**Yapay Sinir Ağları**

**4.Ödev**

**16.01.2021**

**Mehmet Şerbetçioğlu -- 040160056**

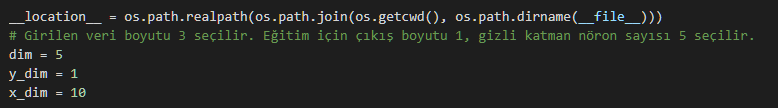
**Ahmet Hulusi Tarhan -- 040170738**

İkinci ödevde verilen fonksiyon üzerinde yaklaşacak Elman Ağı oluşturulacaktır.

Yaklaşılacak fonksiyon:



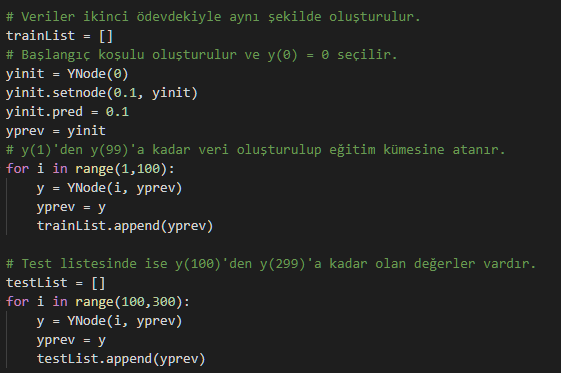
Fonksiyonun argümanları arasında ve bulunmaktadır. gürültü olup küçük rastgele bir değer atanır. Eğitilecek ağda girişler seçilir. Başlangıçta ağ boyutları belirlenir.



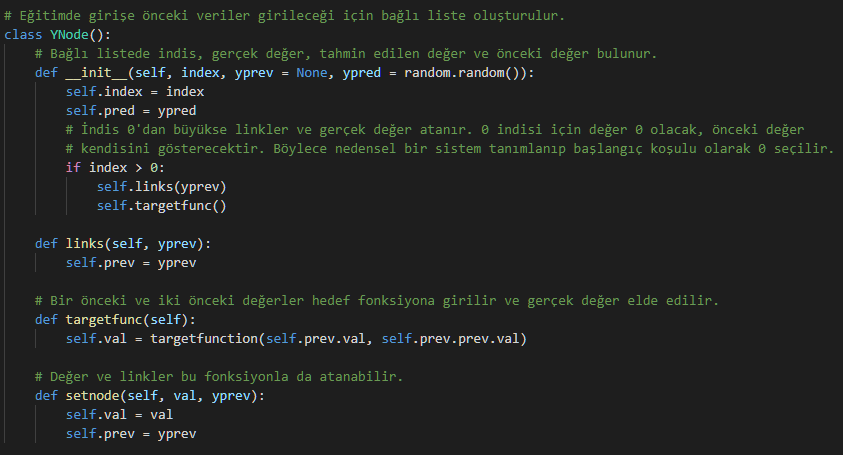
Burada “dim” girişteki verinin boyutu, “y\_dim” ağ çıkışının boyutu ve “x\_dim” gizli katmandaki nöron sayısıdır.

**Verilerin Saklanması:**

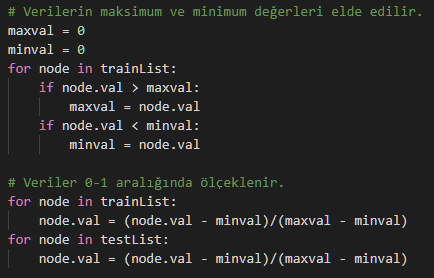
Bu aşamanın ardından ikinci ödevdekiyle aynı şekilde test ve eğitim kümeleri oluşturulur. Eğitim kümesinde değerleri, test kümesinde ise değerleri bulunur.



Oluşturulan veriler düğümler şeklinde saklanır. Bu düğümlerde “yd” değeri, düğümün “k” indisi ve ağın veri için tahmin ettiği “y” değeri bulunmaktadır. Veri kümeleri oluşturuluren başlangıç değeri seçilir ve kendisine bağlanır. Ek olarak, düğümler içinde bir önceki düğümle bağlantı bulunmaktadır. Bu şekilde girişteki verisi her düğüm için çekilebilmektedir.

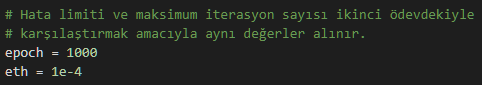


Veri kümeleri oluşturulduktan sonra eğitim kümesi aralığında ölçeklendirilir. Maksimum ve minimum değerleri daha sonra geri ölçeklemede kullanılmak üzere saklanır.



**Eğitim Hazırlıkları:**

Öncelikle eğitimin maksimum iterasyon sayısı ve hata limiti seçilir. Bu değerler ikinci ödevle karşılaştırmak adına, ikinci ödevde kullanılan değerlerle aynı seçilir. Bu değerler olarak seçilir.

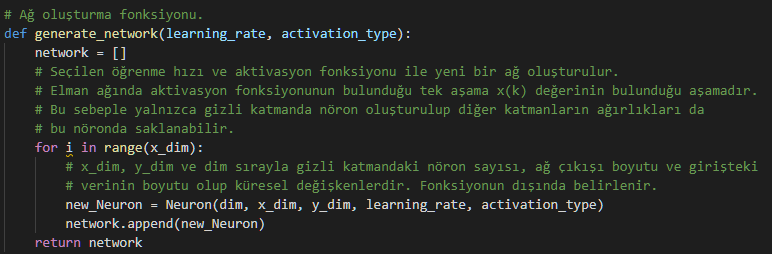


Eğitim için kullanılacak ağ oluşturulur. Ağın öğrenme hızı 0.1 olup kullanılacak aktivasyon fonksiyonu sigmoid olacaktır.

