Week 10: Neural networks

M P Gururajan, Hina Gokhale and N N Viswanathan

Department of Metallurgical Engineering & Materials Science
Indian Institute of Technology Bombay

October, 2024

Outline

Logistic regression: an example

ICI: complete data

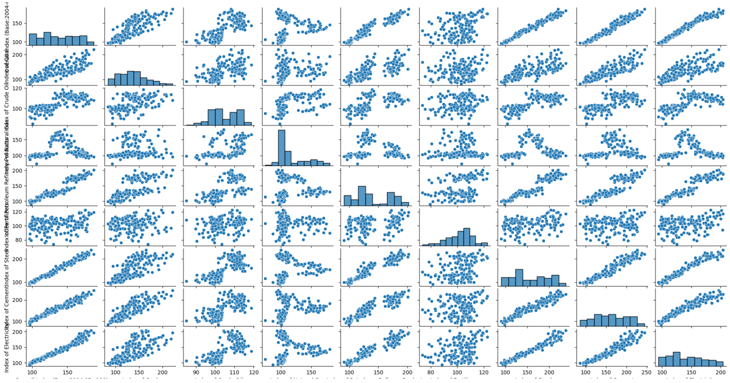


Figure: Is there a linear combination of indices which can be a good indicator of overall index?

Logistic regression

```
...
ScaledResponse[ScaledResponse<0.5] = 0
ScaledResponse[ScaledResponse>0.5] = 1
...
from sklearn.linear_model import LogisticRegression
mlfit = LogisticRegression().fit(X_train,y_train.ravel())
print(mlfit.score(X_test,y_test.ravel()))
```

Neural networks: introduction

Neural networks (NN)

- Idea: reverse engineering how brain processes signals
- Terminology: neurons, axons, ...
- NN: resemble sophisticated linear regression in their mathematical formulation
- Deep learning: well devised neural networks
- Foundation of digital assistants such as Siri, Alexa, ... and real time translation ...

Perceptron

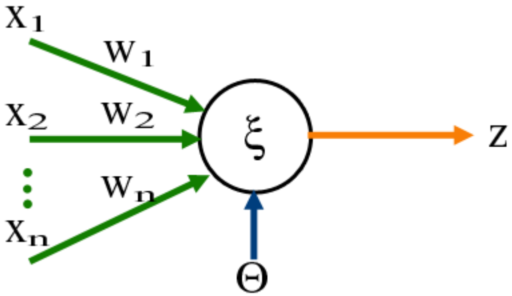


Figure: Perceptron: image from Wikimedia commons by Karla Štěpánová. Θ is the threshold. There is also bias that gets added.

Neuron

- Neuron: perceptron + activation
- Activation: Output is scaled / transformed nonlinearly and is enhanced or dampened depending on a threshold
- Activation functions: tanh, logistic, binary step, rectified linear unit (ReLU), ...
- Activation function: x-axis is the input and y-axis is the transformation
- Note: importance of scaling!

Neural network

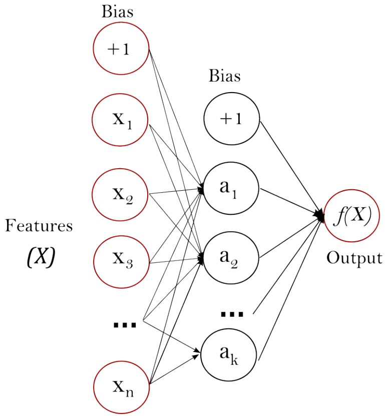


Figure: Schematic neural network from scikit-learn user guide. The first and last layers are called input and output layers; intermediate ones are called hidden layers.

Feed-forward process

- Neural network (NN): guess the target function
- NN: find the weights using the given input data
- NN: universal approximators
- Representation capability: ability to model complex functions
- Learning: means minimizing or maximizing cost or loss function
- How to optimize?

