

«بسمه تعالی»

شبکه عصبی تکاملی

نام و نام خانوادگی: سحر عرشی

نام دانشگاه: دانشکده ی فنی شریعتی

تلفن: ۸۰۰۱۲۷۲ (۰۲۱)

آدرس: تهران , خیابان یوسف آباد , خیابان جهان آرا , کوچه ی شهید محمد زنجانی , پلاک (۱۸) , طبقه ی (۴)

چکیده:

شبکه های عصبی اگرچه دارای محدودیت هایی می باشند ولیکن برای حل مسائل مختلف به کار برده می شوند. GA ها، تکنیک مؤثری برای بهینه سازی هستند و می توانند تا حدود زیادی مشکلات مربوط به شبکه های عصبی را برطرف نماید. ایده ی اصلی شبکه عصبی تکاملی مدل کردن شبکه ی عصبی با کروموزوم است.

شبکه عصبی تکامل (Evolutionary Neural Network) , تقاطع (Crossover) , جهش (Mutation) , کروموزوم (Chromosome) , تابع صلاحیت (Fitness) , والد (Parent) , فرزند (Offspring) , بهینه سازی (Optimization) , توپولوژی (Topology) , وزن (Weight)

فهرست:

۳ مقدمه:
۴ بهینه سازی وزنی تکاملی (<i>Evolutionary Weight Optimisation</i>)
۷ توپولوژی شبکه تکاملی (<i>Evolutionary Network topology</i>)
۹ الگوریتم GA برای معماری شبکه:
۱۰ فهرست مراجع:

شبکه عصبی تکاملی

مقدمه:

در بسیاری از کاربردها تنها به دست آوردن اطلاعاتی راجع به منابع مختلف کافی نبوده و نیاز داریم که تکنولوژی های هوشمند مختلف را با یکدیگر در هم آمیزیم. و نیاز به یک چنین ترکیبی ما را به سمت نیاز به استفاده از سیستم های هوشمند hybird ی سوق داده است.

یک سیستم هوشمند hybird ی، سیستمی است که حداقل ۲ تکنولوژی هوشمند را با هم در هم می آمیزد. به عنوان مثال ترکیب کردن یک شبکه عصبی با یک سیستم فازی یک سیستم فازی پیوندی عصبی را نتیجه می دهد. (hybird-neuro-fuzzy-System) و ترکیب یک الگوریتم ژنتیک با یک شبکه عصبی یک شبکه عصبی تکاملی (Evolutionary-nenal -network) را نتیجه می دهد.

شبکه های عصبی اگرچه دارای محدودیت هایی می باشند و لیکن برای حل مسائل مختلف به کار برده می شوند. یکی از بزرگترین محدودیت هایش مربوط به train کردن (آموزش دادن) به شبکه عصبی است. الگوریتمی را برای این منظور به کار برده می شود. نمی تواند لزوماً یک جواب بهینه را متضمن شود. در کاربردی واقعی الگوریتم (back-propagation) ممکن است که به مجموعه ای از وزن هایی همگرایی (Converge) پیدا کند که بهینه نبوده (کمتر از حد بهینه) هستند و نمی تواند از این مجموعه جواب ها بگریزد.

شکل دیگر مربوط به انتخاب یک توپولوژی بهینه برای شبکه عصبی می باشد. یک معماری درست از شبکه برای یک شکل شخص مشخص معمولاً توسط روشهای ابتکاری انجام می شدند و طراحی یک شبکه عصبی یک هنر است تا یک عمل مهندسی.

GA ها، تکنیک مؤثری برای بهینه سازی هستند و می توانند در آن واحد هم وزن و هم بهینه سازی را برای یک انتخاب توپولوژی راهنمایی کنند.

