Neuronske mreže

Predavanje 2: Softverske biblioteke i alati za neuronske mreže

Predmet: Inteligentni sistemi
Prof dr Zoran Ševarac

Univerzitet u Beogradu, Fakultet organizacionih nauka

Biblioteke i softverski okviri za neuronske mreže

Neural Net - R - https://cran.r-project.org/web/packages/neuralnet/

SciKit Learn - Python https://scikit-learn.org

Neuroph - Java https://neuroph.sourceforge.net/

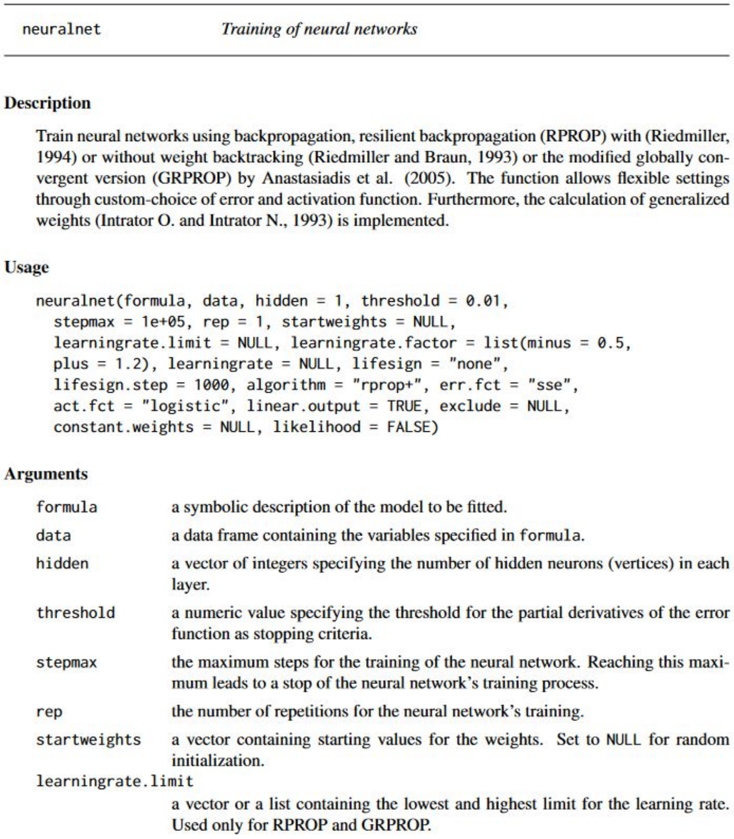
Deep Netts - Java https://www.deepnetts.com/

Tensorflow/Keras - C++/Python/Java https://www.tensorflow.org/

R - neuralnet package

```
# kreiraj i istreniraj n. mrežu
nnet <-  neuralnet(formula,
                   dataset,
                   err.fct="ce",
                   linear.output=FALSE,
                   likelihood=TRUE)

# iscrtaj graf neuronske mreže
plot(nnet)

# ispisi sve predikcije mreze
prediction(nnet)

# ispiši sve detalje u vezi neuronske mreže
print(nnet)
```

https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf



neuralnet *Training of neural networks*

Description

Train neural networks using backpropagation, resilient backpropagation (RPROP) with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version (GRPROP) by Anastasiadis et al. (2005). The function allows flexible settings through custom-choice of error and activation function. Furthermore, the calculation of generalized weights (Intrator O. and Intrator N., 1993) is implemented.

Usage

```
neuralnet(formula, data, hidden = 1, threshold = 0.01,
  stepmax = 1e+05, rep = 1, startweights = NULL,
  learningrate.limit = NULL, learningrate.factor = list(minus = 0.5,
  plus = 1.2), learningrate = NULL, lifesign = "none",
  lifesign.step = 1000, algorithm = "rprop+", err.fct = "sse",
  act.fct = "logistic", linear.output = TRUE, exclude = NULL,
  constant.weights = NULL, likelihood = FALSE)
```

