

Package ‘gkmSVM’

January 21, 2018

Type Package

Title Gapped-Kmer Support Vector Machine

Version 0.79.0

Date 2018-01-15

Author Mahmoud Ghandi

Maintainer Mahmoud Ghandi <mghandi@gmail.com>

Description Imports the 'gkmSVM' v2.0 functionalities into R <<http://www.beerlab.org/gkmsvm/>>
It also uses the 'kernlab' library (separate R package by different authors) for various SVM algorithms.

License GPL (>= 2)

Imports kernlab, seqinr, utils, ROCR, BiocGenerics, IRanges, S4Vectors, Rcpp, grDevices, graphics, stats, Biostrings, GenomeInfoDb, GenomicRanges, rtracklayer

Suggests BSgenome.Hsapiens.UCSC.hg18.masked, BSgenome.Hsapiens.UCSC.hg19.masked

LinkingTo Rcpp

SystemRequirements C++11

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-01-18 18:49:09

RoxygenNote 5.0.1

R topics documented:

| | |
|---------------------------|----|
| gkmSVM-package | 2 |
| genNullSeqs | 4 |
| gkmsvm_classify | 6 |
| gkmsvm_delta | 7 |
| gkmsvm_kernel | 10 |
| gkmsvm_train | 11 |
| gkmsvm_trainCV | 12 |

gkmSVM-package

*Gapped-Kmer Support Vector Machine***Description**

Imports the 'gkmSVM' v2.0 functionalities into R <<http://www.beerlab.org/gkmsvm/>> . It also uses the 'kernlab' library (separate R package by different authors) for various SVM algorithms.

Details

Package: gkmSVM
 Type: Package
 Version: 0.79.0
 Date: 2018-01-15
 License: GPL (>= 2)

The gkm-SVM provides implementation of a new SVM kernel method using gapped k-mers as features for DNA or Protein sequences.

There are three main functions in the gkmSVM package:

gkmsvm_kernel: computes the kernel matrix

gkmsvm_train: computes the SVM coefficients

gkmsvm_classify: scores new sequences using the SVM model

Tutorial

=====

We introduce the users to the basic workflow of our gkmSVM step-by-step. Please refer to help messages for more detailed information of each function.

1) making a kernel matrix

First of all, we should calculate a full kernel matrix before training SVM classifiers. In this tutorial, we are going to use test_positives.fa as a positive set, and test_negatives.fa as a negative set.

#Input file names:

posfn= 'test_positives.fa' #positive set (FASTA format)

negfn= 'test_negatives.fa' #negative set (FASTA format)

testfn= 'test_testset.fa' #test set (FASTA format)

Alternatively if the negative set is not available, and positive set is provided as a bed file, gen-NullSeqs function could be used to generate the negative set and positive set sequences.

#Output file names:

kernelfn= 'test_kernel.txt' #kernel matrix

svmfprfx= 'test_svmtrain' #SVM files

```
outfn = 'output.txt' #output scores for sequences in the test set
```

```
gkmsvm_kernel(posfn, negfn, kernelfn); #computes kernel
```

2) training SVM

We can now train a SVM classifier using the kernel matrix generated above. For that we use `gkmsvm_train` function. It takes four arguments; kernel file, positive sequences file, negative sequences file, and prefix of output file names for the svm model.

```
gkmsvm_train(kernelfn, posfn, negfn, svmfprfx); #trains SVM
```

It will generate two files, `test_svmtrain_svalpha.out` and `test_svmtrain_svseq.fa`, which will then be used for classification/scoring of test sequences as described below.

3) classification using SVM

`gkmsvm_classify` can be used to score any set of sequences. Here, we will score the test sequences which are given in `test_testset.fa`. Note that the same set of parameters used in the `gkmsvm_kernel` should always be specified for optimal classification (here we used default parameters).

```
gkmsvm_classify(testfn, svmfprfx, outfn); #scores test sequences
```

In a more advanced example, we set the word length `L=18`, and the number of non-gapped positions `K=7`, and maximum number of mismatches `maxnmm=4`:

```
gkmsvm_kernel(posfn, negfn, kernelfn, L=18, K=7, maxnmm=4); #computes kernel
```

```
gkmsvm_train(kernelfn, posfn, negfn, svmfprfx); #trains SVM
```

```
gkmsvm_classify(testfn, svmfprfx, outfn, L=18, K=7, maxnmm=4); #scores test sequences
```

