STA2311: Advanced Computational Methods for Statistics I

Class 1: Validation

Radu Craiu   Robert Zimmerman

University of Toronto

September 12, 2023

Section 1

About This Course

Statistical Computation

- For a researcher, *statistical computation* is both a field of study and a tool
- For example: classical inference requires optimization strategies, while Bayesian analysis requires sophisticated sampling techniques
- Both areas (optimization and sampling) are put under pressure by increasingly large datasets and intractable likelihoods
- A graduate student must be able to apply these techniques efficiently and correctly
    - And also be able to tweak them when needed
- For the latter, they need to "look under the hood" and understand the principles behind most algorithms

In This Course

- We will look at some of the most widely used optimization and sampling algorithms
- We will illustrate concepts with examples in R, but emphasis will be placed on understanding techniques and principles rather than programming
- So there will be hands-on practice problems, but also theoretical questions

Course Structure

- The course is roughly divided into two parts
- The first five classes are about basic inference and classical optimization techniques, while the remaining six pertain to sampling
- A class for the midterm will separate the two halves

R

- We will be doing all of our programming in R:

```r
set.seed(2311)

s.means <- rep(0, length=1000)

for (j in 1:1000) {
  s.means[j] <- mean(rpois(n=100, lambda=2))
}

hist((s.means - mean(s.means))/sd(s.means))

shapiro.test(s.means)
```

- You will need a computer with fairly good specs for computations later in the course

Simulating with R

- R allows simulation from most standard distributions (uniform, normal, gamma, *t*, beta, Poisson, binomial, Dirichlet, etc.)

```r
y <- rnorm(n=5, mean=0, sd=1)
print(round(y, 2))
```

```r
y <- rbinom(n=5, size=19, prob=0.55)
print(y)
```

```r
y <- sample(c(1,2,3), size=5, replace=T, prob=c(0.1, 0.3, 0.6))
print(y)
```

Programming with R

- It is essential that you have a solid grasp of R programming
  - Including vectorization!
- Rob will present a quick R bootcamp if you're rusty with the basics (date TBD)
- If/when you already have basic knowledge: "The R Inferno" [Burns, 2012]
- More advanced reading: "Advanced R" by Hadley Wickham [Wickham, 2019]

RStudio

- It is highly recommended that you use RStudio Desktop, which is a popular IDE for R
- RStudio supports Python, C, C++, etc.
- It includes countless features for productivity and is very well supported
- RStudio supports R Markdown, which you will use for your course assignments
  - Also useful/required for other courses (e.g., STA2101H)
- These slides were prepared using RStudio!

Section 2

Model Validation and Comparison

Model Validation and Comparison

- Model validation is a hugely important aspect of statistical inference
  - Sometimes you can think of this as checking model calibration: confidence intervals must have the correct coverage, p-values under the null hypothesis have the correct distribution ($U(0, 1)$), etc.
  - Model validation is most often performed using simulations
- Model comparison has to do with selecting the best model in a set (although all could be pretty bad)
  - Can be based on selection criteria (AIC, BIC, WAIC, etc)
  - Can be done via cross-validation
  - Can be done via simulation
- Different criteria: estimating bias and variance, predictive accuracy, testing power, robustness

Computing Bias and Variance

- Recall: let $\tilde{\theta}(\boldsymbol{Y})$ be an estimator of θ based on data $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$
  - For simplicity we view θ as a scalar here, but for most of the course we will deal with vector parameters
- The *bias* of $\tilde{\theta}(\boldsymbol{Y})$ is defined as

$$\text{Bias}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right) = \mathbb{E}_\theta\left[\tilde{\theta}(\boldsymbol{Y})\right] - \theta$$

- The *mean-squared error (MSE)* of $\tilde{\theta}(\boldsymbol{Y})$ is defined as

$$\text{MSE}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right) = \mathbb{E}_\theta\left[\left(\tilde{\theta}(\boldsymbol{Y}) - \theta\right)^2\right],$$

  where the expectation is taken with respect to $\boldsymbol{Y}$ generated by θ
- The *root-mean-square error (RMSE)* of $\tilde{\theta}(\boldsymbol{Y})$ is defined as

$$\text{RMSE}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right) = \sqrt{\text{MSE}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right)}$$

