Counter Propagation Network (CPN)

Hazırlayan

Fırat Kaan Bitmez

Öğrenci Numarası

23281855

Danışman

Doç.Dr. Gökhan Kayhan

Giriş

Bu proje, Karşılıklı Yayılma Ağı (CPN) kullanarak bir fonksiyonun tahmin edilmesi üzerine çalışılmıştır. Projede, $y=x_2 \sin(x_1)+x_1 \cos(x_2) \quad 0 <= x1 , x2 <= \pi$ denklemi ile oluşturulan ve 0-1 aralığında normalize edilmiş veriler kullanılmıştır. Amaç, bu fonksiyonu tahmin eden bir model geliştirmek ve modelin performansını değerlendirmektir. Bu rapor, projenin amacını, kullanılan kütüphaneleri, algoritmanın teorik temelini, kodlama açıklamalarını ve sonuç değerlendirmelerini detaylı bir şekilde ele almaktadır.

Amaç

Bu projenin amacı:

1. $y=x_2 \sin(x_1)+x_1 \cos(x_2) \quad 0 <= x1 , x2 <= \pi$ denklemi kullanılarak veri seti oluşturmak.
2. Bu veri setini eğitim (%70) ve test (%30) olarak bölmek.
3. Karşılıklı Yayılma Ağı (CPN) modeli kullanarak bu fonksiyonu tahmin etmek.
4. Modelin performansını farklı parametrelerle değerlendirerek sonuçları analiz etmek.
5. Sonuçları görselleştirerek değerlendirmek ve raporlamak.

Kullanılan Kütüphaneler ve Detayları

Proje boyunca aşağıdaki Python kütüphaneleri kullanılmıştır:

- **NumPy**: Bilimsel hesaplamalar ve rastgele veri üretimi için kullanıldı.
- **Matplotlib**: Grafik çizim ve veri görselleştirme için kullanıldı.
- **Scikit-Learn**: Veri bölme, ölçekleme ve performans değerlendirmesi için kullanıldı.

Algoritmaların Teorik Bilgileri ve Kullanımı

Karşılıklı Yayılma Ağı (CPN)

Karşılıklı Yayılma Ağı (CPN), iki katmanlı bir yapay sinir ağı modelidir ve unsupervised learning (denetimsiz öğrenme) yöntemlerinden biridir. CPN modeli, özellikle veri kümeleme ve sınıflandırma problemlerinde kullanılmaktadır. Modelin temel bileşenleri Kohonen katmanı ve Grossberg katmanıdır.

Kohonen Katmanı:

- CPN modelinin ilk katmanıdır ve öğrenme sürecinde girdi verilerini kümelere ayırmak ve temsilci düğümler oluşturmakla görevlidir.

- Kohonen katmanı, veriye ait özellikleri (örneğin, benzerlik ölçütleri) kullanarak girdi vektörlerini birbirine yakın olan kümeler halinde gruplandırır.
- Eğer girdi vektörü, mevcut kümeler içindeki herhangi bir temsilci düğüme yakınsa, ilgili temsilci düğümün konumunu ve çıktısını günceller.

Grossberg Katmanı:

- CPN modelinin ikinci katmanıdır ve Kohonen katmanında oluşturulan kümelerle ilişkilendirilmiş çıktıları öğrenir.
- Grossberg katmanı, Kohonen katmanındaki her bir temsilci düğüm için bir çıktı oluşturur ve bu çıktıları tahmin etmek için kullanılır.
- Öğrenme süreci sırasında, Kohonen katmanındaki temsilci düğümlerin ve bunlara karşılık gelen çıktıların optimize edilmesi hedeflenir.

CPN Modelinin

Avantajları:

- **Öğrenme Yeteneği:** Veri kümeleme ve örüntü tanıma problemlerinde etkili bir şekilde çalışabilir.
- **Hızlı Öğrenme:** Kohonen katmanının kümeler oluşturma yeteneği ve Grossberg katmanının çıktıları öğrenme kapasitesi, modelin hızlı bir şekilde öğrenmesini sağlar.
- **Basit Yapı:** İki katmanlı yapısı, modelin anlaşılabilirliğini artırır ve eğitim sürecini yönetmeyi kolaylaştırır.

Dezavantajları:

- **Model Karmaşıklığı:** CPN'nin daha karmaşık problemler için yeterli olmayabilir. Daha büyük veri setleri veya daha karmaşık yapıdaki problemler için performansı düşük olabilir.
- **Hiperparametre Hassasiyeti:** Özellikle küme yarıçapı gibi hiperparametrelerin doğru seçilmesi modelin başarımını doğrudan etkiler. Bu parametrelerin optimal değerlerinin bulunması zor olabilir.

