функционалното пространство.

Димензионално Намаляване

От гледна точка на статистическото моделно разпознаване, практическата стойност на основните компоненти на анализа е, че той осигурява ефективна техника за намаляване на размерността. По-специално, можем да намалим броя на характеристиките, необходими за ефективно представяне на данни, като изхвърлим тези линейни комбинации в Ур-е (8.27), които имат малки разлики и да се запазят само тези, които имат големи разлики. Нека $\lambda_1, \lambda_2, \dots, \lambda_l$ бележат най-големия l – собствена стойност на

корелационната матрица R . след това можем да сближим дата вектора x , намалявайки разширяването на Ур-то (8.27) след l условията, както следва

$$\begin{aligned}\hat{x} &= \sum_{j=1}^l a_j \mathbf{q}_j \\ &= [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_l] \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix}, \quad l \leq m\end{aligned}\tag{8.28}$$

С даден оригинален дата вектор x , можем да използваме Ур-е (8.25) за изчисление на сета от основни компоненти в Ур-е (8.28), както следва:

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_l^T \end{bmatrix} x, \quad l \leq m\tag{8.29}$$

Линеарната проекция на Ур-е (8.29) от \mathbb{R}^m към \mathbb{R}^l , представлява енкодер за приблизително представяне на дата вектора x , както е показано на Фиг. 8.2а. Съответно, линейната проекция на уравнението (8.28)

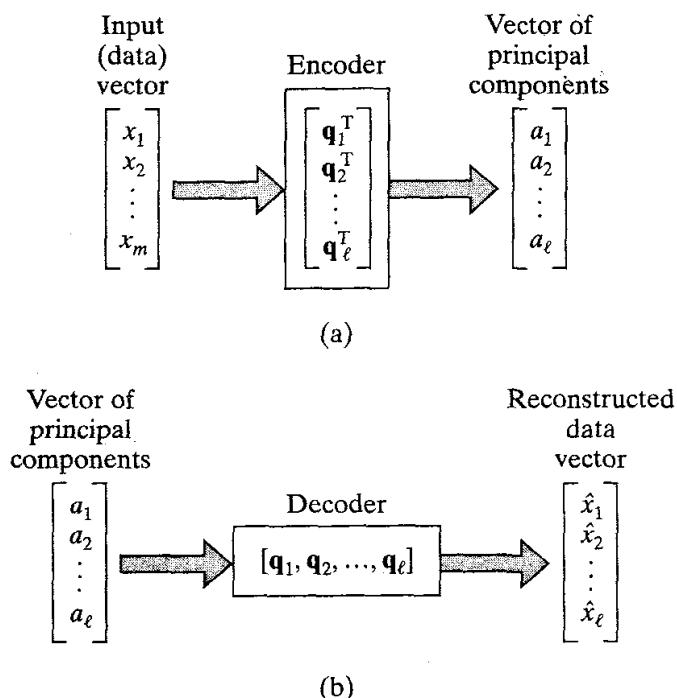


FIGURE 8.2 Illustration of two phases of principal components analysis:
(a) Encoding. (b) Decoding.

От \mathbb{R}^l към \mathbb{R}^m (т.е. картографирането от функционалното пространство обратно към дата пространството) представлява декодер за приблизителна реконструкция на

оригиналния дата вектор \mathbf{x} , показан на фигура (8.2b). Имайте предвид, че доминиращите(т.е. най-големите) собствени стойности $\lambda_1, \lambda_2, \dots, \lambda_l$ не влизат в изчисленията описани в У-ря (8.28) и (8.29), те просто определят броя на основните компоненти, използвани за кодиране и съответно за декодеране.

Приблизителният вектор грешка \mathbf{e} е равен на разликата между оригиналния дата вектор \mathbf{x} и приблизителния дата вектор $\hat{\mathbf{x}}$, както е показано

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} \quad (8.30)$$

Заместването на Ур-я (8.27) и (8.28) в (8.30) дава

$$\mathbf{e} = \sum_{j=l+1}^m a_j \mathbf{q}_j \quad (8.31)$$

Вектор грешката \mathbf{e} е ортогонален на приблизителния дата вектор \mathbf{x} , както е показано на Фиг.8.3 . С други думи, вътршният продукт на векторите \mathbf{x} и \mathbf{e} е нула. Това свойство е показано с помощта на ур-я (8.28) и (8.31), както следва:

$$\begin{aligned} \mathbf{e}^T \hat{\mathbf{x}} &= \sum_{i=l+1}^m a_i \mathbf{q}_i^T \sum_{j=1}^l a_j \mathbf{q}_j \\ &= \sum_{i=l+1}^m \sum_{j=1}^l a_i a_j \mathbf{q}_i^T \mathbf{q}_j \\ &= 0 \end{aligned} \quad (8.32)$$

Където имахме полза от второто условие в Ур-е(8.19). Уравнение (8.32) е познато като *принцип на ортогоналността*.

Общата вариация на m компонентите на дата вектора \mathbf{x} е , чрез Ур-е (8.7) и първия ред на ур-е (8.22),

$$\sum_{j=1}^m \sigma_j^2 = \sum_{j=1}^m \lambda_j \quad (8.33)$$

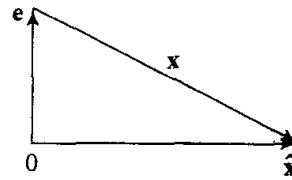
Където σ_j^2 е дисперсията на j -тия основен компонент a_j . Общата вариация на l елементите на приблизителния вектор $\hat{\mathbf{x}}$ е

$$\sum_{j=1}^l \sigma_j^2 = \sum_{j=1}^l \lambda_j \quad (8.34)$$

Общата вариация на $(l - m)$ елементите на приблизителния вектор грешка \mathbf{e} $\mathbf{x} - \hat{\mathbf{x}}$ и за това

$$\sum_{j=l+1}^m \sigma_j^2 = \sum_{j=l+1}^m \lambda_j \quad (8.35)$$

FIGURE 8.3 Illustration of the relationship between vector x , its reconstructed version \hat{x} , and error vector e .



Собствените стойности $\lambda_{l+1}, \dots, \lambda_m$ са най-малките ($m - l$) соб. Стойности на корелационната матрица R ; те отговарят на условията, изхвърлени от разширяването на Ур-е (8.28), използвано за изграждане на приблизителния вектор x . Колкото по-близко са тези собствени стойности до нулата, толкова по-ефективна ще бъде димензионалната редукция (в резултат от прилагането на анализа на основни компоненти върху данни вектора x) в запазването на информационното съдържание в оригиналния му вид. Така, за да се извърши намаляване на размерността на някои входни данни, изчислявме собствените стойности и собствените вектори на корелационната матрица на входния данни вектор; и след това проектираме информацията ортогонална върху подпространството, измерено чрез вектори, принадлежащи към доминиращите собствени стойности. Този метод на представяне на данните обикновено се нарича *подпространствено разлагане* (Оя (Oja) 1983).

Пример 8.1 Дву-измерен набор от данни (*Bivariate Data Set*)

За да илюстрираме прилагането на анализа на основните компоненти, взимаме за пример дву-димензионален набор от данни, изобразен на Фиг. 8.4, където се предполага, че двета модела оси са приблизително на една и съща скала. Хоризонталните и вертикални оси на диаграмата представляват естествените координати на набора от данни. Завъртените оси, означени с 1 и 2, произтичат от прилагането на анализа на основните компоненти на този набор от данни. От фиг. 8.4 виждаме, че проектирането на набора от данни върху ос 1 улавя основната характеристика на данните, а именно факта, че набора от данни е бимодален (т.е. има два кълстера в неговата структура). Всъщност вариацията на проекциите на данните върху оса 1 е по-голяма от всяка друга акцисна проекция на фигурата. От друга страна присъщият бимодален характер на набора от данни е напълно закрит, когато се проектира върху правоъгълната ос 2.

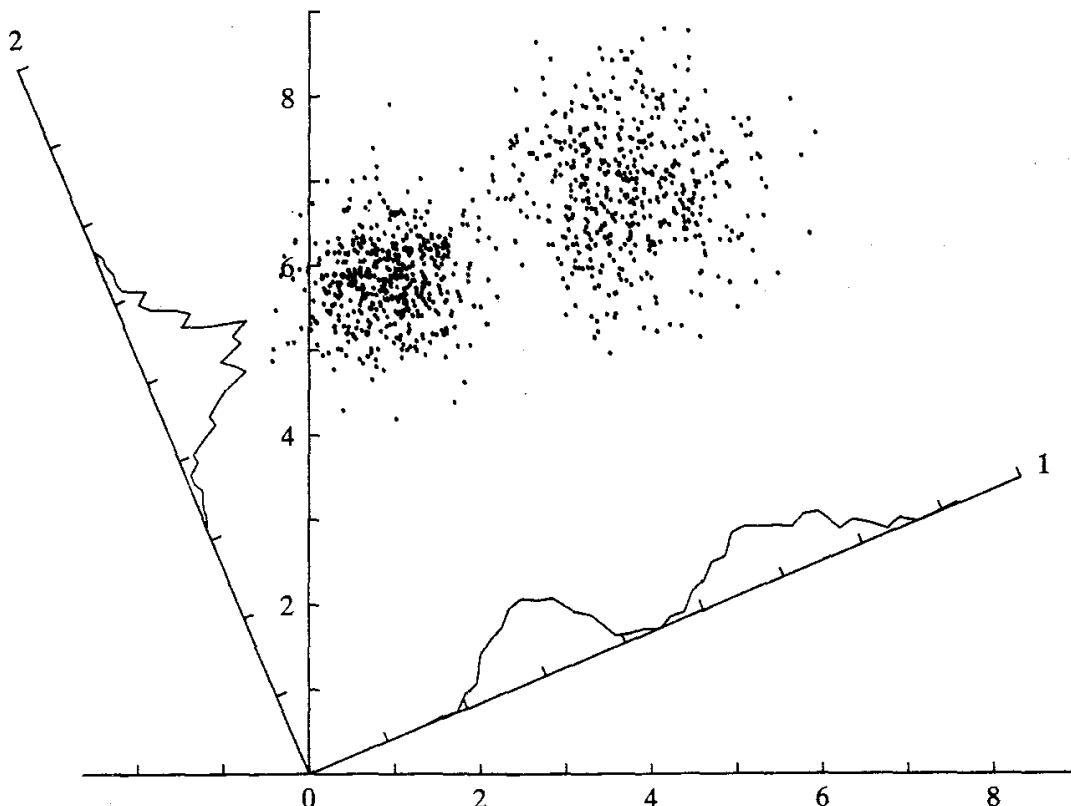


FIGURE 8.4 A cloud of data points is shown in two dimensions, and the density plots formed by projecting this cloud onto each of two axes, 1 and 2, are indicated. The projection onto axis 1 has maximum variance, and clearly shows the bimodal, or clustered character of the data.

Важното, което трябва да се отбележи от този прост пример е, че въпреки че структурата на кълстера на набора от данни е виден от дву-димезионалната площ на даниите в сиров вид, показани в рамките на хоризонталната и вертикална ос, това не винаги е така и на практика. В по-общият случай на по-високо димензионални набори от данни, е напълно възможно да имат вътрешната структура на кълстера от прикрити данни и за да го видите, трябва да извършите статистически анализ, подобрен на анализа на основни компоненти (Линскер, 1988a).

8.4 Хеббиян-базиран максимален собствен филтър

Съществува тясна кореспонденция между поведението на самостоятелно организирана невронна мрежа и статистическия метод на анализиране на основните компоненти. В този раздел ние демонстрираме тази кореспонденция чрез установяването на един забележителен резултат. Един линеен неврон с Хеббиян тип на адаптация на своите синаптични тегла, може да еволюира във филтър за първата основна съставка на входовата дистрибуция(Оя, 1982).

За да продължим с демонстрацията, да вземем в предвид един прост невронен модел, изобразен на Фиг.8.5а. моделът е линеен в смисъл, че моделния изход представлява

линейна комбинация на неговите входове. Неврона получава набор m от входни сигнали x_1, x_2, \dots, x_m чрез кореспондиращ набор от m връзки с тегла w_1, w_2, \dots, w_m респективно. Резултатният моделен изход е това

$$y = \sum_{i=1}^m w_i x_i \quad (8.36)$$

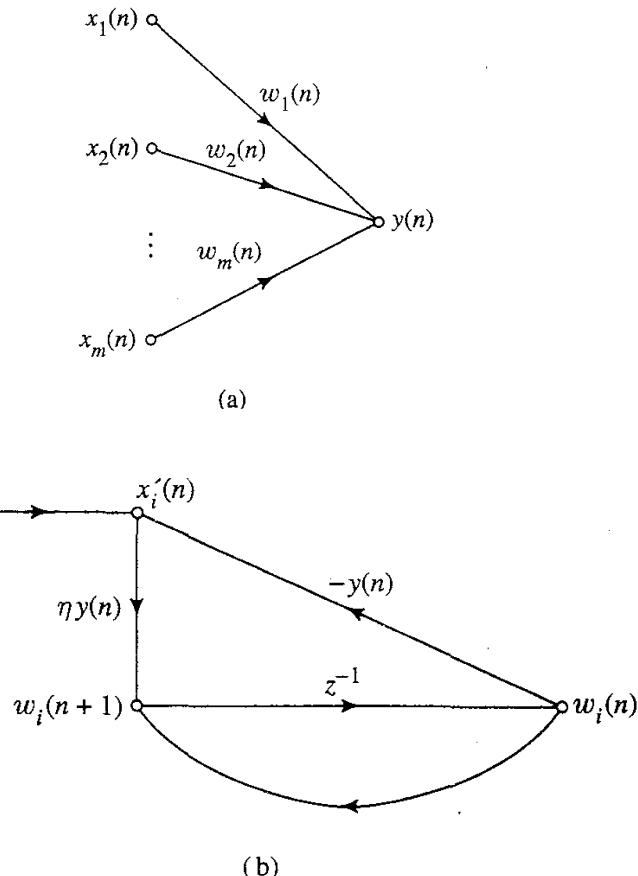


FIGURE 8.5 Signal-flow graph representation of maximum eigenfilter.
(a) Graph of Eq. (8.36).
(b) Graph of Eqs. (8.41) and (8.42).

Забележете, че в ситуацията описана тук имаме работа само с един единствен неврон, следователно не е нужно да използваме „под-скриптове“, за да идентифицираме синаптичните тегла на мрежата.

В съответствие с постулата за обучение на Хебб, една синаптична връзка w_i варира във времето, нараства силно, когато пресинаптичния сигнал x_i и постсинаптичния сигнал y се сблъскват един с друг. Конкретно, пишем

$$w_i(n+1) = w_i(n) + \eta y(n)x_i(n), \quad i = 1, 2, \dots, m \quad (8.37)$$

Където с **н** обозначаваме дискретното време, а с **η** параметъра за обучителния процент (learning-rate parameter). Въпреки това, това правило за обучение в основната си форма води до неограничен растеж на синаптичното тегло w_i , което е неприемливо от физическа гледна точка. Този проблем може да се преодолее, като се включи някаква форма на насищане или нормализиране в обучителното правило за адаптиране на синаптичните тегла. Използването на нормализация има ефекта на въвеждането на

конкуренция между синапсисите на неврона над ограничените ресурси, което от Принцип 2 на самоорганизацията, е от съществено значение за стабилизацията. От математическа гледна точка, една удобна форма на нормализиране е описана от Оя(1982г.).

$$w_i(n + 1) = \frac{w_i(n) + \eta y(n)x_i(n)}{(\sum_{i=1}^m [w_i(n) + \eta y(n)x_i(n)]^2)^{1/2}} \quad (8.38)$$

, където сумирането в знаменателя се разпростира върху пълния набор от синапсисите, свързани с неврона. Ако приемем, че learning-rate параметъра η е малък, може да разширим ур-е (8.38) мощна серия в η и така да напишем

$$w_i(n + 1) = w_i(n) + \eta y(n)[x_i(n) - y(n)w_i(n)] + O(\eta^2) \quad (8.39)$$

Където $O(\eta^2)$ представлява вторият и по-висок ред в η . За малки η основтелно можем да игнорираме това условие и следователно да приближим Ур-е (8.38) до първата цел на η , както следва

$$w_i(n + 1) = w_i(n) + \eta y(n)[x_i(n) - y(n)w_i(n)] \quad (8.40)$$

Терминът $y(n)x_i(n)$ от дясната страна на Ур-е (8.40) представлява обичайната Хеббиянова модификация на синаптичната връзка w_i , за това се причислява към самоусилващия ефект, диктуван от Принцип 1 на самоорганизацията. Включването на негативният термин $-y(n)w_i(n)$ е отговорен за стабилизирането, в съответствие с Принцип 2; модифицира входа $x_i(n)$ във форма, зависеща от асоциираните синаптични тегла $w_i(n)$ и изхода $y(n)$, както е показано

$$x'_i(n) = x_i(n) - y(n)w_i(n) \quad (8.41)$$

Което може да се погледне като *ефективният вход* на i -тия синапсис. Сега може да използваме дефиницията, дадена в Ур-е(8.41), за да пренапишем правилото на Ур-е(8.40) така:

$$w_i(n + 1) = w_i(n) + \eta y(n)x'_i(n) \quad (8.42)$$

Цялостното функциониране на неврона е представено от комбинацията от два сигнално-поточни графа, както е показано на Фиг.8.5. Сигналният поток на графа на Фиг.8.5a показва зависимостта на изхода $y(n)$ от теглата $w_1(n), w_2(n), \dots, w_m(n)$, в съответствие с Ур-е(8.36). Сигналният поток на графа на Фиг.8.5b осигурява изобразяването на Ур-я (8.41) и (8.42); пропускливостта z^{-1} в средната част на графиката представлява оператора *единна-закъснение*. Изходният сигнал $y(n)$ произведен във Фиг.8.5a действа като пропускливост на Фиг.8.5b. Графиката на Фиг.8.5b ясно показва следните две форми на вътрешна връзка, действащи върху неврона:

- Позитивни отзиви за собствена амплификация и следователно растеж на синаптичните тегла $w_i(n)$, според външните си входове $x_i(n)$.