The Bias-Variance Tradeoff

- The *bias-variance tradeoff* states that

$$\mathrm{MSE}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right) = \mathrm{Bias}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right)^2 + \mathrm{Var}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right)$$

- Thus

$$\mathrm{RMSE}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right) = \sqrt{\mathrm{Bias}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right)^2 + \mathrm{Var}_\theta\left(\tilde{\theta}(\boldsymbol{Y})\right)}$$

- In other words: for a fixed MSE or RMSE, we cannot reduce the bias of an estimator without increasing its variance (or vice versa)

Accuracy Measures

- Note that the MSE can be computed or estimated when θ is known, which leads to the idea of generating data given parameter values (i.e., generating synthetic data)
  - We will discuss methods of generating synthetic data soon
- For certain validation criteria, we want the synthetic data to be similar to the observed data
- Want to fit the model of interest to these synthetic data sets many times
- When generating n synthetic datasets using the parameter value θ, leading to n estimates $\hat{\theta}_1, \ldots, \hat{\theta}_n$, then the (empirical) RMSE is

$$\text{RMSE}(\hat{\theta}_1, \ldots, \hat{\theta}_n) = \sqrt{\frac{\sum_{i=1}^{n} (\hat{\theta}_i - \theta)^2}{n}}$$

  - When the context is clear, we call this the RMSE as well

Predictive Modelling

- The bias-variance theory above adapts naturally to predictive modelling
- Assume we are interested in predicting values for n scalar observations
- The observed values are $y_1, \dots, y_n$ and a model makes predictions $\hat{y}_1, \dots, \hat{y}_n$
- The RMSE here is then defined as

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

- The term $(y_i - \hat{y}_i)^2$ can be replaced by another *loss function* $L(y, y_i)$ when appropriate
  - The RMSE uses the *quadratic loss* $L(y, \hat{y}) = (y_i - \hat{y})^2$
  - If the y_i's are binary, one might use the 0-1 loss, for which $L(y, \hat{y}) = 1 - \mathbb{1}_{y-\hat{y}} = \mathbb{1}_{y \neq \hat{y}}$

Example: RMSE for Linear Regression

```r
set.seed(2311)

n <- 1000

# design matrix
dat <- data.frame(X1=rnorm(n=n),
                  X2=rbinom(n=n, size=1, prob=0.2),
                  X3=rpois(n=n, lambda=0.7))

# noisy linear response
dat$y <- 0.4 + 0.7*dat$X1 + 3*dat$X2 - dat$X3 + rnorm(n=n, sd=2)

mod <- lm(y ~ ., data=dat) # fit basic linear model
y.hat <- predict.lm(mod) # make predictions

RMSE <- sqrt(mean((dat$y-y.hat)^2))
```

Section 3

Simulation for Validation

Basic Simulation

- Simulation itself is one of the most useful tools for model validation and comparison
- For now, we only note some basic facts about simulation
- For today's simple examples we rely on classical Monte Carlo:
- In order to estimate

$$I = \mathbb{E}_{\boldsymbol{\theta}}[h(X)] = \int h(x) f_{\boldsymbol{\theta}}(x)\,\mathrm{d}x$$

  for some density/pmf $f_{\boldsymbol{\theta}}(x)$, we sample $X_1, \ldots, X_m \overset{iid}{\sim} f_{\boldsymbol{\theta}}$ and use the estimator

$$\hat{I} = \frac{1}{M} \sum_{i=1}^{M} h(X_i)$$

Proof of Principle Via Simulation

- Given a model $X \sim f_{\boldsymbol{\theta}}$ and data $X_1, \ldots, X_n$, we can produce an estimator $\hat{\boldsymbol{\theta}}(\boldsymbol{X})$
- Usually theory provides asymptotic properties for $\hat{\boldsymbol{\theta}}$, but we are interested in *finite sample properties*, which we can study by simulation
- First we choose a specific $\boldsymbol{\theta}_0$ of interest and a number of simulations M
- Then for $1 \leq i \leq M$, we do the following:
  1. Generate data $X_1^{(i)}, \ldots, X_n^{(i)} \sim f_{\boldsymbol{\theta}_0}$ (these make up one synthetic dataset)
  2. Compute $\hat{\boldsymbol{\theta}}_i := \boldsymbol{\theta}(\boldsymbol{X}^{(i)})$
- The properties of $\hat{\boldsymbol{\theta}}(\boldsymbol{X})$ are explored using the empirical distribution of $\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}}_M$

