The Design and implementation of ML methods using existing commonly used ML toolboxes and platforms

FALL 2023 CS5806 guest lecturer: Hongjie Chen

VIRGINIA TECH

Agenda

1. Introduction of ML toolkits and platforms
2. Design and Implementation
3. Discussion (ask about your applications)

ML toolkits and platforms - implement and test your ML algorithms.

Based on Python: mainstream, frequently updated, support from forums

Other languages:

Matlab, R: statistics

Java, C++: scalable, fast calculation

Julia, Scala, Lua, Prolog: have their specific purposes

Python packages: Tensorflow, PyTorch, Keras, scikit-learns, MXNet, Caffe, CNTK, Theano, DL4J, Fast.ai, Hugging Face, XGBoost, LightGBM, CatBoost, OpenCV, AutoML, etc.

Anaconda and Jupyter Notebook

Anaconda allows you to manage different environments

- env 1: torch version 1
- env 2: torch version 2

Jupyter Notebook is an interactive python software

Packages

Before deep learning

scikit-learns supports many ML algorithms:

1. linear models (linear regression, logistic regression, etc.)
2. decision trees
3. ensemble models (random forests, gradient boost, etc)
4. Naïve Bayes
5. KNN
6. SVM
7. Multi-layer perceptron (a.k.a., Neural Networks)

Also unsupervised: 1. clustering (K-mean, DBSCAN, etc.) 2. Dimensionality Reduction (PCA, t-SNE, SVD, etc.)

sklearn

also has many criteria for model evaluations

1. data splitting
2. cross validation
3. confusion matrix
4. ...

To install and use:

```
pip install scikit-learn
```

```
import sklearn
```

Other ML packages

XGBoost, LightGBM, CatBoost: Gradient Boosting

More?

Deep learning - Big three (personal thought)

Tensorflow, Pytorch, MXNet

1. supported by big communities
2. fine control of models (e.g., change the NN structure during training)
3. many derivatives (dgl for graphs for all, timm for Images from pytorch, torchaudio for audio from pytorch, gluonts for time-series from mxnet)

Note: go to their pages and see how to install them.

name1 and name2 can be different in `pip install name1` and `import name2`

No need for complicated functions?

Keras, CNTK

Caffe: originally for CV

Other packages

Theano: defines computational graphs

DL4J: DL for JVM

Fast.ai, Hugging Face: define many popular models

OpenCV, AutoML, etc.: for more specific uses

Design Workflow

Design - workflow

1. Prepare data
2. Implement your model
3. train your model

Get a trained model. Done!

Life can be more complicated

We need to break down steps to resolve these challenges:

1. Memory is not big enough to store all data at a time.
2. How do we know if the trained model works?
3. We want to use some strategies during training, for example, reduce the update speed when the loss is small.
4. ...

Design - revised workflow

1. Prepare data
    a. support data transformation and splitting
    b. adopt train/val/test set if possible
2. Implement your model
3. train your model
    a. load your data in a generator or an iterator manner so that it yields a batch at a time
    b. train for several epochs, allows learning rate to be updated in different epochs, use a patience indicator to stop after the loss doesn't reduced after certain epochs

Get a trained model. Done!

Life can be even more complicated

We need to add more steps to fulfill these needs:

1. reproduce your (best) result
2. make the training procedure easier to understand
3. make your model easier to understand
4. monitor your model weights update
5. train several models with different hyperparameters and select the best one
6. ...

Design - revised workflow

1. Prepare data
    a. support data transformation and splitting
    b. adopt train/val/test set if possible
2. Implement your model
3. train your model
    a. load your data in a generator or an iterator manner so that it yields a batch at a time
    b. train for several epochs, allows learning rate to be updated in different epochs, use a patience indicator to stop after the loss doesn't reduced after certain epochs

Get a trained model. Done!

Discussion - more to consider?

1. Prepare data
    a. support data transformation and splitting
    b. adopt train/val/test set if possible
2. Implement your model
3. train your model
    a. load your data in a generator or an iterator manner so that it yields a batch at a time
    b. train for several epochs, allows learning rate to be updated in different epochs, use a patience indicator to stop after the loss doesn't reduced after certain epochs

Let's see some real examples

CV: ResNet with PyTorch: https://github.com/Lornatang/ResNet-PyTorch

Audio: Wav2Net: https://github.com/khanld/Wav2vec2-Pretraining/blob/main/run.py

A lot more on hugging face https://huggingface.co/

...

Discussion: your applications