wbs
WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

For the Change Makers

Advanced Programming for Data Science

Week 10: Neural Network and Review

Information Systems and Management
Warwick Business School

Working with Jupyter-notebook for Assessment

- Example not template.
- Markdown "text" as your analysis narrative
- Organize your analysis with Markdown "text".

Neural Network and Deep Learning

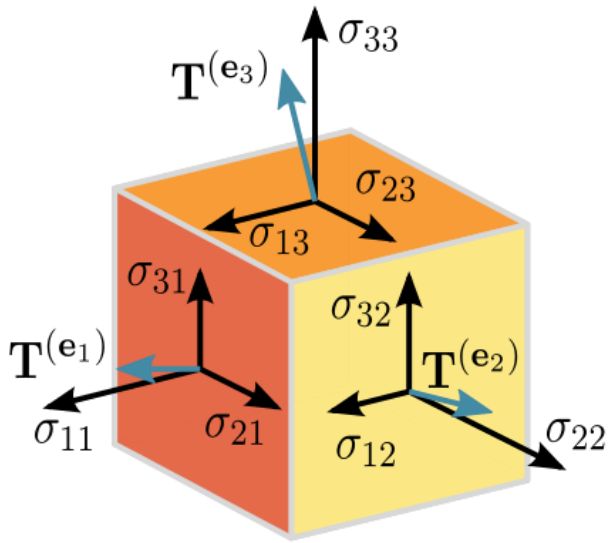
TensorFlow vs. PyTorch

TensorFlow

Tensor

- Tensors are similar to NumPy's ndarrays, with the addition being that Tensors can also be used on a GPU to accelerate computing.
- A vector is a one-dimensional or first order tensor and a matrix is a two-dimensional or second order tensor.

```
x = torch.empty(5, 3)
print(x)
```



First "deep" neural network

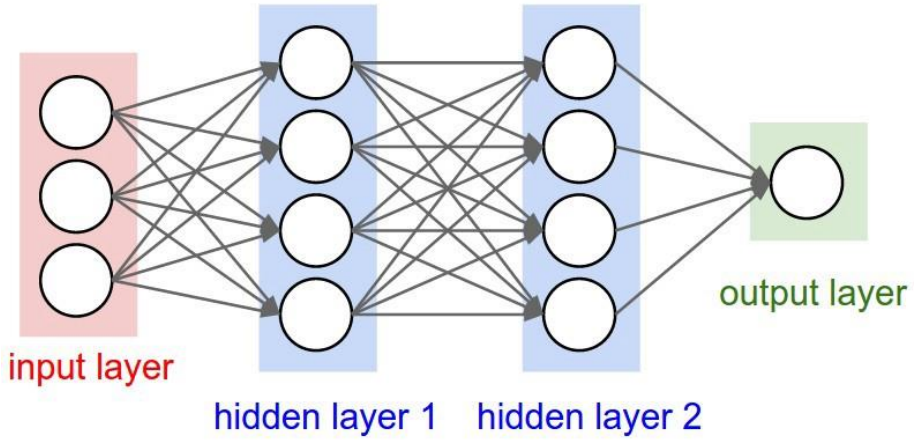
```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
network = Sequential()
network.add(Dense(8, activation='relu', input_dim=4))
network.add(Dense(3, activation='softmax'))
network.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
network.fit(train,labels,epochs=100,batch_size=5)
```

Creating the network

```
network = Sequential()
network.add(Dense(8, activation='relu', input_dim=4))
network.add(Dense(3, activation='softmax'))
network.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
network.fit(train,labels,epochs=100,batch_size=5)
```