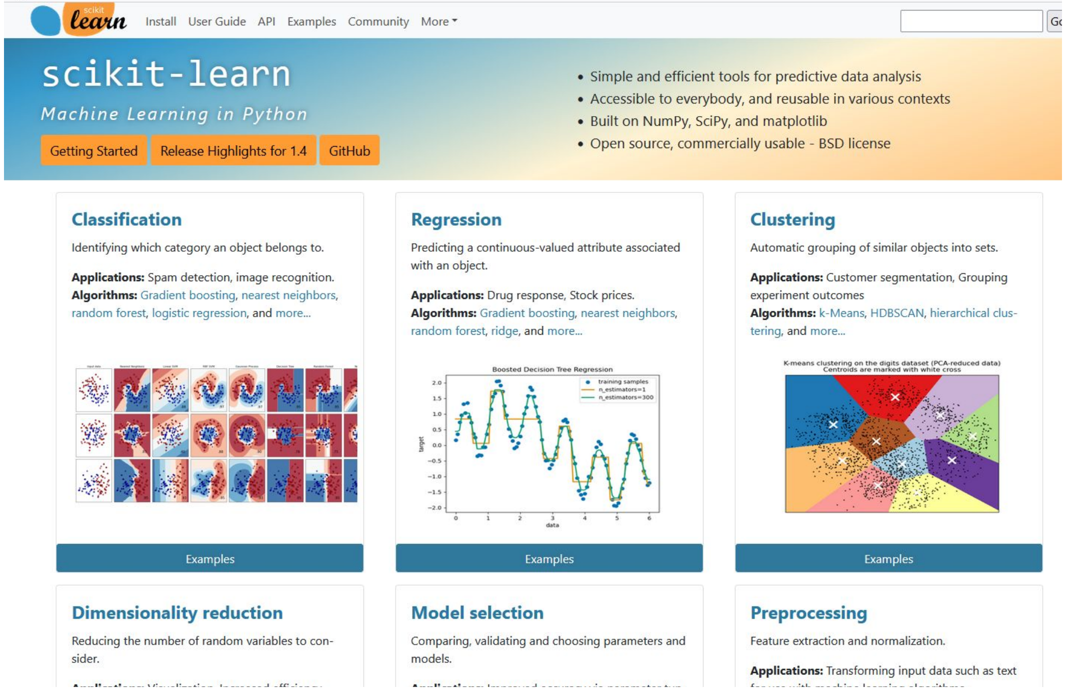
Arguments

formula	a symbolic description of the model to be fitted.
data	a data frame containing the variables specified in formula.
hidden	a vector of integers specifying the number of hidden neurons (vertices) in each layer.
threshold	a numeric value specifying the threshold for the partial derivatives of the error function as stopping criteria.
stepmax	the maximum steps for the training of the neural network. Reaching this maximum leads to a stop of the neural network's training process.
rep	the number of repetitions for the neural network's training.
startweights	a vector containing starting values for the weights. Set to NULL for random initialization.
learningrate.limit	a vector or a list containing the lowest and highest limit for the learning rate. Used only for RPROP and GRPROP.

Scikit learn

https://scikit-learn.org/stable/

- Python
- Laka za učenje i korišćenje
- Praktično standardna,
- Veliki broj algoritama za:
  - Klasifikaciju
  - Regresiju
  - Klasterizaciju
  - Analizu i pripremu podataka
  - Izbor modela
- Za manje skupove podataka
- Podrška za neuronske mreže
  - MLPClassifier
  - MLPRegressor



scikit-learn

Machine Learning in Python

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

Dimensionality reduction

Reducing the number of random variables to consider.

Model selection

Comparing, validating and choosing parameters and models.

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text

Višeslojni perceptron za klasifikaciju sa SciKitLearn

```
from sklearn.neural_network import MLPClassifier

from sklearn.datasets import make_classification

from sklearn.model_selection import train_test_split

X, y = make_classification(n_samples=100, random_state=1) # kreiranje dataseta

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=1)

clf = MLPClassifier(random_state=1, max_iter=300) # configure MLP for classification

cls.fit(X_train, y_train) # train model

clf.predict_proba(X_test[:1]) # predict probabilities

clf.predict(X_test[:5, :]) # predict class

clf.score(X_test, y_test) # test
```

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Višeslojni perceptron za regresiju sa SciKitLearn

```
diabetes = load_diabetes()
X = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
y = diabetes.target
mlp = make_pipeline(
     StandardScaler(),
     MLPRegressor(hidden_layer_sizes=(100, 100), tol=1e-2, max_iter=500, random_state=0),
)
mlp.fit(X, y)
```

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

Neuroph

https://neuroph.sourceforge.net/

Napisan u Java-i

Nastao na FON-u!

