Kernel Methods

Max Turgeon

STAT 4690–Applied Multivariate Analysis

Motivation

- Linearity has been an important assumption for most of the multivariate methods we have discussed.
    - Multivariate Linear Regression
    - PCA, FA, CCA
- This assumption may be more realistic after a transformation of the data.
    - E.g. Log transformation
    - Embedding in a higher dimensional space?

Example i

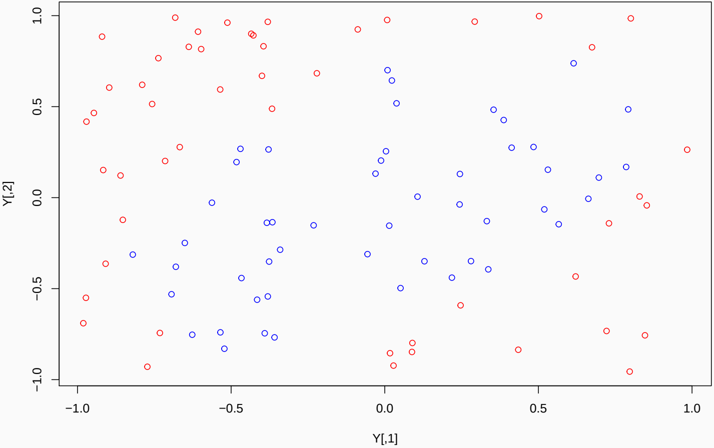
```
set.seed(1234)
n <- 100
# Generate uniform data
Y <- cbind(runif(n, -1, 1),
           runif(n, -1, 1))

# Check if it falls inside ellipse
Sigma <- matrix(c(1, 0.5, 0.5, 1), ncol = 2)
dists <- sqrt(diag(Y %*% solve(Sigma) %*%
                     t(Y)))
inside <- dists < 0.85
```

Example ii

```r
# Plot points
colours <- c("red", "blue")[inside + 1]
plot(Y, col = colours)
```

Example iii

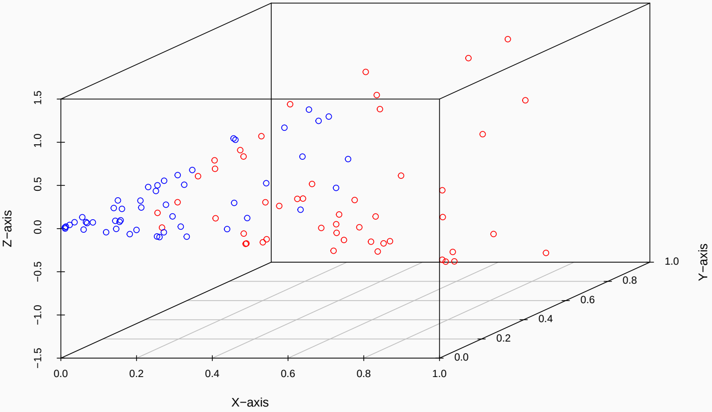


Example iv

```r
# Transform data
# (X, Y) -> (X^2, Y^2, sqrt(2)*X*Y)
Y_transf <- cbind(Y[,1]^2, Y[,2]^2,
                  sqrt(2)*Y[,1]*Y[,2])
```

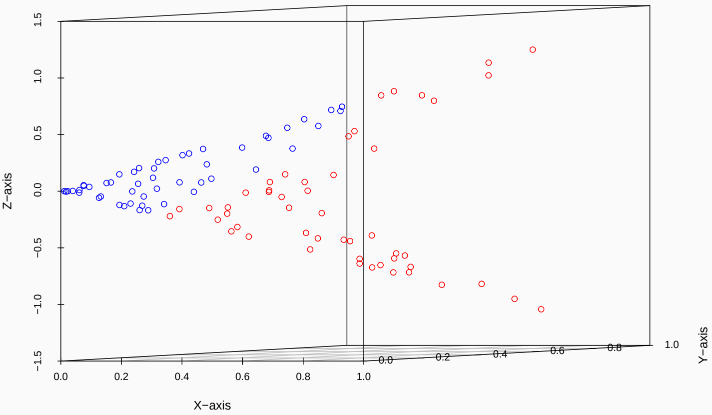
```r
library(scatterplot3d)
scatterplot3d(Y_transf, color = colours,
              xlab = "X-axis",
              ylab = "Y-axis",
              zlab = "Z-axis")
```

