DÜMF Mühendislik Dergisi
web: http://dergipark.gov.tr/dumf

MATLAB ile Levenberg-Marquardt algoritması tabanlı YSA uygulaması: Aylık yağış-akış modellemesi

Umut OKKAN*
Balıkesir Üniversitesi Müh. Fak İnşaat Müh. Böl. Çağış Kampüsü, Balıkesir
umutokkan@balikesir.edu.tr, Tel: (266) 612 11 94 (4203)

Zafer Ali SERBEŞ
Ege Üniversitesi, Ziraat Fak. Tarımsal Yapılar ve Sulama Böl, Bornova İzmir

Nuray GEDİK
Balıkesir Üniversitesi Müh. Fak İnşaat Müh. Böl. Çağış Kampüsü, Balıkesir

Geliş: 07.03.2017, Kabul Tarihi: 06.07.2017

Öz

Su kaynakları mühendisliğinde, aylık yağış-akış modellerinin önemi büyüktür. Yağış-akış ilişkilerinin belirlenmesinde kapalı kutu modellerinin kullanımı, deterministik modellere kıyasla zaman tasarrufu sağlaması, ihtiyaç duyulacak verinin nitelik ve nicelik olarak daha az olması ve daha düşük işlem hacmine sahip olmasından ötürü tercih edilmektedir. Bu kapsamda pek çok farklı metot kullanılarak yağış-akış ilişkileri ele alınmakta olup yapay sinir ağları (YSA) modelleri de adı geçen çalışmalarda önemli bir yer tutmaktadır. Literatürdeki örnekler incelendiğinde bu modellerin genellikle hazır paket programların arayüzleri ile kurulduğu görülmektedir. Çalışmada MATLAB programı kullanılarak YSA modeli uygulamasına yer verilmiştir. Optimizasyon ya da diğer bir deyişle YSA eğitim algoritması olarak Levenberg-Marquardt algoritmasından faydalanılmıştır. Uygulama örneği aylık-yağış modellemesi üzerinden kurgulanmış ve Gediz Havzası'ndaki Medar Çayı'na ait 1962-2005 dönemi verileri bu kapsamda değerlendirilmiştir. Hazırlanan modelin girdisi olarak alansal ortalama yağış (P_t) ve sıcaklık (T_t) değerleri kullanılmıştır. Çalışmada yağış ile akış arasındaki gecikmeler de hesaba katılmış ve sırasıyla bir (P_{t-1}), iki (P_{t-2}) ve üç (P_{t-3}) ay önceki alansal ortalama yağış verileri de girdi vektörüne eklenerek model performansları sınanmıştır. En iyi performansa sahip YSA modeli yapısı, farklı istatistiksel kriterler yardımıyla irdelenmiştir. Önerilen YSA model yapısının uzun dönem istatistiklerinin yanı sıra, aylık ve yıllık homojenlikler incelendiğinde de uygulama havzasını başarıyla temsil edebileceği kanısına varılmıştır. Elde edilen sonuçların yaında, LM algoritması tabanlı YSA modelinin MATLAB ortamında paylaşılan kodlarının yağış-ilişkisi belirleme uygulaması dışında farklı disiplinlerde çalışan araştırmacılara da yararlı olacağı düşünülmektedir.

Anahtar Kelimeler: *YSA, Levenberg-Marquardt algoritması, aylık yağış-akış modellemesi, MATLAB ile kodlama*

* Yazışmaların yapılacağı yazar

DOI:

Giriş

Yağış-akış modelleri, su kaynakları potansiyelinin ve baraj haznelerinin taşkın veya kuraklık gibi durumlardaki davranışlarının tahmininde, bunlara bağlı olarak ileriye dönük senaryoların oluşturulup gerekli önlem ve kararların alınmasında büyük önem taşıyan hidrolojik model çalışmalarının genelini oluşturmaktadır.

Fiziksel esaslı yayılı türden modeller, kavramsal türden parametrik-ortalanmış modeller ve kapalı (kara) kutu modelleri olarak üç grupta kullanılan bu modeller, temel anlamda akarsu havza sınırlarına düşen yağışın akarsu çıkışındaki akışa dönüştüğü sistemi inceler. Fiziksel esaslı yayılı türden modeller ve kavramsal türden modeller ile söz konusu ilişki incelenirken, fiziksel süreç kısmen veya detaylı bir biçimde dikkate alınmaktadır. Ancak bu tarz modellerin bazı öğelerinin belirsizlik taşıması, fazla girdiye ve/veya parametreye ihtiyaç duyması, modellerin karmaşık ve bazen de pratiklikten uzak olmasına neden olabilmektedir (Abbott ve Refsgaard, 1996).

Modeller, yağış ile akış arasındaki ilişkilerdeki bu karmaşıklık ve belirsizliklerden dolayı, olayın fiziksel yönünün ayrıntılarına girilmediği, matematiksel fonksiyonlar kullanılarak girdiler ve çıktılar arasındaki ilişkilerin ifade edilebildiği kapalı kutu modelleri biçiminde de hazırlanabilmektedir. Bu kapsamda geliştirilmiş regresyon analizi, eğri uydurma yaklaşımları ve stokastik yöntemler gibi klasik istatistik yöntemler sıklıkla kullanılmaktadır. Ayrıca, yapay zekâ tekniklerinin öneminin ve popülerliğinin artmasıyla klasik istatistiksel yöntemlerin yanı sıra yapay sinir ağları (YSA) yaklaşımları da kapalı kutu modeli olarak kullanılmaktadır. Su kaynakları mühendisliğinde birçok YSA uygulamasını görmek mümkündür. Örneğin, akarsu akım tahmininde YSA ile klasik istatistiksel yöntemler kıyaslanmıştır (Okkan ve Mollamahmutoğlu, 2010). Bunun dışında YSA, baraj akımlarının modellenmesinde (Razavi ve Araghinejad, 2009), sediment miktarının tahmininde (Tayfur, 2002), bölgesel taşkın frekansı analizinde (Jingyi ve Hall, 2004), yeraltı suyu problemlerinde (Shigidi ve Garcia, 2003) kullanılmıştır. Daha önceki çalışmalar incelendiğinde, genellikle ileri beslemeli-geriye yayılımlı algoritmaların kullanıldığı görülmektedir. Ancak bu tür algoritmaların yanı sıra, radyal tabanlı sinir ağları (Fernando ve Jayawardena, 1998) ve genelleştirilmiş regresyon sinir ağları (Cigizoglu, 2005) gibi algoritma türlerinin kullanımlarına da rastlanmaktadır.

