

Büyük Veri ve Yapay Zeka Laboratuvarı

☰ x Menu

Anasayfa

Büyük Veri

Derin Öğrenme

Veri Setleri

Projeler

Yayınlar

Kişiler

Derin Öğrenme (Yapay Sinir Ağları-3)

📅 16 Nisan 2018 👤 Semiha Makinist

Bu bölümde temel yapay sinir ağlarında kullanılan;

- Teknik Terimler,
- Toplayıcı Fonksiyonlar,
- Aktivasyon Fonksiyonları,
- Transfer Fonksiyonu,
- İleri beslemeli ağlarda yapılan temel işlem adımları
- Geri beslemeli ağlarda yapılan temel işlem adımları

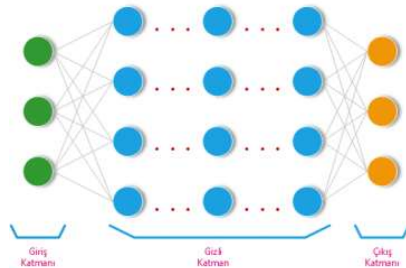
anlatılacaktır.

1. Teknik Terimler:

Bu bölümde Yapay Sinir Ağları içerisinde kullanılan temel terimler açıklanacaktır.

- **Nöron:** Temel biyoloji teriminde nöronlar, aksonları boyunca dendritlerden bir uçtan diğer uca bir elektrik sinyali gönderir. Bu sinyaller daha sonra başka bir nörona geçirilir. Bu işlemler sinir sistemi boyunca iletilerek bilgilerin beyne iletilmesini sağlar. Benzer şekilde de yapay sinir ağlarındaki nöronlar elde ettiği bilgileri diğer nöronlara taşır. Böylece sistemin giriş değerlerine göre çıkış verilerini öğrenmesini sağlar.
- **Ağ:** Nöronların birbirine bağlı olduğu graf yapılarıdır.
- **Katman:** Farklı düzeylerde yer alan nöron gruplarıdır. Yapay sinir ağları üç ana katmandan oluşmaktadır. Bunlar sırasıyla;
 - **Giriş Katmanı:** Sisteme giriş olarak gelen veriler bu katmanda yer alır. Bu katmanda giriş verileri üzerinde hiçbir değişiklik yapmadan bir sonraki katman olan hidden (gizli) katmana aktarır.
 - **Gizli Katman:** Verinin transfer edildiği katmandır. Öğrenme bu katmanda olur.
 - **Çıkış Katmanı:** Sistemin giriş verilerine göre öğrenmesini istenilen çıkış değerleri burada yer alır. Sistem çıktısının alındığı yerdir.
- **Aktivasyon Fonksiyonu:** Nörona gelen bilginin bir sonraki nörona iletilip iletilmeyeceğine karar veren birimdir.
- **İleri beslemeli Ağ:** Giriş katmanından alınan veriler sırası ile gizli katman ve çıkış katmanına iletildiği süreçtir.
- **Geri Beslemeli Ağ:** İleri beslemeli ağ ile elde edilen çıktı değerleri ile gerçek sistem çıktısı arasındaki hata oranlarına bakılarak hatanın geriye doğru yayılımı ile giriş katmanına kadar iletilmesi işlemi için geçen süreçtir.

Temel bir Yapay Sinir ağının şekli aşağıda verilmiştir;



2. Toplayıcı Fonksiyonlar:

Giriş verisinin belli ağırlık ve bias değerleri ile işleme alındığı fonksiyonlardır. Bu adımda toplama, çarpma gibi fonksiyonlar seçilmektedir. Aşağıdaki tabloda kullanılabilecek toplayıcı fonksiyonlar verilmiştir. Bu fonksiyonlar içerisinde en yaygın olarak kullanılan fonksiyon “**Ağırlıklı Toplama**”dır.

Arama

Ara ...

Çok okunan yazılar

- » Derin Sinir Ağları için Akt Fonksiyonları (11.806)
- » Google Colaboratory için Drive üzerinden Dosya Yı (9.585)
- » Veri Setleri (7.433)
- » Derin Öğrenme (Yapay Si Ağları-3) (7.111)
- » Veri Seti Hazırlama Araçl.

