STATISTICS 452/652: Statistical Learning and Prediction

November 30, 2020

Lecture 22: Support Vector Machines

(Reading: Chapter 9)

1 Goals of lecture

1. Learn about support vector machines (SVM), a technique developed specifically for classification
   (a) How to draw optimal linear decision boundaries when classes are separable
   (b) How to adapt linear boundary when classes are not linearly separable
   (c) How to adapt to cases when a nonlinear boundary is needed by expanding the dimension of the problem
      i. Not as weird as it sounds
2. Works in 2 classes only
   (a) Can run "round robin" competition to adapt to $K > 2$ classes.

2 Separating Hyperplanes

Before we get into the main topic, we need to look at a simpler problem first. Then we will extend this problem several times into the solution that works best in general.

We assume that we have $K = 2$ classes throughout, until the very end when we extend it to larger K.

2.1 Separation when $p = 1$, $K = 2$

- Consider a single explanatory variable X_1
- Suppose that there are 5 observed responses from class 1, at X_1 =1,2,3,4,and 5

- Suppose there are also 3 observed responses from class 2, at X_1 =7, 8, and 9.
- This is an example where the classes exhibit *complete separation* with respect to X_1, since there is no overlap ion the regions of X_1occupied by the two classes.
- It is easy to draw a decision boundary between the two classes
  - Any point between 5 and 7 will work
  - Which one is best?
- Imagine using this on future data, that might be sampled beyond the current range of the data in the two classes
  - It makes sense to place the boundary so that it is simultaneously as far from each current class as possible.
  - In our example, that point would be at $X_1 = 6$
  - Refer to this as the OPTIMAL SEPARATING HYPERPLANE
    - A $p-1$-dimensional "hyperplane" is a 0-dimensional point when $p = 1$
  - The gap between the boundaries is called the MARGIN
- Notice that the exact locations of only the two points on the respective class borders are needed in order to determine the optimal separating point
  - In this case at 5 and 7
  - These two points are called the SUPPORT VECTORS for the optimal separating hyperplane
- In general, you only need to figure out which points are support vectors in order to figure out where the optimal separating hyperplane is.
- Finally, we can imagine a function $f(X)$ with the following properties
  - $f(X) = 1$ on the edge of the margin on the Class 1 side
  - $f(X) = -1$ on the edge of the margin on the Class 2 side
- Then $f(X) = 0$ at the decision boundary
  - In our example $f(X) = 6 - X_1$ does this
    - The value of X_1 where $f(X) = 6 - X_1 = 0$, which is $X_1 = 6$, is the decision boundary
  - Classify a new value x_0 according to the sign of $f(x_0)$
    - $f(x_0) > 0$ then class 1
    - $f(x_0) < 0$ then class 2

Figure 1: 2-dimensional X with two classes that are completely separated. Left: Showing that there are many different lines (separating hyperplanes) that *could* be used as decision boundaries. Right: How classification looks for one of those lines. From ISLR

