۲) از استنتاج بیزی (دقیق یا تقریبی) برای محاسبه ی احتمال های قبلی همه ی والدهای گره سود استفاده نمایید .

۳) سود را براساس توزیع احتمال روی مقادیر والدها محاسبه نمایید .

۳. بردار تصمیم گیری دارای بیش ترین سود را برگردان

تصمیم گیری روبوت

۱. مقادیر شناخته شده را به متغیّرهای مدرک نسبت دهید :

باتری سنج ، سرعت سنج و GPS [۱] (همه در زمان t)

۲. برای هر ترکیب ممکن از توان و زاویه ی چرخش [۲] :

۱) از استنتاج بیزی برای محاسبه ی توزیع احتمال برای سطح باتری و وضعیت در زمان t+1 استفاده نمایید .

۲) سپس آن توزیع ها را برای محاسبه ی سود به کار ببرید .

۳. بهترین توان را به دست آورید .

نکته : از سود مقادیر *آینده ی طرح* [۱] سطح باتری و وضعیّت به صورت پایه ای برای انتخاب عملکردهای جاری ، استفاده نمایید .

---

[۱] مخفّف Global Positioning System است و سیستمی ناوبری [مکان نمایی] است که با شبکه ای گسترده از ماهواره ها کار می کند .

[۲] turn-angle

ارزش اطّلاعات

– **ایده :** مقدار به دست آمده برای هر جزء ممکن مدرک را محاسبه نمایید . این کار به طور مستقیم از شبکه ی تصمیم گیری می تواند انجام شود .

مثال : تمرین خرید صحیح روغن . از دو قوطی A و B ، دقیقاً یکی محتوی روغن می باشد ، قیمت را k در نظر بگیرید . احتمال هر کدام را هم ۰.۵ در نظر بگیرید . قیمت فعلی هر قوطی k/2 می باشد . " متخصص " پیشنهاد بررسی A را می دهد . آیا A مناسب است ؟

راه حل : ارزش مورد انتظار اطّلاعات را محاسبه نمایید ، که برابر است با ، مقدار مورد انتظار بهترین عملکرد اطّلاعات ارایه شده منهای مقدار مورد انتظار بهترین عملکرد بدون اطّلاعات . عملکردها خرید A (Buy(A)) و خرید B (Buy(B)) می باشند . بدون اطّلاعات ، ارزش مورد انتظار هر عملکرد برابر است با :

بررسی ممکن است بگوید که " روغن در A است " یا " روغن در A نمی باشد " و احتمال ۰.۵ برای هر کدام داده شده است ! با داشتن اطّلاعات ، مقدار مورد انتظار برای بهترین عملکرد ، برابر است با :

[۰.۵ * ارزش داده شده ی " خرید A " و با فرض این که " روغن درون A می باشد " + ۰.۵ * ارزش " خرید B " با فرض این که " روغن درون A نمی باشد "] که برابر است با :

$=(0.5*k/2)+(0.5*k/2)-0=k/2$

بنابراین ارزش اطّلاعات مورد انتظار برابر است با :

---

[1] PROJECTED FUTURE

انتخاب اندازه گیری [۱]

اطّلاعات معمولاً مستقل نمی باشند . به دست آوردن اندازه گیری ها می تواند پرهزینه باشد : آزمایش های طبی ، تحلیل های شیمیایی اعماق دریا و ... . احتیاج به اولویّت بندی اندازه ها یا آزمایش ها داریم . برای این کار ، آن هایی که منجر به وضعیّت های با بالاترین سود مورد انتظار می شوند را انتخاب نمایید .

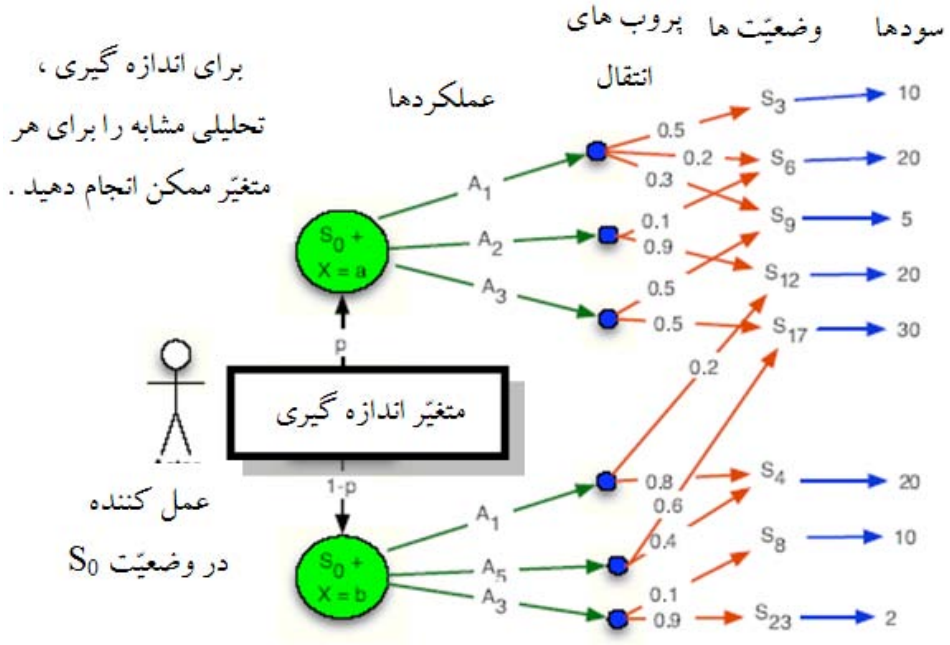
اندازه گیری ها ← اطّلاعات وضعیّت ها ← انتخاب عملکردها ← وضعیّت های کلّی جدید ← ارزشیابی سود [۲]

از آنجایی که تعدادی از این اتّصالات به صورت اتّفاقی می باشند ، ما به توزیع احتمالی که هم نتایج اندازه گیری و هم عملکردهای منظّم را دارا هستند رسیدگی می نماییم .

---

[۱] Measurement Selection

[۲] Utility Assessment



فرمول کل – اگر برای مدرک جاری E ، بهترین عملکرد جاری ، α باشد . نتیجه ها ی عملکرد ممکن S_i و توان بالقوه ی (پتانسیل) مدرک جدید E_j برابر است با :

$$EU(\alpha \mid E) = \max_a \sum_i U(S_i) P(S_i \mid E, a)$$

تصور نمایید که ما می دانیم $E_j = e_{jk}$ ، بنابراین ما $\alpha_{e_{jk}}$ را انتخاب می کنیم .

$$EU(\alpha_{e_{jk}} \mid E, E_j = e_{jk}) = \max_a \sum_i U(S_i) P(S_i \mid E, a, E_j = e_{jk})$$

E_j ، یک متغیّر تصادفی است که مقدار فعلی آن ناشناخته می باشد ، در نتیجه ، باید مقادیر آن را حول تمام مقادیر ممکن آن محاسبه نماییم :

$$VPI_E(E_j) = (\sum_k P(E_j = e_{jk} \mid E) EU(\alpha_{e_{jk}} \mid E, E_j = e_{jk})) - EU(\alpha \mid E)$$

VPI مخفف Value of Perfect Information می باشد ، به معنی ارزش اطّلاعات کامل می باشد .

خصوصیّات ارزش اطّلاعات کامل

- **نا منفی** [۱] می باشد : $\forall j, E : VPI_E(E_j) \geq 0$

- **جمع ناپذیری** [۲] ، در مثال، توجه کنید که E_j دوبار آمده است :

$$VPI_E(E_j, E_k) \neq VPI_E(E_j) + VPI_E(E_k)$$

مستقل از نظم بودن [۳] :

$$VPI_E(E_j, E_k) = VPI_E(E_j) + VPI_{E,E_j}(E_k) = VPI_E(E_k) + VPI_{E,E_k}(E_j)$$

توجّه : در زمانی که بیش تر از یک بخش مدرک می تواند جمع آوری شود ، بیشینه کردن ارزش اطّلاعات کامل برای هر قسمت ، برای انتخاب یکی از آن ها که همیشه هم بهینه نمی باشد منجر به این می شود که جمع کردن مدرک ها تبدیل به یک مسأله ی تصمیم گیری *ترتیبی* شود .