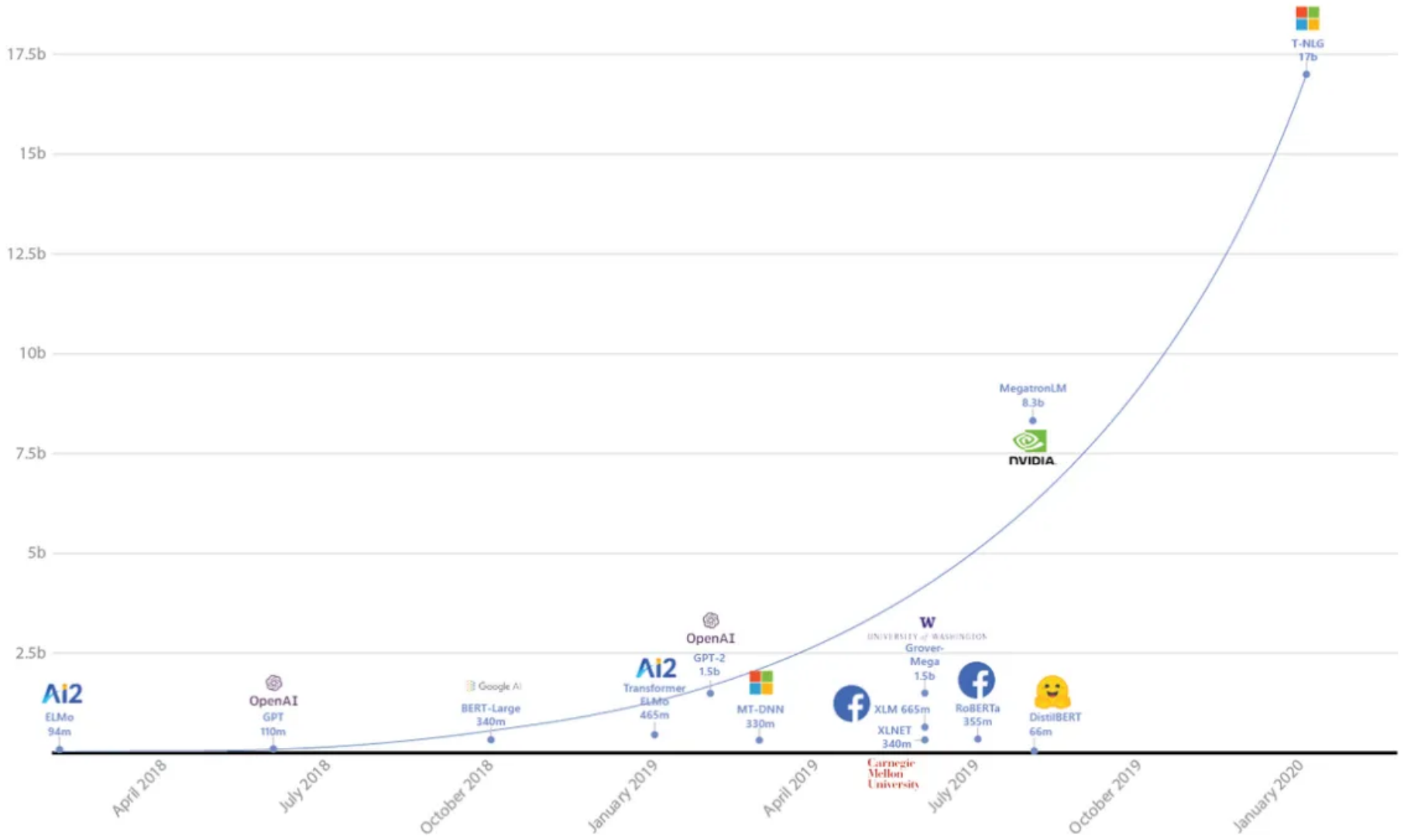
network = Sequential()

- Initialize a network
- A deep neural network is basically a series of layers of "neurons".
- A 3-layer fully connected network.
- Known as multilayer perceptron (MLP) or vanilla neural network.