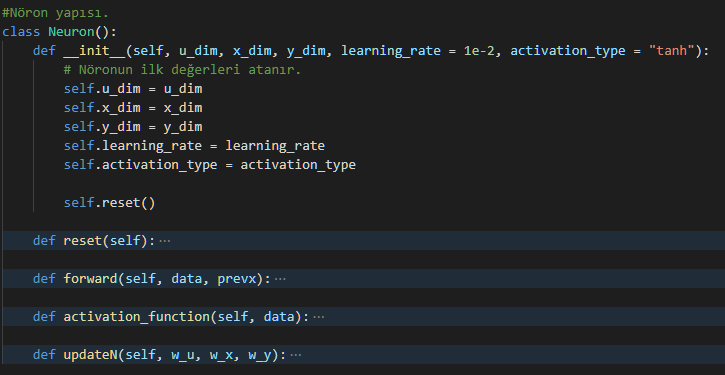


**Ağ Oluşumu ve Nöron Yapısı:**

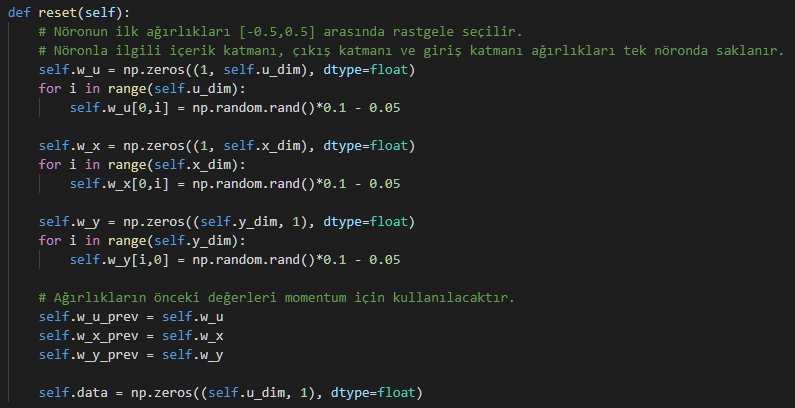
Ağ oluşturulurken “x\_dim” sayısı kadar nöron oluşturulur. Ağda aktivasyon fonksiyonu kullanan tek katman gizli katman olduğu için, çıkış ve giriş katmanı için ayrıca nöron oluşturulması gerekmez.



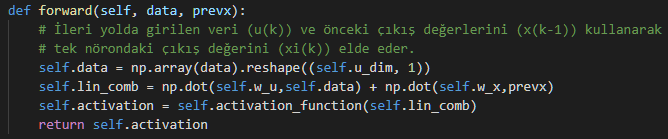
Ağ oluşumunda kullanmak üzere nöron sınıfı tanımlıdır. Nöron sınıfında girişteki verinin boyutu (u\_dim), gizli katmandaki nöron sayısı (x\_dim), ağ çıkışı boyutu (y\_dim), öğrenme hızı ve aktivasyon fonksiyonu bulunur. Nöron öncelikle sıfırlanır.



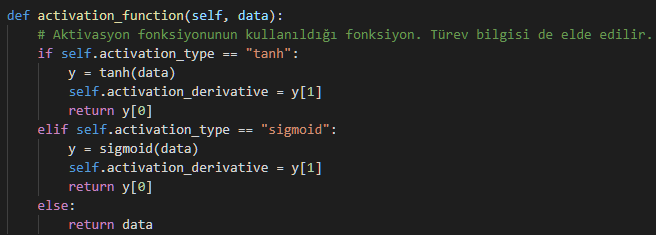
Nöron sıfırlandığında giriş katmanı, gizli katman ve çıkış katmanı ağırlıklarının başlangıç değerleri atanır. Bu değerler aralığında rastgele bir şekilde seçilir. Ayrıca veri için boş bir dizi açılıp önceki ağırlıklar güncel ağırlıklarla eşitlenir. Önceki ağırlıklar momentum için gereklidir.



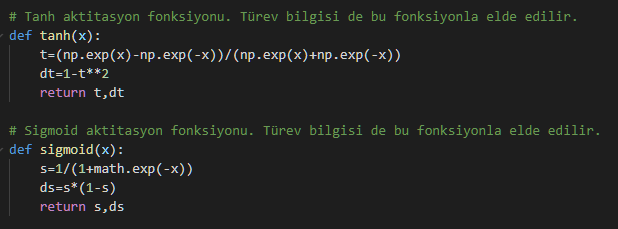
Bunlar dışında nöron yapısında üç fonksiyon bulunmaktadır. İlk olarak “forward” fonksiyonu, ağın ileri yol fonksiyonudur. Ağ girişi ve nöronun önceki iterasyondaki çıkışı ile nöron çıkışındaki değerini elde eder.



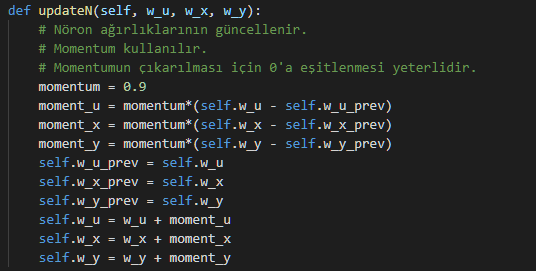
“activation\_function” fonksiyonu ise seçilen aktivasyon fonksiyonuna göre çıkışı ve aktivasyon fonksiyonunun türevini elde eder.



Aktivasyon fonksiyonları için, nöronun içindeki fonksiyona ek olarak farklı aktivasyon fonksiyonlarının tanımlandığı fonksiyonlar bulunmaktadır.

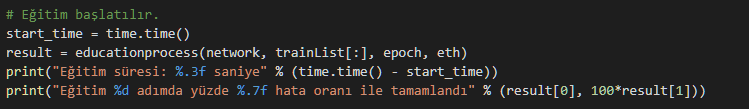


Son olarak nöronların ağırlıklarının güncellendiği bir fonksiyon bulunmaktadır. Bu fonksiyonda momentum da uygulanır. Momentum 0.9 seçilmiş olup çıkarılması için 0’a eşitlenmesi yeterlidir.

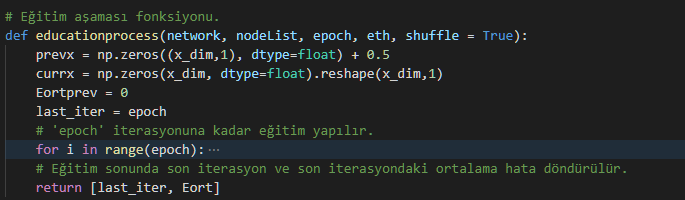


**Eğitim Aşaması:**

Ağ oluşturulduktan sonra, ağ ve eğitim kümesi eğitime girer.