Example v



1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Z−axis
0.0
0.2
0.4
0.6
0.8
1.0
X−axis
0.0
0.2
0.4
0.6
0.8
1.0
Y−axis

Example vi



1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
Z−axis
0.0
0.2
0.4
0.6
0.8
1.0
X−axis
0.0
0.2
0.4
0.6
0.8
1.0
Y−axis

Example vii

```r
# Linear regression
outcome <- ifelse(inside, 1, -1)
head(outcome)
```

```
## [1] -1  1  1 -1 -1  1
```

```r
model1 <- lm(outcome ~ Y)
pred1 <- sign(predict(model1))
table(outcome, pred1) # 67%
```

Example viii

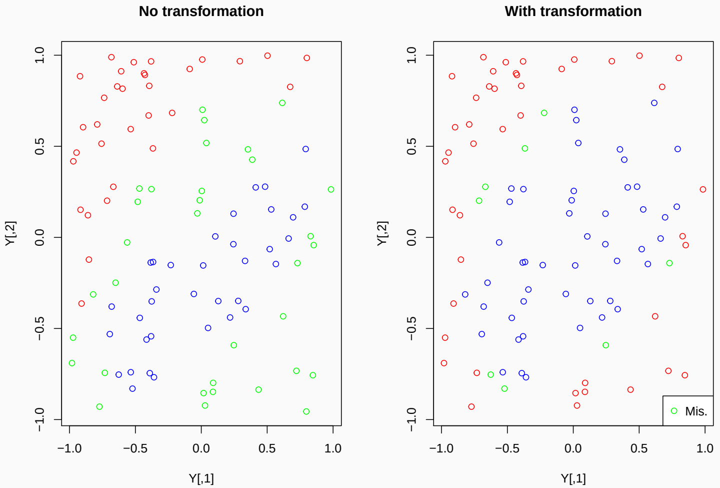
```
##          pred1
## outcome -1  1
##      -1 32 18
##      1  15 35
```

```
model2 <- lm(outcome ~ Y_transf)
pred2 <- sign(predict(model2))
table(outcome, pred2) # 92%
```

Example ix

```
##         pred2
## outcome -1  1
##      -1 44  6
##      1   2 48
```

Example x



No transformation
With transformation
Y[,2]
Y[,1]
1.0
0.5
0.0
−0.5
−1.0
Mis.

Overfitting i

- **Overfitting** is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably" (OED)
  - In other words, a model is overfitted if it explains the *training* data very well, but does poorly on *test* data.
- In regression, this often happens when we have too many covariates
  - Too many *parameters* for the sample size

Overfitting ii

- When embedding our covariates into a higher dimensional space, we are increasing the number of parameters.
    - There is a danger of overfitting.
- **One possible solution**: Regularised (or penalised) regression.
    - We constrain the parameter space using a penalty function.

Ridge regression i

- Let $(Y_i, \mathbf{X}_i), i = 1, \ldots, n$ be a sample of outcome with covariates.
- **Univariate Linear Regression**: Assume that we are interested in the linear model

$$Y_i = \beta^T \mathbf{X}_i + \epsilon_i.$$

- The Least-Squares estimate of β is given by

$$\hat{\beta}_{LS} = (\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X} \mathbf{Y},$$

where

Ridge regression ii

$$\mathbb{X}^T = \begin{pmatrix} \mathbf{X}_1 & \cdots & \mathbf{X}_n \end{pmatrix},$$
$$\mathbf{Y} = (Y_1, \ldots, Y_n).$$

