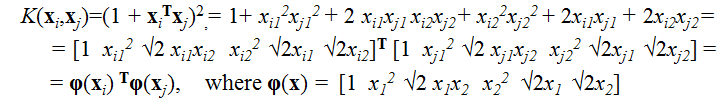
**18 - Support Vector Machines (SVM)**

**The Basic Algorithm:** (linear algebra!)

**e.g.**



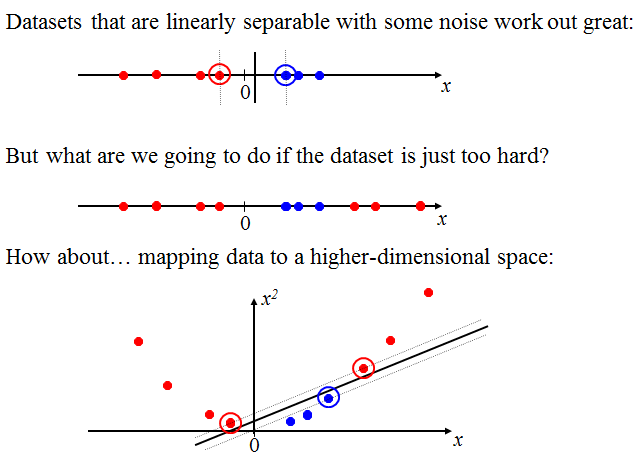
**This is sometimes called the “Kernel Trick”**

Great visualizations of SVM with the use of kernels:

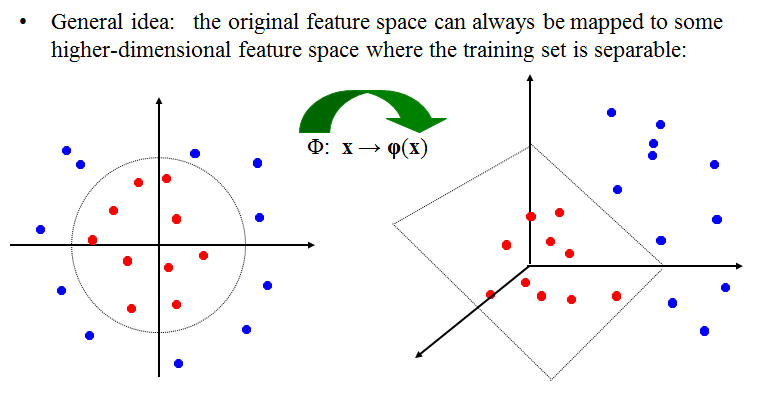
[http://www.cs.ust.hk/irproj/Regularization%20Path/svmKernelpath/2moons.avi  
http://www.cs.ust.hk/irproj/Regularization%20Path/svmKernelpath/2Gauss.avi](http://www.cs.ust.hk/irproj/Regularization%20Path/svmKernelpath/2moons.avihttp://www.cs.ust.hk/irproj/Regularization%20Path/svmKernelpath/2Gauss.avi)

<http://www.youtube.com/watch?v=3liCbRZPrZA>

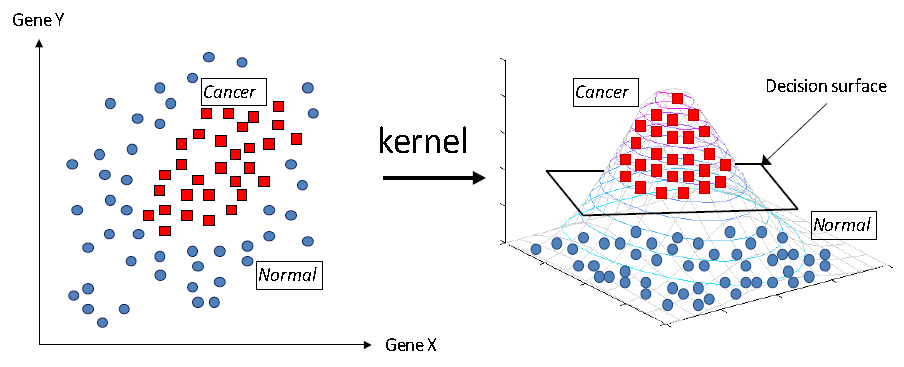
In one dimension (*p = 1*):



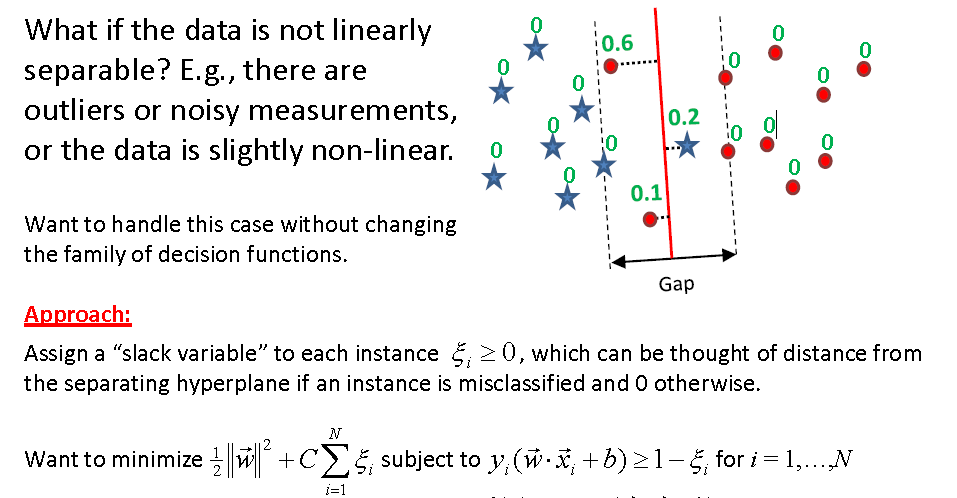
In two dimensions... *(p = 2)*



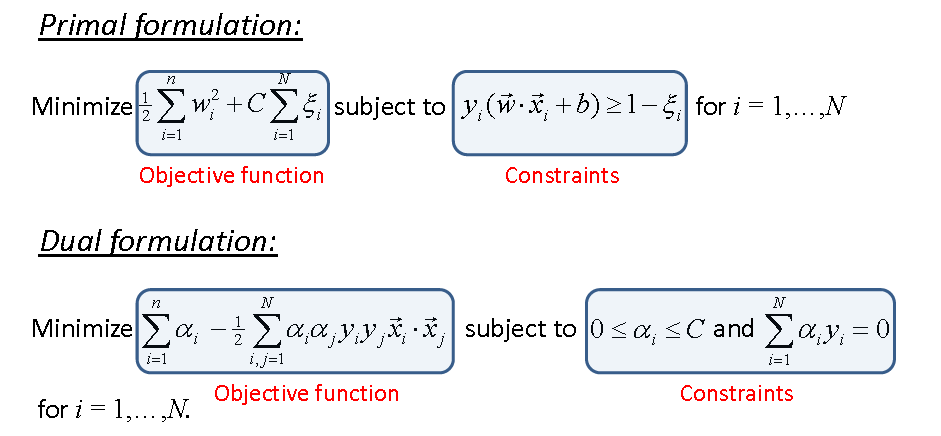
Another (*p = 2*) example:



**Soft-margin SVM**



By using similar arguments to those above, the soft-margin formulation leads to…



In R there are several packages that will fit support vector machines for classification and regression problems and there is no implementation (or plans for one) in JMP.

