DATA 624 - Non-Linear Regression

Zach Herold, Anthony Pagan, Betsy Rosalen

April 21, 2020

Linear Regression Review

Linear Regression model equations can be written either directly or indirectly in the form:

$$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + \ldots + b_P x_{iP} + e_i$$

Where:

- y_i is the outcome or response
- b_0 is the Y-intercept
- P is the number of predictor variables
- b_1 through b_P are the coefficients or parameters of the regression
- x_1 through x_P are the predictor variables
- e_i is the prediction error term

Linear Regression models

Goal: to minimize the sum of squared errors (SSE) or a function of the sum of squared errors

Minimize SSE

OLS - Ordinary Least Squares

PLS - Partial Least Squares

Minimize a function of the SSE

Penalized Models

- Ridge Regression
- Lasso
- Elastic Net

Linear Regression Pros and Cons

Advantages:

- They are highly interpretable
- Estimated coefficients allow for interpretation of the relationships between predictors
- Coefficient standard errors can be calculated and used to assess the statistical significance of each predictor

Disadvantages:

- Useful only when the relationship between the predictors and the response falls along a straight line or flat hyperplane
- They may not be able to adequately capture relationships that are not linear

What to do when you suspect there is a non-linear relationship but don't know the nature of the non-linearity?

Non-Linear Regression

Non-linear regression equations take the form:

$$y = f(x, \beta) + \varepsilon$$

Where:

- x is a vector of p predictors
- β is a vector of k parameters
- $f()$ is a known regression function (that is not linear)
- ε is the prediction error term

Any model equation that cannot be written in the linear form
$y_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + \ldots + b_P x_{iP} + e_i$ ***is non-linear!***

Non-Linear Regression Pros and Cons

Advantages:

- They can fit almost any functional form and you don't need to know the form before training the model
- Can model much more complex relationships between the predictors and the outcome than linear models

Disadvantages:

- Often not very interpretable
- Can be computationally expensive
- Some models can be prone to overfitting

Non-Linear Regression Models

Goal: to find the (non-linear) curve that comes closest to your data

Neural Networks (NN)

Inspired by theories about how the brain works and the interconnectedness of the neurons in the brain

Multivariate Regression Splines (MARS)

Splits each predictor into two groups using a “hinge” or “hockey-stick” function and models linear relationships to the outcome for each group separately

K-Nearest Neighbors (KNN)

Predicts new samples by finding the closest samples in the training set predictor space and taking the mean of their response values. K represents the number of closest samples to consider.

Support Vector Machines (SVM)

Robust regression that aims to minimize the effect of outliers with an approach that uses a subset of the training data points called “support vectors” to predict new values, and counter-intuitively excludes data points closest to the regression line from the prediction equation.

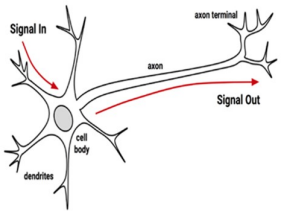
Tree-Based Models

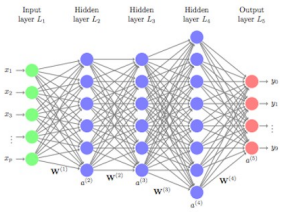
Subject of the next presentation, so stay tuned!

Neural Networks

Description

- Neural Network



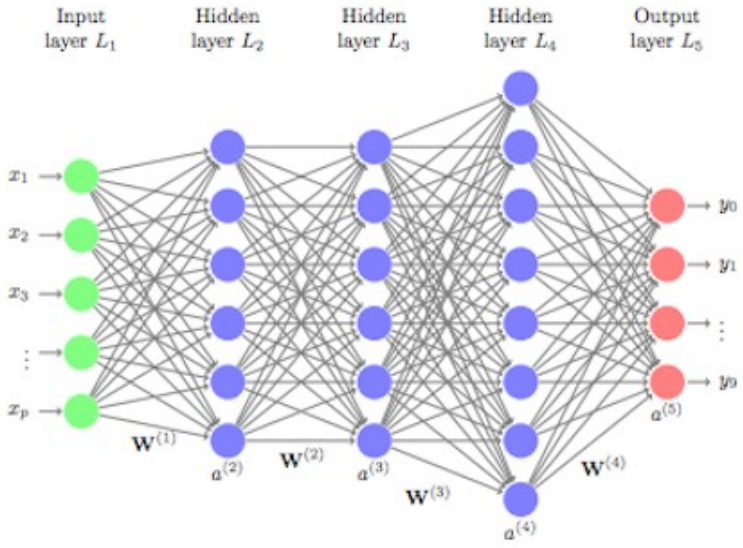


- Deep Neural Network - Modeled by Neural network of the brain
    - *Input-Dendrite (Artificial NN Input values)*
    - *Nucleus (Inputs with unique weights and summed together and passed threshold in 1 to many hidden layers (neural network 1 layer, deep) neural network >1 layers)*
    - *Output – Axon and terminal (Based on about becomes a 0 or 1 with Sigmoid function) Transformed by a nonlinear function g(). Passthrough Synapse to next Dendrite (passed to next neuron)*
    - *For P predictors there are H(P+1)+H+1 parameters*
        - For 228 predictors and 3 Hidden units would be:
            - 3(228+1)+3+1=691 parameters