کیفیّت (چگونگی) رفتارها

اگر انتخاب ، قابل مشاهده می باشد ، اطّلاعات دارای ارزش کمی می باشند . اگر انتخاب ، غیرقابل مشاهده می باشد ، اطّلاعات ، به دلیل بالا بودن گوناگونی سودمندی (U) ، دارای ارزش زیادی می باشند .

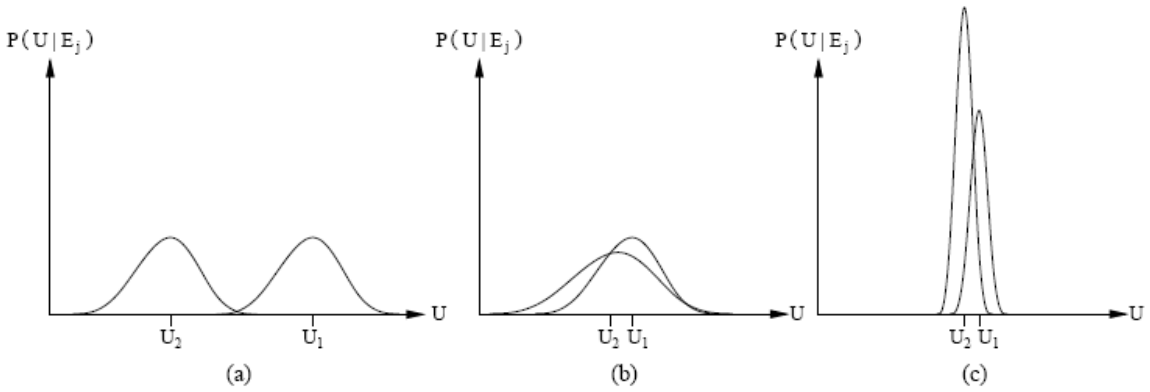
---

[۱] nonnegative

[۲] nonadditive

[۳] order-independent

اگر انتخاب غیرقابل مشاهده می باشد ، اطّلاعات ، به دلیل پایین بودن گوناگونی سودمندی (U) ، دارای ارزش کمی می باشند .



مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

فصل هیجدهم

شبکه های عصبی[1]

[1] neural networks

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

ریوس مطالب

- مغز [۱]
- شبکه های عصبی
- پرسپترون [۲] ها
- پرسپترون های چند سطحی
- کاربردهای شبکه های عصبی

شبکه های عصبی

---

[۱] brains

[۲] Perceptron

هوش مصنوعی

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

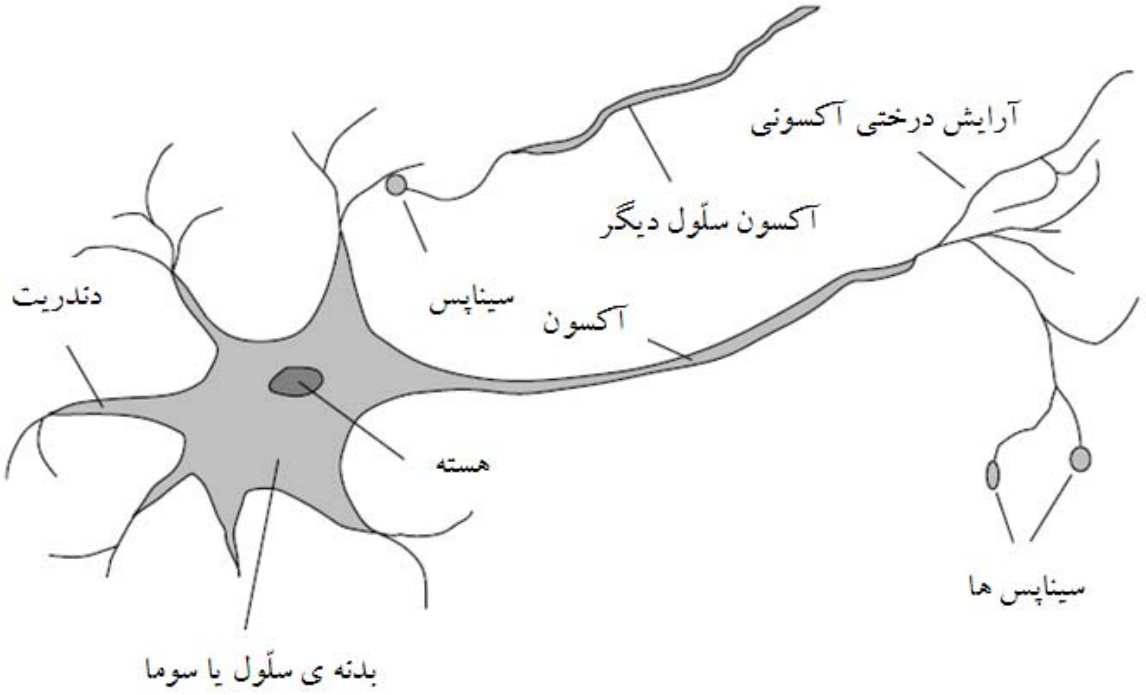
بسیاری از آنچه که تا کنون ما مطالعه کرده ایم می تواند به صورت *هوش مصنوعی نمادین یا نشانه ای* [۱] طبقه بندی شود ؛ در هوش مصنوعی نمادین ، تأکید بر سمبل ها و ارتباطات میان آن ها می باشد . به عنوان مثال ، جستجو ، منطق ، درخت های تصمیم گیری و برنامه ریزی جزء هوش مصنوعی نمادین می باشند . سمبل ها برای رفتار هوشمند ، کلیدی می باشند . شبکه های عصبی بر رفتار *نمادین* تمرکز می نماید . توجّه کنید که رفتار هوشمند از ارتباط اجزای ساده به وجود می آید .

زیست شناسی و علم کامپیوتر

در نورون های [۲] زیست شناسی ، سیگنال های دریافت شده توسط دندریت ها [۳] دریافت می شوند و از طریق آکسون [۴] به دیگر نورون ها می رسند . [در انتقال سیگنال ها] ، علامت دهی [۵] و شلیک [۶] ، خیلی پیچیده می باشند . فکر و رفتار در تعامل هزاران نورون به وجود می آیند . محقّقان هوش مصنوعی اغلب جذب توسعه ی الگوریتم های مؤثّر می باشند ؛ مانند الگوریتم های ژنتیکی ، ما برروی ایده هایی که کاملاً طبیعی هستند کار می کنیم و آن هایی که مفید هستند را به کار می بندیم .

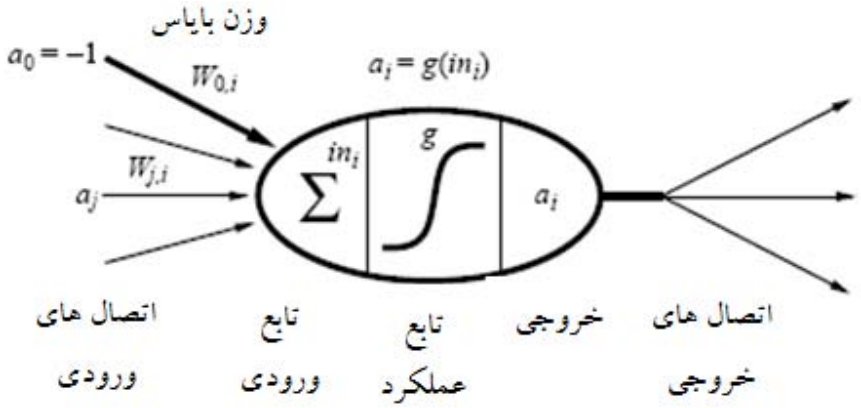
---

۱ symbolic AI
۲ neurons
۳ dendrites
۴ axon
۵ signaling
۶ firing



شبکه های عصبی محاسباتی

شبکه های عصبی از گره ها تشکیل شده اند . این گره ها توسط اتّصال هایی جدا از آکسون ها به هم متّصل می باشند . هر اتصال دارای یک فشار است که قدرت سیگنال (علامت) را مشخص می کند . هر گره دارای یک تابع عملکرد غیرخطی است . خروجی گره به صورت یک تابع توزیع شده ، جمع ورودی ها را مدیریّت می نماید .