- Отрицателна обратна връзка поради – $y(n)$ за контролиране на растежа, като по този начин се достига до стабилизиране на синаптичното тегло $w_i(n)$.

Продуктовия термин – $y(n)w_i(n)$ е свързан с забравящият или разливащ фактор, който често се използва в изучаването на правила, но с една разлика: Забравящият фактор става по-ясно изразен с по-силен отговор $y(n)$. Този вид контрол изглежда, че има невробиологична подкрепа (Стент, 1973г.).

Матрично представяне на Алгоритъма

Нека за удобство на представянето приемем, че

$$\mathbf{x}(n) = [x_1(n), x_2(n), \dots, x_m(n)]^T \quad (8.43)$$

и

$$\mathbf{w}(n) = [w_1(n), w_2(n), \dots, w_m(n)]^T \quad (8.44)$$

Входният вектор $\mathbf{x}(n)$ и синаптичният тегловен вектор са типични реализации на два случайни вектора $\mathbf{w}(n)$. Използвайки това вектор означение можем да пренапишем Ур-е(8.36) под формата на вътрешен продукт, както следва:

$$y(n) = \mathbf{x}^T(n)\mathbf{w}(n) = \mathbf{w}^T(n)\mathbf{x}(n) \quad (8.45)$$

по подобен начин може да направим същото и с Ур-е (8.40)

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta y(n)[\mathbf{x}(n) - y(n)\mathbf{w}(n)] \quad (8.46)$$

Следователно, замествайки ур-е (8.45) в (8.46) дава

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) - \mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)\mathbf{w}(n)] \quad (8.47)$$

Обучителният алгоритъм от Ур-е(8.47) представлява нелинейно стохастично разликово уравнение, което прави конвергенционният анализ на алгоритъма математически сложен. За да се проправи пътя за този анализ ще се отклоним за кратко, за да представим ключовия инструмент за конвергенционния анализ на стохастичните алгоритми за сближаване.

Асимптотична стабилностна система

Самоорганизационният обучителен алгоритъм от Ур-е(8.47) е специален случай на общия стохастичен приблизителен алгоритъм

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)h(\mathbf{w}(n), \mathbf{x}(n)), \quad n = 0, 1, 2, \dots, \quad (8.48)$$

Последователността $\eta(\cdot)$ е прието да бъде положителна скаларна последователност.

Функцията за актуализиране $y(\cdot, \cdot)$ е детерминистична функция с някои редовни условия, наложени върху нея. Тази функция, заедно със скаларната последователност $\eta(\cdot)$ създават пълната структура на алгоритъма.

Целта на процедурата, описана тук е да асоциираме *детерминистично обикновено диференциално уравнение* (ОДУ) със стохастично нелинейно уравнение (8.48).

Стабилностните свойства на диференциалното уравнение се свързват с конвергенционалните свойства на алгоритъма. Тази процедура е доста общ инструмент и има широка проложимост. Тя е разработена самостоятелно от Лунг(Ljung, 1977) и от Кушнер и Кларк(Kushner and Clark, 1978), които са използвали различни подхода.

За да започнем с това, процедурата предполага, че стохастичното сближаване на алгоритъма, описан в уравнение (8.48) отговаря на следния набор от условия, използвайки наша терминология:

1. $\eta(n)$ е намаляваща редица с положителни реални числа, например имаме

$$\text{a.} \quad \sum_{n=1}^{\infty} \eta(n) = \infty \quad (8.49)$$

$$\text{b.} \quad \sum_{n=1}^{\infty} \eta^p(n) < \infty \quad \text{for } p > 1 \quad (8.50)$$

$$\text{c.} \quad \eta(n) \rightarrow 0 \quad \text{as } n \rightarrow \infty \quad (8.51)$$

2. Последователността на параметричният(синаптичното тегло) вектор $w(\cdot)$ е ограден с вероятност 1.
3. Функцията за актуализация $h(w, x)$ е непрекъснато различна по отношение на w и x , а нейните производни са ограничени във времето.
4. Границата

$$\bar{h}(w) = \lim_{n \rightarrow \infty} E[h(w, X)] \quad (8.52)$$

Съществува за всяко w ; статистическият очакван оператор E е над случайния вектор X с реализация, обозначена с x .

5. Има локално асимптотично стабилно (по смисъла на Ляпунов) решение на обикновеното диференциално уравнение

$$\frac{d}{dt} w(t) = \bar{h}(w(t)) \quad (8.53)$$

Където t обозначава продължително време; стабилност, по смисълна на Ляпунов се обсъжда в Глава 14.

6. Нека q_1 обозначи решението на ур-е (8.53) с едно общо привличане $\mathcal{B}(\mathbf{q})$ (или басейн на привлияне); това понятие е дефинирано в глава 14. Тогава параметричният вектор $\mathbf{w}(n)$ спада към подмножеството \mathcal{A} на общото привличане $\mathcal{B}(\mathbf{q})$ безкрайно често, с вероятност 1.

Шестте условия, описани тук са приемливи. По – специално, условието на 1(а) е необходимо условие, което дава възможност на алгоритъмада премести оценката до желания лимит, независимо от първоначалните условия. Условието 1(б) дава условието колко бързо $\eta(n)$ трябва да клони към нула; това условие е значително по-малко ограничено от обичайното условие

$$\sum_{n=1}^{\infty} \eta^2(n) < \infty$$

Условие 4 е гавното предположение, което прави възможно асоциирането на диференциалното уравнение и алгоритъма от Ур-е (8.48).

Да разгледаме тогава стохастичният алгоритъм за сближаване, описан от рекурсивното уравнение (8.48), предмет на предположенията от 1 до 6. След това можем да посочим асимптотичната стабилностна теорема за този клас на стохастичния алгоритъм за сближаване, както следва(Ljung. 1977; Kushner – Klark, 1978):

$$\lim_{n \rightarrow \infty} \mathbf{w}(n) = \mathbf{q}_1, \quad \text{с безкрайна честота с вероятност 1} \quad (8.54)$$

Трябва да подчертаем обаче, въпреки че процедурата описана тук може да ни предостави информация за асимптотичните свойства на алгоритъм (8.48), обикновено не може да ни каже колко е броят на итерациите n , за да бъде резултата от анализа приложим. Нещо повече, в проследявящите проблеми, където има да се проследява времеви вариращ вектор , използвайки алгоритъм (8.48) е нецелесъобразно да се изисква

$$\eta(n) \rightarrow 0 \quad \text{като} \quad n \rightarrow \infty$$

акто е предвидено в условие 1(с). Можем да преодолеем тази трудност чрез присвояване на някои малки положителни стойности на η , чийто размер обикновено зависи от прилагането на интереси. Това обикновено се прави и в практическото използвае на стохастичните приблизителни алгоритми в невронните мрежи.

Анализ на Стабилността на Максималеният Собствен филтър (Stability Analysis of the Maximum Eigenfilter)

В подхода ODE за стабилността, ние имаме инструмента, който ни трябва, за да изследваме поведението на конвергенцията на рекурсивния алгоритъм на уравнение (8.46), отнасящ се до максималния собствен филтър, какт е описано тук.

За да задоволим условие 1 от асимптотичната стабилностна теорема, нека

$$\eta(n) = \frac{1}{n}$$

Следващото, което забелязваме от Ур-е (8.47) е, че актуализиращата функция $h(w, x)$ се определя от

$$\begin{aligned} h(\mathbf{w}, \mathbf{x}) &= \mathbf{x}(n)y(n) - y^2(n)\mathbf{w}(n) \\ &= \mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) - [\mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)]\mathbf{w}(n) \end{aligned} \quad (8.55)$$

което ясно отговаря на условие 3 от Теоремата. Ур-е (8.55) е резултата от използването на x реализация на случаен вектор X във функцията за актуализиране $h(w, X)$ над X . За условие 4 взимаме очакваното на $h(w, X)$ над X и по този начин пишем

$$\begin{aligned} \bar{h} &= \lim_{n \rightarrow \infty} E[\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{w}(n) - (\mathbf{w}^T(n)\mathbf{X}(n)\mathbf{X}^T(n)\mathbf{w}(n))\mathbf{w}(n)] \\ &= \mathbf{R}\mathbf{w}(\infty) - [\mathbf{w}^T(\infty)\mathbf{R}\mathbf{w}(\infty)]\mathbf{w}(\infty) \end{aligned} \quad (8.56)$$

Където R е корелационната матрица на стохастичният процес, презентиран от случайния вектор $X(n)$ и $w(\infty)$ е граничната стойност на синаптичния тегловен вектор.

В съответствие с условие 5 и в светлината на Ур-я (8.53) и (8.56) ние търсим стабилни точки на нелинейни диференциални уравнения

$$\begin{aligned} \frac{d}{dt} \mathbf{w}(t) &= \bar{h}(\mathbf{w}(t)) \\ &= \mathbf{R}\mathbf{w}(t) - [\mathbf{w}^T(t)\mathbf{R}\mathbf{w}(t)]\mathbf{w}(t) \end{aligned} \quad (8.57)$$

Нека $w(t)$ се разшири по отношение на пълния орто-нормален набор от собствени вектори на корелационната матрица R , както следва:

$$\mathbf{w}(t) = \sum_{k=1}^m \theta_k(t)\mathbf{q}_k \quad (8.58)$$

Където q_k е k -тия нормализиран собствен вектор на матрицата R , а коефициента $\theta_k(t)$ е времевата варираща проекция на вектора $w(t)$ върху q_k . Заместваме ур-е (8.58) в (8.57) и използвайки основните дефиниции

$$\mathbf{R}\mathbf{q}_k = \lambda_k \mathbf{q}_k$$

и

$$\mathbf{q}_k^T \mathbf{R} \mathbf{q}_k = \lambda_k$$

Където λ_k е собствената стойност асоциирана с q_k , накрая получаваме

$$\sum_{k=1}^m \frac{d\theta_k(t)}{dt} q_k = \sum_{k=1}^m \lambda_k \theta_k(t) q_k - \left[\sum_{l=1}^m \lambda_l \theta_l^2(t) \right] \sum_{k=1}^m \theta_k(t) q_k \quad (8.59)$$

Еквивалентно можем да напишем

$$\frac{d\theta_k(t)}{dt} = \lambda_k \theta_k(t) - \theta_k(t) \sum_{l=1}^m \lambda_l \theta_l^2(t), \quad k = 1, 2, \dots, m \quad (8.60)$$

По този начин намалихме конвергенционният анализ на стохастичния приблизителен алгоритъм на (8.48) до стабилностен анализ на система от обикновени диференциални уравнения (8.60), включващи основния режим $\theta_k(t)$.

Тука ще бъдат разгледани два случая, в зависимост от определената стойност на индекс k . Случай I съответства на $1 < k \leq m$, а Случай II съответства на $k = 1; m$ е измерението на двете $x(n)$ и $w(n)$. Двата случая са разгледани последователно.

Случай I. $1 < k \leq m$. За разглеждането на този случай дефинираме

$$\alpha_k(t) = \frac{\theta_k(t)}{\theta_1(t)}, \quad 1 < k \leq m \quad (8.61)$$

След като се приема, че $\theta_1(t) \neq 0$, което е вярно с вероятност 1, при условие, че първоначалните стойности на $w(0)$ се избират на случаен принцип. След това като разделяме двете страни на ур-е (8.61) по отношение на времето t , получаваме:

$$\begin{aligned} \frac{d\alpha_k(t)}{dt} &= \frac{1}{\theta_1(t)} \frac{d\theta_k(t)}{dt} - \frac{\theta_k(t)}{\theta_1^2(t)} \frac{d\theta_1(t)}{dt} \\ &= \frac{1}{\theta_1(t)} \frac{d\theta_k(t)}{dt} - \frac{\alpha_k(t)}{\theta_1(t)} \frac{d\theta_1(t)}{dt}, \quad 1 < k \leq m \end{aligned} \quad (8.62)$$

След това, използвайки ур-е (8.60) в (8.62), прилагайки дефиницията от ур-е (8.61) и опростявайки резултата, получаваме

$$\frac{d\alpha_k(t)}{dt} = -(\lambda_1 - \lambda_k)\alpha_k(t), \quad 1 < k \leq m \quad (8.63)$$

Предполагайк, че собствените стойности на корелационната матрица R са различни и подредени в низходящ ред имаме

$$\lambda_1 > \lambda_2 > \dots > \lambda_k > \dots > \lambda_m > 0 \quad (8.64)$$

От това следва, че разликата в собствените стойности $\lambda_1 - \lambda_k$, представляваща реципрочната времева константа в ур-е (8.63) е положителна, така откриваме за Случай 1 :

$$\alpha_k(t) \rightarrow 0 \quad \text{as } t \rightarrow \infty \quad \text{for } 1 < k \leq m \quad (8.65)$$

Случай II. $k = 1$. От ур-е (8.60), този втори случай е описан от диференциалното уравнение

$$\begin{aligned}\frac{d\theta_1(t)}{dt} &= \lambda_1\theta_1(t) - \theta_1(t) \sum_{l=1}^m \lambda_l \theta_l^2(t) \\ &= \lambda_1\theta_1(t) - \lambda_1\theta_1^3(t) - \theta_1(t) \sum_{l=2}^m \lambda_l \theta_l^2(t) \\ &= \lambda_1\theta_1(t) - \lambda_1\theta_1^3(t) - \theta_1^3(t) \sum_{l=2}^m \lambda_l \alpha_l^2(t)\end{aligned}\quad (8.66)$$

От Случай I знаем, че $\alpha_l \rightarrow 0$ за $l \neq 1$ като $t \rightarrow \infty$. Оттук последният термин на дясната страна на ур-е (8.66) се приближава към 0, докато времето t клони към бескрайност. Пренебрегвайки този термин, ур-е (8.66) се опростява до

$$\frac{d\theta_1(t)}{dt} = \lambda_1\theta_1(t)[1 - \theta_1^2(t)] \quad \text{for } t \rightarrow \infty \quad (8.67)$$

Трябва да се отбележи обаче, че ур-е (8.67) стои само в асимптотичен смисъл.

Ур-е (8.67) представлява самостоятелна система (т.е. системата не зависи от време). Стабилността на такава система най-добре борави с помощта на положителна определена функция, наречена функция на Ляпинов, подробното описание на която се намира в Глава 14. Нека s обозначи вектора на състоянието на автономна система, а $V(t)$ Ляпиновата функция на системата. Едно равновесно състояние \bar{s} на системата е асимптотично стабилно ако:

$$\frac{d}{dt} V(t) < 0 \quad \text{for } s \in \mathcal{U} - \bar{s}$$

където \mathcal{U} е малка област около \bar{s} .

За проблема на ръка, ще твърдим, че диференциалното ур-е (8.67) има Ляпинова функция, дефинирана от

$$V(t) = [\theta_1^2(t) - 1]^2 \quad (8.68)$$

За да потвърдим това твърдение, ще трябва да покажем, че $V(t)$ отговаря на две условия

$$1. \frac{dV(t)}{dt} < 0 \quad \text{for all } t \quad (8.69)$$

$$2. V(t) \text{ has a minimum} \quad (8.70)$$

(т.е. 1- за всяко t , а 2 - $V(t)$ има минимум.)

Диференцирайки ур-е (8.68) по отношение на времето, получаваме

$$\begin{aligned}\frac{dV(t)}{dt} &= 4\theta_1(t)[\theta_1(t) - 1] \frac{d\theta_1}{dt} \\ &= -4\lambda_1\theta_1^2(t)[\theta_1^2(t) - 1]^2 \quad \text{for } t \rightarrow \infty\end{aligned}\tag{8.71}$$

Където на втория ред употребихме ур-е (8.67). След като собствената стойност λ_1 е положителна, намираме от ур-е (8.71), че условието от ур-е (8.69) е вярно за t клонящо към безкрайност. Още от ур-е (8.71) виждаме, че $V(t)$ има минимум (т.е. $dV(t)/dt$ е нула) в $\theta_1(t) = \pm 1$ и така условието на ур-е (8.70) е също удовлетворено. За това може да приключим анализа на Случай II като заявим, че

$$\theta_1(t) \rightarrow \pm 1 \quad \text{as } t \rightarrow \infty\tag{8.72}$$

Чрез резултата описан в ур-е (8.72) и определението на ур-е (8.71), можем да преизчислим резултата на Случай I даден в ур-е (8.65) в окончателен вид:

$$\theta_k(t) \rightarrow 0 \quad \text{as } t \rightarrow \infty \quad \text{for } 1 < k \leq m\tag{8.73}$$

Общото заключение, извлечено от анализа на случаите I и II е двоен:

- Единственияя принципен механизъм на стохастичният алгоритъм за сближаване, описан в ур-е (8.47), който ще се конвергира в $\theta_1(t)$; всички останали режими на алгоритъма ще клонят към нула.
- Режимът $\theta_1(t)$ ще клони към ± 1 .

Следователно условие 5 на асимптотичната теорема за стабилност е удовлетворено. Конкретно, в светлината на разширяването, описано в ур-е (8.58) можем официално да посочим, че

$$\mathbf{w}(t) \rightarrow \mathbf{q}_1 \quad \text{as } t \rightarrow \infty$$

Където \mathbf{q}_1 е нормализираният собствен вектор асоцииран с най-голямата собствена стойност λ_1 на корелационната матрица \mathbf{R} .

Следващото, което трябва да покажем, в съответствие с условие 6 от асимптотичната теорема за стабилност е, че съществува подмножество \mathcal{A} на набора от всички вектори, както

$$\lim_{n \rightarrow \infty} \mathbf{w}(n) = \mathbf{q}_1 \quad \text{е безкрайно често с вероятност 1}$$

За да направим това, трябва първо да удовлетворим условие 2, което правим чрез ограничение на входовете на $w(n)$, така че техните величини остават под някакъв праг a . След това можем да дефинираме нормата на $w(n)$, като пишем

$$\|\mathbf{w}(n)\| = \max_j |w_j(n)| \leq a\tag{8.74}$$

Нека A е компактното подмножество на R^m , определено от набора от нормативни вектори по-малки или равни на a . Лесно е да се покаже (Сангър, 1989b), че

Ако $\|w(n)\| \leq a$ и константата a е достатъчно голяма, то $\|w(n+1)\| < \|w(n)\|$ с вероятност 1.

