Deep Neuronal Filter

Generated by Doxygen 1.9.8

Chapter 1

Deep Neuronal Filter (DNF)

Libtorch version

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0277974

A noise reduction filter using deep networks in autoencoder configuration.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 DelayLine Class Reference

Public Member Functions

- void **init** (int delaySamples)
- float **process** (float input)
- float **get** (int i) const
- float **getNewest** () const

The documentation for this class was generated from the following file:

- dnf_torch.h

4.2 DNF Class Reference

Deep Neuronal Filter class.

`#include <dnf_torch.h>`

Public Types

- enum ActMethod { **Act_Sigmoid** = 1 , **Act_Tanh** = 2 , **Act_ReLU** = 3 , **Act_NONE** = 0 }

  Options for activation functions of all neurons in the network.

Public Member Functions

- DNF (const int nLayers, const int nTaps, const float samplingrate, const ActMethod am=Act_Tanh, const bool tryGPU=false)

  Constructor which sets up the delay lines, network layers and also calculates the number of neurons per layer so that the final layer always just has one neuron.
- void **setLearningRate** (float mu)
- float filter (const float signal, const float noise)

  Realtime sample by sample filtering operation.
- int getSignalDelaySteps () const

  Returns the length of the delay line which delays the signal polluted with noise.
- float getDelayedSignal () const

  Returns the delayed with noise polluted signal by the delay indicated by getSignalDelaySteps().
- float getRemover () const

  Returns the remover signal.
- float getOutput () const

  Returns the output of the DNF: the the noise free signal.
- ~**DNF** ()

  Frees the memory used by the DNF.
- std::vector< float > getLayerWeightDistances () const

  Gets the weight distances per layer.
- float getWeightDistance () const

  Gets the overall weight distsance.

4.2.1 Detailed Description

Deep Neuronal Filter class.

It's designed to be as simple as possible with only a few parameters as possible.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 DNF()

```
DNF::DNF (
            const int nLayers,
            const int nTaps,
            const float samplingrate,
            const ActMethod am = Act_Tanh,
            const bool tryGPU = false )
```

Constructor which sets up the delay lines, network layers and also calculates the number of neurons per layer so that the final layer always just has one neuron.

Parameters

nLayers	Number of layers
nTaps	Number of taps for the delay line feeding into the 1st layer
samplingrate	Sampling rate of the signals used in Hz.
am	The activation function for the neurons. Default is tanh.
tryGPU	Tries to do the learning on the GPU.

4.2.3 Member Function Documentation

4.2.3.1 filter()

```
float DNF::filter (
            const float signal,
            const float noise )
```

Realtime sample by sample filtering operation.

Parameters

signal	The signal contaminated with noise. Should be less than one.
noise	The reference noise. Should be less than one.

Returns

The filtered signal where the noise has been removed by the DNF.

4.2.3.2 getDelayedSignal()

```
float DNF::getDelayedSignal ( ) const  [inline]
```

Returns the delayed with noise polluted signal by the delay indicated by getSignalDelaySteps().

Returns

The delayed noise polluted signal sample.

4.2.3.3 getLayerWeightDistances()

```
std::vector< float > DNF::getLayerWeightDistances ( ) const
```

Gets the weight distances per layer.

Returns

The Eucledian weight distance in relation to the initial weights.

4.2.3.4 getOutput()

```
float DNF::getOutput ( ) const  [inline]
```

Returns the output of the DNF: the the noise free signal.

Returns

The current output of the DNF which is idential to filter().

4.2.3.5 getRemover()

```
float DNF::getRemover ( ) const  [inline]
```

Returns the remover signal.

Returns

The current remover signal sample.

4.2.3.6 getSignalDelaySteps()

```
int DNF::getSignalDelaySteps ( ) const  [inline]
```

Returns the length of the delay line which delays the signal polluted with noise.

Returns

Number of delay steps in samples.

4.2.3.7 getWeightDistance()

```
float DNF::getWeightDistance ( ) const
```

Gets the overall weight distsance.

Returns

The sum of all layer weight distances.

The documentation for this class was generated from the following file:

- dnf_torch.h

Chapter 5

File Documentation

5.1 dnf_torch.h

```
00001
00007 #ifndef _DNF_H
00008 #define _DNF_H
00009
00010 #include <stdio.h>
00011 #include <stdlib.h>
00012 #include <math.h>
00013 #include <assert.h>
00014 #include <torch/torch.h>
00015 #include <thread>
00016 #include <iostream>
00017 #include <deque>
00018
00019 #ifdef NDEBUG
00020 const bool debugOutput = false;
00021 #else
00022 const bool debugOutput = true;
00023 #endif
00024
00025 class DelayLine {
00026 public:
00027     void init(int delaySamples) {
00028         delaySamples_ = delaySamples;
00029         buffer_ = std::deque<float>(delaySamples_, 0.0f);
00030     }
00031
00032     inline float process(float input) {
00033         float output = buffer_.front();
00034         buffer_.pop_front();
00035         buffer_.push_back(input);
00036         return output;
00037     }
00038
00039     float get(int i) const {
00040     return buffer_[i];
00041     }
00042
00043     float getNewest() const {
00044     return buffer_.back();
00045     }
00046
00047 private:
00048     int delaySamples_ = 0;
00049     std::deque<float> buffer_;
00050 };
00051
00052
00058 class DNF {
00059 public:
00060
00064     enum ActMethod {Act_Sigmoid = 1, Act_Tanh = 2, Act_ReLU = 3, Act_NONE = 0};
00065
00066 private:
00067     struct Net : public torch::nn::Module {
00068     std::vector<torch::nn::Linear> fc;
00069     Net(int nLayers, int nInput, bool withBias = false);
00070     torch::Tensor forward(torch::Tensor x, ActMethod am);
00071     };
```

```
00072
00073 public:
00084     DNF(const int nLayers,
00085     const int nTaps,
00086     const float samplingrate,
00087     const ActMethod am = Act_Tanh,
00088     const bool tryGPU = false
00089     );
00090
00091     inline void setLearningRate(float mu) {
00092     for (auto& group : optimizer->param_groups()) {
00093             static_cast<torch::optim::SGDOptions&>(group.options()).lr(mu);
00094         }
00095     }
00096
00103     float filter(const float signal, const float noise);
00104
00110     inline int getSignalDelaySteps() const {
00111     return signalDelayLineLength;
00112     }
00113
00119     inline float getDelayedSignal() const {
00120     return signal_delayLine.get(0);
00121     }
00122
00127     inline float getRemover() const {
00128     return remover;
00129     }
00130
00136     inline float getOutput() const {
00137     return f_nn;
00138     }
00139
00143     ~DNF() {
00144     delete optimizer;
00145     delete model;
00146     }
00147
00152     std::vector<float> getLayerWeightDistances() const;
00153
00158     float getWeightDistance() const;
00159
00160 private:
00161
00162     void saveInitialParameters() {
00163     for (const auto& p : model->parameters()) {
00164         initialParameters.push_back(p.detach().clone());
00165     }
00166     }
00167
00168     Net* model = nullptr;
00169     torch::optim::SGD* optimizer = nullptr;
00170     std::vector<torch::Tensor> initialParameters;
00171     const int noiseDelayLineLength;
00172     const int signalDelayLineLength;
00173     const float fs;
00174     const ActMethod actMethod;
00175     DelayLine signal_delayLine;
00176     DelayLine noise_delayLine;
00177     float remover = 0;
00178     float f_nn = 0;
00179     static constexpr double xavierGain = 0.01;
00180     torch::Device device = torch::kCPU;
00181 };
00182
00183 #endif
```

Index