هوش مصنوعی پیشرفته

شبکه‌های عصبی تطوری

Evolutionary Neural Networks

کاظم فولادی

دانشکده مهندسی برق و کامپیوتر

دانشگاه تهران

http://courses.fouladi.ir/aai

هوش مصنوعی

شبکه‌های عصبی تطوری

۱

مقدمه

شبکه‌های عصبی تطوری

مشکلات شبکه‌ی عصبی و راه‌حل آن با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

مشکلات خاص استفاده از شبکه‌های عصبی در حل مسئله	
آموزش (بهنیه‌سازی وزن‌ها) *Training (Weight Optimization)*	**انتخاب معماری بهینه** *Selecting Optimal Architecture*

اگرچه شبکه‌های عصبی برای حل انواع گوناگونی از مسائل استفاده می‌شود،
با این وجود، هنوز محدودیت‌هایی دارد:

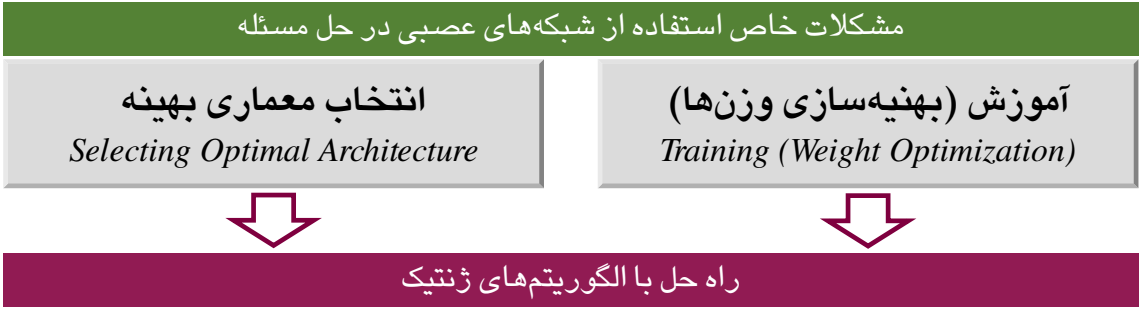
- یکی از متداول‌ترین محدودیت‌های شبکه‌های عصبی، آموزش (بهینه‌سازی وزن‌ها) آن است:
  الگوریتم «پس‌انتشار خطا» نمی‌تواند تضمین کند که جواب بهینه را می‌یابد.
  در کابردهای دنیای واقعی، الگوریتم پس‌انتشار خطا ممکن است به مجموعه‌ای از وزن‌های زیربهنیه همگرا شود که نمی‌تواند از آنها فرار کند.
  در نتیجه، شبکه‌ی عصبی اغلب قادر به یافتن یک راه‌حل مطلوب برای مسئله‌ی موجود نیست!
- مشکل دیگر مربوط به انتخاب معماری بهینه برای شبکه‌ی عصبی است:
  معماری «درست» شبکه برای یک مسئله‌ی خاص اغلب توسط هیوریستیک‌ها انتخاب می‌شود و طراحی معماری شبکه‌ی عصبی، بیشتر یک هنر است تا مهندسی!

الگوریتم‌های ژنتیک یک تکنیک مؤثر بهنیه‌سازی است که
می‌تواند بهینه‌سازی وزن‌ها و انتخاب معماری شبکه را هدایت کند.

شبکه‌های عصبی تطوری

مشکلات شبکه‌ی عصبی و راه‌حل آن با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS



شبکه‌های عصبی تطوری

آموزش وزن‌های یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

مجموعه‌ی وزن‌های شبکه‌ی عصبی را در قالب یک کروموزم کدگذاری می‌کنیم؛
و برای یافتن کروموزوم بهینه از الگوریتم ژنتیک استفاده می‌کنیم.