По този начин, тъй като броят на итерациите n се увеличава, $w(n)$ евентуално ще бъде в рамките на A и ще остане вътре в A (безкрайно често) с вероятност 1. След като басейна на атракция $B(q_1)$ включва всички вектори с ограничена норма, имаме $A \in B(q_1)$. С други думи условието 6 е удовлетворено.

Сега сме удовлетворили висчките шест условия на асимптотичната стабилностна теорема и по този начин сме показали, че (обект на посочените по-горе предположения) стохастичният приблизителен алгоритъм на (8.47) ще предизвика конвергенция на $w(n)$ с вероятност 1 със собствения вектор q_1 , свързан с най-голямата собствена стойност λ_1 на корелационната матрица R . Това не е единствената фиксирана точка на алгоритъма, но е единствената асимптотична стабилна такава.

Обобщени свойства на Хеббиянов базиран Максимален собствен филтър

Конвергенционният анализ показва, че един линеен неврон обучаван от самоорганизационното обучаващо правило на ур-е (8.39), или еквивалентното на ур-е (8.46) адаптивно извлича първата основна съставка на стационарен вход. Този първи принципален компонент съответства на най-голямата собствена стойност λ_1 на корелационната матрица на случайния вектор $X(n)$; всъщност λ_1 е свързан с дисперсията на моделния изход $y(n)$, както е показано тук.

Нека $\sigma^2(n)$ да бележи вариацията на случайна стойност $Y(n)$ с реализация, отбелязана с $y(n)$, което е

$$\sigma^2(n) = E[Y^2(n)] \quad (8.75)$$

Където $Y(n)$ има нулево значение за вход с нулево значение. Оставяйки $n \rightarrow \infty$ в уравнение (8.46) и използвайки факта, че по кореспондиращ начин $w(n)$ се приближава до q_1 , получаваме

$$x(n) = y(n)q_1 \quad \text{for } n \rightarrow \infty$$

Използвайки връзката можем да покажем, че вариацията $\sigma^2(n)$ доближава λ_1 , докато броят на итерациите n доближава безкрайност; погледнете Проблем 8.2.

В обобщение, Хеббияново базиран линеарен неврон, чиято операция е описана в Ур-е (8.46) се доближава с вероятност 1 до фиксирана точка, която е характеризирана както следва (Оя, 1982):

1. Вариацията на изхода на модела се доближава до най-голямата собствена стойност на корелационната матрица R , както е показано

$$\lim_{n \rightarrow \infty} \sigma^2(n) = \lambda_1 \quad (8.76)$$

2. Синаптичният тегловен вектор на модела се доближава до асоциирания собствен вектор, както е показано

$$\lim_{n \rightarrow \infty} \mathbf{w}(n) = \mathbf{q}_1 \quad (8.77)$$

с

$$\lim_{n \rightarrow \infty} \|\mathbf{w}(n)\| = 1 \quad (8.78)$$

Тези резултати предполагат, че корелационната матрица R е категорично положителна с най-голямата собствена стойност λ_1 , с мултиплициране 1. Те остават също и за неотрицателна корелационна матрица R с условие $\lambda_1 > 0$, с мултипликация 1.

Пример 8.2 Съвпадащи Филтъра

Да разгледаме произволен вектор $\mathbf{X}(n)$, както следва

$$\mathbf{X}(n) = \mathbf{s} + \mathbf{V}(n)$$

където s е фиксирана векторна единица, представляваща *сигналния компонент*, а $V(n)$ „с бял шум” нулев компонент. Корелационната матрица на входният вектор е

$$\begin{aligned} \mathbf{R} &= E[\mathbf{X}(n)\mathbf{X}^T(n)] \\ &= \mathbf{s}\mathbf{s}^T + \sigma^2 \mathbf{I} \end{aligned}$$

където σ^2 е вариацията на елементите на шумовия вектор $V(n)$, а I е идентификационната матрица. Ето защо най-голямата стойност на корелационната матрица R е

$$\lambda_1 = 1 + \sigma^2$$

Асоциативният собствен вектор q_1 е

$$\mathbf{q}_1 = \mathbf{s}$$

Ясно е показано, че това решение удовлетворява проблема със собствената стойност

$$\mathbf{R}\mathbf{q}_1 = \lambda_1 \mathbf{q}_1$$

Следователно, за ситуацията, описана в този пример, самостоятелно организираният линеен неврон (при конвергенция до стабилното му състояние) действа като съвпадащ филтър, в смисъл, че неговиш импулсен отговор (представляван от синаптичните тегла) съответства на сигналния компонент s на входния вектор $\mathbf{X}(n)$.

8.5 Хеббияново-базиран Основен Компонентен Анализ

Хеббияново-базирианият максимален собствен филтър от предишната секция извежда първият основен компонент на входа. Този единичен линеарен невронен модел може да бъде разширен до една feedforward мрежа, с единствен слой линейни неврони за целите на анализа на основните компоненти, с произволни размери на входа (Sanger, 1989b).

За да бъдем по-конкретни, да вземем в предвид мрежата, показана на Фиг. 8.6.

Направени са следните две предположения от структурен характер:

1. Всеки неврон от изходният слой на мрежата е линеен.
2. Мрежата има m входа и l изхода, като и двете са уточнени. Освен това мрежата има по-малко изхода, отколкото входа (т.e. $l < m$).

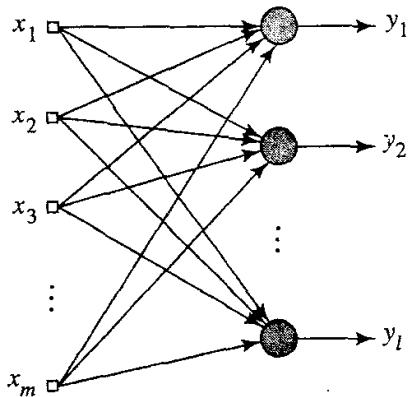


FIGURE 8.6 Feedforward network with a single layer of computation nodes.

Единственият аспект на мрежата, който е предмет на обучение е наборът от синаптични тегла $\{w_{ji}\}$, които свързват източниките възли i във входния слой с изчислителните възли j в изходящия слой, където $i = 1, 2, \dots, m$, а $j = 1, 2, \dots, l$.

Изхода $y_j(n)$ на неврон j по време n , произведен в отговор на набора от входа $\{x_i(n)|i = 1, 2, \dots, m\}$, е даден от (Виж Фиг. 8.7a)

$$y_j(n) = \sum_{i=1}^m w_{ji}(n)x_i(n), \quad j = 1, 2, \dots, l \quad (8.79)$$

Синаптичното тегло $w_{ji}(n)$ е адаптирано в съответствие с генерализираната форма на Хеббияновото обучение, както е показано(Сангър, 1989b):

$$\Delta w_{ji}(n) = \eta \left[y_j(n)x_i(n) - y_j(n) \sum_{k=1}^j w_{ki}(n)y_k(n) \right], \quad \begin{array}{l} i = 1, 2, \dots, m \\ j = 1, 2, \dots, l \end{array} \quad (8.80)$$

където $\Delta w_{ji}(n)$ е промяната, приложена към синаптичното тегло $w_{ji}(n)$ по време n, а η е обучителния-процентов параметър. Генерализираният Хеббиянов Алгоритъм (ГХА - GHA) на ур-е (8.80) за слой от 1 неврони включва алгоритъм на ур-е (8.39) за един неврон като специален случай, който е $j = 1$.

За да развиеме вникването в поведението на обобщения Хебинов алгоритъм ще пренапишеме ур-е (8.80) под формата

$$\Delta w_{ji}(n) = \eta y_j(n)[x'_i(n) - w_{ji}(n)y_j(n)], \quad \begin{array}{l} i = 1, 2, \dots, m \\ j = 1, 2, \dots, l \end{array} \quad (8.81)$$

където $x'_i(n)$ е модифицирана версия на i -тия елемент на входния вектор $\mathbf{x}(n)$; това е функция на j индекса, както е показано

$$\Delta w_{ji}(n) = \eta y_j(n)x''_i(n) \quad (8.83)$$

където

$$x''_i(n) = x'_i - w_{ji}(n)y_j(n) \quad (8.84)$$

По този начин ще отбележим, че

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \quad (8.85)$$

и

$$w_{ji}(n) = z^{-1}[w_{ji}(n+1)] \quad (8.86)$$

Където z^{-1} е оператора за единица закъснение, може да изградим графиката на сигналния поток на Фиг.8.7b за генерализираният Хеббиянов алгоритъм. От тази графика виждаме, че алгоритъмът се поддава на местната форма на изпълнение, при условие, че тя е формулирана като в ур-е (8.85).

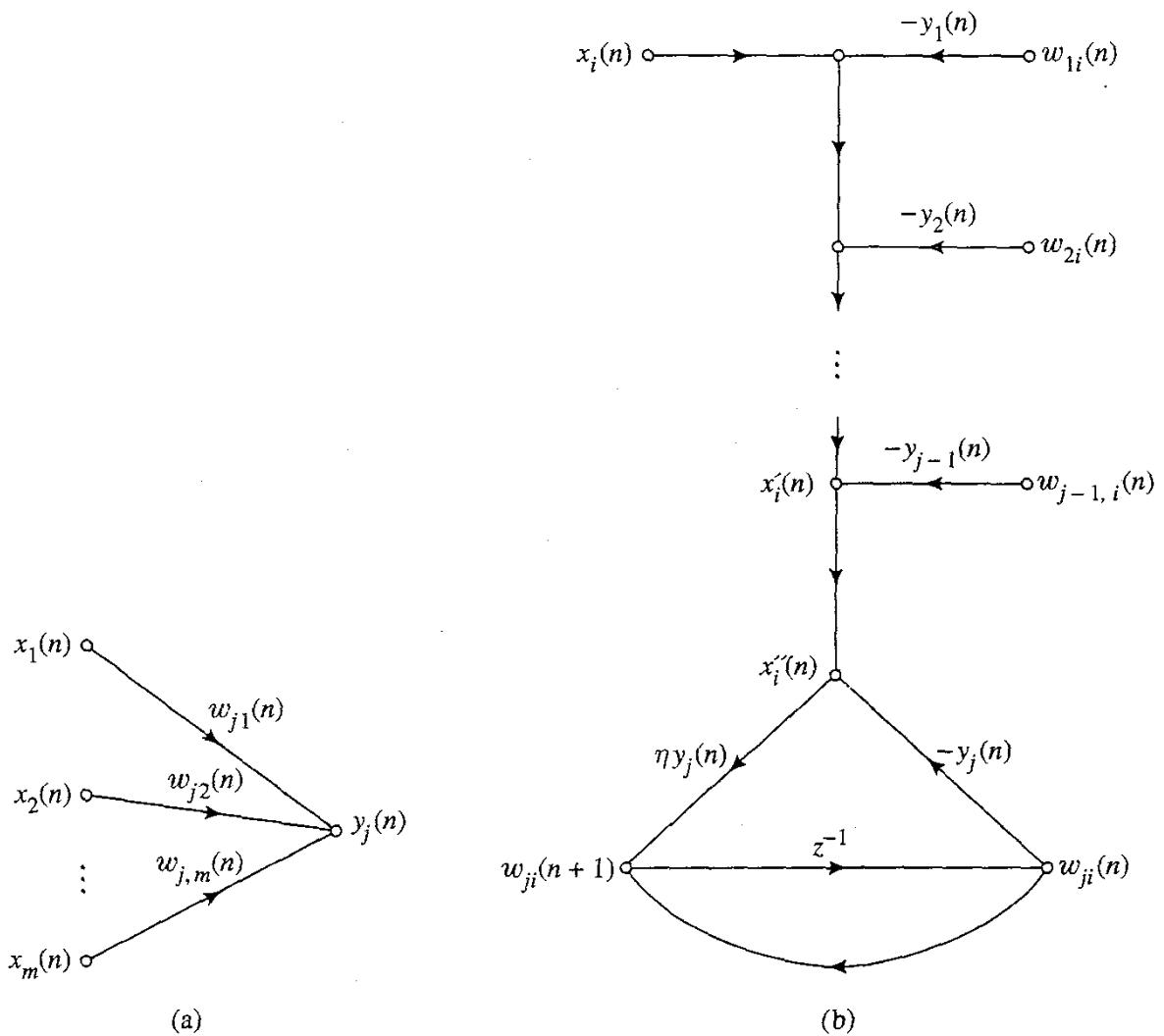


FIGURE 8.7 The signal-flow graph representation of generalized Hebbian algorithm. (a) Graph of Eq. (8.79). (b) Graph of Eqs. (8.80) through (8.81).

Забележете също, че $y_j(n)$, отговорен за обратната връзка в графиката за сигналния поток на фиг.8.7б се определя по ур-е (8.79); представянето на графиката за сигналния поток на това последно ур-е е показано на фиг.8.7а.

За евристичното разбиране на това, как Генерализираният хебианов Алгоритъм действително функционира за първи път ще използваме нотациона матрица, за да пренапишем версията на алгоритъма, определен в ур-е (8.81), както следва:

$$\Delta \mathbf{w}_j(n) = \eta y_j(n) \mathbf{x}'(n) - \eta y_j^2(n) \mathbf{w}_j(n), \quad j = 1, 2, \dots, l \quad (8.87)$$

където

$$\mathbf{x}'(n) = \mathbf{x}(n) - \sum_{k=1}^{j-1} \mathbf{w}_k(n) y_k(n) \quad (8.88)$$

Векторът $\mathbf{x}'(n)$ представлява модифицирана форма на входния вектор. Базират се на представянето, дадено в ур-е (8.87) правим следното наблюдение(Сангър, 1989b):

1. За първият неврон от напредващата мрежа, показана на Фиг. 9.6 имаме:

$$j = 1: \quad \mathbf{x}'(n) = \mathbf{x}(n)$$

В този случай, обобщения алгоритъм на Хебиян намалява до този на ур-е (8.46) за един неврон. От материала, разгледан в раздел 8.5 вече знаем, че този неврон ще открие първата основна съставка на входния вектор $\mathbf{x}(n)$.

2. За вторият неврон на мрежата на Фиг.8.6 пишем

$$j = 2: \quad \mathbf{x}'(n) = \mathbf{x}(n) - \mathbf{w}_1(n)y_1(n)$$

При условие, че първият неврон достига до първия основен компонент , вторият неврон вижда входен вектор $\mathbf{x}'(n)$, от който е отстранен първия собствен вектор на корелационната матрица R. Вторият неврон следователно извлича първият основен компонент на $\mathbf{x}'(n)$, който е еквивалент на втория основен компонент на оригиналния входен вектор $\mathbf{x}(n)$.

3. За третият неврон пишем

$$j = 3: \quad \mathbf{x}'(n) = \mathbf{x}(n) - \mathbf{w}_1(n)y_1(n) - \mathbf{w}_2(n)y_2(n)$$

Да предположим, че първите два компонента са се доближили до първия и втория основен компонент, както е описано в стъпки 1 и 2. Третият неврон сега вижда входен вектор $\mathbf{x}'(n)$, от който първите два собствени вектори са били отстранени. За това извлича първият основен компонент на вектор $\mathbf{x}'(n)$, който е еквивалент на третия основен компонент на оригиналния входен вектор $\mathbf{x}(n)$.

4. Подхождайки по този начин към останалите неврони на feedforward мрежата на Фиг. 8.6 е вече видно, че всеки изход на мрежата е трениран според обобщения Хебиянов алгоритъм от Ур-е(8.81), представляващ отговор на определен собствен вектор на корелационната матрица на входния вектор, а отделните изходи са подредени по намаляване на собствената стойност.

Този метод на изчисляване на собствени вектори е известен като техниката Хотелска дефляция (Kreyszig, 1988); следва процедура, подобна на Gram – Schmidt ортогонализация (Strang, 1980).

Описанието неврон-сед-неврон, описано тук е предназначено единствено за опростяване на обяснението. На практика, всички неврони в обобщения Хебиянов алгоритъм са склонни към събиране.

Конвергентни Съображения

Нека $\mathbf{W}(n) = \{w_{ji}(n)\}$ обозначи l -към- m тегловна синаптична матрица на feedforward мрежата, показана на Фиг.8.6; това е:

$$\mathbf{W}(n) = [\mathbf{w}_1(n), \mathbf{w}_2(n), \dots, \mathbf{w}_l(n)]^T \quad (8.89)$$

Нека проценто-бучаващият параметър на Обобщения Хебиянов Алгоритъм от Ур-e(8.81) приеме времева-варираща форма $\eta(n)$, като тази в лимита и имаме

$$\lim_{n \rightarrow \infty} \eta(n) = 0 \quad \text{and} \quad \sum_{n=0}^{\infty} \eta(n) = \infty \quad (8.90)$$

Тогава можем да пренапишем този алгоритъм в матричната форма

$$\Delta \mathbf{W}(n) = \eta(n) \{ \mathbf{y}(n) \mathbf{x}^T(n) - \text{LT}[\mathbf{y}(n) \mathbf{y}^T(n)] \mathbf{W}(n) \} \quad (8.91)$$

Където оператора $\text{LT}[\cdot]$ определя всички елементи над диагонала на матричния елемент до нула, като по този начин превръща матрицата в по-нисък триъгълник. При този условия и позовавайки се на предположенията, направени в раздел 8.4, конвергенцията на алгоритъма GHA се доказва от една процедура, подобна на тази, представена в предишния раздел за максималния собствен филтър. По този начин може да заявим следната теорема(Сангер(Sanger), 1989b):

Ако синаптичната тегловна матрица $\mathbf{W}(n)$ присвоява произволни стойности във времева стъпка $n=0$, тогава с вероятност 1, Обобщения Хебиянов Алгоритъм от ур-e(8.91) ще достигне до фиксирана точка с $\mathbf{W}^T(n)$, достигайки до матрица, чито колони са първите l собствени вектори на m -към- m корелационна матрица \mathbf{R} , на m -към- l входящ вектори, подредени по намаляване на собствената стойност.

