Machine learning

Regularized regression

Joshua Loftus

Benefits of bias

In "high-dimensions" (p > 2)

Sparse (interpretable?) regression

With the lasso

Regularization

Tuning λ with cross-validation

Inference

Is my high-dimensional model any good?

Bias *can* be good, actually

Especially in higher dimensions

Stein "paradox" and bias

Consider estimating μ from

$$Y_1, \ldots, Y_n \sim N(\mu, \sigma^2 I)$$

The MLE is $\bar{Y}$

The JS estimator shrinks $\bar{Y}$ toward some other point

```
JS_MSE <- function(mu, Z, n, p) {
  Y <- replicate(n, rnorm(p, mean = mu))
  MLE <- rowMeans(Y)
  S <- mean(rowMeans((Y - MLE)^2)^2)
  shrinkage <- max(0, 1 - (p-3) * (S/(n+1)) / sum((MLE-Z)^2
  if (shrinkage == 0) print("yes")
  JS <- Z + (MLE-Z) * shrinkage
  cbind((JS - mu)^2, (MLE - mu)^2)
}
```

James-Stein estimator simulation

Generate and fix μ and $\mathbb{z}$, then simulate many instances

```
JS_MC <- function(n, p, instances = 10000) {
  mu <- rnorm(p) * rpois(p, 3)
  Z <- rnorm(p)
  output <- replicate(instances, JS_MSE(mu, Z, n, p))
  colMeans(apply(output, 2, rowMeans))
}
```


High dimensional? Well, $p > 2$

```
set.seed(42) # reproducibility
```

```
JS_MC(n = 5, p = 4)
```

```
## [1] 0.2015885 0.2023784
```

```
JS_MC(n = 5, p = 4)
```

```
## [1] 0.1977813 0.1985183
```

```
JS_MC(n = 50, p = 4) # larger sample size
```

```
## [1] 0.01987565 0.01989919
```

```
JS_MC(n = 50, p = 4)
```

```
## [1] 0.01968495 0.01969554
```

Higher-dimensional, now $p > n$

```
JS_MC(n = 2, p = 1000)
```

```
## [1] 0.4881896 0.5002847
```

```
JS_MC(n = 2, p = 1000)
```

```
## [1] 0.4858198 0.5002442
```

```
JS_MC(n = 20, p = 100) # larger sample size
```

```
## [1] 0.04986082 0.04999878
```

```
JS_MC(n = 20, p = 100)
```

```
## [1] 0.04969324 0.04988654
```


High-dimensional regression

Estimating parameters in a covariate space rather than the outcome space

Examples with $n > p$ so we can compare OLS to ridge

```
set.seed(42)
instance <- function(X, H, HL, n, p, beta, mu) {
  Y <- mu + rnorm(n)
  y <- Y - mean(Y)
  beta_hat_lm <- H %*% y
  beta_hat <- HL %*% y
  cbind((beta_hat - beta)^2, (beta_hat_lm - beta)^2)
}
```

High-dimensional regression

Fix X and β, hence $\mu = X\beta$, but resample Y

```
high_dim_MSE_MC <- function(n, p, instances = 1000) {
  beta <- rnorm(p) * rpois(p, 3)
  X <- matrix(rnorm(n*p), nrow = n)
  mu <- X %*% beta
  XX <- t(X) %*% X
  H <- MASS::ginv(XX) %*% t(X)
  L <- sqrt(2*log(p)/n)/2
  HL <- MASS::ginv(XX + diag(L, p)) %*% t(X)
  output <- replicate(instances,
                      instance(X, H, HL, n, p, beta, mu))
  colMeans(apply(output, 2, rowMeans))
}
```


MSE(ridge) vs MSE(OLS)

```
high_dim_MSE_MC(n = 100, p = 10)
```

```
## [1] 0.01143915 0.01145554
```

```
high_dim_MSE_MC(n = 100, p = 90)
```

```
## [1] 1.977403 2.738305
```

```
high_dim_MSE_MC(n = 100, p = 90)
```

```
## [1] 0.903344 1.439779
```

```
high_dim_MSE_MC(n = 1000, p = 990, instances = 2000)
```

```
## [1] 0.09942627 0.19938088
```

```
high_dim_MSE_MC(n = 1000, p = 990, instances = 2000)
```

```
## [1] 0.5808623 0.8106186
```

