Package 'msvmpath'

August 21, 2007

Version 0.1-1

Date 2007-2-6

Title The solution path algorithm of multicategory support vector machines. The entire set of codes are based on Hastie's 'svmpath'. It is an add on package to 'svmpath'.

Author Zhenhuan Cui <zhenhuan@stat.ohio-state.edu>, Trevor Hastie

Maintainer Zhenhuan Cui <zhenhuan@stat.ohio-state.edu>

Depends svmpath

Description This function gives the whole set of solutions to the multicategory support vector machines at every possible value of the tuning parameter through one fit of the data.

R topics documented:

`cv.msvmpath` *Cross Validation for msvmpath*

Description

This function cross validates msvmpath for the optimal lambda

Usage

```
cv.msvmpath=function(x,y,fold=10,Nlambda=101,kernel.function =
poly.kernel, gpw.init=0.005, param.kernel = 2^nlm(BSS, p=gpw.init,
iterlim=1000, x=x, y=y)$estimate,  eps = 1e-10, Nmoves=n.class*n,
digits = 15 ,lambda.min = 1e-10,cost=cost.dft,
wt.obs=wt.dft,lag=30,...)
```

Arguments

`x`	a data matrix with n observations (rows) and p covariates (columns) for training.
`y`	a vector with 1,2,...,k valued class labels for training.
`fold`	the number of folds of the cross validation. The only valid value of k larger than the smallest class size is n, for which leave-one-out cross validation will be conducted.
`Nlambda`	the number of equally spaced lambda values in the range of lambda.
`kernel.function`	the kernel function. Only poly.kernel and radial.kernel are available now.
`gpw.init`	2 to the power of `gpw.init` is the initial value of the parameter of the radial kernel.
`param.kernel`	the parameter for the kernel functions.
`eps`	a small value for tolerance in the stopping rule.
`Nmoves`	the maximal number of steps for the solution path.
`digits`	number of digits displayed.
`lambda.min`	the smallest value of lambda for stopping the path.
`cost`	a k by k cost matrix with ijth entry the cost of misclassifying class i as class j. The defaulf value is a k by k matrix with 0's along the diagonal and 1's elsewhere.
`wt.obs`	a vector of length n with weights for each observation. The default value is a vector with n 1's.
`lag`	the number of steps after which the early stopping rule comes into play.

Details

This function fits the solution path with the entire data set to decide a proper range of lambda. Then the range of lambda is discretized into a pre-designated number of lambda values. Cross validation is carried out over the set of lambda values.

Value

`opt.lambda`	the optimal value of lambda chosen by cross validation.
`n.lambda`	the set of lambda values for cross validation.
`cv.error`	the overall cross validation error.
`path`	the msvmpath object fitted with the entire data set.

Author(s)

Zhenhuan Cui

See Also

msvmpath,msvmpath,msvmpath, msvmpath, msvmpath

Examples

```
data(msvmpath.training)
data(msvmpath.tuning)
data(msvmpath.test)
x.training=as.matrix(msvmpath.training[,1:2])
y.training=msvmpath.training$y
x.tuning=as.matrix(msvmpath.tuning[,1:2])
y.tuning=msvmpath.tuning$y
x.test=as.matrix(msvmpath.test[,1:2])
y.test=msvmpath.test$y

mypath.cv=cv.msvmpath(x.training, y.training,fold=5, kernel.function=radial.kernel)
y.cv.pred=predict(mypath.cv$path, x.test, mypath.cv$opt.lambda,type="class")
test.error.rate.cv=mean(y.cv.pred!=y.test)
```

Balanced.Initialization.msvm *msvm internal functions*

Description

Internal msvmpath functions

Author(s)

This function is an extension of the corresponding function in `svmpath` written by Trevor Hastie. Zhenhuan Cui generalizes its R codes to fit it in the framework of multicatory SVM.

---

msvmpath *Characterize the Solution Path for a Multicategory SVM*

Description

This algorithm extends the SVM solution path of binary cases to multicategory cases. It is based on Hastie's 'svmpath'.

msvmpath finds the entire solution path of the multicategory SVM at every possible value of the regularization parameter.This algorithm allows tuning either with a separate tuning set or a subset of the training set determined by cluster analysis.

