Sonar Returns: Data preparation, tuning topology

BROWNLEE 11

YUQI LIU

Content

1. Problem Formulation: Sonar Returns
2. Keras + Theano Implementation
3. Improvements (Feature Engineering & Parameter Tuning)

Problem Formulation

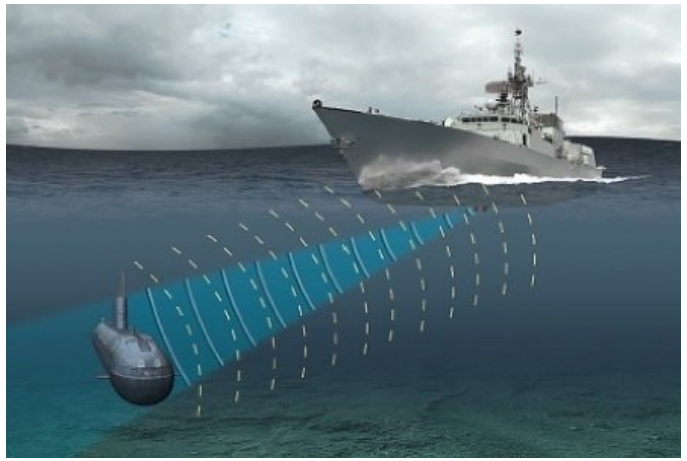
input

Sonar returns
60 strength values
from different angles

Neural Network

output

M mine
R rock



Problem Formulation

input

Sonar returns
60 strength values
from different angles

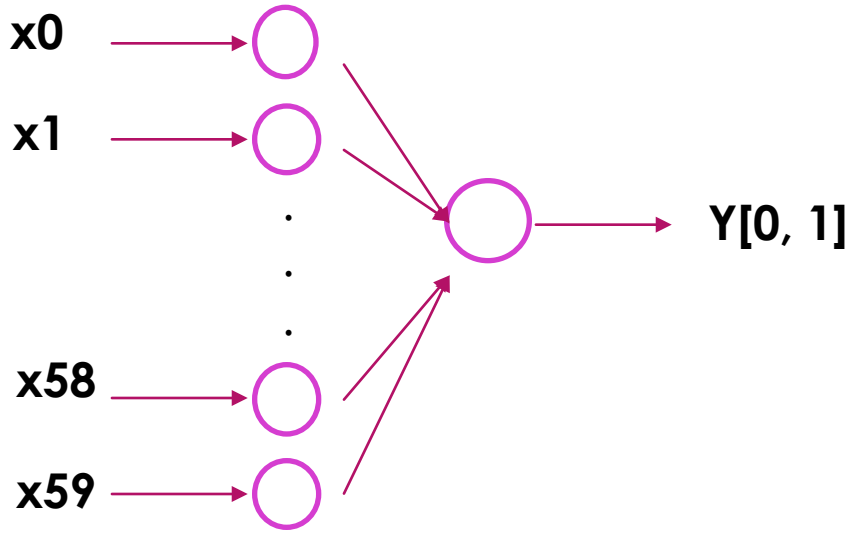
Neural Network

output

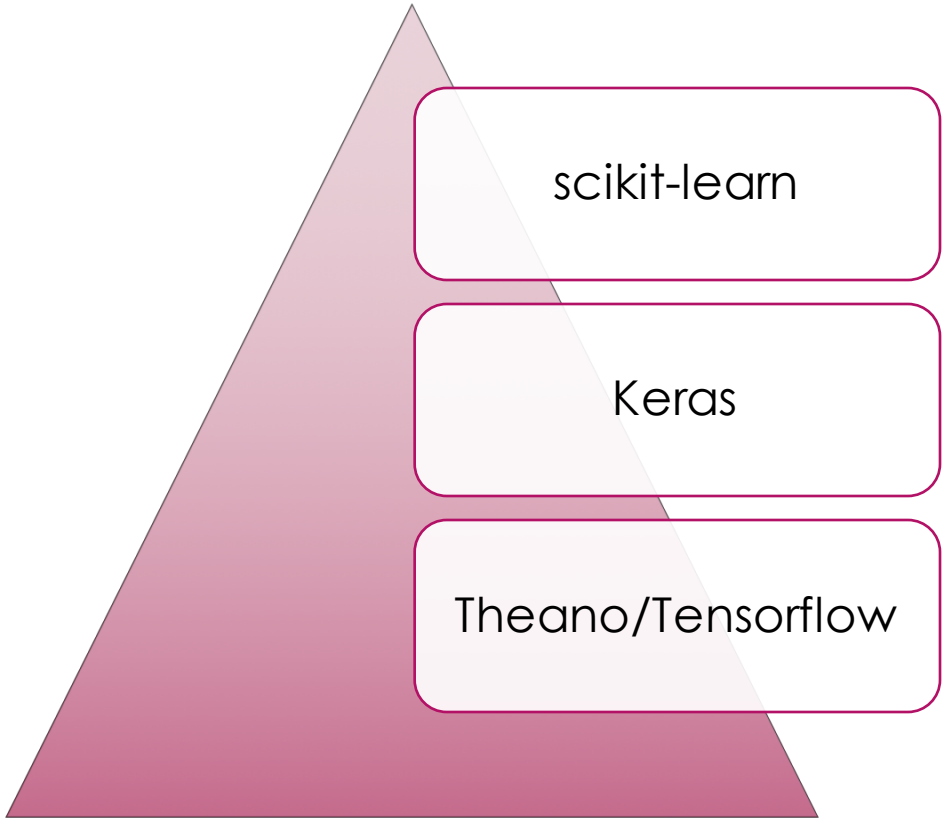
M mine
R rock

Binary Classification:

X[x0, x1, x2 ... x59] → Y[0, 1]



Keras + Theano + scikit-learn Pyramid



Wrapper for Keras
Easy grid-search / k-fold evaluation

Wrapper for Theano/Tensorflow,
Easy to create model and structure

Library for machine learning
Optimized numerical computation on the CPU or GPU

Keras + Theano

Define Model.
Add hidden layer.
Add output layer.
Compile model

```
# baseline model
def create_baseline():
	# create model
	model = Sequential()
	model.add(Dense(60, input_dim=60, init= normal , activation= relu ))