340 9. Support Vector Machines

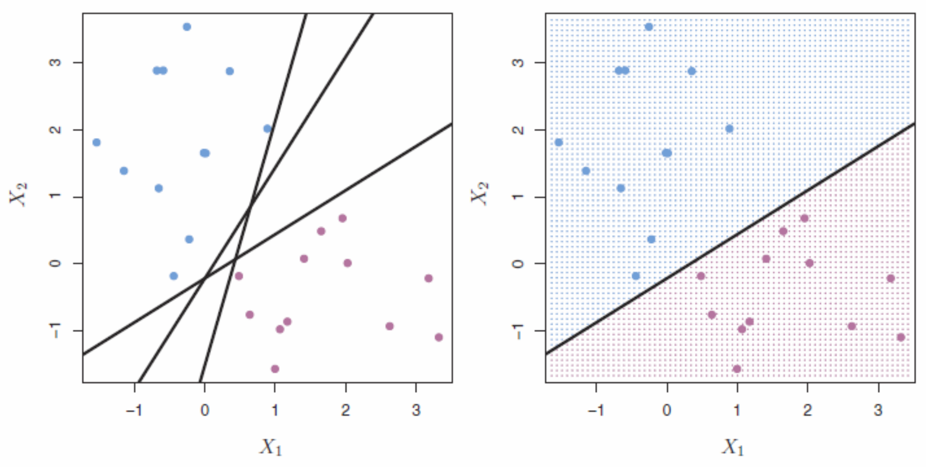


FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

2.2 Separation when $p = 2$

- Now consider two variables, X_1 and X_2, and suppose the classes are again completely separated
- See Figure 1 for the example, and apply the same principles:
  - Now a separating hyperplane is $p - 1 = 1$ dimension—a line
  - There are many different lines that can be drawn that separate the classes
  - Different points may be considered as support vectors for different lines.
  - Different lines create different margins
  - Want to find the *optimum* separating hyperplane, *which is the one that creates the largest margin*
    - Represents the decision rule that separates the classes the most
- The book shows some of the equations. Look if you want. For the rest of us:
  - <...wave a magic wand called mathematics...>

Figure 2: Optimal separating hyperplane for the example from Figure 1

342 9. Support Vector Machines

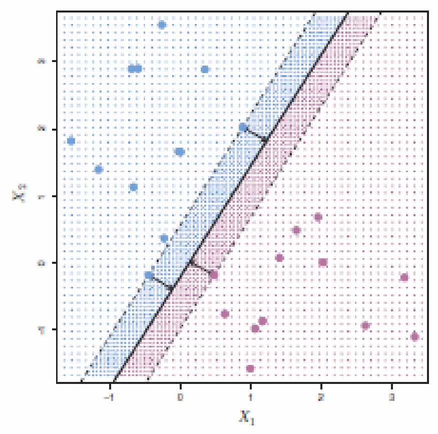


FIGURE 9.3. *There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the margin is indicated by arrows. The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.*

- See Figure 2 for the solution, also called the MAXIMAL MARGIN HYPERPLANE
- Once again, we can define a function $f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$ that is a line running perpendicular to the optimal separating hyperplane where
  - $f(X) = 1$ on the edge of the margin on the Class 1 side
  - $f(X) = -1$ on the edge of the margin on the Class 2 side
- Then $f(X) = 0$ at the decision boundary
- Once again, we need to know the exact locations only of points that are on the margins.
  - These are the support vectors.
  - For these points, $f(X) = 1$
  - For all other points $f(X) > 1$ for Class 1 and < 1 for Class 2
- Classify a new value x_0 according to the sign of $f(x_0)$
  - $f(x_0) > 0$ then class 1
  - $f(x_0) < 0$ then class 2

2.3 Higher dimensions

When X is of higher dimensions, and when classes are completely separated, the same principles apply.

Figure 3:

344 9. Support Vector Machines

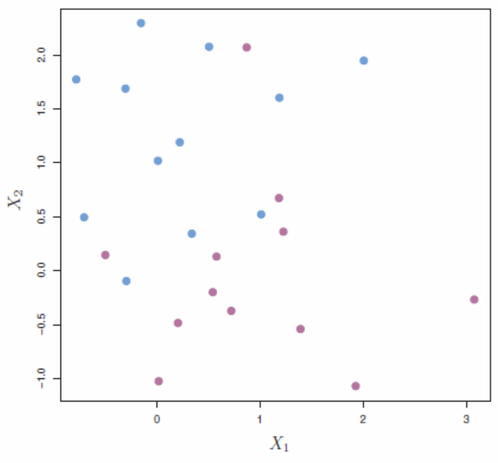


FIGURE 9.4. *There are two classes of observations, shown in blue and in purple. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.*

- Look for the optimal separating hyperplane that maximizes the margin
  - Factoid: it is a $p-1$ dimensional thing; e.g., a plane in 3-dims
- It is the solution to the equation $f(X) = 0$, where
  - $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$
  - $f(X) = \pm 1$ at the support vectors,
  - $|f(X)| > 1$ for all other points
- Only the exact locations of the support vectors matter
- Classify a new value x_0 according to the sign of $f(x_0)$
  - $f(x_0) > 0$ then class 1
  - $f(x_0) < 0$ then class 2

3 Non-Separable Case: Linear Support Vector Machines

Separability is rare. Need something to solve more common problem where there is overlap among classes. See Figure 3 for an example.