**Packages in R with SVM functions**  
**e1071** – contains the function svm()

**kernlab** – features a variety of kernel-based (nonlinear) SVM methods. The main function is ksvm().

**klaR** – includes a function svmlight()

**svmpath** – package developed by Hastie (one of the co-authors of the Elements of Statistical Learning text). We will not be using this one at all in this course.

**Misclassification Function for SVM (or any other classification method)**(takes fitted/predicted class and actual class as arguments)

> misclass = function (fit,y)

{

temp <- table(fit,y)

cat("Table of Misclassification\n")

cat("(row = predicted, col = actual)\n")

print(temp)

cat("\n\n")

numcor <- sum(diag(temp))

numinc <- length(y) - numcor

mcr <- numinc/length(y)

cat(paste("Misclassification Rate = ", format(mcr, digits = 3)))

cat("\n")

}

**Example 18.1 – Italian Olive Oils – classifying growing area (9 classes) and region (3 classes)**

> names(Olives)

[1] "Region.name" "Area.name" "Region" "Area" "palmitic"

[6] "palmitoleic" "strearic" "oleic" "linoleic" "eicosanoic"

[11] "linolenic" "eicosenoic"

> OlivesArea = Olives[,c(2,5:11)] 🡨 grab Area Name and all fatty acids except eicosenoic which was   
 known to have errors in the measured values.  
> names(OlivesArea)

[1] "Area.name" "palmitic" "palmitoleic" "strearic" "oleic"

[6] "linoleic" "eicosanoic" "linolenic"

> OlivesReg = Olives[,c(1,5:11)] 🡨 grab Region name and all but eicosenoic acid.

> names(OlivesReg)

[1] "Region.name" "palmitic" "palmitoleic" "strearic" "oleic"

[6] "linoleic" "eicosanoic" "linolenic"

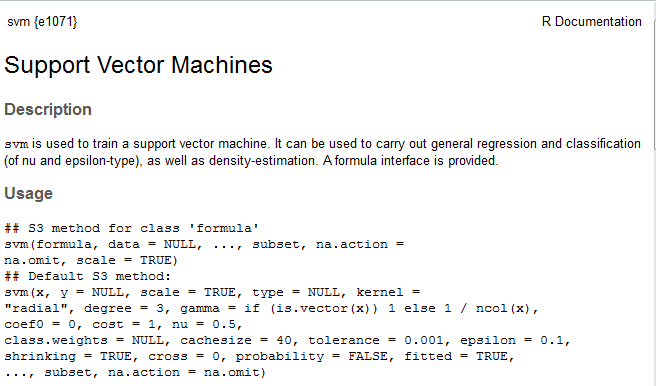
> set.seed(1)

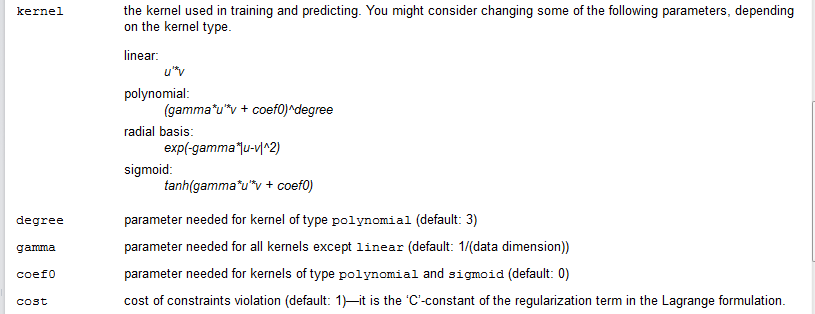
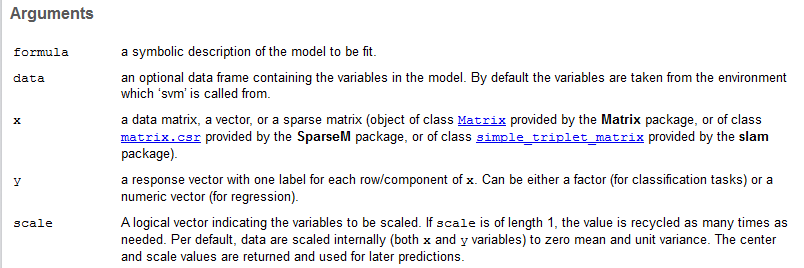
> sam = sample(1:572,floor(572\*.75),replace=F)  
  
> Area.train = OlivesArea[sam,]

> Area.test = OlivesArea[-sam,]

> Region.train = OlivesReg[sam,]

> Region.test = OlivesReg[-sam,]

**svm() from e1071**





**Example 18.1 – Italian Olive Oils (cont’d)**  
> area.svm = svm(Area.name~.,data=Area.train)

> misclass(fitted(area.svm),Area.train$Area.name)

Table of Misclassification

(row = predicted, col = actual)

y

fit Calabria Coastal-Sardinia East-Liguria Inland-Sardinia

Calabria 40 0 0 0

Coastal-Sardinia 0 22 0 0

East-Liguria 0 0 32 0

Inland-Sardinia 0 1 0 41

North-Apulia 0 0 0 0

Sicily 0 0 0 0

South-Apulia 1 0 0 0

Umbria 0 0 0 0

West-Liguria 0 0 0 0

y

fit North-Apulia Sicily South-Apulia Umbria West-Liguria

Calabria 1 3 0 0 0

Coastal-Sardinia 0 0 0 0 0

East-Liguria 2 0 0 0 0

Inland-Sardinia 0 0 0 0 0

North-Apulia 10 2 0 0 0

Sicily 0 12 0 0 0

South-Apulia 0 4 144 0 0

Umbria 0 0 0 30 0

West-Liguria 0 0 0 0 36

Misclassification Rate = 0.0367

> summary(area.svm)

