

**Лекция 14:**  
**Невронни мрежи (продължение)**

**САМООБУЧЕНИЕ НА НЕВРОННИ МРЕЖИ**

При обучението без учител (т.е. при самообучението) се предполага, че не съществува обратна връзка със средата, в която работи мрежата, т.е. няма кой да определи какъв би трябвало да бъде изходът на мрежата и дали формираният от нея изход е коректен. Мрежата трябва сама да открие образци, характеристики, регулярности, корелации или категории във входните данни и да ги кодира на изхода. Затова изкуствените неврони и връзките между тях трябва да притежават (да демонстрират) някаква степен на **самоорганизация**.

В резултат на самообучението може да се получи нещо полезно само когато съществува **изобилие от входни данни**. Без изобилие не може да се намерят никакви характеристики на данните, тъй като в тях има и случайни шумове. В този случай **изобилието дава знание**.

Типът на образаца, който самообучаващата се мрежа открива във входните данни, зависи от **архитектурата ѝ**. Обикновено самообучаващите се мрежи имат проста архитектура – най-често те са прави и включват един входен и един изходен слой. При това обикновено изходите са много по-малко от входовете.

Някои възможни резултати от работата на самообучаваща се невронна мрежа:

**Кластеризация.** Множество от двоични изходи, само един от които е активен в определен момент от времето, могат да определят към коя от няколко категории принадлежи входният образец. Всеки кластер от подобни или близки образци ще бъде класифициран като един изходен клас.

**Кодиране.** Изходът може да бъде кодирана версия на входа в по-малко битове, поддържайки толкова необходима информация, колкото е възможно. Това се използва за компресиране на данни преди подаването им по канал с ограничен обхват, предполагайки, че може да се конструира и обратна декодираща мрежа.

В повечето случаи архитектурите и обучаващите правила се основават на интуитивни предположения. В някои случаи се използват и оптимизационни подходи (цели се максимална икономичност на представянето или максимизиране на някакво количество, например на съдържанието на информацията или минимизиране на изменението на изхода).

Един от най-често използваните методи (подходи) за самообучение е т. нар. **състезателно обучение**. При него **само един изходен неврон или само един неврон от определена група е активен**, т.е. има ненулева активност (активационно ниво, стойност) в даден момент от времето. Изходните неврони се състезават да бъдат активни и затова често се наричат **“печелившият-взема-всичко”** неврони или **“изпреварващи-всички”** елементи (възли).

Целта и тук е да се кластеризират или категоризират входните данни. Подобни входове би трябвало да бъдат класифицирани в една и съща категория и затова би трябвало да активират един и същ изходен неврон. Класовете (кластерите) трябва да бъдат открити от самата мрежа на основата на корелации между входните данни.

### Стандартно състезателно обучение

Мрежите, които използват стандартното състезателно обучение, имат **един входен и един изходен слой**. Всеки от изходните възли  $O_i$  е свързан с всеки от входните възли  $\xi_j$  чрез възбуждаща връзка с тегло  $w_{ij} \geq 0$ . Ще разглеждаме само мрежи с **бинарен вход и изход** (с входни и изходни стойности от множеството  $\{0,1\}$ ). Само един от изходните неврони, наречен **победител**, може да бъде активен (да има ненулева стойност) в даден момент. Обикновено това е възелът с най-голяма стойност (активност)

$$h_i = \sum_j w_{ij} \xi_j = \vec{w}_i \cdot \vec{\xi} \quad \text{за дадения входен вектор } \vec{\xi}.$$

Затова неравенството

$$(1) \vec{w}_{i^*} \cdot \vec{\xi} \geq \vec{w}_i \cdot \vec{\xi} \quad \text{за всяко } i$$

дефинира побеждаващия неврон с  $O_{i^*} = 1$ .

Като правило се изисква теглата за всеки изходен възел да са нормирани:

$$|\vec{w}_i| = 1 \quad \text{за всяко } i.$$

В такъв случай за даден входен вектор  $\vec{\xi}$  побеждаващият изходен неврон  $i^*$  с  $O_{i^*} = 1$  се дефинира посредством неравенството

$$(2) |\vec{w}_{i^*} \cdot \vec{\xi}| \leq |\vec{w}_i \cdot \vec{\xi}| \quad \text{за всяко } i.$$

Една мрежа от тип “печелившият-взема-всичко” реализира класификатор на образци, като използва критерия (1) или (2). **Задачата за обучението ѝ** е свързана с **намиране на тегловите вектори  $\vec{w}_i$  при изискването мрежата да намира по подходящ начин кластери във входните данни**.

Няма принципно значение начинът, по който се реализира мрежа от тип “печелившият-взема-всичко”. При компютърна симулация винаги може да се осъществи търсене на максималното  $h_i$ . Често срещана (и по-естествена в определен смисъл) е и ситуацията, при която изходните неврони се състезават помежду си с цел излъчване на победител чрез **странично подтискане**: всеки неврон подтиска другите чрез връзки с отрицателни тегла и евентуално се самовъзбужда чрез връзка с положително тегло. За целта страничните тегла и нелинейната активационна функция трябва да бъдат подбрани коректно – така, че да е сигурно, че ще бъде избран точно един изход и колебанията ще бъдат избегнати.

В такъв случай най-общо алгоритъмът на стандартното състезателно обучение изглежда по следния начин:

1. Присвояват се малки случайни стойности на теглата. Важно в този момент е да няма симетрия (т.е. теглата да са напълно различни).
2. Избира се входен вектор от обучаващото множество.
3. Пресмята се началната стойност (активност, активационно ниво) за всеки от изходните неврони.
4. Изходните неврони се състезават, докато само един от тях остане активен.
5. Увеличават се теглата на връзките между активния изходен неврон и активните входни неврони и се намаляват теглата на връзките между активния изходен неврон и неактивните входни неврони така, че векторът от тези тегла да остане нормиран.
6. Стъпки 2 – 5 се повтарят за всички входни вектори за много епохи.

## ГЕНЕТИЧНО ПРОГРАМИРАНЕ

**Идеите** му се основават на принципите на *еволюцията* при живите организми: създаване на поколения от по-добри индивиди посредством *промяна (развитие)* чрез *възпроизводство* и *избирателно оцеляване на част от наследниците (подбор)*.

При генетичното програмиране по такъв начин се създават (“отглеждат”) програми.

- *Поколение 0* представлява популация от случайни програми, използващи функции, константи и входни данни, за които се смята, че ще бъдат полезни.
- *Поколение  $i+1$  ( $i>0$ )* включва:
  - някои от най-добрите представители на поколение  $i$ , избрани чрез състезание (турнир), върху малка част (около 1%) от които може да се приложи *мутация*, като случайно избрана част се замени със случайно породена конструкция (по такъв начин се осигурява възможност да се достигне всяка точка от пространството на търсене и да се излезе от евентуален локален минимум);
  - наследници на някои от най-добрите представители на поколение  $i$ , получени чрез *кръстосване*, при което случайно избрана част на “бащата” замества случайно избрана част на “майката”.

Този подход може да се приложи и върху невронни мрежи, като тяхното обучение се редува с промяна чрез възпроизводство и подбор, а при кръстосването изходните възли на родителите стават скрити възли на наследниците.