Makine Öğrenimi ve Derin Öğrenme Projesi Raporu

İçindekiler

1. Giriş

Bu raporda, Fashion MNIST ve MNIST veri setleri üzerinde çeşitli makine öğrenimi ve derin öğrenme algoritmalarının performanslarını karşılaştıracağız. Farklı modellerin doğruluk, F1 skoru, hassasiyet ve duyarlılık gibi metrikler üzerinden değerlendirilmesi yapılacak ve elde edilen sonuçlar tartışılacaktır.

2. Projenin Amacı

Bu projenin temel amacı, farklı makine öğrenimi ve derin öğrenme algoritmalarının Fashion MNIST ve MNIST veri setleri üzerindeki performanslarını incelemektir. Hedefimiz, hangi algoritmanın bu veri setleri üzerinde en iyi performansı gösterdiğini belirlemek ve bu algoritmaların güçlü ve zayıf yönlerini tartışmaktır.

3. Kullanılan Veri Setleri

- **Fashion MNIST**: Zalando'nun makine öğrenimi çabaları için geliştirilmiş, giysi ve aksesuar görüntülerinden oluşan 10 sınıflı bir veri setidir. Her sınıf farklı bir giysi türünü temsil eder ve toplamda 70,000 gri tonlamalı görüntü içerir. Görüntüler 28x28 piksel boyutundadır.
- **MNIST**: El yazısı rakamlardan oluşan 10 sınıflı bir veri setidir. Her sınıf farklı bir rakamı temsil eder ve toplamda 70,000 gri tonlamalı görüntü içerir. Görüntüler 28x28 piksel boyutundadır.

4. Veri Ön İşleme

Veri setleri üzerinde yapılan ön işleme adımları şunlardır:

- Görüntülerin düzleştirilmesi: 28x28 boyutunda olan görüntüler, 784 boyutlu vektörlere dönüştürülmüştür.
- Verilerin normalize edilmesi: StandardScaler kullanılarak veriler normalleştirilmiştir.

```
# Verilerin normalize edilmesi ve şekillendirilmesi
scaler = StandardScaler()
train_images = train_images.reshape((train_images.shape[0], 28 * 28))
test_images = test_images.reshape((test_images.shape[0], 28 * 28))
train_images = scaler.fit_transform(train_images)
test_images = scaler.transform(test_images)
```

5. Kullanılan Yöntemler

K-Nearest Neighbors (KNN)

KNN modeli, en yakın komşu algoritmasıdır ve sınıflandırma problemlerinde sıklıkla kullanılır. KNN modeli, komşu sayısı n_neighbors=10 olarak seçilmiştir.

```
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(train_images, train_labels)
```

Lojistik Regresyon

Lojistik Regresyon, özellikle ikili sınıflandırma problemlerinde kullanılan bir algoritmadır. Bu projede çok sınıflı sınıflandırma için uygulanmıştır.

```
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(train_images, train_labels)
```

Random Forest

Rastgele Orman modeli, birçok karar ağacının bir araya gelerek oluşturduğu bir topluluk modelidir. Bu model, n_estimators=200 parametresi ile uygulanmıştır.

```
rf_classifier = RandomForestClassifier(n_estimators=200, random_state=800)
rf_classifier.fit(train_images, train_labels)
```

Karar Ağacı

Karar ağacı modeli, veri setini çeşitli dallara ayırarak sınıflandırma yapan bir modeldir.

```
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(train_images, train_labels)
```

Gradyan Artırma

Gradyan artırma, zayıf öğrenicileri ardışık olarak ekleyerek güçlü bir öğrenici oluşturan bir yöntemdir.

```
gb_classifier = GradientBoostingClassifier(n_estimators=200, random_state=42)
gb_classifier.fit(train_images, train_labels)
```

Destek Vektör Makineleri (SVM)

SVM, veriyi yüksek boyutlu bir uzaya taşıyarak lineer olarak ayırabilen bir sınıflandırma algoritmasıdır.

```
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(train_images, train_labels)
```

Yapay Sinir Ağları (Deep Learning)

Yapay Sinir Ağı (YSA), biyolojik sinir ağlarından esinlenerek geliştirilmiş bir makine öğrenimi algoritmasıdır. Bu projede iki gizli katman ve bir çıkış katmanı kullanılmıştır.

```
model = Sequential()
model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

# 3. Model Derleme
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# 4. Modeli Eğitme
epochs = 10
batch_size = 32

history = model.fit(X_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(X_test, y_test))
```

6. Deney Sonuçları

Performans Metrikleri

Bu projede kullanılan performans metrikleri şunlardır:

- **Doğruluk (Accuracy)**
- **F1 Skoru**
- **Hassasiyet (Precision)**
- **Duyarlılık (Recall)**
- **Karmaşıklık Matrisi (Confusion Matrix)**