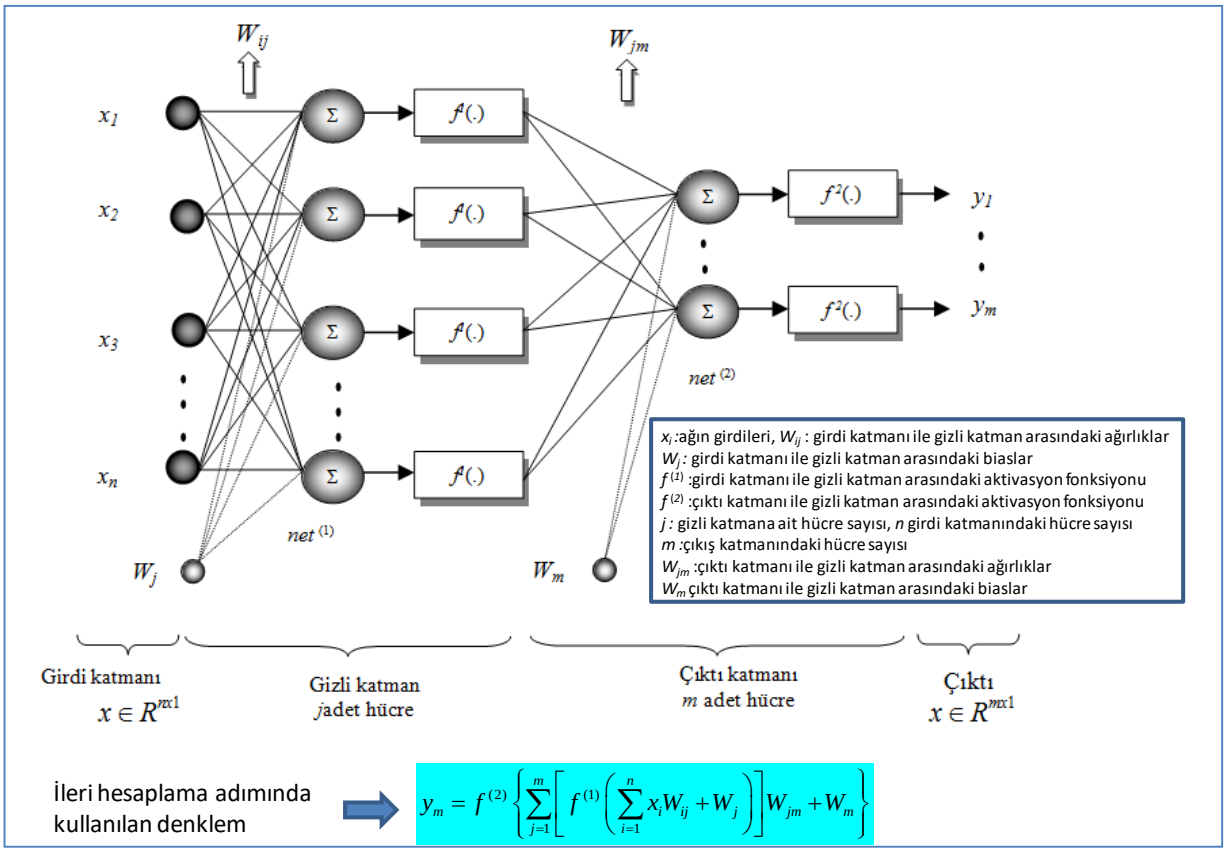
Bu çalışmada, Medar Çayı aylık yağış-akış ilişkisi ileri beslemeli geri yayılım algoritmasını kullanan YSA ile modellenmiştir. Bu kapsamda Levenberg-Marquardt (LM) optimizasyon algoritmasından faydalanılarak farklı meteorolojik girdiler altında model çalıştırılmıştır. Çalışmada yağış-akış modellemesi için MATLAB program kodları yazılmıştır. Geliştirilen bu kodlar makale içeriğinde paylaşılarak algoritmanın matematiksel işleyişi hakkında okuyuculara fayda sağlanması da hedeflenmiştir. Bu bağlamda, sunulan çalışma bilgisayar destekli bir uygulamaya yer vermesi bakımından özgün ve öğretici bir niteliğe sahiptir.

Materyal ve yöntem

YSA'nın yapısı ve MATLAB ortamında kodlanması

Sinir sisteminden esinlenerek geliştirilen matematiksel bir araç olan YSA, hücrelerin bir araya gelmesiyle oluşturulur ve çok katmanlı halde ifade edilir (Şekil 1). Çok katmanlı YSA'nın yapısı ile ilgili detaylara Okkan ve Mollamahmutoğlu (2010) çalışmasından erişilebilir. Şekil 1'de ilk ve son katman sırasıyla giriş ve çıkış katmanı, aradaki katman ise gizli (ara) katmandır.

Gizli katmandaki hücre sayısının (gkns) belirlenmesinde kullanılan ampirik ifadeler olmakla birlikte genellikle deneme-yanılma ile tespit edilmeye çalışılmaktadır. Bu çalışmada, tahmin edilmesi istenen tek bir değişken olduğundan (akarsu akış serisi) çıktı katmanında tek bir hücre yer almaktadır. Aktivasyon fonksiyonu olarak ise logaritmik sigmoid seçilmiştir. Bu fonksiyonlar MATLAB'ın fonksiyon kütüphanesinde de yer almakta olup ***logsig*** **(...)** komutu ile kullanılabilir. Çalışmada kullanılan LM algoritmasının işleyişi *ileriye doğru* ve *geriye doğru hesaplama* olmak üzere iki aşamada gerçekleşmektedir (Ham ve Kostanic, 2001). İleri doğru hesaplamada, girdi katmanındaki girdilerin ağa tanıtılması ile başlanmakta, daha sonra girdiler ağırlıklar ile çarpılarak gizli katmandaki her bir hücreye iletilmekte ve net girdi değerleri hesaplanmaktadır. Sonrasında bu net girdi değerleri belirlenen bir aktivasyon fonksiyonundan geçirilerek gizli katman hücrelerinin çıktısı elde edilmektedir.



