Lipnet

1.0.0

Generated by Doxygen 1.8.20

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 lipnet::activation_t< T, TYPE > Struct Template Reference

The activation_t struct; implementation of the activation functions.

```
#include <activation.hpp>
```

Public Types

- template<typename TT , size_t O, size_t I>
  using **matrix_t** = blaze::StaticMatrix< TT, O, I, blaze::columnMajor >
- template<typename TT , size_t N>
  using **vector_t** = blaze::StaticVector< TT, N, blaze::columnVector >

Static Public Member Functions

- template<size_t N, size_t BATCH = 1>
  static auto forward (const auto &val)

  evaluate activation function
- template<size_t N, size_t BATCH = 1>
  static auto derivative (const auto &val)

  derivative of activation function

3.1.1 Detailed Description

template<typename T, atype_t TYPE>
struct lipnet::activation_t< T, TYPE >

The activation_t struct; implementation of the activation functions.

Template Parameters

T	numerical value type
TYPE	choose the activation type

3.1.2 Member Function Documentation

3.1.2.1 derivative()

```
template<typename T , atype_t TYPE>
template<size_t N, size_t BATCH = 1>
static auto lipnet::activation_t< T, TYPE >::derivative (
             const auto & val )  [inline], [static]
```

derivative of activation function

Template Parameters

N	input dimension
BATCH	batch size

Parameters

val	input vector

Returns

output vector

3.1.2.2 forward()

```
template<typename T , atype_t TYPE>
template<size_t N, size_t BATCH = 1>
static auto lipnet::activation_t< T, TYPE >::forward (
             const auto & val )  [inline], [static]
```

evaluate activation function

Template Parameters

N	input dimension
BATCH	batch size

Parameters

val	input vector

Returns

output vector

$\sigma(x) = \frac{1}{1+\exp -x}$

$\sigma(x) = \tanh(x)$

$\sigma(x) = x$

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

- lipnet/include/lipnet/network/activation.hpp

3.2 lipnet::adam_barrier_t_impl< T, P, VAR, GRAD, feasibility_enabled > Struct Template Reference

Modified adam method for use with barrier functions; it follows the central path.

```
#include <adam_barrier.hpp>
```

Classes

- struct parameter_t

  The parameter_t struct; all meta parameters for optimisation.
- struct statistics_t

  problem specific implementation of statistics_t

Public Member Functions

- void **unpack** (std::tuple< GRAD, T > &&t, GRAD &dx, T &fx) const
- adam_barrier_t_impl (parameter_t &&param=parameter_t{(size_t) 5e5,(size_t) 5, 1e-10, 1e-8, 300, 1.0, 0.02, 0.9, 0.999, 5.0, 0.5, 0.5, 1e-8})

  Default constructor.
- template<bool stats_enabled = false, bool problem_stats_exists = statistics_helper::stats_type_exists<P>::value>
  std::tuple< VAR, T > run (P &prob, VAR &&x, typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &stats) const

  The run method. Implementation of the optimisation algorithm. Modified Adam-method.

Public Attributes

- parameter_t param

  variables to optimize

3.2.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD, bool feasibility_enabled = false>
struct lipnet::adam_barrier_t_impl< T, P, VAR, GRAD, feasibility_enabled >

Modified adam method for use with barrier functions; it follows the central path.

Template Parameters

T	numerical value type
P	problem type
VAR	variable type
GRAD	gradient type
feasibility_enabled	set this value to true if you want to enable feasibility checking

3.2.2 Constructor & Destructor Documentation

3.2.2.1 adam_barrier_t_impl()

```
template<typename T , typename P , typename VAR , typename GRAD , bool feasibility_enabled =
false>
lipnet::adam_barrier_t_impl< T, P, VAR, GRAD, feasibility_enabled >::adam_barrier_t_impl (
            parameter_t && param = parameter_t{ (size_t) 5e5, (size_t) 5, 1e-10, 1e-8, 300, 1.0, 0.02, 0.9, 0
) [inline], [explicit]
```

Default constructor.

Parameters

hyperparameter	of optimisation. Init hyperparameters with (size_t) 5e5, (size_t) 5, 1e-10, 1e-8, 300, 1.0, 0.02, 0.9, 0.999, 5.0, 0.5, 0.5, 1e-8

3.2.3 Member Function Documentation

3.2.3.1 run()

```
template<typename T , typename P , typename VAR , typename GRAD , bool feasibility_enabled =
false>
template<bool stats_enabled = false, bool problem_stats_exists = statistics_helper::stats_↩
type_exists<P>::value>
std::tuple<VAR,T> lipnet::adam_barrier_t_impl< T, P, VAR, GRAD, feasibility_enabled >::run (
            P & prob,
            VAR && x,
            typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &
stats ) const [inline]
```

The run method. Implementation of the optimisation algorithm. Modified Adam-method.

Template Parameters

stats_enabled	enable/disable logging

Parameters

prob	problem
x	start variable / inital variable / start point
stats	statistics holder [5]

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

- lipnet/include/lipnet/optimizer/adam_barrier.hpp

3.3 lipnet::adam_momentum_t_impl< T, P, VAR, GRAD > Struct Template Reference

The Adam method. [5].

```
#include <adam_momentum.hpp>
```

Classes

- struct parameter_t
- struct statistics_t

  problem specific implementation of statistics_t

Public Types

- typedef std::function< bool(const T &, const VAR &, const GRAD &)> **criterion_t**

Public Member Functions

- void **unpack** (std::tuple< GRAD, T > &&t, GRAD &dx, T &fx) const
- adam_momentum_t_impl (parameter_t &&param=parameter_t{(size_t) 5e4, 1e-10, 1e-4, 0.02, 0.9, 0.999, 1e-8}, criterion_t &&c=[ ](const T &, const VAR &, const GRAD &){return true;})

  Default constructor.
- template<bool stats_enabled = false>
  std::tuple< VAR, T > run (P &prob, VAR &&x, typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &stats) const

  The run method. Implementation of the optimisation algorithm. Adam-method.

Public Attributes

- parameter_t param

  variables to optimize
- criterion_t criterion

  custom stopping criterion

3.3.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD>
struct lipnet::adam_momentum_t_impl< T, P, VAR, GRAD >

The Adam method. [5].

Template Parameters

T	numerical value type
P	problem type
VAR	variable type
GRAD	gradient type

3.3.2 Constructor & Destructor Documentation

3.3.2.1 adam_momentum_t_impl()

```
template<typename T , typename P , typename VAR , typename GRAD >
lipnet::adam_momentum_t_impl< T, P, VAR, GRAD >::adam_momentum_t_impl (
             parameter_t && param = parameter_t{ (size_t) 5e4, 1e-10, 1e-4, 0.02, 0.9, 0.999, 1e-8},
             criterion_t && c = [](const T&,const VAR&,const GRAD&){return true;} )  [inline],
[explicit]
```

Default constructor.

Parameters

hyperparameter	of optimisation. Init hyperparameters with (size_t) 5e4, 1e-10, 1e-4, 0.02, 0.9, 0.999, 1e-8

3.3.3 Member Function Documentation

3.3.3.1 run()

```
template<typename T , typename P , typename VAR , typename GRAD >
template<bool stats_enabled = false>
std::tuple<VAR,T> lipnet::adam_momentum_t_impl< T, P, VAR, GRAD >::run (
             P & prob,
             VAR && x,
             typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &
stats ) const  [inline]
```

The run method. Implementation of the optimisation algorithm. Adam-method.

Template Parameters

stats_enabled	enable/disable logging

Parameters

prob	problem
x	start variable / inital variable / start point
stats	statistics holder [5]

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

- lipnet/include/lipnet/optimizer/adam_momentum.hpp

3.4 lipnet::adam_projected_t_impl< T, P, VAR, GRAD > Struct Template Reference

Modified Adam method. Projected Adam method. [5].

```
#include <adam_projected.hpp>
```

Classes

- struct parameter_t
- struct statistics_t

  problem specific implementation of statistics_t

Public Member Functions

- void **unpack** (std::tuple< GRAD, T > &&t, GRAD &dx, T &fx) const
- auto project (const P &prob, VAR &&var) const

  The project method. Call projection method of problem.
- adam_projected_t_impl (parameter_t &&param=parameter_t{(size_t) 1e4, 1e-7, 1e-8, 300, 0.02, 0.9, 0.999, 1e-8 })

  Default constructor.
- template< bool stats_enabled = false >
  std::tuple< VAR, T > run (P &prob, VAR &&x, typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &stats) const

  The run method. Implementation of the optimisation algorithm. Adam-method.

Public Attributes

- parameter_t param

  variables to optimize

3.4.1 Detailed Description

template< typename T, typename P, typename VAR, typename GRAD >
struct lipnet::adam_projected_t_impl< T, P, VAR, GRAD >

Modified Adam method. Projected Adam method. [5].

Template Parameters

T	numerical value type
P	problem type
VAR	variable type
GRAD	gradient type

3.4.2 Constructor & Destructor Documentation

3.4.2.1 adam_projected_t_impl()

```
template<typename T , typename P , typename VAR , typename GRAD >
lipnet::adam_projected_t_impl< T, P, VAR, GRAD >::adam_projected_t_impl (
             parameter_t && param = parameter_t{(size_t) 1e4, 1e-7, 1e-8, 300, 0.02, 0.9, 0.999, 1e-8 }
) [inline], [explicit]
```

Default constructor.

Parameters

hyperparameter	of optimisation. Init hyperparameters with (size_t) 1e4, 1e-7, 1e-8, 300, 0.02, 0.9, 0.999, 1e-8

3.4.3 Member Function Documentation

3.4.3.1 project()

```
template<typename T , typename P , typename VAR , typename GRAD >
auto lipnet::adam_projected_t_impl< T, P, VAR, GRAD >::project (
             const P & prob,
             VAR && var ) const [inline]
```

The project method. Call projection method of problem.

Parameters

prob	problem
var	current variables; will be projected to feasible set

3.4.3.2 run()

```
template<typename T , typename P , typename VAR , typename GRAD >
template<bool stats_enabled = false>
std::tuple<VAR,T> lipnet::adam_projected_t_impl< T, P, VAR, GRAD >::run (
            P & prob,
            VAR && x,
            typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &
stats ) const  [inline]
```

The run method. Implementation of the optimisation algorithm. Adam-method.

Template Parameters

stats_enabled	enable/disable logging

Parameters

prob	problem
x	start variable / inital variable / start point
stats	statistics holder [5]

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

- lipnet/include/lipnet/optimizer/adam_projected.hpp

3.5 lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL > Struct Template Reference

Alternating Direction Method of Multipliers. ADMM [1].

```
#include <admm_optimizer.hpp>
```

Classes

- struct parameter_t
- struct statistics_t

  problem specific implementation of statistics_t

Public Member Functions

- DUAL residual (const P &prob, const X &x, const Z &z) const

  compute residual $Ax + Bz - c$ [1]
- X optimize1 (const P &prob, const X &x, const Z &z, const DUAL &d) const

  optimize first subproblem. $\arg\min_x L_v(x, z^t, y^t)$ [1]
- Z optimize2 (const P &prob, const X &x, const Z &z, const DUAL &d) const

  optimize second subproblem. $\arg\min_z L_v(x^{t+1}, z, y^t)$ [1]

- T evaluate (const P &prob, const X &x, const Z &z) const

  evaluate augmented lagrangian
- admm_optimizer_t_impl (parameter_t &&param=parameter_t{(size_t) 1e4, 2, 1e-1})

  Default constructor.
- template< bool stats_enabled = false>
  std::tuple< X, Z, T > run (P &prob, X &&x, Z &&z, typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &stats) const

  The run method. Implementation of the optimisation algorithm. Adam-method.

Public Attributes

- parameter_t param

  variables to optimize

3.5.1 Detailed Description

template< typename T, typename P, typename X, typename Z, typename DUAL>
struct lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >

Alternating Direction Method of Multipliers. ADMM [1].

Template Parameters

T	numerical value type
P	problem type
X	first variable type
Z	second variable type
DUAL	dual variable type

3.5.2 Constructor & Destructor Documentation

3.5.2.1 admm_optimizer_t_impl()

```
template<typename T , typename P , typename X , typename Z , typename DUAL >
lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::admm_optimizer_t_impl (
             parameter_t && param = parameter_t{ (size_t) 1e4, 2, 1e-1} )  [inline], [explicit]
```

Default constructor.

Parameters

hyperparameter	of optimisation. Init hyperparameters with (size_t) 1e4, 2, 1e-1

3.5.3 Member Function Documentation

3.5.3.1 evaluate()

```
template<typename T , typename P , typename X , typename Z , typename DUAL >
T lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::evaluate (
             const P & prob,
             const X & x,
             const Z & z ) const  [inline]
```

evaluate augmented lagrangian

Parameters

prob	problem
x	variable
z	variable

Returns

loss/objectiv

3.5.3.2 optimize1()

```
template<typename T , typename P , typename X , typename Z , typename DUAL >
X lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::optimize1 (
             const P & prob,
             const X & x,
             const Z & z,
             const DUAL & d ) const  [inline]
```

optimize first subproblem. $\arg\min_x L_v(x, z^t, y^t)$ [1]

Parameters

prob	problem
x	variable
z	const variable
d	dual variable

Returns

optimal point x

3.5.3.3 optimize2()

```
template<typename T , typename P , typename X , typename Z , typename DUAL >
Z lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::optimize2 (
             const P & prob,
             const X & x,
             const Z & z,
             const DUAL & d ) const  [inline]
```

optimize second subproblem. $\arg\min_z L_v(x^{t+1}, z, y^t)$ [1]

Parameters

prob	problem
x	const variable
z	variable
d	dual variable

Returns

optimal point z

3.5.3.4 residual()

```
template<typename T , typename P , typename X , typename Z , typename DUAL >
DUAL lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::residual (
             const P & prob,
             const X & x,
             const Z & z ) const  [inline]
```

compute residual $Ax + Bz - c$ [1]

Parameters

prob	problem
x	variable
z	variable

Returns

residual

3.5.3.5 run()

```
template<typename T , typename P , typename X , typename Z , typename DUAL >
template<bool stats_enabled = false>
```

```
std::tuple<X,Z,T> lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::run (
            P & prob,
            X && x,
            Z && z,
            typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &
stats ) const  [inline]
```

The run method. Implementation of the optimisation algorithm. Adam-method.

Template Parameters

stats_enabled	enable/disable logging

Parameters

prob	problem
x	start variable / inital variable / start point (first variable)
z	start variable / inital variable / start point (second variable)
stats	statistics holder [5]

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

- lipnet/include/lipnet/optimizer/admm_optimizer.hpp

3.6 lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N > Struct Template Reference

The backpropagation_batch_t struct; implmentation of backtracking with batches.

```
#include <backpropagation.hpp>
```

Classes

- struct metainfo_t

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integer_sequence< size_t, N... > **DIMS**
- typedef network_t< T, ATYPE, N... >::layer_t **variable_t**
- typedef generate_batch_data_remove_first< T, BATCH, N... >::type **zdata_t**
- typedef generate_batch_data< T, BATCH, N... >::type **xdata_t**

Public Member Functions

- **backpropagation_batch_t** (LOSS< T > &&l, network_data_t< T, at< 0, N... >(), at< L::value, N... >() > &&data)
- void run (const variable_t &var, metainfo_t &info, variable_t &gradient, T &objective) const

  run function; compute backpropagation
- void compute (const variable_t &var, variable_t &gradient, T &objective) const

  run function; compute backpropagation
- void forward (const variable_t &layers, xdata_t &x, zdata_t &z) const

  forward function; compute forwardpropagation
- void backward (const variable_t &layers, variable_t &gradient, xdata_t &x, zdata_t &delta, zdata_t &z) const

  backward function; compute backpropagation

Public Attributes

- network_data_t< T, at< 0, N... >), at< L::value, N... >) > **training_data**
- LOSS< T > **loss**

3.6.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >

The backpropagation_batch_t struct; implmentation of backtracking with batches.