In another example, we run a 5-fold cross validation to plot the ROC curves:

```
gkmsvm_kernel(posfn, negfn, kernelfn); #computes kernel
```

```
cvres = gkmsvm_trainCV(kernelfn, posfn, negfn, svmfprfx, outputPDFfn='ROC.pdf', outputCVpredfn='cvpred.out');  
#trains SVM, plots ROC and PRC curves, and outputs model predictions.
```

Author(s)

Mahmoud Ghandi

Maintainer: Mahmoud Ghandi <mghandi@gmail.com>

References

Ghandi M, Lee D, Mohammad-Noori M, Beer MA. 2014. Enhanced Regulatory Sequence Prediction Using Gapped k-mer Features. *PLoS Comput Biol* 10: e1003711.

Ghandi M, Mohammad-Noori M, Ghareghani N, Lee D, Garraway LA, and Beer MA. *gkmSVM* an R package for gapped-kmer SVM, *Bioinformatics* (under review)

Examples

```
#Input file names:
```

```
posfn= 'test_positives.fa' #positive set (FASTA format)
```

```
negfn= 'test_negatives.fa' #negative set (FASTA format)
```

```
testfn= 'test_testset.fa' #test set (FASTA format)
```

```
#Output file names:
```

```

kernelfn= 'test_kernel.txt' #kernel matrix
svmfprfx= 'test_svmtrain' #SVM files
outfn = 'output.txt' #output scores for sequences in the test set

# gkmsvm_kernel(posfn, negfn, kernelfn); #computes kernel
# gkmsvm_train(kernelfn, posfn, negfn, svmfprfx); #trains SVM
# gkmsvm_classify(testfn, svmfprfx, outfn); #scores test sequences

# using L=18, K=7, maxnmm=4

# gkmsvm_kernel(posfn, negfn, kernelfn, L=18, K=7, maxnmm=4); #computes kernel
# gkmsvm_train(kernelfn, posfn, negfn, svmfprfx); #trains SVM
# gkmsvm_classify(testfn, svmfprfx, outfn, L=18, K=7, maxnmm=4); #scores test sequences

```

| | |
|-------------|--|
| genNullSeqs | <i>Generating GC/repeat matched randomly selected genomic sequences for the negative set</i> |
|-------------|--|

Description

Generates null sequences (negative set) with matching repeat and GC content as the input bed file for positive set regions.

Usage

```

genNullSeqs(
  inputBedFN,
  genomeVersion='hg19',
  outputBedFN = 'negSet.bed',
  outputPosFastaFN = 'posSet.fa',
  outputNegFastaFN = 'negSet.fa',
  xfold = 1,
  repeat_match_tol = 0.02,
  GC_match_tol = 0.02,
  length_match_tol = 0.02,
  batchsize = 5000,
  nMaxTrials = 20,
  genome = NULL)

```

Arguments

| | |
|---------------|--|
| inputBedFN | positive set regions |
| genomeVersion | genome version: 'hg19' and 'hg18' are supported. Default='hg19'. For other genomes, provide the BSgenome object using parameter 'genome' |
| outputBedFN | output file name for the null sequences genomic regions. Default='negSet.bed' |

| | |
|------------------|--|
| outputPosFastaFN | output file name for the positive set sequences. Default='posSet.fa' |
| outputNegFastaFN | output file name for the negative set sequences. Default='negSet.fa' |
| xfold | controls the desired number of sequences in the negative set. Default=1 (same number as in positive set) |
| repeat_match_tol | tolerance for difference in repeat ratio. Default=0.02 (repeat content difference of 0.02 or less is acceptable) |
| GC_match_tol | tolerance for difference in GC content. Default=0.02 |
| length_match_tol | tolerance for difference in relative sequence length. Default=0.02 |
| batchsize | number of candidate random sequences tested in each trial. Default=5000 |
| nMaxTrials | maximum number of trials. Default=20. |
| genome | BSgenome object. Default=NULL. If this parameter is used, parameter genomeVersion is ignored. |

Value

Writes the null sequences to files with the provided filenames. Outputs the filename for the output negative sequences file.