Şekil 1. Tek gizli katmanlı bir yapar sinir ağının tipik yapısı

Bu benzer işlem sonraki katmanda da tekrarlanmakta ve çıktı katmanının çıktılarının hesaplanmasıyla ileri hesaplama işlemi sonlanmaktadır. İleri hesaplama aşamasında kullanılan denklem Şekil 1 üzerinde ve ayrıca Okkan ve Mollamahmutoğlu (2010)'da verilmiştir. İleri besleme (hesaplama) işlemini içeren ***forward.m*** MATLAB kodu Ek 1'de verilmektedir.

Geriye doğru hesaplama aşamasında ise çıktı ile hedef değerler arasındaki farkın karelerinin toplamının (SSE) azaltılması amaçlanmaktadır. Ek 1'de paylaşılan forward.m kodunda amaç (fitness/cost) fonksiyonu SSE, hata vektörü error, aktivasyon (transfer) fonksiyonları da sırasıyla tf1 ve tf2 değişkenleri olarak atanmıştır. Bu kod ile determinasyon katsayısı (*Rsq*) da hesaplanmaktadır.

YSA'da iterasyonlar başlamadan önce rastgele atanan ağın ağırlıkları, SSE arzu edilen sınırlara düşürülünceye kadar iteratif biçimde güncellenir. Ağırlıkların rastlantısal atanması da farklı şekillerle olabilmektedir. Örneğin ağırlıklar ve biaslar -1 ile 1 tanım aralığında uniform olarak atanabildiği gibi, xavier yöntemi olarak bilinen Glorot ve Bengio (2010) tarafından ortaya atılan yöntem kullanılarak da atanabilmektedir.

Çalışmada ağırlıkların ilk değerlerinin rastgele atanması ile ilgili MATLAB kodu (**initial_weights.m**) Ek 2'de verilmiştir.

Geriye yayılım ile ilgili hesaplar LM algoritmasını muhteva eden bir başka kod vasıtasıyla yapılmaktadır.

LM algoritması ve MATLAB ortamında kodlanması

Newton algoritmasının bir başka türü olan LM algoritması YSA ile birlikte ileri beslemeli-geriye yayılımlı algoritmalar arasında kullanılmaktadır (Hagan ve Menhaj, 1994; Ham ve Kostanic, 2001; Okkan ve Mollamahmutoğlu, 2010). LM algoritması hızlı yakınsaması, kendi içerisinde az parametre muhteva etmesi ve sadece birinci dereceden kısmi türevler ile işletilmesi bakımından avantajlı bir algoritmadır. LM algoritmasında SSE amaç fonksiyonunu minimize edecek optimum ağ ağırlıkları *W*, *k* iterasyon adımında Denklem 1 yardımıyla bulunabilmektedir.

$$\Delta W(k) = -[J_k^T J_k + \lambda_k I]^{-1} J_k^T e_k \quad (1.a)$$

$$W(k+1) = W(k) + \Delta W(k) \quad (1.b)$$

Denklem 1'de $\Delta W(k)$ *k*. iterasyonda ağ ağırlıklarındaki değişimi sembolize etmektedir. Burada *J* Jakobien matrisi ifade etmekte olup ağ hataları olan *e*'nin ağırlıklara göre birinci türevlerinden oluşmaktadır. λ Marquardt parametresini, *I* ise birim matrisi temsil etmektedir. Marquardt parametresi de LM algoritmasında iterasyonlar boyunca güncellenebilen bir parametredir. Eğer herhangi bir iterasyon adımında SSE azalıyorsa, λ bir sonraki iterasyon için belirli bir bozulma oranı (decay rate) ile çarpılarak kullanılmaktadır. Bu değer genelde 0.1 alınabilmektedir. Aksi durumda ise λ bir sonraki iterasyon için seçilen bozulma oranı değerine bölünerek kullanılmaktadır. Böylece her bir adımda ağın performansının iyileştirilmesi amaçlanmaktadır.

İterasyonlara başlamadan önce λ için de bir değer atanması gerekmektedir. λ'nın başlangıç değerinin 0.01 olarak atanması bu çalışmada uygun görülmüştür. Çalışmada kullanılan aktivasyon fonksiyonları her iki katmanda da sigmoid fonksiyonudur. Bu fonksiyonun özelliğinden dolayı girdi ve çıktı değerleri [0 1] tanım aralığında ölçeklendirildikten sonra ağa sunulmuştur. Fonksiyon sıralamasının sigmoid-doğrusal şeklinde olması koşulunda ise girdi ve çıktı değerlerinin ortalaması 0, standart sapması 1 olan bir seri haline getirilerek ağa sunulmasının daha doğru bir yaklaşım olacaktır. Bu nedenle çalışmada veri ölçekleme işlemi (normalizasyon) için iki adet opsiyon içeren bir kod yazılmıştır. Böylece okuyucular (kullanıcılar) farklı aktivasyon fonksiyonu denemeleri için alternatif ölçekleme tipi seçebileceklerdir (Ek 3a, ***normalizasyon.m***). Ağ eğitiminin, yani modelin iteratif işlemlerinin bitiminin ardından ölçeklendirilmiş değerleri elde edilen model çıktılarının orijinal birimlerine dönüştürülmesi gerekmektedir. Bu işlem de ters normalizasyon işlemi olarak adlandırılmış olup bu işleme ait kod Ek 3b'de (***ters_normalizasyon.m***) verilmiştir.

LM algoritmasında ağın ağırlık güncellemesinin Denklem 1 yardımı ile yapıldığı belirtilmişti. Bu denklemdeki en önemli kısım Jacobian matrisin hesaplandığı kısım olup, çalışmada Jacobian matris hesabı için sonlu farklar yönteminden yararlanılmıştır. Hata hesaplama fonksiyonu *e* olmak üzere ağırlıklara bağlı Jacobian hesabı Denklem 2'de verilmiştir. İleri fark ve merkezi fark gibi yaklaşımlar olmakla birlikte bu çalışmada ileri fark yaklaşımı esas alınmıştır. Hesabın yapıldığı MATLAB fonksiyonu ***jacob_sf.m*** ise Ek 4'de verilmiştir.

