نشریه مطالعات مهندسی کامپیوتر آپادانا
دوره ۱، شماره ۱، زمستان ۱۴۰۲، ۱۳-۱۷
ajce.apadana.ac.ir

راهنمای جامع: تسلط بر طبقه‌بندی MNIST با یادگیری عمیق

امیرمحمد ولی‌پور | دانشجوی کارشناسی رشته مهندسی کامپیوتر، مؤسسه آموزش عالی آپادانا، شیراز، ایران.

شیما اکبری * | مربی، گروه کامپیوتر، مؤسسه آموزش عالی آپادانا، شیراز، ایران.

تاریخ دریافت: ۱۴۰۲/۱۰/۰۸
تاریخ پذیرش: ۱۴۰۲/۱۱/۳۰

چکیده

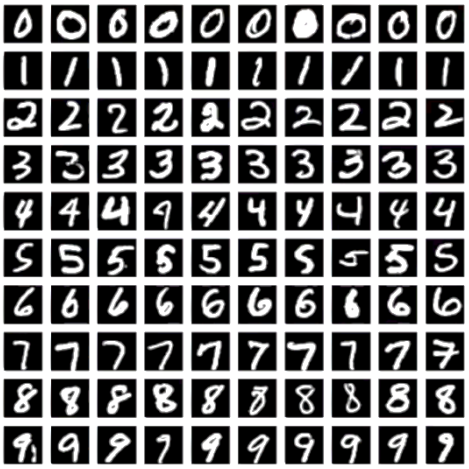
در این مقاله، ما در دنیای یادگیری عمیق کاوش می‌کنیم و کاربرد آن را در طبقه‌بندی مجموعه داده MNIST با استفاده از پایتون و کراس بررسی می‌کنیم. مجموعه داده MNIST، متشکل از ارقام دست‌نویس، یک معیار اساسی برای ارزیابی مدل‌های یادگیری ماشین بوده است. ما یک راهنمای گام‌به‌گام، از بارگذاری و پیش‌پردازش داده‌ها تا ایجاد و ارزیابی مدل ارائه می‌دهیم. معماری شبکه عصبی با Keras، شامل لایه‌های ورودی و خروجی همراه با یک لایه پنهان برای استخراج ویژگی طراحی شده است. این مدل آموزش‌دیده و ارزیابی شده است و دقت خود را بر روی داده‌های تست نشان می‌دهد. از طریق این برنامه آموزشی، خوانندگان بینش‌های عملی در مورد اصول یادگیری عمیق و پتانسیل اعمال این تکنیک‌ها در سایر وظایف بینایی کامپیوتری به دست می‌آورند.

کلیدواژه‌ها: یادگیری ماشینی، یادگیری عمیق، MNIST، بینایی کامپیوتر

* نویسنده مسئول: Abkari.shi@gmail.com

مقدمه

یادگیری عمیق، یک زمینه فرعی از یادگیری ماشینی است که از شبکه‌های عصبی با لایه‌های چندگانه برای یادگیری الگوها و بازنمایی از داده‌ها استفاده می‌کند (LeCun et al., 1998). MNIST، که شامل ۷۰،۰۰۰ عکس سیاه و سفید از ارقام دست‌نویس از ۰ تا ۹ است، یکی از رایج‌ترین مجموعه داده‌های مورد استفاده برای یادگیری عمیق است (LeCun et al., 1998).



در این مقاله، ما به بررسی چگونگی ساخت یک شبکه عصبی ساده با استفاده از پایتون و کراس برای طبقه‌بندی و تشخیص ارقام MNIST می‌پردازیم.

روش

در این مطالعه، ما از مجموعه داده MNIST شامل ۷۰،۰۰۰ تصویر خاکستری از ارقام دست‌نویس استفاده کردیم (LeCun et al., 1998). داده‌ها به ۶۰،۰۰۰ تصویر آموزشی و ۱۰،۰۰۰ تصویر آزمایشی تقسیم شدند. ما یک مدل شبکه عصبی را با استفاده از Keras API با زبان Python توسعه دادیم (Chollet et al., 2015). معماری مدل، شامل یک لایه ورودی برای تبدیل تصاویر ۲۸ x ۲۸ پیکسلی، یک لایه پنهان با ۱۲۸ گره و فعال‌سازی ReLU و یک لایه خروجی با ۱۰ گره برای طبقه‌بندی ارقام با استفاده از فعال‌سازی softmax بود. مدل برای *epoch=۵* با بهینه‌ساز Adam (Kingma & Ba, 2014) و تابع هزینه خطوط متقاطع طبقه‌بندی پراکنده (*sparse _ categorical _ crossentropy*) آموزش داده شد. دقت پس از آموزش، بر روی ۱۰،۰۰۰ تصویر آزمایشی اندازه‌گیری شد.