	model.add(Dense(1, init= normal , activation= sigmoid ))
	# Compile model
	model.compile(loss= binary_crossentropy , optimizer= adam , metrics=[ accuracy ])
	return model
```

scikit-learn to evaluation model

```
# evaluate model with standardized dataset
estimator = KerasClassifier(build_fn=create_baseline, nb_epoch=100, batch_size=5, verbose=0)
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
results = cross_val_score(estimator, X, encoded_Y, cv=kfold)
```

DATA PREPARATION

Scale the value to Standardization where mean is 0 and var is 1.

$X = (X - X_{mean}) / VAR(X)$

```
# evaluate baseline model with standardized dataset
estimators = [] estimators.append(( standardize , StandardScaler()))
estimators.append(( mlp , KerasClassifier(build_fn=create_baseline,
nb_epoch=100, batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
```

Categorical: convert to numerical label

Aggregation / Decomposition: preprocess the data you have

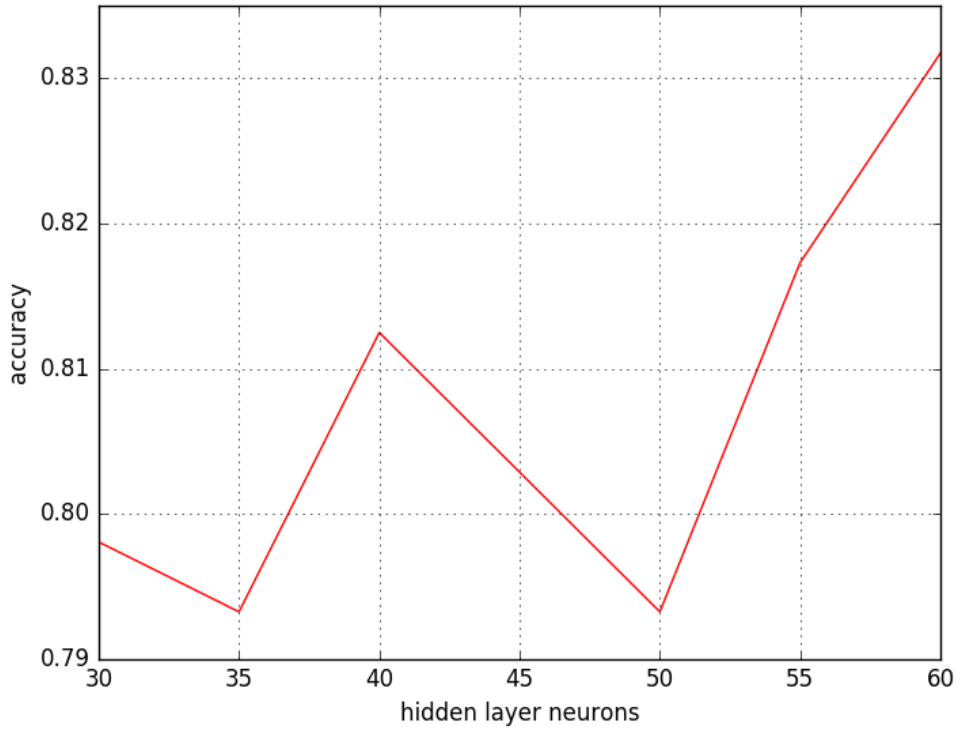
Periodic Variable: convert it to numerical variable such as sin/cos