Author(s)

Mahmoud Ghandi

Examples

```
# Example 1:
# genNullSeqs('ctcfpos.bed' );

#Example 2:
# genNullSeqs('ctcfpos.bed', nMaxTrials=3, xfold=2, genomeVersion = 'hg18' );

#Example 3:
# genNullSeqs('ctcfpos.bed', xfold=2, genomeVersion = 'hg18', outputBedFN = 'ctcf_negSet.bed',
# outputPosFastaFN = 'ctcf_posSet.fa',outputNegFastaFN = 'ctcf_negSet.fa' );

#Example 4:
# Input file names:

posBedFN = 'test_positives.bed' # positive set genomic ranges (bed format)
genomeVer = 'hg19' #genome version
testfn= 'test_testset.fa' #test set (FASTA format)

# output file names:
posfn= 'test_positives.fa' #positive set (FASTA format)
negfn= 'test_negatives.fa' #negative set (FASTA format)
```

```

kernelfn= 'test_kernel.txt' #kernel matrix
svmfnprefix= 'test_svmtrain' #SVM files
outfn = 'output.txt' #output scores for sequences in the test set

# genNullSeqs(posBedFN, genomeVersion = genomeVer,
#   outputPosFastaFN = posfn, outputNegFastaFN = negfn );

# gkmsvm_kernel(posfn, negfn, kernelfn); #computes kernel
# gkmsvm_train(kernelfn, posfn, negfn, svmfnprefix); #trains SVM
# gkmsvm_classify(testfn, svmfnprefix, outfn); #scores test sequences

# using L=18, K=7, maxnmm=4

# gkmsvm_kernel(posfn, negfn, kernelfn, L=18, K=7, maxnmm=4); #computes kernel
# gkmsvm_train(kernelfn, posfn, negfn, svmfnprefix); #trains SVM
# gkmsvm_classify(testfn, svmfnprefix, outfn, L=18, K=7, maxnmm=4); #scores test sequences

```

gkmsvm_classify

Classifying(/scoring) new sequences using the gkmSVM model

Description

Given support vectors SVs and corresponding coefficients alphas and a set of sequences, calculates the SVM scores for the sequences.

Usage

```

gkmsvm_classify(seqfile, svmfnprefix, outfile, L=10, K=6, maxnmm=3,
maxseqlen=10000, maxnumseq=1000000, useTgkm=1, alg=0, addRC=TRUE, usePseudocnt=FALSE,
batchSize=100000, wildcardLambda=1.0, wildcardMismatchM=2, alphabetFN="NULL",
svseqfile=NA, alphafile=NA)

```

Arguments

| | |
|-------------|--|
| seqfile | input sequences file name (FASTA format) |
| svmfnprefix | SVM model file name prefix |
| outfile | output file name |
| L | word length, default=10 |
| K | number of informative columns, default=6 |
| maxnmm | maximum number of mismatches to consider, default=3 |
| maxseqlen | maximum sequence length in the sequence files, default=10000 |
| maxnumseq | maximum number of sequences in the sequence files, default=1000000 |
| useTgkm | filter type: 0(use full filter), 1(use truncated filter: this gaurantees non-negative counts for all L-mers), 2(use h[m], gkm count vector), 3(wildcard), 4(mismatch), default=1 |

| | |
|-------------------|---|
| alg | algorithm type: 0(auto), 1(XOR Hashtable), 2(tree), default=0 |
| addRC | adds reverse complement sequences, default=TRUE |
| usePseudocnt | adds a constant to count estimates, default=FALSE |
| batchSize | number of sequences to compute scores for in batch, default=100000 |
| wildcardLambda | lambda for wildcard kernel, default=0.9 |
| wildcardMismatchM | max mismatch for Mismatch kernel or wildcard kernel, default=2 |
| alphabetFN | alphabets file name, if not specified, it is assumed the inputs are DNA sequences |
| svseqfile | SVM support vectors sequence file name (not needed if svmfprfx is provided) |
| alphafile | SVM support vectors weights file name (not needed if svmfprfx is provided) |

Details

classification using SVM: gkmsvm_classify can be used to score any set of sequences. Note that the same set of parameters (L, K, maxmm) used in the gkmsvm_kernel should be specified for optimal classification.

```
gkmsvm_classify(testfn, svmfprfx, outfn); #scores test sequences
```

Author(s)

Mahmoud Ghandi

Examples

```
#Input file names:
posfn= 'test_positives.fa' #positive set (FASTA format)
negfn= 'test_negatives.fa' #negative set (FASTA format)
testfn= 'test_testset.fa' #test set (FASTA format)

#Output file names:
kernelfn= 'test_kernel.txt' #kernel matrix
svmfprfx= 'test_svmtrain' #SVM files
outfn = 'output.txt' #output scores for sequences in the test set

# gkmsvm_kernel(posfn, negfn, kernelfn); #computes kernel
# gkmsvm_train(kernelfn, posfn, negfn, svmfprfx); #trains SVM
# gkmsvm_classify(testfn, svmfprfx, outfn); #scores test sequences
```

gkmsvm_delta

Calculating deltaSVM scores

Description

Given support vectors SVs and corresponding coefficients alphas and a pair of file test sequence files (one for reference allele, and one for alternate allele), calculates the deltaSVM scores for the sequences.