- Approach is to continue to work with MAXIMAL MARGIN HYPERPLANES, but to introduce correction for overlap

Figure 4:

344 9. Support Vector Machines

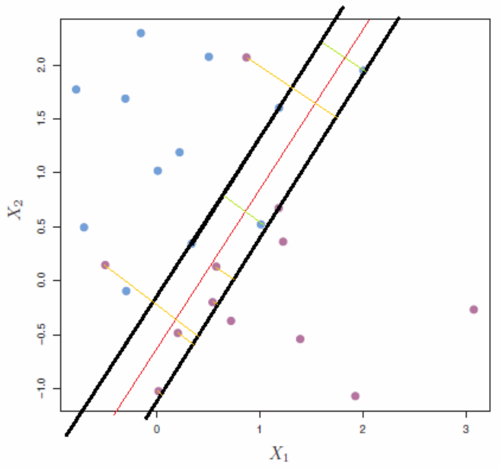


FIGURE 9.4. *There are two classes of observations, shown in blue and in purple. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.*

- To see how we do this, start by imagining any line you might use as a decision boundary
  - For all misclassified cases, measure how far they are from the boundary
    - Add up these distances
    - It seems reasonable that we want to choose a line that makes this total small.
- More specifically, imagine creating a decision boundary based on identifying some support vectors of the "correct" class on each side of the line.
  - Again define $f(X)$ as above so that it is 0 at the boundary and ± 1 at the margins where the support vectors lie
  - For each observation, define its SLACK to be the distance that you have to move that observation in order to put it on the correct side of the margin
    - Let ϵ_i denote the slack for observation $i = 1, \dots, n$.
    - Observations that are on the correct side of the margin need no slack, $\epsilon_i = 0$
    - Observations that are on in the margin but still correctly classified need $0 < \epsilon_i < 1$
    - See Figure 4
- We want to create a line that simultaneously
  - Maximizes the large margin
  - Keeps $\sum_{i=1}^{n} \epsilon_i$ small
- There is no single solution

  - For any total slack, $\sum_{i=1}^{n} \epsilon_i$, there would be a different optimal solution