Ek 1 – Ek 3'de verilen kodlarını çağıran LM algoritması esaslı YSA modeli kodu Ek 5'de verilmiştir. Bu kod YSA parametrelerinin kontrol edilip, eğitim işleminin gerçekleştirildiği asıl koddur.

$$\frac{\partial e_i}{\partial W_j} \approx \frac{e_i(W_1, W2, ..., W_j + \Delta x, ..., W_n) - e_i(W_1, W2, ..., W_j, ..., W_n)}{\Delta x} \quad (2)$$

Uygulama

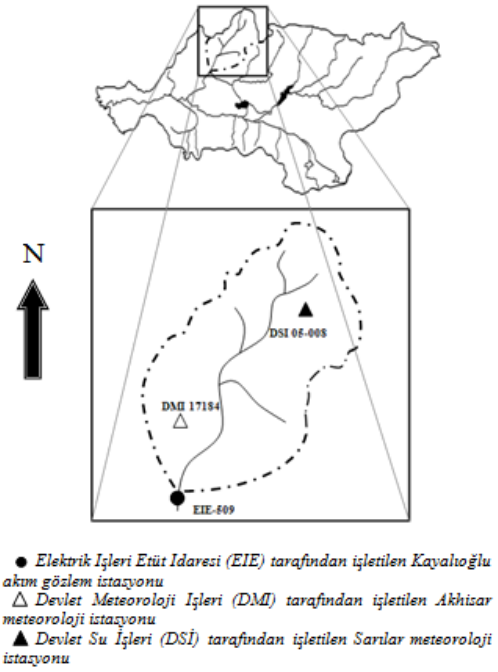
Çalışma Alanı

Aylık bir yağış-akış modeli oluşturulması için yapılan bu çalışmada, Gediz Havzası sınırlarında bulunan ve yaklaşık olarak 900 km^2 drenaj alanına sahip Medar Çayı dikkate alınmıştır. LM optimizasyon algoritması ile eğitilecek (kalibre edilecek) YSA modelinin girdisi olarak aylık zaman ölçeğinde alansal ortalama yağış ve sıcaklık ile yağış ve sonrasında oluşan akım arasındaki gecikmelerden yararlanılmıştır. Bu amaçla çapraz korelasyonlar kullanılarak, önceki alansal ortalama yağış verileri de girdi setine eklenmiş ve model performansları sınanmıştır.

Modellenmesi istenen hedef değerler (*target*), Medar Çayı'nın akımlarını temsil ettiği düşünülen ve EİE tarafından işletilen 509 numaralı Kayalıoğlu akım gözlem istasyonunda gözlenen akımlardır. Çalışmada istasyonun 01.10.1961-01.09.2005 döneminde (1962-2005 su yılları) ölçülen verileri esas alınmıştır.

DMİ Akhisar (17184) ve DSİ Sarılar (05-008) istasyonları ise yağış gözlemleri bakımından havzayı temsil edebilecek istasyonlar olarak belirlenmiştir. Söz konusu yağış istasyonlarının değerlerinden Thiessen yardımı ile alansal ortalama yağışlar elde edilmiştir. Ayrıca modellerde kullanılan aylık ortalama sıcaklık değerleri uzun dönem eksiksiz veri sağlanabilen DMİ Akhisar (17184) istasyonundan temin edilmiştir.

Gediz Havzası ve Medar Çayı'nın havza üzerindeki konumu ve söz konusu meteoroloji istasyonları Şekil 2'de gösterilmiştir.



Şekil 2. Medar Çayı ve istasyonların Gediz Havzası üzerindeki konumları

Modelin uygulanması

Bu çalışmada 01.10.1961-01.09.2005 dönemine ait akım verileri esas alındığından, 1, 2 ve 3 ay önceki yağış verileri sırasıyla 01.09.1961, 01.08.1961 ve 01.07.1961 tarihlerinden başlanarak derlenmiştir. İlk model aylık alansal ortalama yağış (P_t) ve sıcaklık (T_t) girdileri ile kurulmuştur. Daha sonra sırasıyla 1, 2 ve 3 ay önceki yağış ((P_{t-1}), (P_{t-2}), (P_{t-3})) girdileri de modele ilave edilmiştir.

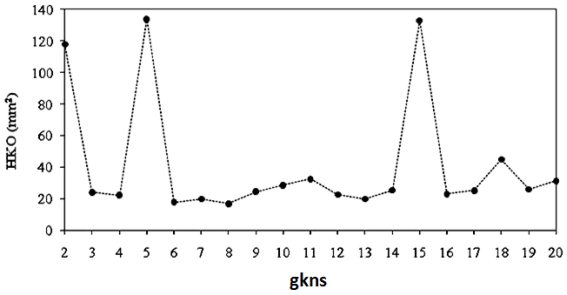
Hidroloji literatüründeki çalışmalara göre, eğitime ayrılacak veri adedi toplam adedin %50'si ila %80'si arasında seçilebilir. Ayrıca test verilerinin rastgele seçilmesi yapay zeka modellemelerinde tercih edilen bir teknik olmasına karşın, hidrolojik modelleme çalışmalarında zaman serileri ile çalışıldığından

ve akarsu akımlarının mevsimsel içsel bağımlılık özelliğinden dolayı bu tekniği kullanmak bazen hatalı kavramsallaştırmaya neden olabilmektedir. Bu bakımdan mevcut periyodu su takvimini esas alarak (Ekim'den Eylül'e kadar olan süreç) belli bir oranda bölümlendirmek daha tutarlıdır. Bu çalışmada modeller veri setinin % 50'si (01.10.1961-01.09.1983) ile kurulmuş (eğitilmiş), geri kalan % 50'si (01.10.1983-01.09.2005) ile test edilmiştir. Ağın eğitiminde, uygunluk fonksiyonunu minimum yapan gizli katmandaki hücre sayısı (gkns) deneme-yanılma yoluyla belirlenmiştir. Çalışmada ayrıca, ağın aşırı öğrenmesi sonucu genelleme yeteneğini kaybetmesi ve buna bağlı olarak test dönemindeki performans değerinin düşmesini önlemek için eğitim aşamasında optimum sayıda iterasyon yaptırılmıştır. Modeller hazırlandıktan sonra performansları hata kareler ortalaması (HKO) ve determinasyon katsayısı (R^2) kullanılarak kıyaslanmıştır. Farklı girdi kombinasyonları kullanılarak hazırlanan modellere ait istatistiksel karşılaştırmalar Tablo 1'de verilmiştir. Sonuçlar incelendiğinde, aylık alansal ortalama yağış ve sıcaklık değişkenlerine ilaveten sırasıyla 1, 2 ve 3 ay gecikmeli alansal ortalama yağış girdilerini de barındıran 4. model yapısının oldukça başarılı olduğu göze çarpmaktadır.