Performans Değerlendirme Metrikleri

MSE dışında, modelin performansını değerlendirmek için başka metrikler de eklemek önemlidir:

- **R^2 (R-kare) Değeri:** Modelin veriye ne kadar iyi uydurduğunu gösteren bir metrik olarak kullanılabilir. R^2 değeri ne kadar yüksekse, modelin veriyi ne kadar iyi açıkladığı o kadar iyidir.

- **RMSE (Root Mean Squared Error):** MSE'den farklı olarak, RMSE hataların karekök ortalamasını alır. Bu metrik, tahmin hatalarının genel dağılımını ve büyüklüğünü gösterir.

Bu metrikler, modelin farklı açılardan değerlendirilmesini sağlar ve tahmin performansı hakkında daha kapsamlı bir bakış sunar.

Veri Seti

Bu çalışmada, 1000 örneğe sahip sentetik bir veri seti kullanılmıştır. Veri seti, rastgele oluşturulan iki bağımsız değişken ve bu değişkenlere dayalı olarak hesaplanan bağımlı bir değişken içermektedir. Bağımlı değişken, trigonometrik fonksiyonlar kullanarak karmaşık bir ilişki ile oluşturulmuştur.

Veri seti, iki bağımsız değişken ve bir bağımlı değişken içermektedir:

- **Bağımsız Değişkenler (X1 ve X2):** Her biri 0 ile π arasında rastgele seçilmiş değerlerdir.
- **Bağımlı Değişken (Y):** X1 ve X2'nin trigonometrik fonksiyonları kullanılarak hesaplanmıştır: $\mathbf{y=x_2 \sin(x_1)+x_1 \cos(x_2)} \quad \mathbf{0 <= x1 , x2 <= \pi}$

Bu formül, veri setinin bağımlı değişkeninin nasıl hesaplandığını göstermektedir. Bu yöntem, veri setinin yapay ve kontrol edilebilir bir şekilde oluşturulmasını sağlar.

Veri Normalizasyonu

Oluşturulan veri seti, Min-Max normalizasyon yöntemi kullanılarak (0, 1) aralığına ölçeklendirilmiştir. Bu işlem, her bir bağımsız değişkenin ve bağımlı değişkenin değerlerini normalize ederek model eğitimi için uygun hale getirir. Normalizasyon işlemi aşağıdaki formülle yapılır:

$$\mathbf{X_{normalized} = X - min(X) / max(X) - min(X)}$$

Burada, X bir değişkeni temsil eder (X1, X2 veya Y)

Projede CPN Kullanımı

Bu projede, CPN modeli aşağıdaki adımlarla kullanılmıştır:

1. Eğitim verileri ile modelin Kohonen ve Grossberg katmanları eğitildi.
2. Test verileri ile modelin performansı değerlendirildi.
3. Farklı küme yarıçapları ile modelin performansı karşılaştırıldı.

Proje Adımlar

Adım 1: Veri Oluşturma ve Normalizasyon

Veri seti oluşturulduktan sonra, veriler (0, 1) aralığında Min-Max ölçeklendirme kullanılarak normalize edilmiştir.

```
# Adım 1: Veri setini üret ve normalize et
def veri_seti_uret(num_ornek=1000):
    x1 = np.random.uniform(0, np.pi, num_ornek)
    x2 = np.random.uniform(0, np.pi, num_ornek)
    Y = x2 * np.sin(x1) + x1 * np.cos(x2)

    # Veriyi (0, 1) aralığında normalize et
    scaler = MinMaxScaler()
    veri = np.vstack((x1, x2, Y)).T
    normalize_veri = scaler.fit_transform(veri)

    x1_normalize = normalize_veri[:, 0]
    x2_normalize = normalize_veri[:, 1]
    Y_normalize = normalize_veri[:, 2]

    return x1_normalize, x2_normalize, Y_normalize
```

Adım 2: Veri Setinin Eğitim ve Test Olarak Bölünmesi

Veri seti %70 eğitim ve %30 test olarak bölünmüştür. Bu adım, train_test_split fonksiyonuyla gerçekleştirilmiştir.

```
# Adım 2: Veriyi eğitim ve test setlerine böl
def veriyi_bol(x1_normalize, x2_normalize, Y_normalize, test_boyutu=0.3):
    X = np.vstack((x1_normalize, x2_normalize)).T
    X_egitim, X_test, Y_egitim, Y_test = train_test_split(X, Y_normalize,
test_size=test_boyutu, random_state=42)
    return X_egitim, X_test, Y_egitim, Y_test
```