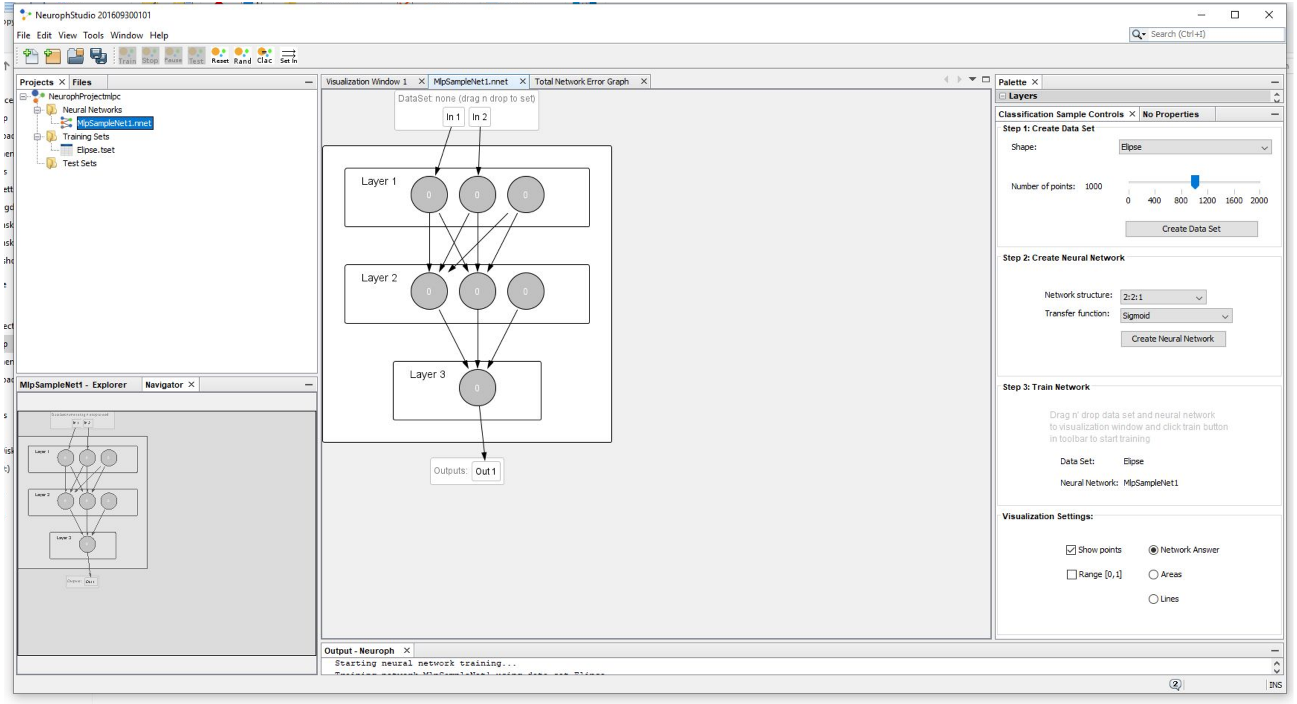
Jednostavan za učenje i razumevanje

Ima grafičko okruženje zasnovano na NetBeans-u

Ograničen na jednostavnije/manje modele i količine podataka

(nema mutithreaded/vector/gpu podršku)

Multi Layer Perceptron in Neuroph Studio



Step 1: Create Data Set

Shape: Elipse

Number of points: 1000

Step 2: Create Neural Network

Network structure: 2:2:1

Transfer function: Sigmoid

Step 3: Train Network

Drag n' drop data set and neural network to visualization window and click train button in toolbar to start training

Data Set: Elipse

Neural Network: MlpSampleNet1

Visualization Settings:

☑ Show points ● Network Answer

□ Range [0,1] ○ Areas

○ Lines

```
Starting neural network training...
Training network MlpSampleNet1 using data set Elipse
```

Osnovne komponente/klase Neuroph okvira

- MultiLayerPerceptron extends NeuralNetwork
- DataSet
- BackPropagation extends LearningRule
- Layer
- Neuron
- Connection
- Weight

Neuroph za klasifikaciju sa višeslojnim perceptronom

```
MultiLayerPerceptron neuralNet = new MultiLayerPerceptron(inputsCount, 30, 25, outputsCount);

BackPropagation learningRule = (BackPropagation) neuralNet.getLearningRule();

learningRule.setLearningRate(0.1);

learningRule.setMaxError(0.01);

neuralNet.learn(trainingSet);
```

https://github.com/neuroph/NeurophFramework/blob/master/neuroph/Samples/src/main/java/org/neuroph/samples/standard10ml/Ionosphere.java

Deep Netts

Evolucija Neuroph-a sa fokusom na primenu i poboljšanje perfromansi

Community Edition / open source
https://github.com/deepnetts/deepnetts-communityedition

Pro Edition / Besplatan za eksperimentisanje, opcije za komercijalnu podršku i produkciju

https://www.deepnetts.com/

Podrška za FeedForward i konvolucione mreže i backpropagation algoritam.