Практическото значение на тази теорема е гаранцията, че Генерализирания Хебиянов алгоритъм ще намери първите l собствени вектори на корелационната матрица \mathbf{R} , приемайки, че свързаните с тях собствени вектори са отделени. Също толкова важно е и че не трябва да изчисляваме корелационната матрица \mathbf{R} . Вместо това, първите собствени вектори на \mathbf{R} се изчисляват чрез алгоритъм, директно от входните данни. В резултат изчислителните спестявания могат да бъдат огромни, особено ако размера m на входното пространство е много голям. Както и необходимият брой собствени вектори, свързани с най-големите l собствени стойности на корелационната матрица \mathbf{R} са една малка част от m .

Конвергентната теорема е формулирана в условията на променящия се във времето обучително-процентов параметър $\eta(n)$. На практика, обучително-процентов параметър е избран да бъде една малка константа η , в чийто случай конвергенцията е гарантирана, със средна квадратна грешка в синаптичните тегла на подредбата η .

В Chattejee и др.(1998), конвергентните свойства на алгоритъма GHA, описани в ур-e(8.91) са изследвани. В анализа, представен там се вижда, че увеличението η води

до по-бърза конвергенция и по-голяма средно-асимптотична грешка, което интуитивно е задоволително. В този документ, компромиса между точността на изчисленията и скоростта на обучение е ясна, наред с други неща.

Оптималност на Генерализираният Хебиянов Алгоритъм

Да предположим, че в предела на допустимото пишем

$$\Delta \mathbf{w}_j(n) \rightarrow \mathbf{0} \quad \text{and} \quad \mathbf{w}_j(n) \rightarrow \mathbf{q}_j \quad \text{as } n \rightarrow \infty \quad \text{for } j = 1, 2, \dots, l \quad (8.92)$$

И че имаме

$$\|\mathbf{w}_j(n)\| = 1 \quad \text{for all } j \quad (8.93)$$

Тогава граничните стойности $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_l$ на синаптично тегловният вектор на неворна в feedforward мрежата на Фиг.8.5 представляват нормализирания собствен вектор, асоцииран с l – доминантната собствена стойност на корелационната матрица \mathbf{R} , подредени по низходяща стойност. При постигнато равновесие може да запишем:

$$\mathbf{q}_j^T \mathbf{R} \mathbf{q}_k = \begin{cases} \lambda_j, & k = j \\ 0, & k \neq j \end{cases} \quad (8.94)$$

Където $\lambda_1 > \lambda_2 > \dots > \lambda_l$.

За изхода на неврон j имаме гранична стойност

$$\lim_{n \rightarrow \infty} y_j(n) = \mathbf{x}^T(n) \mathbf{q}_j = \mathbf{q}_j^T \mathbf{x}(n) \quad (8.95)$$

Нека $Y_j(n)$ бележи случайна променлива с реализация, отбелязана от изхода $y_j(n)$. Напречната корелация между случайните величини $Y_j(n)$ и $Y_k(n)$ при равновесие е дадено от:

$$\begin{aligned} \lim_{n \rightarrow \infty} E[Y_j(n) Y_k(n)] &= E[\mathbf{q}_j^T \mathbf{X}(n) \mathbf{X}^T(n) \mathbf{q}_k] \\ &= \mathbf{q}_j^T \mathbf{R} \mathbf{q}_k \\ &= \begin{cases} \lambda_j, & k = j \\ 0, & k \neq j \end{cases} \end{aligned} \quad (8.96)$$

Следователно, можем да посочим, че при равновесие генерализираният Хебиянов алгоритъм на ур-е(8.91) действа като собствен анализатор на входните данни.

Нека $\hat{\mathbf{x}}(n)$ бележи определената стойност на входния вектор $\mathbf{x}(n)$, за която ограничителните условия на ур-е(8.92) са изпълнени за $j = l - 1$. Следователно, от матричната форма на ур-е (8.80) откриваме, че в рамките на ограничението

$$\hat{\mathbf{x}}(n) = \sum_{k=1}^l y_k(n) \mathbf{q}_k \quad (8.97)$$

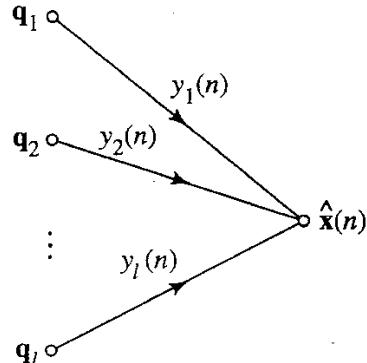
Това означава, че при дадени два набора от количествата, ограничителните стойности $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_l$ на синаптичните тегловни вектори на невроните в мрежата от Фиг.8.5 и съответните изходи $y_1(n), y_2(n), \dots, y_l(n)$, можем да построим най-малка квадратна линейна оценка $\hat{\mathbf{x}}(n)$ на входния вектор $\mathbf{x}(n)$. В действителност, формулата на ур-е (8.97) може да се разглежда като дата-реконструкция, както е показано на Фиг.8.8. Имайте в предвид, че с оглед на дискусията, представена в раздел 8.3, този метод на дата-реконструкция е предмет на приблизителна вектор-грешка, която е ортогонална на оценката $\hat{\mathbf{x}}(n)$.

Обобщение на GNA

Изчисленията, които участват в алгоритъма на Хебиан са прости. Могат да бъдат обобщени, както следва:

1. Инициализирайте синаптичните тегла на мрежата, w_{ji} , с малки случаенни стойности по време $n = 1$. Присвоете малка положителна стойност на обучително-процентовия параметър η .

FIGURE 8.8 Signal-flow graph representation of how the reconstructed vector $\hat{\mathbf{x}}$ is computed.



2. За $n = 1, j = 1, 2, \dots, l$ и $i = 1, 2, \dots, m$, изчислете

$$y_j(n) = \sum_{i=1}^m w_{ji}(n) x_i(n)$$

$$\Delta w_{ji}(n) = \eta \left[y_j(n) x_i(n) - y_j(n) \sum_{k=1}^j w_{ki}(n) y_k(n) \right]$$

където $x_i(n)$ е i -тия компонент на m -към-1 входящия вектор $\mathbf{x}(n)$ и l е търсеното число на основния компонент.

3. Увеличете n към 1, преминете към стъпка 2 и продължете, докато синаптичните тегла w_{ji} , достигнат до равновесно състояние. За голямо n , синаптичното тегло

w_{ji} на неврона j клони към i -тия компонент на собствения вектор с j -тата стойност на собствения вектор на корелационната матрица на входния вектор $x(n)$.

8.6 Компютърен експеримент: Image Coding

Завършваме обсъждането на Генерализираният Хебиянов обучителен Алгоритъм, проверявайки неговата употреба за решаване на образен кодов проблем.

Фиг. 8.9а показва изображение на родители, използвани за обучение; този образ подчертава крайната информация. Бе дигитализирано да формира 256x256 изображение с 256 нива на сивото. Изображението бе кодирано, използвайки линейна feedforward мрежа с единствен слой, съдържащ 8 неврона, всеки с 64 входа. За да обучим мрежата, 8x8 неприпокриващи се блокове на изображението са използвани. Експериментът е извършен с 2000 сканирания на картина и малък процен на обучение $\eta = 10^{-1}$.

Фиг.8.9б показва 8x8 маските представляващи синаптичните тегла, изучени от мрежата. Всяка от осемте маски показва набора от синаптични тегла, свързани с даден неврон на мрежата. Конкретно, възбудените синапси(с положителни тегла) са показани бели, докато инхибиторните синапси(с отрицателни тегла) са показани черни, сивото показва нулева тежест. В нашата нотация, маските представляват колоните на 64x8 синаптична тегловна матрица W^T след като Генерализирания Хебиянов алгоритъм е конвергирал. За да се кодира изображението е била използвана следната процедура:

- Всеки 8x8 блок на изображението е умножено по всяка от 8те маски показани на Фиг.8.9б, което е генерирало 8 коефициента за кодиране на изображението; Фиг.8.9с показва реконструирианият образ, базиран на доминиращите 8 основни компонента, без квантуване.
- Всеки коефициент е разномерно квантуван с броя на битовете, приблизително пропорционален на логаритъма на вариацията на този коефициент, върху изображението. Така на първите 3 маски бяха определени 6 бита, на следващите две по 4 бита на всяка, на другите две маски по 3 бита и на последната маска – 2 бита. Въз основа на това представяне, всичко 34 бита бяха нужни, за да се кодира всеки 8x8 блок от пиксели, даващи резултат в данни на скорост 0.53 бита за пиксел.

За да се реконструира образа от квантувания коефициент, всички маски са претеглени с техните квантувани коефициенти, а след това добавени, за да се реконструира всеки блок от изображението. Реконструираното изображение на родителите с 15 към 1 компресия е показано на Фиг. 8.9д.

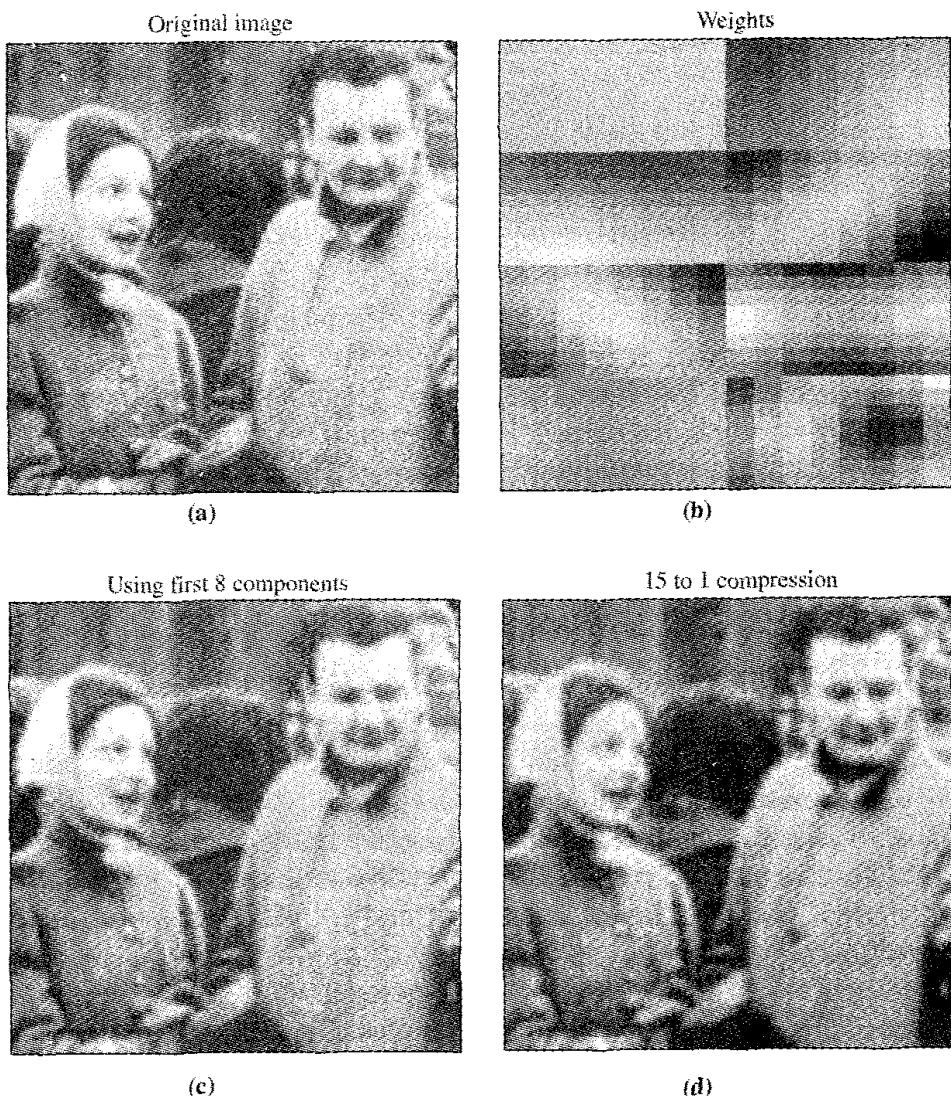


FIGURE 8.9 (a) An image of parents used in the image coding experiment. (b) 8×8 masks representing the synaptictic weights learned by the GHA. (c) Reconstructed image of parents obtained using the dominant 8 principal components without quantization. (d) Reconstructed image of parents with 15 to 1 compression ratio using quantization.

За вариация на първият образ, прилагаме ГХА на изображение на океан, показано на Фиг.8.10а. Това второ изображение подчертава текстурната информация. Фиг.8.10b показва 8×8 маски на синаптичните тегла, изучени от мрежата, подхождайки по описания начин. Обърнете внимание на разликата между тези маски и тези на фиг. 8.9b. Фиг. 8.10c показва реконструирианият образ на сцената с океяна, въз основа на доминиращите 8 основни компонента, без квантуване. За да се проучи ефекта на квантуване, резултатите от първите 2 маски са квантувани с 5 бита, третите с 3 бита и оставащите 5 маски с 2 бита всичка. Така общо 23 бита са необходими за кодиране на всеки 8×8 блок поксела, даващи резултатен процент от 0,36 бита на пиксел. Фиг.8.10d показва реконструирианият образ на сцената с океяна, използвайки собствените си маски, квантувани по току-що описания начин. Компресионното съотношение на този образ е 22 към 1.

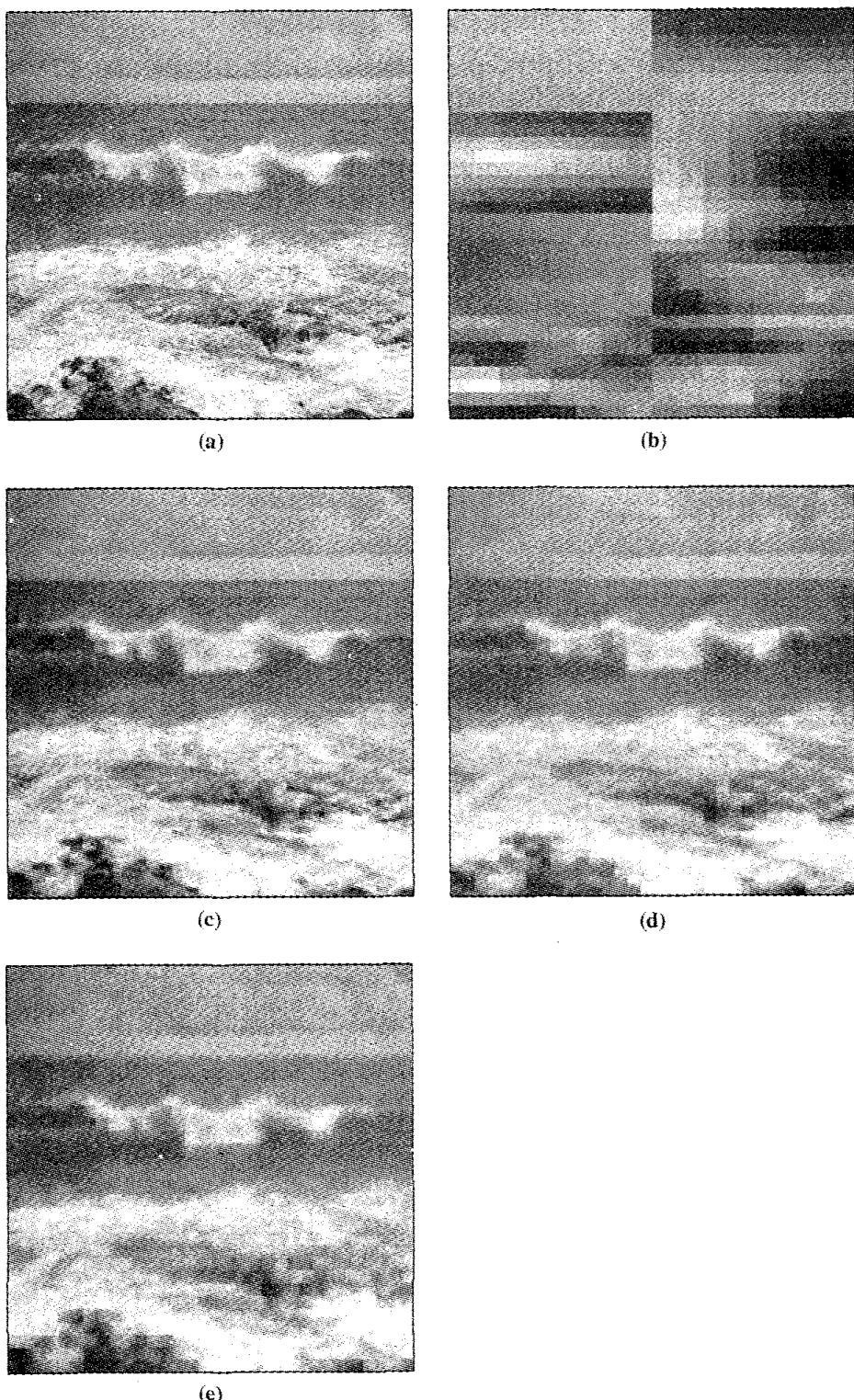


FIGURE 8.10 (a) Image of ocean scene. (b) 8×8 masks representing the synaptic weights learned by the GHA algorithm applied to the ocean scene. (c) Reconstructed image of ocean scene using 8 dominant principal components. (d) Reconstructed image of ocean scene with 22 to 1 compression ratio, using masks of part (b) with quantization. (e) Reconstructed image of ocean scene using the masks of Fig. 8.9(b) for encoding, with quantization for a compression of 22 to 1, same as that in part (d).

За да тестваме „обобщеното“ изпълнение на Генерализираният Хебиянов Алгоритъм, на края използваме маската от Фиг.8.10а и прилагаме същото квантуване, което е използвано за генериране на реконструирания образ на Фиг. 8.10d. Резултатът от това образно реконструиране е показано на Фиг.8.10e с компресно съотношение 22 към 1, същото като на Фиг.8.10d. Докато реконструираните изображения на Фиг.8.10d и 8.10e не са в голямо „съгласие“ едно с друго, може да се види, че Фиг.8.10d притежава повече от „истинската“ текстурна информация и по този начин излежда по-малко „ъгловато“ от Фиг.8.10e. Причината за това поведение се крие в мрежовата тежест. За обучението, проведено върху образите на родителите и океанска сцена, първите четири тежести са много сходни. Въпреки това, за изображението с родителите, последните четири тежести кодират крайната информация, но в случая с океанска сцена тези тежести кодират текстурната информация. В този смисъл, когато се кодира океанска сцена с крайните ъглови тежести, реконструкцията на текстурната дата е сурова, което води до ъгловат вид.

