
DragoNN Documentation

Release 0.1

Johnny Israeli, Michael Wainberg, Avanti Shrikumar

June 15, 2016

CONTENTS

| | | |
|----------|----------------------------|-----------|
| 1 | dragonn package | 3 |
| 1.1 | Subpackages | 3 |
| 1.2 | Submodules | 20 |
| 1.3 | Module contents | 27 |
| 2 | Indices and tables | 29 |
| | Python Module Index | 31 |
| | Index | 33 |

Contents:

DRAGONN PACKAGE

1.1 Subpackages

1.1.1 dragonn.synthetic package

Submodules

dragonn.synthetic.synthetic module

class dragonn.synthetic.synthetic.**AbstractApplySingleMutationFromSet** (*setOfMutations*, *name=None*)

Bases: *dragonn.synthetic.synthetic.AbstractTransformation*

Class for applying a single mutation from a set of mutations; used to transform substrings generated by another method

getClassName ()

getJsonObject ()

selectMutation ()

transform (*stringArr*)

class dragonn.synthetic.synthetic.**AbstractBackgroundGenerator**

Bases: object

Returns the sequence that the embeddings are subsequently inserted into.

generateBackground ()

getJsonObject ()

class dragonn.synthetic.synthetic.**AbstractEmbeddable**

Bases: object

Represents a thing which can be embedded. Note that an Embeddable + a position = an embedding.

canEmbed (*priorEmbeddedThings*, *startPos*)

priorEmbeddedThings: instance of AbstractPriorEmbeddedThings *startPos*: the position you are considering embedding self at returns a boolean indicating whether self can be embedded at *startPos*,

given the things that have already been embedded.

embedInBackgroundStringArr (*priorEmbeddedThings*, *backgroundStringArr*, *startPos*)

Will embed self at startPos in backgroundStringArr, and will update priorEmbeddedThings. priorEmbeddedThings: instance of AbstractPriorEmbeddedThings backgroundStringArr: an array of characters representing the background startPos: the position to embed self at

getDescription ()

class dragonn.synthetic.synthetic.**AbstractEmbeddableGenerator** (*name*)

Bases: *dragonn.synthetic.synthetic.DefaultNameMixin*

Generates an embeddable, usually for embedding in a background sequence.

generateEmbeddable ()

getJSONableObject ()

class dragonn.synthetic.synthetic.**AbstractEmbedder** (*name*)

Bases: *dragonn.synthetic.synthetic.DefaultNameMixin*

class that is used to embed things in a sequence

embed (*backgroundStringArr*, *priorEmbeddedThings*, *additionalInfo=None*)

backgroundStringArr: array of characters representing the background string priorEmbeddedThings: instance of AbstractPriorEmbeddedThings. additionalInfo: instance of AdditionalInfo; allows the embedder to send back info about what it did modifies: backgroundStringArr to include whatever this class has embedded

getJSONableObject ()

class dragonn.synthetic.synthetic.**AbstractLoadedMotifs** (*fileName*, *pseudo-countProb=0.0*, *background=OrderedDict([('A', 0.27), ('C', 0.23), ('G', 0.23), ('T', 0.27)])*)

Bases: object

A class that contains instances of pwm.PWM loaded from a file. The pwms can be accessed by name.

getJSONableObject ()

getPwm (*name*)

returns the pwm.PWM instance with the specified name.

getReadPwmAction (*recordedPwms*)

This is the action that is to be performed on each line of the file when it is read in. recordedPwms is an OrderedDict that stores instances of pwm.PWM

class dragonn.synthetic.synthetic.**AbstractPositionGenerator** (*name*)

Bases: *dragonn.synthetic.synthetic.DefaultNameMixin*

Given the length of the background sequence and the length of the substring you are trying to embed, will return a start position to embed the substring at.

generatePos (*lenBackground*, *lenSubstring*, *additionalInfo=None*)

getJSONableObject ()

class dragonn.synthetic.synthetic.**AbstractPriorEmbeddedThings**

Bases: object

class that is used to keep track of what has already been embedded in a sequence

addEmbedding (*startPos*, *what*)

embeds “what” from startPos to startPos+len(what). Creates an Embedding object

canEmbed (*startPos*, *endPos*)
 returns a boolean indicating whether the region from startPos to endPos is available for embedding

getEmbeddings ()
 returns a collection of Embedding objects

getNumOccupiedPos ()
 returns the number of positions that are filled with some kind of embedding

getTotalPos ()
 returns the total number of positions available to embed things in

class dragonn.synthetic.synthetic.**AbstractQuantityGenerator** (*name*)
 Bases: *dragonn.synthetic.synthetic.DefaultNameMixin*
 class to sample according to a distribution

generateQuantity ()
 returns the sampled value

getJsonObject ()

class dragonn.synthetic.synthetic.**AbstractSequenceSetGenerator**
 Bases: object
 class that is used to return a generator for a collection of generated sequences.

generateSequences ()
 returns a generator of GeneratedSequence objects

getJsonObject ()
 returns an object representing the details of this, which can be converted to json.

class dragonn.synthetic.synthetic.**AbstractSetOfMutations** (*mutationsArr*)
 Bases: object
 Represents a collection of pwm.Mutation objects

getJsonObject ()

getMutationsArr ()

class dragonn.synthetic.synthetic.**AbstractSingleSequenceGenerator** (*namePrefix=None*)
 Bases: object
 When called, generates a single sequence

generateSequence ()
 returns GeneratedSequence object

getJsonObject ()
 returns an object representing the details of this, which can be converted to json.

class dragonn.synthetic.synthetic.**AbstractSubstringGenerator** (*name*)
 Bases: *dragonn.synthetic.synthetic.DefaultNameMixin*
 Generates a substring, usually for embedding in a background sequence.

generateSubstring ()

getJsonObject ()

class dragonn.synthetic.synthetic.**AbstractTransformation** (*name*)
 Bases: *dragonn.synthetic.synthetic.DefaultNameMixin*

takes an array of characters, applies some transformation, returns an array of characters (may be the same (mutated) one or a different one)

getJsonObject ()

transform (*stringArr*)

stringArr is an array of characters. Returns an array of characters that has the transformation applied. May mutate *stringArr*

class `dragonn.synthetic.synthetic.AdditionalInfo`

Bases: `object`

isInTrace (*operatorName*)

updateAdditionalInfo (*operatorName*, *value*)

updateTrace (*operatorName*)

class `dragonn.synthetic.synthetic.AllEmbedders` (*embedders*, *name=None*)

Bases: `dragonn.synthetic.synthetic.AbstractEmbedder`

Wrapper around a list of embedders to make sure all are called Useful in conjunciton with `RandomSubsetOfEmbedders`

getJsonObject ()

class `dragonn.synthetic.synthetic.BernoulliQuantityGenerator` (*prob*, *name=None*)

Bases: `dragonn.synthetic.synthetic.AbstractQuantityGenerator`

Generates 1 or 0 according to a bernoulli distribution

generateQuantity ()

getJsonObject ()

class `dragonn.synthetic.synthetic.BestHitPwm` (*pwm*, *bestHitMode='pwmProb'*, *name=None*)

Bases: `dragonn.synthetic.synthetic.AbstractSubstringGenerator`

always returns the best possible match to the pwm in question when called

generateSubstring ()

getJsonObject ()

class `dragonn.synthetic.synthetic.BestHitPwmFromLoadedMotifs` (*loadedMotifs*, *motifName*, *bestHitMode='pwmProb'*, *name=None*)

Bases: `dragonn.synthetic.synthetic.BestHitPwm`

convenience wrapper class for instantiating parent by pulling the pwm given the name from an `AbstractLoadedMotifs` object (it basically extracts the pwm for you)

getJsonObject ()

class `dragonn.synthetic.synthetic.ChooseMutationAtRandom` (*setOfMutations*, *name=None*)

Bases: `dragonn.synthetic.synthetic.AbstractApplySingleMutationFromSet`

Selects a mutation at random from `self.setOfMutations` to apply; see parent docs.

getClassName ()

selectMutation ()

```

class dragonn.synthetic.synthetic.ChooseValueFromASet (setOfPossibleValues,
                                                    name=None)
    Bases: dragonn.synthetic.synthetic.AbstractQuantityGenerator
    Randomly samples a particular value from a set of values

    generateQuantity ()

    getJsonObject ()

class dragonn.synthetic.synthetic.DefaultNameMixin (name)
    Bases: object

    getDefaultName ()

class dragonn.synthetic.synthetic.EmbedInABackground (backgroundGenerator, embedders,
                                                    namePrefix=None)
    Bases: dragonn.synthetic.synthetic.AbstractSingleSequenceGenerator
    Takes a backgroundGenerator and a series of embedders. Will generate the background and then call each of
    the embedders in succession. Then returns the result.

    generateSequence ()
        generates a background using self.backgroundGenerator, splits it into an array, and passes it to each of
        self.embedders in turn for embedding things. returns an instance of GeneratedSequence

    getJsonObject ()
        see parent

class dragonn.synthetic.synthetic.EmbeddableEmbedder (embeddableGenerator,
                                                    positionGenera-
                                                    tor=<dragonn.synthetic.synthetic.UniformPositionGenerat
                                                    object>, name=None)
    Bases: dragonn.synthetic.synthetic.AbstractEmbedder
    Embeds instances of AbstractEmbeddable within the background sequence, at a position sampled from a distri-
    bution. Only embeds at unoccupied positions

    getJsonObject ()

class dragonn.synthetic.synthetic.Embedding (what, startPos)
    Bases: object
    Represents something that has been embedded in a sequence

    classmethod fromString (string, whatClass=None)

class dragonn.synthetic.synthetic.FixedQuantityGenerator (quantity, name=None)
    Bases: dragonn.synthetic.synthetic.AbstractQuantityGenerator
    returns a fixed number every time generateQuantity is called

    generateQuantity ()

    getJsonObject ()

class dragonn.synthetic.synthetic.FixedSubstringGenerator (fixedSubstring,
                                                    name=None)
    Bases: dragonn.synthetic.synthetic.AbstractSubstringGenerator
    When generateSubstring() is called, always returns the same string. The string also serves as its own description

    generateSubstring ()

    getJsonObject ()

```

class `dragonnn.synthetic.synthetic.GenerateSequenceNTimes` (*singleSetGenerator, N*)
 Bases: `dragonnn.synthetic.synthetic.AbstractSequenceSetGenerator`

If you just want to use a generator of a single sequence and call it N times, use this class.

generateSequences ()
 calls singleSetGenerator N times.

getJsonObject ()

class `dragonnn.synthetic.synthetic.GeneratedSequence` (*seqName, seq, embeddings, additionalInfo*)

Bases: `object`

An object representing a sequence that has been generated.

class `dragonnn.synthetic.synthetic.InsideCentralBp` (*centralBp, name=None*)
 Bases: `dragonnn.synthetic.synthetic.AbstractPositionGenerator`

returns a position within the central region of a background sequence, sampled uniformly at random

getJsonObject ()

class `dragonnn.synthetic.synthetic.IsInTraceLabelGenerator` (*labelNames*)
 Bases: `dragonnn.synthetic.synthetic.LabelGenerator`

class `dragonnn.synthetic.synthetic.LabelGenerator` (*labelNames, labelsFromGeneratedSequenceFunction*)

Bases: `object`

generateLabels (*generatedSequence*)

class `dragonnn.synthetic.synthetic.LoadedEncodeMotifs` (*fileName, pseudocountProb=0.0, background=OrderedDict([('A', 0.27), ('C', 0.23), ('G', 0.23), ('T', 0.27)])*)