Osnovne komponente/klase Deep Netts biblioteke

FeedForwardNetwork extends NeuralNetwork

Layer

BackproagationTrainer

(builder pattern)

Dodavanje novih lejera i arhitektura relativno jednostavno

Klasa Tenzor (vektorska implementacija)

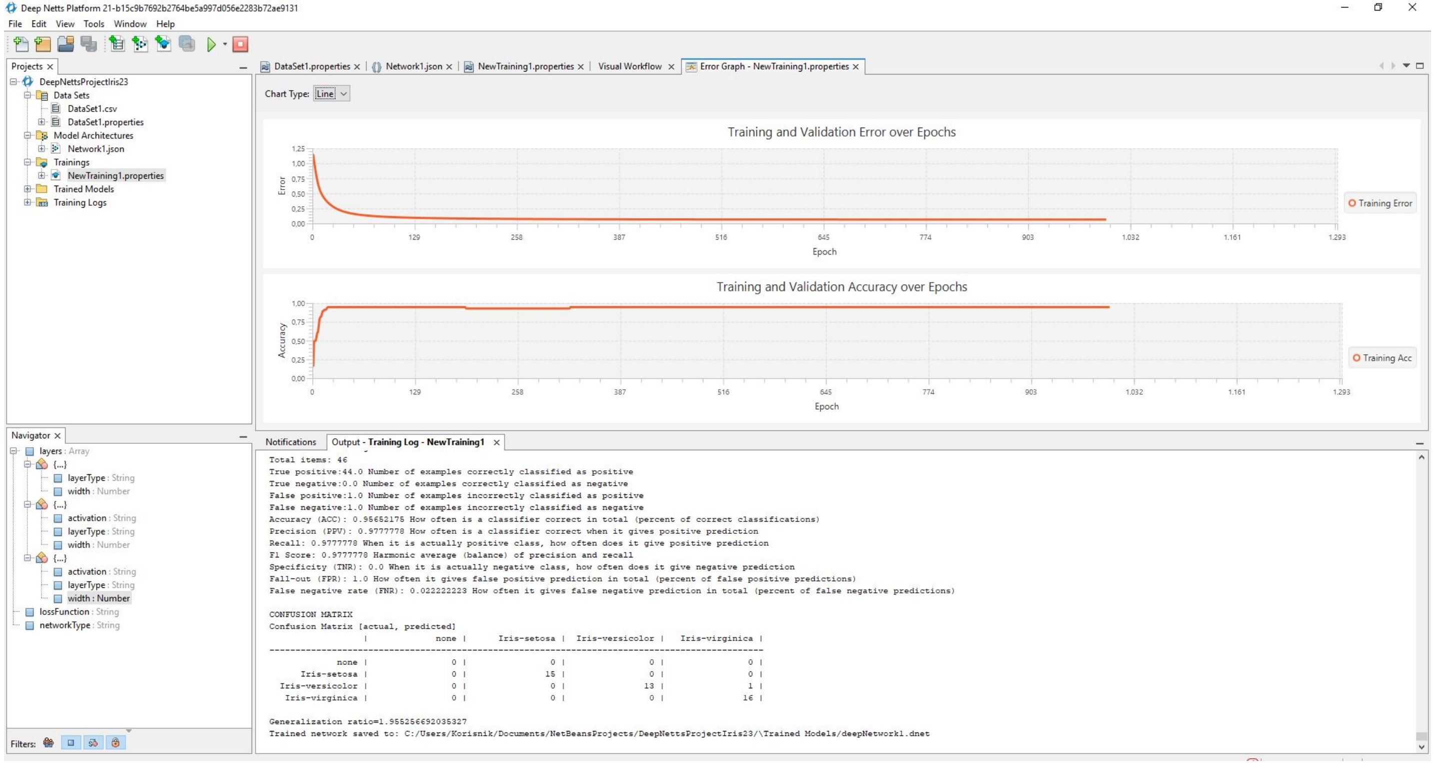
https://github.com/deepnetts/deepnetts-communityedition
https://github.com/deepnetts/deepnetts-communityedition/tree/community-visrec/deepnetts-examples/src/main/java/deepnetts/examples

Classifier using Deep Netts

```
FeedForwardNetwork neuralNet = FeedForwardNetwork.builder()
                .addInputLayer(numInputs)
                .addFullyConnectedLayer(8, ActivationType.RELU)
                .addOutputLayer(numOutputs, ActivationType.SOFTMAX)
                .lossFunction(LossType.CROSS_ENTROPY)
                .build();
```

https://github.com/deepnetts/deepnetts-communityedition/blob/community-visrec/deepnetts-examples/src/main/java/deepnetts/examples/IrisFlowersClassifier.java