Usage

```
msvmpath=function(x,y,kernel.function = poly.kernel, gpw.init=0.005,
param.kernel = 2^nlm(BSS, p=gpw.init, iterlim=1000, x=x,
y=y)$estimate, K = kernel.function(x, x, param.kernel = param.kernel),
x.tune=NULL,y.tune=NULL,fract=1, type="cluster",
tune.rest=FALSE,K.tuning = kernel.function(x.tune, x, param.kernel =
param.kernel), eps = 1e-10, Nmoves=n.class*n, digits = 15 ,lambda.min
= 1e-10, cost=cost.dft, wt.obs=wt.dft,lag=30,...)
```

Arguments

`x`	a data matrix with n observations (rows) and p covariates (columns) for training.
`y`	a vector with 1,2,...,k valued class labels for training.
`kernel.function`	the kernel function. Only poly.kernel and radial.kernel are available now.
`gpw.init`	2 to the power of `gpw.init` is the initial value of the parameter of the radial kernel.
`param.kernel`	the parameter for the kernel functions.
`K`	kernel matrix of the training dataset. It can be extended beyond poly.kernel and radial.kernel.
`x.tune`	a data matrix for tuning (optional).
`y.tune`	a vector of class labels for tuning (optional).
`fract`	the fraction of the training set used to generate the solution path. The default value is 1, which means the entire training set is used.
`type`	two types of data thinning approaches are provided, cluster analysis ("cluster") or simple random sample ("SRS").
`tune.rest`	if fract<1, tune.rest indicates whether the tuning error is computed using the rest of the data. If TRUE, the rest of the data are used for tuning. The default value is FALSE.
`K.tuning`	kernel matrix for the pairs from the training set and the tuning set.
`eps`	a small value for tolerance in the stopping rule.
`Nmoves`	the maximal number of steps for the solution path.
`digits`	number of digits displayed.
`error.margin`	the path is terminated when the tuning error rate increases from the previous smallest value by error.margin.
`lambda.min`	the smallest value of lambda for stopping the path.
`cost`	a k by k cost matrix with ijth entry the cost of misclassifying class i as class j. The defaulf value is a k by k matrix with 0's along the diagonal and 1's elsewhere.
`wt.obs`	a vector of length n with weights for each observation. The default value is a vector with n 1's.
`lag`	the number of steps after which the early stopping rule comes into play.

Value

An "msvmapath" object is returned with associated summary, predict, coef, and print methods.

`alpha`	the Lagrange multipliers at each break point of the tuning parameter along the path.
`alph0`	a scaled intercept at each break point of the tuning parameter along the path.
`lambda`	the sequence of break points of the tuning parameter.
`ind.opt`	the location of the optimal lambda.
`opt.lambda`	the optimal value of the tuning parameter computed with the tuning set.

`Loss`	the empirical loss at each break point of the tuning parameter.
`Error`	the training error at each break point of the tuning paramter.
`tuning.error`	the tuning error at each break point if the tuning set is given.
`Size.Elbow`	the elbow size at each break point.
`Elbow`	the set of indices in each elbow set at every break point of the tuning parameter
`Path.Length`	the length of the solution path.
`Moveto`	the set to which an observation is moving at each step.
`Movefrom`	the set from which an observation is moving at each step.
`Step`	the step number.
`wt`	an n by p matrix with the item-wise product of the cost and the weight adjusted to the dimension of the training data.

Warning

This algorithm is subject to machine errors if eps or lambda.min is too small. When the above problem occurs, increasing either eps or lambda.min can avoid the problem.

When the sample size, the number of classes or the number of covariates is large, this function can provide an approximate solution path.

Author(s)

This function is an extension of the function `svmpath` written by Trevor Hastie. Zhenhuan Cui generalized the R codes of `svmpath` to fit it in the framework of multicatory SVM.

References

Lee. Y and Cui. Z (2005). Characterizing the Solution Path of Multicategory Support Vector Machines. Statistica Sinica, vol. 16, No. 2, 391-409, 2006.

See Also

`summary.msvmpath,summary.msvmpath,summary.msvmpath, summary.msvmpath, summary.msvmpath`

Examples

```
data(msvmpath.training)
data(msvmpath.tuning)
data(msvmpath.test)
x.training=as.matrix(msvmpath.training[,1:2])
y.training=msvmpath.training$y
x.tuning=as.matrix(msvmpath.tuning[,1:2])
y.tuning=msvmpath.tuning$y
x.test=as.matrix(msvmpath.test[,1:2])
y.test=msvmpath.test$y

#-----------with a separate tuning set for the optimal lambda------------------
mypath=msvmpath(x.training, y.training,
kernel.function=radial.kernel, x.tune=x.tuning, y.tune=y.tuning)
summary(mypath)
print(mypath)
y.pred=predict(mypath, x.test, mypath$opt.lambda, type="class")
```

```
test.error.rate=mean(y.pred!=y.test)
test.error.rate

#------------split the training set for the optimal lambda-----------------------
mypath=msvmpath(x.training, y.training, fract=0.5,tune.rest=TRUE,
kernel.function=radial.kernel)
summary(mypath)
print(mypath)
y.pred=predict(mypath, x.test, mypath$opt.lambda, type="class")
test.error.rate.split=mean(y.pred!=y.test)
test.error.rate.split
```

`predict.msvmpath` *Predict Method for an msvmpath Object*

Description

This method predicts the fitted function values or the class labels at new x values or computes the Lagrange multipliers at lambda values.