8.7 Адаптивен Основен компонентен анализ, който използва странично инхибиране

Генерализираният Хебиянов Алгоритъм, описан в предния раздел разчита изключително на използването на feedforward връзки за анализ на основните компоненти. В този раздел описваме друг алгоритъм, наречен Алгоритъм на основната компонентна екстракция (*adaptive principal components extraction (APEX)*) (Kung and Diamantaras, 1990; Diamantaras and Kung, 1996). APEX алгоритъмът използва както feedforward, така и feedback връзките. Алгоритъмът е повтарящ съвсем смисъл такъв, че ако сме дали първия ($j - 1$) основен компонент, j -тия основен компонент е лесно изчислим.

Фиг. 8.11 показва мрежовия модел, използван за получаването на APEX алгоритъма. Както и преди, входният вектор x има димензия m и компоненти, обозначени с x_1, x_2, \dots, x_m . Всеки неврон в мрежата се предполага, че е линеен. Както е описано на Фиг.8.11, има два вида на синаптични връзки в мрежата:

- *Feedforward връзки* - от входните възли до всеки от невроните $1, 2, \dots, j$, като $j < m$. От особен интерес тук са feedforward връзките към неврона j ; тези връзки са представени от feedforward тегловния вектор

$$\mathbf{w}_j = [w_{j1}(n), w_{j2}(n), \dots, w_{jm}(n)]^T$$

Feedforward връзките действат в съответствие с Хебияново обучаващото правило. Те са възбъдени и за това могат да се самоусилват.

- *Страницни връзки* - от отделните изходи на невроните $1, 2, \dots, j - 1$ до неврон j , като по този начин прилагат обратна връзка към мрежата. Тези връзки са представени от feedback тегловния вектор

$$\mathbf{a}_j(n) = [a_{j1}(n), a_{j2}(n), \dots, a_{j,j-1}(n)]^T$$

Страницните връзки работят в съответствие с анти-Хебияновото обучително правило, което има ефекта да ги прави инхибиторни.

На Фиг.8.11 feedforward и feedback връзките на неврона j са удебелени, за да подчертаят, че неврона j е предмета за изучаване.

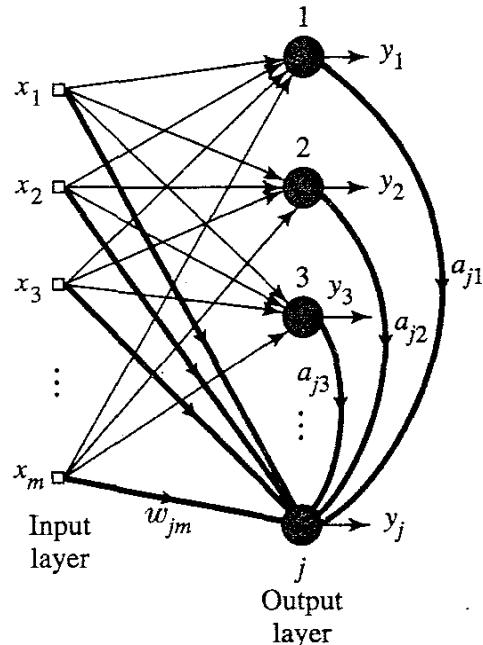


FIGURE 8.11 Network with feedforward and lateral connections for deriving the APEX algorithm.

Изходът $y_j(n)$ на неврона j е зададен от

$$y_j(n) = \mathbf{w}_j^T(n)\mathbf{x}(n) + \mathbf{a}_j^T(n)\mathbf{y}_{j-1}(n) \quad (8.98)$$

където приноса $\mathbf{w}_j^T(n)\mathbf{x}(n)$ е според feedforward връзката, а останалият принос $\mathbf{a}_j^T(n)\mathbf{y}_{j-1}(n)$ е според страницните връзки. Сигналният вектор за обратна връзка $\mathbf{y}_{j-1}(n)$ е дефиниран от изходите на невроните $1, 2, \dots, j - 1$:

$$\mathbf{y}_{j-1}(n) = [y_1(n), y_2(n), \dots, y_{j-1}(n)]^T \quad (8.99)$$

Предполага се, че входният вектор $\mathbf{x}(n)$ е съставен от стационарен процес, чиято корелационна матрица R има различни собствени стойности, подредени в низходящ ред, както следва:

$$\lambda_1 > \lambda_2 > \dots > \lambda_{j-1} > \lambda_j > \dots > \lambda_m \quad (8.100)$$

Освен това се предполага, че невроните $1, 2, \dots, j - 1$ от мрежата на Фиг.8.11 вече са се доближили до съответните им стабилни състояния, както е показано от

$$\mathbf{w}_k(0) = \mathbf{q}_k, \quad k = 1, 2, \dots, j - 1 \quad (8.101)$$

$$\mathbf{a}_k(0) = \mathbf{0}, \quad k = 1, 2, \dots, j - 1 \quad (8.102)$$

Където \mathbf{q}_k е собствения вектор асоцииран с к-тата собствена стойност на корелационната матрица R , а времевата стъпка $n = 0$ се отнася за началото на изчисленията от неврон j на мрежата. Тогава можем да използваме Ур-я (8.98), (8.99), (8.101) и (8.102), за да напишем:

$$\begin{aligned} \mathbf{y}_{j-1}(n) &= [\mathbf{q}_1^T \mathbf{x}(n), \mathbf{q}_2^T \mathbf{x}(n), \dots, \mathbf{q}_{j-1}^T \mathbf{x}(n)] \\ &= \mathbf{Q} \mathbf{x}(n) \end{aligned} \quad (8.103)$$

където \mathbf{Q} е $(j - 1)$ -към- m матрица, определена от гледна точка на собствените вектори $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{j-1}$, свързани с $(j - 1)$ най-големи собствени стойности $\lambda_1, \lambda_2, \dots, \lambda_{j-1}$ на корелационната матрица R . И това е

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{j-1}]^T \quad (8.104)$$

Изискването е да се използва неврон в мрежата на Фиг.8.11 за изчисляване на следващата най-голяма собствена стойност λ_j на корелационната матрица R , на входния вектор $\mathbf{x}(n)$ и асоциираният собствен вектор \mathbf{q}_j .

Актуализираното уравнение за feedforward тегловния вектор $\mathbf{w}_j(n)$ и feedback тегловния вектор $\mathbf{a}_j(n)$ за неврона j са определени, съответно:

$$\mathbf{w}_j(n + 1) = \mathbf{w}_j(n) + \eta [y_j(n) \mathbf{x}(n) - y_j^2(n) \mathbf{w}_j(n)] \quad (8.105)$$

и

$$\mathbf{a}_j(n + 1) = \mathbf{a}_j(n) - \eta [y_j(n) \mathbf{y}_{j-1}(n) + y_j^2(n) \mathbf{a}_j(n)] \quad (8.106)$$

Където η е обучителният процентен параметър, приет да бъде еднакъв и за двете актуализирани уравнения. Понятието $y_j(n) \mathbf{x}(n)$ от дясната страна на ур-я (8.106) представлява Хебияново обучение, а понятието $-y_j(n) \mathbf{y}_{j-1}(n)$ от дясната на ур-е (8.106) представлява анти-Хебияново обучение. Останалите понятия $-y_j^2(n) \mathbf{w}_j(n)$ и $y_j^2(n) \mathbf{a}_j(n)$ са включени в тези две ур-я, за да осигурят стабилността на алгоритъма. По принцип, ур-е(8.105) е векторната форма на обучителното правило на Oja, описано в ур-е(8.40), докато ур-е(8.106) е ново, предвидено за използванет на страничното инхибиране (Kung and Diamantaras, 1990; Diamantaras and Kung, 1996).

Доказваме абсолютната стабилност на невронната мрежа на Фиг.8.11 по индукция, както следва:

- Първо, доказваме, че ако неврони $1, 2, \dots, j - 1$ са се доближили до своите стабилни състояния, тогава неврон j се приближава до своето собствено стабилно състояние, чрез извлечане на ледващата най-голяма собствена стойност λ_j на корелационната матрица R на входния вектор $\mathbf{x}(n)$ и съответния собствен вектор \mathbf{q}_j .

- Следващо, ние вече сме завършили доказателството чрез индукция доказвайки, че неврон 1 все още няма обратна връзка ои следователно вектора на обратната връзка е с тегло нула. Оттук този конкретен неврон работи по абсолютно същия начин, както неврона на Оя и от раздел 8.4 знаем, че този неврон е абсолютно стабилен при определени условия.

Единственият въпрос следователно, който изисква внимание е първата точка.

За да продължиме, можем да се позовем на основните предположения, направени в раздел 8.4 и така да извлечем следната теорема, която е в контекста на неврон j в невронната мрежа от Фиг.8.11, оперираща при условията, описани от Ур-я (8.105) и (8.106) (Kung and Diamantaras, 1990; Diamantaras and Kung, 1996):

Дадено ни е, че на учебния процентов параметър η е дадена достатъчно малка стойност, за да се гарантира, че промените в тегловния вектор вървят бавно. Тогава в границата, feedforward тегловния вектор и средната изходна мощност (вариацията) на неврона j се доближат до нормализирания собствен вектор \mathbf{q}_j и съответно собствената стойност λ_j на корелационната матрица R , както е показано:

$$\lim_{n \rightarrow \infty} \mathbf{w}_j(n) = \mathbf{q}_j$$

и

$$\lim_{n \rightarrow \infty} \sigma_j^2(n) = \lambda_j$$

където $\sigma_j^2(n) = E[y_j^2(n)]$ и $\lambda_1 > \lambda_2 > \dots > \lambda_i > \dots > \lambda_m > 0$. С други думи, давайки на собствените вектори $\mathbf{q}_1, \dots, \mathbf{q}_{i-1}$, неврон j в мрежата на Фиг.8.11 изчислява следващата най-голяма собствена стойност λ_j и свързания с тях собствен вектор \mathbf{q}_j .

За да докажем тази теорема, взимаме първо под внимание Ур-е(8.105). Използвайки ур-е(8.8) и (8.99) и приемайки, че :

$$\mathbf{a}_j^T(n) \mathbf{y}_{j-1}(n) = \mathbf{y}_{j-1}^T(n) \mathbf{a}_j(n)$$

Може да преработим ур-е (8.105), както следва:

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}_j(n) + \mathbf{x}(n)\mathbf{x}^T(n)\mathbf{Q}^T\mathbf{a}_j(n) - y_j^2(n)\mathbf{w}_j(n)] \quad (8.107)$$

Където матрицата \mathbf{Q} е дефинирана от ур-е (8.104). Частта $y_j^2(n)$ в ур-е(8.107) не е променяна поради причина, която ще стане ясна. Позовавайки се на фундаменталните предположения, описани в раздел 8.4, ние откриваме, че прилагането на статистическият очаквателен оператор от двете страни на ур-е (8.107), дава

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta[\mathbf{R}\mathbf{w}_j(n) + \mathbf{R}\mathbf{Q}^T\mathbf{a}_j(n) - \sigma_j^2(n)\mathbf{w}_j(n)] \quad (8.108)$$

където R е корелационната матрица на входния вектор $\mathbf{x}(n)$ и $\sigma_j^2(n)$ е средната изходна мощност на неврона j . Нека синаптичният тегловен вектор $\mathbf{w}_j(n)$ се разшири по отношение на целия орто-нормален набор от собствени вектори на корел. Матрица R , както следва:

$$\mathbf{w}_j(n) = \sum_{k=1}^m \theta_{jk}(n) \mathbf{q}_k \quad (8.109)$$

Където \mathbf{q}_k е собствения вектор, асоцииран със собствената стойност λ_k на матрицата R и $\theta_{jk}(n)$ е времеви-вариращ коефициент на разширението. Тогава може да използваме основната връзка (Вижте ур-е(8.14))

$$R\mathbf{q}_k = \lambda_k \mathbf{q}_k$$

За да отразим матричният продукт $R\mathbf{w}_j(n)$ както следва:

$$\begin{aligned} R\mathbf{w}_j(n) &= \sum_{k=1}^m \theta_{jk}(n) R\mathbf{q}_k \\ &= \sum_{k=1}^m \lambda_k \theta_{jk}(n) \mathbf{q}_k \end{aligned} \quad (8.110)$$

По подобен начин, използвайки ур-е(8.104) може да се изрази матричния продукт $RQ^T \mathbf{a}_j(n)$ като:

$$\begin{aligned} RQ^T \mathbf{a}_j(n) &= R[\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{j-1}] \mathbf{a}_j(n) \\ &= [\lambda_1 \mathbf{q}_1, \lambda_2 \mathbf{q}_2, \dots, \lambda_{j-1} \mathbf{q}_{j-1}] \begin{bmatrix} a_{j1}(n) \\ a_{j2}(n) \\ \vdots \\ a_{j,j-1}(n) \end{bmatrix} \\ &= \sum_{k=1}^{j-1} \lambda_k a_{jk}(n) \mathbf{q}_k \end{aligned} \quad (8.111)$$

Следователно, замествайки ур-я (8.109), (8.110) и (8.111) в (8.108), и опростявайки, получаваме (Kung and Diamantaras, 1990)

$$\begin{aligned} \sum_{k=1}^m \theta_{jk}(n+1) \mathbf{q}_k &= \sum_{k=1}^m \{1 + \eta[\lambda_k - \sigma_j^2(n)]\} \theta_{jk}(n) \mathbf{q}_k \\ &\quad + \eta \sum_{k=1}^{j-1} \lambda_k a_{jk}(n) \mathbf{q}_k \end{aligned} \quad (8.112)$$

В резултат на процедура, подобна на описаната е възможно да се покаже, че актуализационното уравнение (8.106) за feedback тегловния вектор $a_j(n)$ може да се трансформира, както следва (Виж проблем 8.7):

$$\mathbf{a}_j(n+1) = -\eta \lambda_k \theta_{jk}(n) \mathbf{1}_k + \{1 - \eta[\lambda_k + \sigma_j^2(n)]\} \mathbf{a}_j(n) \quad (8.113)$$

Където 1_k е вектор, на който всички j елементи са нула, освен k -тия, който е равен на 1. Индексът k е ограничен да лежи в диапазона $1 \leq k \leq j - 1$.

Трябва да се разгледат два случая, в зависимост от стойността дадена на k , по отношение на $j - 1$. Случай I се отнася до $1 \leq k \leq j - 1$, което се отнася до анализа за „старите“ основни видове на мрежата. Случай II се отнася до $j \leq k \leq m$, който се отнася до останалия „нов“ главен режим. Общият брой на основните режими е m , размера на входния вектор $\mathbf{x}(n)$.

Случай I: $1 \leq k \leq j - 1$

В този случай можем да изведем следните актуализационни уравнения за коефициент $\theta_{jk}(n)$, асоцииран със собствения вектор \mathbf{q}_k и feedback теглото $a_{jk}(n)$ от ур-я (8.112) и (8.113), респективно:

$$\theta_{jk}(n + 1) = \eta \lambda_k a_{jk}(n) + \{1 + \eta[\lambda_k - \sigma_j^2(n)]\} \theta_{jk}(n) \quad (8.114)$$

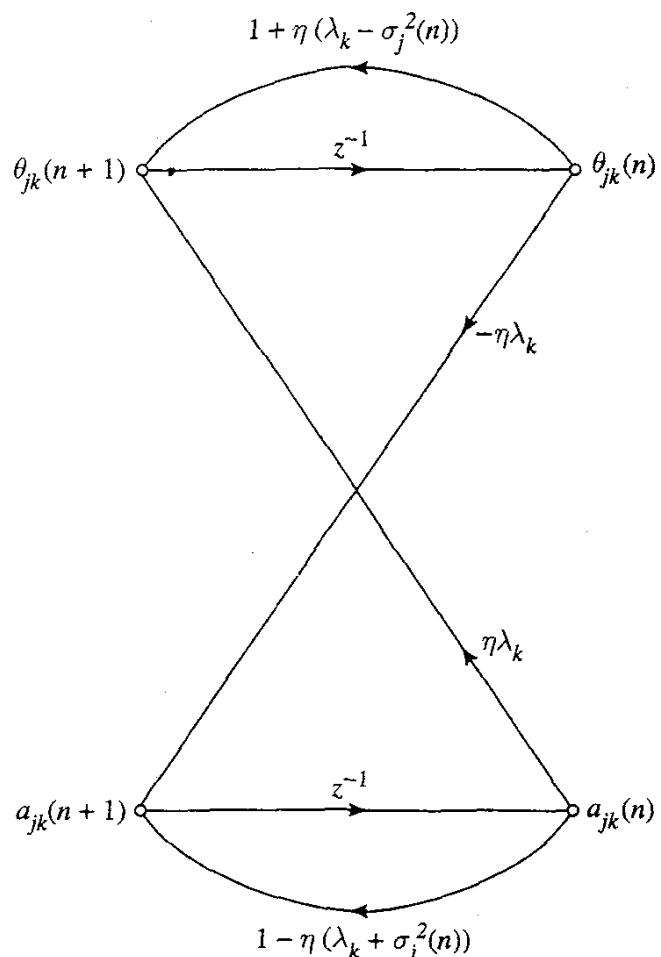


FIGURE 8.12 Signal-flow graph representation of Eqs. (8.114) and (8.115).

и

$$a_{jk}(n + 1) = -\eta\lambda_k \theta_{jk}(n) + \{1 - \eta[\lambda_k + \sigma_j^2(n)]\} a_{jk}(n) \quad (8.115)$$

Фиг.8.12 представя сигнално-потоковото графично представяне на Ур-я(8.114) и (8.115).