Lessons about bias

Bias can help

- Even in such a basic problem as **estimating the multivariate normal mean** (JS)
- More important in higher dimensions

But it depends! Task-specific

- What's the scientific question?
- Which estimator(s) are we evaluating?
- How will the estimator / ML pipeline be used? For what?

e.g. If $\hat{\sigma}$ underestimates σ it may have lower MSE, but do we care about estimating σ? Or do we care about C.I. coverage?

Interpretable

high-dimensional regression

with the lasso

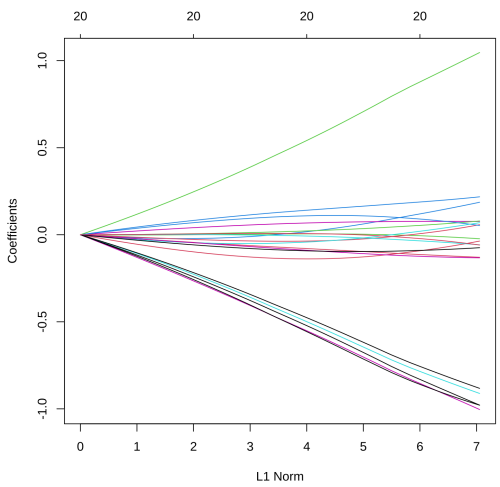
Lasso vs ridge

```
library(glmnet)
library(plotmo) # for plot_glmnet
n <- 100
p <- 20
X <- matrix(rnorm(n*p), nrow = n)
beta = sample(c(-1,0,0,0,1), p, replace = TRUE)
Y <- X %*% beta + rnorm(n)
lasso_fit <- glmnet(X, Y)
ridge_fit <- glmnet(X, Y, alpha = 0)
which(beta != 0)
```

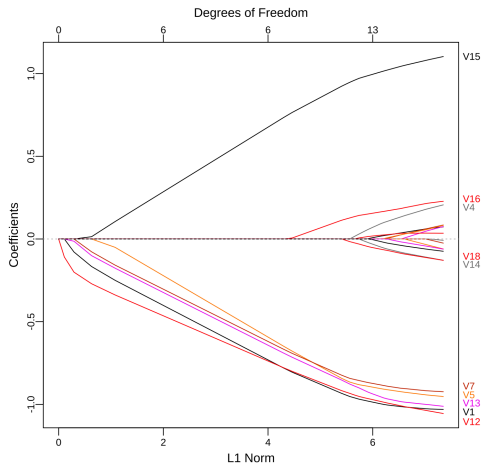
```
## [1]  1  5  7 12 13 15
```


Lasso vs ridge

```
plot(ridge_fit)
```




```
plot_glmnet(lasso_fit, xvar
```




```
## [1]  1  5  7 12 13 15
```

Lasso optimization problem

A simple `diff` to remember lasso/ridge is via the penalty/constraint (1-norm instead of 2-norm). Lasso is

$$\text{minimize } \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\beta\|_2^2 \text{ s. t. } \|\beta\|_1 \leq t$$

where

$$\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$$

Lagrangian form

$$\text{minimize } \frac{1}{2n}\|\mathbf{y}-\mathbf{X}\beta\|_2^2 + \lambda\|\beta\|_1$$

Sparsity of lasso estimators

The L^1 ball in $\mathbb{R}^p$ $\{x : \|x\|_1 = \text{const}\}$ contains

- $2p$ points that are 1-sparse $x_j = \pm 1$, $x_{-j} = 0$
- $\binom{p}{k}2^k$ points k-sparse with elements $\pm k^{-1}$
- Edges connecting (many, not all) pairs of these points
- Higher dimensional "faces" spanning (some) sets of these points, etc

The ellipsoid $\|\mathbf{y}-\mathbf{X}\beta\|_2^2 = \text{const}$ *probably* intersects one of these...