Etiket Bulutu

aktivasyon fonksiyonları Annotation t
indirgeme CelebA Chinese Whispers
deep learning Denetimsiz Kümeleme
öğrenme Discriminator dj
django-api dlib elu GAN Genei
Google Colaboratory G
görselleştirme install_error json Jupyt
Keras matplotlib mnist mous
neural network OpenCV pc
python read file relu sigr
swish t-SNE tanh TensorFlow
tokenizer tsne windows 10 yap
Tespiti Yüz Üretimi

İletişim

Adres

Fırat Üniversitesi
Büyük Veri ve Yapay Zeka Lal
ELAZIĞ

Email

buyukveri@firat.edu.tr

Toplam $Net = \sum_{i=1}^N X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve bulunan değerler birbirleriyle toplanarak Net girdi hesaplanır.
Çarpım $Net = \prod_{i=1}^N X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleriyle çarpılarak Net Girdi Hesaplanır.
Maksimum $Net = \text{Max}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en büyüğü Net girdi olarak kabul edilir.
Minimum $Net = \text{Min}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en küçüğü Net girdi olarak kabul edilir.
Çoğunluk $Net = \sum_{i=1}^N \text{Sgn}(X_i * W_i)$	n adet girdi içinden girdilerle ağırlıklar çarpıldıktan sonra pozitif ile negatif olanların sayısı bulunur. Büyük olan sayı hücrenin net girdisi olarak kabul edilir.
Kümülatif Toplam $Net = \text{Net}(\text{eski}) + \sum_{i=1}^N X_i * W_i$	Hücreye gelen bilgiler ağırlıklı olarak toplanır. Daha önce hücreye gelen bilgilere yeni hesaplanan girdi değerleri eklenerek hücrenin net girdisi hesaplanır.

Bazı Toplama Fonksiyonları (Çayiroğlu, 2015) [2]

Aşağıdaki fonksiyonda örnek bir ağırlıklı toplayıcı fonksiyonun işlemi gösterilmiştir.

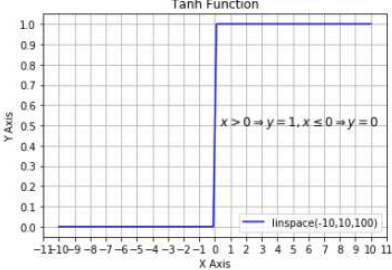
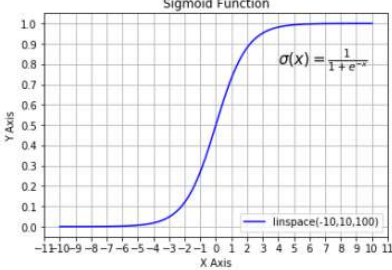
$$W \cdot X = w_1x_1 + w_2x_2 + \dots + w_mx_m = \sum_{i=1}^m w_ix_i$$

3. Aktivasyon Fonksiyonları

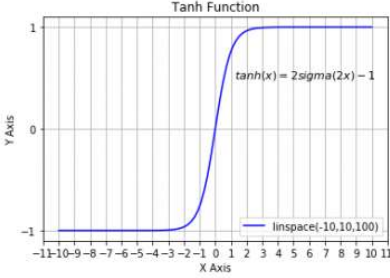
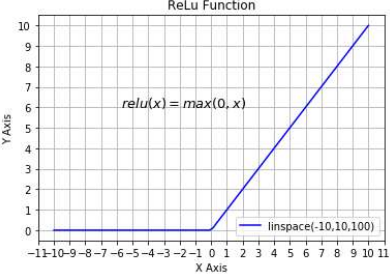
Aktivasyon fonksiyonu ile girdi değerlerine belli ağırlık ve bias değerleri eklenerek yapılan hesaplama sonucu elde edilen bilginin bir sonraki nörona iletip iletilmeyeceğine karar vermeye yaramaktadır. Aktivasyon fonksiyonu seçilir iken iki önemli duruma dikkat edilmelidir. Bunlar sırası ile;

1. Fonksiyonun lineer olmaması. (YSA'ların non-linear olması)
2. Seçilen fonksiyonun türevinin kolay alınması. (Geri beslemeli ağlar da hesaplama karmaşıklığını azaltmak için)

Aşağıdaki tabloda YSA'larda yaygın olarak kullanılan aktivasyon fonksiyonlarının listesi verilmiştir.

Aktivasyon Fonksiyonları	Grafik [1]	Açıklaması	Python Kodu
Step Fonksiyonu		<p>Giriş değeri — 0</p> <p>$x > 0$ ise $y = 1$ — $x < 0$ ise $y = 0$</p>	<pre>1 step_func = lambda x, th: np.heaviside(x, th) ?</pre>
Sigmoid Fonksiyonu		$A = \frac{1}{1+e^{-x}}$	<pre>1 sigmoid = lambda x: 1 / (1 + np.exp(-x)) ?</pre>



Tanh Fonksiyonu		$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ $\tanh(x) = 2 \operatorname{sigmoid}(2x) - 1$	<pre>1 tanh = lambda x: 2*sigmoid(2*x) - 1</pre>
ReLu		$f(x) = \max(0, x)$	<pre>1 relu= lambda x: np.maximum(0, x)</pre>

Yukardaki grafiklerinin python kodu aşağıda verilmiştir.

