MSiA 400 Lab 4

Cross-Validation & Support-Vector Machines

Huiyu Wu

10/24/2020

NORTHWESTERN UNIVERSITY

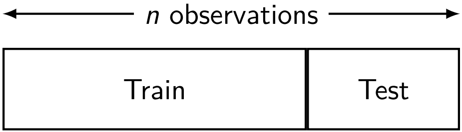
Cross-Validation (CV)

- Used to assess how well a fitted model will predict new observations
- Used to select the best model based on its performance on new data
- We don't want to assess performance on same dataset used to fit the model
    - Overfitting - models with more predictors will perform better than those with fewer

Train, Test, Validation

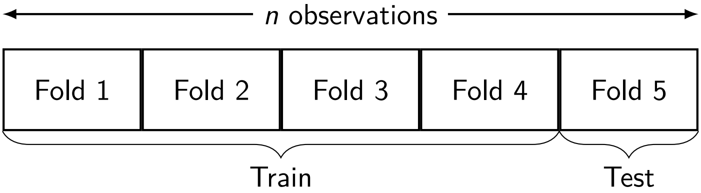
- **Training dataset** - sample of data used to fit the model
- **Validation dataset** - used for fit model evaluation while tuning hyperparameters
  - Evaluation can become biased as validation performance is incorporated into model
- **Test dataset** - used for evaluation of final fit model

Holdout CV



- Randomly assign observations to either training or validation/test sets
- Size of datasets is determined by user
- Involves single run

K-Fold CV



- Randomly assign observations to each fold
- Folds are (roughly) equal size
- Number of folds is determined by user
- Repeat so that each fold is the validation/test dataset

Holdout CV in R

- Dataset of student records:
  - GRE score, GPA, prestige of undergraduate university
  - Whether or not they were admited into graduate school

```
gradAdmit = read.csv('gradAdmit.csv')
set.seed(400)
n = nrow(gradAdmit) # number of samples
# hold out 20% for testing
sample = sample.int(n = n, size = floor(.2*n), replace = F)
train = gradAdmit[-sample,]; test = gradAdmit[sample,]
```

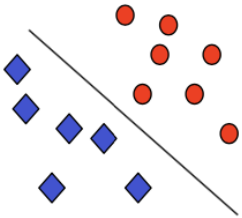
K-Fold CV in R

```r
library(caret)
```

```r
nfolds = 5
folds = createFolds(1:n, k=nfolds)
for (i in 1:nfolds){
    train = gradAdmit[-folds[[i]],]
    test = gradAdmit[folds[[i]],]
    # Train & analyze model
}
```

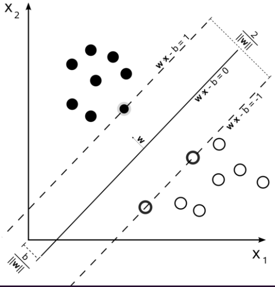
Suport Vector Machines

- Supervised learning method for classification
- Linear discriminant function: $\ell(x) = w^T x + b$
  - w is weight vector
  - b is bias
  - $\ell(x) \geq 0 \rightarrow x$ in class 1
  - $\ell(x) < 0 \rightarrow x$ in class 2



Suport Vector Machines

- Distance between x_i and hyperplane: $d_i = \frac{w^T x_i + b}{||w||}$
- $y_i \in \{-1, 1\}$, $\delta_i = \frac{y_i h(x_i)}{||w||}$
- If data is linearly separable, $\delta_i = y_i(w^T x_i + b) \geq 0$
- Margin defined as minimum distance of all points to hyperplane: $\delta^* = \min_i \frac{y_i h(x_i)}{||w||}$



Suport Vector Machines

- Goal is to maximize the margin: $\arg\max_{w,b} \min_i \frac{y_i h(x_i)}{||w||}$
- If data is linearly separable, w.l.o.g., we can assume $y^i(w^T x_i + b) \geq 1$
- Rewrite optimization problem as: $\arg\max_{w,b} \frac{1}{||w||}$ s.t. $y^i(w^T x_i + b) - 1 \geq 0, \ \forall \ i$
- Or: $\arg\min_{w,b} ||w||^2$ s.t. $y^i(w^T x_i + b) - 1 \geq 0, \ \forall \ i$
- Soft Margin: $\arg\min_{w,b} \frac{1}{2}||w||^2 + C \sum_i \max \left\{0, 1 - y^i(w^T x_i + b)\right\}$
- Dual: $\max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j {x_i}^T x_j$ s.t. $0 \leq \alpha_i \leq C, \ \sum_i \alpha_i y_i = 0$