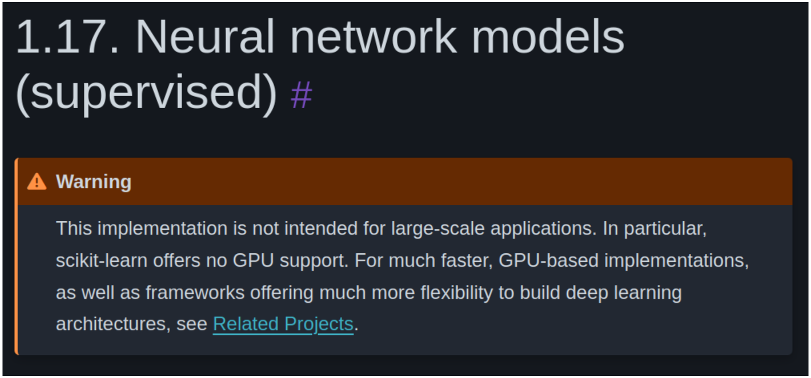
Backpropagation

- Suppose I get some error; how should I change the weight functions?
- Take the error and find out how to apportion it to different neurons
- Backpropagation algorithms: algorithms that tell how to do the error reduction by redistributing the total error to different neurons
- Backpropagation: optimization algorithms come into play
- Learning rate: rate at which we reduce the error!
- Nobel prize in physics in 2024: Hinton – backpropagation algorithm in 1986 (with David Rumelhart and Ronald J. Williams)

Tutorial introduction

- This lecture: how to code NN using scikit-learn function calls
- Theory / foundations of NN: lectures
- Actual machine learning using NN: tensorflow, keras, ...
- NN using scikit-learn: to get an idea and not for actual use
- NN: how to use for regression and classifiction?
- Using the core industry index data set as example case!

A warning



1.17. Neural network models (supervised) #

⚠ **Warning**

This implementation is not intended for large-scale applications. In particular, scikit-learn offers no GPU support. For much faster, GPU-based implementations, as well as frameworks offering much more flexibility to build deep learning architectures, see Related Projects.

Figure: Warning in the scikit-learn user guide.

MLP regression

- MLP: multi-layer pereceptron
- Before MLP
  - Extract data;
  - Scale data; and,
  - Perform PCA.

MLP Regression

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    SRF, ScaledResponse, test_size=0.2, random_state=2)
from sklearn.neural_network import MLPRegressor
nnreg = MLPRegressor(random_state=1,max_iter=500).fit
(X_train,y_train.ravel())
print(nnreg.score(X_test,y_test.ravel()))
from sklearn.model_selection import cross_val_score
nnreg = MLPRegressor(random_state=1,max_iter=500)
scores = cross_val_score(nnreg,SRF,ScaledResponse.ravel(),cv=5)
print(scores)
from sklearn.model_selection import ShuffleSplit
cv = ShuffleSplit(n_splits=10,test_size=0.3,random_state=0)
scores = cross_val_score(nnreg,SRF,ScaledResponse.ravel(),cv=cv)
print(scores)
```

MLP classification

- MLP: multi-layer pereceptron
- Before MLP
  - Extract data;
  - Scale data;
  - Threshold response to 0 or 1 so that it can be used for classification; and,
  - Perform PCA.

MLP Classification

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(SRF,
ScaledResponse, test_size=0.2, random_state=2)
from sklearn.neural_network import MLPClassifier
nncls = MLPClassifier(solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(5, 2), random_state=1)
nncls.fit(X_train,y_train.ravel())
print(nncls.score(X_test,y_test.ravel()))
from sklearn.model_selection import cross_val_score
nncls = MLPClassifier(solver='lbfgs', alpha=1e-5,
hidden_layer_sizes=(5, 2),random_state=1)
scores = cross_val_score(nncls,SRF,ScaledResponse.ravel(),cv=5)
print(scores)
from sklearn.model_selection import ShuffleSplit
cv = ShuffleSplit(n_splits=10,test_size=0.3,random_state=0)
scores = cross_val_score(nncls,SRF,ScaledResponse.ravel(),
cv=cv)
print(scores)
```

Options

- Solvers: sgd (stochastic gradient descent), adam, lbfgs (limited memory BFGS)
- α: term for L2 regularisation (penalty term); penalise weights with large magntitudes so that overfitting is avoided
- Hidden layer: how many hidden layers and how many neurons in each
- Activation: logistic, tanh, relu ...

Thank You!

Questions, clarifications, comments?