Ref: https://www.youtube.com/watch?v=oYbVFhK_olY

Neural Networks

Calculation



- invvalue<-input1value + eachvar*weight+…
- outvalue<-1/(1+exp(-iv))
- invvalue<-input2value+ov*weight
- outvalue<-1/(1+exp(-iv2))

Neural Networks

Computing examples

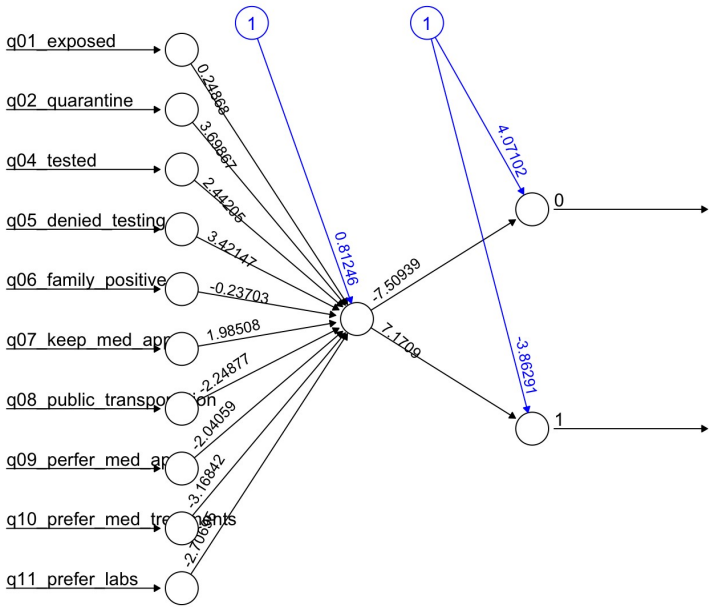
```
# Create Model
library(neuralnet)
nn<-neuralnet(q03_symptoms~., data=training, hidden =1 , linear.output=FALSE, rep=3)
```

- Hidden is number of nodes/neuron in a layer, c(2,1) would be 2layers with and 1 nodes/neurons
- Can add lifesign= ‘full” to get all data points and rep = number of repetition times to run model.
- When plotting with Rep can use plot(n, num) to show plot for rep.

Ref: https://www.youtube.com/watch?v=-Vs9Vae2KI0

Neural Networks

Computing examples




```
## NULL
```

```
#Predict
output<-compute(nn, training[-3])
head(output$net.result)
```

```
##         [,1]       [,2]
## 1 0.9781103 0.01241033
## 2 0.9787063 0.01201374
## 3 0.9787063 0.01201374
## 4 0.9787063 0.01201374
## 6 0.9815269 0.01016525
## 7 0.9787063 0.01201374
```

```
in1<-nn$weights[[2]][[1]][1]+sum(nn$weights[[2]][[1]]*nn$startweights[[2]][[1]])
out<-1/(1+exp(-in1))
in2<-nn$weights[[2]][[2]][1]+sum(nn$weights[[2]][[2]][2]*out)
out2<-1/(1+exp(-in2))
```

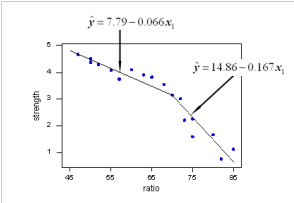
Neural Networks

Pros and Cons

- Pros
    - *Robust with noisy data*
- Cons
    - *Less interpretable*
    - *Need longer training times*
    - *Neural Networks have a tendency to over-fit the relationship between predictor and response due the large coefficients*
    - *Fix*
        - Early stopping
        - Weight Decay with regularization with lambda values 0-.1

Multivariate Adaptive Regression Splines

Description



- Creates 2 contrasted versions to of a predictor
- 1 or 2 predictors at a time
- Breaks predictors to 2 groups and models between
    - *Hockey-stick(hinges)*
    - *Left-hand – values > 0 than cut point*
    - *Right-hand - values < 0 than cut point*
    - *Piece-wise linear model isolated portion of original data*
    - *Predictor/cut-point with smallest error*
    - *X<a, h(x − a) and h(a − x)*

Multivariate Adaptive Regression Splines

Description

- Pruning used to remove parameters
    - *The degree of the features that are added to the model and the number of retained terms.*
    - *The latter parameter can be automatically determined using the default pruning procedure (using GCV), set by the user or determined using an external resampling technique*
- Pros
    - *Model automatically conducts feature selection*
    - *Interpretability, each hinge feature is responsible for modeling a specific region in the predictor space using piecewise linear model.*
    - *MARS require very little pre-processing, transformation and filtering not needed.*
- Cons
    - *Speed, other models may run faster*

Splines

Description

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

$$h(x,\xi) = (x-\xi)_+^3 = \begin{cases} (x-\xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise,} \end{cases}$$

- Types
  - *Polynomial Splines- continuous at the knot*
  - *Cubic Splines- continuous at the knot. Same as linear splines instead of power of 0 its 3.*
  - *Smoothing Splines*
    - Splines without knots
    - Use smooth.splines() function in R. Does leave-one-out cross validation when smooth.spine(var, var2)with no df defined.
    - Find the function g that minimizes where λ is a non-negative tuning parameter.
    - loess() used for Local regression. Can find local regression for range of X by weighted least square.
  - *GAM*