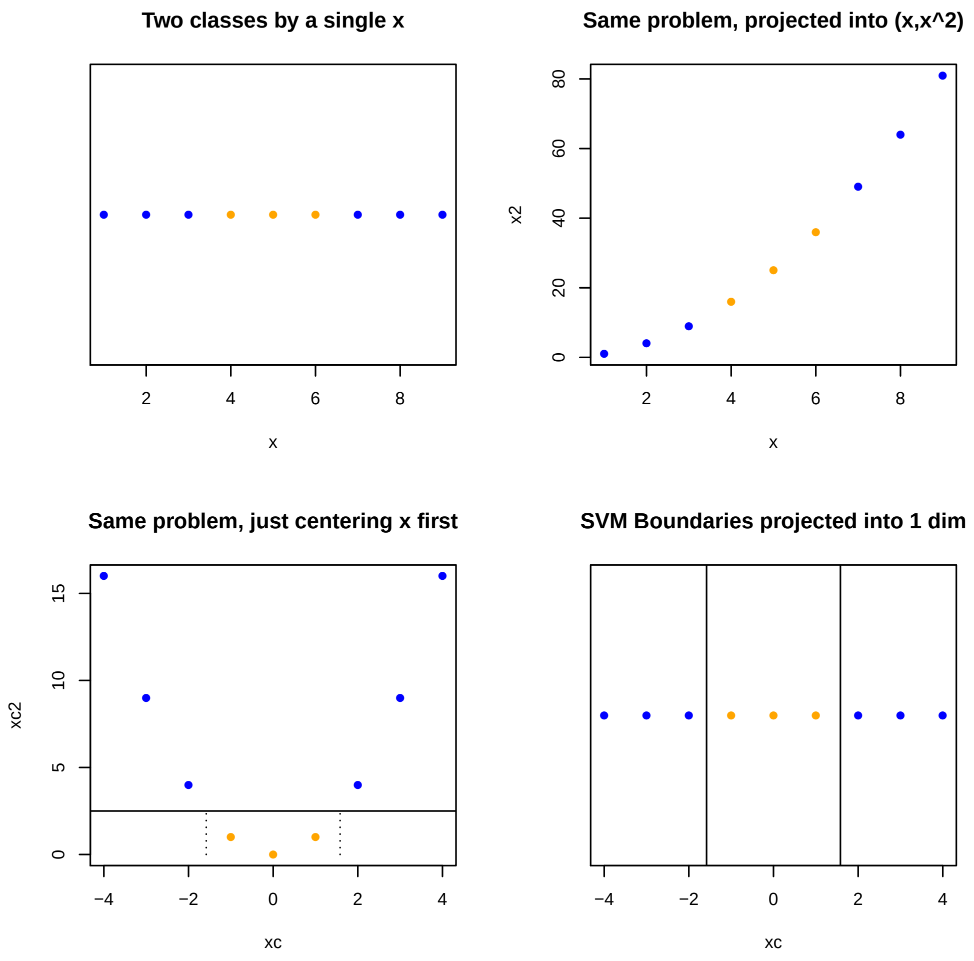
- This is similar to LASSO
  - We minimized RSS while keeping $\sum_{j=1}^{p} |\hat{\beta}_j|$ small
  - We solved that by treating $\sum_{j=1}^{p} |\hat{\beta}_j|$ as a "penalty"
  - Combining RSS and penalty one equation with a weight λ on the penalty term
  - Tune λ to optimize prediction error
- We do the same thing here!
  - Combine equation for maximizing margin with penalty for slack
  - Add a penalty parameter
    - C is the "cost": it is a regularization parameter
    - Higher cost means less slack is tolerated, margins are narrower, will explain importance later
- Result is a linear boundary.
  - The resulting classifier is called a LINEAR SUPPORT VECTOR MACHINE (SVM)
  - *Another* way to build a linear classifier;
    - Nonparametric, which is nice
    - But not yet what we want

4 Nonlinear Support Vector Machines

We finally get to the problem that we really want to solve: creating a flexible, nonlinear, model-free classifier

- Recall how we made linear regression into nonlinear regression through **basis expansion**
  - Turned x into a set of basis functions $h_1(x), h_2(x), \ldots$
    - For example polynomial terms or hinged linear terms knotted at points x_i
    - Potentially *very* large set of functions (an *expansion*).
  - Created linear combination of basis functions, $f(x) = \beta_0 + \sum_l \beta_l h_l(x)$
    - f is a linear function in the high-dimensional space defined by the collection of $h_l(x)$'s
    - Projects back as a nonlinear shape in original space
  - Estimated parameters with shrinkage
- We do the same thing here:

Figure 5: Simple example showing how to expand X space using basis functions, find a separating hyperplane, and project the solution back into original dimensions. Here, the picture is clearer if we center x first, which changes nothing with regard to the conceptual problem.



- Transform variables in X into a new set $\mathcal{H}$ of (much) higher dimension (http://www.youtube.com
- Example: $x = 1, 2, ..., 9$; $y_i = 1$ for $i = 4, 5, 6$; $y_i = 2$ for $i = 1, 2, 3, 7, 8, 9$.
  - Make basis functions $h_1(x) = x$, $h_2(x) = x^2$
  - Consider plot in 2 dimensions: easy to find separating hyperplane!
    - Project solution back into original dimension. See Figure 5.