Usage

```
gkmsvm_delta(seqfile_allele1, seqfile_allele2, svmfprfx, outfile, L = 10,
K = 6, maxnmm = 3, maxseqlen = 10000, maxnumseq = 1e+06, useTgkm = 1, alg = 2,
addRC = TRUE, usePseudocnt = FALSE, batchSize = 1e+05, wildcardLambda = 1,
wildcardMismatchM = 2, alphabetFN = "NULL", svseqfile = NA, alphafile = NA,
outfile_allele1 = NA, outfile_allele2 = NA)
```

Arguments

| | |
|-------------------|--|
| seqfile_allele1 | fasta file containing the test sequences (reference allele) |
| seqfile_allele2 | fasta file containing the test sequences (alternate allele). The sequences in this file should be in the exact same order as in seqfile_allele1. |
| svmfprfx | SVM model file name prefix |
| outfile | output file name |
| L | word length, default=10 |
| K | number of informative columns, default=6 |
| maxnmm | maximum number of mismatches to consider, default=3 |
| maxseqlen | maximum sequence length in the sequence files, default=10000 |
| maxnumseq | maximum number of sequences in the sequence files, default=1000000 |
| useTgkm | filter type: 0(use full filter), 1(use truncated filter: this gaurantees non-negative counts for all L-mers), 2(use h[m], gkm count vector), 3(wildcard), 4(mismatch), default=1 |
| alg | algorithm type: 0(auto), 1(XOR Hashtable), 2(tree), default=0 |
| addRC | adds reverse complement sequences, default=TRUE |
| usePseudocnt | adds a constant to count estimates, default=FALSE |
| batchSize | number of sequences to compute scores for in batch, default=100000 |
| wildcardLambda | lambda for wildcard kernel, defaul=0.9 |
| wildcardMismatchM | max mismatch for Mismatch kernel or wildcard kernel, default=2 |
| alphabetFN | alphabets file name, if not specified, it is assumed the inputs are DNA sequences |
| svseqfile | SVM support vectors sequence file name (not needed if svmfprfx is provided) |
| alphafile | SVM support vectors weights file name (not needed if svmfprfx is provided) |
| outfile_allele1 | output filename for gkmSVM scores for the reference sequences (optional) |
| outfile_allele2 | output filename for gkmSVM scores for the alternate sequences (optional) |

Details

predicting the effect of variants using gkmSVM model: gkmsvm_delta can be used to predict the effect of sequence variants. The sequences corresponding to reference allele and alternate alleles are given in two separate files. gkmSVM is used to score each set of sequences, and the difference in the gkmSVM score for the reference and alternate allele is reported. Note that the same set of parameters (L, K, maxnmm) used in the gkmsvm_kernel should be specified for optimal scoring.

```
gkmsvm_kernel(seqfile_allele1, seqfile_allele2, svmfnprefix, outfn); #scores test sequences
```

Value

deltaSVM scores

Author(s)

Mahmoud Ghandi

References

Ghandi M, Mohammad-Noori M, Ghareghani N, Lee D, Garraway LA, and Beer MA. gkmSVM: an R package for gapped-kmer SVM, Bioinformatics 2016.

Ghandi M, Lee D, Mohammad-Noori M, Beer MA. 2014. Enhanced Regulatory Sequence Prediction Using Gapped k-mer Features. PLoS Comput Biol 10: e1003711.

Lee D, Gorkin DU, Baker M, Strober BJ, Asoni AL, McCallion AS, and Beer MA. A method to predict the impact of regulatory variants from DNA sequence. Nature Genetics 2015.