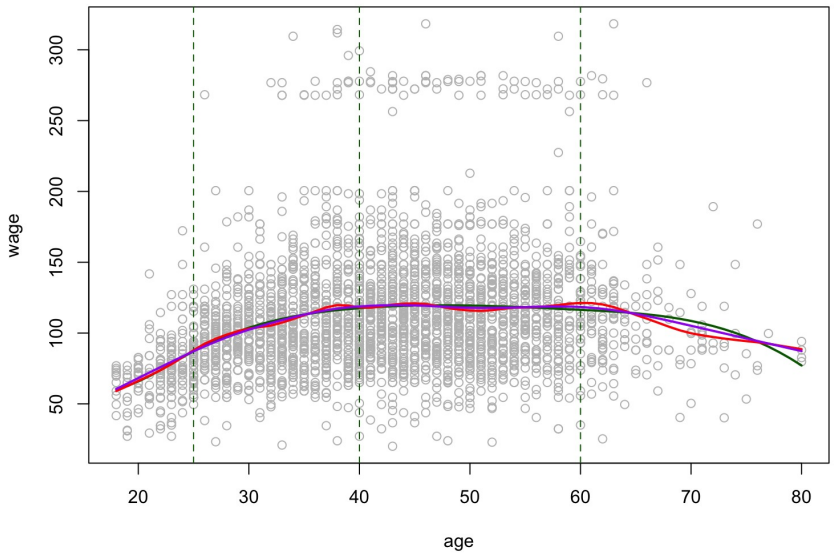
Ref: https://www.youtube.com/watch?v=UDDXkffB-aE&t=329s

Splines

Computing examples Spines

SPLINES

- lm(wage~bs(age,knots=c(25,40,60)), data=Wage) #cubic polynomials-




```
## Call:
## smooth.spline(x = age, y = wage, cv = TRUE)
##
## Smoothing Parameter  spar= 0.6988943  lambda= 0.02792303 (12 iterations)
## Equivalent Degrees of Freedom (Df): 6.794596
## Penalized Criterion (RSS): 75215.9
## PRESS(l.o.o. CV): 1593.383
```

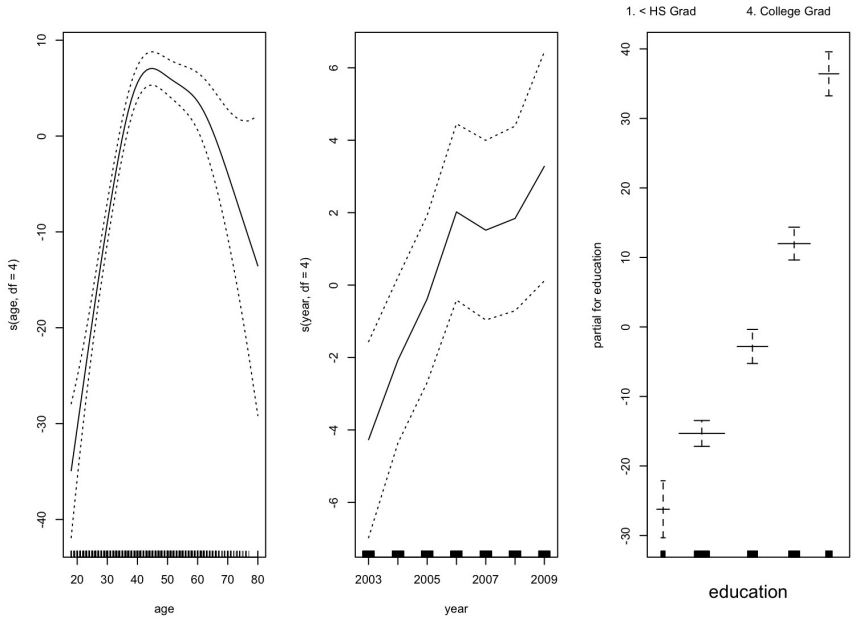
Ref: https://www.youtube.com/watch?v=u-rVXhsFyxo&t=450s

Splines

Computing examples GAM

GAM

- gam(wage~s(age,df=4)+s(year,df=4)+education,data=Wage)

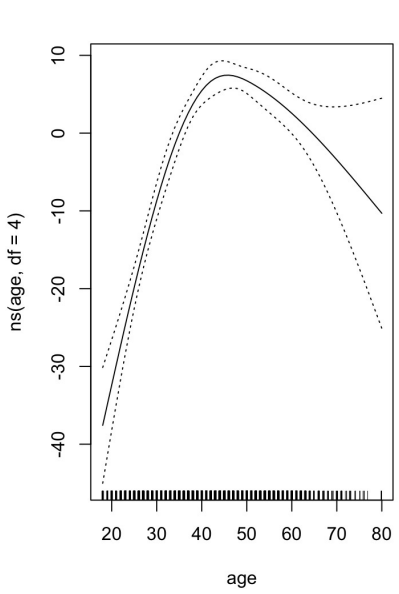


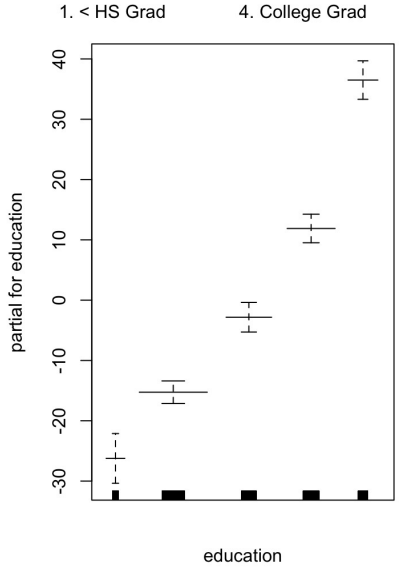
Splines

Computing examples GAM

GAM with Natural Splines

- lm(wage~ns(age,df=4)+education, data=Wage)