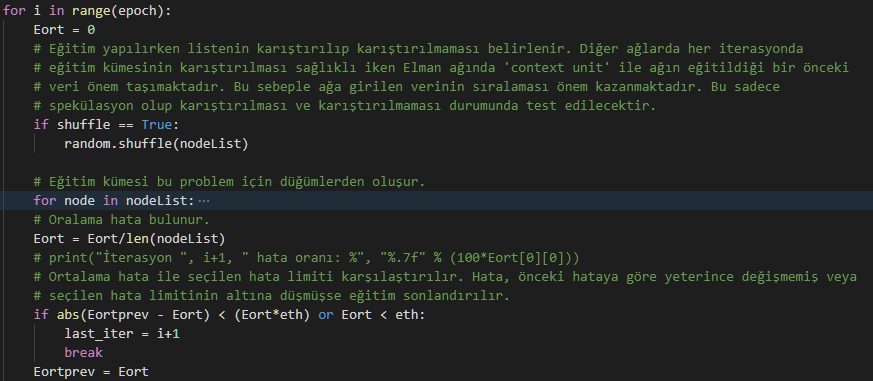
Eğitimde, her iterasyonda ağırlıklar ve nöron çıkışları dışarı çekilip eğitildikten sonra nöronlar güncellenecektir. Eğitim tamamlandığında, son iterasyon ve son iterasyondaki ortalama hata döndürülür. Başlangıçtaki değeri 0.5 olarak alınır.



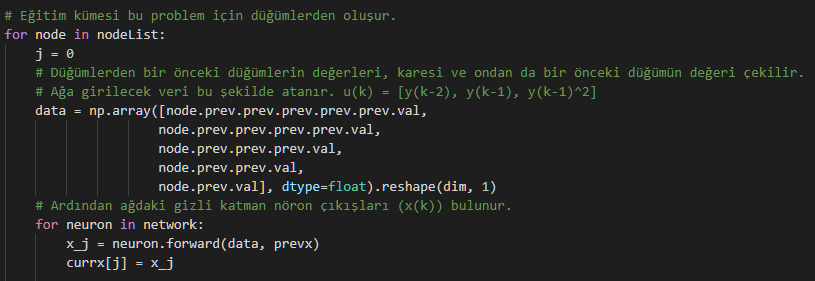
Eğitimin durması için birden fazla koşul vardır. Bu koşullardan birinin sağlanması eğitimi durduracaktır. Bu koşullar;

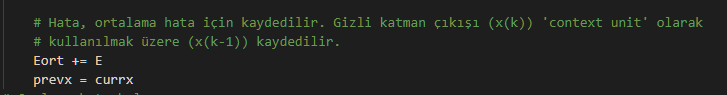
* Ortalama hatanın, hata limitinin altına düşmesi,
* Ortalama hatadaki değişimin, ortalama hatanın hata limitiyle çarpımının altına düşmesi,
* Maksimum iterasyon sayısına ulaşılmasıdır.

Her iterasyonun başında ortalama hata sıfırlanır. Ardından eğer “shuffle” değeri doğruysa (varsayılan durumda doğru), liste karıştırılır. Bu işlem çok katmanlı algılayıcı gibi ağların eğitiminin sağlıklı geçmesini sağlarken, Elman Ağı’nda ağın eğitildiği verilerin sırasının önemi olduğu için, ağın etkisiz kullanılmasını sağlayabilir. Beklenti, eğitim listesinin karıştırılması durumunda gizli katman ağırlıklarının 0’a yaklaştırılıp etkisiz bırakılması ve ağın tek katmanlı bir ağ gibi çalışmasıdır. Bu durum test edilecek ve karıştırılan ve karıştırılmayan durumlar karşılaştırılacaktır.



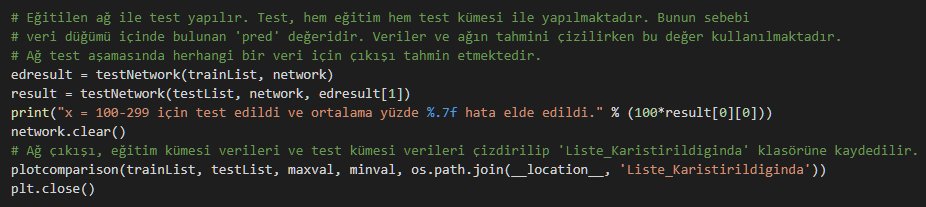
Her veri düğümü için bir eğitim döngüsü vardır. Öncelikle ağa girilecek verileri bu düğüm ile çekilir. Ardından bu veriler ve önceki nöron çıkışları (ilk adımda 0) kullanılarak ağ çıkışı elde edilir. Ağ çıkışı ile hata vektörü hesaplanır ve hesaplanan hata kullanılarak ağ ağırlıkları güncellenir.



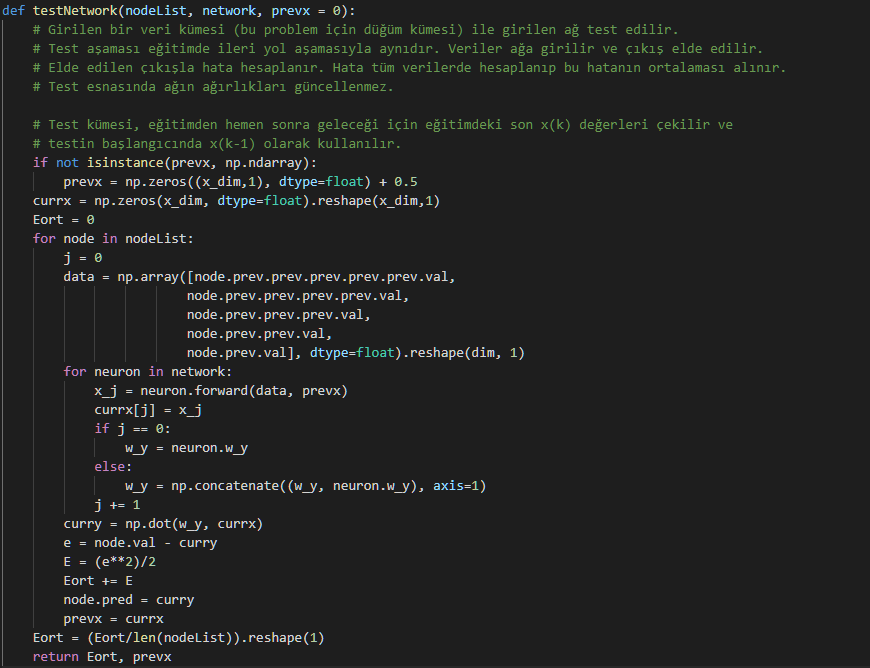
Ortalama hata eklenir, nöron çıkışları “context unit” olarak atanır ve sonraki veri düğümüne geçilir. 