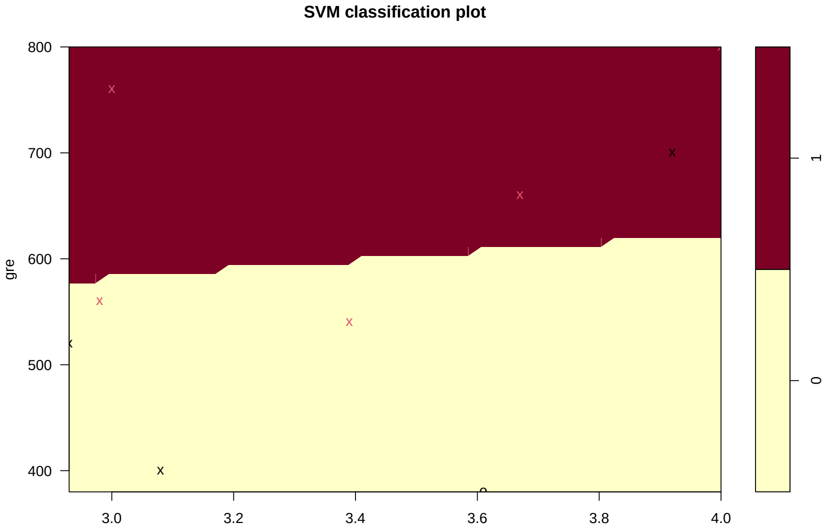
SVM in R

```r
library(e1071)
# https://www.rdocumentation.org/packages/e1071/versions/1.7-2/topics/svm
svm = svm(factor(admit)~gre+gpa, kernel = "linear", data=train[1:10, ], scale=FALSE)
summary(svm)
```

```
##
## Call:
## svm(formula = factor(admit) ~ gre + gpa, data = train[1:10, ], kernel = "linear",
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  9
##
##  ( 4 5 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

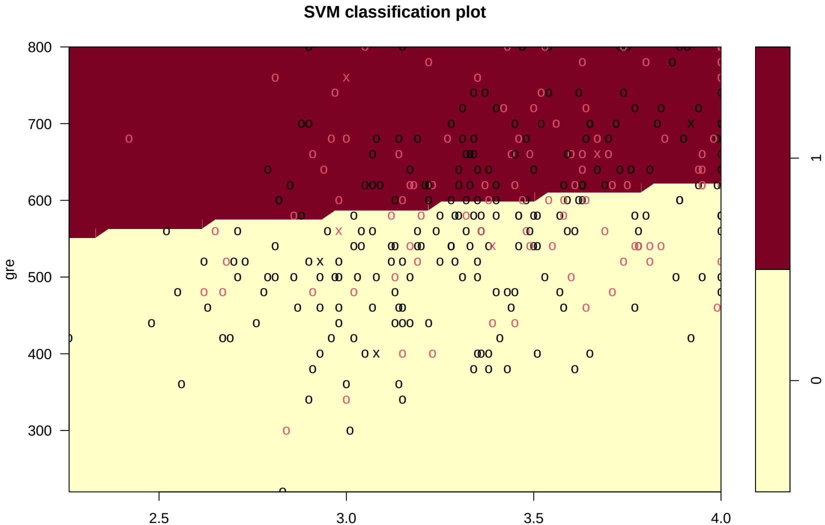
SVM in R

```
plot(svm, train[1:10,], gre~gpa)
```



SVM in R

```
plot(svm, train, gre~gpa)
```



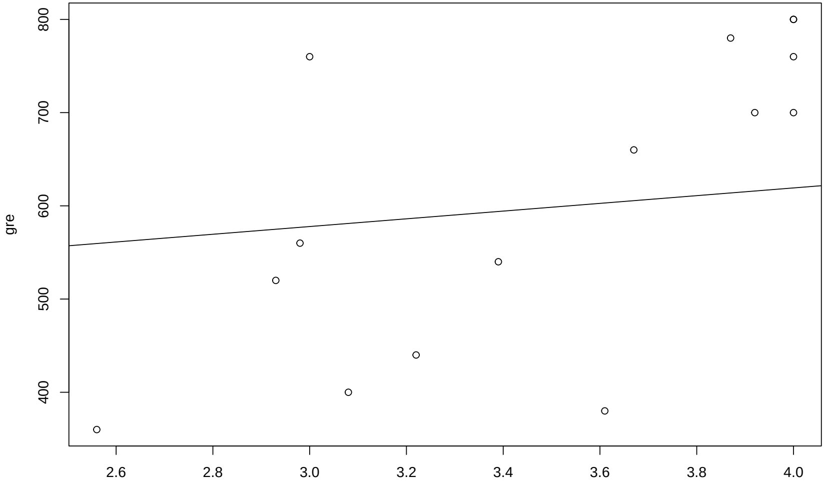
SVM in R