- More generally, create a bunch of basis functions $\mathcal{H}$ of dimension $> p$.
  - e.g., if $p = 2$, so that $X = (X_1, X_2)$, then $\mathcal{H} = (1, X_1, X_2, X_1^2, X_2^2, X_1X_2)$ is a quadratic expansion of the space X into 6 dimensions
- Make dimension sufficiently large that a *separating* hyperplane exists and we can apply the linear SVM.

  - Find the function $f(X) = \beta_0 + \sum_l \beta_l h_l(X)$ that satisfies the usual properties
    - $f(X) = \pm 1$ at the support vectors,
    - $|f(X)| > 1$ for all other points
    - $f(X) = 0$ at decision boundary
  - Classifier is $\text{sign}(f(X))$ again

- Can collapse solution back down into original space, like in Figure 5

- Could get messy in high dimensional space—how do you find the right basis functions?

- Generally use specially chosen KERNEL to generate basis functions
  - For mathematical reasons, functions of pairs of observations, say $x_i, x_{i'}$
  - The most popular Kernel functions are
    - d-DEGREE POLYNOMIAL: $K(x_i, x_{i'}) = (s \sum_{j=1}^{p} x_{ij} x_{i'j} + c)^d$
      - s, c, d are tuning parameters
      - Increasing d increases complexity
      - s, c just affect accuracy, sometimes set to 1, sometimes just tune one
    - GAUSSIAN RADIAL BASIS FUNCTION:

$$\exp\{-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\}$$

      - γ is a tuning parameter

- Solution with separating hyperplane leads to perfect training classification but may overfit
  - Usually use lower dimension with a little overlap and apply slack.
  - Cost parameter C is another tuning parameter:
    - Small C allows relatively more slack
      - Causes a wider margin and smoother fit
      - Analogous to shrinking parameter estimates in basis spline or polynomial regression
    - Large C allows less slack
      - Encourages ϵ_i's to be 0 as much as possible
      - Many points on margin; resulting in very wiggly fit

4.1 Application Notes

1. Often just have to guess as to which Kernel to use
   (a) Gaussian radial seems to be a default

Figure 6: SVM tuning parameters for various examples (Table 11.3 from Izenman)

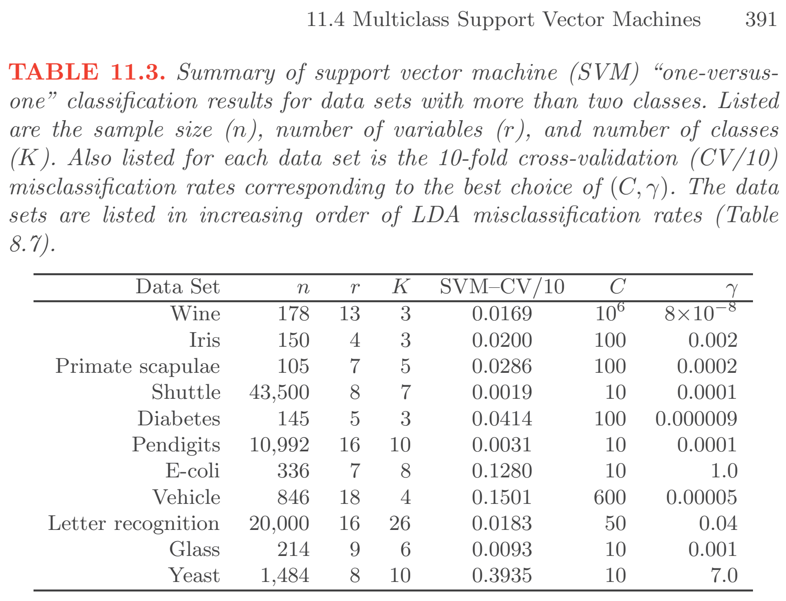

11.4 Multiclass Support Vector Machines 391

TABLE 11.3. *Summary of support vector machine (SVM) "one-versus-one" classification results for data sets with more than two classes. Listed are the sample size (n), number of variables (r), and number of classes (K). Also listed for each data set is the 10-fold cross-validation (CV/10) misclassification rates corresponding to the best choice of (C, γ). The data sets are listed in increasing order of LDA misclassification rates (Table 8.7).*