**Test Aşaması:**

Eğitim bittikten sonra test aşamasına geçilir. Bu aşamada eğitim ve test kümesi, eğitilen ağ ile “testNetwork” fonksiyonuna girilir. Eğitim kümesinin de test aşamasına girmesinin sebebi, karşılaştırma amacıyla ağın eğitim kümesi verileriyle tahmini değerler oluşturmasıdır. Ayrıca test listesinin başlangıcındaki değeri, eğitim listesiyle olan testin son adımından çekilecektir.

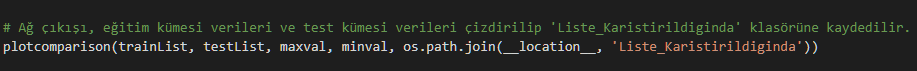


Test fonksiyonu eğitim fonksiyonuna oldukça benzemektedir. Yine ileri yol izlenip ağ çıkışı bulunur ve düğümlerde tahmin değerlerini belirten “pred”e atılır. Ağırlıklar güncellenmez. Elde edilen hata döndürülür. Başlangıçtaki değeri eğer belirtilmemişse 0.5 olarak alınır.

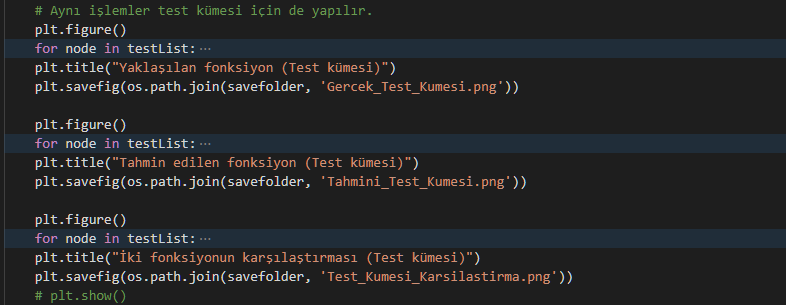
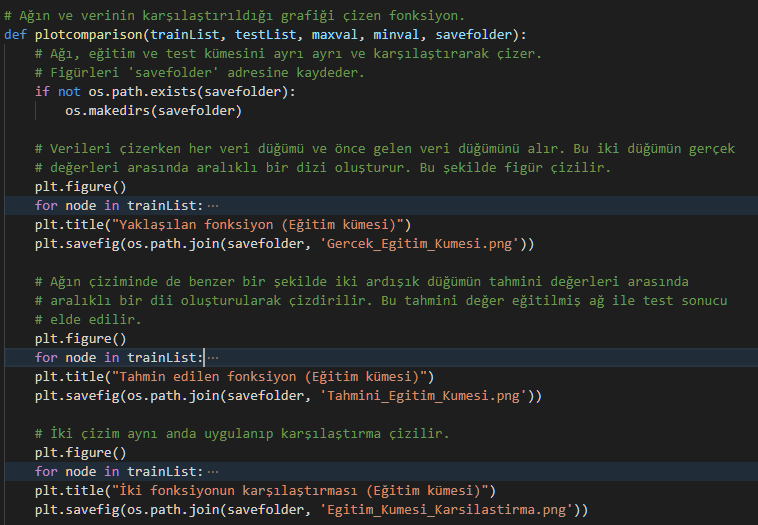


**Verilerin Karşılaştırılması:**

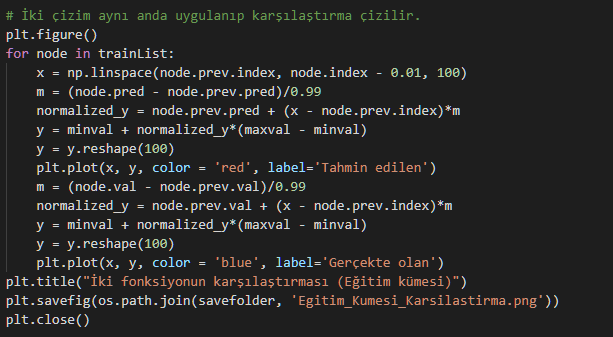
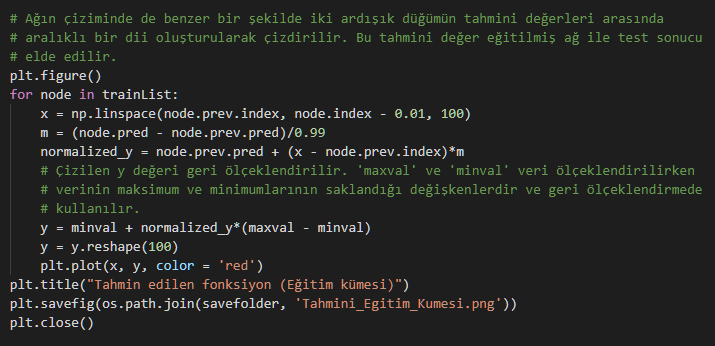
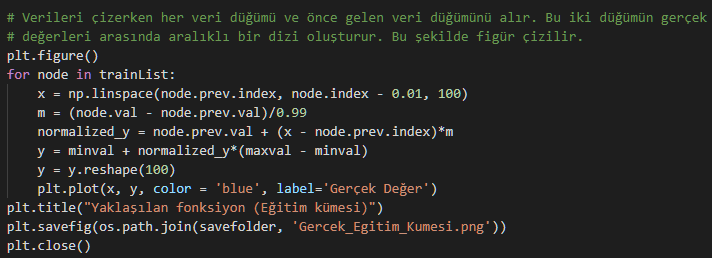
Gerçekte olan değerler ve ağın test aşamasında tahmin ettiği değerler çizdirilerek karşılaştırılır.



