Tekoäly – ohjelmistokehitys

Oppijan arvioitu taitotaso: Asiantuntija

Välineet

Vuodesta 2024 lähtien yleisin ohjelmointikieli tekoälyohjelmistojen kehittämisessä on kiistatta Python, erityisesti sen kirjastot Keras/TensorFlow ja scikit-learn. Muita tekoälyohjelmistokehityksessä käytettyjä kieliä ovat MATLAB (ja sen ilmainen vastine Octave) ja C ++.

Lisäksi on olemassa koodittomia työkaluja. Niitä voidaan käyttää tekoälymallien luomiseen kirjoittamatta koodia. Esimerkki on Lobe-ohjelmisto neuroverkkojen tekemiseen kuvien luokittelutehtäviä varten (Microsoft 2021).

Harjoitus: Tunnista kaksi yleistä työkalua tekoälyohjelmistojen kehittämisessä vuonna 2024.
COBOL Vision
Keras/TensorFlow
scikit-learn
Lobe
Java
GNU Octave
2,3

Perusperiaatteet

Esimerkit on kirjoitettu Pythonilla. Jokaisessa esimerkissä on kaksi koodinpätkää: ensimmäinen on tarkoitettu Keras/TensorFlowlla luoduille neuroverkoille ja toinen scikit-learn -malleille. Näitä voidaan käyttää pohjina; kuitenkin ellipsit (…) on korvattava sovelluskohtaisella koodilla, jotta koodinpätkät toimivat. Tämän materiaalin koodi on muotoiltu ilmaisella verkkotyökalulla (HighlightCode 2024).

Datapisteet ja **tunnisteet** (labels) ovat tekoälymallin keskeisiä käsitteitä. Kukin datapiste on liitetty sitä vastaavaan tunnisteeseen. Tunnisteet voivat olla jatkuvia tai kategorisia.

- **Datapiste** viittaa dataentiteettiin; esimerkiksi analysoitavaan kuvaan.
- **Tunniste** on luokka, johon datapiste kuuluu; esimerkiksi laamaa esittävä kuva voidaan merkitä tunnisteella “laama”. Tämä on esimerkki kategorisesta luokittelusta.

Muita esimerkkejä “datapiste–tunniste”-pareista ovat:

- Ruohon laatu (tunniste) spektroskopiamittauksen (datapiste) perusteella – jatkuva tunnistaminen.

- Laitteen arvioitu käyttöikä ennen huoltotarvetta (tunniste) sensoriarvojen (datapiste) perusteella – jatkuva tunnistaminen.
- Kellonaika (tunniste), jolloin kuva on otettu, väri- ja kirkkaustietojen (datapiste) perusteella – kategorinen tunnistaminen.
- Kirjoitetun viestin tai artikkelin sävy (tunniste) sanojen, välimerkkien, tyylin ja kontekstin (datapiste) perusteella – kategorinen tunnistaminen.

Mallin rakentaminen ja datan käsittely

Tässä on pohja mallin rakentamiseen ja datan esikäsittelyyn Keras/TensorFlow:lla:

```
from keras.models import Sequential
from keras.layers import ...
from keras.layers import Activation, ..., Dense
from keras.models import load_model
from sklearn.model_selection import train_test_split

# X and y contain data and labels, respectively.
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=...,
                                                    random_state=...,
                                                    shuffle=...)

# input_shape and output_shape depend on case
model = Sequential()
model.add(..., input_shape=input_shape)
model.add(Activation(...))
...
model.add(Dense(output_shape))
model.add(Activation(...))

model.compile(...)
```

Yllä olevassa koodissa X sisältää datapisteet ja y niitä vastaavat tunnisteet.

Tässä on vastaava koodi scikit-learn -kirjastolla:

```
from sklearn import [MODELMODULE]
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import joblib

# X and y contain data and labels, respectively.
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
```

```
                                                    test_size=...,
                                                    random_state=.
    ..,
                                                    shuffle=...)

model = [MODELMODULE].[MODEL]
```

Yllä olevassa koodissa [MODEL] on Python-luokka, joka toteuttaa tekoälymallin, ja [MODELMODULE] on moduuli, joka sisältää kyseisen luokan. Esimerkiksi päätöspuun voi alustaa seuraavan mallin mukaisesti.

```
from sklearn import tree
# import other modules here if needed

# Generating the training and testing sets goes here (see code
above for reference)
# ...

model = tree.DecisionTreeClassifier()
```

Koulutus ja testaus

Tässä on esimerkki siitä, miten neuroverkko koulutetaan ja testataan Keras/TensorFlowlla sekä miten koulutushistoria tallennetaan muuttujaan:

```
history = model.fit(X_train, ..., validation_data=X_test)
```

On tärkeää huomata, että ellipsillä (…) merkityt parametrit sisältävät myös y_trainin (koulutusdatan tunnisteet). Jos fit()-funktion “verbose”-parametrin arvo on 1 tai 2, koulutuksen ja testauksen eteneminen sekä joitakin mittareita, kuten mallin tarkkuus (accuracy), voidaan nähdä prosessin aikana. Mitä suurempi tarkkuus, sitä luotettavampia vastauksia mallin odotetaan tuottavan.

Seuraava esimerkki näyttää, miten scikit-learn -malli koulutetaan ilman historian tallennusta:

```
model.fit(X_train, y_train)
```

Validointi

Mallia voidaan validoida käyttämällä sitä ennustamaan sellaisen datapisteen arvo (tyypillisesti luokka), jota se ei ole nähnyt koulutusvaiheessa.
Keras/TensorFlow-mallin validointiin voidaan käyttää seuraavaa lähestymistapaa:

```
value = model.predict(...)
```

Muuttujan “value” arvoa verrataan sitten siihen tunnisteeseen, joka vastaa ellipsillä merkittyä dataa.