Usage

```
predict.msvmpath(object, newx, lambda, type = c("function", "class", "alpha"))
```

Arguments

`object`	an msvmpath object.
`newx`	x values at which the fitted values are computed or the class labels are predicted.
`lambda`	lambda values at which the Lagrange multipliers are computed.
`type`	type of prediction with default ”function”. If type=”alpha”, the Lagrange multipliers are computed.

Details

see predict.svmpath

Value

The returned values of this function are either the fitted values or the corresponding class labels at the input data points, or the corresponding Lagrange multipliers for the input lambda values.

Author(s)

This function is an extension of the function `predict.svmpath` written by Trevor Hastie. Zhenhuan Cui generalized the R codes of `predict.svmpath` to fit it in the framework of multicatory SVM.

See Also

`msvmpath,msvmpath,msvmpath, msvmpath, msvmpath`

Examples

```
data(msvmpath.training)
data(msvmpath.tuning)
data(msvmpath.test)
x.training=as.matrix(msvmpath.training[,1:2])
y.training=msvmpath.training$y
x.tuning=as.matrix(msvmpath.tuning[,1:2])
y.tuning=msvmpath.tuning$y
x.test=as.matrix(msvmpath.test[,1:2])
y.test=msvmpath.test$y

mypath=msvmpath(x.training, y.training,
kernel.function=radial.kernel, x.tune=x.tuning, y.tune=y.tuning,
error.margin=0.02)
summary(mypath)
print(mypath)
y.pred=predict(mypath, x.test, mypath$opt.lambda, type="class")
test.error.rate=mean(y.pred!=y.test)
```

---

print.msvmpath *Print Method for an msvmpath Object*

Description

This method prints the detailed summary of msvm solution path.

Usage

```
print.msvmpath(x, digits=6,...)
```

Arguments

x	an msvmpath object.
digits	number of digits displayed.

Value

This method prints the solution path class by class. For each class, it lists the step number, the class label, the observation involved, the event of the observation, the value of lambda, the loss incurred, the elbow size and the training error. For the event of the observation, "E" stands for the elbow set; "U" stands for the upper set (left set in svmpath), "L" stands for the lower set (right set in svmpath). "->" indicates the direction of the movement. "L->E" means movement from the lower set to the elbow set.

Author(s)

This function is an extension of the function `print.svmpath` written by Trevor Hastie. Zhenhuan Cui generalized the R codes of `print.svmpath` to fit it in the framework of multicatory SVM.

See Also

`msvmpath,msvmpath,msvmpath, msvmpath, msvmpath`

Examples

```
data(msvmpath.training)
data(msvmpath.tuning)
x.training=as.matrix(msvmpath.training[,1:2])
y.training=msvmpath.training$y
x.tuning=as.matrix(msvmpath.tuning[,1:2])
y.tuning=msvmpath.tuning$y

mypath=msvmpath(x.training, y.training,
kernel.function=radial.kernel, x.tune=x.tuning, y.tune=y.tuning,
error.margin=0.02)
print(mypath)
```

---

summary.msvmpath *Summary Method for an msvmpath Object*

Description

This method provides a brief summary of the solution path with a pre-determined number of steps.

Usage

```
summary.msvmpath(object, nsteps = 5, digits = 6, ...)
```

Arguments

`object`	an msvmpath object.
`nsteps`	decides the length of the summary.
`digits`	number of digits displayed.

Details

This method lists the equally spaced steps with the first and last steps included.

Value

The returned value of this function is a data frame with the step number, values of lambda, training error, elbow size, number of support points, and the loss incurred.

Author(s)

This function is an extension of the function `summary.svmpath` written by Trevor Hastie. Zhenhuan Cui generalizes the R codes of `summary.svmpath` to fit it in the framework of multicatory SVM.

See Also

`msvmpath,msvmpath,msvmpath, msvmpath, msvmpath`

Examples

```
data(msvmpath.training)
data(msvmpath.tuning)
x.training=as.matrix(msvmpath.training[,1:2])
y.training=msvmpath.training$y
x.tuning=as.matrix(msvmpath.tuning[,1:2])
y.tuning=msvmpath.tuning$y

mypath=msvmpath(x.training, y.training,
kernel.function=radial.kernel, x.tune=x.tuning, y.tune=y.tuning,
error.margin=0.02)
summary(mypath)
```

Index