توابع عملکرد

در حالت کلّی ، از هر تابع غیرخطی می توان استفاده نمود ؛ عمومی ترین توابع ، دو تابع به صورت زیر می باشند :

۱- **تابع مرحله ای (تابع آستانه)** – خروجی ها ، اگر ورودی مثبت باشد ، یک می باشد و در غیر این صورت صفر می باشد .

۲- **تابع S شکل (سیگموید)** [۱] **یا نقلی** [۲] **:** $1/(1+e^{-x})$ ؛ که به صورت مشتق پذیر و پیوسته [۳] می باشد و دارای تغییر سریع در نزدیکی آستانه می باشد و به صورت تدریجی به سمت بی نهایت می رود .

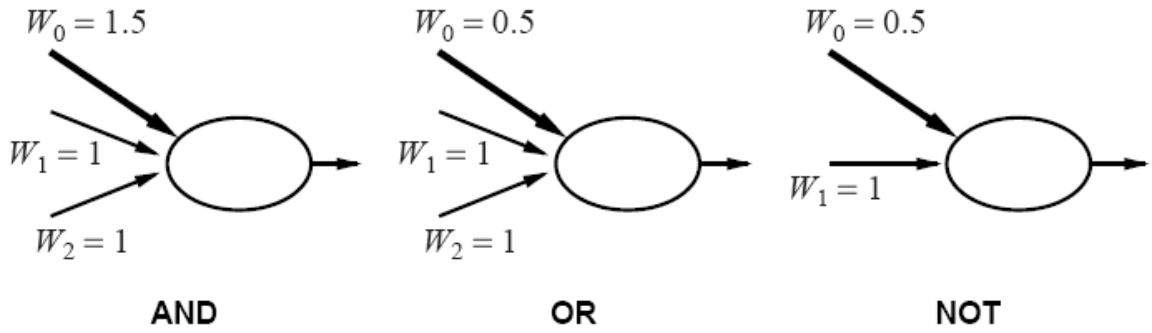
مثال ها :

---

[۱] sigmoid

[۲] logistic

[۳] continuously differentiable



شبکه های عصبی می توانند به سادگی برای انجام برخی از عملیّات استاندارد منطقی با استفاده از تابع عملکرد آستانه ای ساخته شوند ؛ که در این مورد ، باید شما بسته به تابعی که می خواهید ، آستانه را تغییر دهید .

انواع گره ها

ما می توانیم سه نوع گره را تشخیص دهیم :

۱- گره های ورودی
۲- گره های خروجی
۳- گره های پنهان

ما همچنین می توانیم انواع شبکه های زیر را داشته باشیم :

۱- **شبکه های با تغذیه ی مستقیم** [۱] **:** علامت (سیگنال) ها در یک جهت حرکت می کنند و بدون دور یا سیکل می باشند . شبکه های تغذیه ی مستقیم ، توابع را تکمیل می کنند و وضعیت داخلی ندارند .
۲- **شبکه های بازگشت کننده** [۱] **:** در انتشار علامت (سیگنال) ، دور یا سیکل وجود دارد .

---

[۱] feed-forward networks

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

ما در ابتدا روی شبکه های تغذیه ی مستقیم تمرکز می نماییم .

شبکه های تغذیّه ی مستقیم به شکل تخمین زننده ی توابع

شبکه های عصبی با تغذیه ی مستقیم در یک خانواده از الگوریتم ها به نام *تخمین زننده های تابع غیرخطّی* [۲] قرار می گیرند ؛ خروجی یک شبکه ی عصبی تابعی از ورودی هایش می باشد . تابع فعّال سازی غیرخطّی ، ارایه ی توابع پیچیده را اجازه می دهد . با تطبیق دادن وزن ها ، ما تابعی که ارایه می شود را تغییر می دهیم . شبکه های عصبی اغلب برای تخمین توابع پیچیده از داده ها استفاده می شوند .

طبقه بندی با شبکه های عصبی

شبکه های عصبی ، طبقه بندی را هم خیلی خوب انجام می دهند ؛ ورودی ها را به یک یا بیش تر خروجی تبدیل می نمایند و دامنه ی خروجی به کلاس هایی مجزّا تقسیم می شوند و برای کارهای آموزشی در جایی که نمی دانیم " در جستجوی چه هستیم " ، خیلی مفید می باشد ، مثل : صورت شناسی [۳] ، دست خط شناسی [۴] و رانندگی یک اتومبیل .

مثال تغذیه ی مستقیم

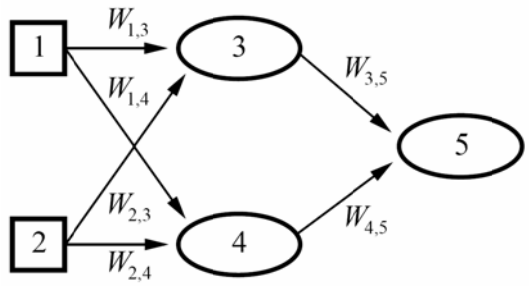
---

[۱] recurrent networks

[۲] nonlinear function approximators

[۳] face recognition

[۴] handwriting recognition



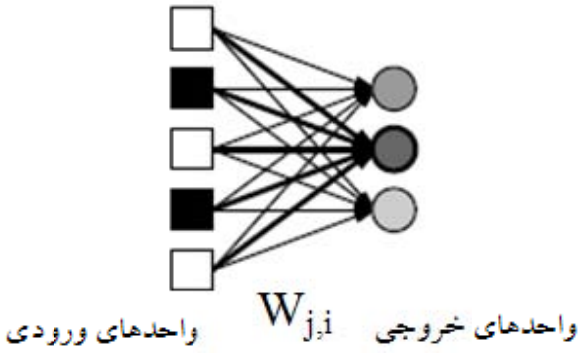
شبکه ی تغذیه ی مستقیم ، یک خانواده ی پارامتربندی شده از توابع غیرخطی است ، داریم :

$$a_5 = g(W_{3,5}.a_3 + W_{4,5}.a_4)$$
$$= g(W_{3,5}.g(W_{1,3}.a_1 + W_{2,3}.a_2) + W_{4,5}.g(W_{1,4}.a_1 + W_{2,4}.a_2))$$

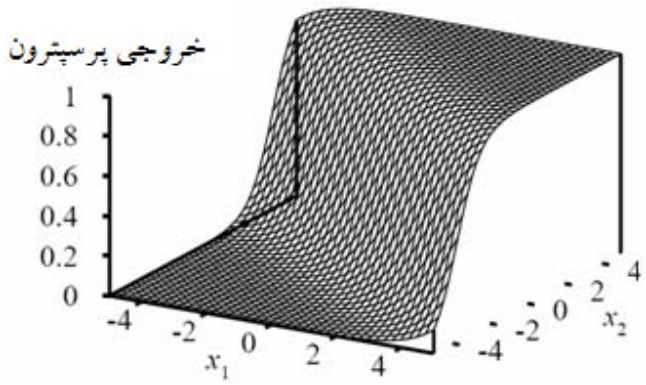
پرسپترون ها

پرسپترون ، شبکه ای از نورون هاست که در آن یک ارتباط دارای وزن میان خروجی برخی از نـرون ها و ورودی دیگر نرون ها وجود دارد ؛ به عنوان تعریفی دیگر ، ابزاری است که می تواند الگوهای یـادگیری (آموزشی) را تشخیص دهد . [1]