Call:

svm(formula = Area.name ~ ., data = Area.train)

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 1

gamma: 0.1428571

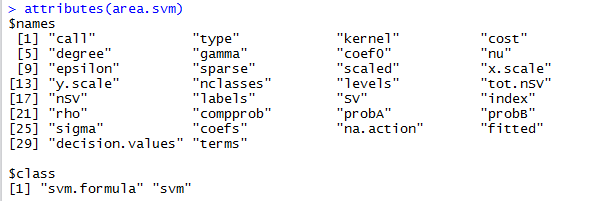
Number of Support Vectors: 183

( 35 20 24 20 13 25 12 21 13 )

Number of Classes: 9

Levels:

Calabria Coastal-Sardinia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria



> ypred = predict(area.svm,newdata=Area.test)

> misclass(ypred,Area.test$Area.name)

Table of Misclassification

(row = predicted, col = actual)

y

fit Calabria Coastal-Sardinia East-Liguria Inland-Sardinia

Calabria 15 0 0 0

Coastal-Sardinia 0 10 0 0

East-Liguria 0 0 16 0

Inland-Sardinia 0 0 0 24

North-Apulia 0 0 0 0

Sicily 0 0 0 0

South-Apulia 0 0 0 0

Umbria 0 0 0 0

West-Liguria 0 0 2 0

y

fit North-Apulia Sicily South-Apulia Umbria West-Liguria

Calabria 0 5 0 0 0

Coastal-Sardinia 0 0 0 0 0

East-Liguria 4 0 0 0 0

Inland-Sardinia 0 0 0 0 0

North-Apulia 7 1 0 0 0

Sicily 0 6 0 0 0

South-Apulia 0 3 62 0 0

Umbria 1 0 0 21 0

West-Liguria 0 0 0 0 14

Misclassification Rate = 0.0838

Increasing the cost (C) parameter will increase penalty for misclassifying observations in training data. This should drive down the misclassification rate on the training data, however this will probably increase test data misclassification due to overfitting.

> area.svm100 = svm(Area.name~.,data=Area.train,cost=100)

> misclass(fitted(area.svm100),Area.train$Area.name)

Table of Misclassification

(row = predicted, col = actual)

y

fit Calabria Coastal-Sardinia East-Liguria Inland-Sardinia

Calabria 41 0 0 0

Coastal-Sardinia 0 23 0 0

East-Liguria 0 0 32 0

Inland-Sardinia 0 0 0 41

North-Apulia 0 0 0 0

Sicily 0 0 0 0

South-Apulia 0 0 0 0

Umbria 0 0 0 0

West-Liguria 0 0 0 0

y

fit North-Apulia Sicily South-Apulia Umbria West-Liguria

Calabria 0 0 0 0 0

Coastal-Sardinia 0 0 0 0 0

East-Liguria 0 0 0 0 0

Inland-Sardinia 0 0 0 0 0

North-Apulia 13 0 0 0 0

Sicily 0 21 0 0 0

South-Apulia 0 0 144 0 0

Umbria 0 0 0 30 0

West-Liguria 0 0 0 0 36

Misclassification Rate = 0

> ypred = predict(area.svm100,newdata=Area.test)

> misclass(ypred,Area.test$Area.name)

Table of Misclassification

(row = predicted, col = actual)

y

fit Calabria Coastal-Sardinia East-Liguria Inland-Sardinia

Calabria 15 0 0 0

Coastal-Sardinia 0 10 0 1

East-Liguria 0 0 15 0

Inland-Sardinia 0 0 0 23

North-Apulia 0 0 0 0

Sicily 0 0 0 0

South-Apulia 0 0 0 0

Umbria 0 0 1 0

West-Liguria 0 0 2 0

y

fit North-Apulia Sicily South-Apulia Umbria West-Liguria

Calabria 0 2 2 0 0

Coastal-Sardinia 0 0 0 0 0

East-Liguria 1 0 0 0 0

Inland-Sardinia 0 0 0 0 0

North-Apulia 9 0 0 0 0

Sicily 0 10 0 0 0

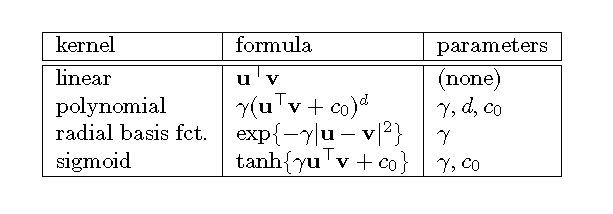
South-Apulia 0 3 60 0 0

Umbria 2 0 0 21 0

West-Liguria 0 0 0 0 14

Misclassification Rate = 0.0733

Kernel options for svm() function with tuning parameters:



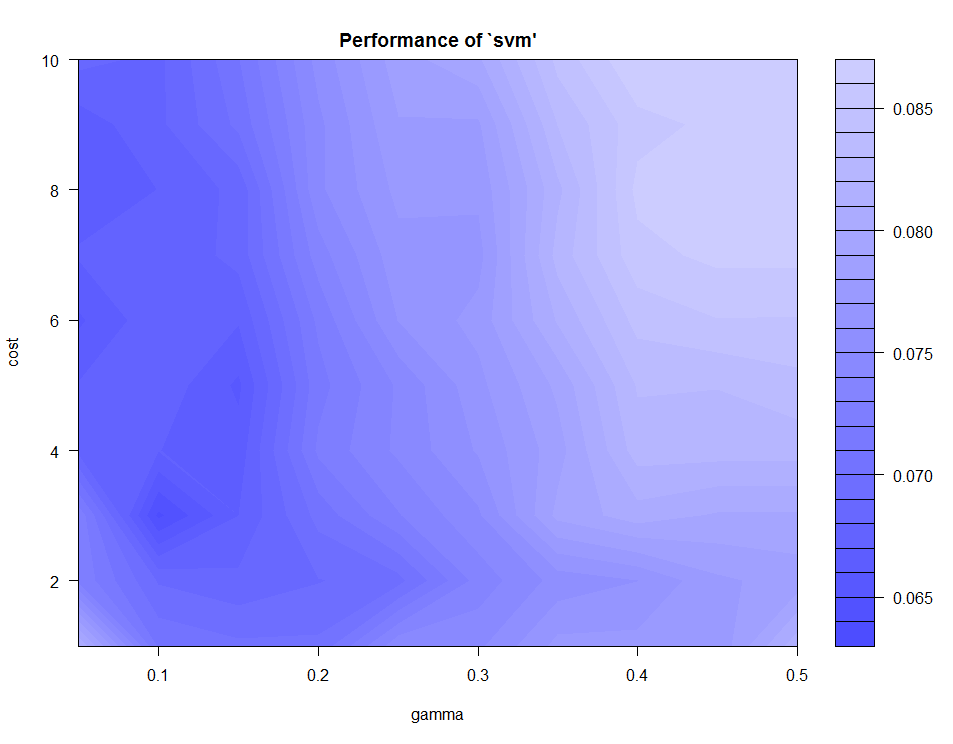
There will be general discussion of the different kernel functions later in the handout.