Example: Toxicity Data

- Consider the following toxicity data, in which x_i is the dose of some toxin administered to the i'th of 9 samples of individuals and y_i is the number of resulting dead:

$$\boldsymbol{x} = (1, 1, 2, 2, 2, 2, 3, 3, 3)$$
$$\boldsymbol{y} = (2, 3, 4, 7, 9, 9, 10, 12, 15)$$

- We assume a standard Poisson regression model in which $Y_i \mid x_i \sim \text{Poisson}(\lambda(x_i))$ and $\log(\lambda(x_i)) = \beta_0 + \beta_1 x_i$
- The model is easily fit using `glm` in R
- For such a small sample, we question the variance estimates and the coverage of confidence intervals (why?)

Example: Toxicity Data (Continued)

```r
x <- c(1,1,2,2,2,2,3,3,3)
y <- c(2,3,4,7,8,9,10,12,15)
mod <- glm(y ~ x, family = poisson(link="log"))
summary(mod)$coefficients[,1:2]
```

```
##              Estimate Std. Error
## (Intercept) 0.4494390  0.4643537
## x           0.6992008  0.1826136
```

- An approximate 95% confidence interval for β_1 is then $(0.341, 1.057)$
- But this is based on the *asymptotic variance* obtained from the observed Fisher information!
- We use simulation to check whether this CI provides the correct coverage

Which Replication Design?

- We need to decide between a *fixed design* model (where the $\boldsymbol{x}$'s are fixed) or a *random design* model (where the $\boldsymbol{x}$'s are drawn from a population)
- This decision is based on the problem's specifics, the inferential focus, etc.
- If the $\boldsymbol{x}$'s represent fixed levels of an experiment and there is no need to study the performance of the drug for other values, then a fixed design is appropriate
  - For example, $\boldsymbol{x}$ might represent dosages that are commonly used in a population
- If the $\boldsymbol{x}$'s are covariate values sampled from a large population, then conclusions are more widely applicable if we include new values of $\boldsymbol{x}$ in the synthetic datasets, and so a random design is appropriate
  - For example, $\boldsymbol{x}$ could be the salary of a UofT graduate and y the size of the graduate's family

Fixed and Random Designs

- In a fixed design setup, we choose values of $\boldsymbol{\theta}$ and generate $Y \sim f_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})$
- In a random design setup:
  1. Choose parameter values for the distribution of $\boldsymbol{x}$ (say ϕ), and generate $\boldsymbol{x} \sim g_{\phi}$
  2. Choose values of $\boldsymbol{\theta}$ and generate $Y \sim f_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{x})$
- For the toxicity data example we follow the random design approach

Example: Toxicity Data (Continued)

- The plan: simulate many $\boldsymbol{x}$'s, and then many $\boldsymbol{Y} \mid \boldsymbol{x}$'s based on the proposed model
  - We assume the x_i's follow a Poisson distribution with mean $\bar{x} = 19/9 \approx 2.11$
- Then compare empirical variance and CIs with the estimated ones
- For $1 \leq j \leq M$:
  1. Sample $x_1^{(j)}, \dots, x_9^{(j)} \overset{iid}{\sim} \text{Poisson}(2.11)$ and for $1 \leq i \leq 9$, sample $Y_i^{(j)} \mid x_i^{(j)} \sim \text{Poisson}(\hat{\lambda}(x_i^{(j)}))$ where $\log\left(\hat{\lambda}(x)\right) = 0.45 + 0.70x$
  2. Fit the Poisson regression model with data $[\boldsymbol{Y}^{(j)}, \boldsymbol{x}^{(j)}]$ and save the estimates of β_1

