Artificial Neural Networks Crash-Course

from the practical point of view

using deeplearning4j

MultiLayerConfiguration

Configuration example

```
MultiLayerConfiguration nn_conf = new NeuralNetConfigurat
        .seed(123)
    .iterations(10)
    .learningRate(0.0001)
    .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GR
    .updater(Updater.NESTEROVS)
    .list(layers)
    .pretrain(true)
    .backprop(true)
    .build();
```

Documentation

Configuration

Core Params

`.seed(123)`

Random number generator seed. Used for example for the initial values for the nodes.

`.iterations(1 ... 10)`

How many times the NN will be optimized by pretrain & back propagation. Not exactly the same as epoch. Greater than 1 only at full-batch training. Default: 1.

`.learningRate(1e-1 ... 1e-6)`

Defines the impact of a back propagation step. Low values causes slow learning, high values may lead to miss the optimum.

Optimization Algorithms

Calculates the error by the gradient

```
.optimizationAlgo(OptimizationAlgorithm.*)
```

- LINE_GRADIENT_DESCENT
- CONJUGATE_GRADIENT
- HESSIAN_FREE *(2nd-order method)*
- LBFGS *(2nd-order method)*
- STOCHASTIC_GRADIENT_DESCENT *(default)*

Updater

The mechanism for weight updates in backpropagation. Adjust often the learning rate.

`.updater( Updater.* )`

- SGD *(stochastic gradient descent, default)*
- ADAM
- ADADELTA
- NESTEROVS *(used* `.momentum(*)` *parameter, recommended)*
- ADAGRAD
- RMSPROP
- NONE
- CUSTOM

Training

`.pretrain(true)`

Activates pretrain for all layer which are pretrain able e.g the layer types RBM and Autoencoder.

`.backprop(true)`

Activates back propgation which updates the weights of the network after every `.fit(data)` based on the evaluated error with the OptimizationAlgorithm and the Updater.

Layer options

`.activation(String)`

Activation function for the neurons. Diagrams [1] [2]

- "relu" [0, 1] (rectified linear, most popular for DNN)
- "leakyrelu"
- "tanh" (-1, 1)
- "sigmoid" (0, 1)? (default)
- "softmax"
- "hardtanh"
- "maxout"
- "softsign" (-1, 1)
- "softplus" (0, infinity)

Layer options

```
.weightInit(WeightInit.*)
```

- Distribution: Sample weights from a distribution based on shape of input
- Normalized: Normalize sample weights
- Size: Sample weights from bound uniform distribution using shape for min and max
- Uniform: Sample weights from bound uniform distribution
- VI: Sample weights from variance normalized initialization
- Zeros
- Xavier (default)
- RELU

Description from source code from deeplearning4j

Recommended configurations

Hidden Layer	Output Layer	WeightInit
relu/leakyrelu	softmax (classification)	RELU
tanh	*linear*	XAVIER

Layer options

```
.lossFunction(LossFunctions.LossFunction.RMSE_XENT)
```

Will be used for pretraining and the OutputLayer

- MSE: (Mean Squared Error, Linear Regression)
- EXPLL: (Exponential log likelihood, Poisson Regression)
- XENT (Cross Entropy, Binary Classification)
- MCXENT (Multiclass Cross Entropy, Classification)
- RMSE_XENT (RMSE Cross Entropy)
- SQUARED_LOSS
- RECONSTRUCTION_CROSSENTROPY (default)
- NEGATIVELOGLIKELIHOOD
- CUSTOM

Generalization options

.dropOut(double)

.l2(double)

.setUseRegularization(boolean)

coming soon

Layer types

RBM Layer

Restricted Boltzmann Machine

```
layera[0] = new RBM.Builder()
.nIn(100)
.nOut(150)
.lossFunction(LossFunctions.LossFunction.RMSE_XENT)
.visibleUnit(VisibleUnit.BINARY)
.hiddenUnit(HiddenUnit.BINARY)
.build()
```

More about RBM

RBM Layer

```
.visibleUnit(VisibleUnit.*).hiddenUnit(HiddenUnit.*)
```

- LINEAR (visible only)
- BINARY (default)
- GAUSSIAN
- SOFTMAX
- RECTIFIED (rectified linear units, hidden only)

Recommended configurations

visible unit	hidden unit	note	stability
BINARY	BINARY	default	++
SOFTMAX	BINARY		+
RECTIFIED	BINARY		0
GAUSSIAN	BINARY		-
GAUSSIAN	RECTIFIED	for continuous data	-
RECTIFIED	RECTIFIED		--
GAUSSIAN	GAUSSIAN		---

less stable configurations needs lower learning rates http://deeplearning4j.org/glossary.html

http://deeplearning4j.org/troubleshootingneuralnets

http://www.dkriesel.com/science/neural_networks