Sparsity of lasso estimators

At the point of intersection of the ellipse and the L^1 ball, the normal vector of the ellipse has to be in the *normal cone* of the L^1 ball (at the same point)

These normal cones are larger (in dimension) at sparser points on the ball (hence more likely to contain the normal vector of the ellipse)

- Normal cones are 1-dimensional on any of the $(p-1)$-dimensional (least sparse) faces
- Normal cones are p-dimensional at the 1-sparse points (most sparse)

Exercise: Prove these two cases (start with a simple def. of normal cone, solve $p = 2$ or 3 first, etc)

KKT optimality conditions

Constrained optimization conditions usually taught in multivariate calculus

- Switch to lagrangian form
- Check stationary points (vanishing gradient)
- Check boundary/singularity points
- Verify feasibility (constraints satisfied)

The Karush-Kuhn-Tucker (KKT) conditions generalize these

Very useful for analysis of constrained optimization problems (i.e. almost all ML methods)

Stationary points

Of the least squares objective (uncorrelated residuals)

$$\frac{1}{n}\mathbf{X}^T(\mathbf{X}\hat{\beta} - \mathbf{y}) = 0$$

For ridge (larger $|\hat{\beta}_j|$ have larger resid. covariance)

$$\frac{1}{n}\mathbf{X}^T(\mathbf{X}\hat{\beta} - \mathbf{y}) = -2\lambda\hat{\beta}$$

For lasso (resid. |covar| = λ if $\hat{\beta}_j \neq 0$ and $\leq \lambda$ otherwise)

$$\frac{1}{n}\mathbf{X}^T(\mathbf{X}\hat{\beta} - \mathbf{y}) = -\lambda\,\mathrm{sign}(\hat{\beta})$$

Lasso treats predictors more "democratically"

L^1 subgradient = sign

Exercise:

Verify that the set-valued sign $s = \mathrm{sign}(\beta)$, defined coordinate-wise as

$$s_j = \begin{cases} 1 & \text{if } \beta_j > 0 \\ -1 & \text{if } \beta_j < 0 \\ [-1, 1] & \text{if } \beta_j = 0 \end{cases}$$

is the subgradient of the L^1 norm function

(part of the point of the exercise is to "recall" the definition of a subgradient!)

Interpreting "interpretable"

Usual linear model interpretation of coefficients

If the conditional expectation function (CEF) is *linear*

$$f(\mathbf{x}) = \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}] = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

Then

$$\hat{\beta}_j \approx \frac{\partial}{\partial x_j} \mathbb{E}[\mathbf{Y}|\mathbf{X} = \mathbf{x}]$$

"Change in CEF *holding other variables constant*"

Small set of **other variables** $\rightarrow$ easy (human) understanding

Lessons about sparsity

Sparsity helps with interpretation

and solving otherwise impossible problems

Curse of dimensionality / NP-hard optimization (best subsets) / unidentifiable statistical estimation / **overfitting vs generalization** (next)

Need special mathematical structure like sparsity to make things tractable

Empirical risk

is a biased estimator

that understimates risk

Generally, more optimization $\rightarrow$ more bias

But! Formulas for the bias in certain cases

Optimism / generalization gap

Observation: training error generally appears lower than test/validation error. Why?

Risk minimization vs *empirical* risk minimization

$$R(g) = \mathbb{E}_F[L(\mathbf{X}, Y, g)]$$

$$\hat{f} = \arg\min_g \hat{R}(g) = \arg\min_g \frac{1}{n} \sum_{i=1}^{n} L(\mathbf{x}_i, y_i, g)$$

Fact: for some $\mathrm{df}(\hat{f}) > 0$ (depends on problem/fun. class)

$$\Delta = \mathbb{E}_{Y|\mathbf{x}_1,\ldots,\mathbf{x}_n}[R(\hat{f}) - \hat{R}(\hat{f})] = \frac{2\sigma_\varepsilon^2}{n}\mathrm{df}(\hat{f}) > 0$$