Template Parameters

T	numerical type
ATYPE	activation function type
LOSS	loss function type
BATCH	batch size
N	network topology

3.6.2 Member Function Documentation

3.6.2.1 backward()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
void lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >::backward (
            const variable_t & layers,
            variable_t & gradient,
            xdata_t & x,
            zdata_t & delta,
            zdata_t & z ) const  [inline]
```

backward function; compute backpropagation

Parameters

layers	weights and biases at each layer
gradient	gradient with respect to the weights and biases
x	
delta	gradients with respect to the layer inputs
z	

3.6.2.2 compute()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
void lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >::compute (
             const variable_t & var,
             variable_t & gradient,
             T & objective ) const  [inline]
```

run function; compute backpropagation

Parameters

var	current position
info	optimisation metainfo which are needed during the iterations
gradient	the computed gradients; the return value
objective	the loss at the current position

3.6.2.3 forward()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
void lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >::forward (
             const variable_t & layers,
             xdata_t & x,
             zdata_t & z ) const  [inline]
```

forward function; compute forwardpropagation

Parameters

layers	weights and biases at each layer
x	
z	

3.6.2.4 run()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO←
SS, size_t BATCH, size_t ...  N>
void lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >::run (
            const variable_t & var,
            metainfo_t & info,
            variable_t & gradient,
            T & objective ) const  [inline]
```

run function; compute backpropagation

Parameters

var	current position
info	optimisation metainfo which are needed during the iterations
gradient	the computed gradients; the return value
objective	the loss at the current position

See also

compute(const variable_t& var, variable_t& gradient, T& objective) const

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

- lipnet/include/lipnet/network/backpropagation.hpp

3.7 lipnet::barrierfunction_t< T, N > Struct Template Reference

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef blaze::IdentityMatrix< T > **eye**
- typedef cholesky_topology< T, N... >::type **cholesky_t**
- typedef inverse_topology< T, N... >::type **inverse_t**
- typedef network_topology< T, N... >::type **weights_t**
- typedef parameter_tparam< T, N... >::type **tparam_t**
- typedef liptrainweights_t< T, N... > **variable_t**
- typedef std::integral_constant< size_t, sizeof...(N) -2 > **LN**
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**

Public Member Functions

- **barrierfunction_t** (const T lipschitz=70.0)
- auto **compute** (const variable_t &var, variable_t &gradient, const T &gamma) const
- template<bool numeric_stability = true, typename kondition = std::ratio<1,100>, typename = typename std::enable_if<kondition::den != 0>::type>
  cholesky_t **chol** (const T lipschitz, const variable_t &var) const
- inverse_t **inv** (const cholesky_t &val) const

Public Attributes

- T **lipschitz**

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

- lipnet/include/lipnet/lipschitz/barrier.hpp

3.8 lipnet::barrierfunction_wot_t< T, N > Struct Template Reference

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef blaze::IdentityMatrix< T > **eye**
- typedef cholesky_topology< T, N... >::type **cholesky_t**
- typedef inverse_topology< T, N... >::type **inverse_t**
- typedef network_topology< T, N... >::type **variable_t**
- typedef parameter_tparam< T, N... >::type **tparam_t**
- typedef std::integral_constant< size_t, sizeof...(N) -2 > **LN**
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**

Public Member Functions

- **barrierfunction_wot_t** (tparam_t &&tmat, const T lipschitz=70.0)
- auto **compute** (const variable_t &var, variable_t &gradient, const T &gamma) const
- cholesky_t **chol** (const T lipschitz, const variable_t &weights, const tparam_t &tparam) const
- inverse_t **inv** (const cholesky_t &val) const

Public Attributes

- T **lipschitz**
- tparam_t **tparam**

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

- lipnet/include/lipnet/lipschitz/barrier_wot.hpp

3.9 lipnet::calculate_lipschitz_t< T, N > Struct Template Reference

compute trivial lipschitz constant

```
#include <trivial.hpp>
```

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef network_topology< T, N... >::type **variable_t**

Static Public Member Functions

- static T **trivial_lipschitz** (const variable_t &var)

3.9.1 Detailed Description

template<typename T, size_t ... N>
struct lipnet::calculate_lipschitz_t< T, N >

compute trivial lipschitz constant

Template Parameters

T	numerical value type
N	network topology [3]

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

- lipnet/include/lipnet/lipschitz/trivial.hpp

3.10 lipnet::cholesky_diagentry< T, N, NARGS > Struct Template Reference

Public Types

- typedef cholesky_diagentry_impl< T, NARGS... >::type **next**
- typedef join_tuples< std::tuple< T >, next >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.11 lipnet::cholesky_diagentry_impl< T, N, NS > Struct Template Reference

Public Types

- typedef cholesky_diagentry_impl< T, NS... >::type **next**
- typedef join_tuples< std::tuple< blaze::LowerMatrix< blaze::StaticMatrix< T, N, N > > >, next >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.12 lipnet::cholesky_diagentry_impl< T, N > Struct Template Reference

Public Types

- typedef std::tuple< blaze::LowerMatrix< blaze::StaticMatrix< T, N, N > > > **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.13 lipnet::cholesky_subentry< T, NI, NO, RE, NARGS > Struct Template Reference

Public Types

- typedef cholesky_subentry_impl< T, NI, NO, RE, NARGS... >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.14 lipnet::cholesky_subentry_impl< T, NI, NO, NS > Struct Template Reference

Public Types

- typedef cholesky_subentry_impl< T, NO, NS... >::type **next**
- typedef join_tuples< std::tuple< blaze::StaticMatrix< T, NO, NI > >, next >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.15 lipnet::cholesky_subentry_impl< T, NI, NO > Struct Template Reference

Public Types

- typedef std::tuple< blaze::StaticMatrix< T, NO, NI > > **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.16 lipnet::cholesky_topology< T, N > Struct Template Reference

Classes

- struct type

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.17 lipnet::cross_entropy_t< T > Struct Template Reference

The cross_entropy_t struct; implementation of the cross entropy objective function.

```
#include <loss.hpp>
```

Public Types

- template<typename TT , size_t O, size_t I>
  using **matrix_t** = blaze::StaticMatrix< TT, O, I, blaze::columnMajor >
- template<typename TT , size_t N>
  using **vector_t** = blaze::StaticVector< TT, N, blaze::columnVector >

Public Member Functions

- template<size_t N, size_t BATCH = 0, typename std::enable_if<!(BATCH<=0), int >::type = 0>
  T evaluate (const matrix_t< T, N, BATCH > &target, const matrix_t< T, N, BATCH > &data) const

  The evaluate function; compute loss.

3.17.1 Detailed Description

template<typename T>
struct lipnet::cross_entropy_t< T >

The cross_entropy_t struct; implementation of the cross entropy objective function.

$$\mathcal{L}(x, y) = \frac{\sum [x == y] \exp -x}{\sum \exp -x}$$

Template Parameters

T	numerical value type
TYPE	choose the activation type

3.17.2 Member Function Documentation

3.17.2.1 evaluate()

```
template<typename T >
template<size_t N, size_t BATCH = 0, typename std::enable_if<!(BATCH<=0), int >::type = 0>
T lipnet::cross_entropy_t< T >::evaluate (
            const matrix_t< T, N, BATCH > & target,
            const matrix_t< T, N, BATCH > & data ) const  [inline]
```

The evaluate function; compute loss.

Template Parameters

N	input dimension type
BATCH	batch size

Parameters

target	real value
estimated	value

Returns

loss

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

- lipnet/include/lipnet/network/loss.hpp

3.18 lipnet::data_container_t< T > Struct Template Reference

trining data holder; data_container_t

```
#include <container.hpp>
```

Classes

- struct data_t
- struct tuple_t
- struct view_t

Public Types

- using **matrix_t** = blaze::DynamicMatrix< T, blaze::rowMajor >

Public Attributes

- matrix_t **x**
- matrix_t **y**

3.18.1 Detailed Description

template<typename T>
struct lipnet::data_container_t< T >

trining data holder; data_container_t

Template Parameters

T	numerical value type

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

- lipnet/include/lipnet/loader/container.hpp

3.19 lipnet::network_t< T, ATYPE, N >::data_serialization_t< saveing > Struct Template Reference

serialization helper struct

```
#include <network.hpp>
```

Public Types

- using **value_t** = typename std::conditional< saveing, const layer_t, layer_t >::type
- using **seq_t** = std::make_integer_sequence< size_t, L::value >

Public Member Functions

- template< class Archive , size_t ... INTS>
  void **serialize_impl** (Archive &ar, const std::integer_sequence< size_t, INTS... > &)
- template< class Archive >
  void **serialize** (Archive &ar)

Public Attributes

- value_t & **layersdata**

3.19.1 Detailed Description

template< typename T, template< typename > typename ATYPE, size_t ... N>
template< bool saveing = true>
struct lipnet::network_t< T, ATYPE, N >::data_serialization_t< saveing >

serialization helper struct

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

- lipnet/include/lipnet/network/network.hpp

3.20 lipnet::data_container_t< T >::data_t< saveing > Struct Template Reference

Public Types

- using **value_t** = typename std::conditional< saveing, const matrix_t, matrix_t >::type

Public Member Functions

- template< class Archive >
  void **serialize** (Archive &ar)

Public Attributes

- value_t & **x**
- value_t & **y**

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

- lipnet/include/lipnet/loader/container.hpp

3.21 lipnet::decompos_diagentry< T, N, NARGS > Struct Template Reference

Public Types

- typedef decompos_diagentry_impl< T, NARGS... >::type **type**

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.22 lipnet::decompos_diagentry_impl< T, N, NS > Struct Template Reference

Public Types

- typedef decompos_diagentry_impl< T, NS... >::type **next**
- typedef join_tuples< std::tuple< blaze::StaticMatrix< T, N, N > >, next >::type **type**

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.23 lipnet::decompos_diagentry_impl< T, N > Struct Template Reference

Public Types

- typedef std::tuple< blaze::StaticMatrix< T, N, N > > **type**

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.24 lipnet::decompos_subentry< T, NI, NO, RE, NARGS > Struct Template Reference

Public Types

- typedef decompos_subentry_impl< T, NI, NO, RE, NARGS... >::type **type**

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.25 lipnet::decompos_subentry_impl< T, NI, NO, NS > Struct Template Reference

Public Types

- typedef decompos_subentry_impl< T, NO, NS... >::type **next**
- typedef join_tuples< std::tuple< blaze::StaticMatrix< T, NO, NI > >, next >::type **type**

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.26 lipnet::decompos_subentry_impl< T, NI, NO > Struct Template Reference

Public Types

- typedef std::tuple< blaze::StaticMatrix< T, NO, NI > > **type**

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.27 std::detail::detector< Default, AlwaysVoid, Op, Args > Struct Template Reference

Public Types

- using **value_t** = std::false_type
- using **type** = Default

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

- lipnet/include/lipnet/traits.hpp

3.28 std::detail::detector< Default, std::void_t< Op< Args... > >, Op, Args... > Struct Template Reference

Public Types

- using **value_t** = std::true_type
- using **type** = Op< Args... >

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

- lipnet/include/lipnet/traits.hpp

3.29 lipnet::equation_system_t< V1, V2 > Struct Template Reference

The equation_system_t struct. Just a interface for all possible types. Solve a system of equations.

```
#include <variable.hpp>
```

3.29.1 Detailed Description

template<typename V1, typename V2>
struct lipnet::equation_system_t< V1, V2 >

The equation_system_t struct. Just a interface for all possible types. Solve a system of equations.

$$Ax = b$$

.

Template Parameters

V1	tensor type of first argument
V2	tensot type of second argument

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

- lipnet/include/lipnet/variable.hpp

3.30 lipnet::equation_system_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >, blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > > Struct Template Reference

The equation_system_t struct for blaze::StaticMatrix.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static auto solve (const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > &A, const blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > &B)

  The solve method. Solve system of equations. $AX = A$.

3.30.1 Detailed Description

template<typename T, size_t N1, size_t N2, size_t N3, size_t N4>
struct lipnet::equation_system_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >, blaze::StaticMatrix< T, N3, N4, blaze::row↩Major > >

The equation_system_t struct for blaze::StaticMatrix.

Template Parameters

T	numerical value type
N1	row dimension of first argument
N2	column dimension of first argument
N3	row dimension of second argument
N4	column dimension of second argument

See also

lipnet::equation_system_t [6]

3.30.2 Member Function Documentation

3.30.2.1 solve()

```
template<typename T , size_t N1, size_t N2, size_t N3, size_t N4>
static auto lipnet::equation_system_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >,
blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > >::solve (
            const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > & A,
            const blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > & B )  [inline], [static]
```

The solve method. Solve system of equations. $AX = A$.

Parameters

A	matrix A (first argument)
B	matrix A (second argument)

Returns

matrix X

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

- lipnet/include/lipnet/tensor.hpp

3.31 lipnet::equation_system_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >, blaze::StaticVector< T, N3, blaze::columnVector > > Struct Template Reference

The equation_system_t struct for blaze::StaticMatrix and blaze::StaticVector.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static auto solve (const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > &A, const blaze::StaticVector< T, N3, blaze::columnVector > &B)

  The solve method. Solve system of equations. $Ax = b$.

3.31.1 Detailed Description

template<typename T, size_t N1, size_t N2, size_t N3>
struct lipnet::equation_system_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >, blaze::StaticVector< T, N3, blaze::columnVector > >

The equation_system_t struct for blaze::StaticMatrix and blaze::StaticVector.

Template Parameters

T	numerical value type
N1	row dimension of first argument
N2	column dimension of first argument
N3	dimension of second argument

See also

lipnet::equation_system_t [6]

3.31.2 Member Function Documentation

3.31.2.1 solve()

```
template<typename T , size_t N1, size_t N2, size_t N3>
static auto lipnet::equation_system_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >,
blaze::StaticVector< T, N3, blaze::columnVector > >::solve (
            const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > & A,
            const blaze::StaticVector< T, N3, blaze::columnVector > & B )  [inline], [static]
```

The solve method. Solve system of equations. $Ax = b$.

Parameters

A	matrix A (first argument)
B	vector b (second argument)

Returns

vector x

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

- lipnet/include/lipnet/tensor.hpp

3.32 lipnet::fast_gradient_descent_t_impl< T, P, VAR, GRAD > Struct Template Reference

gradient descent algorithm.

```
#include <fast_gradient_descent.hpp>
```

Classes

- struct parameter_t
- struct statistics_t

  problem specific implementation of statistics_t

Public Member Functions

- void **unpack** (std::tuple< GRAD, T > &&t, GRAD &dx, T &fx) const
- fast_gradient_descent_t_impl (parameter_t &&param=parameter_t{0.001, 1e-8})

  Default constructor.
- template<bool stats_enabled = false>
  std::tuple< VAR, T > run (P &prob, VAR &&x, typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &stats) const

  The run method. Implementation of the optimisation algorithm.

Public Attributes

- parameter_t param

  variables to optimize

3.32.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD>
struct lipnet::fast_gradient_descent_t_impl< T, P, VAR, GRAD >

gradient descent algorithm.

Template Parameters

T	numerical value type
P	problem type
VAR	variable type
GRAD	gradient type

3.32.2 Constructor & Destructor Documentation

3.32.2.1 fast_gradient_descent_t_impl()

```
template<typename T , typename P , typename VAR , typename GRAD >
lipnet::fast_gradient_descent_t_impl< T, P, VAR, GRAD >::fast_gradient_descent_t_impl (
            parameter_t && param = parameter_t{0.001, 1e-8} )  [inline], [explicit]
```

Default constructor.

Parameters

hyperparameter	of optimisation. Init hyperparameters with 0.001, 1e-8

3.32.3 Member Function Documentation

3.32.3.1 run()

```
template<typename T , typename P , typename VAR , typename GRAD >
template<bool stats_enabled = false>
std::tuple<VAR,T> lipnet::fast_gradient_descent_t_impl< T, P, VAR, GRAD >::run (
            P & prob,
            VAR && x,
            typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &
stats ) const  [inline]
```

The run method. Implementation of the optimisation algorithm.

Template Parameters

stats_enabled	enable/disable logging

Parameters

prob	problem
x	start variable / inital variable / start point
stats	statistics holder

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

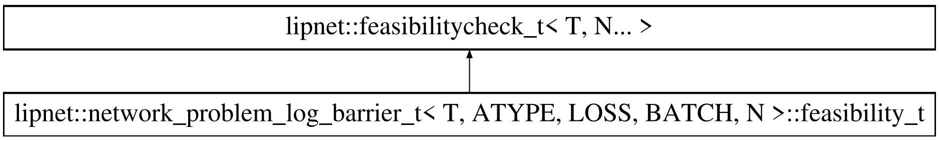
- lipnet/include/lipnet/optimizer/fast_gradient_descent.hpp

3.33 lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >::feasibility_t Struct Reference

The feasibility_t struct. Implementation of feasibility check for this problem.

```
#include <nn_problem_liptrain_barrier.hpp>
```

Inheritance diagram for lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >::feasibility_t:


lipnet::feasibilitycheck_t< T, N... >
lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >::feasibility_t

Public Member Functions

- void **init** (const T r, const variable_t &p)
- void **run** (const variable_t &dir)

Public Attributes

- variable_t **pos**
- T **step**
- T **rho**

Additional Inherited Members

3.33.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >::feasibility_t

The feasibility_t struct. Implementation of feasibility check for this problem.