For svm() you can use the function tune()to find “optimal” choices for the kernel tuning parameters and the cost (C), although you should have some idea for what a reasonable range for the parameters might be.

> area.tune = tune(svm,Area.name~.,data=Area.train,

+ ranges=list(gamma=seq(.05,.5,.05),cost=seq(1,10,1)),  
+ tunecontrol=tune.control(sampling="boot"))

> plot(area.tune)



> area.svmtuned = svm(Area.name~.,data=Area.train,gamma=.1,cost=3)

> misclass(fitted(area.svmtuned),Area.train$Area.name)

Table of Misclassification

(row = predicted, col = actual)

y

fit Calabria Coastal-Sardinia East-Liguria Inland-Sardinia

Calabria 40 0 0 0

Coastal-Sardinia 0 22 0 0

East-Liguria 0 0 32 0

Inland-Sardinia 0 1 0 41

North-Apulia 0 0 0 0

Sicily 0 0 0 0

South-Apulia 1 0 0 0

Umbria 0 0 0 0

West-Liguria 0 0 0 0

y

fit North-Apulia Sicily South-Apulia Umbria West-Liguria

Calabria 0 3 0 0 0

Coastal-Sardinia 0 0 0 0 0

East-Liguria 1 0 0 0 0

Inland-Sardinia 0 0 0 0 0

North-Apulia 12 1 0 0 0

Sicily 0 14 0 0 0

South-Apulia 0 3 144 0 0

Umbria 0 0 0 30 0

West-Liguria 0 0 0 0 36

Misclassification Rate = 0.0262

> ypred = predict(area.svmtuned,newdata=Area.test)

> misclass(ypred,Area.test$Area.name)

Table of Misclassification

(row = predicted, col = actual)

y

fit Calabria Coastal-Sardinia East-Liguria Inland-Sardinia

Calabria 14 0 0 0

Coastal-Sardinia 0 10 0 1

East-Liguria 0 0 15 0

Inland-Sardinia 0 0 0 23

North-Apulia 0 0 0 0

Sicily 0 0 0 0

South-Apulia 1 0 0 0

Umbria 0 0 1 0

West-Liguria 0 0 2 0

y

fit North-Apulia Sicily South-Apulia Umbria West-Liguria

Calabria 0 4 0 0 0

Coastal-Sardinia 0 0 0 0 0

East-Liguria 4 0 0 0 0

Inland-Sardinia 0 0 0 0 0

North-Apulia 6 0 0 0 0

Sicily 0 9 0 0 0

South-Apulia 0 2 62 0 0

Umbria 2 0 0 21 0

West-Liguria 0 0 0 0 14

Misclassification Rate = 0.089

You could also easily code a MCCV function that would allow you to try different combinations of tuning parameters and summarize the results as shown previously in the notes.

svm() also has a built-in k-fold cross-validation option as demonstrated below.

> area.svmcv = svm(Area.name~.,data=Area.train,cross=5)

> summary(area.svmcv)

Call:

svm(formula = Area.name ~ ., data = Area.train, cross = 5)

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 1

gamma: 0.1428571

Number of Support Vectors: 183

( 35 20 24 20 13 25 12 21 13 )

Number of Classes: 9

Levels:

Calabria Coastal-Sardinia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria

5-fold cross-validation on training data:

Total Accuracy: 92.38845

Single Accuracies:

93.42105 90.78947 92.10526 96.05263 89.61039

> area.svmcv = svm(Area.name~.,data=Area.train,cross=5,gamma=.1,cost=3)

> summary(area.svmcv)

Call:

svm(formula = Area.name ~ ., data = Area.train, cross = 5, gamma = 0.1,

cost = 3)

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 3

gamma: 0.1

Number of Support Vectors: 158

( 26 17 22 16 13 21 10 20 13 )

Number of Classes: 9

Levels:

Calabria Coastal-Sardinia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria

5-fold cross-validation on training data:

Total Accuracy: 94.48819

Single Accuracies:

90.78947 93.42105 97.36842 94.73684 96.1039

> area.svmcv = svm(Area.name~.,data=Area.train,cross=5,gamma=.1,cost=10)

> summary(area.svmcv)

Call:

svm(formula = Area.name ~ ., data = Area.train, cross = 5, gamma = 0.1,

cost = 10)

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 10

gamma: 0.1

Number of Support Vectors: 141

( 25 13 22 14 9 19 9 17 13 )

Number of Classes: 9

Levels:

Calabria Coastal-Sardinia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria

5-fold cross-validation on training data:

Total Accuracy: 95.01312

Single Accuracies:

93.42105 94.73684 94.73684 98.68421 93.50649

> area.svmcvopt = svm(Area.name~.,data=Area.train,gamma=.1,cost=10)

> ypred = predict(area.svmcvopt,newdata=Area.test)

> misclass(ypred,Area.test$Area.name)

Table of Misclassification

(row = predicted, col = actual)

y

fit Calabria Coastal-Sardinia East-Liguria Inland-Sardinia

Calabria 15 0 0 0

Coastal-Sardinia 0 10 0 1

East-Liguria 0 0 15 0

Inland-Sardinia 0 0 0 23

North-Apulia 0 0 0 0

Sicily 0 0 0 0

South-Apulia 0 0 0 0

Umbria 0 0 1 0

West-Liguria 0 0 2 0

y

fit North-Apulia Sicily South-Apulia Umbria West-Liguria

Calabria 0 3 0 0 0

Coastal-Sardinia 0 0 0 0 0

East-Liguria 1 0 0 0 0

Inland-Sardinia 0 0 0 0 0

North-Apulia 9 0 0 0 0

Sicily 0 10 0 0 0

South-Apulia 0 2 62 0 0

Umbria 2 0 0 21 0

West-Liguria 0 0 0 0 14

Misclassification Rate = 0.0628 🡨 Best yet?!?