Data Set	n	r	K	SVM–CV/10	C	γ
Wine	178	13	3	0.0169	10^6	8×10^{-8}
Iris	150	4	3	0.0200	100	0.002
Primate scapulae	105	7	5	0.0286	100	0.0002
Shuttle	43,500	8	7	0.0019	10	0.0001
Diabetes	145	5	3	0.0414	100	0.000009
Pendigits	10,992	16	10	0.0031	10	0.0001
E-coli	336	7	8	0.1280	10	1.0
Vehicle	846	18	4	0.1501	600	0.00005
Letter recognition	20,000	16	26	0.0183	50	0.04
Glass	214	9	6	0.0093	10	0.001
Yeast	1,484	8	10	0.3935	10	7.0

i. One tuning parameter beyond C

(b) May be worthwhile to try others

2. With Gaussian radial, typically $\gamma \ll 1$ so try $\gamma = 10^{-a}$, $a = 0, 1, 2, 3, 4, 5$

3. For polynomial, start with $c = 0$ or 1, $s = 1$, and $d = 1$ or 2 and move from there.

   (a) Moving s, c is usually less drastic than incrementing d.

4. Cost is often $\gg 1$ so maybe try $C = 10^b$, $b = 0, 1, 2, 3, 4, 5$. See Table 11.3 in Izenman for optimal tuning parameter values in numerous examples.

5 Multi-class SVM

- When $K > 2$ the usual approach is to use a series of binary classifications in a "round robin" (1 vs 1 for all pairs)
  - For each pair of classes k_1 and k_2, restrict data to only these classes and train a classifier to find the function $f(x)$
  - Repeat for each pair of classes
    - ∗ In the end each chooses one class for each x where $f(x)$ is positive and k_2 if negative.
- In the end, for each x choose the class "wins" the most competitions

- Process can suffer from high variance because some classifiers for pairs k_1, k_2 may be built on very little data.

Example: SVM for Wheat Data (L22 - SVM Classification.R) Again, there are several packages to choose from that do various forms of SVM. The one that seems to be most popular is `e1071::svm()`. It is a front-end to a well-known C++ program that does SVM. It can do internal CV if you want a measure of test accuracy, but for tuning it is better to use `caret::train()`. Be aware that `caret` uses a different implementation of SVM than `e1071` that has different tuning parameter names. Of course, you can also write your own tuning loop.

I start with a simple analysis using defaults ($C = 1, \gamma = 1$) to show how it is working. I threw in a CV using the `cross` option just to see how it works.

```
> svm.1.1 <- svm(data=set1, type ~ ., kernel="radial",
+                  gamma=1, cost=1, cross=10)
> summary(svm.1.1)

Call:
svm(formula = type ~ ., data = set1, kernel = "radial", gamma = 1,
    cost = 1, cross = 10)

Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  192

 ( 67 67 58 )

Number of Classes:  3

Levels:
 Healthy Scab Sprout

10-fold cross-validation on training data:

Total Accuracy: 61.5
Single Accuracies:
 60 60 65 60 60 45 60 75 65 65
```

The model `summary()` shows that in the expanded dimension, the machine used 192 of the 200 observations as support vectors! Then it shows the CV results overall ("accuracy" is one minus the misclassification rate). It estimates that this machine has a 0.385 misclassification rate.