В матрична форма можем да пренапишем Ур-я (8.114) и (8.115) като

$$\begin{bmatrix} \theta_{jk}(n+1) \\ a_{jk}(n+1) \end{bmatrix} = \begin{bmatrix} 1 + \eta[\lambda_k - \sigma_j^2(n)] & \eta\lambda_k \\ -\eta\lambda_k & 1 - \eta[\lambda_k + \sigma_j^2(n)] \end{bmatrix} \begin{bmatrix} \theta_{jk}(n) \\ a_{jk}(n) \end{bmatrix} \quad (8.116)$$

Матрицата на системата, описана в ур-е (8.116) има двойна собствена стойност в

$$\rho_{jk} = [1 - \eta\sigma_j^2(n)]^2 \quad (8.117)$$

От Ур-е (8.117) можем да направим две важни наблюдения:

1. Двойната собствена стойност ρ_{jk} на системната матрица от ур-е (8.116) е независима от собствените стойности λ_k на корелационната матрица R , съответстващи на $k = 1, 2, \dots, j-1$.
2. За всички k , двойната собствена стойност ρ_{jk} зависи единствено от обучително-процентовия параметър η и средната мощност на изхода σ_j^2 на неврона j . За това е по-малко, отколкото единството, при условие, че η е достатъчно малко, положително число.

Като се има предвид, че $\rho_{jk} < 1$, коефициента $\theta_{jk}(n)$ на разширението в ур-е (8.109) и feedback тежестта $a_{jk}(n)$ ще достигне нула асимптотично с еднаква скорост, за всички k , тъй като всички основни модове на мрежата имат еднаква собствена стойност (Кунг и Диамантарас, 1990; Диамантарас и Кунг, 1996). Този резултат е в последствие на свойството, че ортогоналността на собствените вектори на корелационната матрица не зависят от собствените стойности. С други думи, експанзията на $w_j(n)$ в смисъл на ортогоналния набор от собствени вектори на корелационната матрица R , дадено в Ур-е (8.109), което е основата на резултата, описан в Ур-е(8.117) е инвариантно на избора на собствените стойности $\lambda_1, \lambda_2, \dots, \lambda_{j-1}$.

Случай II: $j \leq k \leq m$

В този втори случай, feedback теглата $a_{jk}(n)$ не оказват влияние върху режима на мрежата, както е показано от

$$a_{jk}(n) = 0 \quad \text{for } j \leq k \leq m \quad (8.118)$$

Следователно, за всеки основен режим $k \leq j$ имаме много просто уравнение:

$$\theta_{jk}(n+1) = \{1 + \eta[\lambda_k - \sigma_j^2(n)]\}\theta_{jk}(n) \quad (8.119)$$

Което следва директно от Ур-я (8.112) и (8.118). според Случай I, двете $\theta_{jk}(n)$ и $a_{jk}(n)$ евентуално ще клонят към нула за $k = 1, 2, \dots, j-1$. Със случайната променлива $Y_j(n)$, представляваща изхода на неврон j можем да изразим неговата средна изходнамощност, както следва:

$$\begin{aligned}\sigma_j^2(n) &= E[Y_j^2(n)] \\ &= \sum_{k=j}^m \lambda_k \theta_{jk}^2(n)\end{aligned}\tag{8.120}$$

Където в последния ред използвахме следната връзка:

$$\mathbf{q}_k^T \mathbf{R} \mathbf{q}_l = \begin{cases} \lambda_k, & l = k \\ 0; & \text{otherwise} \end{cases}$$

От тук следва, че ур-е (8.119) не може да се отклонява, защото всеки път, когато $\theta_{jk}(n)$ е голяма, такава че $\sigma_j^2(n) > \lambda_k$, тогава $1 + \eta[\lambda_k - \sigma_j^2(n)]$ става по-малка от единица, в който случай $\theta_{jk}(n)$ ще намалее в магнитуда. Нека инициализираме алгоритъма, с $\theta_{jj}(0) \neq 0$. Също дефинираме

$$r_{jk}(n) = \frac{\theta_{jk}(n)}{\theta_{jj}(n)}, \quad k = j + 1, \dots, m\tag{8.121}$$

Тогава можем да използваме ур-е (8.119), за да напишем

$$r_{jk}(n+1) = \frac{1 + \eta[\lambda_k - \sigma_j^2(n)]}{1 + \eta[\lambda_j - \sigma_j^2(n)]} r_{jk}(n)\tag{8.122}$$

Със собствените стойности на корелационната матрица, подредени в низходящ ред

$$\lambda_1 > \lambda_2 > \dots > \lambda_k > \dots > \lambda_j \dots > \lambda_m$$

Следва, че

$$\frac{\theta_{jk}(n)}{\theta_{jj}(n)} < 1 \quad \text{for all } n, \text{ and for } k = j + 1, \dots, m\tag{8.123}$$

Още, забелязваме от ур-я (8.119) и (8.120), че $\theta_{jj}(n+1)$ остава ограничена; за това,

$$r_{jk}(n) \rightarrow 0 \quad \text{as } n \rightarrow \infty \text{ for } k = j + 1, \dots, m\tag{8.124}$$

Еквивалентно, използвайки дефиницията, дадена в ур-е (8.121) може да заявим, че

$$\theta_{jk}(n) \rightarrow 0 \quad \text{as } n \rightarrow \infty \text{ for } k = j + 1, \dots, m\tag{8.125}$$

При това условие, ур-е (8.120) се опростява до

$$\sigma_j^2(n) = \lambda_j \theta_{jj}^2(n)\tag{8.126}$$

И така ур-е (8.119) за $k = j$ става

$$\theta_{jj}(n+1) = \{1 + \eta \lambda_j [1 - \theta_{jj}(n)]\} \theta_{jj}(n)\tag{8.127}$$

От това уравнение ние веднага извеждаме

$$\theta_{jj}(n) \rightarrow 1 \quad \text{as } n \rightarrow \infty \quad (8.128)$$

Последствията от това ограничително условие и това на ур-е (8.125) са двупосочни:

1. От ур-е (8.126) имаме

$$\sigma_j^2(n) \rightarrow \lambda_j \quad \text{as } n \rightarrow \infty \quad (8.129)$$

2. от ур-е (8.109) имаме

$$\mathbf{w}_j(n) \rightarrow \mathbf{q}_j \quad \text{as } n \rightarrow \infty \quad (8.130)$$

С дури думи, невронно мрежовия модел от Фиг.8.11 извежда j -та собствена стойност и асоциативния собствен вектор на корелационната матрица R на входния вектор $\mathbf{x}(n)$, докато броя на итерациите n достигне безкрайност. Това разбира се предполага, че невроните $1, 2, \dots, j-1$ вече са се доближили до собствените стойности и асоциативните им собствени вектори на корелационната матрица R .

Използване на APEX алгоритъмът, представен тук, почива на предпоставката, че невроните $1, 2, \dots, j-1$ са се доближили преди неврона j да започне да действа. Това беше направено само за обяснение на работата на алгоритъма по прост начин. На практика, обаче, невроните в APEX алгоритъма са слонни да се събират заедно.

Обучителното оценяване(Обучителен процент) – Learning Rate

В APEx алгоритъмът, описан в ур-я (8.105) и (8.106) същият обучително-процентов параметър η се използва, както за актуализиране на feedforward тегловния вектор $w_j(n)$, така и за feedback тегловния вектор $a_j(n)$. Връзката на ур-е(8.117) може да бъде използвана, за да се определи оптималната стойност на обучително-процентовия параметър за всеки неврон j , чрез създаването на двойна собствена стойност ρ_{jk} , равна на нула. В такъв случай имаме

$$\eta_{j,\text{opt}}(n) = \frac{1}{\sigma_j^2(n)} \quad (8.131)$$

Където $\sigma_j^2(n)$ е средната изходна мощност на на неврона j . По-практично решение, обаче, е да се създаде (Кунг и Диамантарас, 1990; Диамантарас и Кунг, 1996):

$$\eta_j = \frac{1}{\lambda_{j-1}} \quad (8.132)$$

Което извежда една подценявана стойност на обучително-процентовия параметър, след като $\lambda_{j-1} > \lambda_j$ и $\sigma_j^2(n) \rightarrow \lambda_j$ като $n \rightarrow \infty$. Забележете, че собствената стойност λ_{j-1} е

изчислена от неврона $j - 1$ и следователно може да се използва при актуализирането на feedback и feedforward тежестта на неврона j .

Обобщение на APEX алгоритъмът

1. Инициализиране на feedforward тегловния вектор w_j и на feedback тегловния вектор a_j с малка произволна стойност с преме $n = 1$, където $j = 1, 2, \dots, m$. Присвояване на малка позитивна стойност на обучително-процентния параметър η .
2. Нека $j = 1$, за $n = 1, 2, \dots$, изчислете

$$y_1(n) = \mathbf{w}_1^T(n)\mathbf{x}(n)$$

$$\mathbf{w}_1(n + 1) = \mathbf{w}_1(n) + \eta[y_1(n)\mathbf{x}(n) - y_1^2(n)\mathbf{w}_1(n)]$$

Където $\mathbf{x}(n)$ е входния вектор. За голямо n , имаме $\mathbf{w}_j(n) \rightarrow \mathbf{q}_j$, където \mathbf{q}_j е собствения вектор асоцииран с най-голямата собствена стойност λ_j на корелационната матрица на $\mathbf{x}(n)$.

3. Нека $j = 2$, за $n = 1, 2, \dots$, изчислете

$$\mathbf{y}_{j-1}(n) = [y_1(n), y_2(n), \dots, y_{j-1}(n)]^T$$

$$y_j(n) = \mathbf{w}_j^T(n)\mathbf{x}(n) + \mathbf{a}_j^T(n)\mathbf{y}_{j-1}(n)$$

$$\mathbf{w}_j(n + 1) = \mathbf{w}_j(n) + \eta[y_j(n)\mathbf{x}(n) - y_j^2(n)\mathbf{w}_j(n)]$$

$$\mathbf{a}_j(n + 1) = \mathbf{a}_j(n) - \eta[y_j(n)\mathbf{y}_{j-1}(n) + y_j^2(n)\mathbf{a}_j(n)]$$

4. Увеличете j с 1, преминете към стъпка 3 и продължете, докато $j = m$, където m е желания брой от основни компоненти. (Забележете, че $j = 1$ оговаря на собствения вектор, асоцииран с най-голямата собствена стойност, за което сме се погрижили в стъпка 2.) За голямо n имаме $\mathbf{w}_j(n) \rightarrow \mathbf{q}_j$ и за $\mathbf{a}_j(n) \rightarrow \mathbf{0}$, където \mathbf{q}_j е собствения вектор, асоцииран с j -та собствена стойност на корелационната матрица на $\mathbf{x}(n)$.

8.8 Два класа на PCA Алгоритмите (Principal Components Analysis)

В допълнение към Генерализираният Хебиянов Алгоритъм(GHA), обсъден в раздел 8.5 и APEX алгоритъма, обсъден в раздел 8.7, няколко други алгоритъма за анализ на главните компоненти (PCA) са били съобщени в литературата*. Различните PCA алгоритми, използващи невронни мрежи, могат да бъдат категоризирани в два класа:

алгоритми за повторно оценяване и алгоритми за пре-аранжиране(reestimation algorithms and decorrelating algorithms).

Според тази класификация, GHA алгоритъм е алгоритъм за преоценяване в ур-то(8.87) и (8.88) може да бъде променено в еквивалентна форма

$$\mathbf{w}_j(n + 1) = \mathbf{w}_j(n) + \eta y_j(n)[\mathbf{x}(n) - \hat{\mathbf{x}}_j(n)] \quad (8.133)$$

Където $\hat{\mathbf{x}}_j(n)$ е „реестиматора“ определен от

$$\hat{\mathbf{x}}_j(n) = \sum_{k=1}^j \mathbf{w}_k(n)y_k(n) \quad (8.134)$$

В един reestimation алгоритъм невронната мрежа има само връзки за напред, чиито силни страни(тежести) са модифицирани по Хебиянов маниер. Успешните изходи на мрежата са принудени да научат различни основни компоненти, чрез извеждане на оценките на предходни компоненти от входа, преди данните да се включат в процеса на обучение.

В контраст с това, алгоритъмът на APEX е decorrelating алгоритъм. В такъв алгоритъм невронната мрежа има връзки, както за наред, така и за назад. Силите на връзката напред следват Хебиянов закон, докато силите на обратните връзки следват анти – Хебиянов закон. Последователните изхода на мрежата са decorrelated, принуждавайки мрежата да отговори на различни основни компоненти.

Principal Subspace – Основно подпространство

В ситуации, в които се изиска само *главното подпространство*(т.е. пространството на главните компоненти), може да използваме *симетричен модел*, при който реестиматора $\hat{\mathbf{x}}_j(n)$ в алгоритъма GHA е заместен от

$$\hat{\mathbf{x}}(n) = \sum_{k=1}^l \mathbf{w}_k(n)y_k(n) \quad \text{for all } l \quad (8.135)$$

В симетричният модел, дефиниран от ур-я (8.133) и (8.135), мрежата се клони към набор от изходи, които обхващат основното подпространство, отколкото самите основни компоненти. В конвергенция, тегловните вектори на мрежата са ортогонални помежду си, тъй както в GHA алгоритъма. Основното подпространство, както е описано тук, може да се разглежда като обобщение на класическото правило на Оя, дефинирано в ур-е (8.46).

8.9 Партидни и Адаптивни методи на изчисление

Дискусия на анализа на основните компоненти би била непълна без някои съображения с изчислителните аспекти на проблема. В този контекст, съществуват два основни подхода за изчисляване на основните компоненти: *партиден и адаптивен метод*.

Методът на собствено разлагане, описан в Раздел 8.3 и свързания с него метод на разлагане на единствена числов стойност, принадлежат на *партидната* категория. От друга страна, GHA и APEX алгоритмите, обсъдени в Раздели 8.5 и 8.7 принадлежат на *адаптивната* категория.

На теория, собственото разлагане се основава на общо-средна корелационна матрица R на произволен вектор $\mathbf{X}(n)$, както е описано в Раздел 8.3. На практика, ние използваме

Оценка на корелационната матрица R . Нека $\{\mathbf{x}(n)\}_{n=1}^N$ обозначи набор от N реализации на рандом вектора $\mathbf{X}(n)$, разпределен равномерно на малки моменти от време. Като се има предвид такъв набор от наблюдения, тогава може да използваме средната стойност на извадката, в смисъл на оценка на съответната матрица, както следва:

$$\hat{\mathbf{R}}(N) = \frac{1}{N} \sum_{n=1}^N \mathbf{x}(n)\mathbf{x}^T(n) \quad (8.136)$$

При условие, че околната средна, представявана от случаен вектор $\mathbf{X}(n)$ е ergodic, средната стойност на извадката $\hat{\mathbf{R}}(N)$ достига R , докато размерът на извадката N клони към безкрайност. На тази основа може да се прилага процедурата Собствено разлагане на извадката $\hat{\mathbf{R}}(N)$ и за това изчислявайки нейните собствени стойности и асоциираните с тях собствени вектори, като включим използването на Ур-е (8.22) с $\hat{\mathbf{R}}(N)$, използвано на мястото на R .

От числена гледна точка обаче, един по-добър метод е да се използва единствено числовата стойност на разлагането (*singular value decomposition (SVD)*), като се прилага директно към дата матрицата. За набора от наблюдения $\{\tilde{\mathbf{x}}(n)\}_{n=1}^N$ дата матрицата е дефинирана от

$$\mathbf{A} = [\mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(N)] \quad (8.137)$$

След това, с изключение на Коефициентът $1 / N$, лесно може да се види, че оценката $\hat{\mathbf{R}}(N)$ на корелационната матрица R е равна на матричния продукт $\mathbf{A}\mathbf{A}^T$. Според Теоремата за разлагане на единствена стойност, описана в Глава 5, дата матрицата $\mathbf{A}(n)$ може да бъде разложен, както следва (Голуб и Ван Лоан, 1996(Golub and Van Loan)):

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \quad (8.138)$$

Където \mathbf{U} и \mathbf{V} са ортогонални матрици, което значи, че

$$\mathbf{U}^{-1} = \mathbf{U}^T \quad (8.139)$$

и

$$\mathbf{V}^{-1} = \mathbf{V}^T \quad (8.140)$$

Колкото до матрицата Σ , има структура на фората

$$\Sigma = \begin{bmatrix} \begin{bmatrix} \sigma_1 & & & \mathbf{0} \\ & \sigma_2 & & \\ & & \ddots & \\ \mathbf{0} & & & \sigma_k \end{bmatrix} & \vdash \mathbf{0} \\ \vdash \mathbf{0} & \begin{bmatrix} & & \\ & & \\ & & \mathbf{0} \end{bmatrix} \end{bmatrix} \quad (8.141)$$

Където $k \leq m$ и където m е размера на наблюдавания вектор $\mathbf{x}(n)$. Числата $\sigma_1, \sigma_2, \dots, \sigma_k$ са наречени единични стойности (singular values) на дата матрицата \mathbf{A} . Съответно, колоните на правоъгълната матрица \mathbf{U} се наричат *леви сингуларни вектори*, а колоните на матрицата \mathbf{V} се нар. *десни сингуларни вектори*. Сингуларното стойностно разлагане на дата матрицата \mathbf{A} е свързано със собственото разлагане на оценката $\hat{\mathbf{R}}(N)$ на корелационната матрица по следните начини:

- Освен за скаларния фактор $1/\sqrt{N}$, сингуларната стойност на дата матрицата \mathbf{A} , са квадратните корени на собствените стойности на оценката $\hat{\mathbf{R}}(N)$.
- Левите сингуларни вектори на \mathbf{A} са собствените вектори на $\hat{\mathbf{R}}(N)$.

Сега можем да видим численото предимство на сингуларното разлагане на стойности над собственото разлагане. За предписана точност на изчисление, сингуларното стойностно разлагане изисква половината цифрова прецизност на процедурата на собственото разлагане. Освен това, няколко алгоритъма и високо точностни „консервирали“ съчетания са на разположение за изпълнение на единствена стойностна процедура за разлагане в един компютър (Голуб и Ван Лоан, 1996; Хайкин, 1996). На практика обаче, изискванията за съхранение може да ограничят използването на тези стойностни съчетания на размера на извадката, които не са твърде прекомерни.