See also

lipnet::feasibilitycheck_t

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

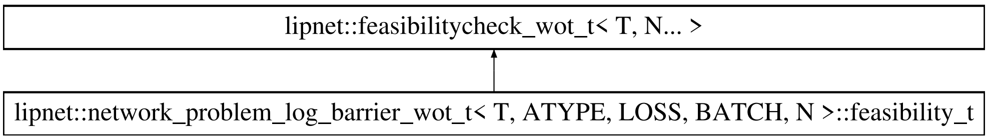
- lipnet/include/lipnet/problem/nn_problem_liptrain_barrier.hpp

3.34 lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::feasibility_t Struct Reference

The feasibility_t struct. Implementation of feasibility check for this problem.

```
#include <nn_problem_liptrain_barrier_wot.hpp>
```

Inheritance diagram for lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::feasibility_t:


lipnet::feasibilitycheck_wot_t< T, N... >
lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::feasibility_t

Public Member Functions

- void **init** (typename self_barrier_t::cholesky_t &&l, const typename self_barrier_t::tparam_t &t)
- void **run** (const variable_t &dir)

Public Attributes

- self_barrier_t::cholesky_t **L**
- T **step**
- std::optional< std::reference_wrapper< const typename self_barrier_t::tparam_t > > **Tparam**

Additional Inherited Members

3.34.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::feasibility_t

The feasibility_t struct. Implementation of feasibility check for this problem.

See also

lipnet::feasibilitycheck_wot_t

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

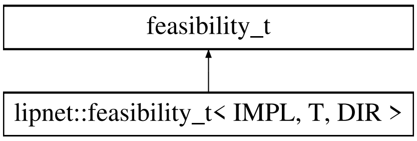
- lipnet/include/lipnet/problem/nn_problem_liptrain_barrier_wot.hpp

3.35 lipnet::feasibility_t< IMPL, T, DIR > Struct Template Reference

The feasibility_t struct. base feasibility struct (basically a placerholder class)

```
#include <problem.hpp>
```

Inheritance diagram for lipnet::feasibility_t< IMPL, T, DIR >:



Public Member Functions

- T operator() () const

  compute max stepsize for problem specific constraint.
- void operator<< (const DIR &dir)

  set direction for evaluation.

3.35.1 Detailed Description

template<typename IMPL, typename T, typename DIR>
struct lipnet::feasibility_t< IMPL, T, DIR >

The feasibility_t struct. base feasibility struct (basically a placerholder class)

Template Parameters

IMPL	problem type
T	numerical value type
DIR	variable type

3.35.2 Member Function Documentation

3.35.2.1 operator()()

```
template<typename IMPL , typename T , typename DIR >
T lipnet::feasibility_t< IMPL, T, DIR >::operator() ( ) const  [inline]
```

compute max stepsize for problem specific constraint.

$$\hat{\alpha} = \max_{\alpha} \alpha \quad \text{s.t.} \quad [x_k - \alpha\Delta x] \text{ is feasible}$$

3.35.2.2 operator<<()

```
template<typename IMPL , typename T , typename DIR >
void lipnet::feasibility_t< IMPL, T, DIR >::operator<< (
            const DIR & dir )  [inline]
```

set direction for evaluation.

Parameters

dir	direction $\Delta x \quad x_{k+1} = x_k - \alpha\Delta x$

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

- lipnet/include/lipnet/problem.hpp

3.36 lipnet::feasibilitycheck_t< T, N > Struct Template Reference

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef blaze::IdentityMatrix< T > **eye**
- typedef cholesky_topology< T, N... >::type **cholesky_t**
- typedef inverse_topology< T, N... >::type **inverse_t**
- typedef network_topology< T, N... >::type **weight_t**
- typedef parameter_tparam< T, N... >::type **tparam_t**
- typedef liptrainweights_t< T, N... > **variable_t**
- typedef std::integral_constant< size_t,(N+...) > **NN**
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**

Public Member Functions

- template<typename kondition = std::ratio<2,1>, typename = typename std::enable_if<kondition::den != 0>::type>
  T **compute** (const variable_t &pos, const variable_t &gradient, const T rho) const

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

- lipnet/include/lipnet/lipschitz/feasibility.hpp

3.37 lipnet::feasibilitycheck_wot_t< T, N > Struct Template Reference

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef blaze::IdentityMatrix< T > **eye**
- typedef cholesky_topology< T, N... >::type **cholesky_t**
- typedef inverse_topology< T, N... >::type **inverse_t**
- typedef network_topology< T, N... >::type **variable_t**
- typedef parameter_tparam< T, N... >::type **tparam_t**
- typedef std::integral_constant< size_t,(N+...) > **NN**
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**

Public Member Functions

- T **compute** (const tparam_t &tparam, const cholesky_t &var, const variable_t &gradient) const

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

- lipnet/include/lipnet/lipschitz/feasibility.hpp

3.38 lipnet::function_t< V > Struct Template Reference

The function_t struct. Just a interface for all possible types. Apply function to tensor elementwise.

```
#include <variable.hpp>
```

3.38.1 Detailed Description

template<typename V>
struct lipnet::function_t< V >

The function_t struct. Just a interface for all possible types. Apply function to tensor elementwise.

Template Parameters

V	tensor type of argument

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

- lipnet/include/lipnet/variable.hpp

3.39 lipnet::function_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > > Struct Template Reference

The function_t struct for blaze::StaticMatrix.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static auto trans (const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > &m)

  transpose matrix $M^\top$

3.39.1 Detailed Description

template<typename T, size_t N1, size_t N2>
struct lipnet::function_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > >

The function_t struct for blaze::StaticMatrix.

Template Parameters

T	numerical value type
N1	row dimension of argument
N2	column dimension of argument

See also

lipnet::function_t [6]

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

- lipnet/include/lipnet/tensor.hpp

3.40 lipnet::function_t< blaze::StaticVector< T, N, blaze::columnVector > > Struct Template Reference

The function_t struct for blaze::StaticVector.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static auto trans (const blaze::StaticVector< T, N, blaze::columnVector > &vec)

  transpose vector $v^\top$
- static auto square (const blaze::StaticVector< T, N, blaze::columnVector > &vec)

  square vector elementwise
- static auto sqrt (const blaze::StaticVector< T, N, blaze::columnVector > &vec)

  take square root of vector elementwise

3.40.1 Detailed Description

template<typename T, size_t N>
struct lipnet::function_t< blaze::StaticVector< T, N, blaze::columnVector > >

The function_t struct for blaze::StaticVector.

Template Parameters

T	numerical value type
N	dimension of argument

See also

lipnet::function_t [6]

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

- lipnet/include/lipnet/tensor.hpp

3.41 lipnet::function_t< layer_t< T, I, O > > Struct Template Reference

Static Public Member Functions

- static auto **square** (const layer_t< T, I, O > &m)
- static auto **sqrt** (const layer_t< T, I, O > &m)

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

- lipnet/include/lipnet/network/layer.hpp

3.42 lipnet::function_t< liptrainweights_t< T, N... > > Struct Template Reference

Static Public Member Functions

- static auto **square** (const liptrainweights_t< T, N... > &m)
- static auto **sqrt** (const liptrainweights_t< T, N... > &m)

Public Attributes

- decltype(liptrainweights_t< T, N... >::W) typedef **arg1_t**
- decltype(liptrainweights_t< T, N... >::t) typedef **arg2_t**

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

- lipnet/include/lipnet/lipschitz/barrier.hpp

3.43 lipnet::function_t< std::tuple< ARGS... > > Struct Template Reference

Static Public Member Functions

- template<size_t ... INTS>
  static auto **square_impl** (const std::tuple< ARGS... > &m, std::integer_sequence< size_t, INTS... >)
- static auto **square** (const std::tuple< ARGS... > &m)
- template<size_t ... INTS>
  static auto **sqrt_impl** (const std::tuple< ARGS... > &m, std::integer_sequence< size_t, INTS... >)
- static auto **sqrt** (const std::tuple< ARGS... > &m)

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

- lipnet/include/lipnet/tuple.hpp

3.44 lipnet::generate_batch_data< T, B, N, NS > Struct Template Reference

helper struct for data

```
#include <topology.hpp>
```

Public Types

- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef generate_batch_data< T, B, NS... >::type **next**
- typedef join_tuples< std::tuple< matrix_t< N, B > >, next >::type **type**

3.44.1 Detailed Description

template<typename T, size_t B, size_t N, size_t ... NS>
struct lipnet::generate_batch_data< T, B, N, NS >

helper struct for data

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

- lipnet/include/lipnet/network/topology.hpp

3.45 lipnet::generate_batch_data< T, B, N > Struct Template Reference

helper struct for data

```
#include <topology.hpp>
```

Public Types

- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::tuple< matrix_t< N, B > > **type**

3.45.1 Detailed Description

template<typename T, size_t B, size_t N>
struct lipnet::generate_batch_data< T, B, N >

helper struct for data

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

- lipnet/include/lipnet/network/topology.hpp

3.46 lipnet::generate_batch_data_remove_first< T, B, N, NS > Struct Template Reference

helper struct for data

```
#include <topology.hpp>
```

Public Types

- typedef generate_batch_data< T, B, NS... >::type **type**

3.46.1 Detailed Description

template<typename T, size_t B, size_t N, size_t ... NS>
struct lipnet::generate_batch_data_remove_first< T, B, N, NS >

helper struct for data

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

- lipnet/include/lipnet/network/topology.hpp

3.47 lipnet::generate_data< T, N, NS > Struct Template Reference

helper struct for data

```
#include <topology.hpp>
```

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- typedef generate_data< T, NS... >::type **next**
- typedef join_tuples< std::tuple< vector_t< N > >, next >::type **type**

3.47.1 Detailed Description

template<typename T, size_t N, size_t ... NS>
struct lipnet::generate_data< T, N, NS >

helper struct for data

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

- lipnet/include/lipnet/network/topology.hpp

3.48 lipnet::generate_data< T, N > Struct Template Reference

helper struct for data

```
#include <topology.hpp>
```

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- typedef std::tuple< vector_t< N > > **type**

3.48.1 Detailed Description

template<typename T, size_t N>
struct lipnet::generate_data< T, N >

helper struct for data

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

- lipnet/include/lipnet/network/topology.hpp

3.49 lipnet::generate_data_remove_first< T, N, NS > Struct Template Reference

helper struct for data

```
#include <topology.hpp>
```

Public Types

- typedef generate_data< T, NS..., 0 >::type **type**

3.49.1 Detailed Description

template<typename T, size_t N, size_t ... NS>
struct lipnet::generate_data_remove_first< T, N, NS >

helper struct for data

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

- lipnet/include/lipnet/network/topology.hpp

3.50 lipnet::generator_t< V > Struct Template Reference

The generator_t struct. Just a interface for all possible types. Instanciate tensor of type V.

```
#include <variable.hpp>
```

3.50.1 Detailed Description

template<typename V>
struct lipnet::generator_t< V >

The generator_t struct. Just a interface for all possible types. Instanciate tensor of type V.

Template Parameters

V	tensor type to create

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

- lipnet/include/lipnet/variable.hpp

3.51 lipnet::generator_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > > Struct Template Reference

The generator_t struct for blaze::StaticMatrix.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static auto make (const T &val)

    uniform distribution constructor $\sim \mathcal{U}(-val, val)$
- static auto unifrom (const T &val)

    uniform distribution constructor $\sim \mathcal{U}(-val, val)$
- static auto identity ()

    identity constructor I

3.51.1 Detailed Description

template<typename T, size_t N1, size_t N2>
struct lipnet::generator_t< blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > >

The generator_t struct for blaze::StaticMatrix.

Template Parameters

T	numerical value type
N1	row dimension of return matrix
N2	column dimension of return matrix

See also

lipnet::generator_t [6]

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

- lipnet/include/lipnet/tensor.hpp

3.52 lipnet::generator_t< blaze::StaticVector< T, N, blaze::columnVector > > Struct Template Reference

The generator_t struct for blaze::StaticVector.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static auto make (const T &val)

    uniform distribution constructor $\sim \mathcal{U}(-val, val)$
- static auto unifrom (const T &val)

    uniform distribution constructor $\sim \mathcal{U}(-val, val)$

3.52.1 Detailed Description

template<typename T, size_t N>
struct lipnet::generator_t< blaze::StaticVector< T, N, blaze::columnVector > >

The generator_t struct for blaze::StaticVector.

Template Parameters

T	numerical value type
N	dimension of return vector

See also

lipnet::generator_t [6]

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

- lipnet/include/lipnet/tensor.hpp

3.53 lipnet::generator_t< layer_t< T, I, O > > Struct Template Reference

Static Public Member Functions

- static layer_t< T, I, O > **make** (T val)

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

- lipnet/include/lipnet/network/layer.hpp

3.54 lipnet::generator_t< liptrainweights_t< T, N... > > Struct Template Reference

Static Public Member Functions

- static liptrainweights_t< T, N... > **make** (T val, T uni)

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

- lipnet/include/lipnet/lipschitz/barrier.hpp

3.55 lipnet::generator_t< parameter_decompo_t< T, N... > > Struct Template Reference

Static Public Member Functions

- static auto **make** (const T &init)

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.56 lipnet::generator_t< std::tuple< ARGS... > > Struct Template Reference

generator_t implementation for std::tuple

```
#include <topology.hpp>
```

Static Public Member Functions

- template<typename T >
  static std::tuple< ARGS... > **make** (T val)

3.56.1 Detailed Description

template<typename ... ARGS>
struct lipnet::generator_t< std::tuple< ARGS... > >

generator_t implementation for std::tuple

See also

lipnet::generator_t

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

- lipnet/include/lipnet/network/topology.hpp

3.57 lipnet::gradient_descent_projected_t_impl< T, P, VAR, GRAD > Struct Template Reference

projected gradient descent algorithm.

```
#include <gradient_descent_projected.hpp>
```

Classes

- struct parameter_t
- struct statistics_t

  problem specific implementation of statistics_t

Public Member Functions

- void **unpack** (std::tuple< GRAD, T > &&t, GRAD &dx, T &fx) const
- auto project (const P &prob, VAR &&var) const

  The project method. Call projection method of problem.
- gradient_descent_projected_t_impl (parameter_t &&param=parameter_t{(size_t) 5e5, 1e-6, 0.001, 1e-8})

  Default constructor.
- template<bool stats_enabled = false>
  std::tuple< VAR, T > run (P &prob, VAR &&x, typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &stats) const

  The run method. Implementation of the optimisation algorithm.

Public Attributes

- parameter_t param

  variables to optimize

3.57.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD>
struct lipnet::gradient_descent_projected_t_impl< T, P, VAR, GRAD >

projected gradient descent algorithm.

Template Parameters

T	numerical value type
P	problem type
VAR	variable type
GRAD	gradient type

3.57.2 Constructor & Destructor Documentation

3.57.2.1 gradient_descent_projected_t_impl()

```
template<typename T , typename P , typename VAR , typename GRAD >
lipnet::gradient_descent_projected_t_impl< T, P, VAR, GRAD >::gradient_descent_projected_t_impl
(
            parameter_t && param = parameter_t{ (size_t) 5e5, 1e-6, 0.001, 1e-8} )  [inline],
[explicit]
```

Default constructor.

Parameters

hyperparameter	of optimisation. Init hyperparameters with (size_t) 5e3, 1e-6, 0.001, 1e-8

3.57.3 Member Function Documentation

3.57.3.1 project()

```
template<typename T , typename P , typename VAR , typename GRAD >
auto lipnet::gradient_descent_projected_t_impl< T, P, VAR, GRAD >::project (
            const P & prob,
            VAR && var ) const  [inline]
```

The project method. Call projection method of problem.

Parameters

prob	problem
var	current variables; will be projected to feasible set

3.57.3.2 run()

```
template<typename T , typename P , typename VAR , typename GRAD >
template<bool stats_enabled = false>
std::tuple<VAR,T> lipnet::gradient_descent_projected_t_impl< T, P, VAR, GRAD >::run (
            P & prob,
            VAR && x,
            typename std::conditional< stats_enabled, statistics_t, std::void_type >::type &
stats ) const  [inline]
```

The run method. Implementation of the optimisation algorithm.

