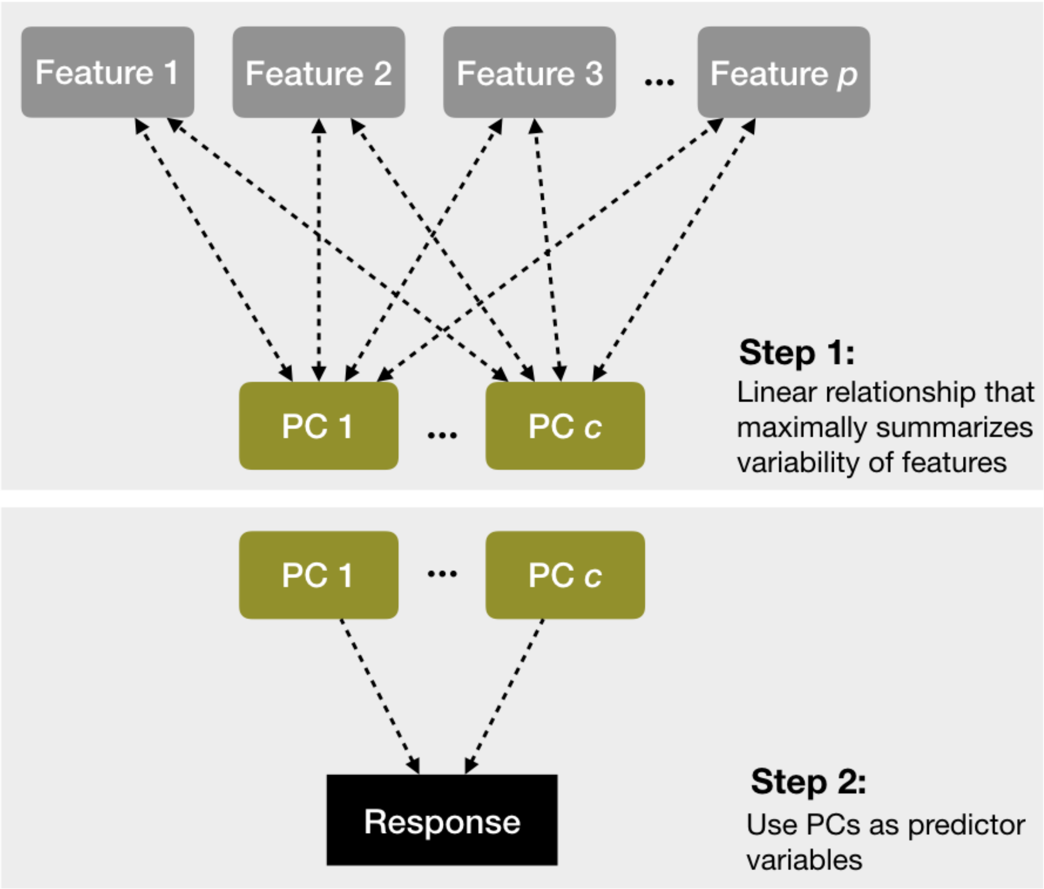
Supervised Learning

Principal component regression and partial least squares

July 6th, 2021

Principal component regression (PCR)



Example data: NFL teams summary

Created dataset using `nflfastR` summarizing NFL team performances from 1999 to 2020

```
library(tidyverse)
nfl_teams_data <- read_csv("http://www.stat.cmu.edu/cmsac/sure/2021/materials/data/regression_pro
nfl_model_data <- nfl_teams_data %>%
  mutate(score_diff = points_scored - points_allowed) %>%
  # Only use rows with air yards
  filter(season >= 2006) %>%
  dplyr::select(-wins, -losses, -ties, -points_scored, -points_allowed, -season, -team)
nfl_model_data
```

```
## # A tibble: 480 × 49
##    offense_com…¹ offen…² offen…³ offen…⁴ offen…⁵ offen…⁶ offen…⁷ offen…⁸ offen…⁹
##            <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1         0.561    3662    1350    6.40    3.28    4284    8.01    1582    4.94
##  2         0.480    2371    2946    5.10    5.56    4698   11.3      942    4.22
##  3         0.612    3435    1667    6.41    3.74    4082    7.88    1391    4.24
##  4         0.564    2718    1555    5.70    3.73    3833    8.91    1243    4.62
##  5         0.569    3264    1674    5.72    4.10    4348    8.07    1553    4.79
##  6         0.525    3286    1940    6.12    4.02    4564    8.90    1374    4.89
##  7         0.588    3827    1648    6.88    3.91    5064    9.76    1466    4.48
##  8         0.565    2893    1347    5.16    3.69    3766    7.43    1533    4.84
##  9         0.569    3838    1954    7.04    4.23    4681    9.38    1427    4.63
```