Tablo 1. Çalışmada kullanılan modellere ait YSA mimarileri ve model performansları

Girdiler	Ağ Yapısı			R^2 (%)		HKO (mm^2)	
	n	*j*	*m*	Eğitim	Test	Eğitim	Test
P_t , T_t	2	3	1	60.24	49.09	108.73	62.37
P_t , T_t, P_{t-1}	3	6	1	79.61	63.04	55.79	41.85
P_t , T_t, P_{t-1},P_{t-2}	4	5	1	85.79	73.34	38.89	29.68
P_t , T_t, P_{t-1}, P_{t-2},P_{t-3}	5	8	1	94.45	84.43	15.41	16.83

Şekil 3'de, 2-20 arasında denenen gizli katmandaki hücre sayılarının (gkns: *j*) 4. modelin test dönemindeki HKO performansları gösterilmektedir. Şekil 3'teki bulgular gkns değerinin neden deneme-yanılma ile belirlendiğinin de bir kanıtıdır. Bulgulara göre, en uygun gkns değerinin 8 olduğu görülmektedir. Bu model YSA (5, 8, 1) şeklinde kısaltılmış olup modelin eğitim ve test dönemlerine ait istatistikler Tablo 2'de verilmiştir. Bu tabloda verilen ortalama (μ), standart sapma (σ) ve çarpıklık katsayısı (C_s) gibi temel istatistikleri eğitim dönemi performansının oldukça başarılı olduğunu göstermektedir. Ortalama değerlere bakıldığında modelin eğitildiği dönemin akışlar bakımından test dönemine kıyasla %43 daha sulak olduğu görülmektedir. Bu gözlemin bir sonucu olarak eğitilen model test döneminde yaklaşık 1.1 mm daha fazla tahmin üretmiştir. Ancak elde edilen istatistikler model sonuçlarının test döneminde de başarılı olduğunu göstermektedir.



Şekil 3. gkns'nin 4.model kombinasyonunda test dönemindeki HKO performansları

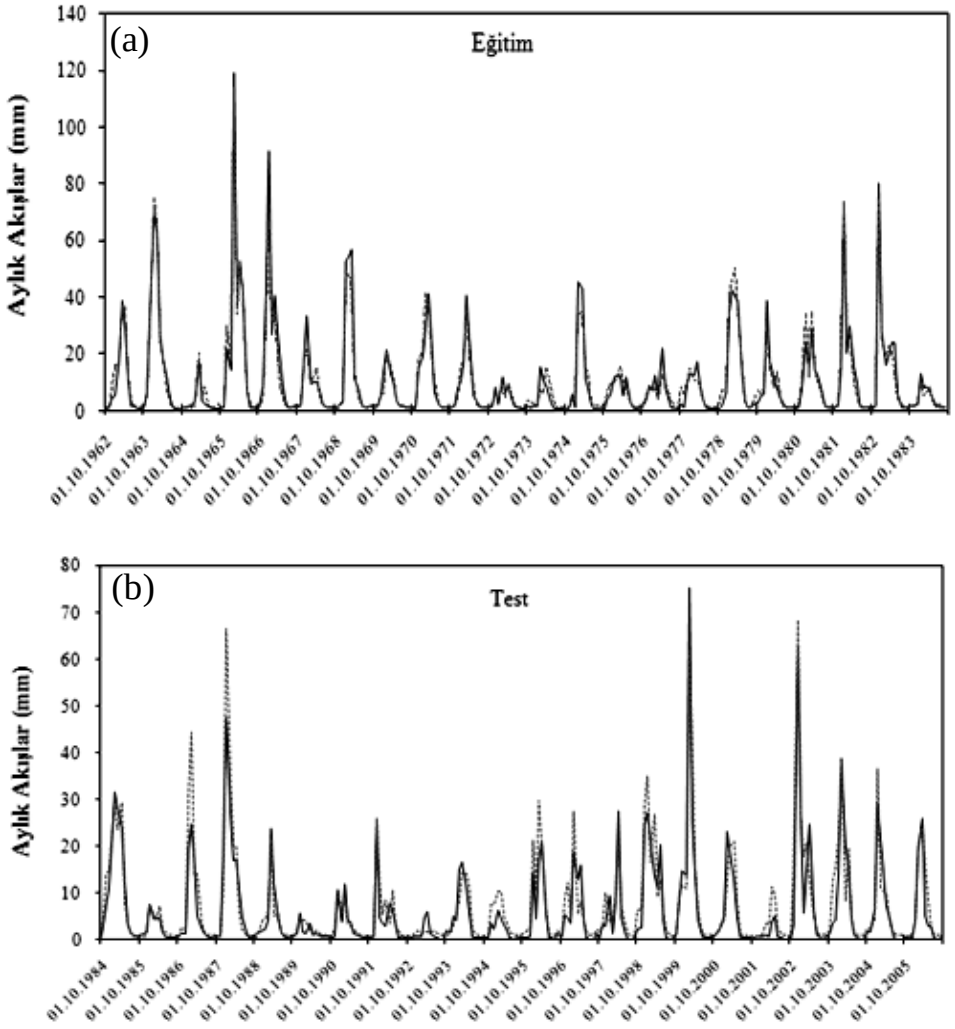
Tablo 2. YSA (5, 8, 1) modelinin eğitim ve test dönemlerine ait istatistikler

		μ *mm*	σ *mm*	C_s -	*Mak mm*	*Min mm*
Eğitim	Gözlenen	10.69	16.57	2.91	119.00	0.37
	YSA	10.56	15.62	2.81	114.01	0.37
Test	Gözlenen	6.13	9.84	3.08	75.30	0.01
	YSA	7.25	10.60	2.84	68.45	0.33

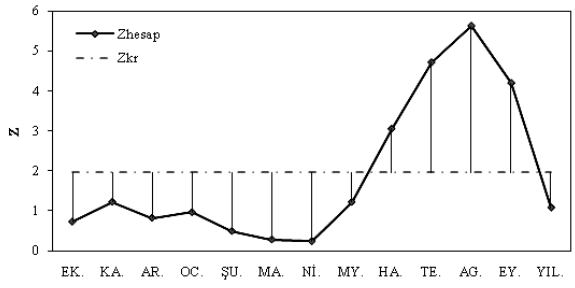
YSA (5,8,1) modelinin eğitim ve test dönemine ait akış-zaman grafikleri de bu uyumu desteklemektedir (Şekil 4).