Splines

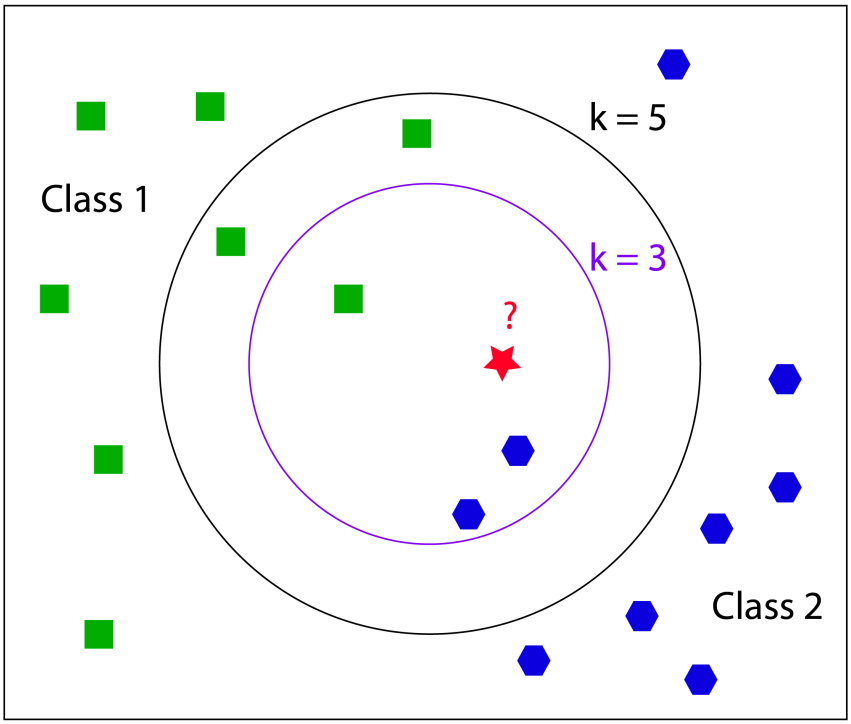
Computing examples GAM

GAM compare with ANOVA

```
gam2<-gam(I(wage>250)~s(age,df=4)+s(year,df=4)+education,data=Wage,family=binomial)#GAM for
  logistic regression
gam2a<-gam(I(wage>250)~s(age,df=4)+year+education,data=Wage,family=binomial)
anova(gam2a, gam2)#compare 2 gams with Anova
```

```
## Analysis of Deviance Table
##
## Model 1: I(wage > 250) ~ s(age, df = 4) + year + education
## Model 2: I(wage > 250) ~ s(age, df = 4) + s(year, df = 4) + education
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      2990     603.78
## 2      2987     602.87  3  0.90498   0.8242
```

K-Nearest Neighbors



K-Nearest Neighbors

Coded Examples - Iris Dataset

```
data(iris)
index <-1:nrow(iris)
test.data.index <-sample(index, trunc(length(index)/4))
test.data <- iris[test.data.index,]
train.data <-iris[-test.data.index,]

training <- as.matrix(train.data[,1:4])
rownames(training) <- NULL
train.labels <- as.integer(train.data[,5])

test <- as.matrix(test.data[,1:4])
rownames(test) <- NULL
test.labels <- as.integer(test.data[,5])
```

K-Nearest Neighbors

Overview

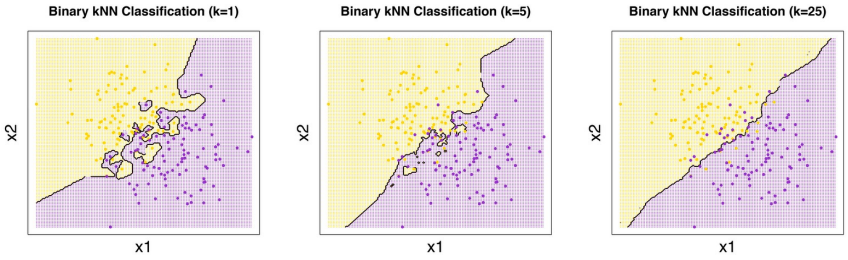
- a nonparametric lazy supervised learning method
  - *does not make any assumptions about data distribution*
  - *does not require an explicit learning phase for generalization*
  - *keeps all training examples in memory*
- finds k training examples closest to x and returns
  - *the majority label (through 'votes'), in case of classification*
  - *the median or mean, in case of regression*

K-Nearest Neighbors

Number of Neighbors

If k = 1, then the new instance is assigned to the class where its nearest neighbor.

If we give a small (large) k input, it may lead to over-fitting (under-fitting). To choose a proper k-value, one can count on cross-validation or bootstrapping.