Implement PCR with `pls` package

Similar syntax to `lm` formula but specify the number of PCs (`ncomp`)

```r
library(pls)
nfl_pcr_fit <- pcr(score_diff ~ ., ncomp = 2, scale = TRUE, data = nfl_model_data)
summary(nfl_pcr_fit)
```

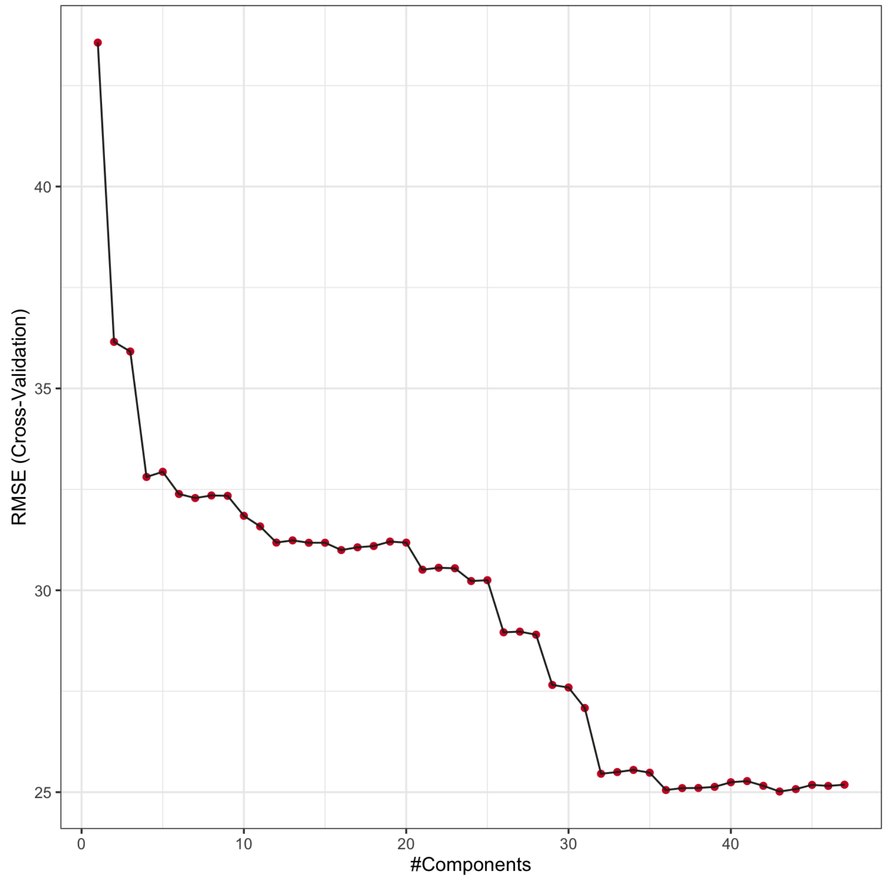
```
## Data:      X dimension: 480 48
##      Y dimension: 480 1
## Fit method: svdpc
## Number of components considered: 2
## TRAINING: % variance explained
##             1 comps  2 comps
## X             21.49    41.62
## score_diff    84.01    87.49
```

Tuning PCR with `caret`

To perform PCR **we need to tune the number of principal components**

- Tune # components in PCR with `caret`
- `train` with 10-fold CV using `pcr` from `pls`

```
set.seed(2013)
library(caret)
cv_model_pcr <- train(
  score_diff ~ .,
  data = nfl_model_data,
  method = "pcr",
  trControl = trainControl(method = "cv", num
  preProcess = c("center", "scale"),
  tuneLength = ncol(nfl_model_data) - 1)
ggplot(cv_model_pcr) + theme_bw()
```