Çalışmada ayrıca non-parametrik Mann-Whitney U testi (M-W) kullanılarak aylık akım tahminlerinin homojenlikleri sınanmıştır. Söz konusu test tüm aylara ve yıllık toplam akımlara uygulanmış ve test dönemine ait bulgular Şekil 5'de özetlenmiştir. Belirlenen z_{hesap} değeri, %5 önem düzeyindeki kritik tablo değeri olan z_{kr} ile karşılaştırılmaktadır. Eğer $z_{hesap}<z_{kr}=1.96$

durumunu sağlanıyorsa gözlenen akımlar ile modellenen akımlar arasında istatistiksel bir farkın olmadığı söylenebilmektedir.



Şekil 4. YSA (5,8,1) modelinin (a) eğitim ve (b) test dönemine ait akış-zaman grafikleri (kesikli çizgi ile verilen seri model tahminlerini temsil etmektedir)



Şekil 5. Test dönemi YSA sonuçları ile gözlenen aylık akışların M-W istatistikleri

Sonuçlar ve Tartışma

Bu çalışmada, Medar Çayı aylık zaman ölçekli yağış ile akış arasındaki ilişki, alansal yağış ile akarsu çıkışında oluşacak akış arasındaki gecikmeler de kullanılarak YSA ile modellenmiştir. Bu aşamada LM algoritmasından yararlanılarak, hazırlanan MATLAB kodu ile amaç fonksiyonunun minimum değerine oldukça çabuk yakınsaması sağlanmış ve güvenilir sonuçlar elde edilmesi mümkün kılınmıştır.

Çalışmada farklı girdi kombinasyonları kullanılmış, aylık alansal ortalama yağış, sıcaklık ve 3 aya kadar gecikmeli alansal ortalama yağış girdileri ile hazırlanan modelin (YSA(5,8,1) modeli) oldukça başarılı olduğu tespit edilmiştir. Gerek uzun dönem istatistikleri ve temel istatistikler, gerekse debi-zaman grafiklerindeki uyum bu görüşü kanıtlar niteliktedir.

Uzun dönem istatistiklerin yanı sıra, mevsimsel homojenlikler M-W testi üzerinden incelendiğinde de Haziran, Temmuz, Ağustos ve Eylül ayları dışındaki tüm aylık ve yıllık akış tahminlerinin homojen olduğu görülmektedir.

Çalışmada kullanılan girdilere ilave olarak önceki akım verileri (Q_{t-1}, Q_{t-2},.., Q_{t-k}) ile akımların içsel bağımlılık etkisi de hesaba katılarak modellerin performansları arttırılabilir (Okkan ve Mahmutoğlu, 2010). Fakat optimum sayıda girdi vasıtası ile akışı kayda değer bir doğrulukla modelleme imkanı sağlanmışsa, aşırı girdi kullanımı ile modelin genelleme yeteneği yitirilebileceğinden ilave tahminleyici değişken kullanmanın bir gereği bulunmamaktadır. Bu nedenle çalışmada kullanılan girdilerin Medar Çayı aylık akışlarını yeterli ölçüde temsil ettiği düşünülmektedir.

Ayrıca, LM algoritması tabanlı YSA modelinin MATLAB ortamında paylaşılan kodlarının yağış-akış ilişkisi belirleme çalışmalarına ilaveten farklı disiplinlerde çalışan araştırmacılara da yararlı olacağı düşünülmektedir.

EK 1: İleri hesaplama kodu (forward.m)

```
%Dr. Umut OKKAN-Ocak 2017
%İLERİ HESAPLAMA

function [error, weights_ilk, bias_ilk, weights_son, bias_son,...
    net_ilk, hidden_out, net_son, Yhat, Rsq, fitness]=...
    forward(girdi,hedef, gkns, W,tf1, tf2)

[veri_adedi, girdi_adedi]=size(girdi);

%weights_ilk ilk katmandaki (girdi-ara katman arası)ağırlıklar,
%bias_ilk ilk katmandaki (girdi-ara katman arası)bias değerleridir.

%weights_son son katmandaki (ara-çıktı katmanı arası )ağırlıklar,
%bias_son son katmandaki (ara-çıktı katmanı arası )biaslardır.

%Ağırlıklar atanıyor

Wd=W(1,1:girdi_adedi*gkns); %önce ilk katman

for i=1:gkns
weights_ilk(i,:)=Wd(1,1:girdi_adedi);
Wd=Wd(1,girdi_adedi+1:end);
end

bias_ilk=(W(girdi_adedi*gkns+1:girdi_adedi*gkns+gkns)).';

weights_son=W(girdi_adedi*gkns+gkns+1: girdi_adedi*gkns + gkns + gkns).';

bias_son=W(girdi_adedi*gkns + gkns + gkns + 1);

%İLERİ HESAPLAMA

%net hesabı
for i=1:gkns
net_ilk(:,i)=(weights_ilk(i,:)*girdi').'+bias_ilk(i,1)*ones(veri_adedi,1);
end

%gizli katman çıktısı hesaplanıyor
for i=1:gkns
hidden_out(:,i)=tf1(net_ilk(:,i));
end

XX=[hidden_out, ones(length(hidden_out),1)];
weights_vector_son= [weights_son;bias_son];

%çıktı katmanından çıkan değerler
net_son =(XX*weights_vector_son);
Yhat = tf2(net_son);

%çıktı katmanı için performans hesabı
error= hedef-Yhat;
SSE=sum(error.^2);
R_cor=corr(hedef,Yhat);
Rsq=(R_cor).^2;

%AMAÇ FONKSİYONU (COST/FITNESS FONK)
fitness=SSE;
end
```

Ek 2: Başlangıç ağırlıklarını atama kodu (initial_weights.m)

```
%Dr. Umut OKKAN-Ocak 2017
function W_initial = initial_weights...
    (yaklasim, girdi_adedi, gkns,epsilon_alt_deger, epsilon_ust_deger)

switch yaklasim

case 'xavier'
%Xavier başlangıç ağırlık atama yöntemi
nn=1/sqrt(girdi_adedi*gkns);
W1_initial=unifrnd(-nn,nn,1,girdi_adedi*gkns);
b1_initial=zeros(1,gkns);
W2_initial=unifrnd(-1/sqrt(gkns),1/sqrt(gkns),1,gkns);
b2_initial=0;

W_initial=[W1_initial,b1_initial,W2_initial, b2_initial];

case 'uniform'
% uniform random ağırlık atama
toplam_agirlik_adedi=(girdi_adedi*gkns) + (gkns) + (gkns) + 1;
Lower_bound=[epsilon_alt_deger*ones(1,toplam_agirlik_adedi)];
Upper_bound=[epsilon_ust_deger*ones(1,toplam_agirlik_adedi)];

for s=1:toplam_agirlik_adedi
  W_initial(1, s)=Lower_bound(s)+rand*(Upper_bound(s)-Lower_bound(s));
end

end
end
```