هوش مصنوعی

شبکه‌های عصبی تطوری

۲

تطور شبکه‌های عصبی با الگوریتم‌های ژنتیک

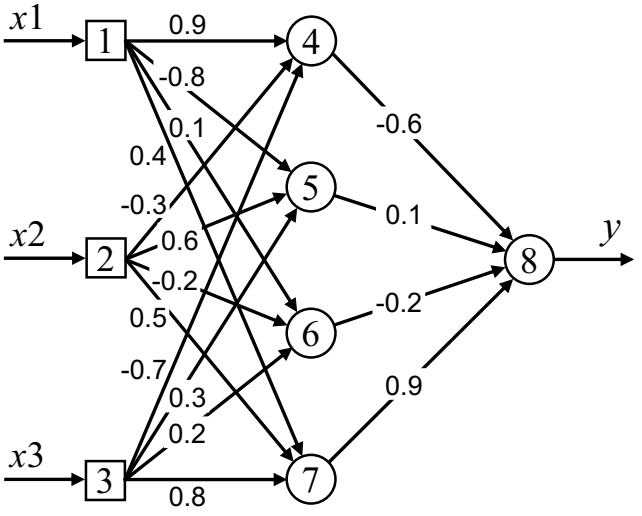
شبکه‌های عصبی تطوری

آموزش وزن‌های یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

کدگذاری یک مجموعه از وزن‌ها در یک کروموزم

From neuron:	1	2	3	4	5	6	7	8
To neuron: 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



Chromosome:

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

شبکه‌های عصبی تطوری

آموزش وزن‌های یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

تعریف یک تابع برازش

گام دوم، تعریف یک **تابع برازش** برای ارزیابی کارآیی کروموزوم است:

این تابع باید کارآیی یک شبکه‌ی عصبی داده شده را تخمین بزند.
در اینجا می‌توانیم از یک تابع ساده تعریف شده با مجموع مربعات خطا استفاده کنیم.

$$\{(y_i, y_i^d)\}_{i=1}^N \quad \Rightarrow \quad f = \sum_{i=1}^N (y_i - y_i^d)^2$$

مجموعه‌ی آموزشی از مثال‌ها به شبکه نمایش داده می‌شود، و
مجموع مربعات خطا محاسبه می‌شود.
هرچه این مجموع کوچک‌تر باشد، کروموزوم مناسب‌تر است.

الگوریتم ژنتیک تلاش می‌کند مجموعه‌ای از وزن‌ها را بیابد که مجموع مربعات خطا را می‌نیمم کند.

شبکه‌های عصبی تطوری

آموزش وزن‌های یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

انتخاب عملگرهای ژنتیکی (تقاطع و جهش)

گام سوم، انتخاب عملگرهای ژنتیکی (تقاطع و جهش) است.

تقاطع
Crossover

دو کروموزم والد را می‌گیرد و یک فرزند واحد را از ترکیب ماده‌ی ژنتیکی هر دو والد می‌سازد.
هر ژن در کروموزوم فرزند با ژن متناظر در والد انتخاب شده به صورت تصادفی بازنمایی می‌شود.