Example: Toxicity Data (Continued)

```r
set.seed(2311)

M <- 1000
ci.ind <- rep(0, M)
beta1.emp <- rep(0, M)

for (i in 1:M) {
  x <- rpois(9, 2.11)
  y <- rpois(9, exp(0.45 + 0.70*x))
  mat <- summary(glm(y~x, family=poisson(link="log")))$coefficients
  beta1.emp[i] <- mat[2,1]
  ci.L <- beta1.emp[i] - 1.96*mat[2,2]
  ci.U <- beta1.emp[i] + 1.96*mat[2,2]
  ci.ind[i] <- 1*(ci.L < 0.70 && 0.70 < ci.U)
}

c(mean(ci.ind), sqrt(var(beta1.emp)))
```

```
## [1] 0.9490000 0.1181653
```

Other Types of Constraints

- Suppose that we are interested in creating synthetic datasets which preserve the values of the response variable $Y = y$
- Then we must simulate from the conditional distribution of $X \mid y$ generated by some pre-specified parameter values α and ϕ
  - Say α specifies the conditional distribution of $X \mid y$ and ϕ specifies the marginal distribution of X
- We have $p_\alpha(y \mid X)$ and $p_\phi(X)$, and we want to sample from $p_{\alpha,\phi}(X \mid y) \propto p_\alpha(y \mid X) \cdot p_\phi(X)$
- We may also want to sample α itself given some $p(\alpha)$
- We will learn how to do this later

Section 4

Robustness Studies

Model Assumptions

- Every statistical model relies on a set of assumptions
- Usually, these are structural and/or stochastic
- For a regression model with covariates $\boldsymbol{x} = (x_1, \ldots, x_p)$ and $\boldsymbol{\beta} \in \mathbb{R}^p$, structural constraints can include...
    - Linear dependence: $E[Y] = \eta(\boldsymbol{x})$ with $\eta(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{x}$
    - Additive models: $E[Y] = \sum_{i=1}^p f_i(x_i)$ with $f_i(x_i) = \cdots$
    - Single index models: $E[Y] = f(\eta(\boldsymbol{x}))$
    - Many nonparametric models assume smoothness
- Parametric assumptions
    - Gaussian noise
    - Poisson regression
    - Logistic/probit regression

Robustness Studies

- A robustness study will explore the performance of a model under various violations of its assumptions
- Theoretical results are limited, so most of the results are empirical
  - i.e., based on simulations
- We create disturbances in the model and study the performance of the estimators

Robustness Study Techniques

- Techniques for testing structural assumptions include...
  - Modifying the structure of the generating model (e.g., use $f(x_1, x_2) = x_1 \cdot x_2 + f(x_1) + f(x_2)$)
  - Introducing dependence (when independence is assumed)
  - Adding new covariates with spurious or real effects
  - Investigating effects of missing data (ignorable and non-ignorable missingness)

- Techniques for testing parametric assumptions include..
  - Changing the distribution of errors
  - Contaminating distributions (using mixtures)
  - Changing the lightness of tails
  - Swapping symmetry for skewness

Example: Toxicity Data (Continued)

- We can also compare the empirical variance and CIs with the estimated ones under *model misspecification*
- That is, we generate the simulated data under a different model than that used originally
- For example, for each $1 \leq i \leq 9$ we can generate $Z_i \mid x_i \sim \text{NegBin}(r, p(x_i))$ where $p(x) = \frac{1}{1+e^{\eta(x)}}$ and $\eta(x) = \alpha_0 + \alpha_1 x$
  - Recall: $Z \sim \text{NegBin}(r, p)$ can be interpreted as the number of failures that occur before r successes (with success probability p) are encountered; we use $r = 1$
- True model: $\log(\mathbb{P}(Z_i = k \mid x_i)) = k \cdot \eta(x_i) - (k+1) \cdot \log\left(1 + e^{\eta(x_i)}\right)$
- Fitted model: $\log(\mathbb{P}(Y_i = k \mid x_i)) = k \cdot \eta(x_i) - e^{\eta(x_i)} - \log(k!)$