KNN Sonuçları

- **Eğitim Doğruluğu**: %99.79
- **Doğrulama Doğruluğu**: %85.84
- **Eğitim F1-skoru**: %99.79
- **Doğrulama F1-skoru**: %85.85

Lojistik Regresyon Sonuçları

- **Eğitim Doğruluğu**: %94.08
- **Doğrulama Doğruluğu**: %84.77
- **Eğitim F1-skoru**: %94.04
- **Doğrulama F1-skoru**: %84.73

Random ForestSonuçları

- **Eğitim Doğruluğu**: %100
- **Doğrulama Doğruluğu**: %87.46
- **Eğitim F1-skoru**: %100
- **Doğrulama F1-skoru**: %87.43

Karar Ağacı Sonuçları

- **Eğitim Doğruluğu**: %100
- **Doğrulama Doğruluğu**: %79.34
- **Eğitim F1-skoru**: %100
- **Doğrulama F1-skoru**: %79.24

Gradyan Artırma Sonuçları

- **Eğitim Doğruluğu**: %98.71
- **Doğrulama Doğruluğu**: %88.57
- **Eğitim F1-skoru**: %98.70
- **Doğrulama F1-skoru**: %88.54

Destek Vektör Makineleri Sonuçları

- **Eğitim Doğruluğu**: %94.23
- **Doğrulama Doğruluğu**: %85.92
- **Eğitim F1-skoru**: %94.21
- **Doğrulama F1-skoru**: %85.90

Yapay Sinir Ağı Sonuçları

- **Eğitim Doğruluğu**: %99.44
- **Doğrulama Doğruluğu**: %88.85

7. Tartışma

Model Karşılaştırmaları

- **KNN** modeli, eğitimde yüksek performans göstermesine rağmen doğrulamada performans düşüşü yaşamıştır. Bu durum, aşırı öğrenmenin bir göstergesi olabilir.
- **Lojistik Regresyon** genellikle daha basit modellerde etkili olmuştur, ancak daha karmaşık verilerde performans kaybı yaşanmıştır.
- **Rastgele Orman** ve **Karar Ağacı** modelleri, genellikle yüksek eğitim doğruluğu gösterirken, doğrulama doğruluğunda düşüş yaşamışlardır.
- **Gradyan Artırma** ve **Yapay Sinir Ağı** modelleri, doğrulama doğruluğu açısından en iyi performansı göstermiştir.
- **SVM**, doğruluk ve F1-skoru açısından dengeli bir performans sergilemiştir.

Aşırı Öğrenme ve Alt Öğrenme

Aşırı öğrenme, modelin eğitim verisine çok iyi uyum sağladığı ancak yeni verilere genelizasyon yapamadığı durumdur. Bu proje kapsamında, Karar Ağacı ve Rastgele Orman modelleri aşırı öğrenme

eğilimi göstermiştir. Alt öğrenme, modelin hem eğitim hem de doğrulama verisinde düşük performans göstermesidir; bu durum lojistik regresyon modeli için geçerli olabilir.

Hiperparametre Optimizasyonu

Hiperparametre optimizasyonu, modellerin performansını artırmak için önemli bir adımdır. Özellikle, KNN'de komşu sayısının, Rastgele Orman'da ağaç sayısının ve Yapay Sinir Ağlarında katman sayısı ve öğrenme oranının optimize edilmesi, model performansını önemli ölçüde iyileştirebilir.

Gelecekteki Çalışmalar

Gelecekteki çalışmalarda aşağıdaki konulara odaklanılabilir:

- **Daha karmaşık veri setleri** üzerinde deneyler yapmak.
- **Hiperparametre optimizasyonu** ve model ince ayarlarını gerçekleştirmek.
- **Daha derin sinir ağları ve transfer öğrenme** tekniklerini uygulamak.
- **Çapraz doğrulama** ve **ensemble yöntemleri** kullanarak model doğruluğunu artırmak.

8. Sonuçlar

Bu raporda, farklı makine öğrenimi ve derin öğrenme algoritmalarının Fashion MNIST ve MNIST veri setleri üzerindeki performanslarını karşılaştırdık. Gradyan Artırma ve Yapay Sinir Ağı modelleri, doğrulama doğruluğu açısından en iyi performansı göstermiştir. Bu sonuçlar, uygulama gereksinimlerine en uygun modelin seçilmesine yardımcı olabilir.

9. Kaynakça

- **Zalando Research**: Fashion MNIST veri seti, Kaynak Link
- **Yann LeCun et al.**: MNIST veri seti, Kaynak Link
- **Scikit-Learn**: Python kütüphanesi, Kaynak Link
- **TensorFlow ve Keras**: Derin öğrenme kütüphaneleri, Kaynak Link

Bu rapor, farklı makine öğrenimi ve derin öğrenme algoritmalarının Fashion MNIST ve MNIST veri setleri üzerindeki performanslarını analiz etmektedir. Her modelin avantajları ve dezavantajları dikkate alınarak, uygulama gereksinimlerine en uygun model seçilmelidir.