پرسپترون های تک لایه ای



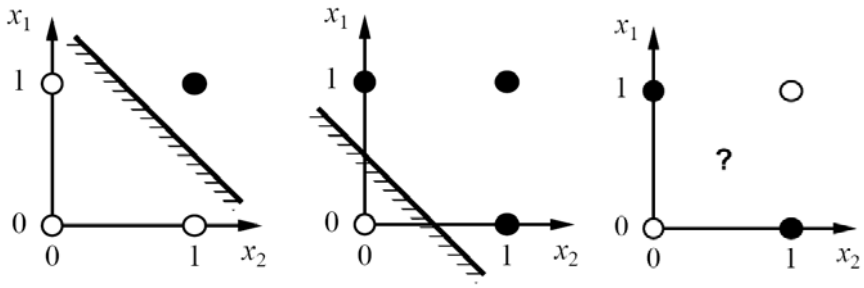
[1] Babylon / English - English



پرسپترون های تک لایه ای ، ساده ترین شکل شبکه ی عصبی می باشند ؛ برای فهم ، ساده می باشند ولی برای محاسبه ، محدود می باشند و در آن ها هر واحد ورودی به صورت مستقیم به یک یا بیش تر واحد خروجی متّصل شده است . اگر $w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n > 0$ باشد ، آن گاه $o(x_1,...,x_n) = 1$. همه ی واحدهای خروجی به طور مستقل عمل می نمایند – نه به صورت اشتراکی .

توجّه نمایید که یک پرسپترون با تابع مرحله ای g ، می تواند AND ، OR ، NOT و ... را ارایه نماید ولی XOR را نمی تواند بیان نماید و در فضای ورودی ، یک مجزّا کننده ی خطی [1] را ارایه می نماید :

$$W.x>0 \text{ یا } \sum_j W_j x_j > 0$$



(a) x_1 **and** x_2 (b) x_1 **or** x_2 (c) x_1 **xor** x_2

[1] linear separator

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

توان نمایش پرسپترون ها

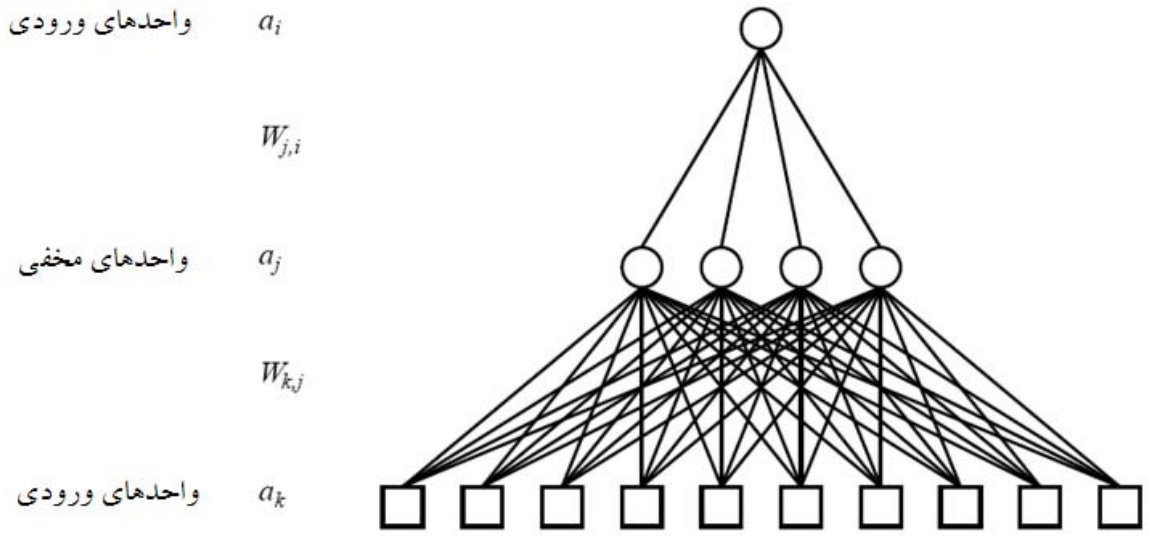
خروجی شبکه ، $\sum_{j=0}^{n} W_j i_j > 0$ می باشد . پرسپترون ها قادر به ارایه ی هر تابع خطی جداشدنی هستند ، ولی متأسّفانه ، تعدادی از توابع مثل XOR به دلیل این که به صورت خطّی جدا شدنی نمی باشند قابل نمایش توسّط پرسپترون ها نمی باشند . در روزهای ابتدایی هوش مصنوعی ، پرسپترون ها ساده بودند ، به علّت این واقعیّت که وزن آن ها می توانست به سادگی پیدا شود .

آموزش پرسپترون

اگر سیگنال خروجی ، خیلی بزرگ باشد ، اوزان باید کاهش یابند . برای این کار ، وزن هایی را که در خروجی شرکت دارند را کاهش دهید ، توجّه کنید که وزن های با ورودی صفر ، مؤثّر نمی باشند . اگر خروجی خیلی کوچک باشد ، اوزان باید افزایش یابند . برای این کار ، وزن هایی را که در خروجی شرکت دارند را افزایش دهید ، در این مورد هم توجّه کنید که وزن های با ورودی صفر ، مؤثّر نمی باشند . آموزش ، در واقع بهینه سازی وزن ها می باشد و به طور متناوب ما در حال انجام یک جستجوی تپّه نوردی در میان فضای وزن هستیم .

پرسپترون های چند لایه ای

در پرسپترون های چند لایه ای ، لایه ها معمولاً به صورت کامل متّصل می شوند و به طور معمول ، تعدادی از واحدهای مخفی به صورت دستی انتخاب می شوند . پرسپترون ها دارای این مزیّت هستند که یک الگوریتم آموزشی ساده دارند ولی عیب آن ها در این است که دارای محدودیّت های محاسباتی هستند . حال سؤال این است که اگر ما یک لایه ی مخفی دیگر اضافه نماییم چه اتفاقی می افتد ؟ پاسخ این است که *توان محاسباتی افزایش می یابد* ؛ با یک لایه ی پنهان ، می تواند هر تابع پیوسته را نمایش دهد و با دو لایه ی پنهان ، می تواند هر تابعی را نمایش دهد . در این مورد ، مشکل این است که چگونه وزن های صحیح را برای گره های پنهان ، پیدا نماییم ؟



انتشار معکوس [۱]

انتشار معکوس ، شاخه ای از الگوریتم آموزشی پرسپترون برای رسیدگی کردن به چندلایه ی گره هاست . گره ها از تابع فعال سازی سیگموید استفاده می نمایند ، داریم :

$$g(input_i) = \frac{1}{1 + e^{-input_i}}$$

$$g'(input_i) = g(input_i)(1 - input_i))$$

یادداشت :

- $h_w(x)$ - بردار خروجی های شبکه
- y – خروجی مطلوب برای یک مثال آموزشی

[۱] backpropagation

- a_j - خروجی j امین واحد مخفی
- o_i - خروجی i امین واحد خروجی
- t_i - خروجی برای i امین مثال آموزشی
- خطای خروجی برای گره ی خروجی i :

$$\Delta_i = (t_i - o_i) * g(input_i) * (1 - g(input_i))$$

- به روز رسانی وزن (خروجی پنهان) :

$$W_{j,i} = W_{j,i} + \alpha * a_j * \Delta_i$$

به روز رسانی وزن های با ورودی پنهان :

ایده : هر گره ی پنهان دارای یک کسر از خطا به صورت Δ_i می باشد . Δ_i را با توجّه به شدّت اتّصال میان گره پنهان و خروجی تقسیم نمایید : $Delta_j = g(input)(1-g(input))\sum_i W_{j,i}\Delta_i$ ، قانون را برای وزن های با ورودی پنهان به روز نمایید : $W_{k,j} = W_{k,j} + \alpha * input_k * \Delta_j$

الگوریتم انتشار معکوس

مادامی که انجام نشده است کارهای زیر را انجام بده :

برای d در مجموعه ی آموزشی

ورودی های d و انتشار مستقیم را به کار ببر .