Vastaavasti scikit-learn -malli voidaan validoida seuraavasti:

```
1. y_pred = model.predict(X_test)
```

Tämän jälkeen y_prediä verrataan y_testiin mahdollisten poikkeamien havaitsemiseksi. scikit-learn -kirjasto sisältää työkaluja validointiin. Esimerkiksi jatkuvia arvoja ennustavia malleja voidaan arvioida keskineliövirheen (mean squared error, MSE) avulla:

```
1. from sklearn.metrics import mean_squared_error
2. mse = mean_squared_error(y_test, y_pred)
```

Kategorisia arvoja ennustavia malleja voidaan arvioida tarkkuudella, luokitteluraportilla tai sekaannusmatriisilla (confusion matrix):

```
1. from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix
2.
3. accuracy = accuracy_score(y_test, y_pred)
4. report = classification_report(y_test, y_pred)
5. cm = confusion_matrix(y_test, y_pred)
```

Tarkkuus saa arvoja välillä 0 (täysin epätarkka) – 1 (täysin tarkka). Luokitteluraportti sisältää precision-, recall-, F1- ja support-arvot. Precision kuvaa, kuinka moni positiivisista ennusteista on oikein, recall kertoo, kuinka moni todellisista positiivisista tapauksista tunnistettiin oikein, ja F1 on tasapainotettu mittari mallin kokonaiskyvylle (precisionin ja recallin harmoninen keskiarvo). Support ilmaisee kunkin luokan todellisten esiintymien määrän aineistossa (Taha & Hanbury 2015). Precision, recall ja F1 vaihtelevat välillä 0–1 kuten tarkkuus. Support kuvaa havaintojen lukumäärää luokassa. Sekaannusmatriisi (confusion matrix) on taulukko, jossa rivit edustavat todellisia arvoja ja sarakkeet ennustettuja arvoja. Matriisin kaikkien arvojen summa vastaa ennusteessa käytettyjen havaintojen määrää. Voit lukea lisää sekaannusmatriiseista ja niiden pisteyttämisestä interaktiivisella työkalulla täältä.

Tekoälyn integrointi sovellukseesi

Mallin tallentaminen/lataaminen

Keras/TensorFlow-malli voidaan **tallentaa** tiedostoon myöhempää käyttöä varten:

```
1. model.save(...)
```

jossa ellipsillä (…) merkityt parametrit sisältävät mallitiedoston nimen.

Scikit-learn-malli voidaan **tallentaa** tiedostoon, joblib-moduulin funktiolla:

```
1. joblib.dump(model, filename)
```

Yllä olevassa koodissa “filename” on kohdetiedoston nimi merkkijonona. Esimerkiksi:

```
1. joblib.dump(model, "my_classifier")
```

tallentaa mallin tiedostona nimeltä "my_classifier".

Keras/TensorFlow-malli voidaan **ladata** tiedostosta:

```
1. model = load_model(...)
```

jossa ellipsillä (…) merkityt parametrit sisältävät mallitiedoston nimen.

Scikit-learn-malli voidaan **ladata** tiedostosta joblib-moduulin funktiolla:

```
1. model = joblib.load(filename)
```

Yllä olevassa koodissa “filename” on lähdetiedoston nimi merkkijonona. Esimerkiksi:

```
1. model = joblib.load("my_classifier")
```

lataa saman “my_classifier”-mallin, joka tallennettiin edellisessä esimerkissä.

Ladatun mallin käyttäminen

Ladattua mallia käytetään kutsumalla sen predict-funktiota siten, että analysoitava data annetaan parametrina.

Ladattua Keras/TensorFlow-mallia voidaan kutsua seuraavasti:

```
1. y_pred = model.predict(X_test)
```

Vastaavasti ladattua scikit-learn -mallia voidaan kutsua:

```
1. y_pred = model.predict(X_test)
```

Molemmissa tapauksissa X_test sisältää datan, josta ennusteet tehdään, ja y_pred sisältää ennusteiden tulokset (tunnisteet tai arvot).
(Nämä kaksi koodiesimerkkiä näyttävät identtisiltä!)

Ohjelmiston hahmotelma

Harjoitus: Laadi tekoäly ohjelman rakenne.	
Hyödynnä mallia sovelluksessasi.	Vaihe 1
Kouluta malli.	Vaihe 2
Luo koulutus- ja testidata.	Vaihe 3
Luo malli.	Vaihe 4
Tallenna malli.	Vaihe 5
Testaa malli.	Vaihe 6
	3,4,2,6,5,1

Seuraava vaihe

Jatka kohdasta Etiikka ja tekoäly

Bibliografia

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. &

Zheng, X. 2015. *TensorFlow: Large-scale machine learning on heterogeneous systems*. Available at: http://download.tensorflow.org/paper/whitepaper2015.pdf. Accessed 17 September 2024.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. Available at: https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf. Accessed 17 September 2024.

Van Rossum, G. & Drake, F.L., 2009. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Viittaukset

HighlightCode. 2024. *Online Highlight Code | Paste into Microsoft Word or OneNote etc*. Available at: https://highlightcode.com/. Accessed 19 June 2024.

Microsoft. 2021. *Lobe*. Available at: https://www.lobe.ai/. Accessed 4 June 2024.

Taha, A.A. & Hanbury, A., 2015. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. *BMC Medical Imaging*, 15(1). Available at: https://bmcmedimaging.biomedcentral.com/articles/10.1186/s12880-015-0068-x. Accessed 15 October 2024.