## Neural Networks Types and Main Features

| | |
|---|---|
| Feedforward neural network | connections between nodes do not have a cycle |
| Multilayer perceptron (MLP) | has at least three layers of nodes |
| Reccurent neural network (RNN) | connections between units have a directed cycle |
| Self-Organising Maps (SOM) | convert input data to low dimensional space |
| Deep Belief Network (DBN) | has connections between layers but not within layer |
| Convolutional Neural Network (CNN) | has one or more convolutional layers and then followed by one or more fully connected layers |
| Generative Adversarial Networks (GAN) | system of two neural nets, contesting with each other |
| Spiking Neural Netorks (SNN) | time information is processed in the form of spikes and there is more than one synapse between neurons |
| Wavelet neural network | use wavelet function as activation function in the neuron |

## Neural Networks Types and Main Features (cont)

| | |
|---|---|
| Wavelet convolutional neural network | combine wavelet transform and CNN |
| Long short-term memory (LSTM) | type of RNN, model for the short-term memory which can last for a long period of time |

## Building Neural Network with Keras and Python

```
from keras.models import Sequential
model = Sequential()
from keras.layers import Dense
model.add(Dense(units=64,
activation='relu', input_dim=100))
model.add(Dense(units=10,
activation='softmax'))
model.compile(loss='categorical_cro
ssentropy',
              optimizer='sgd',
              metrics=
['accuracy'])
model.compile(loss=keras.losses.cat
egorical_crossentropy,
              optimizer=keras.opti
mizers.SGD(lr=0.01, momentum=0.9,
nesterov=True))
model.fit(x_train, y_train,
epochs=5, batch_size=32)
model.train_on_batch(x_batch,
y_batch)
loss_and_metrics =
model.evaluate(x_test, y_test,
batch_size=128)
classes = model.predict(x_test,
batch_size=128)
```

Keras

## Data Preparation for Input to Neural Network

```
from sklearn import preprocessing
def normalize_data(m, XData):
    if m == "":
        m="scaling-no"
    if m == "scaling-no":
        return XData
    if m == "StandardScaler":
        std_scale =
preprocessing.StandardScaler().fit(
XData)
        XData_new =
std_scale.transform(XData)
    if m == "MinMaxScaler":
        minmax_scale =
preprocessing.MinMaxScaler().fit(XD
ata)
        XData_new =
minmax_scale.transform(XData)
    return XData_new
```

## Cheat Sheets about Python and Machine Learning

Quick and Easy Way to get started with common and most used python tasks in data processing

## Neural Network Applications and Most Used Networks

| | |
|---|---|
| Image classification | CNN |
| Image recognition | CNN |
| Time series prediction | RNN, LSTM |
| Text generation | RNN, LSTM |
| Classification | MLP |
| Visualization | SOM |

## Neural Net Weight Update Methods

| | |
|---|---|
| Adam | based on adaptive estimates of lower order moments |
| AdaGrad | Adagrad is an adaptive learning rate method |
| RMSProp | adaptive learning rate method, modification of Adagrad method |
| SGD | Stochastic gradient descent |
| AdaDelta | modification of Adagrad to reduce its aggressive, monotonically decreasing learning rate |
| Newton method | second order method, is not used in deep learning |
| Momentum | method that helps accelerate SGD in the relevant direction |
| Nesterov accelerated gradient | evaluate the gradient at next position instead of current |

References:
ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION
Convolutional Neural Networks for Visual Recognition.
An overview of gradient descent optimization algorithms
Wikipedia -Artificial neural network

## Links