Optimism and degrees of freedom

Linear case

If $\hat{f}$ is linear with p predictors (or p basis function transformations of original predictors) then

$$\mathrm{df}(\hat{f}) = p, \text{ so } \Delta = 2\sigma_\varepsilon^2 \frac{p}{n}$$

Fairly general case

For many ML tasks and fitting procedures

$$\mathrm{df}(\hat{f}) = \frac{1}{\sigma_\varepsilon^2} \mathrm{Tr}[\mathrm{Cov}(\hat{f}(\mathbf{X}), \mathbf{y})] = \frac{1}{\sigma_\varepsilon^2} \sum_{i=1}^{n} \mathrm{Cov}(\hat{f}(\mathbf{x}_i), y_i)$$

Lasso degrees of freedom

The "0-norm" (not really a norm) counts sparsity

$$\|\beta\|_0 = \sum_{j=1}^{p} \mathbf{1}_{\beta_j \neq 0} = |\{j : \beta_j \neq 0\}|$$

e.g. for OLS with deterministic choice of variables

$$\mathrm{df}(\hat{\beta}) = \|\hat{\beta}\|_0$$

Surprisingly, under fairly weak conditions on $\mathbf{X}$ (columns in general position), for the lasso solution $\hat{\beta}(\lambda)$ at any λ

$$\mathbb{E}[\|\hat{\beta}(\lambda)\|_0] = \mathrm{df}(\hat{\beta}(\lambda))$$

Solution sparsity is unbiased estimate of df, and same as OLS!

Choosing/penalizing complexity

Idea: if we have a formula to estimate optimism/gen. gap then change the loss function to

$$\text{minimize } \hat{L}_\Delta(g) = \hat{R}(g) + \hat{\Delta}(g)$$

e.g. C_p, AIC, BIC, etc

Problem: optimization bias, if $\hat{g}$ is the minimizer then

$$\mathbb{E}[\hat{L}_\Delta(\hat{g})] < L_\Delta(\hat{g})$$

Again, more optimization (larger model search) $\rightarrow$ more bias

Benefit: may be more reproducible than random data splitting

Lessons about optimism and generalization

- Empirical risk underestimates actual risk (expected loss on new sample from same distribution)
- Magnitude of the bias is the optimism / generalization gap
- Optimism generally increases with function class complexity
    - e.g. for linear functions, increases linearly in p
- For a fixed function class, optimism decreases linearly in n
- More optimization $\rightarrow$ overfitting $\rightarrow$ more optimism

Estimate test error directly

using test data

i.e. a new set of data, "unseen" by $\hat{f}$

Indep. samples $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ and $D' = \{(\mathbf{x}'_i, y'_i)\}_{i=1}^{n'}$

Estimate $\hat{f}$ on D, evaluate $\hat{f}$ on D'

Motives: deploying an algorithm "in production"

Actual practice at some organizations

Fit model at some time

Use model for predictions on new data

Possibly re-fit/update model periodically

Motives: phil. of science, novelty, and severity

Philosophy of science: prediction vs "accommodation"

Prediction: happens in time before observation/measurement

Accommodation: theory built to explain past observation/data

Accurate prediction is better evidence in favor of a scientific theory than mere accommodation

ML: What's better evidence in favor of the model?

Popper and Lakatos: **temporal novelty**

Zahar, Gardner, Worrall: **use-novelty** (or problem novelty)

Mayo: novelty is not necessary. **Severity** is necessary

Choosing model complexity

Using test/validation data

Indep. samples $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and $D' = \{(\mathbf{x}'_i, y'_i)\}_{i=1}^{n'}$

- Estimate $\hat{f}_\lambda$ on D for a "path" or grid of λ values
- Evaluate $\hat{f}_\lambda$ on D' and choose $\hat{\lambda}$ accordingly (e.g. with minimum loss)
- Refit $\hat{f}_{\hat{\lambda}}$ on full data $D \cup D'$, this is our final model

Cross-validation

Idea: swap D and D' in previous process and get two estimates, $\hat{R}(\hat{f}_\lambda)$ and $\hat{R}(\hat{f}'_\lambda)$