برای گره ی i در لایه ورودی

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

$$Delta_i = output*(1-output)*(t_{\exp}-output)$$

برای هر گره ی پنهان

$$Delta_j = output*(1-output)*\sum W_{k,i}\Delta_i$$

برای هر وزن

$$W_{j,i} = W_{j,i} + \alpha * \Delta_i * input_j$$

متوقّف کردن شرط ها

سؤال : چه موقع ، آموزش را متوقّف نماییم ؟

جواب : وقتی که تعداد تکرارها ثابت شده باشد ، مجموع خطا ، زیر یک مجموعه ی آستانه باشد و همگرایی [۱] وجود داشته باشد ؛ یعنی ، هیچ تغییری در وزن ها به وجود نیاید .

صورت شناسی با استفاده از شبکه های عصبی

یکی از مزایای شبکه های عصبی این است که آن ها به یک توضیح صریح از ارتباط های میانِ بَعدی ها نیاز ندارند . آن ها واقعاً توضیحات را یاد نمی گیرند ، برای مسایلی که یک جواب نمادین ندارند به خوبی مناسب می باشند . شما می توانید کد شبکه ی عصبی موجود را برای تشخیص صورت ها بازنگری نمایید .

نکاتی در مورد انتشار معکوس

[۱] convergence

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

همیشه برای چند لایه ی پنهان کار می کنند . انتشار معکوس فقط برای گرایش دادن (همگرایی) به یک کمینه ی محلی ضمانت شده است و ممکن است مجموعه ی وزن های کامل را پیدا نکنیم . وزن های اولیه ی پایین می توانند در کارکردن شبکه به صورت به مراتب خطّی تر به ما کمک نمایند ، همچنین می توانند از شروع مجدّد تصادفی استفاده نمایند .

شبکه های با عملکرد شعاعی [۱]

یک مسأله ی انتشار معکوس این است که هر گره با خروجی یک راه حل همکاری می نماید ؛ این به این معنی است که همه ی وزن ها باید برای کمینه کردن خطای عمومی میزان شوند . اغتشاش در یک بخش از داده ها می تواند سبب یک ضربه در همه ی خروجی شبکه شود . همچنین رشته های زمانی طولانی می باشند . شبکه های با عملکرد شعاعی ، یک راه حل را برای این مورد بیان می نمایند .

بینش : هر گره در شبکه یک جزء از فضای ورودی را نمایش می دهد و دنبال طبقه بندی نمونه هایی که در اطراف آن رخ می دهند می باشد .

نگرش وانیلی [۲] : برای هر نقطه ی آموزشی $< x_i , f(x_i) >$ ، گره ای را بسازید که در وسط ، x_i قرار دارد . خروجی این گره برای یک ورودی جدید x ، $W * \phi(| x - x_i |)$ خواهد بود . که W ، وزن می باشد و ϕ که تابع اساسی است برابر است با $\phi = \exp(-\frac{x^2}{2\sigma^2})$.

هر گره دارای یک " ناحیّه ی تأثیر " [۳] است که می تواند نمونه های نزدیک را طبقه بندی نماید . آموزش با طبقه بندی غلط ، فقط بر روی گره هایی که نزدیک نمونه هایی که به صورت نادرست طبقه بندی شده اند تأثیر خواهد گذاشت . همچنین ، این شبکه ، تک لایه ای می باشد ؛ وزن ها می توانند با یک معادله ی

---

[۱] Radial Basis Function networks

[۲] vanilla approach

[۳] zone of influence

ماتریسی به دست آیند ، داریم : $\Phi W = t$ ، در نتیجه ، $W = \Phi_t^{-1}$. توجّه کنید که معکوس کردن یک ماتریس ، به مراتب سریع تر از ساختن آن با انتشار معکوس می باشد .

شبکه های عصبی برگشت کننده [۱]

تا کنون ما فقط در مورد شبکه های با تغذیّه ی مستقیم صحبت کرده ایم . در این شبکه ها ، علامت ها (سیگنال ها) در یک جهت منتشر می شوند ، خروجی بلافاصله در دسترس می باشد و دارای الگوریتم هایی هستند که به سادگی قابل فهم می باشند . تعداد زیادی کار می تواند با شبکه های عصبی برگشت کننده انجام شود . لااقل برخی از خروجی ها به پشت ورودی ها متّصل شده اند . در زیر یک شبکه عصبی برگشت کننده ی تک لایه ای را می بینید :

[۱] recurrent NNs(Neural Networks)

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

شبکه های هوپفیلد [۱]

یک شبکه ی هوپفیلد دارای هیچ گره معین ورودی یا خروجی نمی باشد ؛ هر گره یک ورودی و روال های یک خروجی را دریافت می نماید ؛ هر گره به گره های دیگر متّصل شده است و معمولاً ، از توابع آستانه ای استفاده می شود ؛ شبکه بلافاصله یک خروجی را تولید نمی نماید و مردّد می باشد ؛ تحت برخی از وضعیّت های به آسانی قابل دسترس ، سرانجام به حالت موازنه می رسد ؛ وزن ها با استفاده از روش گرم و سرد کردن شبیه سازی شده به دست می آیند . شبکه های هوپفیلد می توانند برای ساختن یک حافظه ی شرکت *پذیر* [۲] به کار روند ، در این شبکه ها ، یک قسمت از یک الگو برای شبکه ارایه می شود و شبکه تمام الگو را به یاد می آورد . برای حرف شناسی [۳] هم به کار می رود ، همچنین برای مسایل بهینه سازی هم به کار می رود و اغلب برای مدل فعّالیّت مغز استفاده می شود .

---

[۱] Hopfield networks

[۲] associative memory

[۳] letter recognition

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

فصل نوزدهم

الگوریتم های ژنتیکی [۱]

[۱] Genetic Algorithms

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

تاریخچه

در دهه ی هفتاد میلادی توسط جان هُلَنـد [۱] بیـان شـد ؛ در دهـه ی هشـتاد مـیلادی عمومیّـت یافـت و براساس تئوری تکاملی داروین (باقی ماندن شایسته ترین) بود . الگوریتم هـای ژنتیکـی مـی تواننـد بـرای حـلّ انواعی از مسایل که برای حل ، آسان نمی باشند با استفاده از روش های دیگر مورد استفاده قرار گیرند .

تکامل در دنیای واقعی

هر سلّول یک موجود زنده دارای *کروموزوم* [۲] هایی می باشد . هر کروموزوم شامل یک مجموعـه از *ژن* [۳] ها می باشد . هر ژن برخی از خواص موجود زنده (مانند رنگ چشم) را تعیین می کند . یک مجموعـه از ژن ها در برخی از موارد *ژنوتیپ* (نوع معرّف و نماینده ی یک جـنس) [۴] نامیـده مـی شـود . یـک مجموعـه از

---

[۱] John Holland
[۲] chromosome
[۳] gene
[۴] genotype

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

خصوصیّات (نظیر رنگ چشم) گاهی *فنوتیپ* [۱] نامیده می شود . انتشار ، شامل بازترکیب [۲] ژن ها از والـدین و سپس مقدار کوچکی *جهش* [۳] (خطاها) در کپی کردن می باشد . شایستگی [۴] یک موجود زنده مقـداری اسـت که می تواند قبل از مردنش انتشار دهد . تکامل ، براساس "بقای مناسب ترین [۵] " می باشد .