Examples

```
#Input file names:
posfn= 'test_positives.fa' #positive set (FASTA format)
negfn= 'test_negatives.fa' #negative set (FASTA format)
testfn_ref= 'test_testsetRef.fa' #test set (reference allele) (FASTA format)
testfn_alt= 'test_testsetAlt.fa' #test set (alternate allele) (FASTA format)

#Output file names:
kernelfn= 'test_kernel.txt' #kernel matrix
svmfnprefix= 'test_svmtrain' #SVM files
outfn = 'output.txt' #output delta svm scores for sequences in the test set

# gkmsvm_kernel(posfn, negfn, kernelfn); #computes kernel
# gkmsvm_train(kernelfn, posfn, negfn, svmfnprefix); #trains SVM
# gkmsvm_delta(testfn_ref, testfn_alt, svmfnprefix, outfn); #scores test sequences
```

gkmsvm_kernel

*Computing the kernel matrix***Description**

Generates a lower triangle of kernel matrix (i.e. pairwise similarities) between the sequences.

Usage

```
gkmsvm_kernel(posfile, negfile, outfile, L=10, K=6, maxnmm=3, maxseqlen=10000,
maxnumseq=1000000, useTgkm=1, alg=0, addRC=TRUE, usePseudocnt=FALSE, wildcardLambda=1.0,
wildcardMismatchM=2, alphabetFN="NULL")
```

Arguments

| | |
|-------------------|--|
| posfile | positive sequences file name (FASTA format) |
| negfile | negative sequences file name (FASTA format) |
| outfile | output file name |
| L | word length, default=10 |
| K | number of informative columns, default=6 |
| maxnmm | maximum number of mismatches to consider, default=3 |
| maxseqlen | maximum sequence length in the sequence files, default=10000 |
| maxnumseq | maximum number of sequences in the sequence files, default=1000000 |
| useTgkm | filter type: 0(use full filter), 1(use truncated filter: this gaurantees non-negative counts for all L-mers), 2(use h[m], gkm count vector), 3(wildcard), 4(mismatch), default=1 |
| alg | algorithm type: 0(auto), 1(XOR Hashtable), 2(tree), default=0 |
| addRC | adds reverse complement sequences, default=TRUE |
| usePseudocnt | adds a constant to count estimates, default=FALSE |
| wildcardLambda | lambda for wildcard kernel, defaul=0.9 |
| wildcardMismatchM | max mismatch for Mismatch kernel or wildcard kernel, default=2 |
| alphabetFN | alphabets file name, if not specified, it is assumed the inputs are DNA sequences |

Details

It calculates the full kernel matrix that can be then used to train an SVM classifier. `gkmsvm_kernel(posfn, negfn, kernelfn)`;

Author(s)

Mahmoud Ghandi

Examples

```

#Input file names:
posfn= 'test_positives.fa' #positive set (FASTA format)
negfn= 'test_negatives.fa' #negative set (FASTA format)
testfn= 'test_testset.fa' #test set (FASTA format)

#Output file names:
kernelfn= 'test_kernel.txt' #kernel matrix
svmfnprefix= 'test_svmtrain' #SVM files
outfn = 'output.txt' #output scores for sequences in the test set

# gkmsvm_kernel(posfn, negfn, kernelfn); #computes kernel
# gkmsvm_train(kernelfn, posfn, negfn, svmfnprefix); #trains SVM
# gkmsvm_classify(testfn, svmfnprefix, outfn); #scores test sequences

```

| | |
|--------------|-------------------------------|
| gkmsvm_train | <i>Training the SVM model</i> |
|--------------|-------------------------------|

Description

Using the kernel matrix created by 'gkmsvm_kernel', this function trains the SVM classifier. Here we rely on the 'kernlab' package, and merely provide a wrapper function.

Usage

```
gkmsvm_train(kernelfn, posfn, negfn, svmfnprefix, Type="C-svc", C=1, shrinking=FALSE, ...)
```

Arguments

| | |
|-------------|--|
| kernelfn | kernel matrix file name |
| posfn | positive sequences file name |
| negfn | negative sequences file name |
| svmfnprefix | output SVM model file name prefix |
| Type | optional: SVM type (default='C-svc'), see 'kernlab' documentation for more details. |
| C | optional: SVM parameter C (default=1), see 'kernlab' documentation for more details. |
| shrinking | optional: shrinking parameter for kernlab (default=FALSE), see 'kernlab' documentation for more details. |
| ... | optional: additional SVM parameters, see 'kernlab' documentation for more details. |

Details

Trains SVM classifier and generates two files: [svmfnprefix]_svalpha.out for SVM alphas and the other for the corresponding SV sequences ([svmfnprefix]_svseq.fa)

Author(s)