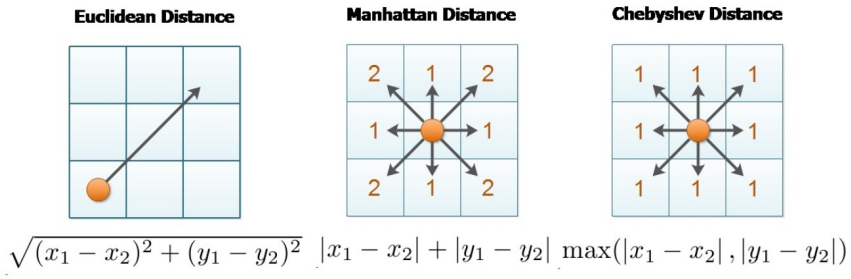
K-Nearest Neighbors

Similarity/ Distance Metrics

- knn algorithm performs:
  - *computation of distance matrix*
  - *ranking of k most similar objects.*

Visual difference

This image summarizes the difference in the three distance metrics:



$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ $|x_1 - x_2| + |y_1 - y_2|$ $\max(|x_1 - x_2|, |y_1 - y_2|)$

K-Nearest Neighbors

Recommended R Package: knnGarden

knnVCN k-Nearest Neighbor Classification of Versatile Distance Version method can be “euclidean”, “maximum”, “manhattan”,“canberra”, “binary” or “minkowski”

knnMCN: Mahalanobis Distance

```
library(knnGarden)
library(caret)
knnMod <- knnVCN(TrnX=training,
                 OrigTrnG=train.labels,
                 TstX=test,
                 ShowObs=TRUE,
                 K=5,
                 method="minkowski",
                 p = 3)
```

K-Nearest Neighbors

Iris Dataset - Perfect Accuracy

```
test.predicted <- knnMod$TstXIBelong

test.predicted <- as.integer(ifelse(test.predicted == 1,1,
                                    ifelse(test.predicted == 2,2,
                                           3)))
xtab <- table(test.predicted, test.labels)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##                test.labels
## test.predicted  1  2  3
##              1 14  0  0
##              2  0  8  1
##              3  0  1 13
##
## Overall Statistics
##
##                Accuracy : 0.9459
##                  95% CI : (0.8181, 0.9934)
##     No Information Rate : 0.3784
##     P-Value [Acc > NIR] : 4.494e-13
##
##                   Kappa : 0.9174
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.8889   0.9286
## Specificity            1.0000   0.9643   0.9565
## Pos Pred Value         1.0000   0.8889   0.9286
## Neg Pred Value         1.0000   0.9643   0.9565
## Prevalence             0.3784   0.2432   0.3784
## Detection Rate         0.3784   0.2162   0.3514
## Detection Prevalence   0.3784   0.2432   0.3784
## Balanced Accuracy      1.0000   0.9266   0.9425
```

K-Nearest Neighbors

Advantages & Disadvantages

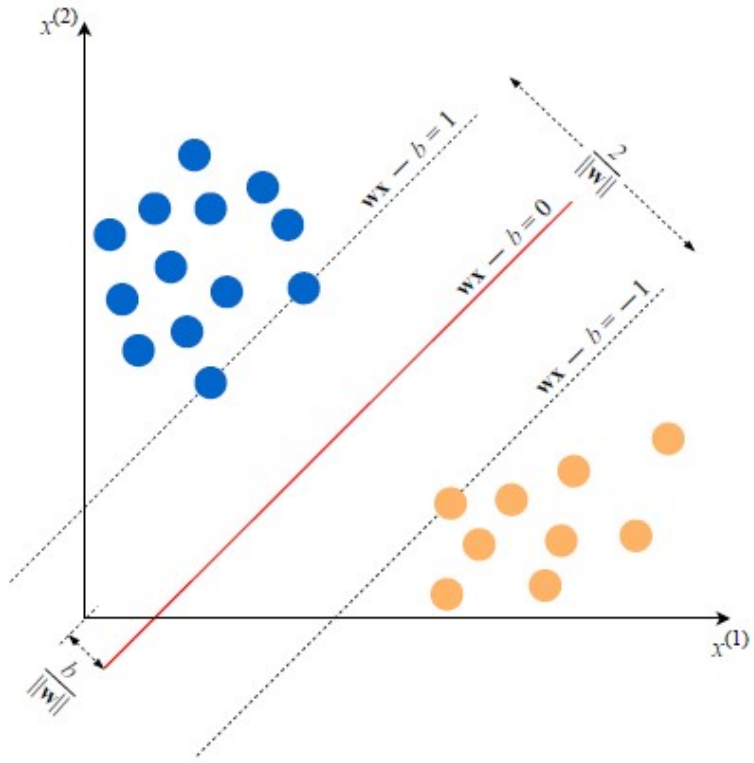
- Strengths
    - *cost of the learning process is zero*
    - *nonparametric, which means that you do not have to make the assumption of data distribution*
- Drawbacks
    - *Doesn't handle categorical features well, requires label encoding*
    - *Expensive computation for a large dataset*
    - *Suffers curse of dimensionality*

K-Nearest Neighbors

Other Packages

- caret
- class
- kknn
- FastKNN
- KernelKnn
- philentropy
- Python: sklearn.neighbors
  KNeighborsClassifier (n_neighbors, p, metric)

Support Vector Machines



Support Vector Machines

Overview

- Black box method
    - *applicable to both supervised regression and classification problems*
    - *Involves optimally separating (maximal margin) hyperplanes*
        - in d-dimensional space, a hyperplane is a d-1 dimensional separator
    - *For non-separable cases, a non-linear mapping transforms the data into a kernel-induced feature space F, and then a linear machine is used to classify them in the feature space*

