بسمه‌تعالی

تکلیف پنجم درس یادگیری ماشین، گروه هوش مصنوعی، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان

تاریخ ارائه: 1401/3/12 موعد تحویل: 1401/4/5

1. مجموعه داده mnist را در نظر بگیرید. شبکه عصبی چندلایه‌ای طراحی کنید و دقت دسته‌بندی را در حالات مختلف گزارش نموده و برای هر قسمت نمودار رسم کنید.
   الف) استفاده از شبکه عصبی یک لایه، دو لایه و سه لایه
   ب) استفاده از تعداد نورون‌های متنوع در هر لایه به عنوان مثال 32، 64 و 16.
   ج) استفاده از تابع هزینه mse و cross_entropy
   د) استفاده از تابع فعالسازی tanh، relu و sigmoid.

2. مجموعه داده imdb مربوط به دسته‌بندی نظرات روی فیلم‌ها (دو دسته مثبت و منفی) را در نظر بگیرید. هریک از قسمت‌های الف تا د مسئله 1 را برای این سوال تکرار نموده و نتایج را تحلیل و گزارش کنید. در صورت نیاز از مفهوم مربوط به مجموعه validation جهت tuning ابرپارامترها استفاده نمایید.

3. مجموعه داده CIFAR10 را در نظر بگیرید.

   الف) شبکه عصبی پیچشی طراحی کنید و دقت دسته‌بندی را ذکر کنید.

   ب) از شبکه VGG16 (شبکه pretrained بر روی مجموعه داده ImageNet) استفاده نموده و دقت دسته-بندی را گزارش نمایید.

   ج) تکنیک dropout چیست؟ آیا استفاده از این تکنیک در هریک از قسمت‌های الف و ب می‌تواند منجر به افزایش دقت شود؟ (دقت‌های دسته‌بندی هنگام استفاده از این تکنیک در هریک از قسمت‌های فوق را بیان کنید)

   د) تکنیک data augmentation چیست؟ آیا استفاده از این تکنیک در هریک از قسمت‌های الف و ب می-تواند منجر به افزایش دقت شود؟ (دقت‌های دسته‌بندی هنگام استفاده از این تکنیک در هریک از قسمت‌های فوق را بیان کنید)

   ه) دقت دسته‌بندی در هریک از قسمت‌های الف و ب هنگام استفاده از ترکیب dropout و data augmentation را ذکر و نتایج تحلیل خود را بیان کنید.

نکته: جهت اطلاع از چگونگی دستیابی به مجموعه داده‌ها در keras می‌توانید از فایل راهنما و همچنین لینک https://keras.io/api/datasets کمک بگیرید. لازم به ذکر است نحوه استفاده از شبکه VGG16 در keras در فایل راهنما گفته شده است.

بسمه‌تعالی

درس یادگیری ماشین، گروه مهندسی هوش مصنوعی، دانشکده مهندسی کامپیوتر، دانشگاه اصفهان

یادگیری عمیق با keras

در این گزارش می‌خواهیم با keras که یکی از محبوب‌ترین پلتفرم‌های یادگیری عمیق است، آشنا شویم.

نصب keras بسیار ساده است و کافی است که با دستور pip آن را به صورت زیر نصب کنیم:

```
pip install keras
```

پیاده‌سازی شبکه عصبی چند لایه

با استفاده از keras می‌خواهیم یک شبکه عصبی چند لایه را پیاده‌سازی کنیم. در این راستا ابتدا باید مجموعه داده مدنظرمان را وارد کنیم. (ممکن است داده خاصی داشته باشیم و باید آن را load کنیم یا می‌توان در صورت لزوم از مجموعه داده‌های موجود در کتابخانه keras استفاده نمود). به عنوان مثال اگر بخواهیم از مجموعه داده mnist استفاده کنیم، اینگونه خواهیم نوشت:

```
from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data ()
```

خروجی کد بالا مجموعه تصاویر آموزشی و تست به همراه برچسب‌هایشان می‌باشد. با توجه به اینکه ورودی شبکه عصبی عمیق چند لایه باید به صورت برداری (دنباله پیکسل‌های تصویر) باشد، نیاز است عملیات reshaping صورت پذیرد که با استفاده از قطعه کد زیر می‌توان این کار را انجام داد.

```
train_images = train_images.reshape((60000, 28*28))
```

همچنین با استفاده از قطعه کد زیر می‌توان نرمالسازی (مرحله پیش‌پردازش) را برای داده‌ها انجام داد. (ابتدا به float تبدیل شده است)

```
train_images = train_images.astype ('float32')/255
```

به طریق مشابه می‌توان تصاویر تست را به بردار تبدیل و سپس نرمال نمود.