As the default kernel is the radial basis function we can could consider other kernel options for these data.

> area.svmpoly = svm(Area.name~.,data=Area.train,kernel="polynomial",cross=5)

> summary(area.svmpoly)

Call:

svm(formula = Area.name ~ ., data = Area.train, kernel = "polynomial", cross = 5)

Parameters:

SVM-Type: C-classification

SVM-Kernel: polynomial

cost: 1

degree: 3

gamma: 0.1428571

coef.0: 0

Number of Support Vectors: 195

( 45 35 25 6 16 31 9 18 10 )

Number of Classes: 9

Levels:

Calabria Coastal-Sardinia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria

5-fold cross-validation on training data:

Total Accuracy: 84.51444

Single Accuracies:

88.15789 77.63158 92.10526 84.21053 80.51948

> area.svmpoly = svm(Area.name~.,data=Area.train,kernel="polynomial",cross=5,coef0=1)

> summary(area.svmpoly)

Call:

svm(formula = Area.name ~ ., data = Area.train, kernel = "polynomial", cross = 5,

coef0 = 1)

Parameters:

SVM-Type: C-classification

SVM-Kernel: polynomial

cost: 1

degree: 3

gamma: 0.1428571

coef.0: 1

Number of Support Vectors: 114

( 19 14 20 6 8 17 6 15 9 )

Number of Classes: 9

Levels:

Calabria Coastal-Sardinia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria

5-fold cross-validation on training data:

Total Accuracy: 94.75066

Single Accuracies:

92.10526 94.73684 98.68421 89.47368 98.7013

> area.svmpoly = svm(Area.name~.,data=Area.train,kernel="polynomial",cross=5,coef0=1)

> misclass(fitted(area.svmpoly),Area.train$Area.name)

Misclassification Rate = 0.021

> ypred = predict(area.svmpoly,newdata=Area.test)

> misclass(ypred,Area.test$Area.name)

Misclassification Rate = 0.0785

The polynomial kernel does not appear to work quite as well for these data. I did tweak the degree as well but it did not improve performance. Trying sigmoid (hyperbolic tangent) kernel did not produce very good results at all (<90% accuracy). Finally we try linear, i.e. no kernel applied.

> area.svmlin = svm(Area.name~.,data=Area.train,kernel="linear",cross=5)

> summary(area.svmlin)

Call:

svm(formula = Area.name ~ ., data = Area.train, kernel = "linear", cross = 5)

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 1

gamma: 0.1428571

Number of Support Vectors: 127

( 16 12 23 10 8 21 8 18 11 )

Number of Classes: 9

Levels:

Calabria Coastal-Sardinia East-Liguria Inland-Sardinia North-Apulia Sicily South-Apulia Umbria West-Liguria

5-fold cross-validation on training data:

Total Accuracy: 95.01312

Single Accuracies:

93.42105 93.42105 97.36842 92.10526 98.7013

> misclass(fitted(area.svmlin),Area.train$Area.name)

Misclassification Rate = 0.0341

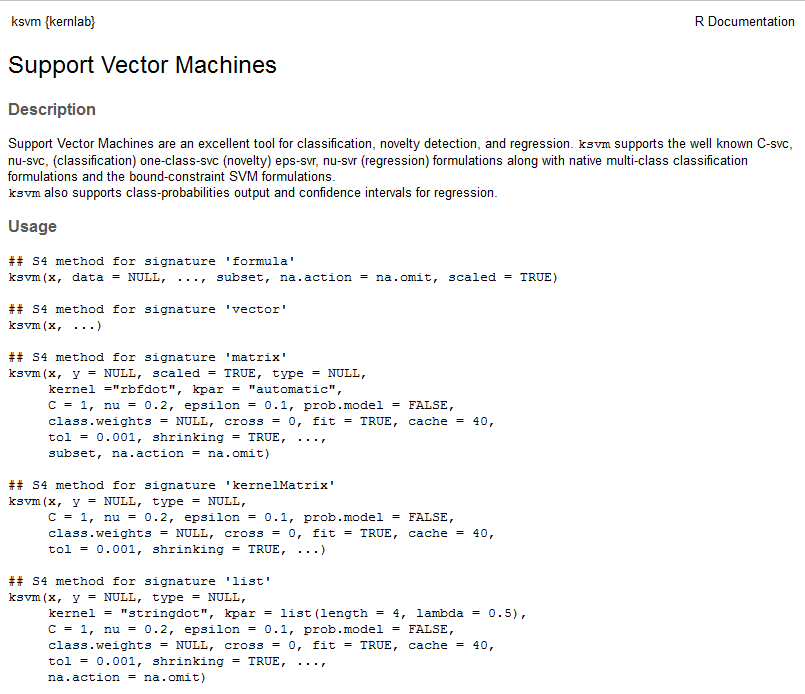
> ypred = predict(area.svmlin,newdata=Area.test)  
> misclass(ypred,Area.test$Area.name)