Support Vector Machines

SVM Applications

- Bioinformatics
    - *Protein Structure Prediction*
    - *Breast Cancer Diagnosis*
- Computer vision
    - *Detecting Steganography in digital images*
    - *Intrusion Detection*
    - *Handwriting Recognition*
- Computational linguistics

Support Vector Machines

Origins

- invented by Boser, Guyon and Vapnik, and first introduced at the Computational Learning Theory (COLT) 1992 conference.
- idea of soft margin, which allows misclassified examples, was suggested by Corinna Cortes and Vladimir N. Vapnik in 1995
- machine learning foundations from the 1960s
    - *large margin hyperplanes in the input space were discussed for example by Duda and Hart, Cover, Vapnik et al.*
    - *the use of kernels was proposed by Aronszajn, Wahba, Poggio*
    - *in 1964, Aizermann et al. introduced the geometrical interpretation of the kernels as inner products in a feature space*

Support Vector Machines

Soft vs. Hard Margins

Allow some misclassification by introducing a slack penalty variable (ξ). T

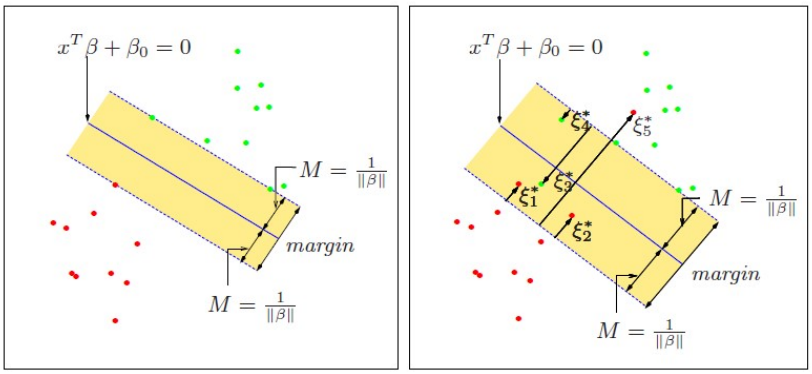


FIGURE 12.1. *Support vector classifiers. The left panel shows the separable case. The decision boundary is the solid line, while broken lines bound the shaded maximal margin of width $2M = 2/\|\beta\|$. The right panel shows the nonseparable (overlap) case. The points labeled ξ_j^* are on the wrong side of their margin by an amount $\xi_j^* = M\xi_j$; points on the correct side have $\xi_j^* = 0$. The margin is maximized subject to a total budget $\sum \xi_i \leq$ constant. Hence $\sum \xi_j^*$ is the total distance of points on the wrong side of their margin.*

Support Vector Machines

Cost Penalty

The slack variable is regulated by hyperparameter cost parameter C. - when C=0, there is a less complex boundary - when C=inf, more complex boundary, as algorithms cannot afford to misclassify a single datapoint (overfitting)

In the case of the 2-norm Soft Margin classification the optimization problem takes the form:

$$\begin{aligned} \text{minimize} \quad & t(w,\xi) = \frac{1}{2}\|w\|^2 + \frac{C}{m}\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\langle x_i, w\rangle + b) \geq 1 - \xi_i \qquad (i = 1,\ldots,m) \\ & \xi_i \geq 0 \qquad (i = 1,\ldots,m) \end{aligned} \tag{12}$$

Support Vector Machines

SVM with Low Cost Parameter

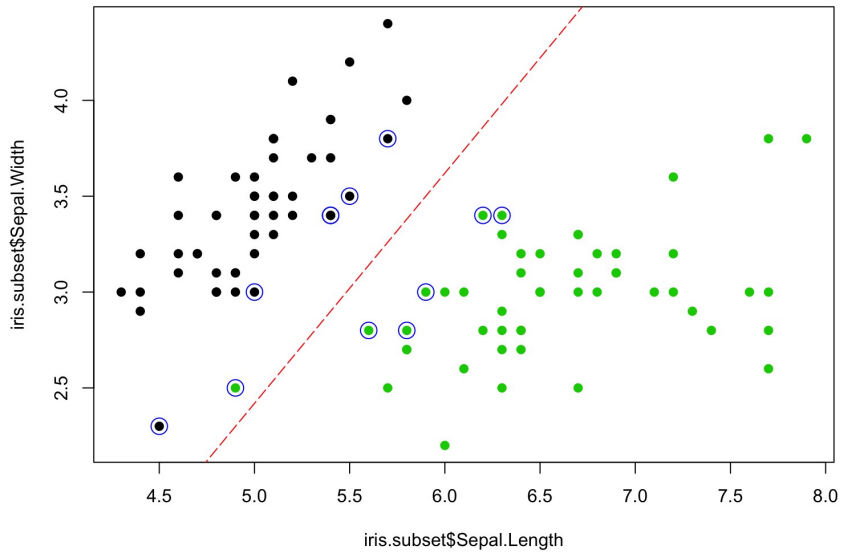
To create a soft margin and allow some misclassification, we use an SVM model with small cost (C= 1)

```
library(e1071)
iris.subset <- iris[iris$Species %in% c("setosa","virginica"),][c("Sepal.Length",
  "Sepal.Width", "Species")]
svm.model = svm(Species ~ ., data=iris.subset, kernel='linear', cost=1, scale=FALSE)
plot(x=iris.subset$Sepal.Length,y=iris.subset$Sepal.Width, col=iris.subset$Species, pch=19)
points(iris.subset[svm.model$index,c(1,2)],col="blue",cex=2) #The index of the resulting
  support vectors in the data matrix.
w = t(svm.model$coefs) %*% svm.model$SV
b = -svm.model$rho
abline(a=-b/w[1,2], b=-w[1,1]/w[1,2], col="red", lty=5)
```