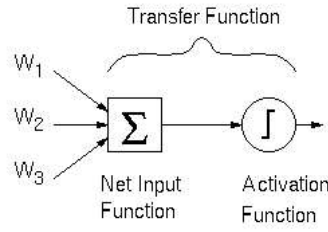
```
1 | #!/usr/bin/env python2
2 | # -*- coding: utf-8 -*-
3 |
4 | from matplotlib import pylab
5 | import pylab as plt
6 | import numpy as np
7 |
8 | from matplotlib.ticker import NullFormatter # useful for `logit` scale
9 |
10 | sigmoid = lambda x: (1 / (1 + np.exp(-x)))
11 | step_func = lambda x, th: np.heaviside(x, th)
12 | tanh = lambda x: 2*sigmoid(x) - 1
13 | relu = lambda x: np.maximum(0, x)
14 |
15 | y = plt.linspace(-10,10,100)
16 |
17 | # plot with various axes scales
18 | plt.figure(1)
19 |
20 | # linear
21 | plt.subplot(221)
22 | plt.plot(y, sigmoid(y), 'b', label='linspace(-10,10,100)')
23 | plt.text(0.5, 0.5, r'$\sigma(x)=\frac{1}{1+e^{-x}}$', fontsize=12)
24 |
25 | plt.yscale('linear')
26 | plt.title('Sigmoid Function')
27 | plt.grid(True)
28 |
29 | # log
30 | plt.subplot(222)
31 | plt.plot(y, step_func(y, 0), 'b', label='linspace(-10,10,100)')
32 | plt.text(0.1, 0.5, r'$x > 0 \rightarrow y=1, x \leq 0 \rightarrow y=0$', fontsize=11)
33 | plt.title('Step Function')
34 | plt.grid(True)
35 |
36 | # symmetric log
37 | plt.subplot(223)
38 | plt.plot(y, tanh(y), 'b', label='linspace(-10,10,100)')
39 | plt.text(-10, 0.5, r'$\tanh(x) = 2\sigma(x) - 1$', fontsize=11)
40 | plt.title('Tanh Function')
41 | plt.grid(True)
42 |
43 | # logit
44 | plt.subplot(224)
45 | plt.plot(y, relu(y), 'b', label='linspace(-10,10,100)')
46 | plt.text(-6, 6, r'$\operatorname{relu}(x) = \max(0, x)$', fontsize=13)
47 | plt.title('ReLu Function')
48 | plt.grid(True)
49 |
50 | # Format the minor tick labels of the y-axis into empty strings with
51 | # 'NullFormatter', to avoid cumbering the axis with too many labels.
52 |
53 | plt.gca().xaxis.set_major_locator(plt.MultipleLocator(1))
54 | plt.gca().yaxis.set_major_locator(plt.MultipleLocator(1))
55 |
56 | # Adjust the subplot layout, because the logit one may take more space
57 | # than usual, due to y-tick labels like "1 - 10^{-3}"
58 | plt.subplots_adjust(top=0.92, bottom=0.08, left=0.10, right=0.95, hspace=0.25,
59 |                    wspace=0.35)
60 |
61 | plt.show()
```

4. Transfer Fonksiyonu

Özetle nöronların iç yapısıdır. Toplayıcı ve aktivasyon fonksiyonlarının bir araya geldiği kaskat yapıya denmektedir.

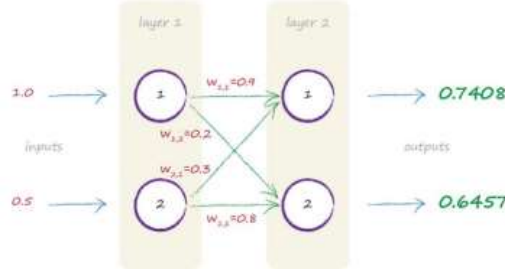
Aşağıdaki transfer fonksiyonun iç yapısı gösterilmiştir.





Örnek 1: Bu örnekte bir giriş katmanındaki verilerin çıkış katmanına nasıl taşındığını gösterilecektir. Yapılacak işlemler sadece 1 adım için geçerlidir.