Bases: `dragonnn.synthetic.synthetic.AbstractLoadedMotifs`

This class is specifically for reading files in the encode motif format - specifically the motifs.txt file that contains Pouya's motifs

getReadPwmAction (*recordedPwms*)

class `dragonnn.synthetic.synthetic.MinMaxWrapper` (*quantityGenerator, theMin=None, theMax=None, name=None*)
 Bases: `dragonnn.synthetic.synthetic.AbstractQuantityGenerator`

Wrapper that restricts a distribution to only return values between the min and the max. If a value outside the range is returned, resamples until it obtains a value within the range. Warns if it resamples too many times.

generateQuantity ()

getJsonObject ()

class `dragonnn.synthetic.synthetic.OutsideCentralBp` (*centralBp, name=None*)
 Bases: `dragonnn.synthetic.synthetic.AbstractPositionGenerator`

Returns a position OUTSIDE the central region of a background sequence, sampled uniformly at random. Complement of InsideCentralBp.

getJsonObject ()

class `dragonnn.synthetic.synthetic.PairEmbeddable` (*string1, string2, separation, embeddableDescription, nothingInBetween=True*)

Bases: `dragonnn.synthetic.synthetic.AbstractEmbeddable`

Represents a pair of strings that are embedded with some separation. Used for motif grammars. See superclass docs.

canEmbed (*priorEmbeddedThings*, *startPos*)

embedInBackgroundStringArr (*priorEmbeddedThings*, *backgroundStringArr*, *startPos*)

getDescription ()

class `dragonnn.synthetic.synthetic.PairEmbeddableGenerator` (*substringGenerator1*,
substringGenerator2,
separationGenerator,
name=None)

Bases: `dragonnn.synthetic.synthetic.AbstractEmbeddableGenerator`

generateEmbeddable ()

getJsonObject ()

class `dragonnn.synthetic.synthetic.PairEmbeddableGenerator_General` (*embeddableGenerator1*,
*embeddable-
Generator2*, *separa-
tionGenerator*,
name=None)

Bases: `dragonnn.synthetic.synthetic.AbstractEmbeddableGenerator`

generateEmbeddable ()

getJsonObject ()

class `dragonnn.synthetic.synthetic.PairEmbeddable_General` (*embeddable1*, *embed-
dable2*, *separation*, *em-
beddableDescription*,
nothingInBetween=True)

Bases: `dragonnn.synthetic.synthetic.AbstractEmbeddable`

embeds two Embeddable objects with some sep

canEmbed (*priorEmbeddedThings*, *startPos*)

embedInBackgroundStringArr (*priorEmbeddedThings*, *backgroundStringArr*, *startPos*)

getDescription ()

class `dragonnn.synthetic.synthetic.PoissonQuantityGenerator` (*mean*, *name=None*)

Bases: `dragonnn.synthetic.synthetic.AbstractQuantityGenerator`

Generates values according to a poisson distribution

generateQuantity ()

getJsonObject ()

class `dragonnn.synthetic.synthetic.PriorEmbeddedThings_numpyArrayBacked` (*seqLen*)

Bases: `dragonnn.synthetic.synthetic.AbstractPriorEmbeddedThings`

uses a numpy array where positions are set to 1 if they are occupied, to determin which positions are occupied and which are not. See parent for more documentation.

addEmbedding (*startPos*, *what*)
what: instance of Embeddable

canEmbed (*startPos*, *endPos*)

getEmbeddings ()

```

    getNumOccupiedPos ()
    getTotalPos ()
class dragonn.synthetic.synthetic.PwmSampler (pwm, name=None)
    Bases: dragonn.synthetic.synthetic.AbstractSubstringGenerator
    samples from the pwm by calling self.pwm.sampleFromPwm
    generateSubstring ()
    getJsonObject ()
class dragonn.synthetic.synthetic.PwmSamplerFromLoadedMotifs (loadedMotifs, motifName, name=None)
    Bases: dragonn.synthetic.synthetic.PwmSampler
    convenience wrapper class for instantiating parent by pulling the pwm given the name from an AbstractLoaded-
    Motifs object (it basically extracts the pwm for you)
    getJsonObject ()
class dragonn.synthetic.synthetic.RandomSubsetOfEmbedders (quantityGenerator, embedders, name=None)
    Bases: dragonn.synthetic.synthetic.AbstractEmbedder
    Takes a quantity generator that generates a quantity of embedders, and executes that many embedders from a
    supplied set, in sequence
    getJsonObject ()
class dragonn.synthetic.synthetic.RepeatedEmbedder (embedder, quantityGenerator, name=None)
    Bases: dragonn.synthetic.synthetic.AbstractEmbedder
    Wrapper around an embedder to call it multiple times according to sampling from a distribution.
    getJsonObject ()
class dragonn.synthetic.synthetic.RepeatedSubstringBackgroundGenerator (substringGenerator, repetitions)
    Bases: dragonn.synthetic.synthetic.AbstractBackgroundGenerator
    generateBackground ()
    getJsonObject ()
class dragonn.synthetic.synthetic.ReverseComplementWrapper (substringGenerator, reverseComplement-Prob=0.5, name=None)
    Bases: dragonn.synthetic.synthetic.AbstractSubstringGenerator
    Wrapper around a AbstractSubstringGenerator that reverse complements it with the specified probability.
    generateSubstring ()
    getJsonObject ()
class dragonn.synthetic.synthetic.RevertToReference (setOfMutations, name=None)
    Bases: dragonn.synthetic.synthetic.AbstractTransformation
    for a series of mutations, reverts the supplied string to the reference (“unmutated”) string
    getJsonObject ()
    transform (stringArr)

```

```

class dragonn.synthetic.synthetic.SampleFromDiscreteDistributionSubstringGenerator (discreteDistribution)
    Bases: dragonn.synthetic.synthetic.AbstractSubstringGenerator

    generateSubstring ()

    getJsonObject ()

class dragonn.synthetic.synthetic.StringEmbeddable (string, stringDescription='')
    Bases: dragonn.synthetic.synthetic.AbstractEmbeddable

    represents a string (such as a sampling from a pwm) that is to be embedded in a background. See docs for
    superclass.

    canEmbed (priorEmbeddedThings, startPos)

    embedInBackgroundStringArr (priorEmbeddedThings, backgroundStringArr, startPos)

    classmethod fromString (theString)

    getDescription ()

class dragonn.synthetic.synthetic.SubstringEmbeddableGenerator (substringGenerator,
                                                                name=None)
    Bases: dragonn.synthetic.synthetic.AbstractEmbeddableGenerator

    generateEmbeddable ()

    getJsonObject ()

class dragonn.synthetic.synthetic.SubstringEmbedder (substringGenerator,
                                                         positionGenerator=<dragonn.synthetic.synthetic.UniformPositionGenerator
                                                         object>, name=None)
    Bases: dragonn.synthetic.synthetic.EmbeddableEmbedder

    embeds a single generated substring within the background sequence, at a position sampled from a distribution.
    Only embeds at unoccupied positions

class dragonn.synthetic.synthetic.TopNMutationsFromPwmRelativeToBestHit (pwm,
                                                                           N,
                                                                           bestHit-
                                                                           Mode)
    Bases: dragonn.synthetic.synthetic.AbstractSetOfMutations

    See docs for parent; here, the collection of mutations are the top N strongest mutations for a PWM as compared
    to the best match for that pwm.

    getJsonObject ()

class dragonn.synthetic.synthetic.TopNMutationsFromPwmRelativeToBestHit_FromLoadedMotifs (loadedMotifs,
                                                                                          pwm-
                                                                                          Name
                                                                                          N,
                                                                                          bestHit-
                                                                                          Mode)
    Bases: dragonn.synthetic.synthetic.TopNMutationsFromPwmRelativeToBestHit

    Like parent, except extracts the pwm.PWM object from an AbstractLoadedMotifs object, saving you a few lines
    of code.

    getJsonObject ()

```

```
class dragonn.synthetic.synthetic.TransformedSubstringGenerator (substringGenerator,
                                                                transformations,
                                                                transforma-
                                                                tionsDescrip-
                                                                tion='transformations',
                                                                name=None)
```

Bases: *dragonn.synthetic.synthetic.AbstractSubstringGenerator*

Takes a substringGenerator and a set of AbstractTransformation objects, applies the transformations to the generated substring

```
generateSubstring ()
```

```
getJsonObject ()
```

```
class dragonn.synthetic.synthetic.UniformIntegerGenerator (minVal,          maxVal,
                                                            name=None)
```

Bases: *dragonn.synthetic.synthetic.AbstractQuantityGenerator*

Randomly samples an integer from minVal to maxVal, inclusive.

```
generateQuantity ()
```

```
getJsonObject ()
```

```
class dragonn.synthetic.synthetic.UniformPositionGenerator (name=None)
```

Bases: *dragonn.synthetic.synthetic.AbstractPositionGenerator*

samples a start position to embed the substring in uniformly at random; does not return positions that are too close to the end of the background sequence to embed the full substring.

```
getJsonObject ()
```

```
class dragonn.synthetic.synthetic.XOREmbedder (embedder1,    embedder2,    probOfFirst,
                                                name=None)
```

Bases: *dragonn.synthetic.synthetic.AbstractEmbedder*

calls exactly one of the supplied embedders

```
getJsonObject ()
```

```
class dragonn.synthetic.synthetic.ZeroInflater (quantityGenerator, zeroProb, name=None)
```

Bases: *dragonn.synthetic.synthetic.AbstractQuantityGenerator*

Wrapper that inflates the number of zeros returned. Flips a coin; if positive, will return zero - otherwise will sample from the wrapped distribution (which may still return 0)

```
generateQuantity ()
```

```
getJsonObject ()
```

```
class dragonn.synthetic.synthetic.ZeroOrderBackgroundGenerator (seqLength,    dis-
                                                                creteDistribu-
                                                                tion=<dragonn.synthetic.util.DiscreteDistrib-
                                                                object>)
```

Bases: *dragonn.synthetic.synthetic.RepeatedSubstringBackgroundGenerator*

returns a sequence with 40% GC content. Each base is sampled independently.

```
dragonn.synthetic.synthetic.generateString (options)
```

```
dragonn.synthetic.synthetic.generateString_zeroOrderMarkov (length,          dis-
                                                            creteDistribu-
                                                            tion=<dragonn.synthetic.util.DiscreteDistribut-
                                                            object>)
```

discreteDistribution: instance of util.DiscreteDistribution


```

dragonn.synthetic.synthetic.getEmbeddingsFromString (string)
dragonn.synthetic.synthetic.getFileNamePieceFromOptions (options)
dragonn.synthetic.synthetic.getGenerationOption (string)
dragonn.synthetic.synthetic.getParentArgparse ()
dragonn.synthetic.synthetic.printSequences (outputFileName, sequenceSetGenerator,
                                           includeEmbeddings=False, labelGenerator=None, includeFasta=False)
    outputFileName: string sequenceSetGenerator: instance of AbstractSequenceSetGenerator
    Given an output filename, and an instance of AbstractSequenceSetGenerator, will call the sequence set generator and print the generated sequences to the output file. Will also create a file "info_outputFileName.txt" in the same directory as outputFileName that contains all the information about sequenceSetGenerator.
    includeEmbeddings: a boolean indicating whether to print a column that lists the embeddings
    labelGenerator: instance of LabelGenerator
dragonn.synthetic.synthetic.printSequencesTransformationPosNeg (outputFileNamePos, outputFileNameNeg, sequenceSetGenerator, transformation)
    outputFileName: string sequenceSetGenerator: instance of AbstractSequenceSetGenerator
    generatedSequences: the sequences that have been generated by sequenceSetGenerator
    Given an output filename, and an instance of AbstractSequenceSetGenerator, will print the generated sequences to the output file. Will also create a file "info_outputFileName.txt" in the same directory as outputFileName that contains all the information about sequenceSetGenerator.
dragonn.synthetic.synthetic.sampleIndexWithinRegionOfLength (length, lengthOfThingToEmbed)
    uniformly at random samples integers from 0 to length-lengthOfThingToEmbedIn