Tuning PCR with `caret`

By default returns model with minimum CV error as `finalModel`

```
summary(cv_model_pcr$finalModel)
```

```
## Data:      X dimension: 480 48
##       Y dimension: 480 1
## Fit method: svdpc
## Number of components considered: 43
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           21.49    41.62    53.04    61.70    66.63    70.79     74.5
## .outcome    84.01    87.49    87.75    89.69    89.69    90.06     90.2
##           8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X           77.81    80.56     83.10     85.27     87.18     88.99     90.43
## .outcome    90.31    90.32     90.66     90.82     91.03     91.05     91.12
##           15 comps  16 comps  17 comps  18 comps  19 comps  20 comps  21 comps
## X            91.85     93.10     94.00     94.76     95.49     96.18     96.79
## .outcome     91.15     91.28     91.28     91.28     91.28     91.34     91.71
##           22 comps  23 comps  24 comps  25 comps  26 comps  27 comps  28 comps
## X            97.30     97.77     98.21     98.57     98.84     99.07     99.29
## .outcome     91.73     91.80     91.95     91.96     92.64     92.72     92.79
##           29 comps  30 comps  31 comps  32 comps  33 comps  34 comps  35 comps
```

Tuning PCR with `caret`

Modify `selectionFunction` in `train` to be the `oneSE` rule

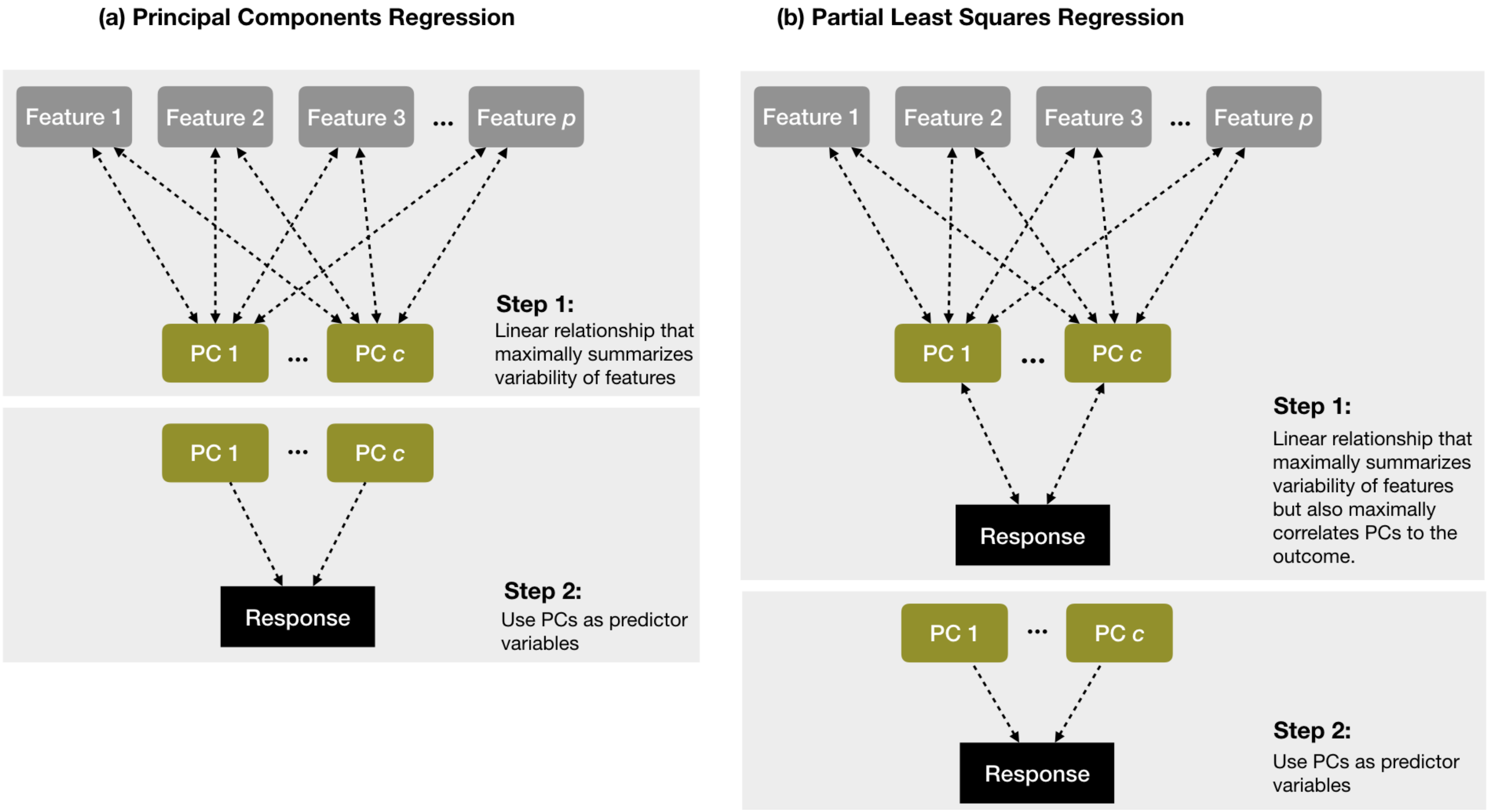
```
set.seed(2013)
cv_model_pcr_onese <- train(
  score_diff ~ .,
  data = nfl_model_data,
  method = "pcr",
  trControl =
    trainControl(method = "cv", number = 10,
                 selectionFunction = "oneSE"),
  preProcess = c("center", "scale"),
  tuneLength = ncol(nfl_model_data) - 1)
```

```
summary(cv_model_pcr_onese$finalModel)
```

```
## Data:      X dimension: 480 48
##       Y dimension: 480 1
## Fit method: svdpc
## Number of components considered: 32
## TRAINING: % variance explained
##             1 comps   2 comps   3 comps   4 comps   5 co
## X             21.49     41.62     53.04     61.70     66
## .outcome      84.01     87.49     87.75     89.69     89
##             8 comps   9 comps  10 comps  11 comps  12
## X             77.81     80.56     83.10     85.27
## .outcome      90.31     90.32     90.66     90.82
##             15 comps  16 comps  17 comps  18 comps
## X              91.85     93.10     94.00     94.76
## .outcome       91.15     91.28     91.28     91.28
##             22 comps  23 comps  24 comps  25 comps
## X              97.30     97.77     98.21     98.57
## .outcome       91.73     91.80     91.95     91.96
```