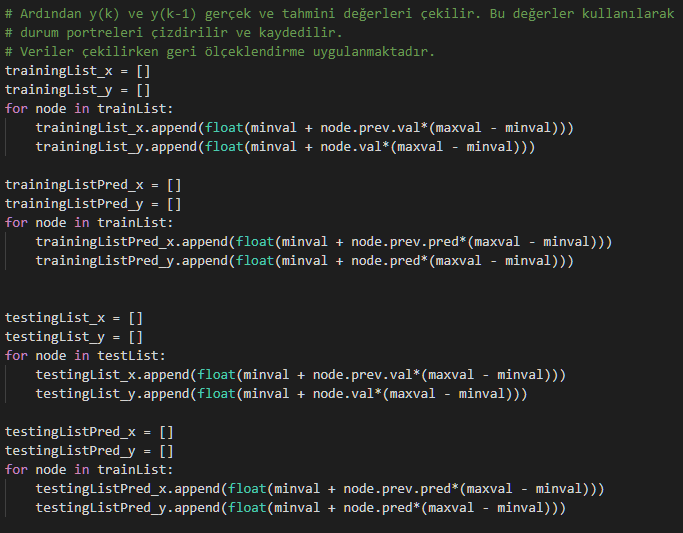
Bu karşılaştırmada eğitim ve test kümeleri için gerçek değerler ve tahmini değerler ayrı ayrı ve birlikte çizdirilir. Bu çizimler belirtilen bir klasörde kaydedilecektir.



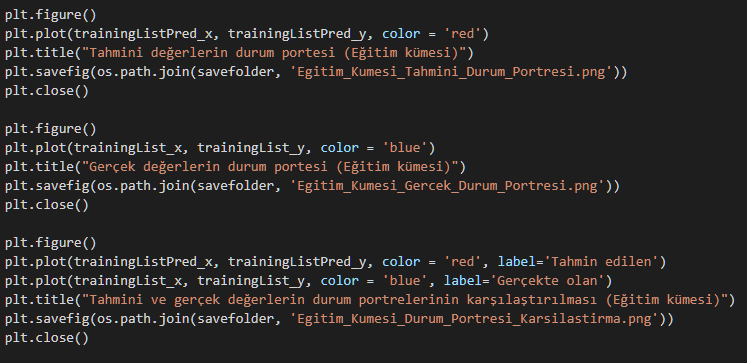
Düğümlerden önce gelen düğümlerin, gerçek ve tahmini değerleri arasında aralıklı diziler oluşturularak çizim yapılır. Düğümlerden çekilen değerler çizdirilmeden önce geri ölçekleme yapılır.

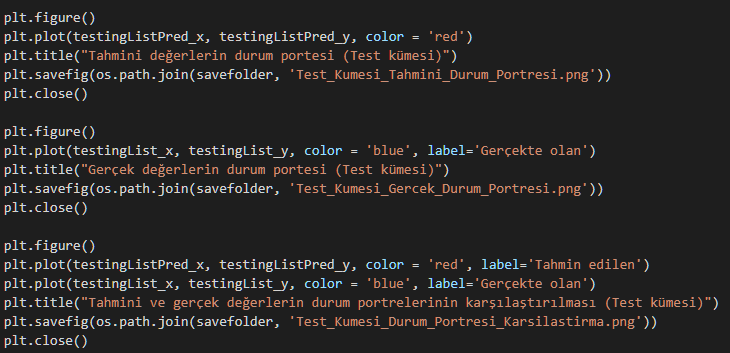


Ek olarak, durum portrelerinin çizimi için eğitim ve test kümelerindeki düğümlerden ve değerleri, gerçek ve tahmini değerler için çekilip geri ölçekleme yapılarak listelerde toplanır.



Daha sonra bu listeler ile durum portreleri çizilir.





**Eğitim ve Test Sonucu:**

Eğitim kümesi, her iterasyonun başında karıştırıldığında aşağıdaki sonuç elde edilir.



Kümenin oluşturulduğu sırayla ağa verildiğinde elde edilen sonuç ise aşağıdaki gibidir.



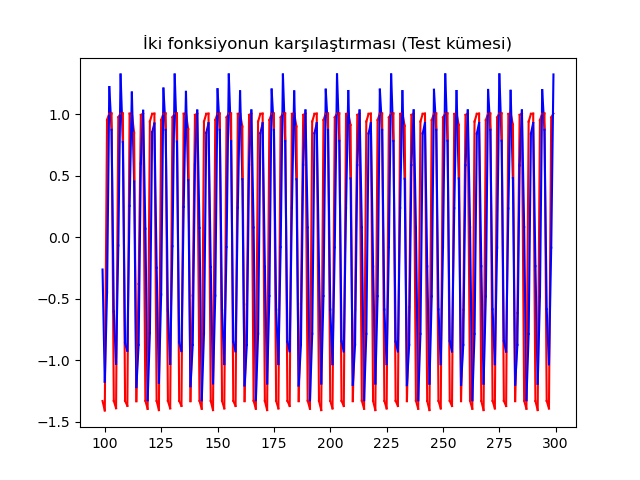
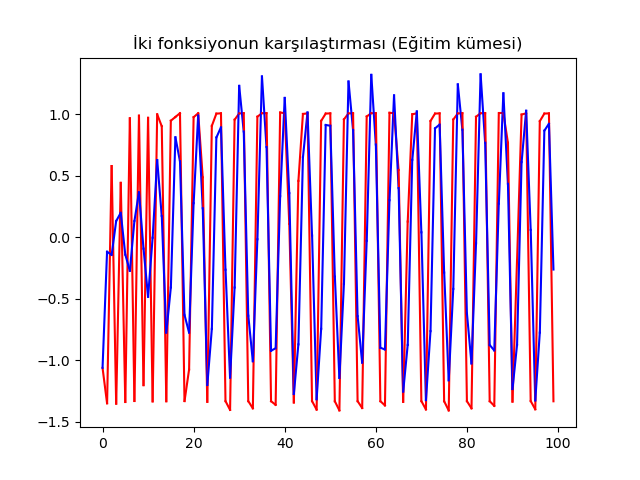
Görüldüğü üzere eğitimdeki verilerin karıştırılmaması Elman Ağı’nı oldukça iyi bir şekilde etkilemiştir. Eğitim süresi iki durumda da hemen hemen aynıdır. Test aşamasında elde edilen hata ise ilk durumda %3.513 civarındayken ikinci durumda %0.044’e kadar düşmüştür. Eğitim kümesinin sırasının Elman Ağı için önemli olduğu gözlemlenir.