```

dragonn.synthetic.util module

```

class dragonn.synthetic.util.ArgParseArgument (argumentName, **kwargs)
    Bases: object
    addToParser (parser)
class dragonn.synthetic.util.ArgsAndKwargs (args, kwargs)
    Bases: tuple
    args
        Alias for field number 0
    kwargs
        Alias for field number 1
class dragonn.synthetic.util.ArgumentToAdd (val, argumentName=None, argNameAndValSep='-')
    Bases: object
    Class to append runtime arguments to a string to facilitate auto-generation of output file names.
    argNamePrefix ()
    transform ()

```

```

class dragonn.synthetic.util.ArrArgument (val, argumentName, sep='+', toStringFunc=<type
                                     'str'>)
    Bases: dragonn.synthetic.util.ArgumentToAdd

    transform()

class dragonn.synthetic.util.ArrOfFileNamesArgument (val, argumentName, sep='+')
    Bases: dragonn.synthetic.util.ArrArgument

class dragonn.synthetic.util.BooleanArgument (val,                                     argumentName=None,
                                     argNameAndValSep='-')
    Bases: dragonn.synthetic.util.ArgumentToAdd

    transform()

dragonn.synthetic.util.CROSSC_NORMFUNC
    alias of Enum

class dragonn.synthetic.util.CoreFileNameArgument (val,                                     argumentName=None,
                                     argNameAndValSep='-')
    Bases: dragonn.synthetic.util.ArgumentToAdd

    transform()

class dragonn.synthetic.util.DiscreteDistribution (valToFreq)
    Bases: object

class dragonn.synthetic.util.Entity (id)
    Bases: object

    addAttribute (attributeName, value)
    getAttribute (attributeName)
    hasAttribute (attributeName)

class dragonn.synthetic.util.GetBest
    Bases: object

    getBest ()
    getBestObj ()
    getBestVal ()
    isBetter (val)
    process (theObject, val)

class dragonn.synthetic.util.GetBest_Max
    Bases: dragonn.synthetic.util.GetBest

    isBetter (val)

class dragonn.synthetic.util.GetBest_Min
    Bases: dragonn.synthetic.util.GetBest

    isBetter (val)

class dragonn.synthetic.util.IterableFromDict (theDict, defaultVal, totalLen)
    Bases: object

    next ()

class dragonn.synthetic.util.Options (**kwargs)
    Bases: object

```

```

dragonn.synthetic.util.PERPOS_NORMFUNC
    alias of Enum

class dragonn.synthetic.util.SparseArrFromDict (theDict, defaultVal, totalLen)
    Bases: object

dragonn.synthetic.util.SplitNames
    alias of Enum

class dragonn.synthetic.util.TeeOutputStreams (*streams)
    Bases: object
    for piping to several output streams

    close()
    closed
    flush()
    write(data)
    writable()
    writelines(lines)

class dragonn.synthetic.util.Titled2DMatrix (colNamesPresent=False, rowNamesPresent=False, rows=None, colNames=None, rowNames=None)
    Bases: object
    has a 2D matrix, rowNames and colNames arrays

    addRow(arr, rowName=None)
    normaliseRows()
    printToFile(fileHandle)
    setColNames(colNames)

class dragonn.synthetic.util.TitledArr (title, arr, colNameToIndex=None)
    Bases: object

    getCol(colName)
    setCol(colName, value)

class dragonn.synthetic.util.TitledMapping (titleArr, flagIfInconsistent=False)
    Bases: object

    When each key maps to an array, and each index in the array is associated with a name.

    addKey(key, arr)
    getArrForKey(key)
    getTitledArrForKey(key)
        returns an instance of util.TitledArr which has: getCol(colName) and setCol(colName)
    keyPresenceCheck(key)
        Throws an error if the key is absent
    printToFile(fileHandle, includeRownames=True)

```

```

class dragonn.synthetic.util.TitledMappingIterator (titledMapping)
    Bases: object

    Returns an iterator over TitledArrs for the keys in titledMapping.mapping

    next ()

class dragonn.synthetic.util.VariableWrapper (var)
    For when I want reference-type access to an immutable

dragonn.synthetic.util.absentOrNone (obj, attr)

dragonn.synthetic.util.addArguments (string, args, joiner=' ')
    args is an array of ArgumentToAdd.

dragonn.synthetic.util.addDictionary (toUpdate, toAdd, initVal=0, mergeFunc=<function
    <lambda>>)
    Defaults to addition, technically applicable any time you want to update a dictionary (toUpdate) with the entries
    of another dictionary (toAdd) using a particular operation (eg: adding corresponding keys)

dragonn.synthetic.util.arrToDict (arr)
    Turn an array into a dictionary where each value maps to '1' used for membership testing.

dragonn.synthetic.util.arrayEquals (arr1, arr2)
    compares corresponding entries in arr1 and arr2

dragonn.synthetic.util.assertAllOrNone (obj, attrNames)

dragonn.synthetic.util.assertArrayElementsEqual (arr1, arr2, threshold=0.0)

dragonn.synthetic.util.assertAtLeastOneSet (obj, attrs)

dragonn.synthetic.util.assertAttributesHaveTheSameLengths (attributes, attribute-
    Names)

dragonn.synthetic.util.assertDoesNotHaveAttributes (obj, attributes, explanation)

dragonn.synthetic.util.assertHasAttributes (obj, attributes, explanation)

dragonn.synthetic.util.assertIsType (instance, theClass, instanceVarName)

dragonn.synthetic.util.assertLessThanOrEqual (obj, smallerAttrName, largerAttrName)

dragonn.synthetic.util.assertMutuallyExclusiveAttributes (obj, attrs)

dragonn.synthetic.util.auPRC (trueY, predictedYscores, plotFileName=None)

dragonn.synthetic.util.augmentArgparseKwargsWithDefault (**argParseKwargs)

dragonn.synthetic.util.autovivisect (theDict, getThingToInitialiseWith, *keys)

dragonn.synthetic.util.avgNumpyArrays (numpyArrays)

dragonn.synthetic.util.chainFunctions (*functions)

dragonn.synthetic.util.checkForAttributes (item, attributesToCheckFor, itemName=None)

dragonn.synthetic.util.check_pid (pid)
    Check For the existence of a unix pid.

dragonn.synthetic.util.combineEnums (*enums)

dragonn.synthetic.util.computeConfusionMatrix (actual, predictions, labelOrder-
    ing=None)

dragonn.synthetic.util.computeCooccurrence (matrix)
    matrix: rows (first dim) are examples

dragonn.synthetic.util.computeRunningWindowMax (arr, windowSize)

```

```

dragonn.synthetic.util.computeRunningWindowMaxActivation_2d (arr, smallerArr, windowSize)
dragonn.synthetic.util.computeRunningWindowOneOverMaxActivation_2d (arr, smallerArr, windowSize)
dragonn.synthetic.util.computeRunningWindowOneOverTwoNorm_2d (arr, smallerArr, windowSize)
dragonn.synthetic.util.computeRunningWindowOp (arr, windowSize, op)
dragonn.synthetic.util.computeRunningWindowSum (arr, windowSize)
dragonn.synthetic.util.computeRunningWindowSum_2d (arr, smallerArr, windowSize)
dragonn.synthetic.util.computeRunningWindowTwoNorm_2d (arr, smallerArr, windowSize)
dragonn.synthetic.util.crossCorrelateArraysLengthwise (arr1, arr2, normaliseFunc, smallerPerPosNormFuncs=[], largerPerPosNormFuncs=[], auxLargerForPerPosNorm=None, auxLargerPerPosNormFuncs=[], pad=True)
dragonn.synthetic.util.crossCorrelation_2d (arr, smallerArr, windowSize)
dragonn.synthetic.util.defaultTransformation ()
dragonn.synthetic.util.dict2str (theDict, sep='\\n')
dragonn.synthetic.util.divideByPerPositionRange (arr)
dragonn.synthetic.util.doPCAonFile (theFile)
dragonn.synthetic.util.doesNotWorkForMultithreading_redirectStdout (func, redirectedStdout)
dragonn.synthetic.util.enum (**enums)
dragonn.synthetic.util.enumerate_skipFirst (aList)
dragonn.synthetic.util.executeAsSystemCall (commandToExecute)
dragonn.synthetic.util.executeForAllFilesInDirectory (directory, function, fileFilterFunction=<function <lambda>> >)
dragonn.synthetic.util.floatRange (start, end, step)
    Like range but for floats...
dragonn.synthetic.util.formatDictAsArgsString (theDict, subDictEnclosingChars="")
dragonn.synthetic.util.formattedJsonDump (jsonData)
dragonn.synthetic.util.fracToRainbowColour (frac)
    frac is a number from 0 to 1. Map to a 3-tuple representing a rainbow colour.
    1 -> (0, 1, 0) #green 0.75 -> (1, 0, 1) #yellow 0.5 -> (1, 0, 0) #red 0.25 -> (1, 1, 0) #violet 0 -> (0, 0, 1) #blue
dragonn.synthetic.util.getAllPossibleSubsets (arr)
dragonn.synthetic.util.getBest (arr, getterFunc, takeMax)