Template Parameters

stats_enabled	enable/disable logging

Parameters

prob	problem
x	start variable / inital variable / start point
stats	statistics holder

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

- lipnet/include/lipnet/optimizer/gradient_descent_projected.hpp

3.58 lipnet::helper_function_t< V > Struct Template Reference

Static Public Attributes

- constexpr static bool **v1**
- constexpr static bool **v2**
- constexpr static bool **value** = v1 && v2

3.58.1 Member Data Documentation

3.58.1.1 v1

```
template<typename V >
constexpr static bool lipnet::helper_function_t< V >::v1  [inline], [static], [constexpr]
```

Initial value:

```
= std::is_invocable_r<V,
                decltype(&function_t<V>::square), const V&>::value
```

3.58.1.2 v2

```
template<typename V >
constexpr static bool lipnet::helper_function_t< V >::v2  [inline], [static], [constexpr]
```

Initial value:

```
= std::is_invocable_r<V,
                decltype(&function_t<V>::sqrt), const V&>::value
```

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

- lipnet/include/lipnet/variable.hpp

3.59 lipnet::helper_inner_t< T, V1, V2 > Struct Template Reference

Static Public Attributes

- constexpr static bool **value**

3.59.1 Member Data Documentation

3.59.1.1 value

```
template<typename T , typename V1 , typename V2 >
constexpr static bool lipnet::helper_inner_t< T, V1, V2 >::value  [inline], [static], [constexpr]
```

Initial value:

```
= std::is_invocable_r<T,
                decltype(&prod_t<T,V1,V2>::inner), const V1&, const V2&>::value
```

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

- lipnet/include/lipnet/variable.hpp

3.60 lipnet::helper_norm_t< T, V > Struct Template Reference

Static Public Attributes

- constexpr static bool **value**

3.60.1 Member Data Documentation

3.60.1.1 value

```
template<typename T , typename V >
constexpr static bool lipnet::helper_norm_t< T, V >::value  [inline], [static], [constexpr]
```

Initial value:

```
= std::is_invocable_r<T,
                decltype(&norm_t<T,V>::norm), const V&>::value
```

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

- lipnet/include/lipnet/variable.hpp

3.61 image_t Struct Reference

Public Types

- typedef blaze::StaticVector< double, 3, blaze::columnVector > **pixel_t**

Public Member Functions

- **image_t** (size_t w, size_t h)
- pixel_t & **operator()** (const size_t &x, const size_t &y)

Public Attributes

- size_t **width**
- size_t **height**
- std::vector< pixel_t > **data**

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

- lipnet/src/plotting_objectivsurface.cpp

3.62 lipnet::inverse_diagentry< T, N, NARGS > Struct Template Reference

Public Types

- typedef inverse_diagentry_impl< T, N, NARGS... >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.63 lipnet::inverse_diagentry_impl< T, N, NS > Struct Template Reference

Public Types

- typedef inverse_diagentry_impl< T, NS... >::type **next**
- typedef join_tuples< std::tuple< blaze::SymmetricMatrix< blaze::StaticMatrix< T, N, N > > >, next >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.64 lipnet::inverse_diagentry_impl< T, N > Struct Template Reference

Public Types

- typedef std::tuple< blaze::SymmetricMatrix< blaze::StaticMatrix< T, N, N > > > **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.65 lipnet::inverse_subentry< T, NI, NO, RE, NARGS > Struct Template Reference

Public Types

- typedef inverse_subentry_impl< T, NI, NO, RE, NARGS... >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.66 lipnet::inverse_subentry_impl< T, NI, NO, NS > Struct Template Reference

Public Types

- typedef inverse_subentry_impl< T, NO, NS... >::type **next**
- typedef join_tuples< std::tuple< blaze::StaticMatrix< T, NO, NI > >, next >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.67 lipnet::inverse_subentry_impl< T, NI, NO > Struct Template Reference

Public Types

- typedef std::tuple< blaze::StaticMatrix< T, NO, NI > > **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.68 lipnet::inverse_topology< T, N > Struct Template Reference

Classes

- struct type

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.69 lipnet::join_tuples< typename, typename > Struct Template Reference

Helper struct to join two tuples. (std::tuple)

```
#include <tuple.hpp>
```

3.69.1 Detailed Description

template<typename, typename>
struct lipnet::join_tuples< typename, typename >

Helper struct to join two tuples. (std::tuple)

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

- lipnet/include/lipnet/tuple.hpp

3.70 lipnet::join_tuples< std::tuple< NEW... >, std::tuple< NEXT... > > Struct Template Reference

Implementation of join_tuples struct to join two tuples. (std::tuple)

```
#include <tuple.hpp>
```

Public Types

- typedef std::tuple< NEW..., NEXT... > **type**

3.70.1 Detailed Description

template<typename... NEW, typename... NEXT>
struct lipnet::join_tuples< std::tuple< NEW... >, std::tuple< NEXT... > >

Implementation of join_tuples struct to join two tuples. (std::tuple)

See also

lipnet::join_tuples

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

- lipnet/include/lipnet/tuple.hpp

3.71 lipnet::layer_t< T, I, O > Struct Template Reference

The layer_t struct; the layer implementation of each layer; contains the weight and the biases.

```
#include <layer.hpp>
```

Public Types

- typedef std::array< T, I ∗O > **weight_array_t**
- typedef std::array< T, O > **bias_array_t**
- typedef blaze::StaticMatrix< T, O, I, blaze::columnMajor > **MT**
- typedef blaze::StaticVector< T, O, blaze::columnVector > **VT**

Public Member Functions

- **layer_t** (MT &&w, VT &&b)
- layer_t (const T &var)

  The layer_t constructor; initilize weight and bias with random values.
- template<class Archive >

  void serialize (Archive &ar)

  serialize layer_t

Public Attributes

- MT **weight**
- VT **bias**

3.71.1 Detailed Description

template<typename T, size_t I, size_t O>
struct lipnet::layer_t< T, I, O >

The layer_t struct; the layer implementation of each layer; contains the weight and the biases.

Template Parameters

T	numerical value type
I	input dimension
O	output dimension

3.71.2 Constructor & Destructor Documentation

3.71.2.1 layer_t()

```
template<typename T , size_t I, size_t O>
lipnet::layer_t< T, I, O >::layer_t (
             const T & var )  [inline], [explicit]
```

The layer_t constructor; initilize weight and bias with random values.

Parameters

var	some kind of variance

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

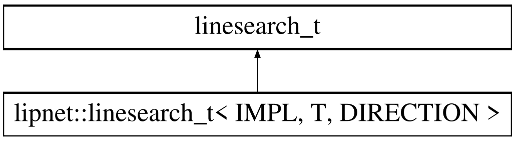
- lipnet/include/lipnet/network/layer.hpp

3.72 lipnet::linesearch_t< IMPL, T, DIRECTION > Struct Template Reference

The linesearch_t struct. base linesearch struct (basically a placerholder class)

```
#include <problem.hpp>
```

Inheritance diagram for lipnet::linesearch_t< IMPL, T, DIRECTION >:



Public Member Functions

- T operator() (const T val) const

  evaluate function with stepsize val.
- void operator<< (const DIRECTION &dir)

  set direction for evaluation.

3.72.1 Detailed Description

template<typename IMPL, typename T, typename DIRECTION>
struct lipnet::linesearch_t< IMPL, T, DIRECTION >

The linesearch_t struct. base linesearch struct (basically a placerholder class)

Template Parameters

IMPL	problem type
T	numerical value type
DIRECTION	variable type

3.72.2 Member Function Documentation

3.72.2.1 operator()()

```
template<typename IMPL , typename T , typename DIRECTION >
T lipnet::linesearch_t< IMPL, T, DIRECTION >::operator() (
            const T val ) const  [inline]
```

evaluate function with stepsize val.

Parameters

val	stepsize $\alpha \quad x_{k+1} = x_k - \alpha \Delta x$

3.72.2.2 operator<<()

```
template<typename IMPL , typename T , typename DIRECTION >
void lipnet::linesearch_t< IMPL, T, DIRECTION >::operator<< (
            const DIRECTION & dir )  [inline]
```

set direction for evaluation.

Parameters

dir	direction $\Delta x \quad x_{k+1} = x_k - \alpha \Delta x$

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

- lipnet/include/lipnet/problem.hpp

3.73 lipnet::lipcalc_parameter_t< T, N > Struct Template Reference

Public Attributes

- T **rho**
- blaze::StaticVector< T, sum< sizeof...(N) -1, N... >) - at< 0, N... >) > **tmat**

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

- lipnet/include/lipnet/lipschitz/parameter.hpp

3.74 lipnet::liptrainweights_t< T, N > Struct Template Reference

Public Member Functions

- template<class Archive >
  void **save** (Archive &ar) const
- template<class Archive >
  void **load** (Archive &ar)

Public Attributes

- network_topology< T, N... >::type **W**
- parameter_tparam< T, N... >::type **t**

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

- lipnet/include/lipnet/lipschitz/barrier.hpp

3.75 lipnet::loader_t< T > Struct Template Reference

struct for loading matrix from csv file;

```
#include <loader.hpp>
```

Public Types

- typedef blaze::DynamicMatrix< T, blaze::rowMajor > **dmatrix_t**

Static Public Member Functions

- static std::optional< dmatrix_t > load (const std::string &path)

  load matrix from csv file;

3.75.1 Detailed Description

template<typename T>
struct lipnet::loader_t< T >

struct for loading matrix from csv file;

Template Parameters

T	numerical value type [7]

3.75.2 Member Function Documentation

3.75.2.1 load()

```
template<typename T >
static std::optional<dmatrix_t> lipnet::loader_t< T >::load (
             const std::string & path )  [inline], [static]
```

load matrix from csv file;

Parameters

path	path to file on filesystem

Returns

matrix

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

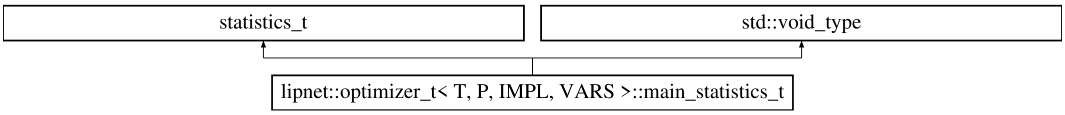
- lipnet/include/lipnet/loader/loader.hpp

3.76 lipnet::optimizer_t< T, P, IMPL, VARS >::main_statistics_t Struct Reference

The main_statistics_t struct.

```
#include <optimizer.hpp>
```

Inheritance diagram for lipnet::optimizer_t< T, P, IMPL, VARS >::main_statistics_t:



Public Member Functions

- template<class Archive >
  void **serialize** (Archive &archive)

Public Attributes

- std::chrono::milliseconds **duration**

3.76.1 Detailed Description

template<typename T, typename P, typename IMPL, typename ... VARS>
struct lipnet::optimizer_t< T, P, IMPL, VARS >::main_statistics_t

The main_statistics_t struct.

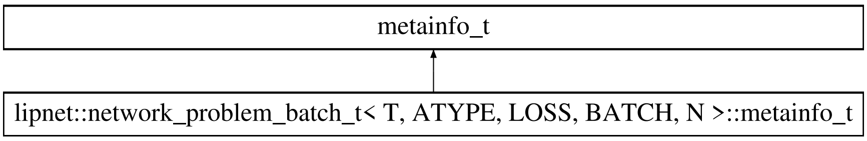
Just contains a variable to, which stores th computation time to solve the problem. The variable stores it's value in milliseconds.

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

- lipnet/include/lipnet/optimizer.hpp

3.77 lipnet::network_problem_batch_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t Struct Reference

Inheritance diagram for lipnet::network_problem_batch_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t:

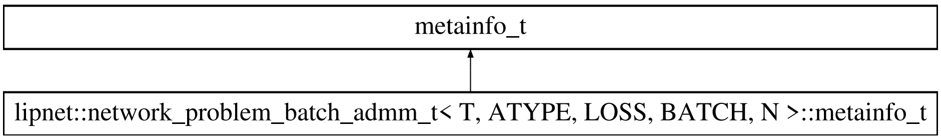

metainfo_t
lipnet::network_problem_batch_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t

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

- lipnet/include/lipnet/problem/nn_problem_batch.hpp

3.78 lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t Struct Reference

Inheritance diagram for lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t:

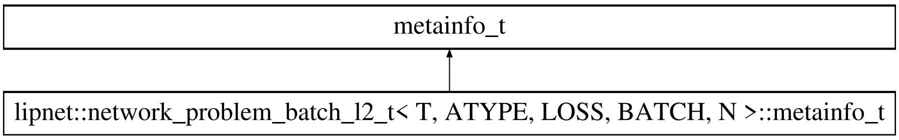

metainfo_t
lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t

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

- lipnet/include/lipnet/problem/nn_problem_batch_admm.hpp

3.79 lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t Struct Reference

Inheritance diagram for lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t:

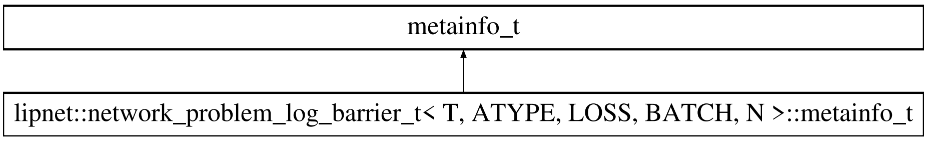

metainfo_t
lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t

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

- lipnet/include/lipnet/problem/nn_problem_batch_l2.hpp

3.80 lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t Struct Reference

Inheritance diagram for lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t:

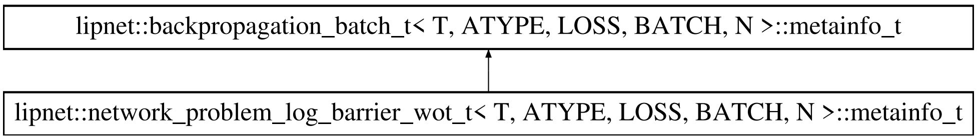

metainfo_t
lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t

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

- lipnet/include/lipnet/problem/nn_problem_liptrain_barrier.hpp

3.81 lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t Struct Reference

Inheritance diagram for lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t:


lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t
lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t

Public Attributes

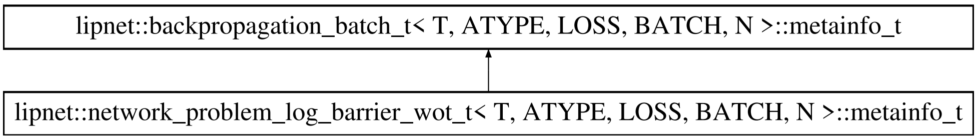
- size_t **iter**

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

- lipnet/include/lipnet/network/backpropagation.hpp

3.82 lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t Struct Reference

Inheritance diagram for lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t:


lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t
lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t

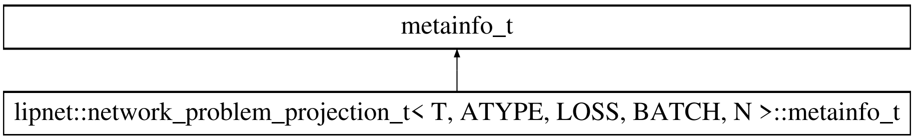
Additional Inherited Members

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

- lipnet/include/lipnet/problem/nn_problem_liptrain_barrier_wot.hpp

3.83 lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t Struct Reference

Inheritance diagram for lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t:


metainfo_t
lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >::metainfo_t

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

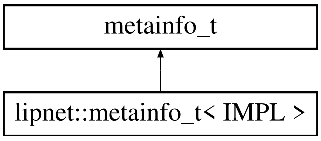
- lipnet/include/lipnet/problem/nn_problem_liptrain_projection.hpp

3.84 lipnet::metainfo_t< IMPL > Struct Template Reference

The metainfo_t struct. Data holder type for data needed during the iterations.

```
#include <problem.hpp>
```

Inheritance diagram for lipnet::metainfo_t< IMPL >:



3.84.1 Detailed Description

template<typename IMPL>
struct lipnet::metainfo_t< IMPL >

The metainfo_t struct. Data holder type for data needed during the iterations.

Template Parameters

IMPL	problem type

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

- lipnet/include/lipnet/problem.hpp

3.85 lipnet::mosek_projection_wot_t< T, N > Struct Template Reference

The mosek_projection_wot_t struct. Compute the projection of the reference weights. It is conic program and will be solved with mosek (interior point method)

```
#include <mosek_projection_wot.hpp>
```

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integer_sequence< size_t, N... > **DIMS**
- typedef std::integral_constant< size_t, sum_from_to< 1, L::value, N... >) > **n**
- typedef network_t< T, identity_activation_t, N... >::layer_t **variable_t**

Static Public Member Functions

- template<size_t R, size_t C>
  static fusion::Matrix::t map (const matrix_t< R, C > &mat)

  map input argument to mosek parameter type
- static variable_t projection (const T lipschitz, variable_t &&ref, const T &tinitval)

  Compute projetion of weights into feasible set.

3.85.1 Detailed Description

template<typename T, size_t ... N>
struct lipnet::mosek_projection_wot_t< T, N >

The mosek_projection_wot_t struct. Compute the projection of the reference weights. It is conic program and will be solved with mosek (interior point method)

```
@f[ \arg \min_{W,\eta} \quad \eta \quad \mathrm{s.t} \chi(\Psi^2,W) \succeq 0 \quad
        \left[ \eta \;\;\; \mathrm{fl}(W - W_\mathrm{ref} )  \right] \succeq_\mathcal{Q} 0 @f]
```

Template Parameters

T	numerical value type
N	network topology

3.85.2 Member Function Documentation

3.85.2.1 projection()

```
template<typename T , size_t ...  N>
static variable_t lipnet::mosek_projection_wot_t< T, N >::projection (
             const T lipschitz,
             variable_t && ref,
             const T & tinitval )  [inline], [static]
```

Compute projetion of weights into feasible set.