https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagtion-ccf8cf369f76

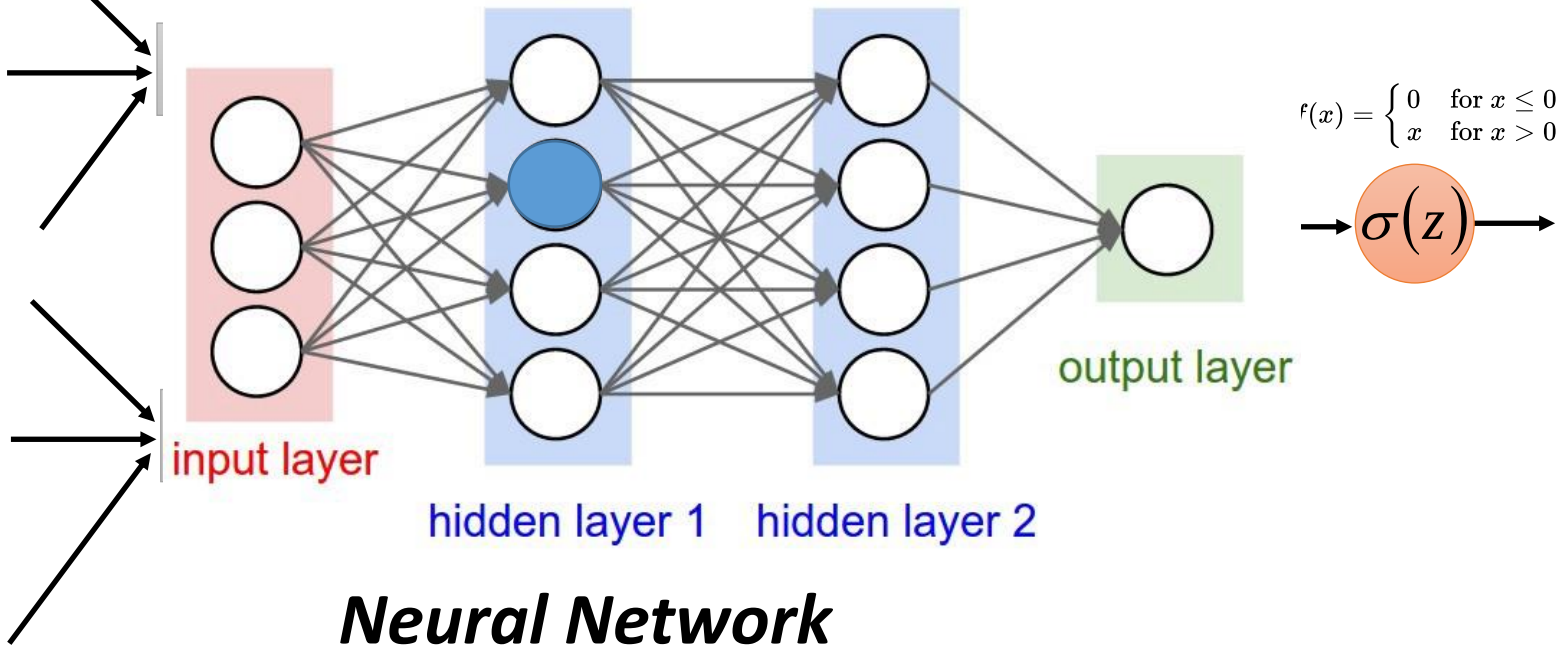
MaxPool 3x3+ 2(S)
Conv 1x1+ 1(S)
Conv 3x3+ 1(S)
Conv 5x5+ 1(S)
MaxPool 3x3+ 1(S)
DepthConcat
AveragePool 5x5+ 3(V)
FC
SoftmaxActivation
softmax0
softmax1


17.5b
15b
12.5b
10b
7.5b
5b
2.5b
T-NLG
17b
MegatronLM
8.3b
NVIDIA
OpenAI
GPT-2
1.5b
Ai2
Transformer
ELMo
465m
Ai2
ELMo
94m
OpenAI
GPT
110m
Google AI
BERT-Large
340m
MT-DNN
330m
Grover-
Mega
1.5b
XLM 665m
XLNET
340m
RoBERTa
355m
DistilBERT
66m
Carnegie
Mellon
University
April 2018
July 2018
October 2018
January 2019
April 2019
July 2019
October 2019
January 2020

A close look of neural network

$y = w_0 + w_1 \cdot x$

dot(W, input) + b



$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$$

Neural Network

$$y = fnn(x) = f_3(f_2(f_1(x))$$

$$f_l(z) = \sigma_l(Wlz + bl)$$

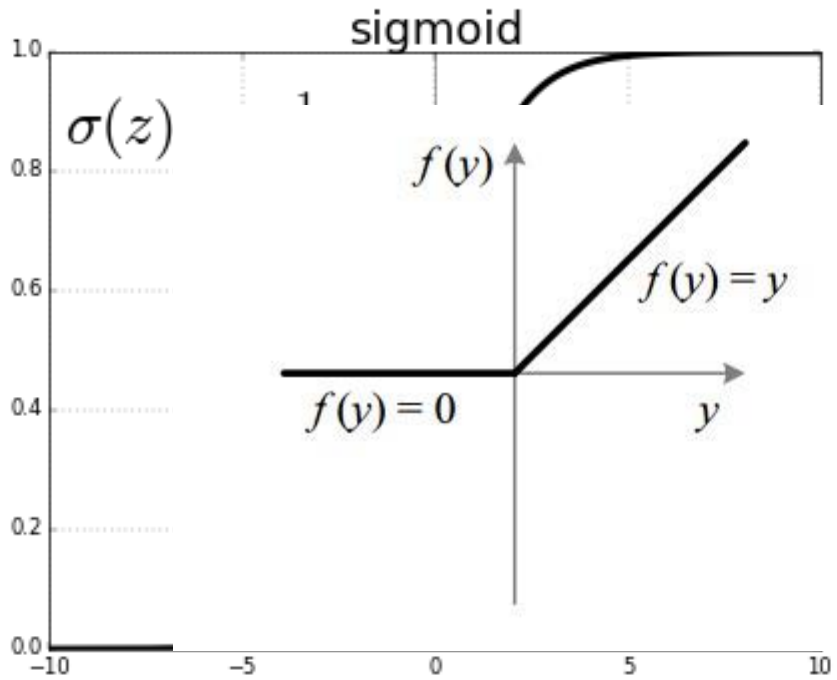
network.add(Dense(8, activation='relu', input_dim=4))