بهینه سازی وزنی تکاملی (Evolutionary Weight Optimisation)

ابتدا نیاز داریم که حوزه مسئله خود را به صورت کروموزوم نشان دهیم. فرض کنید که به عنوان مثال می خواهیم مجموعه ای از وزن های بهینه را برای یک شبکه عصبی چند لایه ای بیایم. وزن های اولیه شبکه به صورت تصادفی در فاصله $[-1, 1]$ انتخاب شده اند. مجموعه ای از وزن ها می تواند به صورت یک ماتریس مربعی همان طور که در جدول (۱) نشان داده شده نمایش داده شود. که در این ماتریس هر عدد مربوط به وزن لینک بین دو نرون می باشد. عدد $+$ یعنی اینکه هیچ ارتباطی بین دو نرون وجود ندارد. و در مجموع ۱۶ وزن ارتباطی بین نرون ها در شکل (۱) وجود دارد. و از آنجایی که هر کروموزوم مجموعه ای از ۱۶ وزن ها می باشد، مجموعه ای از وزن ها می تواند توسط ۱۶ ژن کروموزوم نمایش داده شده و هر ژن مربوط به یک وزن لینک در شبکه می باشد.

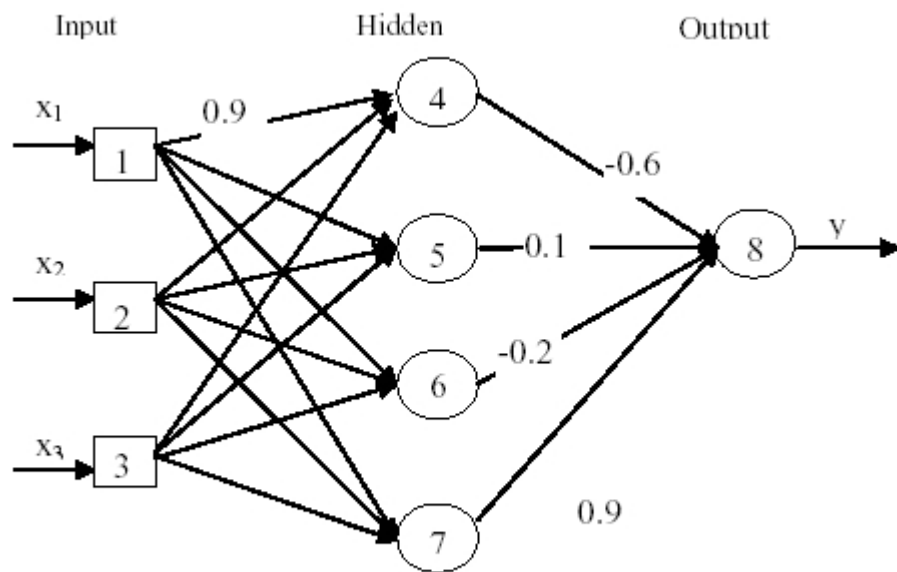


Fig. 1 Example neural network

Table 1 Weights matrix for neural net in Figure 10.3

To\From	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0.9	-0.3	-0.7	0	0	0	0	0
5	-0.8	0.6	0.3	0	0	0	0	0
6	0.1	-0.2	0.2	0	0	0	0	0
7	0.4	0.5	0.8	0	0	0	0	0
8	0	0	0	-0.6	0.1	-0.2	0.9	0

و چنان چه سطرهای ماتریس را بدون احتساب ۰ ها پشت سرهم رشته کنیم به یک کروموزوم می‌رسیم که در شکل (۲) پیداست.

0.9	-0.3	-0.7	-0.8	0.6	0.3	0.1	-0.2	0.2	0.4	0.5	0.8	-0.6	0.1	-0.2	0.9
-----	------	------	------	-----	-----	-----	------	-----	-----	-----	-----	------	-----	------	-----

Figure 2 Encoding a set of weights in a chromosome

دقت کنید که هر سطر هم اکنون مجموعه‌ای از لینک‌های وزن‌ها را برای یک نرون نشان می‌دهد. برای دستیابی به این مقصود به جای مربوط ساختن یک وزن به هر ژن ترجیحاً مجموعه‌ای از وزن‌های ورودی یک نرون را، به هر ژن مربوط سازیم.

