Artificial Intelligence – Software Development

Estimated Learner Skill Level: Expert

Tools

As of 2024, the most common programming language for developing AI software is arguably Python, specifically its libraries Keras/TensorFlow and scikit-learn. Other languages used for AI software development include MATLAB (and its free counterpart, Octave) and C++.

In addition, no-code tools exist. They can be used to create AI models without writing any code. An example is the Lobe software for making neural networks for image classification tasks (Microsoft 2021).

Exercise: Identify two common tools for AI software development in 2024.

COBOL Vision
Keras/TensorFlow
scikit-learn
Lobe
Java
GNU Octave
2,3

Basic Principles

The examples are written in Python. Each example has two code snippets: the first is for neural networks created using Keras/TensorFlow, and the second is for scikit-learn models. These can be used as templates; however, the ellipses must be replaced with application-specific code to make the snippets functional. The codes in this material are formatted using a free online tool (HighlightCode 2024).

Data points and **labels** are key concepts of an AI model. Each data point is paired with its corresponding label. Labels can be continuous or categorical.

- **Data point** refers to a data entity; for example, an image that is analysed.
- **Label** is the category the data point belongs to; for example, an image of a llama can be labelled as “llama”. This is an instance of categorical labelling.

Other examples of “data point and label” pairs are:

- Grass quality (label) based on a spectroscopy measurement (data point) – continuous labelling.

Euroopan unionin osarahoittama

Elinvoimakeskus

centria University of Applied Sciences

Kokkola Karleby

kpedu

- Estimated lifetime of a device before maintenance is needed (label) based on sensor values (data point) – continuous labelling.
- Time-of-day (label) when a photo was taken based on colour and brightness information (data point) – categorical labelling.
- Tone (label) of a written message or article based on words, punctuation, style, and context (data point) – categorical labelling.

Building a Model and Dataset

Here is a template for building a model and dataset with Keras/TensorFlow:

```
1. from keras.models import Sequential
2. from keras.layers import ...
3. from keras.layers import Activation, ..., Dense
4. from keras.models import load_model
5. from sklearn.model_selection import train_test_split
6.
7. # X and y contain data and labels, respectively.
8. X_train, X_test, y_train, y_test = train_test_split(X,
9.                                                     y,
10.                                                    test_size=...,
11.                                                    random_state=.
   ..,
12.                                                    shuffle=...)
13.
14.# input_shape and output_shape depend on case
15.model = Sequential()
16.model.add(..., input_shape=input_shape)
17.model.add(Activation(...))
18....
19.model.add(Dense(output_shape))
20.model.add(Activation(...))
21.
22.model.compile(...)
```

In the code above, X has the data points, and y has the corresponding labels.

Here is a template for building a model and dataset with scikit-learn:

```
1. from sklearn import [MODELMODULE]
2. from sklearn.model_selection import train_test_split
3. from sklearn.metrics import accuracy_score, classification_report
4. import joblib
5.
6. # X and y contain data and labels, respectively.
7. X_train, X_test, y_train, y_test = train_test_split(X,
8.                                                     y,
```

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

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

In the code above, [MODEL] is a Python class implementing AI technology, and [MODELMODULE] is the module containing the class. For example, a decision tree can be set up using the following snippet as a base.

```
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()
```

Training and Testing

Here is an example of how to train and test a neural network with Keras/TensorFlow and store training history into a variable:

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

It is important to note that the parameters represented by the ellipsis should include y_train (the training data labels). If the "verbose" parameter of the fit() function is 1 or 2, the training/testing progress and some metrics, including model accuracy, can be seen during training and testing. The higher the accuracy, the more truthful the answers the model is expected to give.

Here is an example of how to train a scikit-learn model without storing the history:

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

Validating

A model can be validated by using it to predict the value (typically a category) of a data point it has not been trained on.
To validate a Keras/TensorFlow model, one can use the following code as a basis:

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

The value of "value" is then compared to the label corresponding to the data represented by the ellipsis.

Similarly, a scikit-learn model can be validated with

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

Then, y_pred is compared to y_test for discrepancies.
The scikit-learn module has tools for validation. For example, models predicting continuous values can be evaluated with mean square error (MSE):

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

Models predicting categorical values can be evaluated with an accuracy score, classification report, or confusion matrix:

```
from sklearn.metrics import accuracy_score, classification_report,
    confusion_matrix

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
```

The accuracy score ranges from 0 (completely inaccurate) to 1 (fully accurate). The classification report contains precision, recall, F1, and support scores. Precision indicates how many of the predicted positives are correctly predicted, recall expresses how many of the actual positives are correctly predicted, F1 is a balanced metric estimating overall performance of the model (it is the harmonic mean of precision and recall), and support is the number of actual occurrences of each class in the dataset (Taha & Hanbury 2015). Precision, recall, and F1 range from 0 to 1, like the accuracy score (Taha & Hanbury 2015). The support score represents the number of samples in a category. The confusion matrix is a table where rows are the actual values and columns represent the predicted values. The sum of the confusion matrix is the number of samples used in the prediction. You can learn more about confusion matrices and scoring them with an interactive tool here.

Integrating the AI into Your App

Saving/loading a Model

A Keras/TensorFlow model can be **saved** to a file for later use with

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

where the parameters represented by ellipsis include the model filename.

A scikit-learn model can be **saved** to a file with a function from the joblib module:

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

In the above code, “filename” is the destination filename as a string. For example,

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

will save the model as a file named “my_classifier”.

A Keras/TensorFlow model can be **loaded** from a file with

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

where the parameters represented by ellipsis include the model filename.

A scikit-learn model can be **loaded** from a file with a function from the joblib module:

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

In the above code, “filename” is the source filename as a string. For example,

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

loads the same “my_classifier” model we saved in the previous code.

Calling a loaded Model

A loaded model is used by calling its predict function with the data to be analysed among parameters.

A loaded Keras/TensorFlow model can be called with

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

and similarly, a loaded scikit-learn model can be called with

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

In both cases, X_test is the data on which the model will make predictions, and y_pred will contain the prediction results (labels or values).
(The two code snippets look identical!)

Software Outline

Exercise: Create an outline of an AI program.

Use the model in your application.	Step 1
Train the model.	Step 2
Generate train/test data.	Step 3
Create the model.	Step 4
Save the model.	Step 5
Test the model.	Step 6
	3,4,2,6,5,1

Next Step

Move on to Ethics and AI

Bibliography

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.

References

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.