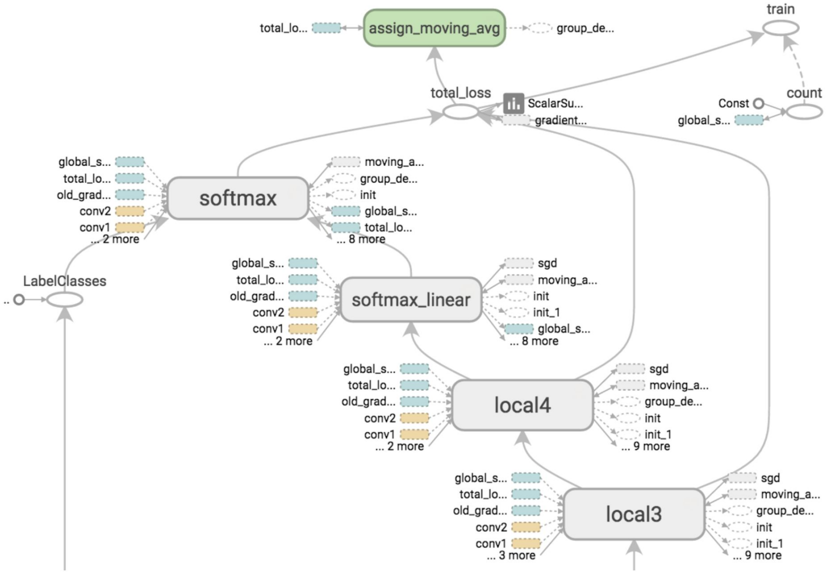
Deep Netts Visual AI Builder



```
Total items: 46
True positive:44.0 Number of examples correctly classified as positive
True negative:0.0 Number of examples correctly classified as negative
False positive:1.0 Number of examples incorrectly classified as positive
False negative:1.0 Number of examples incorrectly classified as negative
Accuracy (ACC): 0.95652175 How often is a classifier correct in total (percent of correct classifications)
Precision (PPV): 0.9777778 How often is a classifier correct when it gives positive prediction
Recall: 0.9777778 When it is actually positive class, how often does it give positive prediction
F1 Score: 0.9777778 Harmonic average (balance) of precision and recall
Specificity (TNR): 0.0 When it is actually negative class, how often does it give negative prediction
Fall-out (FPR): 1.0 How often it gives false positive prediction in total (percent of false positive predictions)
False negative rate (FNR): 0.022222223 How often it gives false negative prediction in total (percent of false negative predictions)

CONFUSION MATRIX
Confusion Matrix [actual, predicted]
                  |             none |      Iris-setosa |  Iris-versicolor |   Iris-virginica |
-----------------------------------------------------------------------------------------------
             none |                0 |                0 |                0 |                0 |
      Iris-setosa |                0 |               15 |                0 |                0 |
  Iris-versicolor |                0 |                0 |               13 |                1 |
   Iris-virginica |                0 |                0 |                0 |               16 |

Generalization ratio=1.955256692035327
Trained network saved to: C:/Users/Korisnik/Documents/NetBeansProjects/DeepNettsProjectIris23/\Trained Models/deepNetwork1.dnet
```

Keras / Tensorflow

```
model = keras.Sequential([

    keras.Input(shape=(784)),

    layers.Dense(32, activation='relu'),

    layers.Dense(10),

])

model.compile(...)

model.fit(...)
```



Koji izabrati?

Pitanja na koja treba odgovoriti

- Količina podataka
- Kako će se koristiti u produkciji
- Da li ce se koristiti za online ili offline režimu
- U koji sistem/ okruženje će se integrisati
- Da li podržava sve što je potrebno za konkretnu primenu
- Distribuirano procesiranje
- Ukupni troškovi održavanja i razvoja

Uporedni pregled

	neuralnet R	SciKit Learn	Neuroph	Deep Netts	Tensorflow
Jezik:	R	Python	Java	Java	C++/Python
Količina podataka	mala	mala	mala	velika	velika
Distribuirano procesiranje	ne	ne	ne	Da	Da
Brzina učenja	da	da	da	da	ne
Jednostavnost korišćenja	da	da	da	da	da

Šta dalje

Projektni radovi za izborne predmete

Završni radovi

Istraživački projekti u okviru Laboratorije za veštački inteligenciju

Praksa u kompanijama kroz projekte iz oblasti neuronskih mreža

Laboratorija za veštačku inteligenciju
Centar za razvoj softvera otvorenog koda