Parameters

lipschitz	lipschitz integral_constant
ref	reference weights; computed during gradient descent step
tinitval	hyperparameter T of chi matrix

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

- lipnet/include/lipnet/extern/mosek_projection_wot.hpp

3.86 lipnet::network_data_t< T, IN, OUT > Struct Template Reference

The network_data_t struct; training dataset.

```
#include <backpropagation.hpp>
```

Public Attributes

- blaze::DynamicMatrix< T, blaze::rowMajor > **idata**
- blaze::DynamicMatrix< T, blaze::rowMajor > **tdata**

3.86.1 Detailed Description

template<typename T, size_t IN, size_t OUT>
struct lipnet::network_data_t< T, IN, OUT >

The network_data_t struct; training dataset.

Template Parameters

T	numerical value type
IN	input dimension
OUT	output dimension

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

- lipnet/include/lipnet/network/backpropagation.hpp

3.87 lipnet::network_libcalc_t< T, N > Struct Template Reference

```
#include <nn_lipcalc.hpp>
```

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integer_sequence< size_t, N... > **DIMS**
- typedef network_topology< T, N... >::type **variable_t**

Static Public Member Functions

- static std::tuple< T, vector_t< sum_from_to< 1, L::value, N... >)> > solve (const variable_t &var)

  solve sdp; via mosek; via interior point method

3.87.1 Detailed Description

template<typename T, size_t ... N>
struct lipnet::network_libcalc_t< T, N >

@breif calculate lipschitz constant of neural network via conic program (SDP)

$$\arg\min_{\Psi,T} \quad \Psi^2 \quad \text{s.t}\chi(\Psi^2, W) \succeq 0$$

Template Parameters

T	numerical value type
N	network topology [2]

3.87.2 Member Function Documentation

3.87.2.1 solve()

```
template<typename T , size_t ...  N>
static std::tuple<T, vector_t<sum_from_to<1, L::value, N...>)> > lipnet::network_libcalc_t<
T, N >::solve (
            const variable_t & var )  [inline], [static]
```

solve sdp; via mosek; via interior point method

Parameters

var	network weights [2]

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

- lipnet/include/lipnet/extern/nn_lipcalc.hpp

3.88 lipnet::network_libtrain_enforcing_t< T, N > Struct Template Reference

network_libtrain_enforcing_; Implementaion of the second subproblem of the admm method

```
#include <nn_liptrain_enforcing.hpp>
```

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integer_sequence< size_t, N... > **DIMS**
- typedef std::integral_constant< size_t, sum_from_to< 1, L::value, N... >) > **n**
- typedef network_t< T, identity_activation_t, N... >::layer_t **variable_t**

Static Public Member Functions

- static variable_t train (const T lipschitz, const T mu, const variable_t &Rvar, const vector_t< n::value > &SDT, const variable_t &dual)

  solve second admm subproblem

3.88.1 Detailed Description

template<typename T, size_t ... N>
struct lipnet::network_libtrain_enforcing_t< T, N >

network_libtrain_enforcing_; Implementaion of the second subproblem of the admm method

$$\arg\min_{\tilde{W},\eta} \quad \operatorname{tr}(Y(W-\tilde{W})) + \frac{v}{2}\eta \quad \text{s.t.} \quad \chi(\Psi^2,\tilde{W}) \succeq 0 \quad \begin{bmatrix}\eta & \operatorname{fl}(W-\tilde{W})\end{bmatrix} \succeq_{\mathcal{Q}} 0$$

Template Parameters

T	numerical value type
N	network topology

3.88.2 Member Function Documentation

3.88.2.1 train()

```
template<typename T , size_t ...  N>
static variable_t lipnet::network_libtrain_enforcing_t< T, N >::train (
             const T lipschitz,
             const T mu,
             const variable_t & Rvar,
             const vector_t< n::value > & SDT,
             const variable_t & dual )  [inline], [static]
```

solve second admm subproblem

Parameters

lipschitz	lipschitz constant
mu	admm hyperparameter; augmented lagrange multipliers
Rvar	refernce weights W
SDT	hyperparameter T of matrix $\chi(\Psi^2, W)$
dual	dual variable

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

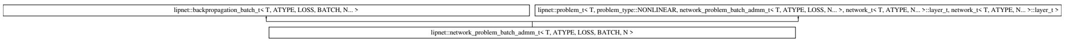
- lipnet/include/lipnet/extern/nn_liptrain_enforcing.hpp

3.89 lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N > Struct Template Reference

The network_problem_batch_admm_t struct. The problem implementation of admm neural network training in batches.

```
#include <nn_problem_batch_admm.hpp>
```

Inheritance diagram for lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N >:


lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... >
lipnet::problem_t< T, problem_type::NONLINEAR, network_problem_batch_admm_t< T, ATYPE, LOSS, N... >, network_t< T, ATYPE, N... >::layer_t, network_t< T, ATYPE, N... >::layer_t >
lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N >

Classes

- struct metainfo_t

Public Types

- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integer_sequence< size_t, N... > **DIMS**
- typedef backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... > **self_back_t**
- typedef self_back_t::variable_t **variable_t**

Public Member Functions

- **network_problem_batch_admm_t** (LOSS< T > &&l, network_data_t< T, at< 0, N... >(), at< L::value, N... >() > &&data, const T rho, const variable_t &dualvariable, const variable_t &weights_bar)
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info) const

  The operator () function. compute gradient.

Public Attributes

- const variable_t & dualvariable

    dual variable
- const variable_t & weights_bar
- const T rho

    admm hyperparameter

3.89.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N >

The network_problem_batch_admm_t struct. The problem implementation of admm neural network training in batches.

$$\nabla_{W,b}\mathcal{L}(f_{W,b}) + L_v(W,\tilde{W},y)$$

Template Parameters

T	Base numeric type (eg. double, float, ...).
ATPYE	Activation type of this neural network.
LOSS	Objectiv function type of this neural network
BATCH	Const integer value specifying the batch size.
N	Neural network topology. Array of postive integer values specifying the number of neurons at each layer.

3.89.2 Member Function Documentation

3.89.2.1 operator()()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ... N>
std::tuple<variable_t,T> lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N >↩
::operator() (
             const variable_t & var,
             metainfo_t & info ) const [inline]
```

The operator () function. compute gradient.

Parameters

var	current position

Returns

gradient and loss at specified position

3.89.3 Member Data Documentation

3.89.3.1 weights_bar

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
const variable_t& lipnet::network_problem_batch_admm_t< T, ATYPE, LOSS, BATCH, N >::weights_↩
bar
```

weights and biases variable x

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

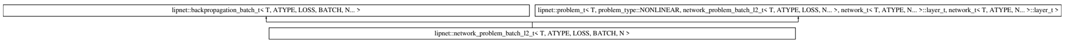
- lipnet/include/lipnet/problem/nn_problem_batch_admm.hpp

3.90 lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N > Struct Template Reference

The network_problem_batch_l2_t struct. The problem implementation of l2 neural network training in batches.

```
#include <nn_problem_batch_l2.hpp>
```

Inheritance diagram for lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N >:


lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... >
lipnet::problem_t< T, problem_type::NONLINEAR, network_problem_batch_l2_t< T, ATYPE, LOSS, N... >, network_t< T, ATYPE, N... >::layer_t, network_t< T, ATYPE, N... >::layer_t >
lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N >

Classes

- struct metainfo_t

Public Types

- typedef backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... > **self_back_t**
- typedef self_back_t::variable_t **variable_t**

Public Member Functions

- network_problem_batch_l2_t (LOSS< T > &&l, network_data_t< T, at< 0, N... >(), at< self_back_t::L↩
::value, N... >() > &&data, const T rho=1.0)

  network_problem_batch_l2_t; default constructor
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info) const

  The operator () function. compute gradient.

Public Attributes

- const T **rho** = 1.0

3.90.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N >

The network_problem_batch_l2_t struct. The problem implementation of l2 neural network training in batches.

$$\nabla_{W,b}\mathcal{L}(f_{W,b}) + \frac{\rho}{2}||W||^2 + \frac{\rho}{2}||b||^2$$

Template Parameters

T	Base numeric type (eg. double, float, ...).
ATPYE	Activation type of this neural network.
LOSS	Objectiv function type of this neural network
BATCH	Const integer value specifying the batch size.
N	Neural network topology. Array of postive integer values specifying the number of neurons at each layer.

3.90.2 Constructor & Destructor Documentation

3.90.2.1 network_problem_batch_l2_t()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N >::network_problem_batch_l2_t (
            LOSS< T > && l,
            network_data_t< T, at< 0, N...  >(), at< self_back_t::L::value, N...  >() > &&
data,
            const T rho = 1.0 )  [inline], [explicit]
```

network_problem_batch_l2_t; default constructor

Parameters

l	loss object
data	traiing data
rho	hyperparameter of L2 regularisation

3.90.3 Member Function Documentation

3.90.3.1 operator()()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_batch_l2_t< T, ATYPE, LOSS, BATCH, N >↩
::operator() (
              const variable_t & var,
              metainfo_t & info ) const  [inline]
```

The operator () function. compute gradient.

Parameters

var	Current position

Returns

Gradient and loss at specified position

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

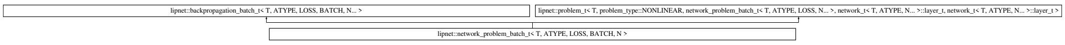
- lipnet/include/lipnet/problem/nn_problem_batch_l2.hpp

3.91 lipnet::network_problem_batch_t< T, ATYPE, LOSS, BATCH, N > Struct Template Reference

The network_problem_batch_t struct. The problem implementation of nominal neural network training in batches.

```
#include <nn_problem_batch.hpp>
```

Inheritance diagram for lipnet::network_problem_batch_t< T, ATYPE, LOSS, BATCH, N >:


lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... >
lipnet::problem_t< T, problem_type::NONLINEAR, network_problem_batch_t< T, ATYPE, LOSS, N... >, network_t< T, ATYPE, N... >::layer_t, network_t< T, ATYPE, N... >::layer_t >
lipnet::network_problem_batch_t< T, ATYPE, LOSS, BATCH, N >

Classes

- struct metainfo_t

Public Types

- typedef backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... > **self_back_t**
- typedef self_back_t::variable_t **variable_t**

Public Member Functions

- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info) const

  The operator () function. compute gradient.

Additional Inherited Members

3.91.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_batch_t< T, ATYPE, LOSS, BATCH, N >

The network_problem_batch_t struct. The problem implementation of nominal neural network training in batches.

$$\nabla_{W,b}\mathcal{L}(f_{W,b})$$

Template Parameters

T	Base numeric type (eg. double, float, ...).
ATPYE	Activation type of this neural network.
LOSS	Objectiv function type of this neural network
BATCH	Const integer value specifying the batch size.
N	Neural network topology. Array of postive integer values specifying the number of neurons at each layer.

3.91.2 Member Function Documentation

3.91.2.1 operator()()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_batch_t< T, ATYPE, LOSS, BATCH, N >::operator()
(
            const variable_t & var,
            metainfo_t & info ) const  [inline]
```

The operator () function. compute gradient.

Parameters

var	current position

Returns

gradient and loss at specified position

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

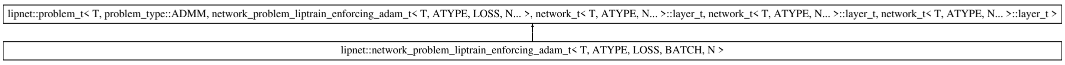
- lipnet/include/lipnet/problem/nn_problem_batch.hpp

3.92 lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N > Struct Template Reference

The network_problem_liptrain_enforcing_adam_t struct. The problem implementation of admm neural network training to enforce lipschitz bound.

```
#include <nn_problem_liptrain_admm.hpp>
```

Inheritance diagram for lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N >:


lipnet::problem_t< T, problem_type::ADMM, network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, N... >, network_t< T, ATYPE, N... >::layer_t, network_t< T, ATYPE, N... >::layer_t, network_t< T, ATYPE, N... >::layer_t >
lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N >

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integer_sequence< size_t, N... > **DIMS**
- typedef network_t< T, ATYPE, N... >::layer_t **variable_t**

Public Member Functions

- network_problem_liptrain_enforcing_adam_t (const network_data_t< T, at< 0, N... >(), at< L::value, N... >() > &&data, const T lip=70.0)

  network_problem_liptrain_enforcing_adam_t; default constructor
- variable_t residual (const variable_t &x, const variable_t &z) const

  The residual method; compute residual.
- variable_t optimize1 (const T rho, const variable_t &var, const variable_t &varbar, const variable_t &dvar) const

  optimize first subproblem; with nominell training; adam method
- variable_t optimize2 (const T rho, const variable_t &var, const variable_t &varbar, const variable_t &dvar) const

  optimize second variable; conic programm; mosek; interior point method
- T loss (const T rho, const variable_t &var, const variable_t &varbar) const

  compute lipschitz constant; mosek; interior point method;

Public Attributes

- network_data_t< T, at< 0, N... >), at< L::value, N... >) > **training_data**
- const T **lipschitz**

3.92.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N >

The network_problem_liptrain_enforcing_adam_t struct. The problem implementation of admm neural network training to enforce lipschitz bound.

Template Parameters

T	Base numeric type (eg. double, float, ...).
ATPYE	Activation type of this neural network.
LOSS	Objectiv function type of this neural network
BATCH	Const integer value specifying the batch size.
N	Neural network topology. Array of postive integer values specifying the number of neurons at each layer.

3.92.2 Constructor & Destructor Documentation

3.92.2.1 network_problem_liptrain_enforcing_adam_t()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO←
SS, size_t BATCH, size_t ...  N>
lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N >::network_problem_liptrain_enfor
(
            const network_data_t< T, at< 0, N...  >(), at< L::value, N...  >() > && data,
            const T lip = 70.0 )  [inline], [explicit]
```

network_problem_liptrain_enforcing_adam_t; default constructor

Parameters

data	training data
lip	lipschitz constant

3.92.3 Member Function Documentation

3.92.3.1 loss()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
T lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N >::loss (
             const T rho,
             const variable_t & var,
             const variable_t & varbar ) const  [inline]
```

compute lipschitz constant; mosek; interior point method;

Parameters

rho	admm hyperparameter; augmented lagrange multiplier
var	first const variable
varbar	second const variable

Returns

lipschitz constant

3.92.3.2 optimize1()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
variable_t lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N >↩
::optimize1 (
             const T rho,
             const variable_t & var,
             const variable_t & varbar,
             const variable_t & dvar ) const  [inline]
```

optimize first subproblem; with nominell training; adam method

```
@f[ \arg \min_{W,b} L_v(W,b,\tilde{W},Y) @f]
```

Parameters

rho	admm hyperparameter; augmented lagrange multiplier
var	variable to optimize
varbar	second const variable
dvar	dual variable

Returns

optimal point var

3.92.3.3 optimize2()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
variable_t lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N >↩
::optimize2 (
             const T rho,
             const variable_t & var,
             const variable_t & varbar,
             const variable_t & dvar ) const  [inline]
```

optimize second variable; conic programm; mosek; interior point method

```
         @f[ \arg \min_{\tilde{W}} L_v(W,b,\tilde{W},Y) @f]
```

Parameters

rho	admm hyperparameter; augmented lagrange multiplier
var	first const variable
varbar	variable
dvar	dual variable

Returns

optimal point varvar

3.92.3.4 residual()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
variable_t lipnet::network_problem_liptrain_enforcing_adam_t< T, ATYPE, LOSS, BATCH, N >↩
::residual (
             const variable_t & x,
             const variable_t & z ) const  [inline]
```

The residual method; compute residual.