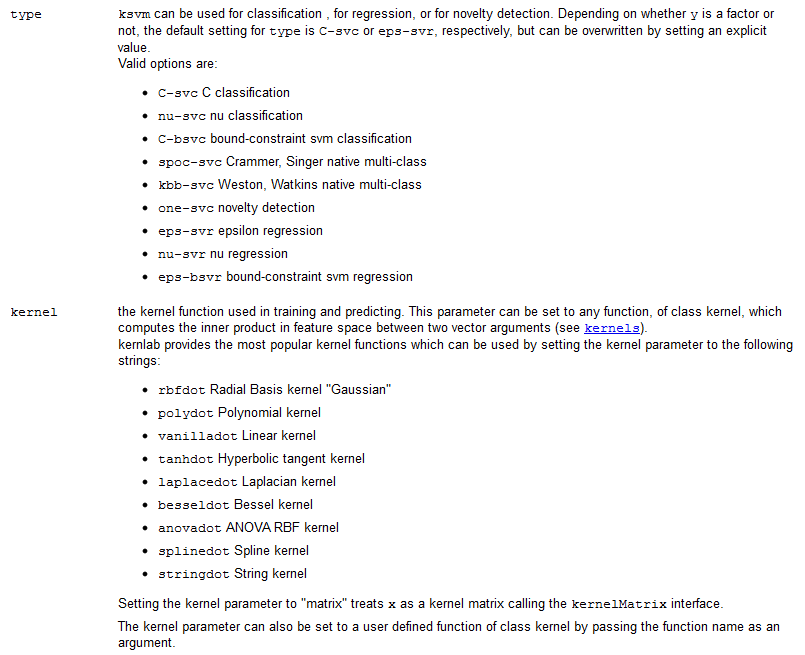
Misclassification Rate = 0.0785

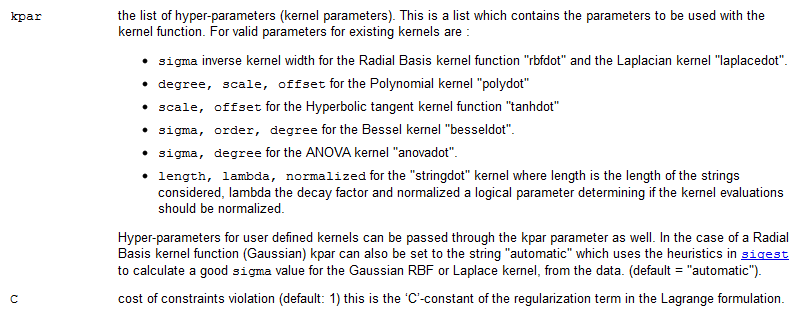
Linear (i.e. no nonlinear kernel applied) did a reasonable job for these data, but it did not outperform the tuned radial basis kernel for these data in predicting the test cases. There is a reason the radial basis kernel is the default.

The kernlab library also has an implementation of SVM with a wider array of kernel options via the function ksvm().

**ksvm() from kernlab**







ksvm()also has built in k-fold cross-validation functionality so you can obtain a better estimate of the misclassification error rate.  
  
> area.ksvm = ksvm(Area.name~.,data=Area.train,cross=5)

> summary(area.ksvm) # There isn’t a summary feature for ksvm models.

Length Class Mode

1 ksvm S4

> area.ksvm # Type the name of the ksvm object to view performance.

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 1

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.194360024599412

Number of Support Vectors : 188

Training error : 0.031414

Cross validation error : 0.052392   
  
Besides considering other kernel functions we can tune the parameter (C) which increases the penalty for misclassified observations in the training data. Increasing C should lower the training error, however it will likely increase the validation error. It is possible that a happy medium value for C exists.

> area.ksvm = ksvm(Area.name~.,data=Area.train,cross=5,C=5)

> area.ksvm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 5

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.211435327129462

Number of Support Vectors : 164

Training error : 0.010471

Cross validation error : 0.065414

> area.ksvm = ksvm(Area.name~.,data=Area.train,cross=5,C=3)

> area.ksvm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 3

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.164066633331375

Number of Support Vectors : 161

Training error : 0.018325

Cross validation error : 0.047163

> area.ksvm = ksvm(Area.name~.,data=Area.train,cross=5,C=2)

> area.ksvm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 2

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.255097950033082

Number of Support Vectors : 173

Training error : 0.020942

Cross validation error : 0.044429

We will now consider predicting the test cases.

> ypred = predict(area.ksvm,newdata=Area.test,type="response")

> misclass(ypred,Area.test$Area.names)

Misclassification Rate = 0.0632

We could try other kernel functions besides the radial basis function. I tried vanilladot (not transformation), polydot, laplacedot, and tanhdot with different hyperparameter settings and found nothing better than the radial basis kernel.

There are several SVM methods (differing in their formulation of the objective function) developed specifically for multi-class problems. They can be chosen by using the type option.

* C-svc C classification 🡨 default method for classification problems
* nu-svc nu classification
* C-bsvc bound-constraint svm classification

These could be used in place of the default method for this 9 class problem.

* spoc-svc Crammer, Singer native multi-class \*
* kbb-svc Weston, Watkins native multi-class \*

Using multi-class methods we obtain…

Crammer-Singer

> area.spoc = ksvm(Area.name~.,data=Area.train,cross=5,type="spoc-svc")

> area.spoc

Support Vector Machine object of class "ksvm"

SV type: spoc-svc (classification)

parameter : cost C = 1

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.17384108072842

Number of Support Vectors : 133

Training error : 0.026178

Cross validation error : 0.078674

> area.spoc = ksvm(Area.name~.,data=Area.train,cross=5,type="spoc-svc",C=5)

> area.spoc

Support Vector Machine object of class "ksvm"

**SV type: spoc-svc** (classification)

parameter : cost C = 5

**Gaussian Radial Basis kernel function.**

**Hyperparameter : sigma = 0.24283805422656**

Number of Support Vectors : 117

Objective Function Value : 40

**Training error : 0.007853**

**Cross validation error : 0.060185**

> area.spoc = ksvm(Area.name~.,data=Area.train,cross=5,type="spoc-svc",C=3)

> area.spoc

Support Vector Machine object of class "ksvm"

**SV type: spoc-svc** (classification)

parameter : cost C = 3

**Gaussian Radial Basis kernel function.**

**Hyperparameter : sigma = 0.235987366210243**

Number of Support Vectors : 125

Objective Function Value : 0

**Training error : 0.013089**

**Cross validation error : 0.049658**

Predicting the test cases

> ypred = predict(area.spoc,newdata=Area.test,type="response")

> misclass(ypred,Area.test$Area.name)

Misclassification Rate = 0.0684

Weston-Watkins

> area.kbb = ksvm(Area.name~.,data=Area.train,cross=5,type="kbb-svc")

> area.kbb

Support Vector Machine object of class "ksvm"

SV type: kbb-svc (classification)

parameter : cost C = 1

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.245715879003861

Number of Support Vectors : 144

Objective Function Value : 0

Training error : 0.020942

Cross validation error : 0.05499

> area.kbb = ksvm(Area.name~.,data=Area.train,cross=5,type="kbb-svc",C=3)

> area.kbb

Support Vector Machine object of class "ksvm"