Average these and choose $\hat{\lambda}$ using the average (e.g. minimizer)

Idea: apply the same process with multiple independent "folds" of data

K-fold cross-validation

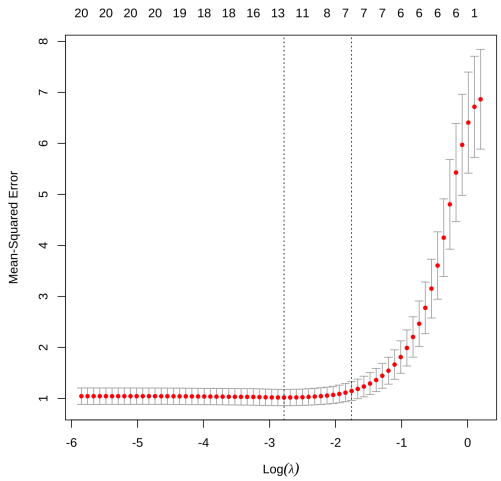
Each subset used once as test set, and $K-1$ times for training

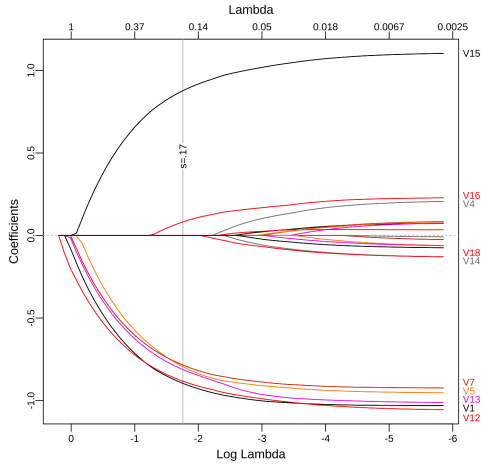
Minimize $\hat{R}_{K\text{-cv}}(\lambda) = \frac{1}{K}\sum_{k=1}^{K}\hat{R}_k(\hat{f}^{(k)}_\lambda)$

plot(cv.glmnet) and plot(glmnet)

```
cv_fit <- cv.glmnet(X, Y)
plot(cv_fit)
```

```
plot_glmnet(lasso_fit,
    s = cv_fit$lambda.1se)
```





Lessons about cross-validation

- Think of it as a way to **choose model complexity**
- **Beware** common cross-validation errors! From Duda and Hart quoted in MLstory

> ... the same data were often used for designing and testing the classifier. This mistake is frequently referred to as "testing on the training data." A related but less obvious problem arises *when a classifier undergoes a long series of refinements guided by the results of repeated testing on the same data. This form of "training on the testing data" often escapes attention until new test samples are obtained*.

Lessons about cross-validation

- **Beware** common cross-validation errors! From ESL, also quoted in MLstory

> Ideally, the test set should be kept in a "vault," and be brought out only at the end of the data analysis. *Suppose instead that we use the test-set repeatedly, choosing the model with smallest test-set error. Then the test set error of the final chosen model will underestimate the true test error*, sometimes substantially.

- Cross-validate the entire pipeline, not just one step
- Choosing K: larger $\rightarrow \hat{R}_{K\text{-cv}}$ has lower bias, more variance

Inference

for high-dimensional regression

We have used regularization to avoid overfitting

But this results in bias, e.g. $||\hat{\beta}||$ is smaller

Inference must correct for this somehow

Approaches to inference

- Debiased inference
- Selective inference
- Post-selection inference
- Stability selection