Преминаваме към категорията на *адаптивните методи*, тези методи на работа с произволно голям размер на извадката N . За всички практически цели, няма ограничение на N . Адаптивните методи са приер за Хебиянови невронни мрежи, чиято експлоатация е вдъхновена от идеите от невробиологията. Изискванията за съхранението на тези методи са сравнително скромни, тъй като междуинните стойности на собствените стойности и свързаните с тях собствени вектори не е нужно да се съхраняват. Друга привлекателна черта на адаптивните алгоритми е, че в нестационарна околнна среда, те

имат присъщи възможности за проследяване на постепенните промени в оптималното решение по един евтин начин, в сравнение с партидните методи. Въпреки това, основния недостатък на адаптивните алгоритми на стохастичния тип е техния относително бавен темп на конвергенция, което ги поставя в неизгодно положение в сравнение с класическите техники на партидите; това е така, особено на големите стационарни проблеми, дори когато адаптивните методи се прилагат на паралелни невронни хардуери(Котилайнен, 1993).

8.10 Същностен Анализ на Основните Компоненти(KERNEL PRINCIPAL COMPONENTS ANALYSIS)

Формата на PCA обсъдена до този момент в главата включва изчисления във входното(дата) пространство. Сега ще разгледаме друга форма на PCA, където изчисленията се извършват в една отделна част от пространството, която е нелинейно свързана към входното пространство. Характерната (отделната) част от пространството, която имаме в предвид е определена от вътрешно-продуктово ядро, което е в съответствие с теоремата на Мерцер; понятието за вътрешно-продуктовото ядро е обсъдена в Глава 6 на Поддържащи вектор машини.

Поради нелинейната връзка между входното и функционалното пространство, ядрото PCA е нелинейно. Въпреки това, за разлика от други форми на нелинейни PCA, прилагането на PCA ядрото разчита на линейната алгебра. За това можем да мислим за ядрото PCA като за естествено продължение на обикновените PCA.

Нека вектор $\varphi(\mathbf{x}_j)$ обозначи изображението на входния вектор \mathbf{x}_j , индуциран във функционално пространство, определено от нелинейната карта : $\varphi: \mathbb{R}^{m_0} \rightarrow \mathbb{R}^{m_1}$, където m_0 е размерността на входното пространство и m_1 е размерността на функционалното пространство. Дадено ни е набор от примери $\{\tilde{\mathbf{x}}_i\}_{i=1}^N$, като имаме съответно и набор от функционални вектори $\{\varphi(\mathbf{x}_i)\}_{i=1}^N$. Съответно можем да дефинираме една $m_1 -$ към – m_1 корелационна матрица във функционалното пространство, обозначена с $\tilde{\mathbf{R}}$, както следва:

$$\tilde{\mathbf{R}} = \frac{1}{N} \sum_{i=1}^N \varphi(\mathbf{x}_i) \varphi^T(\mathbf{x}_i) \quad (8.142)$$

Както и при нормални PCA, първото нещо, което трябва да подсигурим е набора от функционални вектори $\{\varphi(\mathbf{x}_i)\}_{i=1}^N$ да имат нулево значение:

$$\frac{1}{N} \sum_{i=1}^N \varphi(\mathbf{x}_i) = \mathbf{0}$$

За да удовлетворим това условие във функционалното пространство е по-сложно, отколкото във входното пространство; в проблем 8.10 описахме процедура за кетъринг на това изискване, въз основа на предположението, че функционалните вектори са центрирани, можем да адаптираме ур-е (8.14) за нашата сегашна ситуация

$$\tilde{\mathbf{R}}\tilde{\mathbf{q}} = \tilde{\lambda}\tilde{\mathbf{q}} \quad (8.143)$$

Където $\tilde{\lambda}$ е собствена стойност на корелационната матрица $\tilde{\mathbf{R}}$, а $\tilde{\mathbf{q}}$ е асоциираният собствен вектор. Забелязваме, че всички собствени вектори, които удовлетворяват ур-е (8.143) за $\tilde{\lambda} \neq 0$ лежат в участъка на набора от функционални вектори $\{\varphi(\mathbf{x}_j)\}_{j=1}^N$.

Следователно не съществува съответен набор от коефициенти $\{\alpha_j\}_{j=1}^N$, за които можем да напишем:

$$\tilde{\mathbf{q}} = \sum_{j=1}^N \alpha_j \varphi(\mathbf{x}_j) \quad (8.144)$$

Така заместваме ур-е (8.142) и (8.144) в (8.143), и получаваме

$$\sum_{i=1}^N \sum_{j=1}^N \alpha_j \varphi(\mathbf{x}_i) K(\mathbf{x}_i, \mathbf{x}_j) = N \tilde{\lambda} \sum_{j=1}^N \alpha_j \varphi(\mathbf{x}_j) \quad (8.145)$$

Където $K(\mathbf{x}_i, \mathbf{x}_j)$ е вътрешно-продуктово ядро, определено по отношение на функционалните вектори

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi^T(\mathbf{x}_i) \varphi(\mathbf{x}_j) \quad (8.146)$$

Трябва да отидеме една стъпка напред с ур-е (8.154), така че връзката се изразява изцяло по отношение на вътрешния продукт на ядрото. За да направиме това, ние трябва да преумножим и от двете страни на уравнението (8.145) с транспонирания вектор $\varphi^T(\mathbf{x}_k)$, за това получаваме

$$\sum_{i=1}^N \sum_{j=1}^N \alpha_j K(\mathbf{x}_k, \mathbf{x}_i) K(\mathbf{x}_i, \mathbf{x}_j) = N \tilde{\lambda} \sum_{j=1}^N \alpha_j K(\mathbf{x}_k, \mathbf{x}_j), \quad k = 1, 2, \dots, N \quad (8.147)$$

Където дефиницията на $K(\mathbf{x}_k, \mathbf{x}_i)$ и $K(\mathbf{x}_k, \mathbf{x}_j)$ следвайки ур-е (8.146).

Сега представяме две матрични дефиниции:

- N-към-N матрицата K , наречена матрица на ядрото(или още кернел матрица), където е ij -тия елемент във вътрешно-продуктовото ядро $K(\mathbf{x}_i, \mathbf{x}_j)$

- N-към-1 вектора α , чийто j-ти елемент е коефициента α_j .

Съответно можем да преработим и ур-е (8.147) в компактната матрична форма:

$$\mathbf{K}^2 \alpha = N \tilde{\lambda} \mathbf{K} \alpha \quad (8.148)$$

Където матрицата на квадрат \mathbf{K}^2 обозначава произведението на \mathbf{K} със самата себе си. Тъй като \mathbf{K} е обща и за двете страни на ур-то (8.148), всичките решения на проблема за собствените стойности, които са от интерес са еднакво добре представени в опростения проблем за собствената стойност:

$$\mathbf{K} \alpha = N \tilde{\lambda} \alpha \quad (8.149)$$

Нека $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$ бележат собствените стойности на кернел матрицата \mathbf{K} ; това е

$$\lambda_j = N \tilde{\lambda}_j, \quad j = 1, 2, \dots, N \quad (8.150)$$

Където $\tilde{\lambda}_j$ е j-тата собствена стойност на корелационната матрица $\tilde{\mathbf{R}}$. Тогава ур-е (8.149) приема стандартната форма

$$\mathbf{K} \alpha = \lambda \alpha \quad (8.151)$$

Където коефициент-вектора α играе ролята на собствен вектор, асоцииран със собствената стойност λ на кернел матрицата \mathbf{K} . Векторът α е нормализиран, като се изисква собственият вектор $\tilde{\mathbf{q}}$ на корел.матрица $\tilde{\mathbf{R}}$ да е нормализиран до единица дължина; това е

$$\tilde{\mathbf{q}}_k^T \tilde{\mathbf{q}}_k = 1 \quad \text{for } k = 1, 2, \dots, p \quad (8.152)$$

Където е прието, че собствените стойности са аранжирани по низходящ ред, с λ р най-малка ненулева собствена стойност на кернел матрицата \mathbf{K} . Използвайки ур-е (8.144) и след това ур-е (8.151), може да покажем, че нормализираното състояние на ур-е (8.152) е еквивалентно на

$$\alpha_k^T \alpha_k = \frac{1}{\lambda_k}, \quad k = 1, 2, \dots, p \quad (8.153)$$

За извлечането на основните компоненти, трябва да изчислим прогнозите на собствените вектори $\tilde{\mathbf{q}}_k$ във функционалното пространство, както е показано

$$\begin{aligned} \tilde{\mathbf{q}}_k^T \varphi(\mathbf{x}) &= \sum_{j=1}^N \alpha_{k,j} \varphi^T(\mathbf{x}_j) \varphi(\mathbf{x}) \\ &= \sum_{j=1}^N \alpha_{k,j} K(\mathbf{x}_j, \mathbf{x}), \quad k = 1, 2, \dots, p \end{aligned} \quad (8.154)$$

Където векторът \mathbf{x} е „тест” точка, а a_{kj} е j -тия коефициент на собствения вектор \mathbf{a}_k асоцииран с k -тата собствена стойност на матрицата \mathbf{K} . Проекциите на ур-е(8.154) определят нелинейните основни компоненти в m_1 -димензионално характеристично пространство.

Фиг.8.13 илюстрира основната идея на ядрото PCA, където функционалното пространство е нелинейно свързано с входното пространство, чрез трансформацията $\varphi(\mathbf{x})$. Частите a и b се отнасят за входното пространство и функционалното пространство, респективно. Контурните линии, показани на Фиг.8.13b представляват постоянна проекция върху основен собствен вектор, който е изобразен като пунктирена стрелка. На тази фигура е прието, че $\varphi(\mathbf{x})$ е избрано по такъв начин, че образите на самите данни точки включени в характеристичното пространство се събират заедно на собствения вектор. Фиг.8.13a изобразява нелинеарните контурни линии на входното пространство, които съответстват на тези от функционалното пространство. Забележете, че нарочно не сме показали предварителното изображение на собствения вектор във входното пространство, като всъщност той може и изобщо да не съществува(Шолкопф и др., 1998).

За вътрешно-продуктови ядра, определени в съответствие с теоремата на Мерцер, ние основно работим с обикновени PCA в m_1 -димензионални характеристични пространства, където димензията m_1 е конструктивен параметър. Всички свойства на обикновените PCA, които са описани в Раздел 8.3 се отнасят и за ядрата PCA. И по-специално, ядрото PCA е линейно във функционалното пространство, но нелинейно във входното. Като такова, може да се прилага във всички области, в които обикновените PCA са били използвани за извлечане или намаляване на данните, за които нелинейното удължаване би имало смисъл.

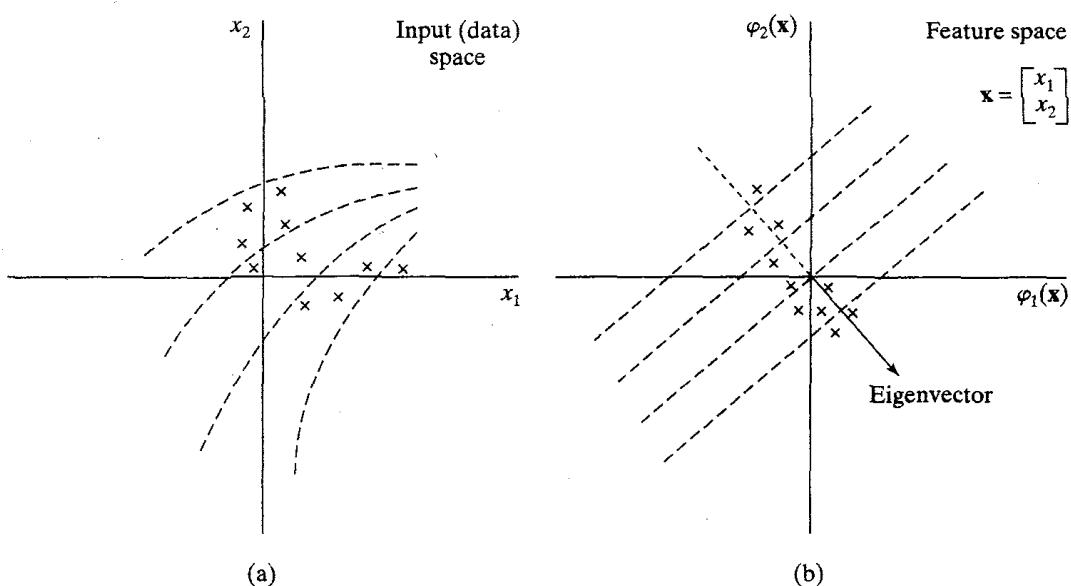


FIGURE 8.13 Illustration of the kernel PCA. (a) Two-dimensional input space, displaying a set of data points. (b) Two-dimensional feature space, displaying the induced images of the data points congregating around a principal eigenvector. The uniformly spaced dashed lines in part (b) represent contours of constant projections onto the eigenvector; the corresponding contours are nonlinear in the input space.

В Глава 6 представихме три метода за конструиране на вътрешно-продуктови ядра, които бяха базирани на използването на полиноми, радиално-базирани функции и хиперболични функции; вижте Таблица 6.1. въпросът, как да изберете ядрото, което да отговаря най-добре на поставената задача(т.е. подходящо функционално пространство) е отворен проблем (Шолкопф, 1997).

Обобщение на ядрото PCA

1. Като се има в предвид тренировъчните примери $\{\mathbf{x}_i\}_{i=1}^N$ изчислете N-към-N кернел матрицата $\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}$, където

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}^T(\mathbf{x}_i) \boldsymbol{\varphi}(\mathbf{x}_j)$$

2. Решете проблема със собствената стойност:

$$\mathbf{K}\mathbf{a} = \lambda \mathbf{a}$$

където λ е собствена стойност на кернел матрицата K и алфа е асоциирания собствен вектор.

3. нормализирайте собствения вектор така изчислен, като се има в предвид

$$\boldsymbol{\alpha}_k^T \boldsymbol{\alpha}_k = \frac{1}{\lambda_k}, \quad k = 1, 2, \dots, p$$

Където λ_p е най-малката ненулева собствена стойност на матрицата K, имайки в предвид, че собствените стойности са аранжирани по низходящ ред.

4. за извличането на основни компоненти на тест точка x, изчислете проекциите

$$\begin{aligned} a_k &= \tilde{\mathbf{q}}_k^T \boldsymbol{\varphi}(\mathbf{x}) \\ &= \sum_{j=1}^N \alpha_{k,j} K(\mathbf{x}_j, \mathbf{x}), \quad k = 1, 2, \dots, p \end{aligned}$$

където $\alpha_{k,j}$ е j-тия елемент на собствения вектор a_k .

Пример 8.3

За да предоставим няколко интуитивни разбирания за работата на ядрото PCA, покажахме на Фиг.8.14 резултатите от един прост експеримент, описан в Шолкопф и др.(1998г.). Двузимерни данни, състоящи се от компоненти x_1 и x_2 ,

използвани в този експеримент са били генериирани, както следва: x_1 стойностите са равномерно разпределени в интервала $[-1, 1]$. А x_2 - стойностите са нелинейно свързани с x_1 стойностите чрез формулата:

$$x_2 = x_1^2 + v$$

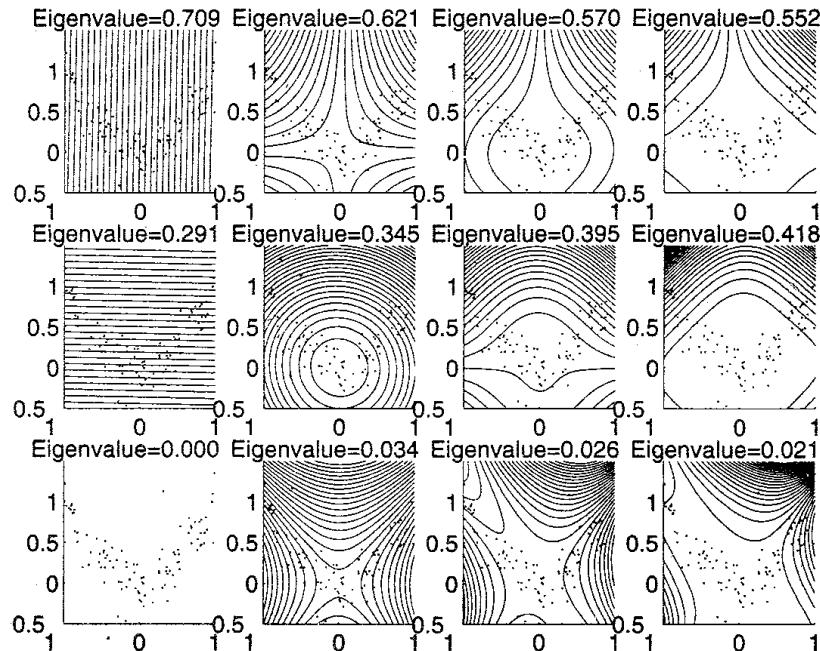


FIGURE 8.14 Two-dimensional example illustrating the kernel PCA. From left to right, the polynomial degree of the kernel is $d = 1, 2, 3, 4$. From top to bottom, the first three eigenvectors in the feature space are shown. The first column corresponds to ordinary PCA, and the other three columns correspond to kernel PCA with polynomial degree $d = 2, 3, 4$. (Reproduced with permission from Dr. Klaus-Robert Müller.)

Където v е адитивен Гаузионен шум с нулево значение и вариация 0.04. Резултата на PCA показан на Фиг.8.14 са получени при използването на ядрени полиноми:

$$K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x}^T \mathbf{x}_i)^d, \quad d = 1, 2, 3, 4$$

Където $d = 1$ отговаря на линеарен PCA, а $d = 2, 3, 4$ отговаря на ядрен PCA. Линеарният PCA отляво на Фиг.8.14 извежда резултат само в две собствени стойности, след като димензията на входното пространство е две. В контраст с това, ядрения PCA позволява извеждането на компоненти от по-висок ред, както е показано от резултатите от колони 2,3 и 4 на Фиг.8.14, кореспондиращи с полиноминалната степен $d = 2, 3, 4$, респективно. Контурните линии показани във всяка част на фигурата(с изключение на нулевата стойност в случая на линеарен PCA) представляват константни основни стойности(т.е. константни проекции фърху собствения вектор асоцииран със собствената стойност).