یافته‌ها

نتایج تحقیق نشان داد که مدل ما در ۵ دوره آموزش، ۹۷.۶۲٪ دقت را در مجموعه آزمون MNIST به دست آورد. این سطح از عملکرد، توانایی یک شبکه عصبی نسبتاً ساده را برای طبقه‌بندی مؤثر ارقام دست‌نویس براساس مقادیر پیکسل grayscale نشان می‌دهد. با تنظیم بیشتر hyperparameters مانند اندازه لایه، توابع فعال‌سازی و پارامترهای آموزشی، دقت می‌تواند حتی بیشتر بهبود یابد. این مدل قادر به تعمیم به داده‌های دیده نشده آزمون بود، که نشان‌دهنده استخراج موفق ویژگی‌های معنادار از تصاویر آموزشی بود. این یک پایه و اساس خوبی را فراهم می‌کند که می‌توان

از آن به‌عنوان نقطه شروعی برای انجام وظایف پیچیده‌تر بینایی کامپیوتر استفاده نمود. داده‌های MNIST همچنان معیار ارزشمندی برای ارزیابی تکنیک‌های یادگیری ماشین هستند.

پیش‌نیاز

قبل از اینکه کد را بررسی کنیم، اطمینان حاصل کنید که ملزومات زیر را نصب کرده‌اید:

۱. پایتون: می‌توانید آخرین نسخه پایتون را از وب سایت رسمی دانلود کنید (Python, n.d.).
۲. TensorFlow و Keras: TensorFlow یک کتابخانه یادگیری عمیق محبوب است (Abadi et al., 2016)، و Keras یک API شبکه عصبی سطح بالا است که بر روی TensorFlow اجرا می‌شود (Chollet et al., 2015).

مرور کد[1]

برای درک بهتر اینکه هر بخش چه کاری انجام می‌دهد، ما مرحله‌به‌مرحله کد را مرور می‌کنیم:

```
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
```

در این بخش، ما کتابخانه‌های مورد نیاز را وارد می‌کنیم: *Keras* از TensorFlow (Abadi et al., 2016)، *NumPy* برای عملیات‌های عددی، و *matplotlib* برای طراحی و رسم تصاویر (Hunter, 2007).

```
mnist = keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

سپس، مجموعه داده MNIST را با استفاده از ماژول *keras.datasets.mnist* بارگذاری می‌کنیم (Keras, n.d.). مجموعه داده به مجموعه‌های آموزش و تست یا آزمون تقسیم می‌شود. ما همچنین مقادیر پیکسل تصاویر را با تقسیم آن‌ها به ۲۵۵/۰ نرمال می‌کنیم، که مقادیر بین ۰ و ۱ را مقیاس‌گذاری می‌کند، و فرایند آموزش را کارآمدتر می‌سازد (LeCun et al., 1998).

```
model = keras.models.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(128, activation='relu'),
  keras.layers.Dropout(0.2),
  keras.layers.Dense(10, activation='softmax')
])
```

ما معماری شبکه عصبی خود را با استفاده از *Keras Sequential* تعریف می‌کنیم(Chollet et al., 2015). شبکه ما از سه‌لایه تشکیل شده‌است:

۱. لایه ورودی: لایه Flatten یا مسطح برای تبدیل تصاویر ۲۸x ۲۸ به یک آرایه یک بعدی با اندازه ۷۸۴ (۲۸ * ۲۸)استفاده می‌شود. این لایه به‌عنوان لایه ورودی برای شبکه عصبی ما عمل می‌کند.
۲. لایه پنهان: لایه Dense یا متراکم با ۱۲۸ نورون و تابع فعال‌سازی ReLU. تابع فعال‌سازی ReLU (واحد خطی تصحیح شده)، غیرخطی بودن را به مدل معرفی می‌کند که به آن اجازه یادگیری الگوهای پیچیده در داده‌ها را می‌دهد (Nair & Hinton, 2010).

[1]. Code Walkthrough

۳. لایه خروجی: لایه Dense یا متراکم با ۱۰ نورون (یک نورون برای هر رقم) و تابع فعال‌سازی سافت‌ماکس. سافت‌ماکس، نمرات خام لایه خروجی را به احتمالات تبدیل می‌کند و به ما کمک می‌کند تا طبقه‌ای با بیش‌ترین احتمال را به‌عنوان کلاس پیش‌بینی‌شده تعیین کنیم(Bridle, 1990).

```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

پس از تعریف مدل، باید آن را کامپایل کنیم. در طول جمع‌آوری، ما بهینه‌ساز، تابع زیان و معیارهای ارزیابی عملکرد مدل را مشخص می‌کنیم (Chollet et al., 2015). در این مورد، ما از بهینه‌ساز *adam* استفاده می‌کنیم، که یک نوع کارآمد از گرادیان کاهشی تصادفی (*SGD*) است (Kingma & Ba, 2014). برای تابع زیان، ما از خطوط متقاطع طبقه‌بندی پراکنده (*sparse _ categorical _ crossentropy*) استفاده می‌کنیم چون برچسب‌های ما اعداد صحیح هستند. در نهایت، ما از دقت به‌عنوان معیاری برای نظارت در طول آموزش استفاده می‌کنیم.

```
model.fit(x_train, y_train, epochs=5)
```