Partial least squares (PLS)

PCR is agnostic of response variable



PLS as supervised dimension reduction

First principal component in PCA:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

In PLS we set ϕ_{j1} to the coefficient from **simple linear regression** of Y on each X_j

- Remember this slope is proportional to the correlation! $\widehat{\beta} = r_{X,Y} \cdot \frac{s_Y}{s_X}$
- Thus Z_1 in PLS places most weight on variables strongly related to response Y

To compute Z_2 for PLS:

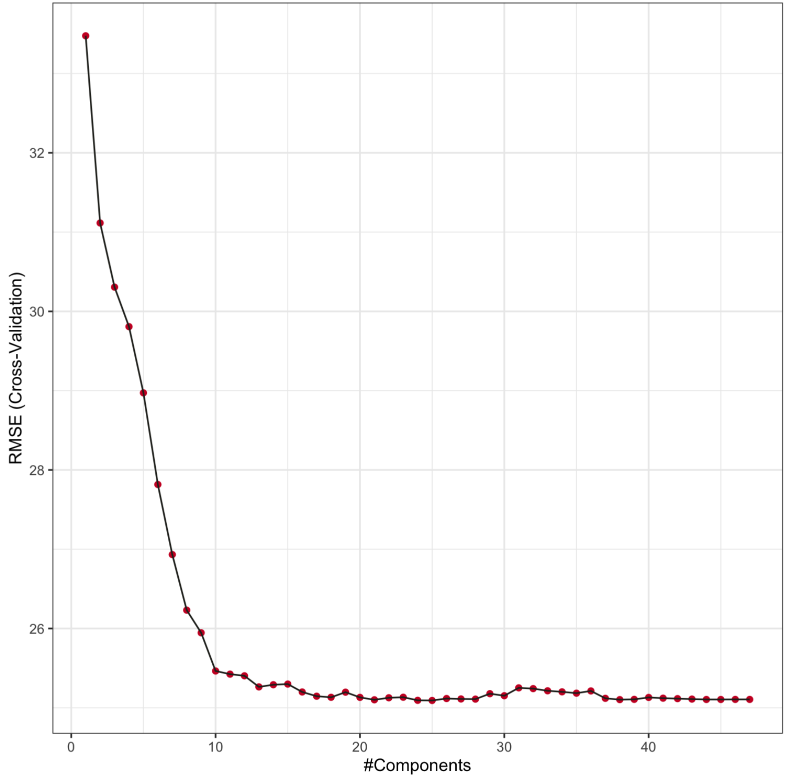
- Regress each X_j on Z_1, residuals capture signal not explained by Z_1
- Set ϕ_{j2} to the coefficient from **simple linear regression** of Y on these residuals for each variable