Support Vector Machines

Support vectors circled with separation line (Low Cost Parameter)

To create a soft margin and allow some misclassification, we use an SVM model with small cost (C= 1)



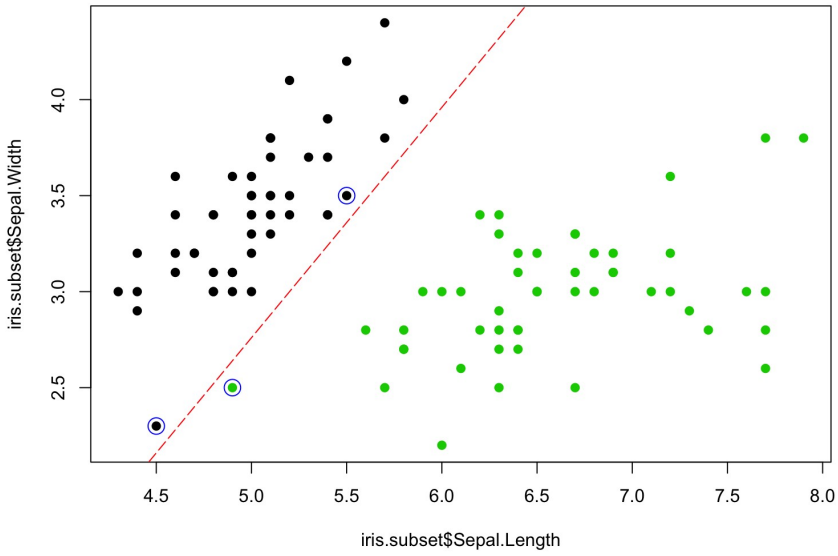
Support Vector Machines

SVM with High Cost Parameter

```
svm.model = svm(Species ~ ., data=iris.subset, type='C-classification', kernel='linear',
  cost=10000, scale=FALSE)
plot(x=iris.subset$Sepal.Length,y=iris.subset$Sepal.Width, col=iris.subset$Species, pch=19)
points(iris.subset[svm.model$index,c(1,2)],col="blue",cex=2)
w = t(svm.model$coefs) %*% svm.model$SV
b = -svm.model$rho #The negative intercept.
abline(a=-b/w[1,2], b=-w[1,1]/w[1,2], col="red", lty=5)
```

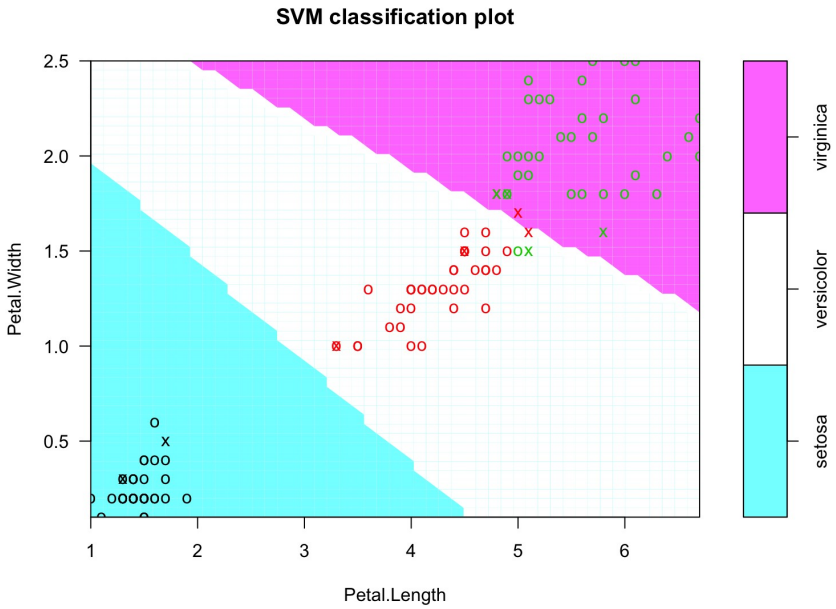
Support Vector Machines

Support vectors circled with separation line (High Cost Parameter)