شروع با یک تصوّر

فرض کنید که شما مشکلی دارید و نمی دانید که چگونه آن را حل نمایید . برای حل این مشکل چـه کار می توانید بکنید ؟ آیا می توانید به طریقی از یک کامپیوتر برای پیدا کردن یک راه حل اسـتفاده نماییـد ؟ این کار باید خوب باشد ! می توانید آن را انجام دهید ؟

یک راه حل گنگ

یک الگوریتم "تولید و تست گنگ" :

تکرار کن

یک راه حل ممکن تصادفی را تولید کن

راه حل را امتحان کن و خوب بودن آن را بسنج

تا موقعی که راه حل به اندازه ی کافی خوب باشد

---

[۱] phenotype
[۲] recombination
[۳] mutation
[۴] fitness
[۵] survival of the fittest

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

آیا ما می توانیم از این ایده ی گنگ استفاده نماییم ؟

برخی اوقات بله : در صورتی که تعداد راه حل های کمی وجود داشته باشد و شما به اندازه ی کافی زمان در اختیار داشته باشید ، آن گاه روش های این فرمی می توانند استفاده شوند . *امّا برای بیش تر مسایل ، ما نمی توانیم از این راه حل ها استفاده نماییم :* در زمانی که راه حل های ممکن ، زیاد باشند و شما به اندازه ی کافی زمان برای امتحان همه ی آن ها نداشته باشید ، این راه حل ها نمی توانند استفاده شوند .

ایده ای که کم تر دارای گنگ بودن می باشد (الگوریتم ژنتیکی)

یک مجموعه ی تصادفی از راه حل ها را تولید نمایید

تکرار کن

هر راه حل را در مجموعه امتحان کن و آن ها را رتبه بندی کن

برخی از راه حل های بد را از مجموعه بردار

برخی از راه حل های خوب را تکثیر کن

برخی از تغییرات کوچک را در مورد آن ها اعمال کن

تا زمانی که بهترین راه حل به اندازه ی کافی خوب شود

چگونه شما یک راه حل را کد می کنید ؟

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

بدیهی است که این وابسته به مسأله می باشد ! الگوریتم های ژنتیکی ، اغلب راه حل ها را به صورت رشته های بیتی [۱] باطول ثابت ، کد می کنند (به عنوان مثال ، ۱۰۱۱۱۰ ، ۱۱۱۱۱۱ ، ۰۰۰۱۰۱) . هر بیت برخی از خصوصیّات راه حل های ارایه شده برای مسأله را ارایه می کند . برای این که الگوریتم های ژنتیکی کار کنند ، ما نیاز به این داریم که هر رشته را تست نماییم و به آن امتیازی بدهیم که نشان دهنده ی چگونگی خوب بودن آن باشد .

مثال نادان – حفر برای روغن

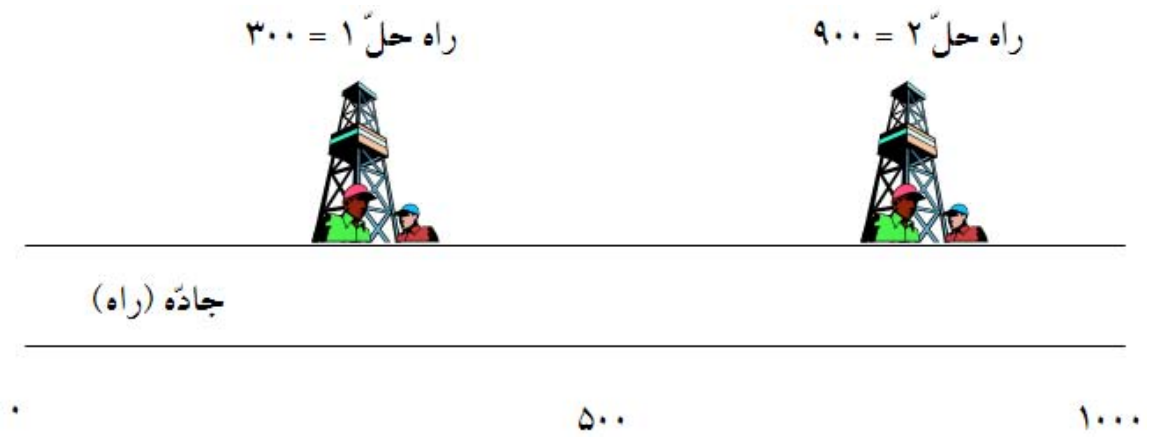
تصوّر کنید که شما باید برای روغن جایی را در طول یک جاده ی بیابانی یک کیلومتری ، حفر نمایید .

مسأله : بهترین مکان در جادّه را که بیش ترین مقدار روغن را در روز تولید می کند ، انتخاب نمایید .

ما می توانیم هر راه حل را به صورت یک وضعیّت در جادّه نمایش دهیم . تصوّر نمایید که تمام اعداد بین [۱۰۰۰، ۰] باشند .

کجا را برای روغن ، حفر نماییم ؟

[۱] bitstrings

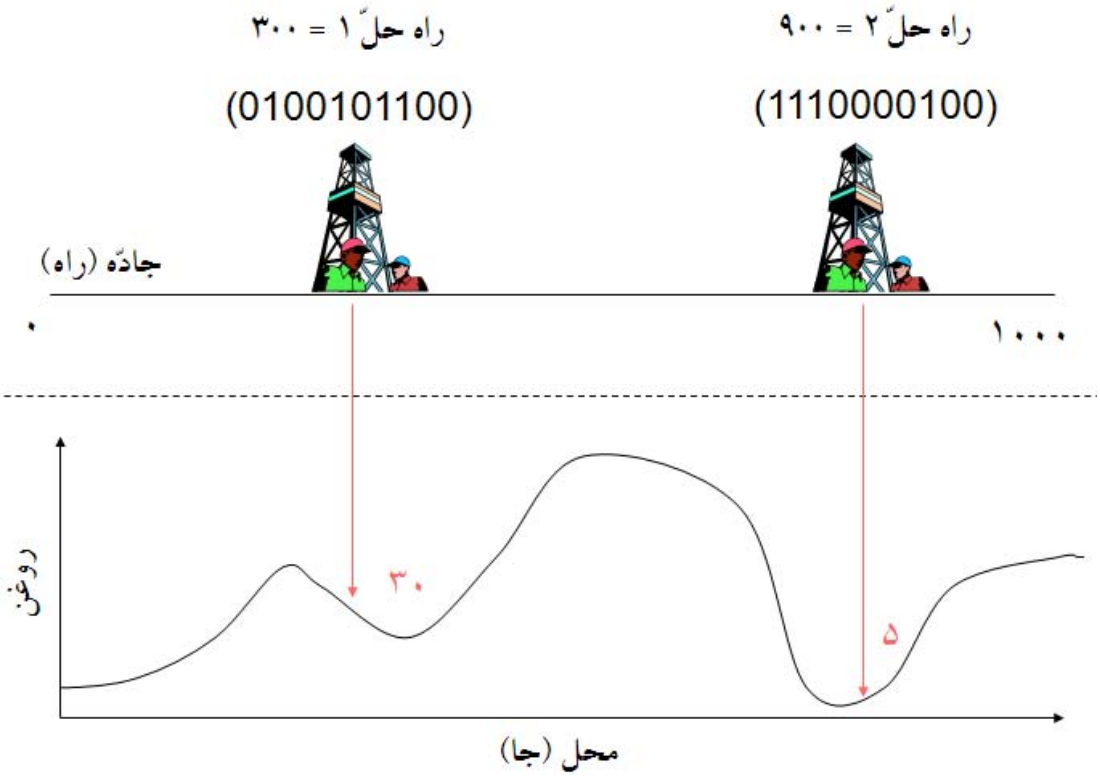


مجموعه ی همه ی راه حل های ممکن [۰،۱۰۰۰] ، فضای جستجو یا فضای حالت نام دارد . در این مورد ، این فقط یک عدد است اما می تواند تعداد زیادی عدد یا سمبل باشد . اغلب ، الگوریتم های ژنتیکی اعداد را به صورت دودویی (باینری) کد می کنند . در مورد این مثال ، ده بیت را که برای نمایش ۰ ... ۱۰۰۰ کافی است را انتخاب می نماییم .