```

Will return a tuple of the index and the value of the best as extracted by `getterFunc`

```
dragonnn.synthetic.util.getBestLengthwiseCrossCorrelationOfArrays (arr1, arr2,
                                                                    normalise-
                                                                    Func, small-
                                                                    erPerPos-
                                                                    NormFuncs,
                                                                    largerPer-
                                                                    PosNorm-
                                                                    Funcs)
```

```
dragonnn.synthetic.util.getDateTimeString (datetimeFormat='%y-%m-%d-%H-%M')
```

```
dragonnn.synthetic.util.getErrorTraceback ()
```

```
dragonnn.synthetic.util.getExtremeN (toSort, N, keyFunc)
    Returns the indices
```

```
dragonnn.synthetic.util.getFromEnum (theEnum, enumName, string)
```

```
dragonnn.synthetic.util.getIntervals (minVal, numSteps, **kwargs)
```

```
dragonnn.synthetic.util.getMaxIndex (arr)
```

```
dragonnn.synthetic.util.getNthInterval (minVal, maxVal, numSteps, n, logarithmic, roundTo,
                                         cast)
    logarithmic: boolean indicating if want log numSteps vs linear
    roundTo: can be set to None for no rounding
    cast: can be set to just lambda x: x
```

```
dragonnn.synthetic.util.getRandomString (size)
```

```
dragonnn.synthetic.util.getSingleton (name)
```

```
dragonnn.synthetic.util.getStackTrace ()
```

```
dragonnn.synthetic.util.getTempDir ()
```

```
dragonnn.synthetic.util.imageToSeq (image)
```

```
dragonnn.synthetic.util.intersects (chromStartEnd1, chromStartEnd2)
```

```
dragonnn.synthetic.util.invertIndices (selectedIndices, fullSetOfIndices)
    Returns all indices in fullSet but not in selected.
```

```
dragonnn.synthetic.util.invertPermutation (permutation)
```

```
dragonnn.synthetic.util.isBetter (value, referenceValue, isLargerBetter)
```

```
dragonnn.synthetic.util.isBetterOrEqual (value, referenceValue, isLargerBetter)
```

```
dragonnn.synthetic.util.isNumpy (obj)
```

```
dragonnn.synthetic.util.iter_skipFirst (aList)
```

```
dragonnn.synthetic.util.linecount (filename)
```

```
dragonnn.synthetic.util.makeChromStartEnd (chrom, start, end)
```

```
dragonnn.synthetic.util.multiprocessing_map_printProgress (secondsBetweenUpdates,
                                                            numThreads, func,
                                                            iterable)
```

```
dragonnn.synthetic.util.normaliseByRowsAndColumns (theMatrix)
    The matrix is as a dictionary
```

```
dragonnn.synthetic.util.normaliseEntriesByMeanAndSdev (arr)
```

```
dragonnn.synthetic.util.normaliseEntriesByMeanAndTwoNorm (arr)
```

```

dragonn.synthetic.util.normaliseEntriesBySdev (arr)
dragonn.synthetic.util.normaliseEntriesByTwoNorm (arr)
dragonn.synthetic.util.normaliseEntriesZeroToOne (arr)
dragonn.synthetic.util.normaliseRowsByMeanAndSdev (arr)
dragonn.synthetic.util.normaliseRowsByMeanAndSdev_firstFourSeq (arr)
dragonn.synthetic.util.npArrayIfList (arr)
dragonn.synthetic.util.objectFromArgsAndKwargs (classOfObject, args=[], kwargs={})
dragonn.synthetic.util.objectFromArgsAndKwargsFromYaml (classOfObject, yamlWith-
                                                         ArgsAndKwargs)

dragonn.synthetic.util.overrides (interface_class)
dragonn.synthetic.util.parseJsonFile (fileName)
dragonn.synthetic.util.parseMultipleYamlFiles (pathsToYaml)
dragonn.synthetic.util.parseYamlFile (fileName)
dragonn.synthetic.util.parseYamlFileHandle (fileHandle)
dragonn.synthetic.util.plotPRC (precision, recall, auc, plotFileName)
dragonn.synthetic.util.presentAndNotNone (obj, attr)
dragonn.synthetic.util.presentAndNotNoneOrFalse (obj, attr)
dragonn.synthetic.util.printAttributes (entities, attributesToPrint, outputFile)
dragonn.synthetic.util.printCoordinatesForLabelSubsets (regionIds, labels, labelSet-
                                                         sToFilterFor, outputFilePre-
                                                         fix)
    assumes regionIds of the form chr:start-end labelSetsToFilter as an iterable of iterables of the
    label you want to subset. Will be incorporated into the filename

    outputFile will be outputFilePrefix+"_" + "-" + ".join(str(x) for x in labelsToFilter)

dragonn.synthetic.util.printRegionIds (regionIds, labels, labelFilter, outputFile, idTransfor-
                                                         mation=<function <lambda>>)
dragonn.synthetic.util.randomlySampleFromArr (arr)
dragonn.synthetic.util.readMultilineRawInput (prompt)
dragonn.synthetic.util.redirectStdoutToString (func, logger=None, emailError-
                                                         Func=None)
dragonn.synthetic.util.reverseComplement (sequence)
dragonn.synthetic.util.reverse_enumerate (aList)
dragonn.synthetic.util.roundToNearest (val, nearest)
dragonn.synthetic.util.rowNormalise (matrix)
dragonn.synthetic.util.runningWindowOneOver (func, arr, smallerArr, windowSize)
dragonn.synthetic.util.sampleFromDiscreteDistribution (discreteDistribution)
    Expecting an instance of DiscreteDistribution
dragonn.synthetic.util.sampleFromNumSteps (numSteps, **kwargs)

```

`dragonn.synthetic.util.sampleFromProbsArr` (*arrWithProbs*)
Will return a sampled index

`dragonn.synthetic.util.sampleFromRangeWithStepSize` (*minVal, maxVal, stepSize, cast*)
cast can be just max-min

`dragonn.synthetic.util.sampleNinstancesFromDiscreteDistribution` (*N, discreteDistribution*)

`dragonn.synthetic.util.sampleWithoutReplacement` (*arr, numToSample*)

`dragonn.synthetic.util.sendEmail` (*to, frm, subject, contents*)

`dragonn.synthetic.util.sendEmails` (*tos, frm, subject, contents*)

`dragonn.synthetic.util.seqTo2DImages_fillInArray` (*zerosArray, sequence*)

`dragonn.synthetic.util.seqTo2DImage` (*sequence*)

`dragonn.synthetic.util.setOfSeqsTo2DImages` (*sequences*)

`dragonn.synthetic.util.shuffleArray` (**arrs*)

`dragonn.synthetic.util.sortByLabels` (*arr, labels*)
intended use case: sorting by cluster labels for plotting a heatmap

`dragonn.synthetic.util.splitChromStartEnd` (*chromId*)

`dragonn.synthetic.util.splitIgnoringQuotes` (*string, charToSplitOn=' '*)
will split on `charToSplitOn`, ignoring things that are in quotes

`dragonn.synthetic.util.splitIntegerIntoProportions` (*integer, proportions*)

`dragonn.synthetic.util.submitProcess` (*command*)

`dragonn.synthetic.util.sumAbsDifferences` (*arr1, arr2*)

`dragonn.synthetic.util.sumNumpyArrays` (*numpyArrays*)

`dragonn.synthetic.util.swapIndices` (*arr, idx1, idx2*)

`dragonn.synthetic.util.throwErrorIfUnequalSets` (*given, expected*)

`dragonn.synthetic.util.transformType` (*inp, theType*)

`dragonn.synthetic.util.valToIndexMap` (*arr*)
A map from a value to the index at which it occurs

`dragonn.synthetic.util.yamlToArgsString` (*yamlString, subDictEnclosingChars=""*)

Module contents

1.2 Submodules

1.2.1 `dragonn.hyperparameter_search` module

1.2.2 `dragonn.metrics` module

class `dragonn.metrics.ClassificationResult` (*labels, predictions, task_names=None*)
Bases: object

class `dragonn.metrics.IgnoreNumpyErrors`
Bases: object


```

dragonn.metrics.auPRC (labels, predictions)
dragonn.metrics.auPRG (labels, predictions)
dragonn.metrics.auROC (labels, predictions)
dragonn.metrics.balanced_accuracy (labels, predictions, threshold=0.5)
dragonn.metrics.negative_accuracy (labels, predictions, threshold=0.5)
dragonn.metrics.positive_accuracy (labels, predictions, threshold=0.5)
dragonn.metrics.recall_at_precision_threshold (labels, predictions, precision_threshold)

```

1.2.3 dragonn.models module

```

class dragonn.models.DecisionTree
    Bases: dragonn.models.Model

    predict (X)

    train (X, y, validation_data=None)

class dragonn.models.Model (**hyperparameters)
    Bases: object

    predict (X)

    score (X, y, metric)

    test (X, y)

    train (X, y, validation_data)

class dragonn.models.MotifScoreRNN (input_shape, gru_size=10, tdd_size=4)
    Bases: dragonn.models.Model

    predict (X)

    train (X, y, validation_data)

class dragonn.models.RandomForest
    Bases: dragonn.models.DecisionTree

class dragonn.models.SVC
    Bases: dragonn.models.Model

    predict (X)

    train (X, y, validation_data=None)

class dragonn.models.SequenceDNN (seq_length,      use_deep_CNN=False,      use_RNN=False,
                                     num_tasks=1,      num_filters=15,      conv_width=15,
                                     num_filters_2=15, conv_width_2=15, num_filters_3=15,
                                     conv_width_3=15, pool_width=35, L1=0, dropout=0.0,
                                     GRU_size=35, TDD_size=15, verbose=1)
    Bases: dragonn.models.Model

    Sequence DNN models.

    seq_length [int] length of input sequence.

    use_deep_CNN [bool, optional] uses 3 layered CNN if True, 1 layered CNN if False. Default: False.

    num_tasks [int,] number of tasks. Default: 1.

```

num_filters [int] number of 1st layer convolutional filters. Default: 15.

conv_width [int] width of 1st layer convolutional filters. Default: 15.

pool_width [int] width of max pooling. Default: 35.

num_filters_2 [int] number of 2nd layer convolutional filters. Default: 15.

conv_width_2 [int] width of 2nd layer convolutional filters. Default: 15.

num_filters_3 [int] number of 3rd layer convolutional filters. Default: 15.

conv_width_3 [int] width of 3rd layer convolutional filters. Default: 15.

L1 [float] strength of L1 penalty.

dropout [float] dropout probability in every convolutional layer. Default: 0.

num_tasks [int] Number of prediction tasks or labels. Default: 1.

verbose: int Verbosity level during training. Valid values: 0, 1, 2.

Compiled DNN model.

class LossHistory (*X_train, y_train, validation_data, sequence_DNN*)

Bases: `keras.callbacks.Callback`

on_epoch_end (*epoch, logs={}*)

class SequenceDNN.PrintMetrics (*validation_data, sequence_DNN*)

Bases: `keras.callbacks.Callback`

on_epoch_end (*epoch, logs={}*)

`SequenceDNN.deeplift` (*X, batch_size=200*)

Returns (num_task, num_samples, input_shape) deeplift score array.

`SequenceDNN.get_sequence_filters` ()

Returns list with sequence filter 2darrays.

`SequenceDNN.in_silico_mutagenesis` (*X*)

`SequenceDNN.predict` (*X*)

`SequenceDNN.train` (*X, y, validation_data*)

class `dragonn.models.gkmSVM` (*prefix='./gkmSVM', word_length=11, mismatches=3, C=1*)

Bases: `dragonn.models.Model`

static `encode_sequence_into_fasta_file` (*sequence_iterator, ofname*)

writes sequences into fasta file

model_file

predict (*X*)

train (*X, y, validation_data=None*)

Trains gkm-svm, saves model file.

1.2.4 dragonn.plot module

class `dragonn.plot.Polygon` (*context*)

Bases: `object`

exterior

geom_type

interiors

`dragonn.plot.PolygonPatch` (*polygon*, ***kwargs*)

Constructs a matplotlib patch from a geometric object

The *polygon* may be a Shapely or GeoJSON-like object with or without holes. The *kwargs* are those supported by the `matplotlib.patches.Polygon` class constructor. Returns an instance of `matplotlib.patches.PathPatch`.

Example (using Shapely Point and a matplotlib axes):

```
>>> b = Point(0, 0).buffer(1.0)
>>> patch = PolygonPatch(b, fc='blue', ec='blue', alpha=0.5)
>>> axis.add_patch(patch)
```

`dragonn.plot.PolygonPath` (*polygon*)

Constructs a compound matplotlib path from a Shapely or GeoJSON-like geometric object

`dragonn.plot.add_letter_to_axis` (*ax*, *let*, *x*, *y*, *height*)

Add 'let' with position x,y and height height to matplotlib axis 'ax'.

`dragonn.plot.add_letters_to_axis` (*ax*, *letter_heights*)

Plots letter on user-specified axis.

ax : axis *letter_heights*: Nx4 array

`dragonn.plot.plot_bases` (*letter_heights*, *figsize*=(12, 6), *ylab*='bits')

Plot the N letters with heights taken from the Nx4 matrix *letter_heights*.

letter_heights: Nx4 array *ylab*: y axis label

pyplot figure

`dragonn.plot.plot_motif` (*motif_name*, *figsize*, *ylab*='bits', *information_content*=True)

Plot motifs from encode motifs file

`dragonn.plot.plot_pwm` (*letter_heights*, *figsize*=(12, 6), *ylab*='bits', *information_content*=True)

Plots pwm. Displays information content by default.

`dragonn.plot.standardize_polygons_str` (*data_str*)

Given a POLYGON string, standardize the coordinates to a 1x1 grid.

Input : *data_str* (taken from above) Output: tuple of polygon objects

1.2.5 dragonn.simulations module

`dragonn.simulations.get_distribution` (*GC_fraction*)

`dragonn.simulations.motif_density` (*motif_name*, *seq_length*, *num_seqs*, *min_counts*,
max_counts, *GC_fraction*, *central_bp*=None)

returns sequences with motif density.