Parameters

x	variable
z	variable

Returns

residual

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

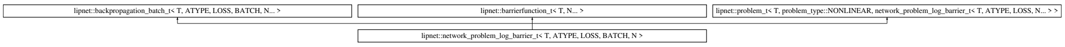
- lipnet/include/lipnet/problem/nn_problem_liptrain_admm.hpp

3.93 lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N > Struct Template Reference

The network_problem_log_barrier_t struct. The problem implementation of barrier neural network training in batches.

```
#include <nn_problem_liptrain_barrier.hpp>
```

Inheritance diagram for lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >:


lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... >
lipnet::barrierfunction_t< T, N... >
lipnet::problem_t< T, problem_type::NONLINEAR, network_problem_log_barrier_t< T, ATYPE, LOSS, N... > >
lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >

Classes

- struct feasibility_t

  The feasibility_t struct. Implementation of feasibility check for this problem.
- struct metainfo_t

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integral_constant< size_t,(N+...) - at< 0, N... >) - at< L::value, N... >) > **TN**
- typedef backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... > **self_back_t**
- typedef barrierfunction_t< T, N... > **self_barrier_t**
- typedef self_barrier_t::variable_t **variable_t**

Public Member Functions

- network_problem_log_barrier_t (LOSS< T > &&l, network_data_t< T, at< 0, N... >(), at< L::value, N... >() > &&data, const T lipschitz=70.0)

  network_problem_log_barrier_t; default constructor
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info, feasibility_t &line, T &gamma) const

  compute gradients
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info, feasibility_t &line) const
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info, const T &gamma) const
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info) const
- template<bool feasibility_enabled = false, bool gamma_enabled = false>
  std::tuple< variable_t, T > run (const variable_t &var, metainfo_t &info, typename std::conditional< feasibility_enabled, feasibility_t, std::void_type >::type &feasibility, typename std::conditional< gamma_enabled, T, std::void_type >::type level) const

  compute gradient of objectiv function

Additional Inherited Members

3.93.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >

The network_problem_log_barrier_t struct. The problem implementation of barrier neural network training in batches.

$$\nabla_{W,b}\mathcal{L}(f_{W,b}) - \rho \log\det(\chi(\Psi^2, W))$$

Template Parameters

T	Base numeric type (eg. double, float, ...).
ATPYE	Activation type of this neural network.
LOSS	Objectiv function type of this neural network
BATCH	Const integer value specifying the batch size.
N	Neural network topology. Array of postive integer values specifying the number of neurons at each layer.

3.93.2 Constructor & Destructor Documentation

3.93.2.1 network_problem_log_barrier_t()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO←
SS, size_t BATCH, size_t ...  N>
lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >::network_problem_log_barrier_t
(
            LOSS< T > && l,
            network_data_t< T, at< 0, N...  >(), at< L::value, N...  >() > && data,
            const T lipschitz = 70.0 )  [inline], [explicit]
```

network_problem_log_barrier_t; default constructor

Parameters

l	loss object
data	training data
lipschitz	lipschitz constant

3.93.3 Member Function Documentation

3.93.3.1 operator()() [1/4]

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >↩
::operator() (
            const variable_t & var,
            metainfo_t & info ) const  [inline]
```

See also

run(const variable_t& var, metainfo_t &info, typename std::conditional<feasibility_enabled, feasibility_t, std↩
::void_type >::type &feasibility, typename std::conditional<gamma_enabled, T, std::void_type >::type level)
const

3.93.3.2 operator()() [2/4]

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >↩
::operator() (
            const variable_t & var,
            metainfo_t & info,
            const T & gamma ) const  [inline]
```

See also

run(const variable_t& var, metainfo_t &info, typename std::conditional<feasibility_enabled, feasibility_t, std↩
::void_type >::type &feasibility, typename std::conditional<gamma_enabled, T, std::void_type >::type level)
const

3.93.3.3 operator()() [3/4]

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >↩
::operator() (
            const variable_t & var,
            metainfo_t & info,
            feasibility_t & line ) const  [inline]
```

See also

run(const variable_t& var, metainfo_t &info, typename std::conditional<feasibility_enabled, feasibility_t, std↩
::void_type >::type &feasibility, typename std::conditional<gamma_enabled, T, std::void_type >::type level)
const

3.93.3.4 operator()() [4/4]

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO←
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >←
::operator() (
            const variable_t & var,
            metainfo_t & info,
            feasibility_t & line,
            T & gamma ) const  [inline]
```

compute gradients

Parameters

var	variable
info	metainfo
line	feasibility check
gamma	hyperparameter

Returns

gradients

See also

run(const variable_t& var, metainfo_t &info, typename std::conditional<feasibility_enabled, feasibility_t, std←::void_type >::type &feasibility, typename std::conditional<gamma_enabled, T, std::void_type >::type level) const

3.93.3.5 run()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO←
SS, size_t BATCH, size_t ...  N>
template<bool feasibility_enabled = false, bool gamma_enabled = false>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_t< T, ATYPE, LOSS, BATCH, N >←
::run (
            const variable_t & var,
            metainfo_t & info,
            typename std::conditional< feasibility_enabled, feasibility_t, std::void_type >←
::type & feasibility,
            typename std::conditional< gamma_enabled, T, std::void_type >::type level ) const
[inline]
```

compute gradient of objectiv function

Template Parameters

feasibility_enabled	enable/disable feasibility checking
gamma_enabled	enable/disable set init hyperparameter gamma

Parameters

var	variable
info	metainfo
line	feasibility check
level	hyperparameter

Returns

gradients

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

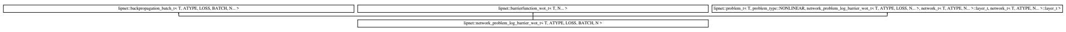
- lipnet/include/lipnet/problem/nn_problem_liptrain_barrier.hpp

3.94 lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N > Struct Template Reference

The network_problem_log_barrier_wot_t struct. The problem implementation of barrier (without T) neural network training in batches.

```
#include <nn_problem_liptrain_barrier_wot.hpp>
```

Inheritance diagram for lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >:


lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... >
lipnet::barrierfunction_wot_t< T, N... >
lipnet::problem_t< T, problem_type::NONLINEAR, network_problem_log_barrier_wot_t< T, ATYPE, LOSS, N... >, network_t< T, ATYPE, N... >::layer_t, network_t< T, ATYPE, N... >::layer_t >
lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >

Classes

- struct feasibility_t

  The feasibility_t struct. Implementation of feasibility check for this problem.
- struct metainfo_t

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integral_constant< size_t,(N+...) - at< 0, N... >) - at< L::value, N... >) > **TN**
- typedef backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... > **self_back_t**
- typedef barrierfunction_wot_t< T, N... > **self_barrier_t**
- typedef self_barrier_t::tparam_t **param_t**
- typedef self_back_t::variable_t **variable_t**

Public Member Functions

- network_problem_log_barrier_wot_t (LOSS< T > &&l, network_data_t< T, at< 0, N... >(), at< L::value, N... >() > &&data, param_t &&tparam, const T lipschitz=70.0)

  network_problem_log_barrier_wot_t; default constructor
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info, feasibility_t &line, T &gamma) const

  compute gradients
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info, feasibility_t &line) const
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info, const T &gamma) const
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info) const
- template<bool feasibility_enabled = false, bool gamma_enabled = false>
  std::tuple< variable_t, T > run (const variable_t &var, metainfo_t &info, typename std::conditional< feasibility_enabled, feasibility_t, std::void_type >::type &feasibility, typename std::conditional< gamma_enabled, T, std::void_type >::type level) const

  compute gradient of objectiv function

Additional Inherited Members

3.94.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >

The network_problem_log_barrier_wot_t struct. The problem implementation of barrier (without T) neural network training in batches.

$$\nabla_{W,b}\mathcal{L}(f_{W,b}) - \rho \log\det(\chi(\Psi^2, W))$$

Template Parameters

T	Base numeric type (eg. double, float, ...).
ATPYE	Activation type of this neural network.
LOSS	Objectiv function type of this neural network
BATCH	Const integer value specifying the batch size.
N	Neural network topology. Array of postive integer values specifying the number of neurons at each layer.

3.94.2 Constructor & Destructor Documentation

3.94.2.1 network_problem_log_barrier_wot_t()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ...  N>
```

```
lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N >::network_problem_log_barrier_wot_t
(
             LOSS< T > && l,
             network_data_t< T, at< 0, N...  >(), at< L::value, N...  >() > && data,
             param_t && tparam,
             const T lipschitz = 70.0 )  [inline], [explicit]
```

network_problem_log_barrier_wot_t; default constructor

Parameters

l	loss object
data	training data
tparam	T hyperparameter from $\chi(\Psi^2, W)$
lipschitz	lipschitz constant

3.94.3 Member Function Documentation

3.94.3.1 operator()() [1/4]

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N
>::operator() (
             const variable_t & var,
             metainfo_t & info ) const  [inline]
```

See also

run(const variable_t& var, metainfo_t &info, typename std::conditional<feasibility_enabled, feasibility_t, std↩ ::void_type >::type &feasibility, typename std::conditional<gamma_enabled, T, std::void_type >::type level) const

3.94.3.2 operator()() [2/4]

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N
>::operator() (
             const variable_t & var,
             metainfo_t & info,
             const T & gamma ) const  [inline]
```

See also

run(const variable_t& var, metainfo_t &info, typename std::conditional<feasibility_enabled, feasibility_t, std↩ ::void_type >::type &feasibility, typename std::conditional<gamma_enabled, T, std::void_type >::type level) const

3.94.3.3 operator()() [3/4]

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N
>::operator() (
            const variable_t & var,
            metainfo_t & info,
            feasibility_t & line ) const  [inline]
```

See also

> run(const variable_t& var, metainfo_t &info, typename std::conditional<feasibility_enabled, feasibility_t, std↩ ::void_type >::type &feasibility, typename std::conditional<gamma_enabled, T, std::void_type >::type level) const

3.94.3.4 operator()() [4/4]

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N
>::operator() (
            const variable_t & var,
            metainfo_t & info,
            feasibility_t & line,
            T & gamma ) const  [inline]
```

compute gradients

Parameters

var	variable
info	metainfo
line	feasibility check
gamma	hyperparameter

Returns

> gradients

See also

> run(const variable_t& var, metainfo_t &info, typename std::conditional<feasibility_enabled, feasibility_t, std↩ ::void_type >::type &feasibility, typename std::conditional<gamma_enabled, T, std::void_type >::type level) const

3.94.3.5 run()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
template<bool feasibility_enabled = false, bool gamma_enabled = false>
std::tuple<variable_t,T> lipnet::network_problem_log_barrier_wot_t< T, ATYPE, LOSS, BATCH, N
>::run (
            const variable_t & var,
            metainfo_t & info,
            typename std::conditional< feasibility_enabled, feasibility_t, std::void_type >↩
::type & feasibility,
            typename std::conditional< gamma_enabled, T, std::void_type >::type level ) const
[inline]
```

compute gradient of objectiv function

Template Parameters

feasibility_enabled	enable/disable feasibility checking
gamma_enabled	enable/disable set init hyperparameter gamma

Parameters

var	variable
info	metainfo
line	feasibility check
level	hyperparameter

Returns

gradients

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

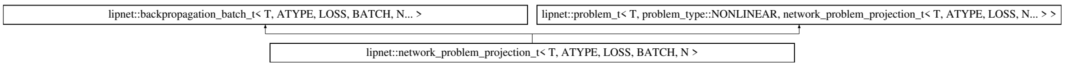
- lipnet/include/lipnet/problem/nn_problem_liptrain_barrier_wot.hpp

3.95 lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N > Struct Template Reference

The network_problem_projection_t struct. The problem implementation of projected neural network training in batches.

```
#include <nn_problem_liptrain_projection.hpp>
```

Inheritance diagram for lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >:


lipnet::backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... >
lipnet::problem_t< T, problem_type::NONLINEAR, network_problem_projection_t< T, ATYPE, LOSS, N... > >
lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >

Classes

- struct metainfo_t

Public Types

- template<size_t NN>
  using **vector_t** = blaze::StaticVector< T, NN, blaze::columnVector >
- template<size_t NN1, size_t NN2>
  using **matrix_t** = blaze::StaticMatrix< T, NN1, NN2, blaze::rowMajor >
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **QN**
- typedef std::integral_constant< size_t,(N+...) - at< 0, N... >) - at< L::value, N... >) > **TN**
- typedef std::integer_sequence< size_t, N... > **DIMS**
- typedef backpropagation_batch_t< T, ATYPE, LOSS, BATCH, N... > **self_back_t**
- typedef self_back_t::variable_t **variable_t**

Public Member Functions

- network_problem_projection_t (LOSS< T > &&l, network_data_t< T, at< 0, N... >(), at< L::value, N... >() > &&data, const T &lip=70.0, const T &tparam=100.0)

  network_problem_projection_t; default constructor
- std::tuple< variable_t, T > operator() (const variable_t &var, metainfo_t &info) const

  compute gradient of objectiv function linke nominell training
- variable_t projection (variable_t &&var) const

  The projection method. Compute projection.

Public Attributes

- T **lipschitz**
- T **tparaminit**

3.95.1 Detailed Description

template<typename T, template< typename > typename ATYPE, template< typename > typename LOSS, size_t BATCH, size_t ... N>
struct lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >

The network_problem_projection_t struct. The problem implementation of projected neural network training in batches.

$$\nabla_{W,b}\mathcal{L}(f_{W,b})$$

Template Parameters

T	Base numeric type (eg. double, float, ...).
ATPYE	Activation type of this neural network.
LOSS	Objectiv function type of this neural network
BATCH	Const integer value specifying the batch size.
N	Neural network topology. Array of postive integer values specifying the number of neurons at each layer.

3.95.2 Constructor & Destructor Documentation

3.95.2.1 network_problem_projection_t()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >::network_problem_projection_t
(
             LOSS< T > && l,
             network_data_t< T, at< 0, N...  >(), at< L::value, N...  >() > && data,
             const T & lip = 70.0,
             const T & tparam = 100.0 )  [inline], [explicit]
```

network_problem_projection_t; default constructor

Parameters

l	loss object
data	tarining data
lip	lipschitz constant
tparam	T hyperparameter from $\chi(\Psi^2, W)$

3.95.3 Member Function Documentation

3.95.3.1 operator()()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
std::tuple<variable_t,T> lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >↩
::operator() (
             const variable_t & var,
             metainfo_t & info ) const  [inline]
```

compute gradient of objectiv function linke nominell training

Parameters

var	variable
info	metainfo

Returns

gradients

3.95.3.2 projection()

```
template<typename T , template< typename > typename ATYPE, template< typename > typename LO↩
SS, size_t BATCH, size_t ...  N>
variable_t lipnet::network_problem_projection_t< T, ATYPE, LOSS, BATCH, N >::projection (
             variable_t && var ) const  [inline]
```

The projection method. Compute projection.

$$\min ||W - \tilde{W}||^2 \quad \text{s.t} \quad \chi(\Psi^2, W) \succeq 0$$

See also

lipnet::mosek_projection_wot_t

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

- lipnet/include/lipnet/problem/nn_problem_liptrain_projection.hpp

3.96 lipnet::network_t< T, ATYPE, N > Struct Template Reference

The network_t struct; neural network implementation.

`#include <network.hpp>`

Classes

- struct data_serialization_t

  serialization helper struct
- struct topology_serialization_t

  serialization helper struct

Public Types

- typedef network_topology< T, N... >::type **layer_t**
- typedef std::integral_constant< size_t, sizeof...(N) -1 > **L**
- typedef std::integral_constant< size_t,(N+...)> **NL**
- typedef std::integer_sequence< size_t, N... > **DIMS**
- typedef blaze::StaticVector< T, at< L::value, N... >), blaze::columnVector > **outvec_t**
- typedef blaze::StaticVector< T, at< 0, N... >), blaze::columnVector > **invec_t**

Public Member Functions

- outvec_t query (const invec_t &input) const

  query the neural network
- template<class Archive >
  void save (Archive &ar) const

  serialize network
- template<class Archive >
  void load (Archive &ar)

  deserialize network

Public Attributes

- layer_t layers

  weights and biases

3.96.1 Detailed Description

template<typename T, template< typename > typename ATYPE, size_t ... N>
struct lipnet::network_t< T, ATYPE, N >

The network_t struct; neural network implementation.