- If the matrix $\mathbb{X}^T\mathbb{X}$ is almost singular, then the least-squares estimate will be unstable.
- **Solution**: Add a small quantity along its diagonal.
  - $\mathbb{X}^T\mathbb{X} \to \mathbb{X}^T\mathbb{X} + \lambda I$
  - Bias-Variance trade-off

Ridge regression iii

- The Ridge estimate of β is given by

$$\hat{\beta}_R = (\mathbb{X}^T\mathbb{X} + \lambda I)^{-1}\mathbb{X}^T\mathbf{Y}.$$

Example i

```r
library(ElemStatLearn)
library(tidyverse)

data_train <- prostate %>%
  filter(train == TRUE) %>%
  dplyr::select(-train)
data_test <- prostate %>%
  filter(train == FALSE) %>%
  dplyr::select(-train)
```

Example ii

```r
model1 <- lm(lpsa ~ .,
             data = data_train)
pred1 <- predict(model1, data_test)

mean((data_test$lpsa - pred1)^2)
```

```
## [1] 0.521274
```

Example iii

```r
# glmnet does lasso, elastic-net and
# ridge regression
library(glmnet)
X_train <- model.matrix(lpsa ~ . - 1,
             data = data_train)
model2 <- glmnet(X_train, data_train$lpsa,
                 alpha = 0, lambda = 0.7)
```

Example iv

```r
X_test <- model.matrix(lpsa ~ . -  1,
              data = data_test)
pred2 <- predict(model2, X_test)

mean((data_test$lpsa - pred2)^2)
```

```
## [1] 0.5060843
```

Dual problem i

- The Ridge estimate actually minimises a **regularized** least-squares function:

$$RLS(\beta) = \frac{1}{2}\sum_{i=1}^{n}(Y_i - \beta^T \mathbf{X}_i)^2 + \frac{\lambda}{2}\beta^T\beta.$$

- If we take the derivative with respect to β, we get

$$\frac{\partial}{\partial\beta}RLS(\beta) = -\sum_{i=1}^{n}(Y_i - \beta^T \mathbf{X}_i)\mathbf{X}_i + \lambda\beta.$$

- Setting it equal to 0 and rearranging, we get

$$\beta = \frac{1}{\lambda}\sum_{i=1}^{n}(Y_i - \beta^T \mathbf{X}_i)\mathbf{X}_i.$$

Dual problem ii

- Define $a_i = \frac{1}{\lambda}(Y_i - \beta^T \mathbf{X}_i)$. We then get

$$\beta = \sum_{i=1}^{n} a_i \mathbf{X}_i = \mathbb{X}^T \alpha,$$

  where $\alpha = (a_1, \ldots, a_n)$.
- **Why?** We can now rewrite $RLS(\beta)$ as a function of α. First note that

$$RLS(\beta) = \frac{1}{2}(\mathbf{Y} - \mathbb{X}\beta)^T(\mathbf{Y} - \mathbb{X}\beta) + \frac{\lambda}{2}\beta^T \beta.$$

Dual problem iii

- Now we can substitute $\beta = \mathbb{X}^T\alpha$:

$$\begin{aligned} RLS(\alpha) &= \frac{1}{2}(\mathbf{Y} - \mathbb{X}\mathbb{X}^T\alpha)^T(\mathbf{Y} - \mathbb{X}\mathbb{X}^T\alpha) + \frac{\lambda}{2}(\mathbb{X}^T\alpha)^T(\mathbb{X}^T\alpha) \\ &= \frac{1}{2}(\mathbf{Y} - (\mathbb{X}\mathbb{X}^T)\alpha)^T(\mathbf{Y} - (\mathbb{X}\mathbb{X}^T)\alpha) + \frac{\lambda}{2}\alpha^T(\mathbb{X}\mathbb{X}^T)\alpha. \end{aligned}$$

- This formulation of regularised least squares in terms of α is called the **dual problem**.
- **Key observation**: $RLS(\alpha)$ depends on X_i *only* through the Gram matrix $\mathbb{X}\mathbb{X}^T$.
  - If we all we know are the dot products of the covariates X_i, we can still solve the ridge regression problem.