Ek 3: Verilerin (a) normalizasyonu ve (b) çıktının ters normalizasyonunda kullanılan kodlar

(a)

```
function [pn,ps,tn,ts] = normalizasyon(normalizasyon_no, inputs, target)

if normalizasyon_no==1;

[pn,ps] = mapstd(inputs.');
[tn,ts] = mapstd(target.');

else
[pn,ps] = mapminmax(inputs.',0,1);
[tn,ts] = mapminmax(target.',0,1);

end
```

(b)

```
function [cikti] = ters_normalizasyon(normalizasyon_no, Yhat, ts)

if normalizasyon_no==1;

cikti = mapstd('reverse',Yhat,ts);

else

cikti = mapminmax('reverse',Yhat,ts);

end
```

Ek 4: Sonlu farklar ile Jacobian hesabının yapıldığı kod

```
%Dr. Umut OKKAN-OCAK 2017
%SONLU FARKLAR İLE JACOBIAN HESABI

function [Jacobian_matris]=...
    jacob_sf(x,delta_x,train_girdi,train_hedef, gkns, tf1, tf2)

[m,n] = size(x);
for j = 1:n
 xx = x;
 xx(j) = x(j) + delta_x;
 Jacobian_matris(:,j) =...
(forward(train_girdi,train_hedef, gkns, xx, tf1, tf2)...
-forward(train_girdi,train_hedef, gkns, x, tf1, tf2))/delta_x;
end

end
```

Ek 5: LM algoritması içeren YSA kodu

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% LM optimizasyon algoritması ile MLP modeli eğitimi
% Dr.Umut OKKAN-Ocak 2017
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc;clear all;close all

load data;
%inputs ve target dosyalarını içerir (sütün vektörü olarak)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TANIMLANACAK DEĞİŞKENLER
tp=0.5; % tp:eğitime ayrılacak veri büyüklüğü oranı
lamda=0.01; % başlangıç lamda değeri
decay=0.1; %bozulma oranı
gkns=8;%gizli katmandaki nöron sayısı (gkns)
iter_max=30; %maksimum iterasyon adedi

%transfer fonksiyonları
tf1=@logsig;
tf2=@logsig;

normalizasyon_no=2; % 1 veya 2
%1..>mapstd  (tf1=logsig, tf2=purelin durumunda tercih edilebilir)
%2..>mapminmax (tf1=tf2=logsig durumunda tercih edilebilir)

baslangic_agirlik_atama_tipi= 'xavier'; % 'uniform' or 'xavier'

%uniform initilization yaklaşımında başlangıç ağırlıklarının limit değerleri
epsilon_min=-1; epsilon_max=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[veri_adedi, girdi_adedi]=size(inputs);

% Agın egitilmesinde kullanılacak olan veriler normalize ediliyor.
[pn,ps,tn,ts] = normalizasyon(normalizasyon_no, inputs, target);

%eğitim ve test kısmında kullanılacak veriler ayrılıyor.
iitr=1:round(veri_adedi*(tp));  % eğitim kısmı
iitst=round(veri_adedi*(tp))+1:veri_adedi; % test kısmı

% Training (egitim) verileri hazırlanıyor
train_girdi=(pn(:,iitr)).';
train_hedef=(tn(:,iitr)).';

% Test verileri hazırlanıyor
test_girdi=(pn(:,iitst)).';
test_hedef=(tn(:,iitst)).';

%Başlangıçta ağ ağırlıkları rastgele atanıyor
toplam_agirlik_adedi=(girdi_adedi*gkns) + (gkns) + (gkns) + 1;

W_initial = initial_weights...
(baslangic_agirlik_atama_tipi, girdi_adedi, gkns,epsilon_min, epsilon_max);

W=W_initial;

for iter=1:iter_max;
%% LEVENBERG-MARQUARDT ALGORİTMASI

[error_train,weights_ilk, bias_ilk, weights_son, bias_son,net_ilk_train,...
hidden_out_train,net_son_train, Yhat_train,  Rsq_train, fitness_train]...
=forward(train_girdi,train_hedef, gkns, W, tf1, tf2);

%SONLU FARKLAR İLE JACOBIAN HESABI YAPILIYOR

J=zeros(length(train_girdi), toplam_agirlik_adedi); % J: Jacobian

delta_x=10^-5; %sonlu farklarda kullanılan adım hassasiyet miktarı

J=jacob_sf(W,delta_x,train_girdi,train_hedef, gkns, tf1, tf2);

H=J'*J; %Hessian matrisin yaklaşık çözümü

H_lm=H+(lamda*eye(toplam_agirlik_adedi,toplam_agirlik_adedi));

dW=-inv(H_lm)*(J'*error_train);
W_lm=W+dW.';

[error_train_lm,weights_ilk_lm, bias_ilk_lm, weights_son_lm,...
    bias_son_lm, net_ilk_train_lm, hidden_out_train_lm,net_son_train_lm,...
    Yhat_train_lm,  Rsq_train_lm, fitness_train_lm]=...
forward(train_girdi,train_hedef, gkns, W_lm, tf1, tf2);

if fitness_train_lm<fitness_train
    lamda=lamda*decay;
    W=W_lm;
    fitness_train=fitness_train_lm;
else
    lamda=lamda/decay;
end

%iterasyonlara bağlı olarak  W ve fitness değerleri
W_listed(iter,:)=W;
fitness_listed(iter,:)=fitness_train;
lamda_listed(iter,:)=lamda;
end %iterasyon için end
```

Ek 5 (devamı)

```
%% NİHAİ SONUÇLAR ELDE EDİLİYOR
W_updated=W_listed(iter,:);

[error_train,weights_ilk, bias_ilk, weights_son, bias_son, ...
    net_ilk_train, hidden_out_train, net_son_train, Yhat_train,  ...
    Rsq_train, fitness_train]=forward(train_girdi,train_hedef, ...
    gkns, W_updated, tf1, tf2);