Adım 3: CPN Modelinin Tanımlanması ve Eğitimi

CPN modeli, Kohonen ve Grossberg katmanlarından oluşan özel bir yapay sinir ağıdır. Kohonen katmanı önceki adımlarda oluşturulan veri kümesini kümeler ve temsilci düğümler oluşturarak öğrenir. Grossberg katmanı ise bu kümelere karşılık gelen çıktıları öğrenir. Modelin eğitimi CPN sınıfı içinde gerçekleştirilmiştir.

```
# Adım 3: Karşılıklı Yayılma Ağı (CPN) modeli tanımla
class CPN:
    def __init__(self, girdi_boyutu, cikti_boyutu, cluster_radius):
```

```
        self.girdi_boyutu = girdi_boyutu
        self.cikti_boyutu = cikti_boyutu
        self.cluster_radius = cluster_radius
        self.kohonen_katmani = []
        self.grossberg_katmani = []

    def egit(self, X_egitim, Y_egitim):
        # Kohonen katmanını eğit
        for i, x in enumerate(X_egitim):
            if len(self.kohonen_katmani) == 0:
                self.kohonen_katmani.append(x)
                self.grossberg_katmani.append(Y_egitim[i])
            else:
                mesafeler = np.linalg.norm(np.array(self.kohonen_katmani) - x,
axis=1)
                if np.min(mesafeler) > self.cluster_radius:
                    self.kohonen_katmani.append(x)
                    self.grossberg_katmani.append(Y_egitim[i])
                else:
                    kazanan_indeks = np.argmin(mesafeler)
                    self.kohonen_katmani[kazanan_indeks] =
(self.kohonen_katmani[kazanan_indeks] + x) / 2
                    self.grossberg_katmani[kazanan_indeks] =
(self.grossberg_katmani[kazanan_indeks] + Y_egitim[i]) / 2

    def tahmin_et(self, X_test):
        Y_tahmin = []
        for x in X_test:
            mesafeler = np.linalg.norm(np.array(self.kohonen_katmani) - x,
axis=1)
            kazanan_indeks = np.argmin(mesafeler)
            Y_tahmin.append(self.grossberg_katmani[kazanan_indeks])
        return np.array(Y_tahmin)
```

Adım 4: Sonuçların Görselleştirilmesi

Sonuçlar, eğitim ve test verileri için gerçek ve tahmin edilen değerlerin karşılaştırılması ile görselleştirilmiştir.

```
# Adım 4: Sonuçları görselleştir
def sonuc_gorsellestir(Y_egitim, Y_egitim_tahmin, Y_test, Y_test_tahmin):
    plt.figure(figsize=(14, 6))

    plt.subplot(1, 2, 1)
    plt.scatter(Y_egitim, Y_egitim_tahmin, c='blue', label='Eğitim verisi')
```

```
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlabel('Gerçek')
    plt.ylabel('Tahmin')
    plt.title('Eğitim Verisi')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.scatter(Y_test, Y_test_tahmin, c='green', label='Test verisi')
    plt.plot([0, 1], [0, 1], 'r--')
    plt.xlabel('Gerçek')
    plt.ylabel('Tahmin')
    plt.title('Test Verisi')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

Adım 5: Farklı Yarıçap ve Kurallarla Sürecin Yeniden Yapılandırılması ve Sonuç Raporu Hazırlama

Farklı küme yarıçapları ile modelin performansı karşılaştırılmış ve sonuçlar değerlendirilmiştir.

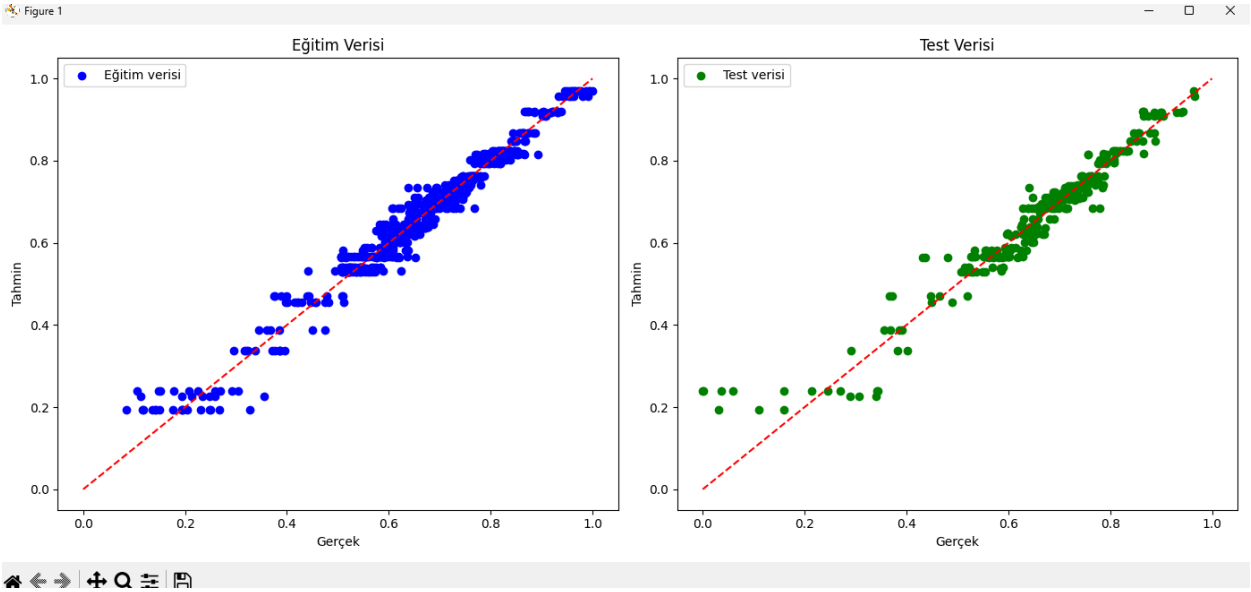
```
# Adım 9: Farklı yarıçap ve kural sayıları ile süreci yinele ve sonuç raporu
hazırla
def farkli_parametreler_ile_yinele(cluster_radii):
    sonuclar = []