The output also shows the "decision values", which are the values of $f(X)$ that are computed from each 1 vs. 1 classifier. These are shown below for six observations that all get classified as `Healthy`. Notice that the decision value is positive if that pairwise classifier favours the first group listed in the column header, and is negative if it favours the second. We see that all of the first and second column values are positive, favouring `Healthy`. In the last column, five of the six favour `Sprout`, but `Sprout` has already lost to `Healthy`, to that "victory" is meaningless.

```
> head(svm.1.1$decision.values)
  Healthy/Sprout Healthy/Scab Sprout/Scab
1      0.9997571    0.7936783  -0.7867331
2      1.0668171    1.0558313   0.3546594
4      0.9998492    1.0281926   0.0612438
5      0.9318003    1.0945850   0.2184231
6      1.0004599    1.3113337   0.5467795
7      0.9999182    0.9999104   0.4124579
> head(svm.1.1$fitted)
      1       2       4       5       6       7
Healthy Healthy Healthy Healthy Healthy Healthy
Levels: Healthy Scab Sprout
```

Next I tuned using `caret::train` with 10-fold CV as we did last time. I chose a grid of $C = 10^0, 10^1, \ldots, 10^5$ and $\gamma = 10^{-5}, 10^{-4}, \ldots, 10^0$. The relative misclassification rates are shown in Figure 7, and separated by tuning parameter in Figure 8. There seems to be a general low region around $C = 10^3$–10^5 and $\gamma = 10^{-3}$–10^{-2}. The minimum is at $C = 10^4, \gamma = 10^{-3}$, which is in the heart of the low region, so that seems a reasonable final choice. I refit the final tuned machine and achieved test error of 0.333, which is the best we have seen so far!

Method	Training	Test	Test "SE"
KNN-Opt(12)	0.28	0.49	0.06
Logistic	0.28	0.40	
Logist-LASSO-min	0.28	0.43	
LDA	0.28	0.39	
QDA	0.26	0.43	
GAM	0.24	0.37	
NB-normal	0.32	0.39	
NB-kernel	0.23	0.43	
Tree-min	0.21	0.44	
RF OOB-Tuned	0	0.41	
NeuralNet(3,.01)	0.20	0.35	
SVM rad (1,1)	0.11	0.43	
SVM Tuned $(10^4, 10^{-3})$	0.23	0.33	

Figure 7: SVM tuning for the Wheat data: Relative misclassification with all combinations