Example: Toxicity Data (Continued)

```r
set.seed(2311)

M <- 1000
x <- c(1,1,2,2,2,2,3,3,3)
y <- c(2,3,4,7,8,9,10,12,15)
p <- 1/(1 + exp(0.45 + 0.70*x))
ci.ind <- rep(0, M)
beta1.emp <- rep(0, M)

for (i in 1:M) {
  z <- sapply(1:9, function(j) rnbinom(n=1, size=1, prob=p[j]))
  mat <- summary(glm(z~x, family=poisson(link="log")))$coefficients
  beta1.emp[i] <- mat[2,1]
  ci.L <- beta1.emp[i] - 1.96*mat[2,2]
  ci.U <- beta1.emp[i] + 1.96*mat[2,2]
  ci.ind[i] <- 1*(ci.L < 0.70 && 0.70 < ci.U)
}

c(mean(ci.ind), sqrt(var(beta1.emp)))
```

```
## [1] 0.495000 0.563285
```

Section 5

Cross-Validation

Basic Cross-Validation

- Motivation: when training/fitting a model on data, one is concerned about the performance of said model on new data (presumably selected from the same population)
- This goes back to the idea of replicating data, but this time without knowing the parameters in the model.
- Idea: train/fit the model on part(s) of the dataset $\mathcal{D}_1$ (the "training" set), and then test it on the remaining part(s) $\mathcal{D}_2$ (the "testing" set)
- Also called the *holdout method*
- More involved: repeat the above many times in some principled way

Motivation: Parameter Tuning/Model Selection

- Suppose (θ, λ) are parameters in the model, with θ of interest, and λ a tuning parameter one must choose from a set $\{\lambda_1, \ldots, \lambda_K\}$
- First we fit model onto the data $\mathcal{D}_1$ using λ_j to get $\hat{\theta}^{(j)}$ for $1 \leq j \leq K$
- Then we compute a goodness-of-fit measure (predictive RMSE, likelihood, etc.) when using the model with parameter values set to $(\hat{\theta}^{(j)}, \lambda_j)$ on $\mathcal{D}_2$
- Examples: LASSO, bandwidth selection in NP, many instances of model selection, etc.

Motivation: Risk Assessment/Evaluation of Loss

- Suppose that θ is the parameter of interest
- Using $\mathcal{D}_1$, we obtain $\hat{\theta}^{(1)}$ and using using $\mathcal{D}_2$, we compute $\hat{\theta}^{(2)}$
- If we do this multiple times we end up with pairs $(\hat{\theta}^{(1)}, \hat{\theta}^{(2)})_i$ so we can analyze the stochastic behaviour of the difference $\delta_i = \hat{\theta}_i^{(1)} - \hat{\theta}_i^{(2)}$
  - e.g. $\mathbb{E}[\delta_i]$, $\mathrm{Var}(\delta_i)$ or $\mathsf{MSE}(\delta_i) = \mathbb{E}[\delta_i^2]$
- The latter is

$$\mathbb{E}\left[(\hat{\theta}^{(1)} - \hat{\theta}^{(2)})^2\right] = \underbrace{\mathbb{E}\left[(\hat{\theta}^{(1)} - \theta)^2\right]}_{\mathsf{MSE}_1} + \underbrace{\mathbb{E}\left[(\hat{\theta}^{(2)} - \theta)^2\right]}_{\mathsf{MSE}_2} - 2\mathrm{Cov}\left(\hat{\theta}^{(1)}, \hat{\theta}^{(2)}\right)$$

- When $\mathcal{D}_1$ is independent of $\mathcal{D}_2$ and the $\hat{\theta}$'s are unbiased, this reduces to $\mathsf{MSE}(\delta_i) = \mathsf{MSE}_1 + \mathsf{MSE}_2$

k-Fold Cross-Validation

- Example: *k-fold cross-validation*:
  1. Partition the data into k equal-sized batches
  2. For each $1 \leq i \leq k$, use batch i for testing and the other $k-1$ batches for training
  3. Average the k ‘results’ (these are all δ_i’s)
- The k results are not independent
- The resulting estimator has high variance when k is small