مرحله بعد پیش‌پردازش برچسب‌های مربوط به داده‌های تصاویر آموزشی و تست (پیش‌پردازش one-hot coding) می‌باشد که به صورت زیر انجام می‌شود:

```
from keras.utils import to_categorical
train_labels = to_categorical (train_labesl)
test_labels = to_categorical (test_labels)
```

قدم بعد طراحی **معماری شبکه** می‌باشد. از آنجایی که هدف، طراحی یک شبکه عصبی عمیق چند لایه است، از معماری sequential استفاده می‌کنیم. منظور از معماری sequential این است که تعدادی لایه داریم و هر لایه به لایه قبل و بعد از آن متصل می‌شود.

```
from keras import models
from keras import layers
network = models.sequential ()
```

سپس به شبکه ایجاد شده لایه‌های Dense یا fully connected اضافه می‌کنیم.

تعداد نرون‌های موجود در لایه اول: 512

```
network.add (layers.Dense (512, activation='relu', input_shape = (28*28)))
```

تعداد نرون‌های موجود در لایه دوم: 10 (به اندازه تعداد کلاس‌ها)

```
network.add (layers.Dense (10, activation='softmax'))
```

مرحله بعد شامل **کامپایل کردن** شبکه به صورت زیر است: (optimizer ، loss function را می‌گیرد و طبق آن با هدف کاهش خطا، وزن‌های شبکه را آپدیت می‌کند.)

```
network.compile (optimizer ='rmsprop', loss= 'categorical_crossentropy', metrics= ['accuracy'])
```

برای انجام مرحله **آموزش**، باید از دستور زیر استفاده کنیم:

```
network.fit (train_images, train_labels, epochs = 5, batch_size = 128)
```

جهت **ارزیابی شبکه آموزش داده شده بر روی داده‌های تست**، از دستور زیر استفاده می‌کنیم:

```
test_loss, test_acc = network. evaluate (test_images, test_labels)
print ('test_acc:', test_acc)
```

پیاده‌سازی شبکه عصبی کانولوشنی

با استفاده از keras می‌خواهیم یک شبکه عصبی کانولوشنی پیاده‌سازی کنیم. دقت داشته باشید در این قسمت شبکه به جای اینکه جنس لایه‌ها از نوع dense باشند، از نوع کانولوشن و pooling هستند. همچنین ورودی شبکه‌های کانولوشنی بردار نیست.

```
from keras import layers
from keras import models

model = models. sequential ()
model.add (layers. Conv2D (32, (3, 3), activation= 'relu', input_shape = (28, 28, 1)))
model.add (layers.MaxPooling2D ((2, 2)))
model.add (layers. Conv2D (64, (3, 3), activation= 'relu'))
model.add (layers.MaxPooling2D ((2, 2)))
model.add (layers. Conv2D (64, (3, 3), activation= 'relu'))
```

پس از تعریف لایه‌هایی از جنس کانولوشن و pooling، لازم است لایه‌های دیگری جهت دسته‌بندی داده‌ها در نظر گرفته شوند.

```
model.add (layers.Flatten ())
model.add (layers.Dense (64, activation =’relu’))
model.add (layers.Dense (10, activation = ‘softmax’))
```

تابعی در keras وجود دارد که معماری شبکه را به صورت جدول نشان می‌دهد و به صورت زیر قابل استفاده است:

```
model.summary ()
```

جهت آموزش شبکه به صورت زیر عمل می‌کنیم:

```
from keras.datasets import mnist
from keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data ()
train_images = train_images.reshape ((60000, 28, 28, 1))
train_images = train_images.astype (‘float32’)/255

test_images = test_images.reshape ((10000, 28, 28, 1))
test_images = test_images.astype (‘float32’)/255

train_labels = to_categorical (train_labesl)
test_labels = to_categorical (test_labels)

model.compile (optimizer =’rmsprop’, loss= ‘categorical_crossentropy’, metrics= [‘accuracy’])
model.fit (train_images, train_labels, epochs = 5, batch_size = 64)
```

جهت **ارزیابی شبکه آموزش داده شده بر روی داده‌های تست**، از دستور زیر استفاده می‌کنیم:

```
test_loss, test_acc = model. evaluate (test_images, test_labels)
print (‘test_acc:’, test_acc)
```

استفاده از شبکه‌های pretrained

برای استفاده از شبکه‌های pretrained مثل VGG16 می‌توان از کد زیر استفاده نمود:

```
from keras.applications import VGG16
conv_base = VGG16 (weights = ‘imagenet’, include_top = ‘false’, input_shape = (150, 150, 3)
```

در نهایت با استفاده از دستور conv_base.summary () می‌توان مشخصات شبکه کانولوشنی را مشاهده نمود.