Template Parameters

T	numerical value type
ATYPE	activation function type
N	network topology

3.96.2 Member Function Documentation

3.96.2.1 query()

```
template<typename T , template< typename > typename ATYPE, size_t ...  N>
outvec_t lipnet::network_t< T, ATYPE, N >::query (
                const invec_t & input ) const  [inline]
```

query the neural network

$$z_l = W_l x_l \quad x_{l+1} = \sigma(z_l) \quad \cdots$$

Parameters

input	vector

Returns

output vector

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

- lipnet/include/lipnet/network/network.hpp

3.97 lipnet::network_topology< T, NI, NO, NARGS > Struct Template Reference

```
#include <topology.hpp>
```

Public Types

- typedef network_topology_impl< T, NI, NO, NARGS... >::type **type**

3.97.1 Detailed Description

template<typename T, size_t NI, size_t NO, size_t ... NARGS>
struct lipnet::network_topology< T, NI, NO, NARGS >

See also

network_topology_impl

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

- lipnet/include/lipnet/network/topology.hpp

3.98 lipnet::network_topology_impl< T, NI, NO, NS > Struct Template Reference

newtork layer holder and creator struct; helper struct to create compile time layers in stack memory -> performance

```
#include <topology.hpp>
```

Public Types

- typedef network_topology_impl< T, NO, NS... >::type **next**
- typedef join_tuples< std::tuple< layer_t< T, NI, NO > >, next >::type **type**

3.98.1 Detailed Description

template<typename T, size_t NI, size_t NO, size_t ... NS>
struct lipnet::network_topology_impl< T, NI, NO, NS >

newtork layer holder and creator struct; helper struct to create compile time layers in stack memory -> performance

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

- lipnet/include/lipnet/network/topology.hpp

3.99 lipnet::network_topology_impl< T, NI, NO > Struct Template Reference

```
#include <topology.hpp>
```

Public Types

- typedef std::tuple< layer_t< T, NI, NO > > **type**

3.99.1 Detailed Description

template<typename T, size_t NI, size_t NO>
struct lipnet::network_topology_impl< T, NI, NO >

See also

network_topology_impl

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

- lipnet/include/lipnet/network/topology.hpp

3.100 std::nonesuch Struct Reference

Public Member Functions

- **nonesuch** (nonesuch const &)=delete
- void **operator=** (nonesuch const &)=delete

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

- lipnet/include/lipnet/traits.hpp

3.101 lipnet::norm_t< T, V > Struct Template Reference

The norm_t struct. Just a interface for all possible types. Compute norm of argument.

```
#include <variable.hpp>
```

3.101.1 Detailed Description

template<typename T, typename V>
struct lipnet::norm_t< T, V >

The norm_t struct. Just a interface for all possible types. Compute norm of argument.

$$||V||_2$$

.

Template Parameters

T	numerical value type
V	tensor type of argument

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

- lipnet/include/lipnet/variable.hpp

3.102 lipnet::norm_t< T, blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > > Struct Template Reference

The norm_t struct for blaze::StaticMatrix.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static T norm (const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > &m)

  The norm method. Compute norm of vector m. $||m||_{\mathrm{2-ind.}}$.

3.102.1 Detailed Description

template<typename T, size_t N1, size_t N2>
struct lipnet::norm_t< T, blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > >

The norm_t struct for blaze::StaticMatrix.

Template Parameters

T	numerical value type
N1	row dimension of argument
N2	column dimension of argument

See also

lipnet::norm_t [6]

3.102.2 Member Function Documentation

3.102.2.1 norm()

```
template<typename T , size_t N1, size_t N2>
static T lipnet::norm_t< T, blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > >::norm (
            const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > & m )  [inline], [static]
```

The norm method. Compute norm of vector m. $||m||_{2-\mathrm{ind.}}$.

Parameters

m	input matrix

Returns

norm of matrix m

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

- lipnet/include/lipnet/tensor.hpp

3.103 lipnet::norm_t< T, blaze::StaticVector< T, N, blaze::columnVector > > Struct Template Reference

The norm_t struct for blaze::StaticVector.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static T norm (const blaze::StaticVector< T, N, blaze::columnVector > &m)

  The norm method. Compute norm of vector m. $||m||_2$.

3.103.1 Detailed Description

template<typename T, size_t N>
struct lipnet::norm_t< T, blaze::StaticVector< T, N, blaze::columnVector > >

The norm_t struct for blaze::StaticVector.

Template Parameters

T	numerical value type
N	dimension of argument

See also

lipnet::norm_t [6]

3.103.2 Member Function Documentation

3.103.2.1 norm()

```
template<typename T , size_t N>
static T lipnet::norm_t< T, blaze::StaticVector< T, N, blaze::columnVector > >::norm (
            const blaze::StaticVector< T, N, blaze::columnVector > & m )  [inline], [static]
```

The norm method. Compute norm of vector m. $||m||_2$.

Parameters

m	input vector

Returns

norm of vector m

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

- lipnet/include/lipnet/tensor.hpp

3.104 lipnet::norm_t< T, layer_t< T, I, O > > Struct Template Reference

Static Public Member Functions

- static T **norm** (const layer_t< T, I, O > &m)

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

- lipnet/include/lipnet/network/layer.hpp

3.105 lipnet::norm_t< T, lipcalc_parameter_t< T, N... > > Struct Template Reference

Static Public Member Functions

- static T **norm** (const lipcalc_parameter_t< T, N... > &m)

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

- lipnet/include/lipnet/lipschitz/parameter.hpp

3.106 lipnet::norm_t< T, liptrainweights_t< T, N... > > Struct Template Reference

Static Public Member Functions

- static T **norm** (const liptrainweights_t< T, N... > &m)

Public Attributes

- decltype(liptrainweights_t< T, N... >::W) typedef **arg1_t**
- decltype(liptrainweights_t< T, N... >::t) typedef **arg2_t**

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

- lipnet/include/lipnet/lipschitz/barrier.hpp

3.107 lipnet::norm_t< T, std::tuple< ARGS... > > Struct Template Reference

Static Public Member Functions

- template<size_t ... INTS>
  static T **norm_impl** (const std::tuple< ARGS... > &m, std::integer_sequence< size_t, INTS... >)
- static T **norm** (const std::tuple< ARGS... > &m)

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

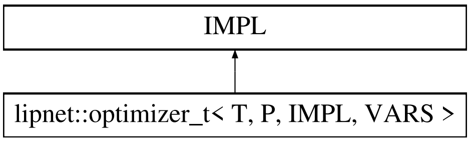
- lipnet/include/lipnet/tuple.hpp

3.108 lipnet::optimizer_t< T, P, IMPL, VARS > Struct Template Reference

The optimizer_t struct. On instantiation a class with the implementation as base class will be created.

```
#include <optimizer.hpp>
```

Inheritance diagram for lipnet::optimizer_t< T, P, IMPL, VARS >:



Classes

- struct main_statistics_t

  The main_statistics_t struct.
- struct stats_type_exists
- struct stats_type_exists< TT, typename void_t< typename TT::statistics_t >::type >
- struct void_t

  Type holder.

Public Types

- typedef stats_type_exists< P >::type **statistics_problem_t**

Public Member Functions

- template<bool stats_enabled = false>
  std::tuple< VARS..., T > run (P &prob, VARS &&...vars, typename std::conditional< stats_enabled, main_statistics_t, std::void_type >::type &stats) const

  The mainoptimization function.
- std::tuple< VARS..., T > operator() (P &prob, VARS &&...vars, main_statistics_t &stats) const

  The operator() function. A wrapper for run(P &prob, VARS&& ...vars , typename std::conditional<stats_enabled, main_statistics_t, std::void_type >::type &stats) with statistics enabled.
- std::tuple< VARS..., T > operator() (P &prob, VARS &&...vars) const

  The operator() function. A wrapper for run(P &prob, VARS&& ...vars , typename std::conditional<stats_enabled, main_statistics_t, std::void_type >::type &stats) with statistics disabled.

3.108.1 Detailed Description

template<typename T, typename P, typename IMPL, typename ... VARS>
struct lipnet::optimizer_t< T, P, IMPL, VARS >

The optimizer_t struct. On instantiation a class with the implementation as base class will be created.

Template Parameters

T	The numeric base type (e.g. double, float, ...)
P	The problem struct, which should be solved (e.g. lasso_problem, ...)
IMPL	The implementation of the solver, which should be used
VARS	Parameterpack of all type the implementation needs to solve the problem (e.g. VAR, GRADIENT, DUAL, ...)

3.108.2 Member Function Documentation

3.108.2.1 operator()() [1/2]

```
template<typename T , typename P , typename IMPL , typename ...  VARS>
std::tuple<VARS...,T> lipnet::optimizer_t< T, P, IMPL, VARS >::operator() (
            P & prob,
            VARS &&...  vars ) const  [inline]
```

The operator() function. A wrapper for run(P &prob, VARS&& ...vars , typename std::conditional<stats_enabled, main_statistics_t, std::void_type >::type &stats) with statistics disabled.

Parameters

prob	
vars	
stats	

Returns

Optimal value and optimal loss

See also

run(P &prob, VARS&& ...vars , typename std::conditional<stats_enabled, main_statistics_t, std::void_type >::type &stats)

3.108.2.2 operator()() [2/2]

```
template<typename T , typename P , typename IMPL , typename ...  VARS>
std::tuple<VARS...,T> lipnet::optimizer_t< T, P, IMPL, VARS >::operator() (
            P & prob,
            VARS &&...  vars,
            main_statistics_t & stats ) const  [inline]
```

The operator() function. A wrapper for run(P &prob, VARS&& ...vars , typename std::conditional<stats_enabled, main_statistics_t, std::void_type >::type &stats) with statistics enabled.

Parameters

prob	
vars	
stats	

Returns

Optimal value and optimal loss

See also

run(P &prob, VARS&& ...vars , typename std::conditional<stats_enabled, main_statistics_t, std::void_type >::type &stats)

3.108.2.3 run()

```
template<typename T , typename P , typename IMPL , typename ...  VARS>
template<bool stats_enabled = false>
std::tuple<VARS...,T> lipnet::optimizer_t< T, P, IMPL, VARS >::run (
             P & prob,
             VARS &&...  vars,
             typename std::conditional< stats_enabled, main_statistics_t, std::void_type >↩
::type & stats ) const  [inline]
```

The mainoptimization function.

Template Parameters

stats_enabled	Boolean value to decide if you want to create a statistic about this optimization process.

Parameters

prob	The problem variable
vars	The initial values over which you want to optimize
stats	The statistics struct if you want to create statistics or just a void_type if not.

Returns

Optimal value and optimal loss

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

- lipnet/include/lipnet/optimizer.hpp

3.109 lipnet::parameter_decompo_t< T, N > Struct Template Reference

Public Attributes

- decompos_subentry< T, N... >::type **subdiagonals**
- decompos_diagentry< T, N... >::type **diagonals**

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.110 lipnet::adam_barrier_t_impl< T, P, VAR, GRAD, feasibility_enabled >::parameter_t Struct Reference

The parameter_t struct; all meta parameters for optimisation.

```
#include <adam_barrier.hpp>
```

Public Attributes

- size_t **max_iter**
- size_t cpsteps

  maximal iterations (default = 5e5)
- T diff

  central path steps (default = 5)
- T threshold

  stopping criterion loss difference (default = 1e-10)
- size_t window

  stopping criterion window threshold (default = 1e-8)
- T gamma

  stopping criterion window size (default = 300)
- T alpha

  barriere factor (default = 1)
- T beta1

  stepsize (default = 0.02)
- T beta2

  adam meta parameter beta1 (default = 0.9)
- T beta3

  adam meta parameter beta2 (default = 0.999)
- T alphadec

  meta parameter loss difference decrease factor (default = 5.0)
- T gammadec

  meta parameter stepsize decrease factor (default = 0.5)
- T eps

  meta parameter gamma decrease factor (default = 0.5)

3.110.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD, bool feasibility_enabled = false>
struct lipnet::adam_barrier_t_impl< T, P, VAR, GRAD, feasibility_enabled >::parameter_t

The parameter_t struct; all meta parameters for optimisation.

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

- lipnet/include/lipnet/optimizer/adam_barrier.hpp

3.111 lipnet::fast_gradient_descent_t_impl< T, P, VAR, GRAD >::parameter_t Struct Reference

Public Attributes

- T **gamma**
- T eps

  stepsize (default = 0.001)

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

- lipnet/include/lipnet/optimizer/fast_gradient_descent.hpp

3.112 lipnet::adam_momentum_t_impl< T, P, VAR, GRAD >::parameter_t Struct Reference

Public Attributes

- size_t **max_iter**
- T diff

  max iterations (default = 5e5)
- T graddiff

  stopping criterion loss difference (default = 1e-10)
- T alpha

  stopping criterion gradient norm (default = 1e-4)
- T beta1

  stepsize (default = 0.02)
- T beta2

  adam meta parameter beta1 (default = 0.9)
- T eps

  adam meta parameter beta2 (default = 0.999)

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

- lipnet/include/lipnet/optimizer/adam_momentum.hpp

3.113 lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::parameter_t Struct Reference

Public Attributes

- size_t **max_iter**
- T rho

  max iterations (default = 1e4)
- T eps

  admm hyperparameter (augmented lagrange multiplier parameter) (default = 2)

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

- lipnet/include/lipnet/optimizer/admm_optimizer.hpp

3.114 lipnet::adam_projected_t_impl< T, P, VAR, GRAD >::parameter_t Struct Reference

Public Attributes

- size_t **max_iter**
- T diff

  max iterations (default = 5e5)
- T threshold

  stopping criterion loss difference (default = 1e-10)
- size_t window

  stopping criterion window threshold (default = 1e-8)
- T alpha

  stopping criterion window size (default = 300)
- T beta1

  stepsize (default = 0.02)
- T beta2

  adam meta parameter beta1 (default = 0.9)
- T eps

  adam meta parameter beta2 (default = 0.999)

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

- lipnet/include/lipnet/optimizer/adam_projected.hpp

3.115 lipnet::gradient_descent_projected_t_impl< T, P, VAR, GRAD >::parameter_t Struct Reference

Public Attributes

- size_t **max_iter**
- T diff

  max iterations (default = 5e5)
- T gamma

  stopping criterion loss difference (default = 1e-6)
- T eps

  stepsize (default = 0.001)

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

- lipnet/include/lipnet/optimizer/gradient_descent_projected.hpp

3.116 lipnet::parameter_tparam< T, N, NARGS > Struct Template Reference

Public Types

- typedef parameter_tparam_impl< T, NARGS... >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.117 lipnet::parameter_tparam_impl< T, N, NS > Struct Template Reference

Public Types

- typedef parameter_tparam_impl< T, NS... >::type **next**
- typedef join_tuples< std::tuple< blaze::StaticVector< T, N, blaze::columnVector > >, next >::type **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.118 lipnet::parameter_tparam_impl< T, N, R > Struct Template Reference

Public Types

- typedef std::tuple< blaze::StaticVector< T, N, blaze::columnVector > > **type**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.119 lipnet::problem_t< T, TYPE, IMPL, ARGS > Struct Template Reference

The problem_t struct; base problem struct (basically a placerholder class)

```
#include <problem.hpp>
```

3.119.1 Detailed Description

template<typename T, problem_type TYPE, typename IMPL, typename ... ARGS>
struct lipnet::problem_t< T, TYPE, IMPL, ARGS >

The problem_t struct; base problem struct (basically a placerholder class)

Template Parameters

T	numerical value type
TYPE	problem class
IMPL	actual problem struct
ARGS	problem specific types (passthrough)

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

- lipnet/include/lipnet/problem.hpp

3.120 lipnet::prod_t< T, V1, V2 > Struct Template Reference

The prod_t struct. Just a interface for all possible types. Compute inner/outer/... products.

```
#include <variable.hpp>
```

3.120.1 Detailed Description

template<typename T, typename V1, typename V2>
struct lipnet::prod_t< T, V1, V2 >

The prod_t struct. Just a interface for all possible types. Compute inner/outer/... products.

$$V_1 V_2^\top; \;\; V_1^\top V_2; \;\; \cdots$$

.

Template Parameters

T	numerical value type
V1	tensor type of first argument
V2	tensot type of second argument

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

- lipnet/include/lipnet/variable.hpp

3.121 lipnet::prod_t< T, blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >, blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > > Struct Template Reference

The prod_t struct for blaze::StaticMatrix.

```
#include <tensor.hpp>
```

Static Public Member Functions

- static T inner (const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > &m1, const blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > &m2)

  The inner method. Implemention of the inner product of blaze::StaticVector type. $m_1^\top m_2$.
- static auto outer (const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > &m1, const blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > &m2)

  The outer method. Implemention of the outer product of blaze::StaticMatrix type.

3.121.1 Detailed Description

template<typename T, size_t N1, size_t N2, size_t N3, size_t N4>
struct lipnet::prod_t< T, blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >, blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > >

The prod_t struct for blaze::StaticMatrix.

Template Parameters

T	numerical value type
N1	row dimension of first argument
N2	column dimension of first argument
N3	row dimension of second argument
N4	column dimension of second argument

See also

lipnet::prod_t [6]

3.121.2 Member Function Documentation

3.121.2.1 inner()

```
template<typename T , size_t N1, size_t N2, size_t N3, size_t N4>
static T lipnet::prod_t< T, blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >, blaze::↩
StaticMatrix< T, N3, N4, blaze::rowMajor > >::inner (
            const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > & m1,
            const blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > & m2 )  [inline], [static]
```

The inner method. Implemention of the inner product of blaze::StaticVector type. $m_1^\top m_2$.