قدم دوم مشخص کردن یک تابع *fitness* (صلاحیت) برای ارزیابی کردن اجرای کروموزوم می‌باشد و این تابع می‌بایستی که *performance* یک شبکه عصبی را برآورد کند. می‌توانیم در اینجا یک مثال روشن‌تابی که توسط آن مجموع مربعات خطاها تعریف شده است. ندانیم برای ارزیابی *fitness* یک کروموزوم هر وزن موجود در کروموزوم به صورت نسبی به یک *link* در شبکه منتسب شده است. سپس مجموعه‌ای از مثال‌های *training* به شبکه اعمال می‌شوند. مجموع مربعات خطاها محاسبه می‌شود.

هر چقدر که این مجموع کوچکتر باشد آن کروموزوم مناسبتر است. به تعبیر دیگر، GA می‌کوشد تا مجموعه‌ای از وزن‌ها را که مجموع مربعات خطاها را به حداقل می‌رساند را محاسبه کند.

قدم سوم انتخاب عملگرهای ژنتیکی Crossover و mutation است. یک عملکرد crossover از ۲ کروموزوم والد یک فرزند را با materials (مصالح ژنتیکی) والدین می‌سازد. و هر ژن از کروموزوم فرزند، مربوط به ژن والد منتخب تصادفی می‌باشد. و شکل (۳) زیر کاربرد عملی crossover را نشان می‌دهد.