R packages for some of these

Topic for future study? 😄

One example

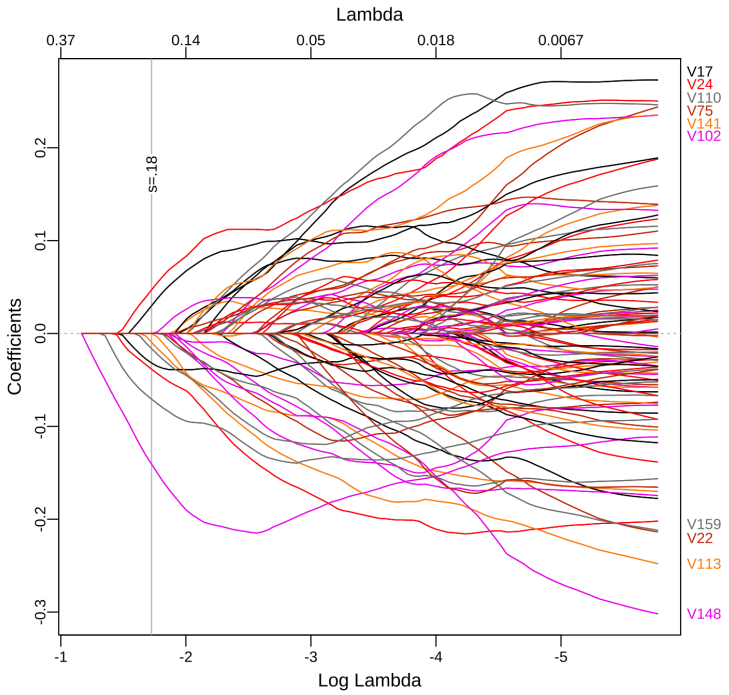
```
set.seed(1)
n <- 100; p <- 200
X <- matrix(rnorm(n*p), nrow = n)
beta = sample(c(-1, rep(0, 20), 1), p, replace = TRUE)
Y <- X %*% beta + rnorm(n)
```

Cross-validation plot (next slide)

```
beta_hat <- coef(lasso_fit, s = cv_lasso$lambda.min)[-1]
vars <- which(beta_hat != 0)
vars
```

```
## [1]  24  34  43  90 111 125 156 168
```

Idea: since $\hat{\beta}$ is biased by penalization, how about refitting OLS (unpenalized) using only these variables?



```
summary(lm(Y ~ X[,vars]-1))
```

```
##
## Call:
## lm(formula = Y ~ X[, vars] - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5978 -0.2385  0.2125  0.7718  2.6094
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## X[, vars]1  0.23733    0.08711   2.725  0.00771 **
## X[, vars]2 -0.13354    0.08056  -1.658  0.10079
## X[, vars]3 -0.16768    0.07682  -2.183  0.03160 *
## X[, vars]4 -0.30632    0.08171  -3.749  0.00031 ***
## X[, vars]5 -0.13511    0.07967  -1.696  0.09330 .
## X[, vars]6  0.18805    0.08077   2.328  0.02209 *
## X[, vars]7 -0.14805    0.08157  -1.815  0.07279 .
## X[, vars]8 -0.15296    0.08071  -1.895  0.06121 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8091 on 92 degrees of freedom
## Multiple R-squared:  0.3747,    Adjusted R-squared:  0.3203
## F-statistic: 6.891 on 8 and 92 DF,  p-value: 4.501e-07
```


Looks good, time to publish!

- Sparse, interpretable model
- Some significant predictors
- Decent R^2 value showing predictive accuracy

Pretty good... hey wait, what was this line in the code...

```
Y <- rnorm(n)
lasso_fit <- glmnet(X, Y)
cv_lasso <- cv.glmnet(X, Y)
```

Model was *actually* fit on pure noise

😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱, 😱

Idea: compute inferences (`summary`) on new validation data

Lessons about high-dimensional regression

- Can fit to noise, even with cross-validation
- Theoretical results

Lasso "performs well" (prediction error, estimation error, sparse support recovery) under various sets of sufficient conditions, derived/proven using KKT conditions and probability bounds (see SLS Chapter 11)

Roughly:

- $\mathbf{X}$ has to be well-conditioned in some sense
- True β has to be sparse enough
- Solution still generally includes some false positives

References

Optimism / generalization gap (ESL 7.4-6)

Covariance penalty and degrees of freedom (CASI 12.3)

Cross-validation (ESL 7.10)

SLS 2 for lasso, cross-validation, degrees of freedom

CASI 16 for lasso

CASI 20 / SLS 6 for inference after model selection

SLS 5.2 for KKT conditions

SLS 11 for theoretical resuls about lasso