Gözlemlenen bir başka durum ise eğitimin maksimum iterasyon sayısı seçilen 1000 iterasyonda sonlanmasıdır. Daha sağlıklı testler yapılabilmesi için maksimum iterasyon sayısı 3000’e çıkarılır.

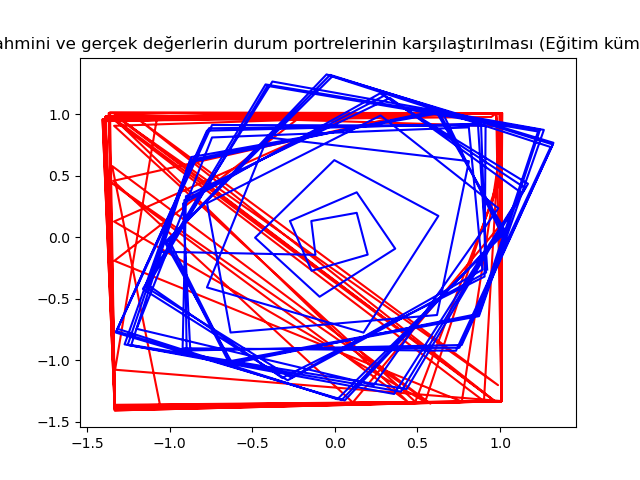
**Elde Edilen Figürler:**

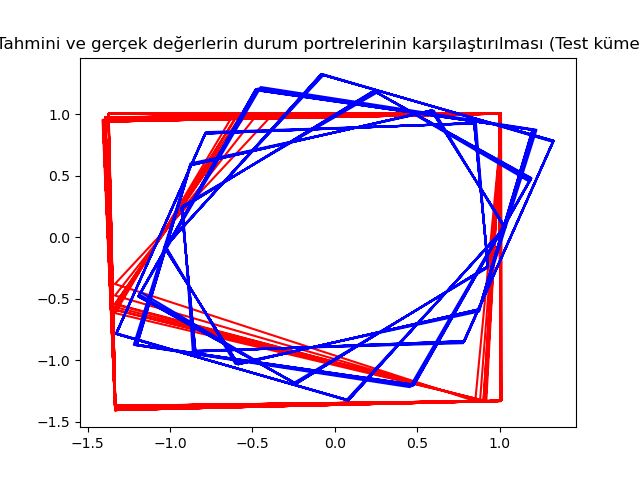
Bu bölümde ağın başarısını incelemek için oluşturulan figürler incelenecektir. Elde edilen figürlerde kırmızı renk tahmin edilen, ağ çıkışındaki sonuçları temsil eder. Mavi renk ise fonksiyon çıkışında elde edilen gerçek değerleri temsil eder.

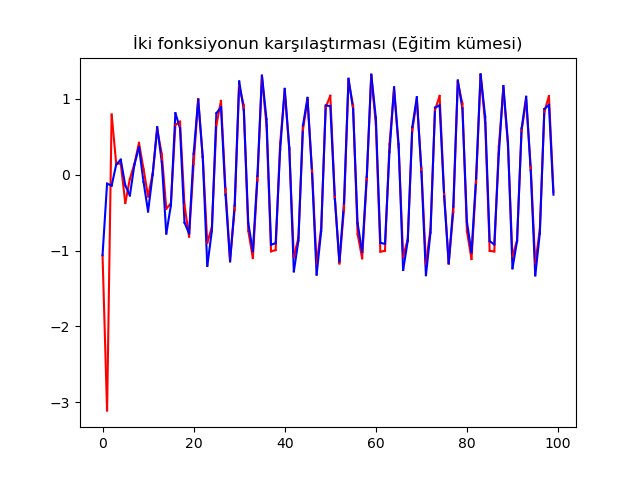
Öncelikle, başlangıçtaki liste karıştırıldığında eğitilen ağın sonuçları şu şekildedir:

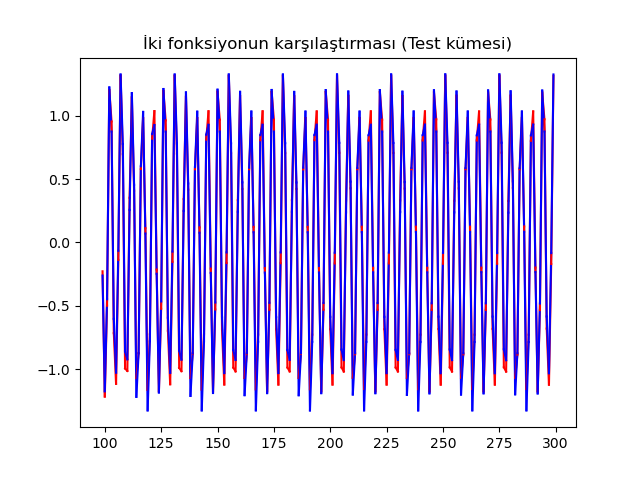


Görüldüğü üzere, liste karıştırıldığında ağ eğitim kümesine yaklaşamamış, test kümesindeki verileri de oldukça yanlış bir şekilde tahmin etmiştir. Bu durum, durum portresine bakıldığında daha net bir şekilde gözlemlenir.

