VE472 Lecture 8

Jing Liu

UM-SJTU Joint Institute

Summer

true number of features that has something to do with the response

- In this case study, we consider the various models in terms of n, k, and p.
  - linear model `lm`
  - Shrinkage: `ridge`
  - Models using principal components `pcr90` and `pc.ridge90`

components selection and then do ridge

using simulated data, where we have full control over the number of cases n and the number of features k, and the relationships between the features.

- We generate values for the response vector $\mathbf{y}$ using

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

k elements in total

where $\boldsymbol{\varepsilon} \sim \mathrm{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$ for $0 < \sigma < \infty$. Only p element(s) of $\boldsymbol{\beta}$ is nonzero.

- No intercept is included and the test set always contains 500 cases

$$\mathbf{X}^*_{500\times k}, \qquad \text{and} \qquad \mathbf{y}^*_{500\times 1}$$

while the training set has different n and k from one scenario to another.

$$\mathbf{X}^\dagger_{n\times k}, \qquad \text{and} \qquad \mathbf{y}^\dagger_{n\times 1}$$

- We have discuss the motivation and the details of the ridge regression,

$$\arg\min_{\mathbf{b}\in\mathbb{R}^{k+1}} \left\{ \|\mathbf{y} - \mathbf{X}\mathbf{b}\|^2 + P_\lambda(\mathbf{b}) \right\}$$

where P_λ is the penalty condition,

L2 norm

$$P_\lambda(\mathbf{b}) = \lambda\|\mathbf{b}\|^2, \qquad \text{and} \quad 0 \le \lambda < \infty$$

- In general, one could use other penalty conditions:

lasso: $P_\lambda(\mathbf{b}) = \lambda\|\mathbf{b}\|_1$ L1 norm: absolute value

Elastic Net: $P_\lambda(\mathbf{b}) = \lambda\left(\alpha\|\mathbf{b}\|_1 + (1-\alpha)\|\mathbf{b}\|^2\right) \quad \text{for} \quad \alpha \in [0, 1]$

combination of lasso and ridge

SCAD: $P_\lambda(\mathbf{b}) = \sum_{j=1}^{k} p_j(\mathbf{b})$

where $p_j(\mathbf{b}) = \begin{cases} \lambda|\beta_j| & \text{if} \quad |\beta_j| \le \lambda, \\ -\frac{|\beta_j|^2 - 2a\lambda|\beta_j| + \lambda^2}{2(a-1)} & \text{if} \quad |\beta_j| \in (\lambda, a\lambda], \\ \frac{1}{2}(a+1)\lambda^2 & \text{if} \quad |\beta_j| > a\lambda, \end{cases}$ and $2 < a < \infty$.

- Be aware of the memory consumption with respect to n and k.

```
> n = 100000; k = 10;
> DATA=data_generation()
> pryr::object_size(DATA)
```

```
8.85 MB
```

```
> n = 100000; k = 1000;
> DATA=data_generation()
> pryr::object_size(DATA)
```

```
805 MB
```

```
> n = 1000000; k = 100;
> DATA=data_generation()
> pryr::object_size(DATA)
```

大小要小于computer memory

```
808 MB
```

- Next thing to be aware is the bottleneck.

```
> n = 200; k = 100;
> system.time({
+     lm_run = lm(Y~.-1, data = data.df[-(1:500),])
+   })
```

```
   user  system elapsed
  0.007   0.000   0.007
```

```
>  system.time({
+     cv=cv.glmnet(x=DATA$X_train,
+                  y=DATA$Y_train,alpha=0,
+                  foldid=foldid, nlambda=100)
+     ridge_run=glmnet(x=DATA$X_train,
+                      y=DATA$Y_train,alpha=0,
+                      lambda=cv$lambda)
+   })
```

```
   user  system elapsed
  0.104   0.006   0.110
```

ridge: much slower than lse

matrix太大不用leave-one-out cross validation (test set of traing set only contain 1 element

- Notice ℓ-fold CV was used instead of leave-one-out CV,

divide training set to l-components. Totally n/l sets. test set of traing set: one of them (cheaper but less accurate)

```
>   system.time({
+     cv=cv.glmnet(x=DATA$X_train,y=DATA$Y_train,
+                  nfolds = nrow(DATA$X_train),
+                  alpha=0,nlambda=100)
+     ridge_run=glmnet(x=DATA$X_train,
+                      y=DATA$Y_train,alpha=0,
+                      lambda=cv$lambda)
+   })
```

```
  user  system elapsed
 1.641   0.140   1.782
```