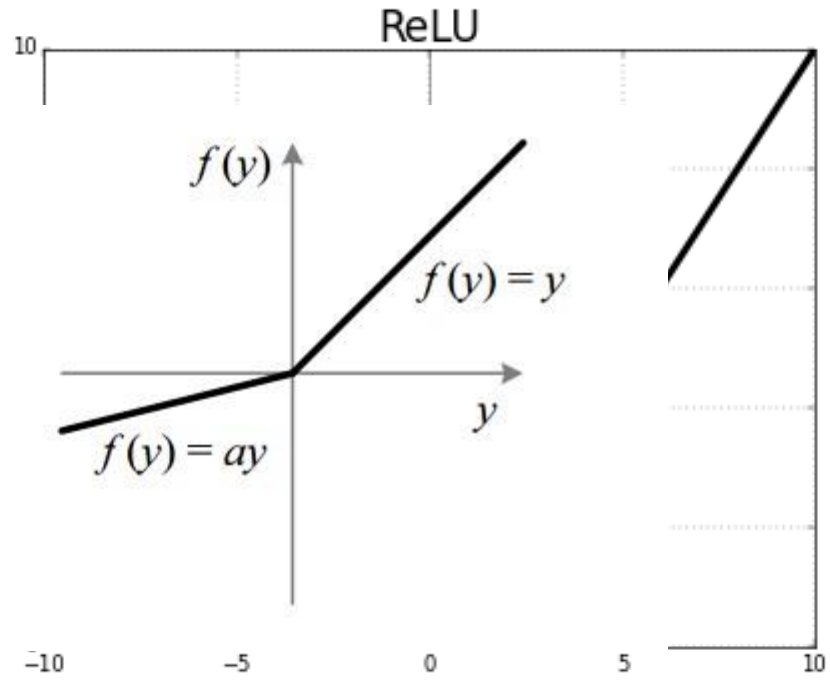
- Creating a deep neural network is like building lego set.
- You can add different layers sequentially.
- We add our first layer, which is a dense layer, with a 4 dimensional input, 8 dimensional output, and an activation function of "relu".
  - Dense means it is a densely-connected (also called "fully-connected") neural layer.
  - It takes input from our original training set, which is a 4-dimensional data.
  - It outputs an 8-dimensional result for next layer.

Activation function

- Activation function is the function to determine the final outcome of the neuron.
- It decides if the neuron is activated or not.
- Therefore, activation function is a non-linear function that generally transforms the linear operation output into a value between 0 to 1.
- Why?
  - Stacking of linear functions is still a linear function, which defeat the purpose of multiple layers.