`dragonn.simulations.simple_motif_embedding` (*motif_name*, *seq_length*, *num_seqs*,
GC_fraction)

returns sequence array

`dragonn.simulations.simulate_differential_accessibility` (*pos_motif_names*,
neg_motif_names,
seq_length,
min_num_motifs,
max_num_motifs,
num_pos, *num_neg*,
GC_fraction)

Generates data for differential accessibility task.

pos_motif_names [list] List of strings.

neg_motif_names [list] List of strings.

seq_length : int **min_num_motifs** : int **max_num_motifs** : int **num_pos** : int **num_neg** : int **GC_fraction** : float

sequence_arr [1darray] Contains sequence strings.

y [1darray] Contains labels.

```
dragonn.simulations.simulate_heterodimer_grammar(motif1,    motif2,    seq_length,
                                                  min_spacing,    max_spacing,
                                                  num_pos, num_neg, GC_fraction)
```

Simulates two classes of sequences with motif1 and motif2:

- Positive class sequences with motif1 and motif2 positioned min_spacing and max_spacing
- Negative class sequences with independent motif1 and motif2 positioned

anywhere in the sequence, not as a heterodimer grammar

seq_length : int, length of sequence **GC_fraction** : float, GC fraction in background sequence **num_pos** : int, number of positive class sequences **num_neg** : int, number of negative class sequences **motif1** : str, encode motif name **motif2** : str, encode motif name **min_spacing** : int, minimum inter motif spacing **max_spacing** : int, maximum inter motif spacing

sequence_arr [1darray] Array with sequence strings.

y [1darray] Array with positive/negative class labels.

```
dragonn.simulations.simulate_motif_counting(motif_name,    seq_length,    pos_counts,
                                                  neg_counts,    num_pos,    num_neg,
                                                  GC_fraction)
```

Generates data for motif counting task. Parameters ——— **motif_name** : str **seq_length** : int **pos_counts** : list (min_counts, max_counts) for positive set.

neg_counts [list] (min_counts, max_counts) for negative set.

num_pos : int **num_neg** : int **GC_fraction** : float Returns ——— **sequence_arr** : 1darray

Contains sequence strings.

y [1darray] Contains labels.

```
dragonn.simulations.simulate_motif_density_localization(motif_name,
                                                         seq_length,    center_size,
                                                         min_motif_counts,
                                                         max_motif_counts,
                                                         num_pos,    num_neg,
                                                         GC_fraction)
```

Simulates two classes of sequences:

- Positive class sequences with multiple motif instances in center of the sequence.
- Negative class sequences with multiple motif instances anywhere in the sequence.

The number of motif instances is uniformly sampled between minimum and maximum motif counts.

motif_name [str] encode motif name

seq_length [int] length of sequence

center_size [int] length of central part of the sequence where motifs can be positioned

min_motif_counts [int] minimum number of motif instances

max_motif_counts [int] maximum number of motif instances

num_pos [int] number of positive class sequences

num_neg [int] number of negative class sequences

GC_fraction [float] GC fraction in background sequence

sequence_arr [1darray] Contains sequence strings.

y [1darray] Contains labels.

```
dragonn.simulations.simulate_multi_motif_embedding(motif_names,      seq_length,
                                                    min_num_motifs,
                                                    max_num_motifs,      num_seqs,
                                                    GC_fraction)
```

Generates data for multi motif recognition task. Parameters ——— motif_names : list

List of strings.

seq_length : int min_num_motifs : int max_num_motifs : int num_seqs : int GC_fraction : float Returns ———
sequence_arr : 1darray

Contains sequence strings.

y [ndarray] Contains labels for each motif.

```
dragonn.simulations.simulate_single_motif_detection(motif_name,      seq_length,
                                                    num_pos,      num_neg,
                                                    GC_fraction)
```

Simulates two classes of sequences:

- Positive class sequence with a motif embedded anywhere in the sequence
- Negative class sequence without the motif

motif_name [str] encode motif name

seq_length [int] length of sequence

num_pos [int] number of positive class sequences

num_neg [int] number of negative class sequences

GC_fraction [float] GC fraction in background sequence

sequence_arr [1darray] Array with sequence strings.

y [1darray] Array with positive/negative class labels.

1.2.6 dragonn.tutorial_utils module

class dragonn.tutorial_utils.**Data** (*X_train, X_valid, X_test, y_train, y_valid, y_test, motif_names*)

Bases: tuple

X_test

Alias for field number 2

X_train
Alias for field number 0

X_valid
Alias for field number 1

motif_names
Alias for field number 6

y_test
Alias for field number 5

y_train
Alias for field number 3

y_valid
Alias for field number 4

```

dragonn.tutorial_utils.SequenceDNN_learning_curve(dnn)
dragonn.tutorial_utils.get_SequenceDNN(SequenceDNN_parameters)
dragonn.tutorial_utils.get_available_simulations()
dragonn.tutorial_utils.get_simulation_data(simulation_name, simulation_parameters,
                                           test_set_size=4000, validation_set_size=3200)
dragonn.tutorial_utils.get_simulation_function(simulation_name)
dragonn.tutorial_utils.inspect_SequenceDNN()
dragonn.tutorial_utils.interpret_SequenceDNN_distributed(dnn, simulation_data,
                                                         plot_layer_outputs=False)
dragonn.tutorial_utils.interpret_SequenceDNN_integrative(dnn, simulation_data)
dragonn.tutorial_utils.plot_SequenceDNN_layer_outputs(dnn, simulation_data)
dragonn.tutorial_utils.plot_motifs(simulation_data)
dragonn.tutorial_utils.plot_sequence_filters(dnn)
dragonn.tutorial_utils.print_available_simulations()
dragonn.tutorial_utils.print_simulation_info(simulation_name)
dragonn.tutorial_utils.test_SequenceDNN(dnn, simulation_data)
dragonn.tutorial_utils.train_SequenceDNN(dnn, simulation_data)

```

1.2.7 dragonn.utils module

```

dragonn.utils.get_motif_scores(encoded_sequences, motif_names, max_scores=None,
                               return_positions=False, GC_fraction=0.4)

```

Computes pwm log odds.

encoded_sequences : 4darray motif_names : list of strings max_scores : int, optional return_positions : boolean, optional GC_fraction : float, optional

(num_samples, seq_length, num_motifs) complete score array by default. If max_scores, (num_samples, num_motifs*max_scores) max score array. If max_scores and return_positions, (num_samples, 2*num_motifs*max_scores) array with max scores and their positions.

```

dragonn.utils.get_pssm_scores(encoded_sequences, pssm)

```

`dragonn.utils.one_hot_encode` (*sequences*)

`dragonn.utils.reverse_complement` (*encoded_seqs*)

1.3 Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

d

`dragonn`, [27](#)
`dragonn.metrics`, [20](#)
`dragonn.models`, [21](#)
`dragonn.plot`, [22](#)
`dragonn.simulations`, [23](#)
`dragonn.synthetic`, [20](#)
`dragonn.synthetic.synthetic`, [3](#)
`dragonn.synthetic.util`, [13](#)
`dragonn.tutorial_utils`, [25](#)
`dragonn.utils`, [26](#)

A

- `absentOrNone()` (in module `dragonn.synthetic.util`), 16
- `AbstractApplySingleMutationFromSet` (class in `dragonn.synthetic.synthetic`), 3
- `AbstractBackgroundGenerator` (class in `dragonn.synthetic.synthetic`), 3
- `AbstractEmbeddable` (class in `dragonn.synthetic.synthetic`), 3
- `AbstractEmbeddableGenerator` (class in `dragonn.synthetic.synthetic`), 4
- `AbstractEmbedder` (class in `dragonn.synthetic.synthetic`), 4
- `AbstractLoadedMotifs` (class in `dragonn.synthetic.synthetic`), 4
- `AbstractPositionGenerator` (class in `dragonn.synthetic.synthetic`), 4
- `AbstractPriorEmbeddedThings` (class in `dragonn.synthetic.synthetic`), 4
- `AbstractQuantityGenerator` (class in `dragonn.synthetic.synthetic`), 5
- `AbstractSequenceSetGenerator` (class in `dragonn.synthetic.synthetic`), 5
- `AbstractSetOfMutations` (class in `dragonn.synthetic.synthetic`), 5
- `AbstractSingleSequenceGenerator` (class in `dragonn.synthetic.synthetic`), 5
- `AbstractSubstringGenerator` (class in `dragonn.synthetic.synthetic`), 5
- `AbstractTransformation` (class in `dragonn.synthetic.synthetic`), 5
- `add_letter_to_axis()` (in module `dragonn.plot`), 23
- `add_letters_to_axis()` (in module `dragonn.plot`), 23
- `addArguments()` (in module `dragonn.synthetic.util`), 16
- `addAttribute()` (`dragonn.synthetic.util.Entity` method), 14
- `addDictionary()` (in module `dragonn.synthetic.util`), 16
- `addEmbedding()` (`dragonn.synthetic.synthetic.AbstractPriorEmbeddedThings` method), 4
- `addEmbedding()` (`dragonn.synthetic.synthetic.PriorEmbeddedThings_numpyArrayBacked` method), 9
- `AdditionalInfo` (class in `dragonn.synthetic.synthetic`), 6
- `addKey()` (`dragonn.synthetic.util.TitledMapping` method), 15
- `addRow()` (`dragonn.synthetic.util.Titled2DMatrix` method), 15
- `addToParser()` (`dragonn.synthetic.util.ArgParseArgument` method), 13
- `AllEmbedders` (class in `dragonn.synthetic.synthetic`), 6
- `argNamePrefix()` (`dragonn.synthetic.util.ArgumentToAdd` method), 13
- `ArgParseArgument` (class in `dragonn.synthetic.util`), 13
- `args` (`dragonn.synthetic.util.ArgsAndKwargs` attribute), 13
- `ArgsAndKwargs` (class in `dragonn.synthetic.util`), 13
- `ArgumentToAdd` (class in `dragonn.synthetic.util`), 13
- `ArrArgument` (class in `dragonn.synthetic.util`), 13
- `arrayEquals()` (in module `dragonn.synthetic.util`), 16
- `ArrOfFileNamesArgument` (class in `dragonn.synthetic.util`), 14
- `arrToDict()` (in module `dragonn.synthetic.util`), 16
- `assertAllOrNone()` (in module `dragonn.synthetic.util`), 16
- `assertArrayElementsEqual()` (in module `dragonn.synthetic.util`), 16
- `assertAtLeastOneSet()` (in module `dragonn.synthetic.util`), 16
- `assertAttributesHaveTheSameLengths()` (in module `dragonn.synthetic.util`), 16
- `assertDoesNotHaveAttributes()` (in module `dragonn.synthetic.util`), 16
- `assertHasAttributes()` (in module `dragonn.synthetic.util`), 16
- `assertIsType()` (in module `dragonn.synthetic.util`), 16
- `assertLessThanOrEqual()` (in module `dragonn.synthetic.util`), 16
- `assertMutuallyExclusiveAttributes()` (in module `dragonn.synthetic.util`), 16
- `augmentArgparseKwargsHelpWithDefault()` (in module `dragonn.synthetic.util`), 16
- `auPRC()` (in module `dragonn.metrics`), 21
- `auPRC()` (in module `dragonn.synthetic.util`), 16
- `auPRG()` (in module `dragonn.metrics`), 21
- `auROC()` (in module `dragonn.metrics`), 21
- `autovivisection()` (in module `dragonn.synthetic.util`), 16