LOOCV is about one magnitude slower in this case of $n = 200$ and $k = 100$

- If we increase the data size by 10 times, for example, $n = 2000$ and $k = 100$,

```
  user  system elapsed
 0.209   0.012   0.221
  user  system elapsed
34.161   2.456  36.621
```

- PCA could be another bottleneck but it is nowhere as bad as LOOCV

```
>   system.time({
+     data.pca = data.df %>%
+       select(-Y) %>%
+       prcomp(scale = TRUE)
+   })
```

get principle component: cost little

```
  user  system elapsed
 0.024   0.001   0.025
```

when data size increases, e.g., the following dataset is about 10 times bigger

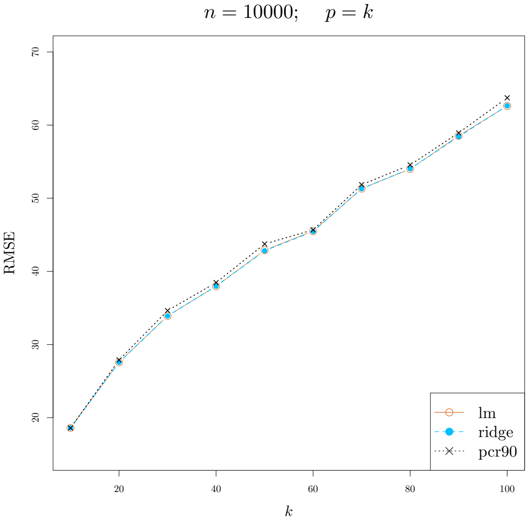
```
>   system.time({
+     data.pca = data.df %>%
+       select(-Y) %>%
+       prcomp(scale = TRUE)
+   })
```

time does not grow very fast

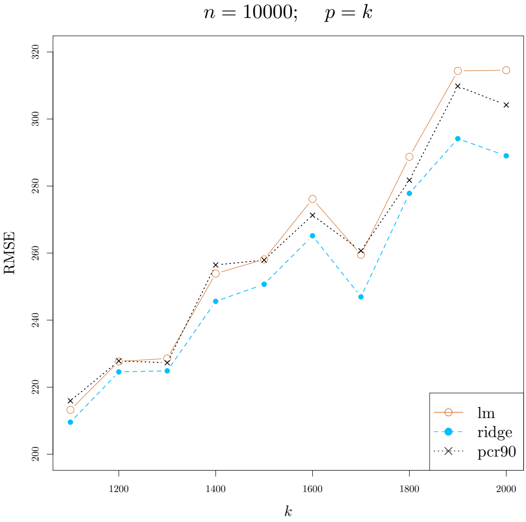
```
  user  system elapsed
 0.073   0.003   0.076
```