Common activation functions



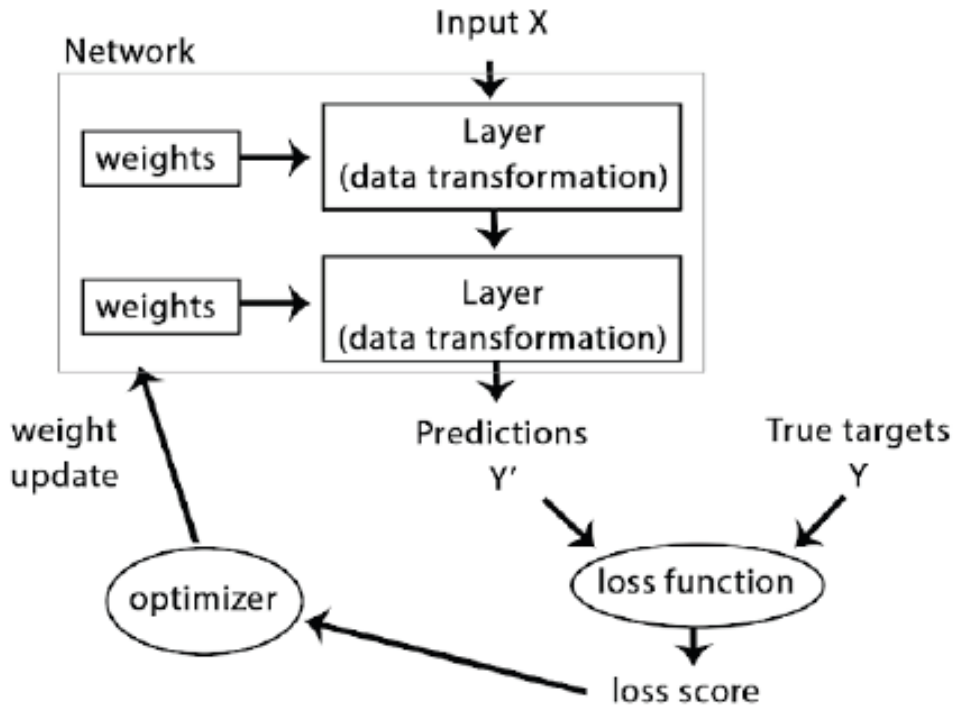


network.add(Dense(3, activation='softmax'))

- We further add the second layer, which is also the last layer, the output layer.
  - It is still a dense layer.
  - From the second layer, you don't need to specify the input shape as the model will automatically determine it from the previous layer.
  - Since this is the last layer, we aim to produce 3 groups of results. Thus the output dimension is 3.
  - We use an activation function 'softmax', which is normally only used in the last layer for classification problems.

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

- Overall flow



Loss function

- Loss function is a function used to assess how well does the model perform in terms of predicting the expected results. Most commonly used include:
  1. MSE (Mean Square Error)
  2. Binary_crossentropy
  3. Categorical_crossentropy

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multi-class, single-label classification	`softmax`	`categorical_crossentropy`
Multi-class, multi-label classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	`None`	`mse`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

Learning: Weight update

For each neuron, output = relu(dot(W, input) + b). Initially, W and b will be assigned randomly. They will then get updated based on the loss and optimizer.

Gradient-based optimization

Optimizer

- Adam
- Gradient Descent (SGD, Mini Batch, etc.)
- Momentum

Tweaking the network

- Develop a model that does better than a baseline.
    1. Choice of the last-layer activation.
    2. Choice of loss function.
    3. Choice of optimization configuration.
- Scale up: develop a model that overfits
    1. Add layers.
    2. Make your layers bigger.
    3. Train for more epochs.
- Regularize your model and tune your hyperparameters.
- Dropout
- Different architectures.
- Different hyperparameters.
- Iterate feature engineering.

Review

Data Collection

To-do	Tools
Data retrieve via SQL queries	BigQuery, MySQL, etc.
Web Scraping	Requests, BeautifulSoup, etc.
Data access via JSON	json, csv

Data Collection

To-do	Tools
Data retrieve via SQL queries	BigQuery, MySQL, etc.
Web Scraping	Requests, BeautifulSoup, etc.
Data access via JSON	json, csv

Data Visualization

To-do	Tools
Visual inspection	Matlibplot, Seaborn, etc.
Presenting results	Tableau, Google Data Studio

Data Processing

To-do	Tools
Load the dataset	Pandas, csv, etc
Deal with missing data	Pandas: dropna(), fillna() Sklearn imputer.
Wrangling	Pandas: drop(), get_dummies(), cut(), .str.extract() Sklearn.preprocessing: SimpleImputer, StandardScaler, MinMaxScaler, OneHotEncoder, QCA, etc.

Machine Learning

To-do	Tools
Unsupervised Learning: clustering ✓ n_clusters, ✓ max_iter, ✓ Etc.	sklearn.cluster: KMeans
Supervised Learning: classification ✓ Decision Tree ✓ Ensemble ✓ criterion, max_depth, etc.	sklearn.tree: DecisionTreeClassifier sklearn.ensemble: RandomForestClassifier, AdaBoostClassifier, StackingClassifier, etc.
Supervised Learning: regression ✓ Linear regression ✓ Tree regressor	sklearn.linear_model: LinearRegression sklearn.tree: DecisionTreeRegressor

Machine Learning

To-do	Tools
Model selection ✓ Metrics • accuracy, recall, precision, f1, etc. • mse, mae, etc. ✓ Cross validation ✓ Grid Search	sklearn.metrics: accuracy_score, recall_score, precision_score, f1_score, etc. sklearn.metrics: mean_absolute_error, mean_sequared_error, etc. Sklearn.model_selection: Kfold, StratifiedKFold Sklearn.model_selection: GridSearchCV
Sklearn pipeline	ColumnTransformer, FunctionTransformer, Pipeline, etc.
Deep learning	TensorFlow, PyTorch, Keras, etc.