avgNumpyArrays() (in module dragonn.synthetic.util), 16

B

balanced_accuracy() (in module dragonn.metrics), 21

BernoulliQuantityGenerator (class in dragonn.synthetic.synthetic), 6

BestHitPwm (class in dragonn.synthetic.synthetic), 6

BestHitPwmFromLoadedMotifs (class in dragonn.synthetic.synthetic), 6

BooleanArgument (class in dragonn.synthetic.util), 14

C

canEmbed() (dragonn.synthetic.synthetic.AbstractEmbeddable method), 3

canEmbed() (dragonn.synthetic.synthetic.AbstractPriorEmbeddable method), 4

canEmbed() (dragonn.synthetic.synthetic.PairEmbeddable method), 9

canEmbed() (dragonn.synthetic.synthetic.PairEmbeddable_General method), 9

canEmbed() (dragonn.synthetic.synthetic.PriorEmbeddableThings_humpyArrayBacked method), 9

canEmbed() (dragonn.synthetic.synthetic.StringEmbeddable method), 11

chainFunctions() (in module dragonn.synthetic.util), 16

check_pid() (in module dragonn.synthetic.util), 16

checkForAttributes() (in module dragonn.synthetic.util), 16

ChooseMutationAtRandom (class in dragonn.synthetic.synthetic), 6

ChooseValueFromASet (class in dragonn.synthetic.synthetic), 6

ClassificationResult (class in dragonn.metrics), 20

close() (dragonn.synthetic.util.TeeOutputStreams method), 15

closed (dragonn.synthetic.util.TeeOutputStreams attribute), 15

combineEnums() (in module dragonn.synthetic.util), 16

computeConfusionMatrix() (in module dragonn.synthetic.util), 16

computeCooccurrence() (in module dragonn.synthetic.util), 16

computeRunningWindowMax() (in module dragonn.synthetic.util), 16

computeRunningWindowMaxActivation_2d() (in module dragonn.synthetic.util), 17

computeRunningWindowOneOverMaxActivation_2d() (in module dragonn.synthetic.util), 17

computeRunningWindowOneOverTwoNorm_2d() (in module dragonn.synthetic.util), 17

computeRunningWindowOp() (in module dragonn.synthetic.util), 17

computeRunningWindowSum() (in module dragonn.synthetic.util), 17

computeRunningWindowSum_2d() (in module dragonn.synthetic.util), 17

computeRunningWindowTwoNorm_2d() (in module dragonn.synthetic.util), 17

CoreFileNameArgument (class in dragonn.synthetic.util), 14

CROSSC_NORMFUNC (in module dragonn.synthetic.util), 14

crossCorrelateArraysLengthwise() (in module dragonn.synthetic.util), 17

crossCorrelation_2d() (in module dragonn.synthetic.util), 17

D

DefaultNameMixin

Data (class in dragonn.tutorial_utils), 25

DecisionTree (class in dragonn.models), 21

deeplift() (dragonn.models.SequenceDNN method), 22

DefaultNameMixin (class in dragonn.synthetic.synthetic), 7

defaultTransformation() (in module dragonn.synthetic.util), 17

dict2str() (in module dragonn.synthetic.util), 17

DiscreteDistribution (class in dragonn.synthetic.util), 14

divideByPerPositionRange() (in module dragonn.synthetic.util), 17

doesNotWorkForMultithreading_redirectStdout() (in module dragonn.synthetic.util), 17

doPCAonFile() (in module dragonn.synthetic.util), 17

dragonn (module), 27

dragonn.metrics (module), 20

dragonn.models (module), 21

dragonn.plot (module), 22

dragonn.simulations (module), 23

dragonn.synthetic (module), 20

dragonn.synthetic.synthetic (module), 3

dragonn.synthetic.util (module), 13

dragonn.tutorial_utils (module), 25

dragonn.utils (module), 26

E

embed() (dragonn.synthetic.synthetic.AbstractEmbedder method), 4

EmbeddableEmbedder (class in dragonn.synthetic.synthetic), 7

Embedding (class in dragonn.synthetic.synthetic), 7

EmbedInABackground (class in dragonn.synthetic.synthetic), 7

embedInBackgroundStringArr() (dragonn.synthetic.synthetic.AbstractEmbeddable method), 3

embedInBackgroundStringArr() (dragonn.synthetic.synthetic.PairEmbeddable method), 3

method), 9

embedInBackgroundStringArr() (dragonn.synthetic.synthetic.PairEmbeddable_General method), 9

embedInBackgroundStringArr() (dragonn.synthetic.synthetic.StringEmbeddable method), 11

encode_sequence_into_fasta_file() (dragonn.models.gkmSVM static method), 22

Entity (class in dragonn.synthetic.util), 14

enum() (in module dragonn.synthetic.util), 17

enumerate_skipFirst() (in module dragonn.synthetic.util), 17

executeAsSystemCall() (in module dragonn.synthetic.util), 17

executeForAllFilesInDirectory() (in module dragonn.synthetic.util), 17

exterior (dragonn.plot.Polygon attribute), 22

F

FixedQuantityGenerator (class in dragonn.synthetic.synthetic), 7

FixedSubstringGenerator (class in dragonn.synthetic.synthetic), 7

floatRange() (in module dragonn.synthetic.util), 17

flush() (dragonn.synthetic.util.TeeOutputStreams method), 15

formatDictAsArgsString() (in module dragonn.synthetic.util), 17

formattedJsonDump() (in module dragonn.synthetic.util), 17

fracToRainbowColour() (in module dragonn.synthetic.util), 17

fromString() (dragonn.synthetic.synthetic.Embedding class method), 7

fromString() (dragonn.synthetic.synthetic.StringEmbeddable class method), 11

G

generateBackground() (dragonn.synthetic.synthetic.AbstractBackgroundGenerator method), 3

generateBackground() (dragonn.synthetic.synthetic.RepeatedSubstringBackgroundGenerator method), 10

GeneratedSequence (class in dragonn.synthetic.synthetic), 8

generateEmbeddable() (dragonn.synthetic.synthetic.AbstractEmbeddableGenerator method), 4

generateEmbeddable() (dragonn.synthetic.synthetic.PairEmbeddableGenerator method), 9

generateEmbeddable() (dragonn.synthetic.synthetic.PairEmbeddableGenerator_General method), 9

generateEmbeddable() (dragonn.synthetic.synthetic.SubstringEmbeddableGenerator method), 11

generateLabels() (dragonn.synthetic.synthetic.LabelGenerator method), 8

generatePos() (dragonn.synthetic.synthetic.AbstractPositionGenerator method), 4

generateQuantity() (dragonn.synthetic.synthetic.AbstractQuantityGenerator method), 5

generateQuantity() (dragonn.synthetic.synthetic.BernoulliQuantityGenerator method), 6

generateQuantity() (dragonn.synthetic.synthetic.ChooseValueFromASet method), 7

generateQuantity() (dragonn.synthetic.synthetic.FixedQuantityGenerator method), 7

generateQuantity() (dragonn.synthetic.synthetic.MinMaxWrapper method), 8

generateQuantity() (dragonn.synthetic.synthetic.PoissonQuantityGenerator method), 9

generateQuantity() (dragonn.synthetic.synthetic.UniformIntegerGenerator method), 12

generateQuantity() (dragonn.synthetic.synthetic.ZeroInflater method), 12

generateSequence() (dragonn.synthetic.synthetic.AbstractSingleSequenceGenerator method), 5

generateSequence() (dragonn.synthetic.synthetic.EmbedInABackground method), 7

GenerateSequenceNTimes (class in dragonn.synthetic.synthetic), 7

generateSequences() (dragonn.synthetic.synthetic.AbstractSequenceSetGenerator method), 5

generateSequences() (dragonn.synthetic.synthetic.GenerateSequenceNTimes method), 8

generateString() (in module dragonn.synthetic.synthetic), 12

generateString_zeroOrderMarkov() (in module dragonn.synthetic.synthetic), 12

generateSubstring() (dragonn.synthetic.synthetic.PairEmbeddableGenerator method), 9

[onn.synthetic.synthetic.AbstractSubstringGenerator](#) 18
[onn.synthetic.synthetic.AbstractSubstringGenerator](#) method), 5
[generateSubstring\(\)](#) (drag-
[onn.synthetic.synthetic.BestHitPwm](#) method),
6
[generateSubstring\(\)](#) (drag-
[onn.synthetic.synthetic.FixedSubstringGenerator](#)
method), 7
[generateSubstring\(\)](#) (drag-
[onn.synthetic.synthetic.PwmSampler](#) method),
10
[generateSubstring\(\)](#) (drag-
[onn.synthetic.synthetic.ReverseComplementWrapper](#)
method), 10
[generateSubstring\(\)](#) (drag-
[onn.synthetic.synthetic.SampleFromDiscreteDistribution](#)
method), 11
[generateSubstring\(\)](#) (drag-
[onn.synthetic.synthetic.TransformedSubstringGenerator](#)
method), 12
[geom_type](#) ([dragonn.plot.Polygon](#) attribute), 22
[get_available_simulations\(\)](#) (in module drag-
[onn.tutorial_utils](#)), 26
[get_distribution\(\)](#) (in module [dragonn.simulations](#)), 23
[get_motif_scores\(\)](#) (in module [dragonn.utils](#)), 26
[get_pssm_scores\(\)](#) (in module [dragonn.utils](#)), 26
[get_sequence_filters\(\)](#) ([dragonn.models.SequenceDNN](#)
method), 22
[get_SequenceDNN\(\)](#) (in module [dragonn.tutorial_utils](#)),
26
[get_simulation_data\(\)](#) (in module [dragonn.tutorial_utils](#)),
26
[get_simulation_function\(\)](#) (in module drag-
[onn.tutorial_utils](#)), 26
[getAllPossibleSubsets\(\)](#) (in module drag-
[onn.synthetic.util](#)), 17
[getArrForKey\(\)](#) ([dragonn.synthetic.util.TitledMapping](#)
method), 15
[getAttribute\(\)](#) ([dragonn.synthetic.util.Entity](#) method), 14
[GetBest](#) (class in [dragonn.synthetic.util](#)), 14
[getBest\(\)](#) ([dragonn.synthetic.util.GetBest](#) method), 14
[getBest\(\)](#) (in module [dragonn.synthetic.util](#)), 17
[GetBest_Max](#) (class in [dragonn.synthetic.util](#)), 14
[GetBest_Min](#) (class in [dragonn.synthetic.util](#)), 14
[getBestLengthwiseCrossCorrelationOfArrays\(\)](#) (in mod-
ule [dragonn.synthetic.util](#)), 18
[getBestObj\(\)](#) ([dragonn.synthetic.util.GetBest](#) method), 14
[getBestVal\(\)](#) ([dragonn.synthetic.util.GetBest](#) method), 14
[getClassname\(\)](#) ([dragonn.synthetic.synthetic.AbstractApplySingleMutationFromSet](#)
method), 3
[getClassname\(\)](#) ([dragonn.synthetic.synthetic.ChooseMutationFromSet](#)
method), 6
[getCol\(\)](#) ([dragonn.synthetic.util.TitledArr](#) method), 15
[getDateTimeString\(\)](#) (in module [dragonn.synthetic.util](#)),