Large n and small k with independence between features

sqrt of MSE

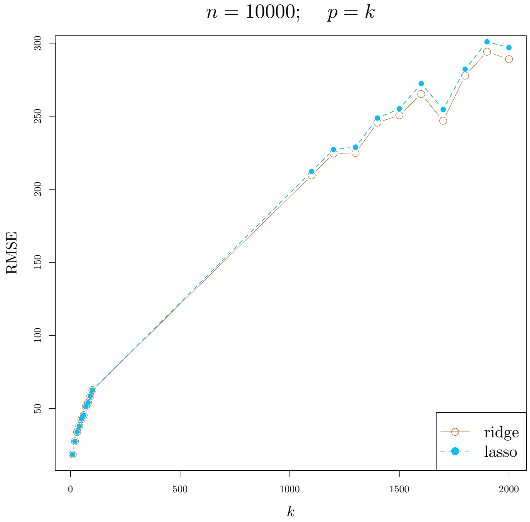


Large n and large k with independence between features



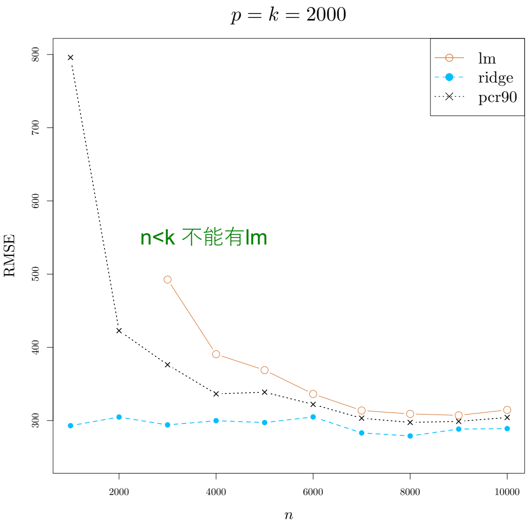
ridge is better

Lasso did marginally worse than ridge for large k

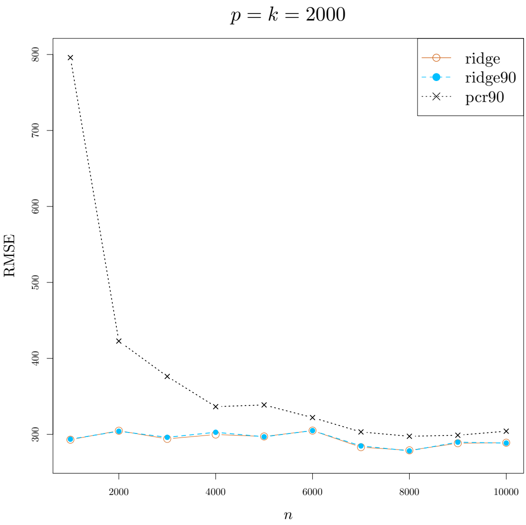


p = k，选了所有的coefficient

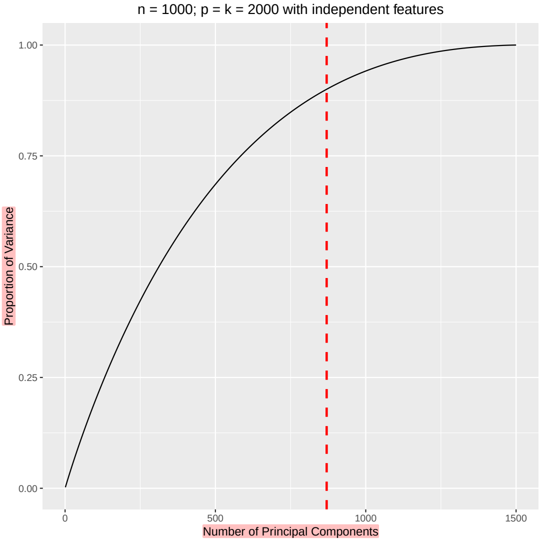
Small n and large k with independence between features