جهش
Mutation

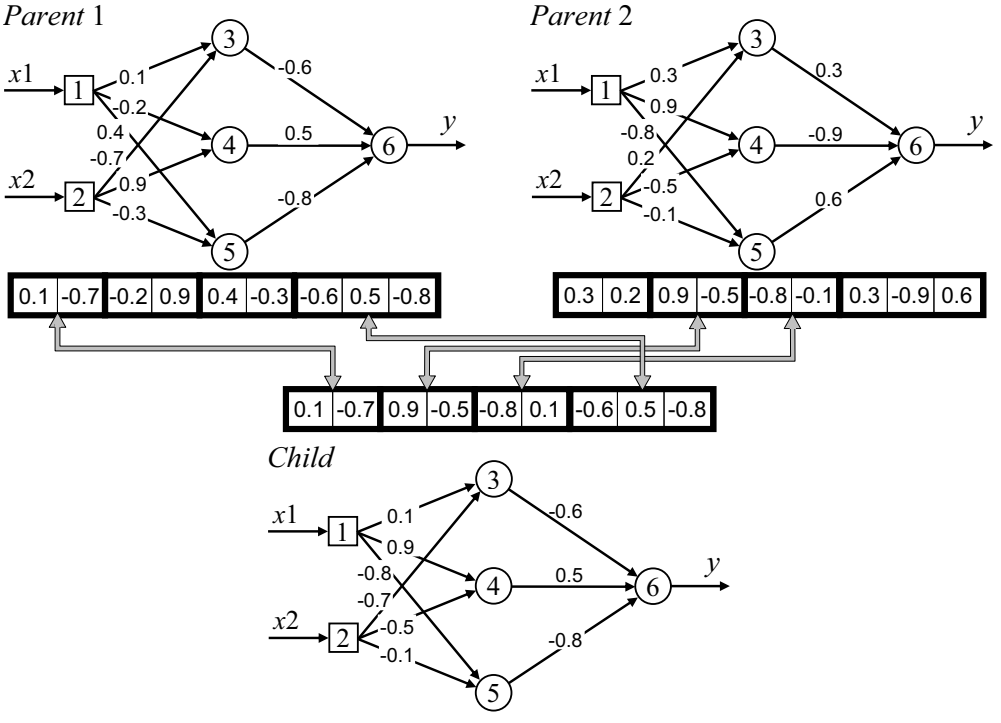
یک ژن در یک کروموزوم را می‌گیرد و یک مقدار کوچک تصادفی در بازه‌ی $[-1,1]$ را به همه‌ی وزن‌ها در آن ژن اضافه می‌کند.