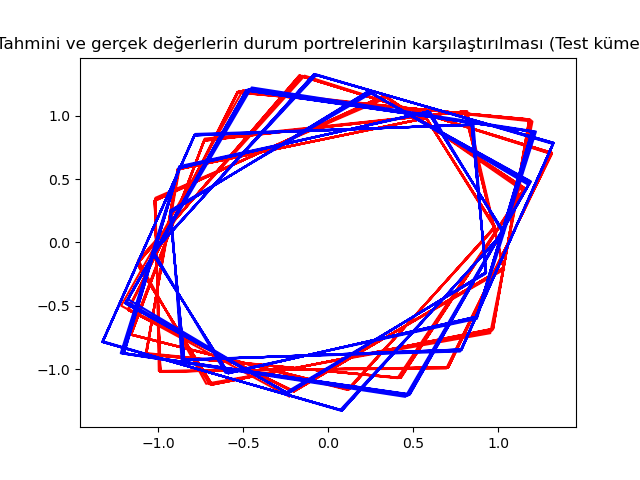




Şekillerden de görüldüğü gibi, ağ sistem davranışını temsil edememektedir. Eğitim sırasında verilerin karıştırılmadığı durum ise aşağıdaki gibidir.

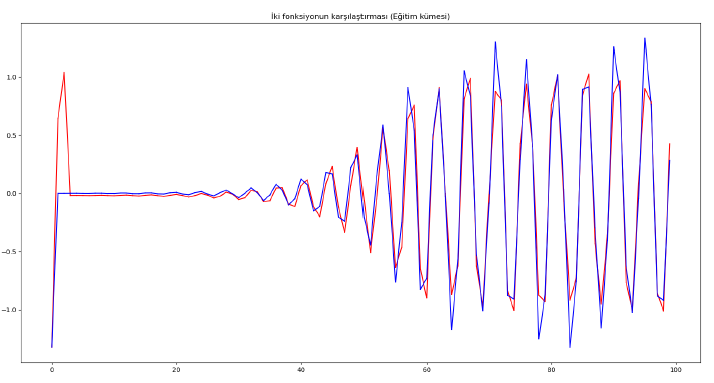


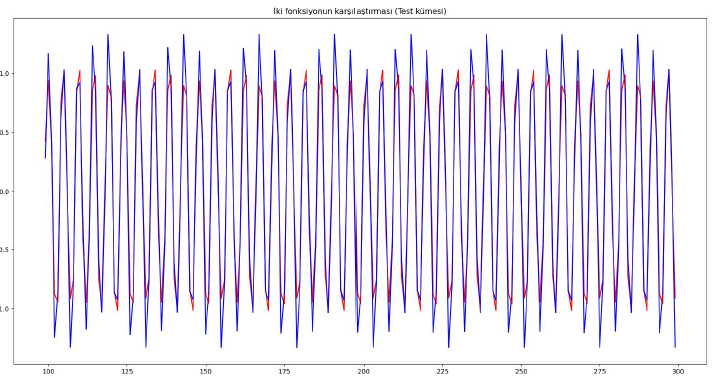
Görüldüğü gibi, eğitim kümesiyle karşılaştırıldığında başlangıçtaki ayrık davranış dışında sisteme oldukça yakındır. Eğitim başarılı bir şekilde tamamlanmış, ağ sistemi temsil edebilmektedir. Ağın, test kümesindeki verilerin de yakın bir şekilde tahmin ettiği görülür. Bazı uç noktalarda ayrıklık gösterse de iki şekil arasında çok büyük bir farklılık yoktur. Durum portreleri de ağın başarısını göstermektedir.



Görüldüğü gibi, başlangıçtaki ayrıklık dışında ağın ve sistemin durum portreleri birbirine oldukça yakındır. Yakın davranışları sergilerler fakat ağın, sistemin bir miktar ötelenmiş bir durum portresi sergilediği de görülmektedir.

**Çok Katmanlı Algılayıcı Karşılaştırması:**

Bu probleme daha önce çok katmanlı algılayıcı ile de yaklaşılmıştır. Çok katmanlı algılayıcı kullanıldığında elde edilen ağ sonuçları şu şekildedir: 



Elman Ağı’nın elde ettiği sonuçla karşılaştırıldığında, iki ağın da sisteme benzer bir başarıyla yaklaştığı görülmektedir. Fakat çok katmanlı algılayıcı kullanıldığında test sonucunda yaklaşık %0.32 hata elde edilmişken aynı parametrelerle kurulan Elman Ağı %0.044 hata elde edilerek, Elman Ağı’nın bu problem için daha uygun bir yapı olduğu görülür.

**Farklı Öğrenme Hızları:**

Kurulan ağ, farklı öğrenme hızları kullanılarak eğitim süresi, eğitim iterasyon sayısı ve hat oranı için test edilir. Maksimum iterasyon sayısı 3000, momentum 0.9 ve hata limiti 0.001 kullanılacaktır. Elde edilen sonuçlar tablodaki gibidir.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Öğrenme Hızı | 0,001 | 0,01 | 0,1 | 0,3 | 0,5 | 0,7 | 0,9 | 0,99 | 0,999 |
| **Eğitim İterasyonu** | 244 | 1978,6 | 1643,4 | 613,2 | 892,4 | 625 | 270 | 215 | 181,4 |
| **Eğitim Süresi** | 16,40236 | 147,0553 | 124,8397 | 47,17149 | 63,84811 | 43,5614 | 18,8339 | 15,05942 | 12,76322 |
| **Hata Oranı** | 0,001187 | 0,000674 | 0,000721 | 0,000284 | 0,000248 | 0,000716 | 0,001239 | 0,03775 | 0,002354 |

Öncelikle, eğitim iterasyonu sayısı incelendiğinde 0.01 ve 0.1 öğrenme hızlarındaki testlerin diğer testlerden çok daha yüksek iterasyonlara ulaştığı görülür. Çok düşük ve çok yüksek öğrenme hızlarında iterasyon sayısı en düşüktür.

Bu test için, testler arasında ağ yapısında bir değişiklik olmadığı için eğitim süresi, iterasyon sayısıyla aynı şekilde hareket etmektedir. Eğitim ortalama 50 saniye civarında gerçekleşmekte olup, en hızlı 0.999 öğrenme hızıyla 12.76 saniyede gerçekleşmiş ve en yavaş 147.06 saniyeyle 0.01 öğrenme hızıyla gerçekleşmiştir.