تبدیل به رشته ی باینری

	۵۱۲	۲۵۶	۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱
۹۰۰	۱	۱	۱	۰	۰	۰	۰	۱	۰	۰
۳۰۰	۰	۱	۰	۰	۱	۰	۱	۱	۰	۰
۱۰۲۳	۱	۱	۱	۱	۱	۱	۱	۱	۱	۱

در الگوریتم های ژنتیکی، این رشته های کد شده گاهی ژنوتیپ ها یا کروموزوم ها نامیده می شوند و بیت های تکی گاهی اوقات ژن ها نامیده می شوند .



خلاصه

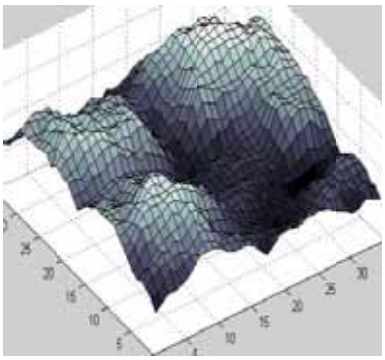
ما تا کنون دیدیم که چگونه راه حل ها را به صورت یک عدد بیان نماییم . یک عدد را به صورت یک رشته ی بیتی کد کردیم . یک رتبه یا درجه را برای هر عدد داده شده به منظور تعیین میزان خوب بودن آن راه به آن عدد نسبت دادیم که اغلب *تابع شایستگی* [1] نامیده می شود . مثال روغن در واقع بهینه سازی با استفاده از یک تابع $f(x)$ است که ما باید پارامتر x را تطبیق دهیم .

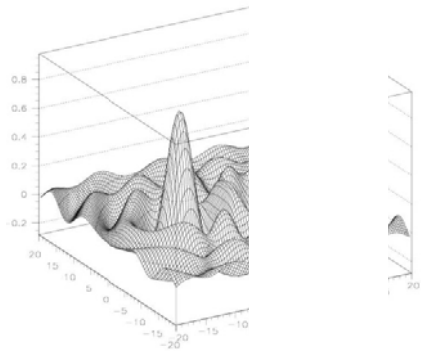
[1] fitness function

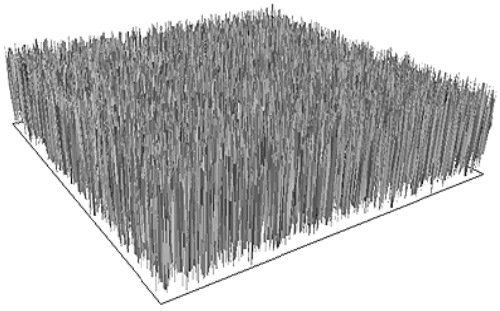
مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

فضای جستجو

برای یک تابع ساده ی f(x) ، فضای جستجو یک بعدی است . امّا با کد کردن چند مقدار به کروموزوم ، ابعاد زیادی می تواند به وجود آید به عنوان مثال برای حالت دوبعدی تابع f به صورت f(x,y) خواهد بود . فضای جستجو می تواند به صورت یک سطح باشد . هر ژنوتایپ ممکن ، یک نقطه در فضا است . یک الگوریتم ژنتیکی تلاش می کند که نقطه ها را به جاهای بهتر (با شایستگی بالاتر) در فضا منتقل نماید . در زیر سه نمونه از منظره های شایستگی را مشاهده می نماییم .







مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

بدیهی است که نوع فضای جستجو تعیین می کند که چگونه یک الگوریتم ژنتیکی کار کند . یک فضای کاملاً تصادفی برای یک الگوریتم ژنتیکی ، بد خواهد بود . همچنین الگوریتم های ژنتیکی ، اگر فضاهای جستجو شامل تعداد زیادی از موارد تصادفی باشند ، ممکن است در ماکزیمم محلّی بیفتند .

اضافه کردن جنسیّت – Crossover

اگرچه که الگوریتمی که ما گفتیم برای فضاهای جستجوی ساده کار می کند امّا این الگوریتم هنوز خیلی ساده است . این الگوریتم به جهش های تصادفی برای یافتن یک راه حلّ خوب ، وابسته می باشد . با اضافه نمودن جنسیّت به این الگوریتم می توانیم نتایج بهتری را به دست آوریم . این کار با انتخاب دو والد در موقع تکثیر [۱] و ترکیب ژن های آن ها برای تولید فرزند انجام می شود . دو رشته ی بیتی (کروموزوم) والد با امتیاز بالا انتخاب می شوند و با نرخ Crossover با هم ترکیب می شوند . دو فرزند (رشته ی بیتی) به وجود می آیند و هر فرزند ممکن است به صورت تصادفی تغییر داده شود (جهش) .

انتخاب والدها

روش های زیادی برای انتخاب کروموزوم های با امتیاز بهتر وجود دارد . امتیاز ، اغلب *شایستگی* [۲] نامیده می شود . از روش " چرخش رولت " [۳] می توان به صورت زیر استفاده نمود :

- شایستگی همه ی کروموزوم ها را اضافه نمایید .
- یک عدد تصادفی R را در محدوده ی آن به وجود آورید .
- اوّلین کروموزوم موجود در جمعیّت را با این شرط که زمانی که همه ی شایستگی های قبلی اضافه می شوند ، لااقل مقدار R را می دهد انتخاب نمایید .