```r
pred = predict(svm, newdata=test, type='response')
pred[1:15]
```

```
##  4 14 16 26 32 41 45 46 51 58 65 69 72 73 75
##  1  1  0  1  1  1  1  0  1  0  0  0  0  0  1
## Levels: 0 1
```

SVM in R

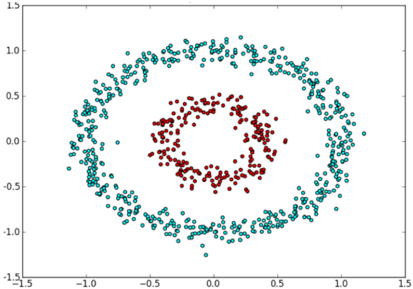
```
beta = t(svm$coefs)%*%svm$SV
beta0 = svm$rho
plot(train[1:15, c(3,2)])
abline(beta0 / beta[1], -beta[2] / beta[1])
```



Kernel Trick

- If linear discriminant not effective:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \phi(x) = \begin{pmatrix} x_1 \\ x_1 x_2 \\ x_3^2 \\ x_1 x_3 \end{pmatrix}$$



Kernel Trick

- Define kernel $K(x, z) = \phi(x)^T \phi(z)$
- Can define kernel to be easier to compute than inner product, or even $\phi(\cdot)$ (which need not be formed)
- Polynomial kernel: $K(x, z) = (x^T z + c)^d$
- Guassian kernel: $K(x, z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right)$
- Optimization: $\begin{cases} \max\limits_{\alpha} \sum\limits_i \alpha_i - \frac{1}{2} \sum\limits_i \sum\limits_j \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{s.t. } 0 \leq \alpha_i \leq C, \ \sum\limits_i \alpha_i y_i = 0 \end{cases}$

SVM in R

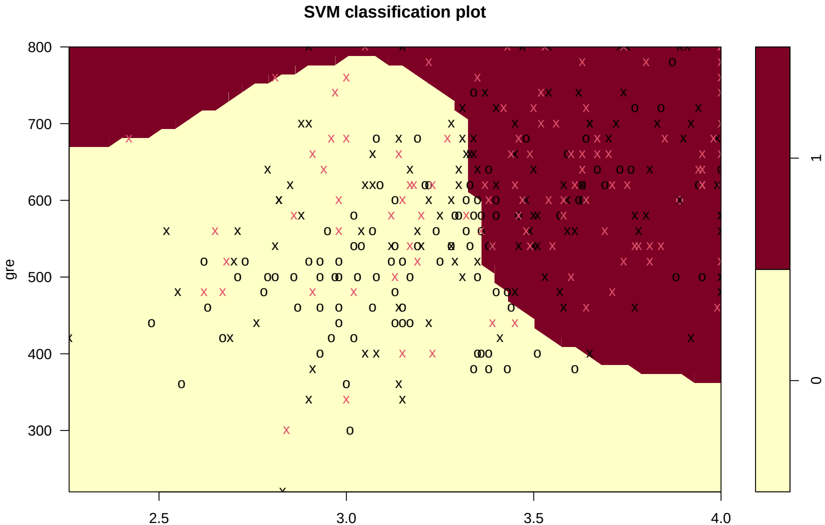
```
library(e1071)
```

```
# https://www.rdocumentation.org/packages/e1071/versions/1.7-2/topics/svm
svm = svm(factor(admit)~., data=train)
summary(svm)
```

```
##
## Call:
## svm(formula = factor(admit) ~ ., data = train)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  219
##
##  ( 116 103 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

SVM in R

```
plot(svm, train, gre~gpa)
```



SVM in R

```r
pred = predict(svm, newdata=test, type='response')
pred[1:15]
```

```
##  4 14 16 26 32 41 45 46 51 58 65 69 72 73 75
##  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0
## Levels: 0 1
```