Parameters

m1	first argument (blaze::StaticVector<T,N1,blaze::columnVector>)
m2	second argument (blaze::StaticVector<T,N2,blaze::columnVector>)

Returns

inner product of m1 and m2

3.121.2.2 outer()

```
template<typename T , size_t N1, size_t N2, size_t N3, size_t N4>
static auto lipnet::prod_t< T, blaze::StaticMatrix< T, N1, N2, blaze::rowMajor >, blaze::↩
StaticMatrix< T, N3, N4, blaze::rowMajor > >::outer (
            const blaze::StaticMatrix< T, N1, N2, blaze::rowMajor > & m1,
            const blaze::StaticMatrix< T, N3, N4, blaze::rowMajor > & m2 )  [inline], [static]
```

The outer method. Implemention of the outer product of blaze::StaticMatrix type.

Parameters

m1	first argument (blaze::StaticMatrix<T,N1,N2,blaze::rowMajor>)
m2	second argument (blaze::StaticMatrix<T,N3,N4,blaze::rowMajor>)

Returns

kronecker product of m1 and m2

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

- lipnet/include/lipnet/tensor.hpp

3.122 lipnet::prod_t< T, blaze::StaticVector< T, N1, blaze::columnVector >, blaze::StaticVector< T, N2, blaze::columnVector > > Struct Template Reference

The prod_t struct for blaze::StaticVector.

`#include <tensor.hpp>`

Static Public Member Functions

- static T inner (const blaze::StaticVector< T, N1, blaze::columnVector > &m1, const blaze::StaticVector< T, N2, blaze::columnVector > &m2)

  The inner method. Implemention of the inner product of blaze::StaticVector type. $m_1^\top m_2$.
- static auto outer (const blaze::StaticVector< T, N1, blaze::columnVector > &m1, const blaze::StaticVector< T, N2, blaze::columnVector > &m2)

  The outer method. Implemention of the outer product of blaze::StaticVector type. $m_1 m_2^\top$.

3.122.1 Detailed Description

template<typename T, size_t N1, size_t N2>
struct lipnet::prod_t< T, blaze::StaticVector< T, N1, blaze::columnVector >, blaze::StaticVector< T, N2, blaze::columnVector > >

The prod_t struct for blaze::StaticVector.

Template Parameters

T	numerical value type
N1	dimension of first argument
N2	dimension of second argument

See also

lipnet::prod_t [6]

3.122.2 Member Function Documentation

3.122.2.1 inner()

```
template<typename T , size_t N1, size_t N2>
static T lipnet::prod_t< T, blaze::StaticVector< T, N1, blaze::columnVector >, blaze::↩
StaticVector< T, N2, blaze::columnVector > >::inner (
            const blaze::StaticVector< T, N1, blaze::columnVector > & m1,
            const blaze::StaticVector< T, N2, blaze::columnVector > & m2 )  [inline], [static]
```

The inner method. Implemention of the inner product of blaze::StaticVector type. $m_1^\top m_2$.

Parameters

m1	first argument (blaze::StaticVector<T,N1,blaze::columnVector>)
m2	second argument (blaze::StaticVector<T,N2,blaze::columnVector>)

Returns

inner product of m1 and m2

3.122.2.2 outer()

```
template<typename T , size_t N1, size_t N2>
static auto lipnet::prod_t< T, blaze::StaticVector< T, N1, blaze::columnVector >, blaze::↩
StaticVector< T, N2, blaze::columnVector > >::outer (
            const blaze::StaticVector< T, N1, blaze::columnVector > & m1,
            const blaze::StaticVector< T, N2, blaze::columnVector > & m2 )  [inline], [static]
```

The outer method. Implemention of the outer product of blaze::StaticVector type. $m_1 m_2^\top$.

Parameters

m1	first argument (blaze::StaticVector<T,N1,blaze::columnVector>)
m2	second argument (blaze::StaticVector<T,N2,blaze::columnVector>)

Returns

outer product of m1 and m2

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

- lipnet/include/lipnet/tensor.hpp

3.123 lipnet::prod_t< T, layer_t< T, I1, O1 >, layer_t< T, I2, O2 > > Struct Template Reference

Static Public Member Functions

- static T **inner** (const layer_t< T, I1, O1 > &m1, const layer_t< T, I2, O2 > &m2)

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

- lipnet/include/lipnet/network/layer.hpp

3.124 lipnet::prod_t< T, lipcalc_parameter_t< T, N... >, lipcalc_parameter_t< T, N... > > Struct Template Reference

Static Public Member Functions

- static T **inner** (const lipcalc_parameter_t< T, N... > &m1, const lipcalc_parameter_t< T, N... > &m2)

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

- lipnet/include/lipnet/lipschitz/parameter.hpp

3.125 lipnet::prod_t< T, liptrainweights_t< T, N... >, liptrainweights_t< T, N... > > Struct Template Reference

Static Public Member Functions

- static T **inner** (const liptrainweights_t< T, N... > &m1, const liptrainweights_t< T, N... > &m2)

Public Attributes

- decltype(liptrainweights_t< T, N... >::W) typedef **arg1_t**
- decltype(liptrainweights_t< T, N... >::t) typedef **arg2_t**

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

- lipnet/include/lipnet/lipschitz/barrier.hpp

3.126 lipnet::prod_t< T, parameter_decompo_t< T, N... >, parameter_decompo_t< T, N... > > Struct Template Reference

Static Public Member Functions

- static T **inner** (const parameter_decompo_t< T, N... > &m1, const parameter_decompo_t< T, N... > &m2)

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

- lipnet/include/lipnet/lipschitz/decompos.hpp

3.127 lipnet::prod_t< T, std::tuple< ARGS1... >, std::tuple< ARGS2... > > Struct Template Reference

Static Public Member Functions

- template<size_t ... INTS>
  static T **inner_impl** (const std::tuple< ARGS1... > &m1, const std::tuple< ARGS2... > &m2, std::integer_sequence< size_t, INTS... >)
- static T **inner** (const std::tuple< ARGS1... > &m1, const std::tuple< ARGS2... > &m2)

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

- lipnet/include/lipnet/tuple.hpp

3.128 lipnet::series_t< T > Struct Template Reference

The series_t struct. Base struct for logging.

```
#include <statistics.hpp>
```

Public Member Functions

- **series_t** (const size_t size=0)
- T & **operator()** (const size_t index)
- series_t< T > & **operator<<** (const T point)

Public Attributes

- std::vector< T > **data**

3.128.1 Detailed Description

template<typename T>
struct lipnet::series_t< T >

The series_t struct. Base struct for logging.

Template Parameters

T	numerical type

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

- lipnet/include/lipnet/statistics.hpp

3.129 lipnet::solve_function_helper< P, VAR > Struct Template Reference

Public Types

- template<typename T >
  using **member_solve_t** = decltype(std::declval< T >().solve(std::declval< const VAR & >()))

Static Public Attributes

- constexpr static bool **value** = std::is_detected<member_solve_t, P>::value

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

- lipnet/include/lipnet/optimizer.hpp

3.130 lipnet::squared_error_t< T > Struct Template Reference

The squared_error_t struct; implementation of the squarred error objective function.

```
#include <loss.hpp>
```

Public Types

- template<typename TT , size_t O, size_t I>
  using **matrix_t** = blaze::StaticMatrix< TT, O, I, blaze::columnMajor >
- template<typename TT , size_t N>
  using **vector_t** = blaze::StaticVector< TT, N, blaze::columnVector >

Public Member Functions

- template<size_t N, size_t BATCH = 0, typename std::enable_if<!(BATCH<=0), int >::type = 0>
  T evaluate (const matrix_t< T, N, BATCH > &target, const matrix_t< T, N, BATCH > &data) const

  The evaluate function; compute loss.

3.130.1 Detailed Description

template<typename T>
struct lipnet::squared_error_t< T >

The squared_error_t struct; implementation of the squarred error objective function.

Template Parameters

T	numerical value type
TYPE	choose the activation type

3.130.2 Member Function Documentation

3.130.2.1 evaluate()

```
template<typename T >
template<size_t N, size_t BATCH = 0, typename std::enable_if<!(BATCH<=0), int >::type = 0>
T lipnet::squared_error_t< T >::evaluate (
            const matrix_t< T, N, BATCH > & target,
            const matrix_t< T, N, BATCH > & data ) const  [inline]
```

The evaluate function; compute loss.

$$\mathcal{L}(x,y) = (x-y)^\top(x-y)$$

Template Parameters

N	input dimension type
BATCH	batch size

Parameters

target	real value
estimated	value

Returns

loss

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

- lipnet/include/lipnet/network/loss.hpp

3.131 lipnet::statistics_helper Struct Reference

The statistics_helper struct. Helper function to disable logging for performence reasons if it is desired.

```
#include <statistics.hpp>
```

Classes

- struct stats_type_exists
- struct stats_type_exists< TT, typename void_t< typename TT::statistics_t >::type >
- struct void_t

3.131.1 Detailed Description

The statistics_helper struct. Helper function to disable logging for performence reasons if it is desired.

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

- lipnet/include/lipnet/statistics.hpp

3.132 lipnet::gradient_descent_projected_t_impl< T, P, VAR, GRAD >::statistics_t Struct Reference

problem specific implementation of statistics_t

```
#include <gradient_descent_projected.hpp>
```

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &archive)

Public Attributes

- series_t< T > **loss**

3.132.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD>
struct lipnet::gradient_descent_projected_t_impl< T, P, VAR, GRAD >::statistics_t

problem specific implementation of statistics_t

See also

lipnet statistics_t [4]

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

- lipnet/include/lipnet/optimizer/gradient_descent_projected.hpp

3.133 lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::statistics_t Struct Reference

problem specific implementation of statistics_t

```
#include <admm_optimizer.hpp>
```

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &archive)

Public Attributes

- series_t< T > **loss**

3.133.1 Detailed Description

template<typename T, typename P, typename X, typename Z, typename DUAL>
struct lipnet::admm_optimizer_t_impl< T, P, X, Z, DUAL >::statistics_t

problem specific implementation of statistics_t

See also

lipnet statistics_t [4]

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

- lipnet/include/lipnet/optimizer/admm_optimizer.hpp

3.134 lipnet::adam_barrier_t_impl< T, P, VAR, GRAD, feasibility_enabled >::statistics_t Struct Reference

problem specific implementation of statistics_t

```
#include <adam_barrier.hpp>
```

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &archive)

Public Attributes

- series_t< T > **loss**

3.134.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD, bool feasibility_enabled = false>
struct lipnet::adam_barrier_t_impl< T, P, VAR, GRAD, feasibility_enabled >::statistics_t

problem specific implementation of statistics_t

See also

lipnet statistics_t [4]

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

- lipnet/include/lipnet/optimizer/adam_barrier.hpp

3.135 lipnet::fast_gradient_descent_t_impl< T, P, VAR, GRAD >::statistics_t Struct Reference

problem specific implementation of statistics_t

```
#include <fast_gradient_descent.hpp>
```

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &archive)

Public Attributes

- series_t< T > **loss**

3.135.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD>
struct lipnet::fast_gradient_descent_t_impl< T, P, VAR, GRAD >::statistics_t

problem specific implementation of statistics_t

See also

lipnet statistics_t [4]

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

- lipnet/include/lipnet/optimizer/fast_gradient_descent.hpp

3.136 lipnet::adam_momentum_t_impl< T, P, VAR, GRAD >::statistics_t Struct Reference

problem specific implementation of statistics_t

```
#include <adam_momentum.hpp>
```

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &archive)

Public Attributes

- series_t< T > **loss**

3.136.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD>
struct lipnet::adam_momentum_t_impl< T, P, VAR, GRAD >::statistics_t

problem specific implementation of statistics_t

See also

lipnet statistics_t [4]

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

- lipnet/include/lipnet/optimizer/adam_momentum.hpp

3.137 lipnet::adam_projected_t_impl< T, P, VAR, GRAD >::statistics_t Struct Reference

problem specific implementation of statistics_t

```
#include <adam_projected.hpp>
```

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &archive)

Public Attributes

- series_t< T > **loss**

3.137.1 Detailed Description

template<typename T, typename P, typename VAR, typename GRAD>
struct lipnet::adam_projected_t_impl< T, P, VAR, GRAD >::statistics_t

problem specific implementation of statistics_t

See also

lipnet statistics_t [4]

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

- lipnet/include/lipnet/optimizer/adam_projected.hpp

3.138 lipnet::optimizer_t< T, P, IMPL, VARS >::stats_type_exists< TT, U > Struct Template Reference

Public Types

- enum { **value** = 0 }
- typedef std::void_type **type**

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

- lipnet/include/lipnet/optimizer.hpp

3.139 lipnet::statistics_helper::stats_type_exists< TT, U > Struct Template Reference

Public Types

- enum { **value** = 0 }
- typedef std::void_type **type**

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

- lipnet/include/lipnet/statistics.hpp

3.140 lipnet::optimizer_t< T, P, IMPL, VARS >::stats_type_exists< TT, typename void_t< typename TT::statistics_t >::type > Struct Template Reference

Public Types

- enum { **value** = 1 }
- typedef TT::statistics_t **type**

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

- lipnet/include/lipnet/optimizer.hpp

3.141 lipnet::statistics_helper::stats_type_exists< TT, typename void_t< typename TT::statistics_t >::type > Struct Template Reference

Public Types

- enum { **value** = 1 }
- typedef TT::statistics_t **type**

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

- lipnet/include/lipnet/statistics.hpp

3.142 lipnet::network_t< T, ATYPE, N >::topology_serialization_t Struct Reference

serialization helper struct

```
#include <network.hpp>
```

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &ar)

3.142.1 Detailed Description

template<typename T, template< typename > typename ATYPE, size_t ... N>
struct lipnet::network_t< T, ATYPE, N >::topology_serialization_t

serialization helper struct

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

- lipnet/include/lipnet/network/network.hpp

3.143 lipnet::data_container_t< T >::tuple_t< saveing > Struct Template Reference

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &ar)

Public Attributes

- view_t< saveing > **x**
- view_t< saveing > **y**

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

- lipnet/include/lipnet/loader/container.hpp

3.144 lipnet::cholesky_topology< T, N >::type Struct Reference

Public Attributes

- cholesky_diagentry< T, N... >::type **D**
- cholesky_subentry< T, N... >::type **L**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.145 lipnet::inverse_topology< T, N >::type Struct Reference

Public Attributes

- inverse_diagentry< T, N... >::type **P**
- inverse_subentry< T, N... >::type **K**

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

- lipnet/include/lipnet/lipschitz/topology.hpp

3.146 lipnet::data_container_t< T >::view_t< saveing > Struct Template Reference

Public Types

- using **refer_t** = decltype(blaze::row(std::declval< typename std::conditional< saveing, const matrix_↩ t, matrix_t >::type >(), std::declval< int >()))
- using **item_t** = typename std::conditional< saveing, const T, T >::type

Public Member Functions

- template<class Archive >
  void **serialize** (Archive &ar)

Public Attributes

- refer_t **value**

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

- lipnet/include/lipnet/loader/container.hpp

3.147 lipnet::optimizer_t< T, P, IMPL, VARS >::void_t< TT > Struct Template Reference

Type holder.

```
#include <optimizer.hpp>
```

Public Types

- typedef void **type**

3.147.1 Detailed Description

template<typename T, typename P, typename IMPL, typename ... VARS>
template<class TT>
struct lipnet::optimizer_t< T, P, IMPL, VARS >::void_t< TT >

Type holder.

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

- lipnet/include/lipnet/optimizer.hpp

3.148 lipnet::statistics_helper::void_t< TT > Struct Template Reference

Public Types

- typedef void **type**

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

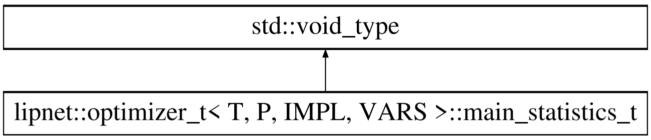
- lipnet/include/lipnet/statistics.hpp

3.149 std::void_type Struct Reference

void type. Holdes nothing.

```
#include <traits.hpp>
```

Inheritance diagram for std::void_type:



3.149.1 Detailed Description

void type. Holdes nothing.

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

- lipnet/include/lipnet/traits.hpp

Bibliography

[1] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011. 5, 19, 20, 21, 22

[2] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. 2019. cite arxiv:1906.04893. 74

[3] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, pages 1–24, 2020. 29

[4] W. Shane Grant and Randolph (2017) Voorhies. cereal - a c++11 library for serialization, 2020. `http://uscilab.github.io/cereal/`. 120, 121, 122, 123, 124

[5] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 5, 15, 16, 17, 19, 23

[6] Georg Hager Klaus Iglberger. Blaze, a high performance c++ math library, 2020. `https://bitbucket.org/blaze-lib/blaze/src/master/`. 38, 39, 47, 48, 53, 54, 101, 103, 113, 115

[7] Keichi Takahashi Pranav. csv2, 2020. `https://github.com/p-ranav/csv2`. 67

Index