Support Vector Machines

SVM Classification Plot

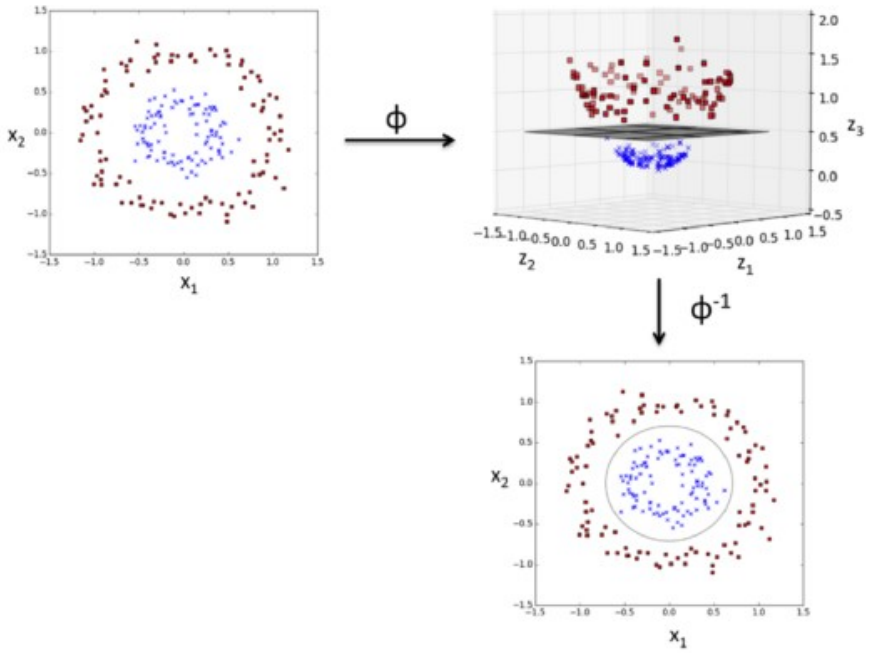


Within the scatter plot, the X symbol shows the support vector and the O symbol represents the data points. These two symbols can be altered through the configuration of the svSymbol and dataSymbol options. Both the support vectors and true classes are highlighted and colored depending on their label (green refers to viginica, red refers to versicolor, and black refers to setosa). The last argument, slice, is set when there are more than two variables. Therefore, in this example, we use the additional variables, Sepal.width and Sepal.length, by assigning a constant of 3 and 4.

Support Vector Machines

Non-Linear Cases – Theory

Cover's Theorem (Thomas M. Cover (1965): given any random set of finite points, then with high probability these points can be made linearly separable by mapping them to a higher dimension.



Support Vector Machines

Kernel Trick

All kernel functions take two feature vectors as parameters and return the scalar dot (inner) product of the vectors. Have property of symmetry and is positive semi-definite.

By performing convex quadratic optimization, we may rewrite the algorithm so that it is independent of transforming function ϕ

Support vector machines (Vapnik 1998) have gained prominence in the field of machine learning and pattern classification and regression. The solutions to classification and regression problems sought by kernel-based algorithms such as the SVM are linear functions in the feature space:

$$f(x) = w^\top \Phi(x) \tag{10}$$

for some weight vector $w \in F$. The kernel trick can be exploited in this whenever the weight vector w can be expressed as a linear combination of the training points, $w = \sum_{i=1}^{n} \alpha_i \Phi(x_i)$, implying that f can be written as

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x) \tag{11}$$

Support Vector Machines

Choice of Kernel in kernlab (I)

The package includes implementations of the following kernels:

- the linear `vanilladot` kernel implements the simplest of all kernel functions

$$k(x, x') = \langle x, x' \rangle \tag{2}$$

  which is useful specially when dealing with large sparse data vectors x as is usually the case in text categorization.

- the Gaussian radial basis function `rbfdot`

$$k(x, x') = \exp(-\sigma \|x - x'\|^2) \tag{3}$$

  which is a general purpose kernel and is typically used when no further prior knowledge is available about the data.

- the polynomial kernel `polydot`

$$k(x, x') = (\text{scale} \cdot \langle x, x' \rangle + \text{offset})^{\text{degree}}. \tag{4}$$

  which is used in classification of images.

Support Vector Machines

Choice of Kernel in kernlab (II)

- the hyperbolic tangent kernel `tanhdot`

$$k(x, x') = \tanh\left(\text{scale} \cdot \langle x, x' \rangle + \text{offset}\right) \tag{5}$$

  which is mainly used as a proxy for neural networks.

- the Bessel function of the first kind kernel `besseldot`

$$k(x, x') = \frac{\text{Bessel}^n_{(\nu+1)}(\sigma\|x - x'\|)}{(\|x - x'\|)^{-n(\nu+1)}}. \tag{6}$$

  is a general purpose kernel and is typically used when no further prior knowledge is available and mainly popular in the Gaussian process community.

- the Laplace radial basis kernel `laplacedot`

$$k(x, x') = \exp(-\sigma\|x - x'\|) \tag{7}$$

  which is a general purpose kernel and is typically used when no further prior knowledge is available.

- the ANOVA radial basis kernel `anovadot` performs well in multidimensional regression problems

$$k(x, x') = \left(\sum_{k=1}^{n} \exp(-\sigma(x^k - {x'}^k)^2)\right)^d \tag{8}$$

  where x^k is the kth component of x.

Support Vector Machines

Model Tuning (Cost & Gamma)

- One can tune hyperparameters (cost and gamma) using tune.svm() in e1071 and train() in caret

- For the gamma argument,
  - *the default value is equal to (1/data dimension), and*
  - *it controls the shape of the separating hyperplane.*
  - *Increasing the gamma argument usually increases the number of support vectors.*

```
tuned = tune.svm(Species~., data = iris.subset, gamma = 10^(-6:-1),cost = 10^(0:2))
model.tuned = svm(Species~., data = iris.subset, gamma = tuned$best.parameters$gamma, cost =
  tuned$best.parameters$cost)
```

Support Vector Machines

Advantages & Disadvantages

- Strengths
  - *effective in high dimensional spaces (dim > N)*
  - *works well with even unstructured & semi-structured data (text, images)*
  - *does not suffer from local optimal and multicollinearity*
- Drawbacks
  - *difficult to interpret the final model, variable weights, and meld with business logic*
  - *forcing separation of the data can easily lead to overfitting, particularly when noise is present in the data*
  - *choosing a "good" kernel function is not easy*
  - *long training time for large datasets*