شبکه‌های عصبی تطوری

آموزش وزن‌های یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

عملگر «تقاطع» برای بهینه‌سازی وزن‌ها

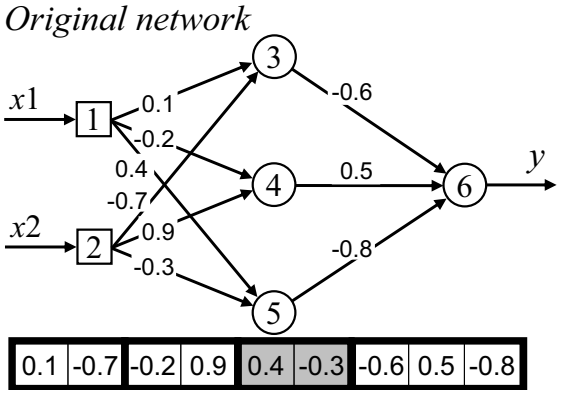


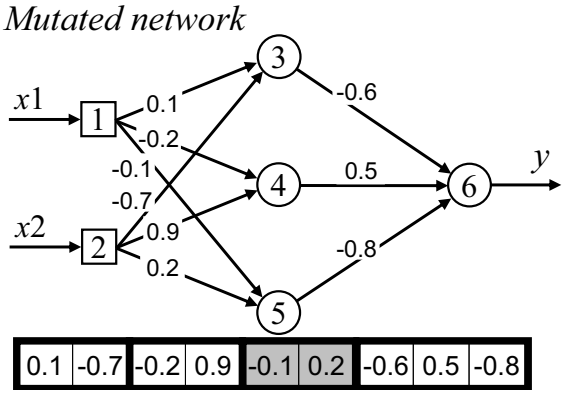
شبکه‌های عصبی تطوری

آموزش وزن‌های یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

عملگر «جهش» برای بهینه‌سازی وزن‌ها

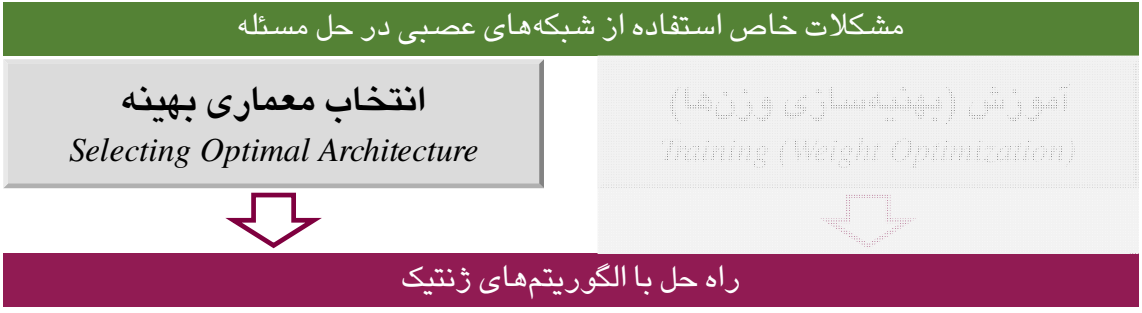




شبکه‌های عصبی تطوری

انتخاب معماری بهینه‌ی یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS



معماری شبکه (یعنی: تعداد نرون‌ها و چگونگی اتصالات آنها)
اغلب موفقیت یا شکست کاربرد شبکه‌ی عصبی را تعیین می‌کند.

معماری شبکه معمولاً به صورت سعی و خطا تعیین می‌شود؛
در اینجا نیازی بالایی برای یک روش طراحی خودکار معماری شبکه برای یک کاربرد خاص وجود دارد.

الگوریتم‌های ژنتیک می‌توانند به‌خوبی برای این کار استفاده شوند.

شبکه‌های عصبی تطوری

انتخاب معماری بهینه‌ی یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

کدگذاری معماری شبکه

ایده‌ی اصلی برای تطور یک معماری مناسب شبکه‌ی عصبی،
هدایت یک **جستجوی ژنتیکی** در یک جمعیت ممکن از معماری‌هاست.

برای این کار باید روشی برای کدگذاری یک معماری شبکه در قالب یک کروموزم بیابیم.

اتصالات توپولوژی یک شبکه‌ی عصبی، می‌تواند توسط یک **ماتریس مربعی همبندی** بازنمایی شود:

- هر درایه در این ماتریس نوع اتصالات از نرون مبدأ (ستون) به نرون مقصد (سطر) را نشان می‌دهد:
  0: به‌معنی عدم اتصال، 1: به‌معنی اتصال (که وزن آن می‌تواند در طول یادگیری تغییر کند.)
- برای تبدیل ماتریس همبندی به یک کروموزم، سطرهای این ماتریس را در قالب یک رشته کنار هم قرار می‌دهیم.

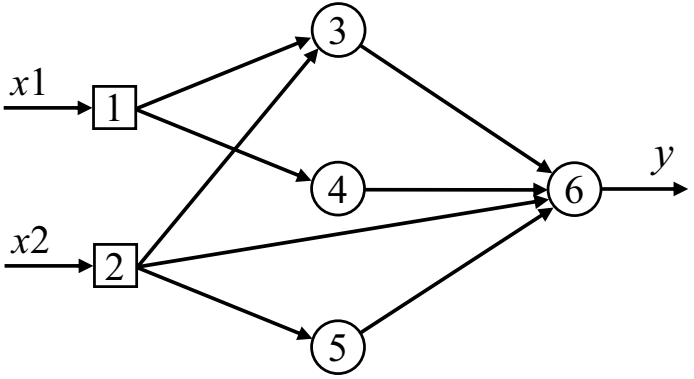
شبکه‌های عصبی تطوری

انتخاب معماری بهینه‌ی یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

کدگذاری معماری شبکه

From neuron:	1	2	3	4	5	6
To neuron: 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



Chromosome:

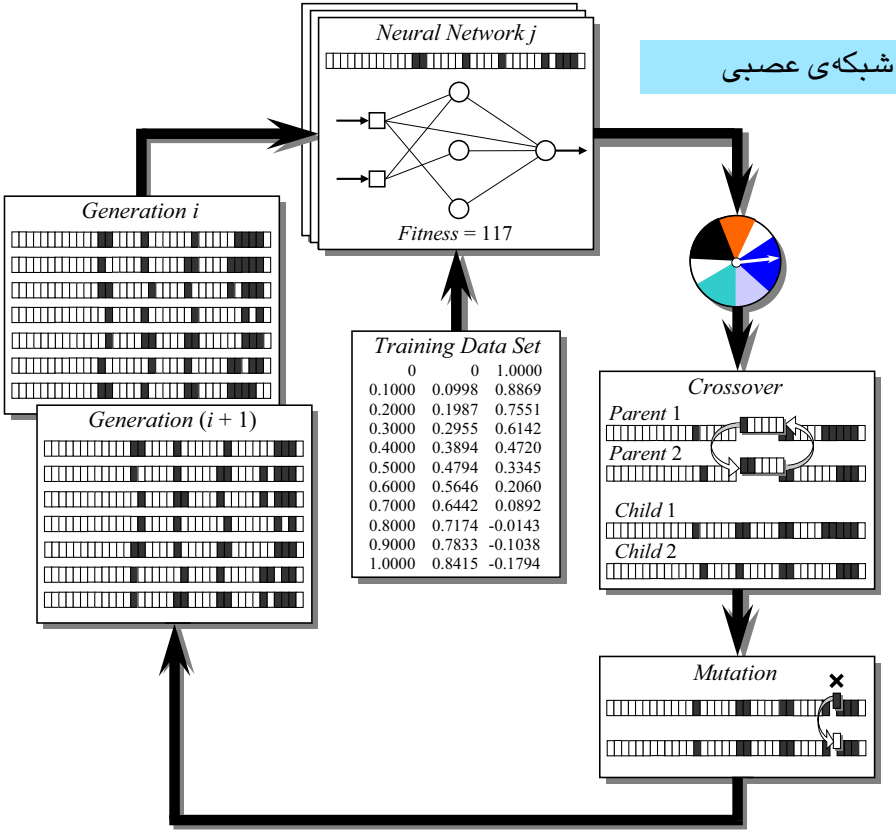
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0

شبکه‌های عصبی تطوری

انتخاب معماری بهینه‌ی یک شبکه‌ی عصبی با الگوریتم‌های ژنتیک

EVOLUTIONARY NEURAL NETWORKS

چرخه‌ی تطور توپولوژی یک شبکه‌ی عصبی



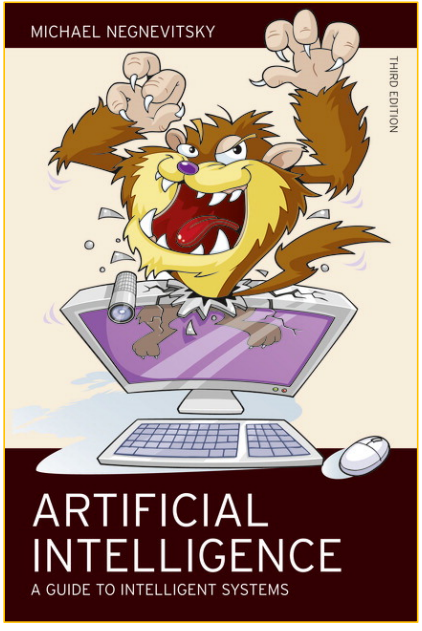
هوش مصنوعی

شبکه‌های عصبی تطوری

۳

منابع، مطالعه، تکلیف

منبع اصلی



Michael Negnevitsky,
Artificial Intelligence: A Guide to Intelligent Systems,
Pearson Education Canada, 2011.
Chapter 8 (8-5)

Hybrid intelligent systems 8

In which we consider the combination of expert systems, fuzzy logic, neural networks and evolutionary computation, and discuss the emergence of hybrid intelligent systems.

8.1 Introduction, or how to combine German mechanics with Italian love

In previous chapters, we considered several intelligent technologies, including probabilistic reasoning, fuzzy logic, neural networks and evolutionary computation. We discussed the strong and weak points of these technologies, and noticed that in many real-world applications we would need not only to acquire knowledge from various sources, but also to combine different intelligent technologies. The need for such a combination has led to the emergence of **hybrid intelligent systems**.

A hybrid intelligent system is one that combines at least two intelligent technologies. For example, combining a neural network with a fuzzy system results in a hybrid neuro-fuzzy system.