Hyperparameter Tuning

Turn the hyperparameter of the network. The number of layers and the size of neurons.

With standardization:

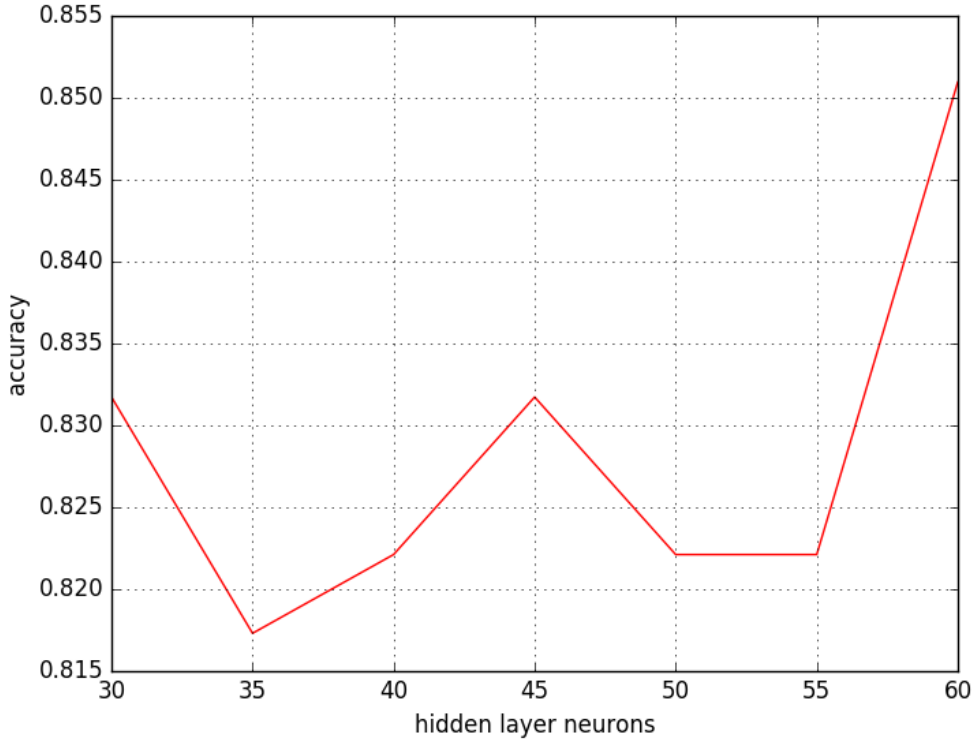
60 input → [60] → 1 output	84.07%
60 input → [30] → 1 output	84.61%
60 input → [60, 30] → 1 output	86.47%



Hyperparameter Tuning

Turn the hyperparameter of the network. The number of layers and the size of neurons.

With standardization:	
60 input → [60] → 1 output	84.07%
60 input → [30] → 1 output	84.61%
60 input → [60, 30] → 1 output	86.47%



Takeaway Messages

1. There are many open-source deep learning library to use!
   Theano, TensorFlow, Chainer, Torch, Caffe, cuda-convnet

2. Feature Engineering is important!