[error_test,weights_ilk, bias_ilk, weights_son, bias_son, ...
    net_ilk_test, hidden_out_test, net_son_test, Yhat_test,  ...
    Rsq_test, fitness_test]=forward(test_girdi,test_hedef, ...
    gkns, W_updated, tf1, tf2);

%gerçek ölçekli eğitim ve test dönemi çıktısı değerleri
egitim_ciktisi = ters_normalizasyon(normalizasyon_no,Yhat_train,ts);
egitim_gozlenen = ters_normalizasyon(normalizasyon_no,train_hedef,ts);

error_egitim_real_scaled=egitim_gozlenen-egitim_ciktisi;
MSE_egitim_real_scaled=mse(error_egitim_real_scaled);
RMSE_egitim_real_scaled=sqrt(MSE_egitim_real_scaled);

test_ciktisi = ters_normalizasyon(normalizasyon_no,Yhat_test,ts);
test_gozlenen = ters_normalizasyon(normalizasyon_no,test_hedef,ts);

error_test_real_scaled=test_gozlenen-test_ciktisi;
MSE_test_real_scaled=mse(error_test_real_scaled);
RMSE_test_real_scaled=sqrt(MSE_test_real_scaled);

RMSE_real_scaled=[RMSE_egitim_real_scaled, RMSE_test_real_scaled];
R_sq=[Rsq_train, Rsq_test];

%Uygunluk fonksiyonunun ve lamdanın iterasyona bağlı değişimi çizdiriliyor
figure (1)
plot(fitness_listed(1:iter,1)); figure(gcf)
xlabel('Iterasyon');
ylabel('Uygunluk Değerleri');
title('LM ile MLP modeli eğitimi');
grid;

figure (2)
plot(lamda_listed(1:iter,1)); figure(gcf)
xlabel('Iterasyon');
ylabel('Lamda Değerleri');
grid;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SAÇILIM GRAFİKLERİ
figure (3)
postreg(egitim_ciktisi,egitim_gozlenen);
xlabel('Eğitim Gözlenen');
ylabel('Eğitim Tahmin');

figure (4)
postreg(test_ciktisi,test_gozlenen);
xlabel('Test Gözlenen');
ylabel('Test Tahmin');
```

Kaynaklar

Abbott, M.B. ve Refsgaard, J.C., (1996). Distributed hydrological modeling, *Kluwer Academic Publishers*, Dordrecht.

Cigizoglu H.K., (2005). Application of the generalized regression neural networks to ıntermittent flow forecasting and estimation, *Journal of Hydrologic Engineering*, 10, 4, 336-341.

Fernando, D.A.K. ve Jayawardena, A.W., (1998). Runoff forecasting using RBF networks with OLS algorithm, *Journal of Hydrologic Engineering*, 3, 3, 203-209.

Glorot, X. ve Bengio, Y., (2010). Understanding th difficulty of training deep feedforward neural networks. Appearing in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics. Journal of Machine Learning Research W&CP 9:249-256.

Hagan, M.T. ve Menhaj, M.B., (1994). Training feed forward techniques with the Marquardt Algorithm, *IEEE Transactions on Neural Networks*, 5, 6, 989–993.

Ham, F. ve Kostanic, I., (2001). *Principles of Neurocomputing for Science and Engineering*, Macgraw-Hill. USA.

Jingyi, Z. ve Hall, M. J., (2004). Regional flood frequency analysis for the Gan-Ming River basin in China, *Journal of Hydrology*, 296, 98–117.

Okkan, U. ve Mollamahmutoğlu, A., (2010). Yiğitler Çayı günlük akımlarının yapay sinir ağları ve regresyon analizi ile modellenmesi, *Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 23, 33-48.

Razavi, S. ve Araghinejad, S., (2009). Reservoir inflow modeling using temporal neural networks with forgetting factor approach, *Water Resources Management*, 23, 39–55.

Shigidi, A. ve Garcia, A.L., (2003). Parameter estimation in groundwater hydrology using artificial neural networks, *Journal Of Computing In Civil Engineering*, 17, 4, 281-289.

Tayfur, G., (2002). Artificial neural networks for sheet sediment transport, *Hydrological Sciences Journal*, 47, 6, 879–892.

Levenberg-Marquardt algorithm based ANN application using MATLAB: A monthly rainfall-runoff modeling example

Extended abstract

Rainfall-runoff models are utilized in the design of hydraulic structures, in the prediction of dam reservoir behavior under extreme conditions such as flood or drought, and to prepare prospective scenarios and take necessary precautions and decisions depending on projections.

These models, which are assessed in three groups as physically distributed type models, conceptual parametric-lumped models and black box models, basically examine the system in which the rainfall is converted into the flow at the stream outlet for a basin. The physical process can be taken into account in part or in detail, from models based on physically distributed models and models from conceptual ones. However, both the uncertainty of some items of such models and the need for extra input and/or parameter can make the models complex and impractical.

Models can also be prepared in the form of black-box models, where the details of the physical phenomenon are not considered owing to the mentioned complexities and uncertainties existed in the non-linear relation between rainfall and runoff mechanism, and thus the relationships between inputs and output, which is runoff variable, can be expressed as some mathematical functions. In this context, classical statistical methods such as regression analysis, curve fitting approaches and stochastic methods are frequently used. In addition, artificial neural network (ANN) approaches are also used as black box models with increasing popularity of artificial intelligence techniques.

It is possible to come across many ANN applications in the water resources engineering content. In the study, the monthly rainfall-runoff relation of Medar Stream located at Gediz Basin/Turkey is modeled by ANN based on a feed forward-back propagation algorithm. In this context, the model was trained under different meteorological inputs by using the Levenberg-Marquardt (LM) optimization algorithm, which is also a feed forward-back propagation algorithm. As a result of various input and model structure trials, it was concluded that the calibrated ANN model structure can represent the rainfall-runoff relation of basin successfully when the long term performance statistics and seasonal homogeneities are examined.

When the literature about ANN implementations is examined, it is seen that researchers generally use MATLAB's ANN toolbox interface. In the prepared study, the LM based model was prepared in the MATLAB environment but not in the ANN toolbox. The developed MATLAB codes are also shared in the article content, and it is aimed to benefit the readers about the mathematical procedure of the algorithm. In this context, the presented work has a unique and instructive quality in terms of providing a computer-aided application.

Keywords: *ANN, Levenberg-Marquardt algorithm, monthly rainfall-runoff modeling, coding via MATLAB*