Mahmoud Ghandi

Examples

```

#Input file names:
posfn= 'test_positives.fa'  #positive set (FASTA format)
negfn= 'test_negatives.fa'  #negative set (FASTA format)
testfn= 'test_testset.fa'   #test set (FASTA format)

#Output file names:
kernelfn= 'test_kernel.txt' #kernel matrix
svmfnprefix= 'test_svmtrain' #SVM files
outfn = 'output.txt'        #output scores for sequences in the test set

# gkmsvm_kernel(posfn, negfn, kernelfn);           #computes kernel
# gkmsvm_train(kernelfn, posfn, negfn, svmfnprefix); #trains SVM
# gkmsvm_classify(testfn, svmfnprefix, outfn);      #scores test sequences

```

| | |
|----------------|--|
| gkmsvm_trainCV | <i>Training the SVM model, using repeated CV to tune parameter C and plot ROC curves</i> |
|----------------|--|

Description

Using the kernel matrix created by 'gkmsvm_kernel', this function trains the SVM classifier. It uses repeated CV to find optimum SVM parameter C. Also generates ROC and PRC curves.

Usage

```

gkmsvm_trainCV(kernelfn, posfn, negfn, svmfnprefix=NA,
  nCV=5, nrepeat=1, cv=NA, Type="C-svc", C=1, shrinking=FALSE,
  showPlots=TRUE, outputPDFfn=NA, outputCVpredfn=NA, outputROCFn=NA, ...)

```

Arguments

| | |
|-------------|---|
| kernelfn | kernel matrix file name |
| posfn | positive sequences file name |
| negfn | negative sequences file name |
| svmfnprefix | (optional) output SVM model file name prefix |
| nCV | (optional) number of CV folds |
| nrepeat | (optional) number of repeated CVs |
| cv | (optional) CV group label. An array of length (npos+nneg), containing CV group number (between 1 and nCV) for each sequence |
| Type | (optional) SVM type (default='C-svc'), see 'kernlab' documentation for more details. |

| | |
|----------------|--|
| C | (optional)a vector of all values of C (SVM parameter) to be tested. (default=1), see 'kernlab' documentation for more details. |
| shrinking | optional: shrinking parameter for kernlab (default=FALSE), see 'kernlab' documentation for more details. |
| showPlots | generate plots (default==TRUE) |
| outputPDFfn | filename for output PDF, default=NA (no PDF output) |
| outputCVpredfn | filename for output cvpred (predicted CV values), default=NA (no output) |
| outputROCFn | filename for output auROC (Area Under an ROC Curve) and auPRC (Area Under the Precision Recall Curve) values, default=NA (no output) |
| ... | optional: additional SVM parameters, see 'kernlab' documentation for more details. |

Details

Trains SVM classifier and generates two files: [svmfnprefix]_svalpha.out for SVM alphas and the other for the corresponding SV sequences ([svmfnprefix]_svseq.fa)

Author(s)

Mahmoud Ghandi

Examples

```
#Input file names:
posfn= 'test_positives.fa' #positive set (FASTA format)
negfn= 'test_negatives.fa' #negative set (FASTA format)
testfn= 'test_testset.fa' #test set (FASTA format)

#Output file names:
kernelfn= 'test_kernel.txt' #kernel matrix
svmfnprefix= 'test_svmtrain' #SVM files
outfn = 'output.txt' #output scores for sequences in the test set

# gkmsvm_kernel(posfn, negfn, kernelfn); #computes kernel
# cvres = gkmsvm_trainCV(kernelfn,posfn, negfn, svmfnprefix,
#   outputPDFfn='ROC.pdf', outputCVpredfn='cvpred.out');
# #trains SVM, plots ROC and PRC curves, and outputs model predictions.
# gkmsvm_classify(testfn, svmfnprefix, outfn); #scores test sequences
```

Index

- *Topic **SVM**
 - gkmSVM-package, [2](#)
- *Topic **\textasciitildekwd1**
 - gkmsvm_delta, [7](#)
- *Topic **\textasciitildekwd2**
 - gkmsvm_delta, [7](#)
- *Topic **gkmSVM**
 - gkmSVM-package, [2](#)
- *Topic **kernel**
 - gkmSVM-package, [2](#)
- *Topic **package**
 - gkmSVM-package, [2](#)

- genNullSeqs, [4](#)
- gkmSVM (gkmSVM-package), [2](#)
- gkmSVM-package, [2](#)
- gkmsvm_classify, [6](#)
- gkmsvm_delta, [7](#)
- gkmsvm_kernel, [10](#)
- gkmsvm_train, [11](#)
- gkmsvm_trainCV, [12](#)