SV type: kbb-svc (classification)

parameter : cost C = 3

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.220430593684119

Number of Support Vectors : 122

Objective Function Value : -14.4714

Training error : 0.018325

Cross validation error : 0.047095

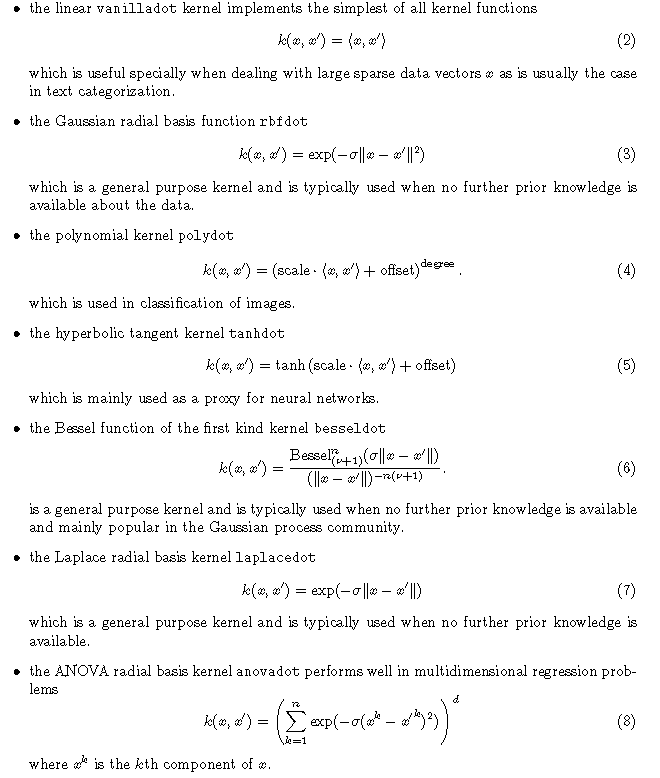
Predicting the test cases…

> ypred = predict(area.kbb,newdata=Area.test,type="response")

> misclass(ypred,Area.test$Area.name)

Misclassification Rate = 0.0632

As we have seen the choice of kernel is a “tuning parameter”. This is true for both svm() and ksvm(), although ksvm has more kernel options. In most of the above the default Gaussian Radial Basis Function kernel was used(rbfdot). The general formulae for the different kernels and guidelines for their use are given on the following page. As you can also see some of the kernel functions have additional tuning parameters (that can be supplied depending on which is used.

  
 (taken from *Support Vector Machines in R* by Karatzoglou, Meyer, and Hornick, 2006).

**Example 18.2 – Breast Cancer Data**

> dim(BreastDiag)

[1] 569 32

> names(BreastDiag)

[1] "Id" "Diagnosis" "Radius" "Texture" "Perimeter" "Area"

[7] "Smoothness" "Compactness" "Concavity" "ConcavePts" "Symmetry" "FracDim"

[13] "serad" "setex" "seperi" "searea" "sesmoo" "secomp"

[19] "seconc" "seconpts" "sesym" "sefd" "wrad" "wtex"

[25] "wperi" "warea" "wsmoo" "wcomp" "wconc" "wconpts"

[31] "wsym" "wfd"

> BreastDiag = BreastDiag[,-1]

> set.seed(1)

> sam = sample(1:569,ceiling(569\*.6666),replace=F)

> BC.train = BreastDiag[sam,]

> BC.test = BreastDiag[-sam,]

> bc.svm = ksvm(Diagnosis~.,data=BC.train,cross=10,prob.model=T)

> bc.svm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 1

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.0495511440771743

Number of Support Vectors : 112

Objective Function Value : -44.5398

Training error : 0.010526

Cross validation error : 0.026316

Probability model included.

> ypred = predict(bc.svm,newdata=BC.test,type="response")

> misclass(ypred,BC.test$Diagnosis)

Table of Misclassification

(row = predicted, col = actual)

y

fit B M

B 115 4

M 1 69

Misclassification Rate = 0.0265

> yprob = predict(bc.svm,newdata=BC.test,type="probabilities")  
> head(yprob)

B M

[1,] 0.0660065036 0.9339935

[2,] 0.0024935210 0.9975065

[3,] 0.0006730814 0.9993269

[4,] 0.2445472404 0.7554528

[5,] 0.0098442228 0.9901558

[6,] 0.0001829446 0.9998171

> bc.svm = ksvm(Diagnosis~.,data=BC.train,cross=10,prob.model=T,C=10)

> bc.svm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 10

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.0394561334853963

Number of Support Vectors : 81

Objective Function Value : -125.3412

Training error : 0.005263

Cross validation error : 0.028947

Probability model included.

> ypred = predict(bc.svm,newdata=BC.test,type="response")

> misclass(ypred,BC.test$Diagnosis)

Table of Misclassification

(row = predicted, col = actual)

y

fit B M

B 116 3

M 0 70

Misclassification Rate = 0.0159

**Example 18.3 – Classifying Music Genre**

> Genre.train = read.table(file.choose(),header=T,sep=",")

> dim(Genre.train)

[1] 10000 192

> sam = sample(1:10000,6000,replace=F)

> G.train = Genre.train[sam,]

> G.test = Genre.train[-sam,]

> genre.svm = ksvm(GENRE~.,data=G.train,cross=5)

> genre.svm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 1

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.00360831969340755

Number of Support Vectors : 2796

Objective Function Value : -88.3549 -118.0005 -102.3189 -93.1794 -145.919 -594.9859 -48.8332 -63.4586 -96.8279 -81.0202 -112.8308 -252.329 -84.2885 -234.9993 -182.6058

Training error : 0.029167

Cross validation error : 0.0675

> genre.svm = ksvm(GENRE~.,data=G.train,cross=5,C=3)

> genre.svm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 3

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.00352787774964907

Number of Support Vectors : 2516

Objective Function Value : -104.3883 -137.0583 -145.6637 -117.6832 -189.8191 -983.2368 -58.257 -70.9524 -127.4441 -99.6824 -126.9137 -411.1709 -114.3854 -382.5273 -278.1773

Training error : 0.006333

Cross validation error : 0.046667

> genre.svm = ksvm(GENRE~.,data=G.train,cross=5,C=5)

> genre.svm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 5

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.00366260466795278

Number of Support Vectors : 2476

Objective Function Value : -104.4585 -136.7267 -149.7259 -120.0498 -194.0999 -1109.912 -58.2991 -72.3169 -130.0716 -99.074 -127.7852 -462.2354 -118.6181 -439.2703 -294.8357

Training error : 0.001833

Cross validation error : 0.042667

> genre.svm = ksvm(GENRE~.,data=G.train,cross=5,C=7)

> genre.svm

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 7

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.00358615785261152

Number of Support Vectors : 2424

Objective Function Value : -106.3836 -138.776 -153.8409 -123.2675 -200.3872 -1216.744 -58.9344 -72.9406 -133.1255 -100.1456 -128.9627 -505.3002 -122.3717 -488.0872 -309.907

Training error : 0.001167

Cross validation error : 0.0425

> ypred = predict(genre.svm,newdata=G.test,type="response")

> misclass(ypred,G.test$GENRE)

Table of Misclassification

(row = predicted, col = actual)

y

fit Blues Classical Jazz Metal Pop Rock

Blues 499 0 0 4 4 1

Classical 0 1084 41 0 2 2

Jazz 3 32 940 8 6 16

Metal 0 0 0 277 1 5

Pop 1 0 1 4 457 13

Rock 0 0 7 7 10 575

Misclassification Rate = 0.042 🡨 this is better than best attained so far.