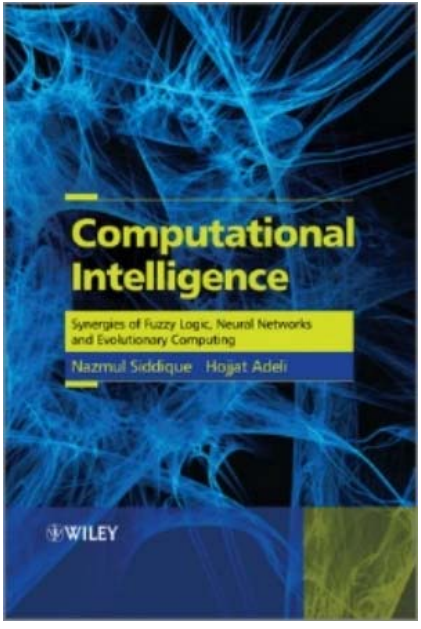
The combination of probabilistic reasoning, fuzzy logic, neural networks and evolutionary computation forms the core of **soft computing** (SC), an emerging approach to building hybrid intelligent systems capable of reasoning and learning in an uncertain and imprecise environment.

The potential of soft computing was first realised by Lotfi Zadeh, the 'father' of fuzzy logic. In March 1991, he established the Berkeley Initiative in Soft Computing. This group includes students, professors, employees of private and government organisations, and other individuals interested in soft computing. The rapid growth of the group suggests that the impact of soft computing on science and technology will be increasingly felt in coming years.

What do we mean by 'soft' computing?

While traditional or 'hard' computing uses **crisp values**, or numbers, soft computing deals with **soft values**, or fuzzy sets. Soft computing is capable of operating with uncertain, imprecise and incomplete information in a manner that reflects human thinking. In real life, humans normally use soft data

منبع دوم



Nazmul Siddique, Hojjat Adeli,
Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing,
John Wiley & Sons, 2013.
Chapter 9

9

Evolutionary Neural Networks

9.1 Introduction

Layered feedforward neural networks have become very popular, for several reasons: they have been found in practice to generalize well and there are well-known training algorithms such as Widrow–Hoff, backpropagation, Hebbean, winner-takes-all, Kohonen self-organizing map which can often find a good set of weights. Despite using minimal training sets, the learning time very often increases exponentially and they often cannot be constructed (Muehlenbein, 1990). When global minima are hidden among the local minima, the backpropagation (BP) algorithm can end up bouncing between local minima without much overall improvement, which leads to very slow training. BP is a method requiring the computation of the gradient of error with respect to weights, which again needs differentiability. As a result, BP cannot handle discontinuous optimality criteria or discontinuous node transfer functions. BP's speed and robustness are sensitive to parameters such as learning rate, momentum and acceleration constant, and the best parameters to use seem to vary from problem to problem (Badi and Homik, 1995). A method called momentum decreases BP's sensitivity to small details in the error surface. This helps the network avoid getting stuck in shallow minima which would prevent the network from finding a lower-error solution (Vogt *et al.*, 1988).

The automatic design of artificial neural networks has two basic sides: parametric learning and structural learning. In structural learning, both the architecture and parametric information must be learned through the process of training. Basically, we can consider three models of structural learning: constructive algorithms, destructive algorithms and evolutionary computation. Constructive algorithms (Gallant, 1993; Honavar and Uhr, 1993; Parekh *et al.*, 2000) start with a small network (usually a single neuron). This network is trained until it is unable to continue learning, then new components are added to the network. This process is repeated until a satisfactory solution is found. These methods are usually trapped in local minima (Angeline *et al.*, 1994) and tend to produce big networks. Destructive methods, also known as pruning algorithms (Reed, 1993), start with a big network that is able to learn but usually ends in over-fitting and try to remove the connections and nodes that are not useful. A major problem with pruning methods is the assignment of credit to structural components of the network in order to decide whether a connection or node must be removed. Both methods, constructive

Computational Intelligence: Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing, First Edition.
Nazmul Siddique and Hojjat Adeli.