Базирайки се на резултатите показани на Фиг.8.14 правим следните наблюдения:

- Както се очакваше, линейния PCA не успя да осигури адекватно представяне на нелинейно въвеждане на данни.
- Във всеки случай, пържият основен компонент варира монотонно по парабола, която е в основата на входните данни.
- В ядрения PCA, втория и третия основен компонент демонстрира поведение, което се явява нещо подобно за различните стойности на полиномната степен d .
- В случая, когато полиномната степен $d = 2$, третия основен компонент на ядрения PCA изглежда, че взуна разсейването от добавката на Гаузионната шум v . Премахвайки дължимата вноска за този компонент, искаме да има ефект чрез някаква форма на намаляване на шума.

8.11 Обобщение и Дискусия

В тази глава ще представим материал, занимаващ се с теорията на основните компоненти на анализа и използването на невронните мрежи за прилагането му. Целесъобразно е, че сега рефлектираме над този материал и питаме: Колко полезен е анализа на основните компоненти? Отговорът на този въпрос, разбира се, зависи от прилагането на интереси.

Ако основната цел е само да се постигне добро *компресиране* на данни, като същевременно се запази, колкото се може повече информация за входа, използването на анализа на основните компоненти предлага полезна самоорганизация на учебната процедура. Тук ние отбеляваме от материала, че използването на подпространствения метод на разлагане, на базата на „първите l основните компоненти” на входните данни, осигурява линейно картографиране, което е оптимално в смисъл такъв, че той позволява реконструкция на оригиналните входни данни в смисъла на средната квадратна грешка. Освен това, репрезентация, базирана на първия l основен компонент, е за предпочитане пред произволно представено подпространство, защото основните компоненти на входните данни естествено подредени в низходяща собствена стойности, или еквивалентно в низходяща вариация. Съответно можем да оптимизираме използването на анализа на основните компоненти за компресиране на данни, чрез използването на най-голямото точно число, за да корогира първата и

основна съставка на входа и прогресивно по-малко прецизното, за да кодира останалите $l - 1$ компоненти.

Произтичаща грешка е представянето на данни, съставени от съвкупност от няколко кълстера. За да може кълстерите да са индивидуално видими, разделението между тях трябва да бъде по-голямо, отколкото на вътрешнот разсейване на кълстерите. Ако се случи така, че да има само няколко кълстера в набора от данни, тогава водещите основни оси, намерени като е използван основния компонентен анализ, ще се стремят да правят проекции на кълстерите с добро разделяне и по този начин, осигурявайки ефективна основна за извлечане на характеристики.

В този последен контекст ние говорим за полезното приложение на основния компонентен анализатор, като подпроцесори за надзор на невронната мрежа(например многослоен перцепtron, обушен с back-propagation алгоритъма). Мотивацията тук е да се ускори сближаването на учебния процес чрез де-корелиране на входните данни. Контролирана учебна процедура, каквато е back-propagation, разчита на стръмното спускане. Процеса на сближаване в тази форма на обучение обикновено е бавен, поради взаимодействияния ефект на многослойни перцепtronови синаптични тегла върху сигнала за грешка, дори с изпозването на просто местно ускоряване на процедурите, като итерация и адаптивно обучителния процент за индивидуални тегла. Ако обаче, входа на многослойния перцепtron се състои от несвързани компоненти, и след това от дискусията представена в Глава 4 заселязваме, че Хебиановата матрица на функцията на разходите $\mathcal{E}(n)$, с оглед на свободните параметри на мрежата е по-скоро диагонална, отколкото по друг другче представена. С тази форма на диагонализация на места, употребата на прости локални процедури за ускоряване, позволява значително забързване в процеса на сближаване, което е станало възможно чрез подходящото намаляване на размерите на учебния лихвен процент според всяка тегловна ос по отделно(Бекер,1991).

С Хебианово-базираният алгоритъм от тази глава, мотивиран от идеите, взети от невробиологията, подходящо е да се приключи нашата дискусия като коментираме ролята на анализа на основните компоненти в биологичните системи на възприятие. Линскер(1990а) задава въпроса за „достатъчността“ на анализа на основните компоненти като принцип за определяне на отговорността за собственост, разработена от един неврон за анализ на една общност от входни „сцени“. По-специално, оптималността на анализа на основните компоненти по отношение на точната реконструкция на един входен сигнал от невронов отговор се счита, че ще бъде със съмнително значение като цяло изглежда, че мозъкът прави много повече, отколкото просто да се опитва да възпроизвежда входни сцени, получени от неговите сетивни единици. Напротив, някои основни „смислени знаци“ или функции се извличат така, че да се позволява високо ниво на интерпретация на входовете. За това може да „изостри“ въпросът зададен в началото на тази дискусия и да попитаме: Колко полезен е анализът на основни компоненти за възприятийна обработка?

Амброс-Ингерсон и други (1990) посочват значението на алгоритмите, изложени от Оя(1982) и Сангър(1989а) за анализ на основните компоненти (т.е. Хебияново вдъхновените алгоритми, обсъдени в раздели 8.4 и 8.5) в йерархичната кълстеризация на алгоритми. Те изтъкват хипотезата, че йерархичната кълстеризация може да се появи като фундаментално свойство (поне частично) на спомени, базирани на дългосрочно потенциране(long-term potentiation - LTP) – подобно на синаптичната модификация на вида, открит в cortico-bulbar мрежите и схеми от подобен тип в други региони на мозъка, чието свойство може да се използва за разпознаване на ориентири от околната среда. Въпросът е, че самостоятелно организираният анализ на сновните компоненти може да бъде значителен в йерархичното групиране на научни ориентири в кората на главния мозък, не заради неговото оптимално реконструктивно свойство, а по-скоро заради присъщата му вътрешно силно свойство за подбиране на проекции на кълстери с добро разделяне.

Друга интересна роля на анализа на основните компоненти е , че обработката при възприемане се явява като подход при *форма-при-засенчване* проблема, представен от Атик и др.(1996). Проблемът може да се постави по следния начин: Как е способен мозъкът да възприема триизмерна форма от засенчените модели в двуизмерен образ? Атик и др. предлагат йерархично решение за *форма-при-засенчване* проблема, състоящ се от две понятия:

1. Мозъка, чрез еволюция или предишен опит е открил, че обектите могат да бъдат класифицирани в по-ниски измерения от обектни класове по отношение на тяхната форма. Това понятие в действителност се основава на факта, че ориентирите, които мозъкът използва, за да извлече триизмерната интерпретация са добре разбрани.
2. От понятие 1, извличането на формата от засенчените модели е намалена до много по-прост проблем на оценяване на параметри в пространство с по-ниско измерение.

Например, цялостната структура на човешката форма на главата е неизменно една и съща, в смисъл, че всички хора имат носове, представляващи издатини, очните кухини, представляващи вдълбнатини и челото, и бузите , представляващи по-хубави региони. Тази инвариантност предполага, че всяко лице изразено като $r(\theta, l)$ в цилиндрични координати, може да бъде описана като сума от два компонента:

$$r(\theta, l) = r_0(\theta, l) + \rho(\theta, l)$$

Където $r_0(\theta, l)$ означава „осреднена глава“ за определена категория хора (например, възрастни мъже или възрастни жени) и $\rho(\theta, l)$ означава смущения, които улавят самоличността на конкретно лице. Обикновено $\rho(\theta, l)$ е малко в сравнение с $r_0(\theta, l)$. За да представи $\rho(\theta, l)$, Атик и др. използват анализа на основни компоненти , където колебанията са представени по отношение на набора от собствени функции(т.е. двуизмерни контрагенти на собствени вектори). Резултатите, представени в Атик и др.(1996), демонстрират способността на дву-етапен йерархичен подход за

възстановяване на триизмерната повърхност за дадено лице от единичен дву-димензионален образ на това лице.

Бележки и референции

1. Анализа на главните компоненти(PCA) е може би най-старата и известна техника в многовариантния анализ (Jolliffe, 1986; Preisendorfer, 1988). За първи път бе представено от Pearson(1901),които го е използвал в биологичен контекст, за да се преработи линеен регресионен анализ в основна форма. Тогава е разработено от Hotelling (1993) в работа, разработвана по психометрия. Той се появява отново и съвсем независимо в създаването на теорията на вероятностите, както се счита от Кархунен(1947) и в последствие обобщено от Лоеве(1963).
2. Подходът, приет от Люнг(1977) и Кушнер и Кларк(1978) за изучаване на динамичното поведение на стохастичен приблизителен алгоритъм, намалява проблема за изучаване на динамиката на диференциално уравнение. Въпреки това, тези два подхода са коренно различни. Подходът на Люнг включва използването на функцията на Ляпинов, докато подходът възприет от Кушнер и Кларк предполага линеен интерполяционен процес и се поззовава на т.нар. Арзела-Асколи теорема(Дънфорд и Шварц, 1966). Подходът на Кушнер и Кларк е последван от Диамантарас и Кун(1996) за изучаването на конвергенцията на Хебиянов максимален собствен филтър. Заключениета, направени в него са същите като тези, получени с помощта на Люнг.
3. Фолдиак (Foldiak,1989) разширява невронната конфигурация на мрежата за анализ на основните компоненти, включвайки анти-Хебиянова feedback връзка. Мотивацията на тази модификация е получена от няколко по-ранни разработки на Барлоу и Фолдиак (Barlow and Foldiak (1989)) за адаптиране и де-корелация в зрителен контекст, там бе заявено, че ако невроните си взаимодействват според анти-Хебияновото правило, тогава изходите на невроните определят координатна система в която не съществуват корелации, дори когато постъпващите сигнали имат силни корелации.

Използването на страничното задържане сред изходните неврони също бе предложено от Рубнер и Таван(1989) и Рубнер и Шултен(1990) (Rubner and Tavan (1989) and Rubner and Schulten (1990)). Въпреки това, за разлика от модела, предложен от Фолдиак, латералната мрежа считана от Рубнер и др. не е симетрично свързана. Напротив, тя е йерархична, в това, че неврона i (да речем) потиска всички други неврони в мрежата, освен за $1, 2, \dots, i - 1$, където $i = 1, 2, \dots$.

Ш АРЕХ моделът, изучаван в Кунг и Диамантарас(1990) има същата мрежова топология, както на модела на Рубнер и др., но използва правилото за единичен

неврон на Оя(описано в Раздел 8.4) за коригиране на синаптичните тегла и в двете – feedforward и латералните връзки в модела.

4. Строго доказателство за конвергенцията на APEX алгоритъмът, с всички неврони с тенденция да се събират заедно, е дадена в Чен и Лиу(Chen and Liu(1992)).
5. За дискусия на няколко основни модела за анализ на основните компоненти и тяхното сравнение, вижте книгата на Диамантарас и Кун(1996).
6. Нелинейни PCA методи, с изключение на кернел PCA, могат да бъдат групирани в три класа (Диамантарас и Кун,1996):
 - Хебиянови мрежи, получени чрез замяна на линейни неврони в Хебияново базирани PCA алгоритми с нелинейни (Karhunen и Joutsensalo, 1995).
 - Репликаторни мрежи или автоматични енкодери, които са изградени около многослойни перцептрони: репликаторни мрежи, са разгледани в Глава 4.
 - Основни криви, които се основават на една повтаряща се оценка на крива или повърхностно улавяне на структурата на данните (Hastie and Stuetzle, 1989). В Ритер и др.(1992) и Черкаски и Мулиер (Cherkassky and Mulier (1995)), се посочва, че самоорганизационната карта на Кохонен може да се разглежда като изчислителна процедура за намиране на дискретното сближаване на основните криви; самоорганизиращите се карти са обсъдени в следващата Глава.

Проблеми

Хебияново-базиран максимален собствен филтър

8.1 за съответстващият филтър, разгледан в Пример 8.2, собствените стойности и свързаните с нея собствени вектори q_2 са дефинирани от

$$\lambda_1 = 1 + \sigma^2$$

$$q_1 = s$$

Показва, че тези параметри са задоволени основната връзка

$$R q_1 = \lambda_1 q_1$$

Където R е корелационната матрица на входния вектор X.

8.2 Предвид максималният собствен филтър, където тегловния вектор $w(n)$ се развива в съответствие с ур-е (8.46). Показано е, че вариацията на входния филтър доближава λ_{\max} , докато n се приближава до безкрайност, където λ_{\max} е най-голямата собствена стойност на корелационната матрица на входния вектор.

8.3 Минорен компонентен анализ (Minor components analysis - MCA) е противоположното на основния компонентен анализ. В MCA търсиме да откриеме тези указания, които минимизират проекционното разсейване. Така намерените посоки са собствените вектори, съответстващи на най-малките (минимални) собствени стойности на корелационната матрица R на входния вектор $X(n)$.

Чрез този проблем ние изследваме как се променя един неврон от Раздел 8.4, така, че да намериме минорния компонент на R . И по-специално, правим промяна в знака на обучаващото правило в Ур-е (8.40), получавайки (X_u и др., 1992)

$$w_i(n+1) = w_i(n) - \eta y(n)[x_i(n) - y(n) w_i(n)]$$

Покажете, че ако най-малката собствена стойност на корелационната матрица R е λ_m с множител 1, тогава

$$\lim_{n \rightarrow \infty} w(n) = \eta q_m$$

Където q_m е собствения вектор асоцииран с λ_m .

Хебияново-базиран основен компонентен анализ

8.4 Постройте сигнално-поточна графика, за да представя векторово-стойностни ур-я (8.87) и (8.88).

8.5 Обикновеният подход на диференциалното уравнение за конвергенционен анализ, описан в Раздел 8.4 не се прилага директно към обобщения Хебиянов алгоритъм за обучение (GHA). Въпреки това, чрез изразяване на синаптичната тегловна матрица $W(n)$ в ур-е (8.91) като вектор, съставен от отделните колони на $W(n)$, може да интерпретираме актуализационната функция $h(\cdot, \cdot)$ по обикновения начин, след това продължавайки с прилагането на асимптоматичната стабилностна теорема. Следователно, чрез това, което е казано тук, опознайте конвергенционната теорема за обобщения Хебиянов алгоритъм за обучение.

8.6 В този проблем изследваме ползата на обобщения Хебиянов алгоритъм за проучване на дву-димензионални възприемливи полета, произвеждани от произволен вход (Сангър, 1990). Случайният вход се състои от двуизмерна област на независим Гаузионен шум с нулево значение и единица вариация, което е усукано с Гаузионна маска (филтър) и след това умножено с Гаузионен „прозорец“. Маската Гаузиан е със стандартно отклонение от 2 пиксела, както и на Гаузионния прозорец

стандартното отклонение е 8 пиксела. Резултатният случаен вход $x(r, s)$ на позиция (r, s) може да бъде написан по следния начин:

$$x(r, s) = m(r, s)[g(r, s) * w(r, s)]$$

Където $w(r, s)$ е полето от независим и еднакво производствен Гаузионен шум, $g(r, s)$ в Гаузионната маска, $m(r, s)$ е Гаузионния функционален прозорец. Кръговата спираловидност на $g(r, s)$ и $w(r, s)$ се определя от

$$g(r, s) * w(r, s) = \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} g(p, q)w(r - p, s - q)$$

Където $g(r, s)$ и $w(r, s)$ е приет, че са периодични.

Използвайте 2000 проби на случаен вход $x(r, s)$, за да се обучи една единнослойна feedforward мрежа с помощта на обобщения Хебиянов алгоритъм. Мрежата има 4096 входа, подредени като 64×64 пикселова мрежа, със 16 изхода. В резултат, синаптичните тегла на тренировъчната мрежа е представена от 64×64 масив от числа. Извършете изчисленията, описани в настоящия документ и изведете 16 масиви на синаптичните тегла като двудимензионални маски. Коментирайте резултата.

8.7 Ур-е (8.113) определя трансформираната версия на актуализационното ур-е (8.106) за изчисляване на обратната връзка на тегловния вектор $a_j(n)$. Трансформацията се основава на определянето на синаптичния тегловен вектор $w_j(n)$ по отношение на т м основни режими на мрежата в ур-е (8.109). Изведете ур-е (8.113).

8.8 Да разгледаме системната матрица на ур-е (8.116), представена от сигнално-поточната графика на Фиг.8.12, която съответства на $1 \leq k \leq j - 1$.

- a) Формулирайте характерното ур-е на тази 2×2 матрица
- б) Покажете, че матрицата има двойна собствена стойност.
- в) Оправдайте твърдението, че всички основни режими на мрежата една и съща собствена стойност.

8.9 GHA използва само forward връзки, докато APEX алгоритъма използва и forward и латерални връзки. Но въпреки различията, дългосрочното конвергенционно поведение на алгоритъма APEX е на теория същото като GHA. Оправдайте валидността на това твърдение.

Ядрено PCA

8.10 Нека \bar{K}_{ij} обозначи централния „колега“ на ij -тия елемент на K_{ij} на кернел матрицата K . Покажете, че (Шъолкопф, 1997)

$$\begin{aligned}\bar{K}_{ij} = K_{ij} - \frac{1}{N} \sum_{m=1}^N \boldsymbol{\varphi}^T(\mathbf{x}_m) \boldsymbol{\varphi}(\mathbf{x}_j) - \frac{1}{N} \sum_{n=1}^N \boldsymbol{\varphi}^T(\mathbf{x}_i) \boldsymbol{\varphi}(\mathbf{x}_n) \\ + \frac{1}{N^2} \sum_{m=1}^N \sum_{n=1}^N \boldsymbol{\varphi}^T(\mathbf{x}_m) \boldsymbol{\varphi}(\mathbf{x}_n)\end{aligned}$$

Предложете компактна репрезентация на тази релация в матрична форма.

8.11 Покажете, че нормализацията на собствения вектор α на кернел матрицата K е еквивалентна на изискването, така че ур-е (8.153) да бъде удовлетворено.

8.12 Обощете свойствата на ядрения PCA.