Repeat process until all $Z_1, Z_2, \ldots, Z_p$ are computed (**PLS components**)

Then regress Y on $Z_1, Z_2, \ldots, Z_p^*$, where $p^* < p$ is a tuning parameter

Tuning PLS with `caret`

```r
set.seed(2013)
cv_model_pls <- train(
  score_diff ~ .,
  data = nfl_model_data,
  method = "pls",
  trControl =
    trainControl(method = "cv", number = 10,
                 selectionFunction = "oneSE"),
  preProcess = c("center", "scale"),
  tuneLength = ncol(nfl_model_data) - 1)
ggplot(cv_model_pls) + theme_bw()
```



Sharp contrast with PCR results!

Fewer PLS components because they are guided by the response variable

But how do we summarize variable relationships without a single coefficient?

Variable importance with `vip` package

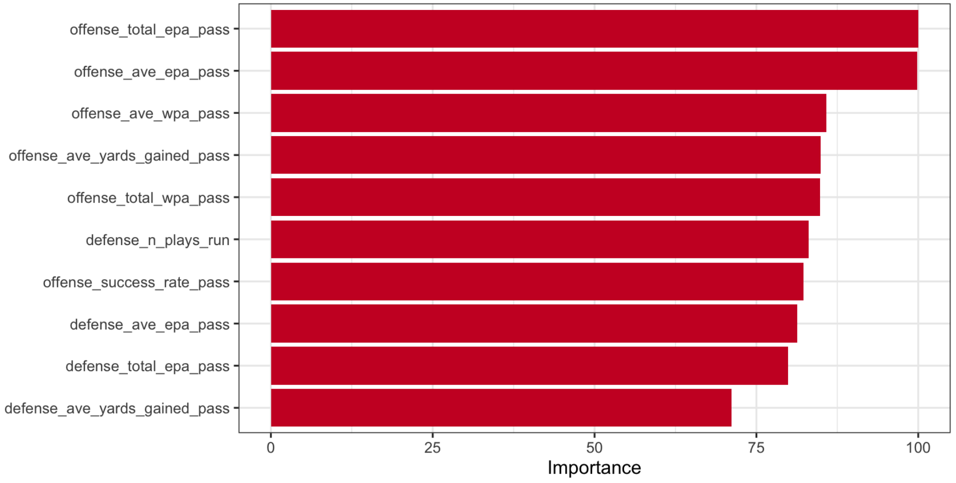
Variable importance attempts to quantify how influential variables are in the model

- e.g., absolute value of t-statistic in regression

For PLS: weighted sums of the absolute regression coefficients across components

- Weights are function of reduction of RSS across the number of PLS components

```
# Check out `cv_model_pls$finalModel$coeffici
library(vip)
vip(cv_model_pls, num_features = 10,
    method = "model") +
  theme_bw()
```



Partial dependence plots (PDP) with pdp package

PDPs display the change in the average predicted response as the predictor varies over their marginal distribution

- More useful for non-linear models later on!

```r
library(pdp)
partial(cv_model_pls, "offense_total_epa_pass", plot = TRUE)
```