**Example 18.4 – ZIP Code Digit Recognition**

> library(ElemStatLearn)

> ZIP.train = data.frame(zip.train)

> ZIP.test = data.frame(zip.test)

> ZIP.train = data.frame(Digit=as.factor(ZIP.train$X1),ZIP.train[,-1])

> ZIP.test = data.frame(Digit=as.factor(ZIP.test$X1),ZIP.test[,-1])

> dim(ZIP.train)

[1] 7291 257

> set.seed(1)

> sam = sample(1:7291,5000,replace=F)

> Z.train = ZIP.train[sam,]

> Z.test = ZIP.train[-sam,]

> digit.rbf = ksvm(Digit~.,data=Z.train,kpar="automatic",cross=5)

> digit.rbf

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 1

Gaussian Radial Basis kernel function (rbf)

Hyperparameter : sigma = 0.00273765832179067

Number of Support Vectors : 1930

Training error : 0.0102

Cross validation error : 0.0306

> digit.rbf = ksvm(Digit~.,data=Z.train,kpar="automatic",cross=5,C=5)

> digit.rbf

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 5

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.00266572640137537

Number of Support Vectors : 1763

Training error : 0.001

Cross validation error : 0.0256

> digit.rbf = ksvm(Digit~.,data=Z.train,kpar="automatic",cross=5,C=10)

> digit.rbf

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 10

Gaussian Radial Basis kernel function.

Hyperparameter : sigma = 0.00272106631451962

Number of Support Vectors : 1768

Training error : 4e-04

Cross validation error : 0.0268

For image recognition problems, i.e. where the are pixel information for an image, it has been shown that the polynomial kernel function (polydot in ksvm) is a good kernel choice.

In the ksvm function these tuning parameters ( are referred to as the degree, scale, and offset respectively. They can set in the call to ksvm function by specifying them in the form of *list* using the kpar option as shown in the examples below.

> digit.poly1 = ksvm(Digit~.,data=Z.train,cross=5,C=1,kernel="polydot",  
kpar=list(degree=2,scale=1,offset=1))

> digit.poly1

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 1

Polynomial kernel function.

Hyperparameters : degree = 2 scale = 1 offset = 1

Number of Support Vectors : 1921

Training error : 0

Cross validation error : 0.0306

> digit.poly1 = ksvm(Digit~.,data=Z.train,cross=5,C=1,kernel="polydot",

kpar=list(degree=2,scale=.1,offset=1))

> digit.poly1

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 1

Polynomial kernel function.

Hyperparameters : degree = 2 scale = 0.1 offset = 1

Number of Support Vectors : 1810

Training error : 0

Cross validation error : 0.0242

> digit.poly1 = ksvm(Digit~.,data=Z.train,cross=5,C=3,kernel="polydot",  
kpar=list(degree=2,scale=.1,offset=1))

> digit.poly1

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 3

Polynomial kernel function.

Hyperparameters : degree = 2 scale = 0.1 offset = 1

Number of Support Vectors : 1809

Training error : 0

Cross validation error : 0.0236

> digit.poly1 = ksvm(Digit~.,data=Z.train,cross=5,C=5,kernel="polydot",kpar=list(degree=2,scale=.1,offset=1))

> digit.poly1

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 5

Polynomial kernel function.

Hyperparameters : degree = 2 scale = 0.1 offset = 1

Number of Support Vectors : 1809

Training error : 0

Cross validation error : 0.0254

> digit.poly1 = ksvm(Digit~.,data=Z.train,cross=5,C=5,kernel="polydot",

kpar=list(degree=3,scale=.1,offset=1))  
> digit.poly1

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 5

Polynomial kernel function.

Hyperparameters : degree = 3 scale = 0.1 offset = 1

Number of Support Vectors : 2024

Training error : 0

Cross validation error : 0.022

> digit.poly1 = ksvm(Digit~.,data=Z.train,cross=5,C=3,kernel="polydot",

kpar=list(degree=3,scale=.1,offset=1))

> digit.poly1

Support Vector Machine object of class "ksvm"

SV type: C-svc (classification)

parameter : cost C = 3

Polynomial kernel function.

Hyperparameters : degree = 3 scale = 0.1 offset = 1

Number of Support Vectors : 2024

Training error : 0

Cross validation error : 0.0242

Using the SVM fit from settings in the boxed output above we obtain the following prediction error on the test cases.

> ypred = predict(digit.poly1,newdata=Z.test,type="response")

> misclass(ypred,Z.test$Digit)

Table of Misclassification

(row = predicted, col = actual)

y

fit 0 1 2 3 4 5 6 7 8 9

0 360 0 0 0 0 1 2 0 0 1

1 0 335 0 0 0 0 0 0 2 0

2 3 0 212 3 1 2 0 0 0 0

3 1 0 1 186 0 5 0 0 2 2

4 1 0 1 0 202 0 0 1 1 1

5 2 0 0 4 0 166 0 0 5 0

6 0 0 0 0 0 0 214 0 0 0

7 0 1 0 0 0 0 0 192 1 5

8 0 0 1 0 0 0 0 2 161 0

9 0 0 0 0 0 0 0 2 0 209

Misclassification Rate = 0.0236

> digit.rf = randomForest(Digit~.,data=Z.train)

> ypred = predict(digit.rf,newdata=Z.test)  
> misclass(ypred,Z.test$Digit)

Table of Misclassification

(row = predicted, col = actual)

y

fit 0 1 2 3 4 5 6 7 8 9

0 362 0 0 0 1 6 1 0 1 0

1 0 334 0 0 0 0 0 0 2 0

2 3 0 210 3 3 3 1 1 0 1tab

3 0 0 1 181 0 4 0 0 1 0

4 1 0 2 0 199 0 0 3 3 5

5 1 0 0 8 0 157 0 0 3 0

6 0 1 0 0 0 2 213 0 0 0

7 0 1 0 0 0 0 0 187 2 4

8 0 0 2 0 0 1 1 2 160 1

9 0 0 0 1 0 1 0 4 0 207

Misclassification Rate = 0.0354