---

[۱] reproduction

[۲] fitness

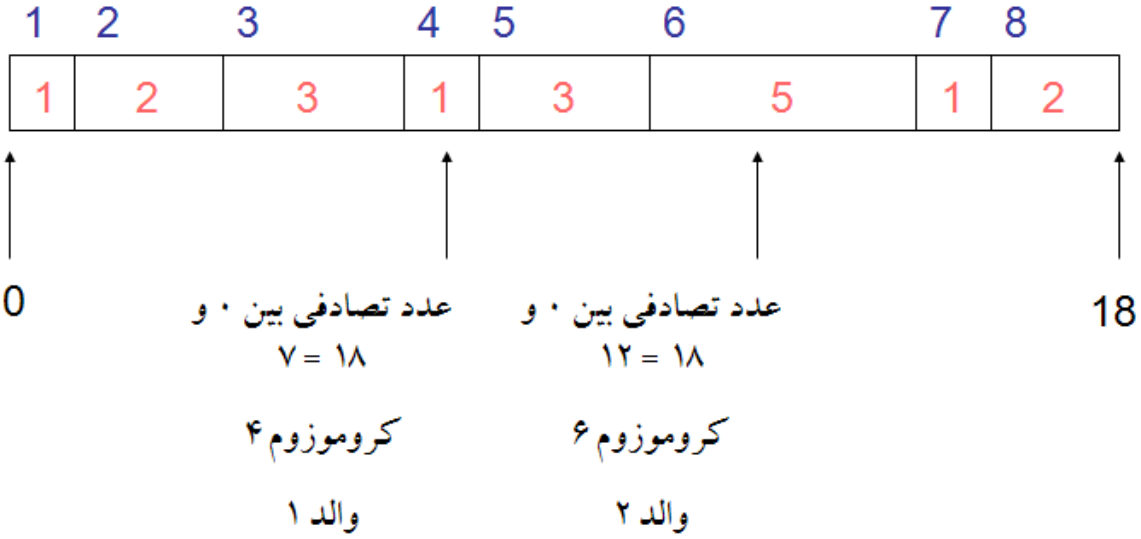
[۳] Roulette Wheel

جمعیّت نمونه

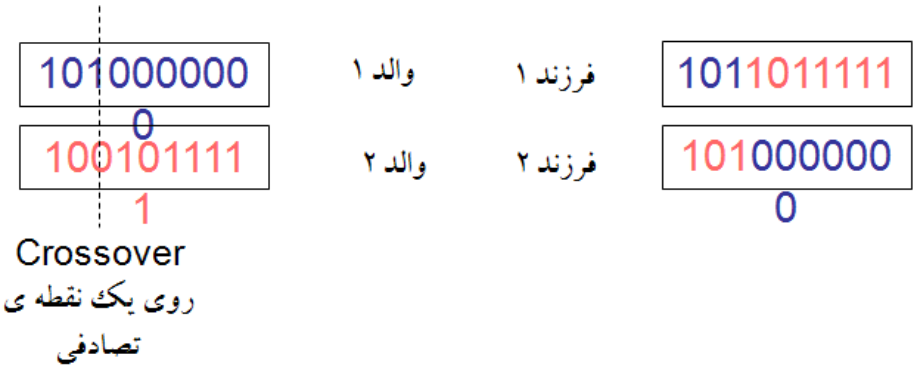
شماره	کروموزوم	شایستگی
1	1010011010	1
2	1111100001	2
3	1011001100	3
4	1010000000	1
5	0000010000	3
6	1001011111	5
7	0101010101	1
8	1011100111	2

انتخاب چرخش رولت

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸



Crossover – بازترکیب (جفت گیری) [1]



با احتمال بالا (نرخ Crossover) ، Crossover را به والدها اعمال نمایید . (مقدارهای عادّی بین ۰.۸ تا ۰.۹۵ می باشند)

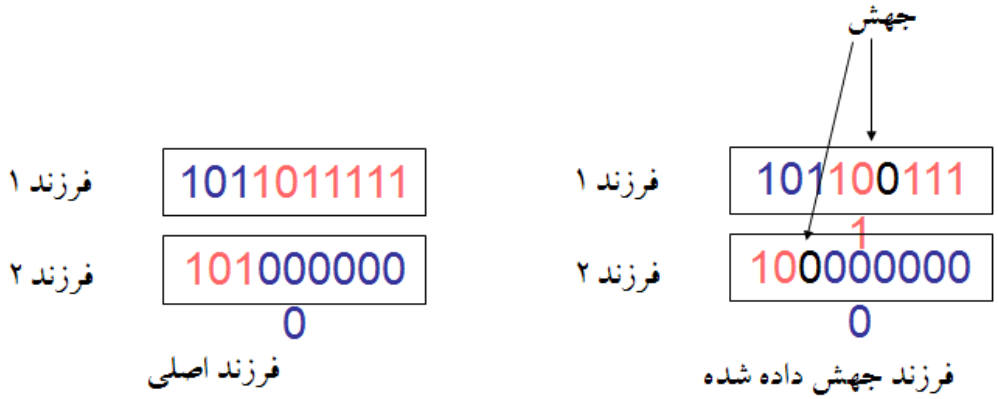
جهش

[1] Recombination : ترکیب جدیدی از فنوتیپها در نتاج حاصل از والدین با فنوتیپی متفاوت .

یک یا بیش تر ژن به صورت تصادفی به یک مقدار تصادفی ، جهش داده می شوند ، مثلاً :

۱۱۱۱۱۱۱۱۱۱ → ۱۱۰۱۰۱۱۱۱۱

در مورد مثال قبلی داریم :



برگشت به الگوریتم ژنتیکی

یک جمعیّت را با استفاده از کروموزوم های تصادفی ، تولید کن

برای هر نسل کارهای زیر را انجام بده

شایستگی هر کروموزوم را محاسبه کن

کارهای زیر را تکرار کن

انتخاب رولت را برای انتخاب جفت های والدها به کار ببر

فرزند را با استفاده از Crossover و جهش تولید نما

تا زمانی که جمعیّت جدید تولید شود

مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

تا زمانی که بهترین راه حل به اندازه ی کافی خوب شود

تنوّع های زیاد الگوریتم های ژنتیکی

به خاطر موارد زیر ، الگوریتم های ژنتیکی دارای تنوّع زیادی هستند :

- انواع مختلف انتخاب که رولت نیستند ؛ مثل ، مسابقه [۱] ، نخبه سالاری [۲] و غیره .
- تفاوت در جفت گیری ؛ مثل ، Crossover چند نقطه ای .
- انواع مختلف کد کردن رشته ی بیتی ؛ مثل ، مقدارهای صحیح و مجموعه ی منظّم از سمبل ها
- انواع مختلف جهش

پارامترهای زیاد برای تنظیم

هر پیاده سازی الگوریتم ژنتیکی به تصمیم گیری در مورد تعدادی از پارامترها نیاز دارد ؛ مثل ، اندازه ی جمعیّت (N) ، نرخ جهش (m) ، نرخ Crossover (c) . که اغلب ، این ها باید براساس نتیجه های به دست آمده تنظیم شوند . مقدارهای معمولی شاید $N = 50$ ، $m = 0.05$ و $c = 0.9$ باشند .

چگونه Crossover کار می کند ؟

تعداد زیادی نظریّه در این مورد وجود دارد البتّه برخی مخالفت ها هم با این نظریّه ها وجود دارد . آقای هُلَند نظریّه ی " الگو " را معرّفی کرد ؛ این ایده می گوید که Crossover ، بیت های خوب والدین مختلف را نگهداری می کند و آن ها را برای تولید راه حل های بهتر ترکیب می نماید . بنابراین ، یک الگوی خوب کد کننده باید بیت های خوب را در طول Crossover و جهش نگهداری نماید .

---

[۱] Tournament

[۲] Elitism

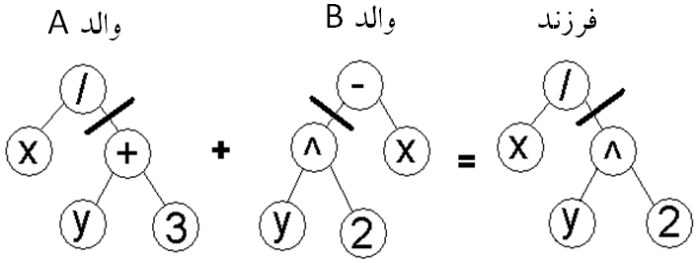
مترجم: سهراب جلوه گر
ویرایش دوّم ، بهار ۱۳۸۸

کاربردهای الگوریتم های ژنتیکی

الگوریتم های ژنتیکی در بسیاری از موارد و زمینه های تحقیقاتی ، نظیر مسایل عددی ، مسأله ی مسافرت شخص دوره گرد ، مدار همیلتونی ، مسیر اویلری ، پیدا کردن شکل مولکول های پروتئین ، مسایل NP سخت ، طراحی شبکه های عصبی ، توابع برای به وجود آوردن تصاویر ، مدلسازی شناختی و تئوری بازی ها به کار می روند .

برنامه نویسی ژنتیکی

زمانی که کروموزوم یک برنامه یا تابع را کد می کند ، این کار برنامه نویسی ژنتیکی نامیده می شود . برای انجام این کار معمولاً از ارایه ی درختی استفاده می شود . Crossover ، باعث جابه جا شدن زیر درخت ها میان والدین می شود .



این کار برای برنامه های کوچک ، مفید است ولی برای برنامه های بزرگ با توابع پیچیده ، ناکارآمد می باشد .

توابع شایستگی ناآشکار (ضمنی)

اغلب الگوریتم های ژنتیکی از توابع شایستگی ضمنی استفاده می کنند (همانند آنچه که در مورد مثال روغن دیدید) . برخی از الگوریتم های ژنتیکی (نظیر زندگی مصنوعی یا روبوتیک تکاملی) از توابع شایستگی ضمنی و پویا استفاده می کنند .