İleri Beslemeli Ağlarda 1 Adım ilerlemek; (basit matris çarpımı uygulanır)

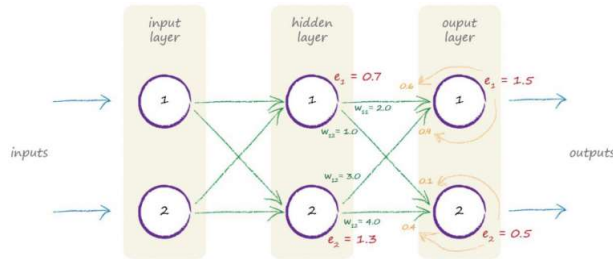


- $O1 = [(x1 \cdot 0.9 + x2 \cdot 0.3) \cdot (x1 \cdot 0.2 + x2 \cdot 0.8)]$
- $x1 = 1.0$ ve $x2 = 0.5$ 'dir;
- $O1 = [1.05 \ 0.6]$

Ağırlık toplama sonucu sigmoid aktivasyon fonksiyonuna verilir.

- $y = [\text{sigmoid}(1.05) \ \text{sigmoid}(0.6)]$
- $y = [0.7408 \ 0.6457]$

Geri Beslemeli Ağlarda 1 Adım ilerlemek (3 katmanlı nöral için);



- Matrisel gösterimi:

$$\text{Error}_{\text{hidden}} = \begin{pmatrix} \frac{w_{11}}{w_{11} + w_{21}} & \frac{w_{12}}{w_{12} + w_{22}} \\ \frac{w_{21}}{w_{21} + w_{11}} & \frac{w_{22}}{w_{22} + w_{12}} \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}$$

- $e1=1.5, e2=0.5, w11=2.0, w12=1.0, w21=3.0$ ve $w22=4.0$;
- $\text{error_hidden} = [(e1 \cdot (w11/(w11+w21)) + e2 \cdot (w12/(w12+w22))) \cdot (e1 \cdot (w21/(w11+w21)) + e2 \cdot (w22/(w12+w22)))]$
- $\text{error_hidden} = [(1.5 \cdot (2.0/(2.0+3.0)) + 0.5 \cdot (1.0/(1.0+4.0))) \cdot (1.5 \cdot (3.0/(2.0+3.0)) + 0.5 \cdot (4.0/(1.0+4.0)))]$
- $\text{error_hidden} = [(0.6 + 0.1) \cdot (0.9 + 0.4)] = [0.7 \ 1.3]$

Tüm bu işlemlerin için python ile örnek bir uygulamasını yapalım

1. Giriş katmanında yer alan parametre ve değerleri:

x1	0.9
x2	0.1
x3	0.8

2. Hidden (Gizli) Katman ağırlık değerleri;

w_input-hidden		
0.9	0.3	0.4

0.2	0.8	0.2
0.1	0.5	0.6

w_hidden-output		
0.3	0.7	0.5
0.6	0.5	0.2
0.8	0.1	0.9

Bu verileri kullanarak python programlama dilinde ileri beslemeli tek iterasyonluk küçük bir program örneği aşağıda verilmiştir.

```

1  # -*- coding: utf-8 -*-
2  import numpy as np
3
4  sigmoid = lambda Z: (1/(1+np.power(np.e, -1*Z)))
5
6  input_x_2=np.array([[0.9], [0.1], [0.8]])
7  w_input_hidden=np.array([[0.9, 0.3, 0.4],[0.2, 0.8, 0.2], [0.1, 0.5, 0.6]])
8  w_hidden_output=np.array([[0.3, 0.7, 0.5], [0.6, 0.5, 0.2], [0.8, 0.1, 0.9]])
9
10 x1=w_input_hidden.dot(input_x_2)
11 A1=sigmoid(x1)
12
13 x2=w_hidden_output.dot(A1)
14 A2=sigmoid(x2)

```

A1 Çıktı Değeri:

A1_1	0.761333
A1_2	0.603483
A1_3	0.650219

A2 Çıktı Değeri:

A2_1	0.726303
A2_2	0.708598
A2_3	0.778097

Tüm bu işlemler tüm hidden katmana uygulanır.

Bir sonraki derste ağırlık değerlerini (w) nasıl güncellendiği anlatılacaktır.

Not: Yazıdaki YSA ile ilgili tüm resim, tablo ve denklemler [3] kaynağından alınmıştır.

KAYNAKLAR:

[1] <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>

[2] <http://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/>

[3] A Gentle Introduction to Neural Networks with Python

Derin Öğrenme (Yapay Sinir Ağları-2)

Post Views: 7.111

 Derin Öğrenme  yapay sinir ağları, yapay zeka

Colab'da TensorBoard Çalıştırma (colab_utils)

Derin Sinir Ağları için Aktivasyon Fonksiyonları