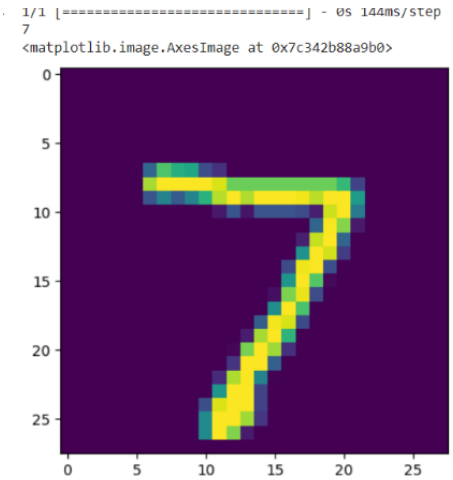
با گردآوری مدل، ما اکنون می‌توانیم آن را بر روی داده‌های آموزشی، آموزش دهیم. ما از روش *fit* استفاده می‌کنیم و در داده‌های آموزشی *x_train* و برچسب‌های متناظر *y_train* عبور می‌دهیم. آرگومان *epochs=5* تعداد دفعاتی را مشخص می‌کند که کل مجموعه داده‌های آموزشی در طول آموزش از طریق شبکه عبور خواهند کرد (Chollet et al., 2015).

```
model.evaluate(x_test, y_test, verbose=2)
```

هنگامی که مدل آموزش داده شد، می‌توانیم عملکرد آن را بر روی داده‌های آزمون یا تست با استفاده از روش ارزیابی، ارزیابی کنیم که در مدل ما این دقت عدد ۰.۹۷۶۶ رسیدیم. آرایه‌های *x _ test* و *y _ test* برای ارزیابی دقت مدل در داده‌های نادیده به‌کار می‌روند (Keras, n.d.).

```
print(np.argmax(model.predict(np.reshape(x_test[0], [-1, 28, 28]))))
plt.imshow(x_test[0])
```

در نهایت، ما پیش‌بینی‌هایی را بر روی یک داده تست (اولین مورد در کد بالا) با استفاده از *model.predict* و تابع *np.argmax* برای یافتن شاخص بالاترین احتمال استفاده می‌شود که مربوط به برچسب کلاس پیش‌بینی‌شده است (Numpy, n.d.). سپس تصویر را با استفاده از *matplotlib* و با *plt.imshow* رسم می‌کنیم (Hunter, 2007).



نتیجه‌گیری

در این مقاله، ما به بررسی چگونگی ساخت یک شبکه عصبی ساده با استفاده از پایتون و کراس برای طبقه‌بندی مجموعه داده ارقام MNIST پرداختیم. ما مراحل بارگذاری داده‌ها، ایجاد مدل، تدوین، آموزش، ارزیابی و پیش‌بینی را پوشش دادیم. مجموعه داده MNIST یک نقطه شروع عالی برای مبتدیان در یادگیری عمیق است و این کد یک پایه محکم برای ساخت پروژه‌های یادگیری عمیق پیچیده‌تر فراهم می‌کند.

به یاد داشته باشید که یادگیری عمیق یک حوزه در حال تحول است؛ و روش‌های متعددی برای بهبود عملکرد مدل از طریق تنظیم بیش از حد پارامترها، اصلاحات معماری، و افزایش داده‌ها وجود دارد (LeCun et al., 1998).

با آزمایش و بررسی بیشتر با مجموعه داده MNIST یا هر پروژه یادگیری عمیق دیگری که مشتاق به یادگیری هستید، بپردازید تا نتایج بهتر به دست آورید.

برای مشاهده دفترچه و فایل این مقاله به لینک زیر مراجعه کنید:

https://colab.research.google.com/drive/1kzD852WD9Q07tMz2MKpSO2HZafp413wr?usp=sharing

منابع

Abadi, M., Agarwal, A., & Barham, P. (2015). TensorFlow: Large-scale Machine Learning on Heterogeneous Systems, v1. 14.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications* (pp. 227-236). Berlin, Heidelberg: Springer Berlin Heidelberg.

Chollet, F., et al. (2015). Keras. https://keras.io.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science & engineering*, *9*(03), 90-95.

Keras. (n.d.). MNIST digits classification dataset. https://keras.io/api/datasets/mnist/

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y., Cortes, C., & Burges, C. J. (2010). MNIST handwritten digit database. AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 2.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807-814).

Numpy (n.d.). Numpy. https://numpy.org/

Oliphant, T. E. (2006). *A guide to NumPy* (Vol. 1). Trelgol Publishing USA.

Python. (n.d.). Welcome to Python.org. https://www.python.org/

استناد به این مقاله: ولی‌پور، امیرمحمدو اکبری، شیما. (۱۴۰۲). راهنمای جامع: تسلط بر طبقه‌بندی MNIST با یادگیری عمیق. *نشریه مطالعات مهندسی کامپیوتر آپادانا*، ۱(۱)، ۱۳-۱۷.