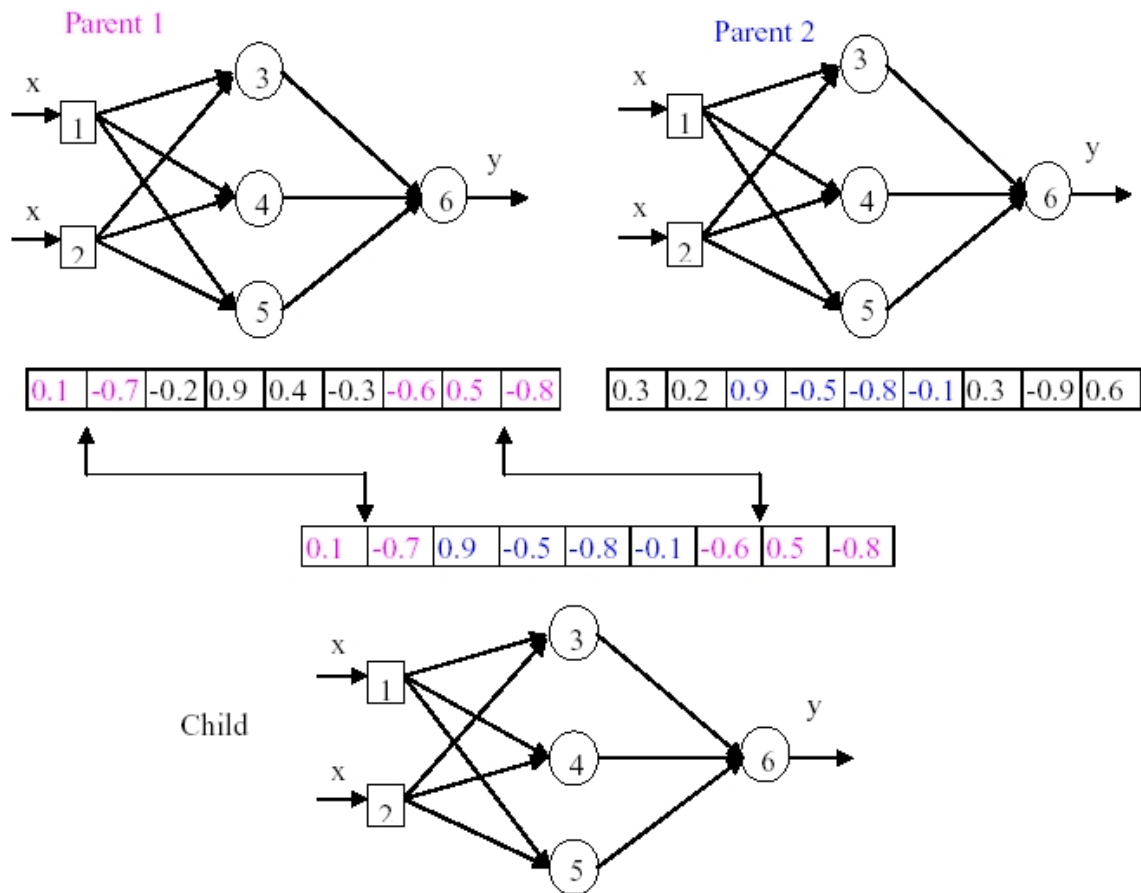


Figure 3 Crossover operation

یک عملگر mutation یک ژن را در کروموزوم به صورت تصادفی انتخاب می‌کند و یک مقدار تصادفی کوچک بین ۱-۰ را به هر وزن در شبکه اضافه می‌کند. شکل (۴) مثالی از mutation است. که عدد ۰/۵- را در این مثال اضافه می‌کند.

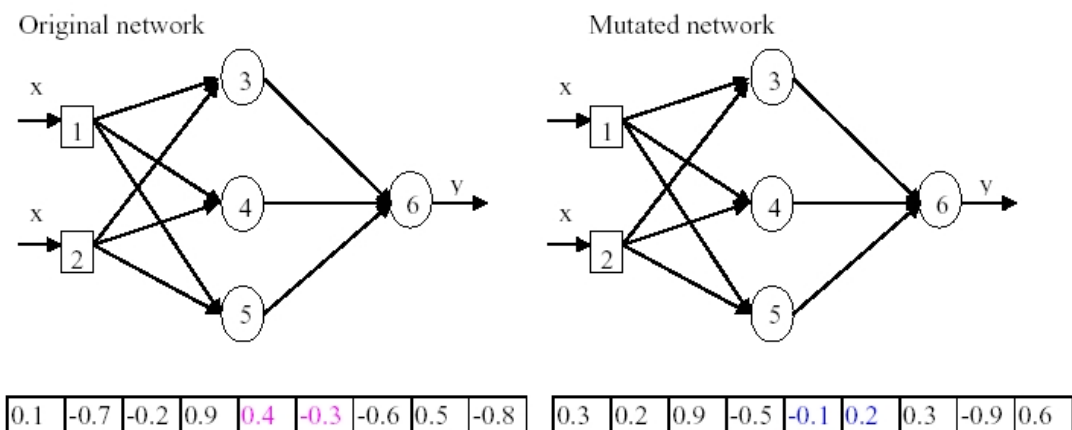


Figure 4 Mutation operation

قدم بعدی مشخص کردن سائز **population**. سائز شبکه ها با وزن های مختلف، احتمالات **crossover** و **mutation** و تعداد نسل ها می باشد.

توپولوژی شبکه تکاملی (*Evolutionary Network topology*)

قبلاً فرض کردیم که ساختمان شبکه ثابت باشد و آموزش **evolution** فقط برای بهینه کردن وزن ها در یک شبکه داده شده باشد. اگرچه معماری یک شبکه (تعداد نرون ها و ارتباطات) آنها با یکدیگر معمولاً پیروز یا شکست **application** را مشخص می کند، معمولاً معماری شبکه به روش سعی و خطا انجام می شود.

و ما نیازمند یک متد قوی برای طراحی اتوماتیک معماری یک **application** خاص هستیم.

ایده اصلی مناسب نمو دادن معماری شبکه یک **search** ژنتیکی در **population** معماری ها می باشد. البته می بایستی اول یک متد برای کد کردن معماری شبکه به کروموزوم داشته باشیم.

روش های مختلفی برای کد کردن ساختمان شبکه وجود دارند. و کلید اصلی این است که تصمیم بگیریم چه مقدار اطلاعات برای نمایش شبکه لازم می باشد. و هر چه قدر که تعداد پارامترها در معماری شبکه بیشتر شود هزینه های محاسبات هم طبعاً بیشتر می شود. برای توضیح بیشتر ما یک متد مستقیم برای کد کردن را در نظر می گیریم. اگرچه کد کردن مستقیم یک تکنیک محدود کننده می باشد و تنها می تواند به شبکه های با تعداد ثابت نرون اعمال شود می تواند به خوبی نشان دهد که چگونه یک **توپولوژی** ارتباط می تواند نمود کند. **توپولوژی** ارتباط یک شبکه عصبی می تواند توسط یک ماتریس مربعی اتصال نشان داده شود.