Example: k-Fold Cross-Validation for Linear Regression

```r
set.seed(2311)

k <- 10

dat$group <- sample(rep(1:k, times=n/k), size=n)

RMSE.vec <- 0*(1:k)

for (i in 1:k) {
  dat.i <- subset(dat, group == i)
  dat.noti <- subset(dat, group != i)
  mod.i <- lm(y ~ ., data=dat.noti)
  y.hat.i <- predict.lm(mod.i, newdata=dat.i)
  RMSE.vec[i] <- sqrt(mean((dat.i$y-y.hat.i)^2))
}

RMSE.kfold <- mean(RMSE.vec)
```

Leave-p-Out Cross-Validation

- Example: *Leave-p-out cross-validation*:
  1. There are $\binom{n}{p}$ unique ways of partitioning the data into one batch of size $n - p$ and another of size p
  2. For each partition, use the $(n - p)$-sized batch for training and the p-sized batch for testing
  3. Average the $\binom{n}{p}$ results
- For moderately large datasets, this is only practical when p is small (often p is taken to be 1 or 2)

Monte Carlo Cross-Validation

- Idea: instead of working exhaustively, use repeated random sampling
- Example: *Monte Carlo cross-validation*:
  1. Choose the number of simulations b
  2. For $1 \leq i \leq b$, perform standard validation with a random partition of the data (of fixed length)
  3. Average the b results

Example: Monte Carlo Cross-Validation for Linear Regression

```r
set.seed(2311)

b <- 100

RMSE.MC.vec <- 0*(1:b)

for (i in 1:b) {
  inds.train <- sample(1:n, size=0.7*n)
  dat.i.train <- dat[inds.train,]
  dat.i.test <- dat[-inds.train,]
  mod.i <- lm(y ~ ., data=dat.i.train)
  y.hat.i <- predict.lm(mod.i, newdata=dat.i.test)
  RMSE.MC.vec[i] <- sqrt(mean((dat.i.test$y-y.hat.i)^2))
}

RMSE.MC <- mean(RMSE.MC.vec)
```

More on Cross-Validation

- There are countless variations of cross-validation in use [Arlot and Celisse, 2010]
- While the basic idea may seem simple, much theory has been developed to support it
- For example, it has been shown that when using maximum likelihood, using cross-validation for model selection is asymptotically equivalent to the AIC [Stone, 1977]
- Cross-validation can be also used for density estimation in an optimal way [Celisse, 2014]
- But the theory is tricky!
  - See the above references for examples

Bayesian Modelling

- We have not discussed yet Bayesian techniques...
- The basic model has $Y \sim p(y \mid \theta)$ *and* a prior distribution $\theta \sim p(\theta \mid \lambda)$, which leads to a joint probability model with density

$$p(y, \theta \mid \lambda) = p(y \mid \theta) p(\theta \mid \lambda)$$

- Inference is based on the posterior distribution, which has density

$$\pi(\theta \mid y, \lambda) = \frac{p(y, \theta \mid \lambda)}{\int p(y, \theta | \lambda) \, \mathrm{d}\theta}$$

- The denominator is the *marginal probability of the data* (sometimes denoted as $m(y)$) and is often impossible to compute analytically

Questions of Interest

- What is the interpretation of Bayesian posterior?
- What are replicates in a Bayesian context?
- Does it still make sense to study frequentist properties for Bayesian estimators (e.g., consistency of the posterior mean, or the posterior mode, coverage of credible regions)?

References I

Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. 2010.

Patrick Burns. The r inferno. https://www.burns-stat.com/pages/Tutor/R_inferno.pdf, 2012.

Alain Celisse. Optimal cross-validation in density estimation with the l^2-loss. 2014.

Mervyn Stone. An asymptotic equivalence of choice of model by cross-validation and akaike's criterion. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):44–47, 1977.

Hadley Wickham. *Advanced r*. CRC press, 2019.