Kernel ridge regression i

- Suppose we have a transformation $\Phi : \mathbb{R}^p \to \mathbb{R}^N$, where N is typically larger than p and can even be infinity.
- Let K be the $n \times n$ matrix whose (i, j)-th entry is the dot product between $\Phi(\mathbf{X}_i)$ and $\Phi(\mathbf{X}_j)$:

$$K_{ij} = \Phi(\mathbf{X}_i)^T \Phi(\mathbf{X}_j).$$

- **Important observation**: This actually induces a map on pairs of points in $\mathbb{R}^p$:

$$k(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i)^T \Phi(\mathbf{X}_j).$$

- We will call the function k the **kernel function**.

Kernel ridge regression ii

- Now, we can use the dual formulation of ridge regression to fit a linear model between Y_i and the transformed $\Phi(X_i)$:

$$Y_i = \beta^T \Phi(\mathbf{X}_i) + \epsilon_i.$$

- By setting the derivative of $RLS(\alpha)$ equal to zero and solving for α, we see that

$$\hat{\alpha} = (K + \lambda I_n)^{-1}\mathbf{Y}.$$

Kernel ridge regression iii

- Note that we would need to know all the images $\Phi(\mathbf{X}_i)$ to recover $\hat{\beta}$ from $\hat{\alpha}$. On the other hand, we don't actually need $\hat{\beta}$ to obtain the *fitted* values:

$$\hat{\mathbf{Y}} = \Phi(\mathbb{X})\hat{\beta} = \Phi(\mathbb{X})\Phi(\mathbb{X})^T\hat{\alpha} = K\hat{\alpha}.$$

- To obtain the predicted value for a new covariate profile $\tilde{\mathbf{X}}$, first compute all the dot products in the feature space:

$$\mathbf{k} = (k(\mathbf{X}_1, \tilde{\mathbf{X}}), \ldots, k(\mathbf{X}_n, \tilde{\mathbf{X}})).$$

Kernel ridge regression iv

- We can then obtain the predicted value:

$$\begin{aligned}
\tilde{Y} &= \hat{\beta}^T \Phi(\tilde{\mathbf{X}}) \\
&= \hat{\alpha}^T \Phi(\mathbb{X}) \Phi(\tilde{\mathbf{X}}) \\
&= \hat{\alpha}^T \mathbf{k} \\
&= \mathbf{k}^T (K + \lambda I_n)^{-1} \mathbf{Y}.
\end{aligned}$$

Example (cont'd) i

```r
# Let's start with the identity map for Phi
# We should get the same results as Ridge regression
X_train <- model.matrix(lpsa ~ .,
                        data = data_train)
Y_train <- data_train$lpsa
```

Example (cont'd) ii

```
# Ridge regression
beta_hat <- solve(crossprod(X_train) +
                    0.7*diag(ncol(X_train))) %*%
  t(X_train) %*% Y_train

beta_hat[1:3]
```

```
## [1] 0.1323063 0.5709660 0.6160020
```

Example (cont'd) iii

```r
# Dual problem
alpha_hat <- solve(tcrossprod(X_train) +
                     0.7*diag(nrow(X_train))) %*%
  Y_train

(t(X_train) %*% alpha_hat)[1:3]
```

```
## [1] 0.1323063 0.5709660 0.6160020
```

```r
all.equal(beta_hat, t(X_train) %*% alpha_hat)
```

```
## [1] TRUE
```

Important observation

- We assumed that we had an embedding of the data into a higher dimensional space.
- But our derivation only required the **dot products** of our observations in the feature space.
- Therefore, we don't need to explicitly define the transformation.
- All we need is to define a **kernel function**.