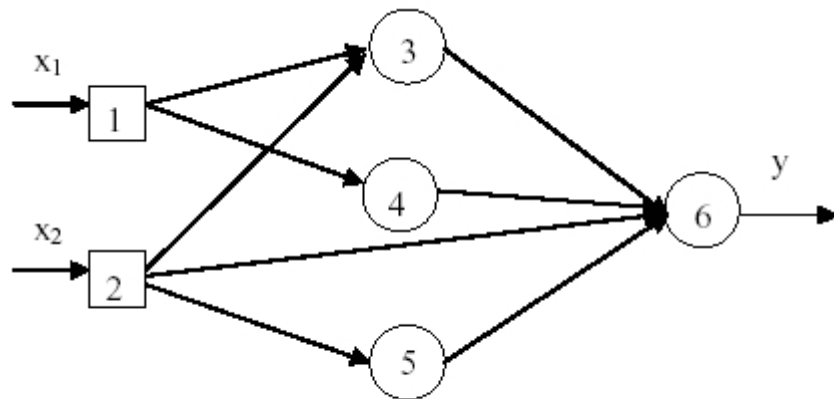


Figure 5 Network Topology

Table 2 Direct Encoding of the Network Topology

To\From	1	2	3	4	5	6
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	1	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0
6	0	1	1	1	1	0

همانطور که در جدول (۲) پیداست و هر خانه در ماتریس نشان دهنده یک نوع ارتباط از یک نرون در ستون به نرون دیگر در سطر می باشد. عدد ۰ نشان دهنده این است که هیچ نوع ارتباطی بین نرون ها موجود نمی باشد. و عدد ۱ نوع ارتباطی را مشخص می کند که در آن وزن می تواند در حین یادگیری (Learning) تغییر یابد. برای اینکه ماتریس اتصالات را با کروموزوم شبیه سازی کنیم همان طور که در شکل (۶) پیداست و تنها کافی است که سطرهای ماتریس را به دنبال هم رشته کنیم.

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Figure 6 Chromosome representation of Network Topology

الگوریتم GA برای معماری شبکه

گام ۱: سائز **population** کروموزوم و احتمالات **crossover** و **mutation** را انتخاب کن و تعداد دوره های **training** را تعریف کن.

گام ۲: تابع **fitness** را برای محاسبه **performance** یا **fitness** یک کروموزوم اولیه تعریف درحالت کلی **fitness** یک شبکه نباید تنها بر مبنای صحت آن باشد. بلکه بایستی سرعت یادگیری (**Learning speed**)، سائز (**size**) و پیچیدگی (**Complexity**) آن نیز مدنظر قرار گیرد.

گام ۳: تولید تصادفی جمعیت آغازی کروموزوم ها

گام ۴: یک کروموزوم خاص را به شبکه عصبی کد می کنیم. از آنجایی که همه شبکه ها می خواهند **Feedforward** شوند از همه ارتباطات پسخورد (**feedback**) صرف نظر کنید. مقدار آنها و شبکه را به تصادفی کوچک **set** می کنیم. همان فاصله [۱-۱۰] می باشد.

مجموع مربعات خطاها را محاسبه کنید و **fitness** شبکه را مشخص کنید.

گام ۵: مرحله ی ۴ را بر روی همای نمونه ها در جمعیت اعمال کن.

گام ۶: دو کروموزوم والد را متناسب با تابع **fitness** شان انتخاب کن.

گام ۷: مجموعه کروموزومهای فرزند را با اعمال عملگرهای ژنتیکی **Crossover** و **Mutation** بدست آور.

گام ۸: مجموعه کروموزومهای فرزند را جایگزین جمعیت قبلی کن.

گام ۹: مرحله ۹ را تا زمانی که سائز جدید جمعیت برابر سائز قبلی شد تکرار کن.

گام ۱۰: به گام ۴ برو و این پروسه را تا زمانی که تعدد مشخصی از نسل جدید به جود آمدند تکرار کن.

مراجع:

[1] Evolutionary Neural Network :The University of Melbourne :*Assoc.Professor*
Baikunth Nath