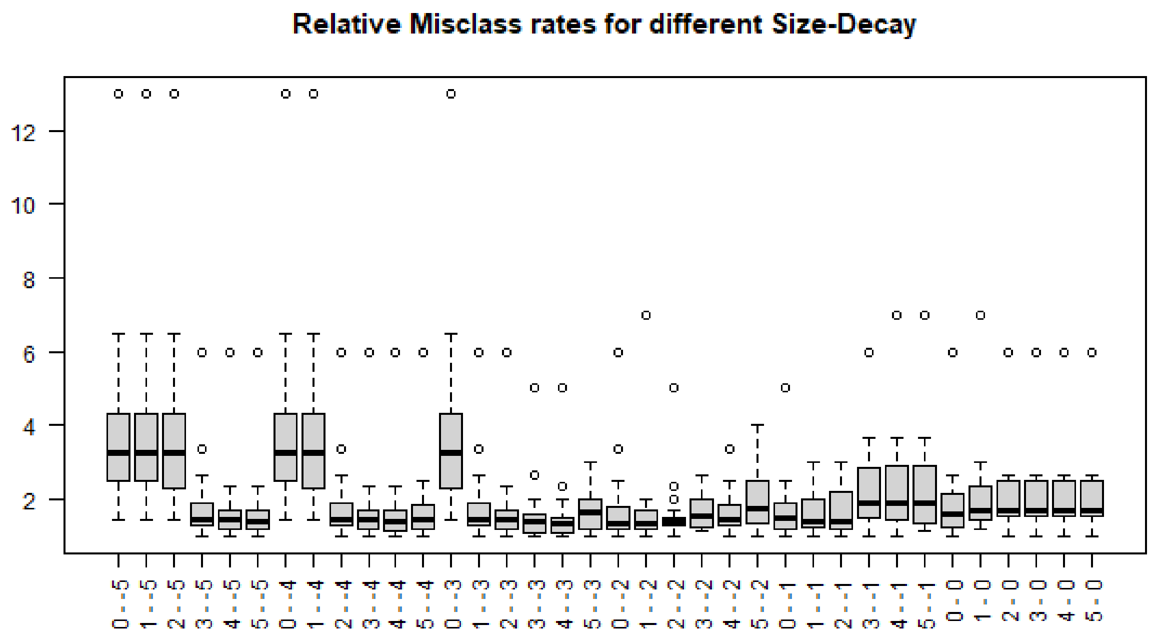
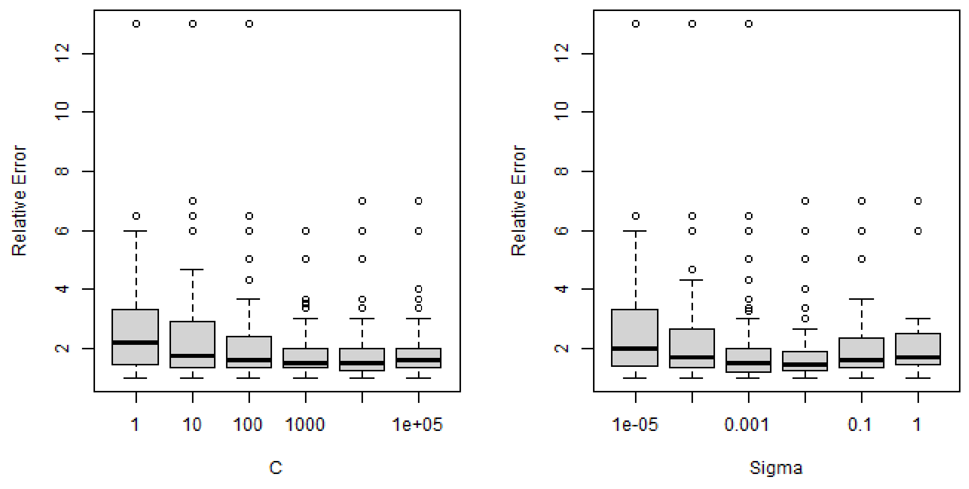


Figure 8: SVM tuning for the Wheat data: Relative misclassification by parameter



6 What to take away from this

1. SVM is an excellent candidate for classification in many problems.
2. Its biggest weakness may be that it does not do a good job at recognizing when a *reduced* dimension is appropriate (i.e., instead of all p dimensions, only a few are relevant to classification).
   (a) May want to apply dimension reduction first.
   (b) (There are papers on this...)

7 Exercises

Application

Return to the Vehicle data used in the previous lecture. Use the same split as before.

```
set.seed(46685326, kind="Mersenne-Twister")
perm <- sample(x=nrow(vehdata))
set1 <- vehdata[which(perm <= 3*nrow(vehdata)/4), ]
set2 <- vehdata[which(perm > 3*nrow(vehdata)/4), ]
```

We will tune SVM on the training set and estimate error on the test set.

1. Reset the seed for the Mersenne Twister to 49289448 before tuning. Run `train()` in `caret` but use 5-fold CV with 2 replicates. Use the same tuning grid that I used in the example.
   (a) **Print out the table summarizing the mean accuracy for each combination of tuning parameters**
      i. **Does there seem to be a region where the accuracy is maximized? If so, where?**
   (b) **Show the plot of relative misclassification rates for the combinations and separately by parameter.**
      i. **Comment on the plots: what do they suggest about good values for the parameters?**
   (c) Considering all evidence, do you think any more tuning is needed? **Explain, and if you believe more is needed, what would be your next grid?**
   (d) Regardless of this answer, use the best combination of parameters from the current grid to compute test misclassification error. **Report this rate and compare it to other methods.**