Hata oranları karşılaştırıldığında ise 0.99 öğrenme hızının diğerlerine göre çok daha yüksek bir hata oranına sahip olduğu görülür. Eğitim süresi en kısa olan diğer iki test, 0.001 ve 0.999 öğrenme hızları, 0.03775 kadar yüksek bir değer olmasa da diğer testlere göre kötü bir performans sergilemiştir. 0.3 ve 0.5 öğrenme hızlarının bu problemde eğitim süresi ve hata oranı açısından iyi performans sergilediği görülmektedir.

**Farklı Nöron Sayıları:**

Kurulan ağ, farklı gizli katman nöron sayıları kullanılarak eğitim süresi, eğitim iterasyon sayısı ve hat oranı için test edilir. Maksimum iterasyon sayısı 3000, momentum 0.9 ve hata limiti 0.001 kullanılacaktır. Sonuçlar tablodaki gibidir.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Gizli Katman Nöron Sayısı | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 |
| Eğitim İterasyonu | 1211 | 2001,4 | 1594,6 | 1847,8 | 1623,4 | 1220,2 | 1635,4 | 1260,8 |
| Eğitim Süresi | 52,99369 | 164,9695 | 164,4044 | 244,414 | 318,6643 | 239,6137 | 377,9592 | 339,0323 |
| Hata Oranı | 0,000462 | 0,000131 | 0,000759 | 0,000698 | 0,000672 | 0,00231 | 0,000677 | 0,001021 |

Eğitim iterasyonları karşılaştırıldığında düzenl bir değişim gözlenmez.

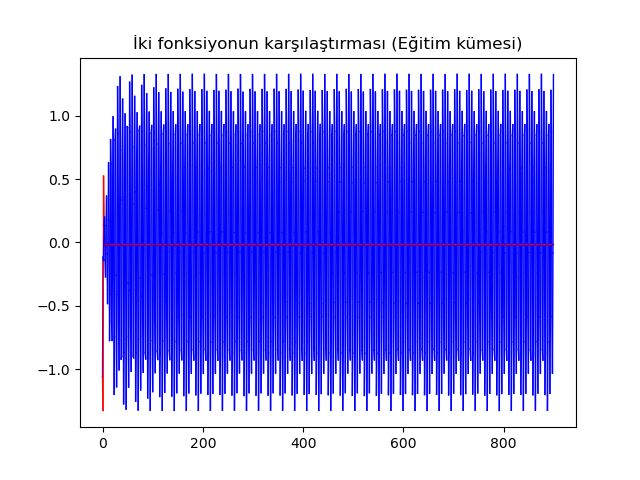
Fakat eğitim süreleri, farklı testler yapılırken ağ yapısı değiştiği için iterasyon sayısıyla aynı davranışı sergilememektedir. Aynı değerler etrafında değişen iterasyon sayılarına rağmen, her testte artan nöron sayısı eğitim süresinin de artmasına sebep olur. En düşük eğitim süresi 5 nöron kullanıldığı testteki 52.99 saniyeyken bu süre 35 nöron kullanıldığında 377.96 saniyeye kadar yükselmiştir.

Testlerdeki hata oranları karşılaştırıldığında, yüksek nöron sayısının çoğunlukla daha isabetli bir eğitime yol açmadığı görülür. 15 nöron ve sonrasındaki testlerde hata oranı yüksek kalmış, 30 nöronda 0.00231’e kadar yükselmiştir. Büyük bir farkla en isabetli ağ yapısının 0.00013 hata oranı ile 10 nöron kullanılan ağ olduğu görülür.

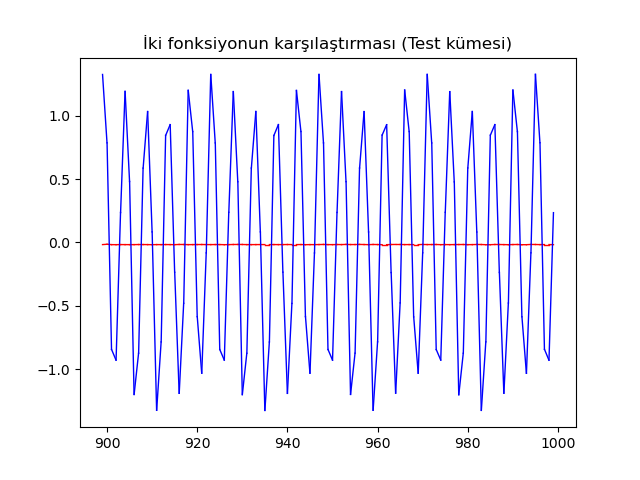
**Girişe e(k) Verilmesi:**

Ağ girişi olan u(k), yalnızca beyaz gürültü olduğunda ağ çalışmamaktadır. Bu yapının denendiği kod, önceki kodla aynı olup yalnızca eğitim ve test aşamalarında data için önceki veriler yerine np.random.normal(0,0.1) şeklinde 0 medyan ve 0.1 standart sapmalı bir beyaz gürültü kullanılmıştır. Bu kod “hw4test.py” olarak klasörde bulunmaktadır. Başlangıç ağırlıkları [-0.5,0.5] aralığına çekilmiş, eğitim kümesi ilk 900 veriden ve test kümesi sonraki 100 veriden oluşturulmuştur. Verilerin başlangıç değeri de önceki ağdaki gibi 0.1 alınmıştır. Bu değerler okuma önerilerinde bulunan ve bu ağı anlatan Pham\_96 makalesinden tavsiyelerle seçilmiştir.

Ağ çalıştırıldığında, 25 deneme içinden başarılı olan hiçbir yapı olmamış, hepsi benzer özellikler göstermiştir. Farklı denemelerin sonuçları “Test” klasöründe bulunmaktadır. Herhangi bir ağın eğitimi sonrası alınan sonuç şekildeki gibidir.



Görüldüğü gibi ağ çıkışı hızlı bir şekilde 0’a yakınsayıp, ağa benzer bir özellik göstermemiştir. Bu durum test kümesiyle karşılaştırılırken de belirgindir.



Durum portreleri incelendiğinde de ağın hızlı bir şekilde bir noktaya yakınsadığı görülür.