[getDefaultName\(\)](#) (drag-
[onn.synthetic.synthetic.DefaultNameMixin](#)
method), 7
[getDescription\(\)](#) ([dragonn.synthetic.synthetic.AbstractEmbeddable](#)
method), 4
[getDescription\(\)](#) ([dragonn.synthetic.synthetic.PairEmbeddable](#)
method), 9
[getDescription\(\)](#) ([dragonn.synthetic.synthetic.PairEmbeddable_General](#)
method), 9
[getDescription\(\)](#) ([dragonn.synthetic.synthetic.StringEmbeddable](#)
method), 11
[getEmbeddings\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractPriorEmbeddedThings](#)
method), 5
[getEmbeddingsFromGenerator\(\)](#) (drag-
[onn.synthetic.synthetic.PriorEmbeddedThings_numpyArrayBack](#)
method), 9
[getEmbeddingsFromString\(\)](#) (in module drag-
[onn.synthetic.synthetic](#)), 12
[getErrorTraceback\(\)](#) (in module [dragonn.synthetic.util](#)),
18
[getExtremeN\(\)](#) (in module [dragonn.synthetic.util](#)), 18
[getFileNamePieceFromOptions\(\)](#) (in module drag-
[onn.synthetic.synthetic](#)), 13
[getFromEnum\(\)](#) (in module [dragonn.synthetic.util](#)), 18
[getGenerationOption\(\)](#) (in module drag-
[onn.synthetic.synthetic](#)), 13
[getIntervals\(\)](#) (in module [dragonn.synthetic.util](#)), 18
[getJsonObject\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractApplySingleMutationFromSet](#)
method), 3
[getJsonObject\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractBackgroundGenerator](#)
method), 3
[getJsonObject\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractEmbeddableGenerator](#)
method), 4
[getJsonObject\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractEmbedder](#)
method), 4
[getJsonObject\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractLoadedMotifs](#)
method), 4
[getJsonObject\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractPositionGenerator](#)
method), 4
[getJsonObject\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractQuantityGenerator](#)
method), 5
[getJsonObject\(\)](#) (drag-
[onn.synthetic.synthetic.AbstractSequenceSetGenerator](#)
method), 5
[getJsonObject\(\)](#) (drag-

| | |
|---|--|
| onn.synthetic.synthetic.AbstractSetOfMutations method), 5 | onn.synthetic.synthetic.PairEmbeddableGenerator_General method), 9 |
| getJsonableObject() (drag- onn.synthetic.synthetic.AbstractSingleSequenceGenerator method), 5 | getJsonableObject() (drag- onn.synthetic.synthetic.PoissonQuantityGenerator method), 9 |
| getJsonableObject() (drag- onn.synthetic.synthetic.AbstractSubstringGenerator method), 5 | getJsonableObject() (drag- onn.synthetic.synthetic.PwmSampler method), 10 |
| getJsonableObject() (drag- onn.synthetic.synthetic.AbstractTransformation method), 6 | getJsonableObject() (drag- onn.synthetic.synthetic.PwmSamplerFromLoadedMotifs method), 10 |
| getJsonableObject() (drag- onn.synthetic.synthetic.AllEmbedders method), 6 | getJsonableObject() (drag- onn.synthetic.synthetic.RandomSubsetOfEmbedders method), 10 |
| getJsonableObject() (drag- onn.synthetic.synthetic.BernoulliQuantityGenerator method), 6 | getJsonableObject() (drag- onn.synthetic.synthetic.RepeatedEmbedder method), 10 |
| getJsonableObject() (drag- onn.synthetic.synthetic.BestHitPwm method), 6 | getJsonableObject() (drag- onn.synthetic.synthetic.RepeatedSubstringBackgroundGenerator method), 10 |
| getJsonableObject() (drag- onn.synthetic.synthetic.BestHitPwmFromLoadedMotifs method), 6 | getJsonableObject() (drag- onn.synthetic.synthetic.ReverseComplementWrapper method), 10 |
| getJsonableObject() (drag- onn.synthetic.synthetic.ChooseValueFromASet method), 7 | getJsonableObject() (drag- onn.synthetic.synthetic.RevertToReference method), 10 |
| getJsonableObject() (drag- onn.synthetic.synthetic.EmbeddableEmbedder method), 7 | getJsonableObject() (drag- onn.synthetic.synthetic.SampleFromDiscreteDistributionSubstring method), 11 |
| getJsonableObject() (drag- onn.synthetic.synthetic.EmbedInABackground method), 7 | getJsonableObject() (drag- onn.synthetic.synthetic.SubstringEmbeddableGenerator method), 11 |
| getJsonableObject() (drag- onn.synthetic.synthetic.FixedQuantityGenerator method), 7 | getJsonableObject() (drag- onn.synthetic.synthetic.TopNMutationsFromPwmRelativeToBest method), 11 |
| getJsonableObject() (drag- onn.synthetic.synthetic.FixedSubstringGenerator method), 7 | getJsonableObject() (drag- onn.synthetic.synthetic.TopNMutationsFromPwmRelativeToBest method), 11 |
| getJsonableObject() (drag- onn.synthetic.synthetic.GenerateSequenceNTimes method), 8 | getJsonableObject() (drag- onn.synthetic.synthetic.TransformedSubstringGenerator method), 12 |
| getJsonableObject() (drag- onn.synthetic.synthetic.InsideCentralBp method), 8 | getJsonableObject() (drag- onn.synthetic.synthetic.UniformIntegerGenerator method), 12 |
| getJsonableObject() (drag- onn.synthetic.synthetic.MinMaxWrapper method), 8 | getJsonableObject() (drag- onn.synthetic.synthetic.UniformPositionGenerator method), 12 |
| getJsonableObject() (drag- onn.synthetic.synthetic.OutsideCentralBp method), 8 | getJsonableObject() (drag- onn.synthetic.synthetic.XOREmbedder method), 12 |
| getJsonableObject() (drag- onn.synthetic.synthetic.PairEmbeddableGenerator method), 9 | getJsonableObject() (drag- onn.synthetic.synthetic.ZeroInflater method), 12 |
| getJsonableObject() (drag- | getMaxIndex() (in module dragonn.synthetic.util), 18 |

[getMutationsArr\(\)](#) (drag-onn.synthetic.synthetic.AbstractSetOfMutations method), 5
[getNthInterval\(\)](#) (in module dragonn.synthetic.util), 18
[getNumOccupiedPos\(\)](#) (drag-onn.synthetic.synthetic.AbstractPriorEmbeddedThings_numpyArrayBacked method), 5
[getNumOccupiedPos\(\)](#) (drag-onn.synthetic.synthetic.PriorEmbeddedThings_numpyArrayBacked method), 9
[getParentArgparse\(\)](#) (in module drag-onn.synthetic.synthetic), 13
[getPwm\(\)](#) (dragonn.synthetic.synthetic.AbstractLoadedMotifs method), 4
[getRandomString\(\)](#) (in module dragonn.synthetic.util), 18
[getReadPwmAction\(\)](#) (drag-onn.synthetic.synthetic.AbstractLoadedMotifs method), 4
[getReadPwmAction\(\)](#) (drag-onn.synthetic.synthetic.LoadedEncodeMotifs method), 8
[getSingleton\(\)](#) (in module dragonn.synthetic.util), 18
[getStackTrace\(\)](#) (in module dragonn.synthetic.util), 18
[getTempDir\(\)](#) (in module dragonn.synthetic.util), 18
[getTitleArrForKey\(\)](#) (drag-onn.synthetic.util.TitledMapping method), 15
[getTotalPos\(\)](#) (dragonn.synthetic.synthetic.AbstractPriorEmbeddedThings method), 5
[getTotalPos\(\)](#) (dragonn.synthetic.synthetic.PriorEmbeddedThings_numpyArrayBacked method), 10
[gkmSVM](#) (class in dragonn.models), 22

H

[hasAttribute\(\)](#) (dragonn.synthetic.util.Entity method), 14

I

[IgnoreNumpyErrors](#) (class in dragonn.metrics), 20
[imageToSeq\(\)](#) (in module dragonn.synthetic.util), 18
[in_silico_mutagenesis\(\)](#) (dragonn.models.SequenceDNN method), 22
[InsideCentralBp](#) (class in dragonn.synthetic.synthetic), 8
[inspect_SequenceDNN\(\)](#) (in module dragonn.tutorial_utils), 26
[interiors](#) (dragonn.plot.Polygon attribute), 23
[interpret_SequenceDNN_distributed\(\)](#) (in module dragonn.tutorial_utils), 26
[interpret_SequenceDNN_integrative\(\)](#) (in module dragonn.tutorial_utils), 26
[intersects\(\)](#) (in module dragonn.synthetic.util), 18
[invertIndices\(\)](#) (in module dragonn.synthetic.util), 18
[invertPermutation\(\)](#) (in module dragonn.synthetic.util), 18
[isBetter\(\)](#) (dragonn.synthetic.util.GetBest method), 14

[isBetter\(\)](#) (dragonn.synthetic.util.GetBest_Max method), 14
[isBetter\(\)](#) (dragonn.synthetic.util.GetBest_Min method), 14
[isBetter\(\)](#) (in module dragonn.synthetic.util), 18
[isBetterOrEqual\(\)](#) (in module dragonn.synthetic.util), 18
[isInTrace\(\)](#) (dragonn.synthetic.synthetic.AdditionalInfo method), 6
[LabelGenerator](#) (class in dragonn.synthetic.synthetic), 8
[isNumpy\(\)](#) (in module dragonn.synthetic.util), 18
[iter_skipFirst\(\)](#) (in module dragonn.synthetic.util), 18
[IterableFromDict](#) (class in dragonn.synthetic.util), 14

K

[keyPresenceCheck\(\)](#) (drag-onn.synthetic.util.TitledMapping method), 15
[kwargs](#) (dragonn.synthetic.util.ArgsAndKwargs attribute), 13

L

[LabelGenerator](#) (class in dragonn.synthetic.synthetic), 8
[linecount\(\)](#) (in module dragonn.synthetic.util), 18
[LoadedEncodeMotifs](#) (class in dragonn.synthetic.synthetic), 8

M

[makeChromStartEnd\(\)](#) (in module dragonn.synthetic.util), 18
[MinMaxWrapper](#) (class in dragonn.synthetic.synthetic), 8
[Model](#) (class in dragonn.models), 21
[model_file](#) (dragonn.models.gkmSVM attribute), 22
[motif_density\(\)](#) (in module dragonn.simulations), 23
[motif_names](#) (dragonn.tutorial_utils.Data attribute), 26
[MotifScoreRNN](#) (class in dragonn.models), 21
[multiprocessing_map_printProgress\(\)](#) (in module dragonn.synthetic.util), 18

N

[negative_accuracy\(\)](#) (in module dragonn.metrics), 21
[next\(\)](#) (dragonn.synthetic.util.IterableFromDict method), 14
[next\(\)](#) (dragonn.synthetic.util.TitledMappingIterator method), 16
[normaliseByRowsAndColumns\(\)](#) (in module dragonn.synthetic.util), 18
[normaliseEntriesByMeanAndSdev\(\)](#) (in module dragonn.synthetic.util), 18
[normaliseEntriesByMeanAndTwoNorm\(\)](#) (in module dragonn.synthetic.util), 18
[normaliseEntriesBySdev\(\)](#) (in module dragonn.synthetic.util), 18