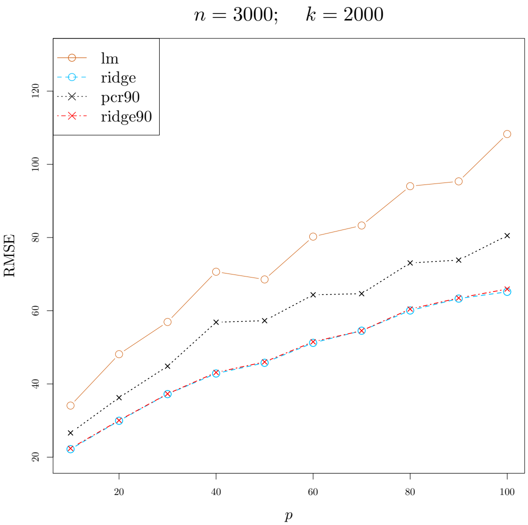
Ridge with principal components



Using principal components that explain 90% variability in $\mathbf{X}$

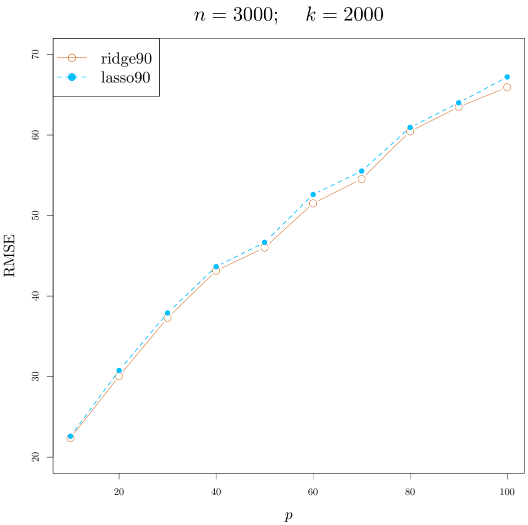


Small n and large k but small p with independent features

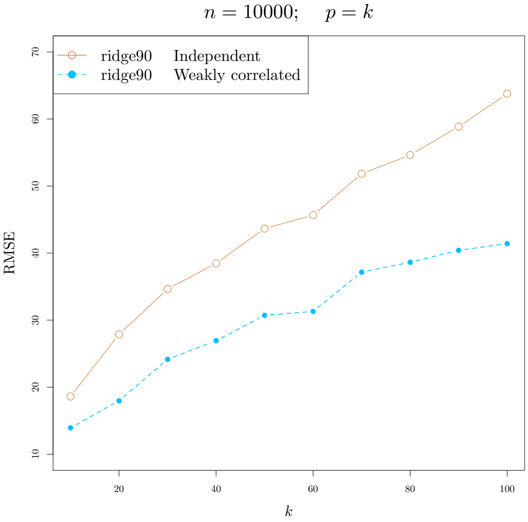


k太大ridge太慢，ridge90更好

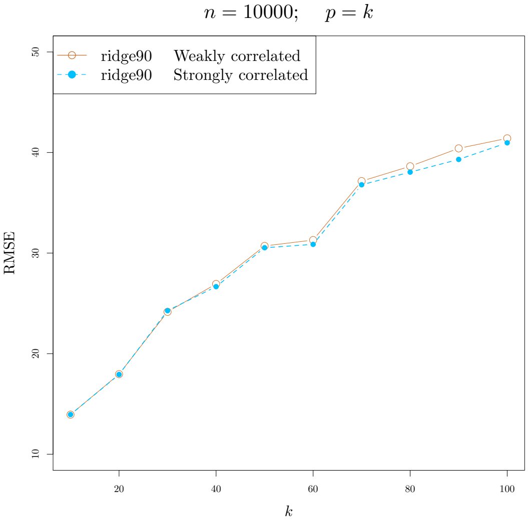
lasso90 did marginally worse than ridge90



Large n and small k with weak correlation between features

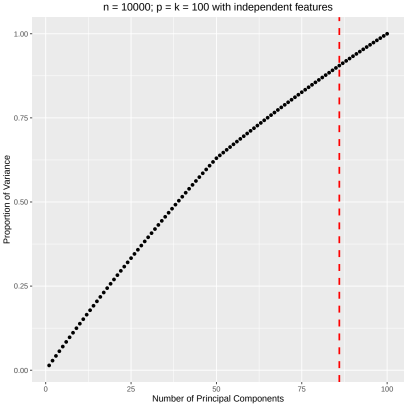


Large n and small k with strong correlation between features

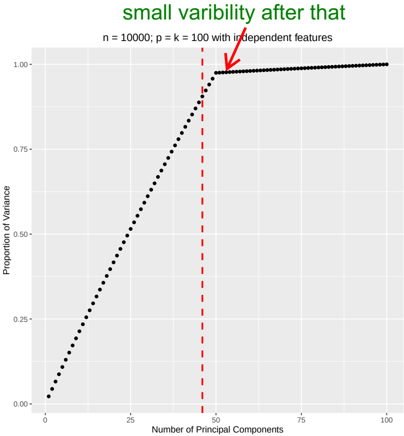


Weak and strong correlation between features

- Notice fewer principal components are needed when correlation is strong.

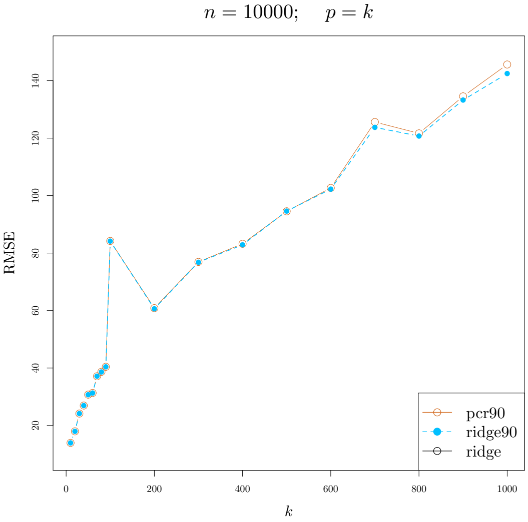


weak correlation

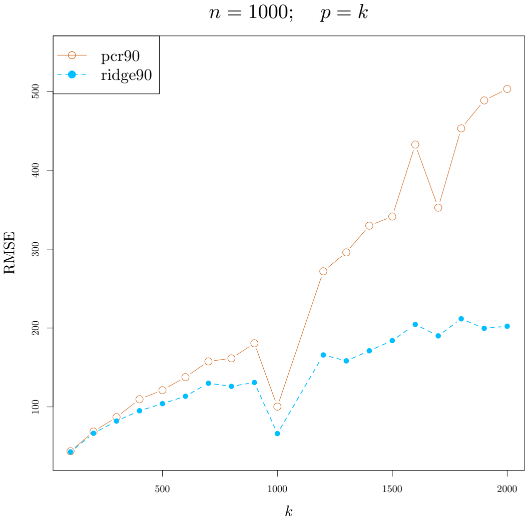


strong correlation

Large n and large k with weak correlation between features



Small n and large k with weak correlation between features



Large k but small p with weak/strong correlation

- The gap in performance seems to be bigger when the correlation is weak.