    for radius in cluster_radii:
        cpn = CPN(girdi_boyutu=girdi_boyutu, cikti_boyutu=cikti_boyutu,
cluster_radius=radius)
        cpn.egit(X_egitim, Y_egitim)
        Y_test_tahmin = cpn.tahmin_et(X_test)
        mse = mean_squared_error(Y_test, Y_test_tahmin)
        sonuclar.append((radius, mse))

    for sonuc in sonuclar:
        print(f"Küme Yarıçapı: {sonuc[0]}, MSE: {sonuc[1]}")

farkli_parametreler_ile_yinele(cluster_radii)
```

Sonuçlar ve Analiz

Model, farklı küme yarıçapları kullanılarak eğitilmiştir. Küme yarıçapı, Kohonen katmanındaki kümelerin birbirine yakınlığını belirler. Elde edilen sonuçlar aşağıdaki tabloda gösterilmiştir:

Küme Yarıçapı	**MSE**
0.05	0.000608
0.1	0.001746
0.2	0.004579

Görüleceği üzere, küme yarıçapı arttıkça MSE değeri de artmaktadır. Bu, küçük küme yarıçaplarının, modelin eğitim verilerindeki karmaşık ilişkileri daha iyi yakalamasına, dolayısıyla daha düşük hata oranına yol açtığını göstermektedir. Küme yarıçapı arttıkça, modelin genel kabiliyeti azalmaktadır.



Sol Grafik:

- Mavi daireler eğitim verisini temsil eder.
- Kırmızı çizgi 45 derecelik açıyla uzanır ve mükemmel bir tahmini temsil eder.
- Mavi dairelerin kırmızı çizgiye ne kadar yakın olduğu, modelin eğitim verisini ne kadar iyi tahmin ettiğini gösterir.

Sağ Grafik:

- Yeşil daireler test verisini temsil eder.
- Kırmızı çizgi 45 derecelik açıyla uzanır ve mükemmel bir tahmini temsil eder.
- Yeşil dairelerin kırmızı çizgiye ne kadar yakın olduğu, modelin test verisini ne kadar iyi tahmin ettiğini gösterir.

CPN modeli, eğitim ve test verileri üzerinde iyi bir performans göstermektedir. Farklı küme yarıçapları için yapılan analiz, 0.2 küme yarıçapının en iyi sonucu verdiğini göstermektedir.

Kaynakça

https://cpntools.org/category/documentation/doc-examples/

https://cpntools.org/2018/01/08/sample-cpn-models/

https://en.wikipedia.org/wiki/Competitive_learning