- ul style="list-style-type: none; padding-left: 0;">
- normaliseEntriesByTwoNorm() (in module dragonn.synthetic.util), 19
- normaliseEntriesZeroToOne() (in module dragonn.synthetic.util), 19
- normaliseRows() (dragonn.synthetic.util.Titled2DMatrix method), 15
- normaliseRowsByMeanAndSdev() (in module dragonn.synthetic.util), 19
- normaliseRowsByMeanAndSdev_firstFourSeq() (in module dragonn.synthetic.util), 19
- npArrayIfList() (in module dragonn.synthetic.util), 19
- ## O
- objectFromArgsAndKwargs() (in module dragonn.synthetic.util), 19
 - objectFromArgsAndKwargsFromYaml() (in module dragonn.synthetic.util), 19
 - on_epoch_end() (dragonn.models.SequenceDNN.LossHistory method), 22
 - on_epoch_end() (dragonn.models.SequenceDNN.PrintMetrics method), 22
 - one_hot_encode() (in module dragonn.utils), 26
 - Options (class in dragonn.synthetic.util), 14
 - OutsideCentralBp (class in dragonn.synthetic.synthetic), 8
 - overrides() (in module dragonn.synthetic.util), 19
- ## P
- PairEmbeddable (class in dragonn.synthetic.synthetic), 8
 - PairEmbeddable_General (class in dragonn.synthetic.synthetic), 9
 - PairEmbeddableGenerator (class in dragonn.synthetic.synthetic), 9
 - PairEmbeddableGenerator_General (class in dragonn.synthetic.synthetic), 9
 - parseJsonFile() (in module dragonn.synthetic.util), 19
 - parseMultipleYamlFiles() (in module dragonn.synthetic.util), 19
 - parseYamlFile() (in module dragonn.synthetic.util), 19
 - parseYamlFileHandle() (in module dragonn.synthetic.util), 19
 - PERPOS_NORMFUNC (in module dragonn.synthetic.util), 14
 - plot_bases() (in module dragonn.plot), 23
 - plot_motif() (in module dragonn.plot), 23
 - plot_motifs() (in module dragonn.tutorial_utils), 26
 - plot_pwm() (in module dragonn.plot), 23
 - plot_sequence_filters() (in module dragonn.tutorial_utils), 26
 - plot_SequenceDNN_layer_outputs() (in module dragonn.tutorial_utils), 26
 - plotPRC() (in module dragonn.synthetic.util), 19
 - PoissonQuantityGenerator (class in dragonn.synthetic.synthetic), 9
 - Polygon (class in dragonn.plot), 22
 - PolygonPatch() (in module dragonn.plot), 23
 - PolygonPath() (in module dragonn.plot), 23
 - positive_accuracy() (in module dragonn.metrics), 21
 - predict() (dragonn.models.DecisionTree method), 21
 - predict() (dragonn.models.gkmSVM method), 22
 - predict() (dragonn.models.Model method), 21
 - predict() (dragonn.models.MotifScoreRNN method), 21
 - predict() (dragonn.models.SequenceDNN method), 22
 - predict() (dragonn.models.SVC method), 21
 - presentAndNotNone() (in module dragonn.synthetic.util), 19
 - presentAndNotNoneOrFalse() (in module dragonn.synthetic.util), 19
 - print_available_simulations() (in module dragonn.tutorial_utils), 26
 - print_simulation_info() (in module dragonn.tutorial_utils), 26
 - printAttributes() (in module dragonn.synthetic.util), 19
 - printCoordinatesForLabelSubsets() (in module dragonn.synthetic.util), 19
 - printRegionIds() (in module dragonn.synthetic.util), 19
 - printSequences() (in module dragonn.synthetic.synthetic), 13
 - printSequencesTransformationPosNeg() (in module dragonn.synthetic.synthetic), 13
 - printToFile() (dragonn.synthetic.util.Titled2DMatrix method), 15
 - printToFile() (dragonn.synthetic.util.TitledMapping method), 15
 - PriorEmbeddedThings_numpyArrayBacked (class in dragonn.synthetic.synthetic), 9
 - process() (dragonn.synthetic.util.GetBest method), 14
 - PwmSampler (class in dragonn.synthetic.synthetic), 10
 - PwmSamplerFromLoadedMotifs (class in dragonn.synthetic.synthetic), 10
- ## R
- RandomForest (class in dragonn.models), 21
 - randomlySampleFromArr() (in module dragonn.synthetic.util), 19
 - RandomSubsetOfEmbedders (class in dragonn.synthetic.synthetic), 10
 - readMultilineRawInput() (in module dragonn.synthetic.util), 19
 - recall_at_precision_threshold() (in module dragonn.metrics), 21
 - redirectStdoutToString() (in module dragonn.synthetic.util), 19
 - RepeatedEmbedder (class in dragonn.synthetic.synthetic), 10
 - RepeatedSubstringBackgroundGenerator (class in dragonn.synthetic.synthetic), 10
 - reverse_complement() (in module dragonn.utils), 27

reverse_enumerate() (in module dragonn.synthetic.util), 19
reverseComplement() (in module dragonn.synthetic.util), 19
ReverseComplementWrapper (class in dragonn.synthetic.synthetic), 10
RevertToReference (class in dragonn.synthetic.synthetic), 10
roundToNearest() (in module dragonn.synthetic.util), 19
rowNormalise() (in module dragonn.synthetic.util), 19
runningWindowOneOver() (in module dragonn.synthetic.util), 19

S

sampleFromDiscreteDistribution() (in module dragonn.synthetic.util), 19
SampleFromDiscreteDistributionSubstringGenerator (class in dragonn.synthetic.synthetic), 10
sampleFromNumSteps() (in module dragonn.synthetic.util), 19
sampleFromProbsArr() (in module dragonn.synthetic.util), 19
sampleFromRangeWithStepSize() (in module dragonn.synthetic.util), 20
sampleIndexWithinRegionOfLength() (in module dragonn.synthetic.synthetic), 13
sampleNinstancesFromDiscreteDistribution() (in module dragonn.synthetic.util), 20
sampleWithoutReplacement() (in module dragonn.synthetic.util), 20
score() (dragonn.models.Model method), 21
selectMutation() (dragonn.synthetic.synthetic.AbstractApplySingleMutationFromSet method), 3
selectMutation() (dragonn.synthetic.synthetic.ChooseMutationAtRandom method), 6
sendEmail() (in module dragonn.synthetic.util), 20
sendEmails() (in module dragonn.synthetic.util), 20
seqTo2Dimage() (in module dragonn.synthetic.util), 20
seqTo2DImages_fillInArray() (in module dragonn.synthetic.util), 20
SequenceDNN (class in dragonn.models), 21
SequenceDNN.LossHistory (class in dragonn.models), 22
SequenceDNN.PrintMetrics (class in dragonn.models), 22
SequenceDNN_learning_curve() (in module dragonn.tutorial_utils), 26
setCol() (dragonn.synthetic.util.TitledArr method), 15
setColNames() (dragonn.synthetic.util.Titled2DMatrix method), 15
setOfSeqsTo2Dimages() (in module dragonn.synthetic.util), 20
shuffleArray() (in module dragonn.synthetic.util), 20

simple_motif_embedding() (in module dragonn.simulations), 23
simulate_differential_accessibility() (in module dragonn.simulations), 23
simulate_heterodimer_grammar() (in module dragonn.simulations), 24
simulate_motif_counting() (in module dragonn.simulations), 24
simulate_motif_density_localization() (in module dragonn.simulations), 24
simulate_multi_motif_embedding() (in module dragonn.simulations), 25
simulate_single_motif_detection() (in module dragonn.simulations), 25
sortByLabels() (in module dragonn.synthetic.util), 20
SparseArrFromDict (class in dragonn.synthetic.util), 15
splitChromStartEnd() (in module dragonn.synthetic.util), 20
splitIgnoringQuotes() (in module dragonn.synthetic.util), 20
splitIntegerIntoProportions() (in module dragonn.synthetic.util), 20
SplitNames (in module dragonn.synthetic.util), 15
standardize_polygons_str() (in module dragonn.plot), 23
StringEmbeddable (class in dragonn.synthetic.synthetic), 11
submitProcess() (in module dragonn.synthetic.util), 20
SubstringEmbeddableGenerator (class in dragonn.synthetic.synthetic), 11
SubstringEmbedder (class in dragonn.synthetic.synthetic), 11
sumAbsDifferences() (in module dragonn.synthetic.util), 20
sumNumpyArrays() (in module dragonn.synthetic.util), 20
SVC (class in dragonn.models), 21
swapIndices() (in module dragonn.synthetic.util), 20

T

TeeOutputStreams (class in dragonn.synthetic.util), 15
test() (dragonn.models.Model method), 21
test_SequenceDNN() (in module dragonn.tutorial_utils), 26
throwErrorIfUnequalSets() (in module dragonn.synthetic.util), 20
Titled2DMatrix (class in dragonn.synthetic.util), 15
TitledArr (class in dragonn.synthetic.util), 15
TitledMapping (class in dragonn.synthetic.util), 15
TitledMappingIterator (class in dragonn.synthetic.util), 15
TopNMutationsFromPwmRelativeToBestHit (class in dragonn.synthetic.synthetic), 11
TopNMutationsFromPwmRelativeToBestHit_FromLoadedMotifs (class in dragonn.synthetic.synthetic), 11

[train\(\)](#) (dragonn.models.DecisionTree method), [21](#)
[train\(\)](#) (dragonn.models.gkmSVM method), [22](#)
[train\(\)](#) (dragonn.models.Model method), [21](#)
[train\(\)](#) (dragonn.models.MotifScoreRNN method), [21](#)
[train\(\)](#) (dragonn.models.SequenceDNN method), [22](#)
[train\(\)](#) (dragonn.models.SVC method), [21](#)
[train_SequenceDNN\(\)](#) (in module dragonn.tutorial_utils), [26](#)
[transform\(\)](#) (dragonn.synthetic.synthetic.AbstractApplySingleMutationFromSet method), [3](#)
[transform\(\)](#) (dragonn.synthetic.synthetic.AbstractTransformation method), [6](#)
[transform\(\)](#) (dragonn.synthetic.synthetic.RevertToReference method), [10](#)
[transform\(\)](#) (dragonn.synthetic.util.ArgumentToAdd method), [13](#)
[transform\(\)](#) (dragonn.synthetic.util.ArrArgument method), [14](#)
[transform\(\)](#) (dragonn.synthetic.util.BooleanArgument method), [14](#)
[transform\(\)](#) (dragonn.synthetic.util.CoreFileNameArgument method), [14](#)
[TransformedSubstringGenerator](#) (class in dragonn.synthetic.synthetic), [11](#)
[transformType\(\)](#) (in module dragonn.synthetic.util), [20](#)

U

[UniformIntegerGenerator](#) (class in dragonn.synthetic.synthetic), [12](#)
[UniformPositionGenerator](#) (class in dragonn.synthetic.synthetic), [12](#)
[updateAdditionalInfo\(\)](#) (dragonn.synthetic.synthetic.AdditionalInfo method), [6](#)
[updateTrace\(\)](#) (dragonn.synthetic.synthetic.AdditionalInfo method), [6](#)

V

[valToIndexMap\(\)](#) (in module dragonn.synthetic.util), [20](#)
[VariableWrapper](#) (class in dragonn.synthetic.util), [16](#)

W

[write\(\)](#) (dragonn.synthetic.util.TeeOutputStreams method), [15](#)
[writeable\(\)](#) (dragonn.synthetic.util.TeeOutputStreams method), [15](#)
[writelines\(\)](#) (dragonn.synthetic.util.TeeOutputStreams method), [15](#)

X

[X_test](#) (dragonn.tutorial_utils.Data attribute), [25](#)
[X_train](#) (dragonn.tutorial_utils.Data attribute), [25](#)
[X_valid](#) (dragonn.tutorial_utils.Data attribute), [26](#)

[XOREmbedder](#) (class in dragonn.synthetic.synthetic), [12](#)

Y

[y_test](#) (dragonn.tutorial_utils.Data attribute), [26](#)
[y_train](#) (dragonn.tutorial_utils.Data attribute), [26](#)
[y_valid](#) (dragonn.tutorial_utils.Data attribute), [26](#)
[yamlToArgsString\(\)](#) (in module dragonn.synthetic.util), [20](#)

Z

[ZeroInflator](#) (class in dragonn.synthetic.synthetic), [12](#)
[ZeroOrderBackgroundGenerator](#) (class in dragonn.synthetic.synthetic), [12](#)