

Jstacs reference card

Data handling

Alphabet: A set of symbols

new DiscreteAlphabet(caseInsensitive,alphabet): Create an arbitrary discrete alphabet

new ContinuousAlphabet(min,max): Create a continuous alphabet between min and max

DNAAlphabet.SINGLETON: Singleton instance of a DNA-alphabet

AlphabetContainer: A set of [Alphabets](#) and their assignment to positions

new AlphabetContainer(alphabets): Create an aggregate alphabet out of [Alphabets](#)

DNAAlphabetContainer.SINGLETON: Singleton instance of aggregate DNA-alphabet

Sequence seq: Representing a biological sequence

Sequence.create(alphabets,string): Create a sequence from a string

seq.getLength(): Obtain the length of a sequence

seq.discreteVal(pos): Obtain the discrete value at a position (counting from 0) of a sequence

seq.continuousVal(pos): Obtain the continuous value at a position (counting from 0) of a sequence

DataSet data: A set of sequences using the same [AlphabetContainer](#)

new DataSet(annotation,sequences): Create a data set from sequences

new DNADataset(filename): Create a data set of DNA sequences from a FastA file

data.getNumberOfelements(): Obtain the number of sequences in a data set

data.getElementAt(index): Obtain the sequence at index from a data set

data.getInfixDataSet(start,length): Get a data set containing all infixes of a given length starting at a given position of all sequences in the current data set

Statistical models

StatisticalModel statMod: Interface for all statistical models

TrainableStatisticalModel trainSM: Interface for statistical models that can be trained from a single data set

DifferentiableStatisticalModel diffSM: Interface for statistical models that can be trained using gradient-based methods

TrainableStatisticalModelFactory: Factory for standard implementations of [TrainableStatisticalModels](#)

TrainableStatisticalModelFactory.createPWM(alphabets,length,ess): Create a PWM model of a given length

TrainableStatisticalModelFactory.createInhomogeneousMarkovModel(alphabets,length,ess,order): Create an inhomogeneous Markov model of a given length and order

TrainableStatisticalModelFactory.createHomogeneousMarkovModel(alphabets,ess,order): Create a homogeneous Markov model of a given order

TrainableStatisticalModelFactory.createMixtureModel(hyperpars,models): Create a mixture model from [TrainableStatisticalModels](#)

DifferentiableStatisticalModelFactory: Factory for standard implementations of [DifferentiableStatisticalModels](#)

DifferentiableStatisticalModelFactory.createPWM(alphabets,length,ess): Create a PWM model of a given length

DifferentiableStatisticalModelFactory.createInhomogeneousMarkovModel(alphabets,length,ess,order): Create an inhomogeneous Markov model of a given length and order

DifferentiableStatisticalModelFactory.createHomogeneousMarkovModel(alphabets,ess,order,priorLength): Create a homogeneous Markov model of a given order

DifferentiableStatisticalModelFactory.createMixtureModel(models): Create a mixture model from [DifferentiableStatisticalModels](#)

HMMFactory: Factory for standard implementations of hidden Markov models

statMod.emitDataSet(number,length): Generate a given number of sequences with specified length from this [StatisticalModel](#) using the current parameter values

statMod.getLogProbFor(sequence): Obtain the log probability (likelihood) of a sequence for this [StatisticalModel](#)

trainSM.train(data): Train a [TrainableStatisticalModel](#) from a data set

diffSM.initializeFunctionRandomly(): Initialize the parameters of this [DifferentiableStatisticalModel](#) randomly

diffSM.getLogScoreFor(sequence): Obtain a log score (typically proportional to the log-likelihood) of a sequence for this [DifferentiableStatisticalModel](#)

diffSM.getLogScoreAndPartialDerivation(sequence,indices,partialDers): Compute the partial derivations wrt. all parameters of this [DifferentiableStatisticalModel](#) for the given sequences and store the parameter indexes and corresponding partial derivations in given lists

Classifiers

AbstractClassifier classif: Abstract class of a classifier

new TrainSMBasedClassifier(models): Create a classifier from [TrainableStatisticalModels](#) that is learned by ML or MAP

new MSPClassifier(params,prior,models): Create a classifier from [DifferentiableStatisticalModels](#) that is learned by MCL or MSP

new GenDisMixClassifier(params,prior,learnPrinc,models): Classifier that learns [DifferentiableStatisticalModels](#) using a unified learning principle

ClassifierFactory: Factory for standard classifiers

classif.train(dataSets): Train a classifier from training data sets

classif.classify(sequence): Classify a sequence

classif.evaluate(performanceMeasures,exc,dataSet): Evaluate performance measures for a given classifier on test data sets

AbstractPerformanceMeasure: Abstract class of all performance measures

new NumericalPerformanceMeasureParameterSet(): Create a set of scalar standard performance measures that are applicable to two-class problems (binary classification)

new PerformanceMeasureParameterSet(measures): Create a set of performance measures

PerformanceMeasureParameterSet.createFilledParameters(): Create a set of scalar standard performance measures for binary classification problems that can immediately be used

Utilities

Storable: Interface of objects that can be stored to XML

XMLParser.appendObjectWithTags(buffer,storable,tag): Append storable object to StringBuffer with given tags

XMLParser.extractObjectForTags(buffer,tag): Extract storable object within tags from StringBuffer

Alignment align: Class for optimal pairwise alignments of sequences

new Alignment(type,costs): Create an object for alignments of sequences

align.getAlignment(seq1,seq2): Align two sequences

ArrayHandler.clone(array): Deep clone a multi-dimensional array

ArrayHandler.createArrayOf(template,num): Create an array containing num clones of a template

ToolBox: Compute several statistics from double arrays

Normalisation.getLogSum(doubles): Compute the logarithm of a sum of values given as their logs

Normalisation.sumNormalisation(double): Normalize a given array to probabilities