--- title: "Homework 4" author: "[Insert your name here]{style='background-color: yellow;'}" toc: true title-block-banner: true title-block-style: default format: html # format: pdf ---

---

Please read the instructions carefully before submitting your assignment.

1. This assignment requires you to only upload a `PDF` file on Canvas
2. Don't collapse any code cells before submitting.
3. Remember to make sure all your code output is rendered properly before uploading your submission.

⚠️ Please add your name to the author information in the frontmatter before submitting your assignment ⚠️

We will be using the following libraries:

{R} packages <- c("dplyr", "readr", "tidyr", "purrr", "stringr", "corrplot", "car", "caret", "torch", "nnet", "broom") # renv::install(packages) sapply(packages, require, character.only=T)

<br><br><br><br>

Question 1

30 points

Automatic differentiation using `torch`

1.1 (5 points)

Consider $g(x, y)$ given by $$ g(x, y) = (x - 3)^2 + (y - 4)^2. $$

Using elementary calculus derive the expressions for

$$ \frac{d}{dx}g(x, y), \quad \text{and} \quad \frac{d}{dy}g(x, y). $$

Using your answer from above, what is the answer to $$ \frac{d}{dx}g(x, y) \Bigg|_{(x=3, y=4)} \quad \text{and} \quad \frac{d}{dy}g(x, y) \Bigg|_{(x=3, y=4)} ? $$

Define $g(x, y)$ as a function in R, compute the gradient of $g(x, y)$ with respect to $x=3$ and $y=4$. Does the answer match what you expected?

{R} g <- function(x, y) { (x - 3)^2 + (y - 4)^2 } x <- torch_tensor(3, requires_grad = TRUE) y <- torch_tensor(4, requires_grad = TRUE) result <- g(x, y) result$backward() x$grad y$grad

---

The results were as expected

(10 points)

$$\newcommand{\u}{\boldsymbol{u}}\newcommand{\v}{\boldsymbol{v}}$$

Consider $h(\u, \v)$ given by $$ h(\u, \v) = (\u \cdot \v)^3, $$ where $\u \cdot \v$ denotes the dot product of two vectors, i.e., $\u \cdot \v = \sum_{i=1}^n u_i v_i.$

Using elementary calculus derive the expressions for the gradients

$$ \begin{aligned} \nabla_\u h(\u, \v) &= \Bigg(\frac{d}{du_1}h(\u, \v), \frac{d}{du_2}h(\u, \v), \dots, \frac{d}{du_n}h(\u, \v)\Bigg) \end{aligned} $$

Using your answer from above, what is the answer to $\nabla_\u h(\u, \v)$ when $n=10$ and

$$ \begin{aligned} \u = (-1, +1, -1, +1, -1, +1, -1, +1, -1, +1)\\ \v = (-1, -1, -1, -1, -1, +1, +1, +1, +1, +1) \end{aligned} $$

Define $h(\u, \v)$ as a function in R, initialize the two vectors $\u$ and $\v$ as `torch_tensor`s. Compute the gradient of $h(\u, \v)$ with respect to $\u$. Does the answer match what you expected?

{R} u <- torch_tensor(c(-1, 1, -1, 1, -1, 1, -1, 1, -1, 1), dtype=torch_float32(), requires_grad = TRUE) v <- torch_tensor(c(-1, -1, -1, -1, -1, 1, 1, 1, 1, 1), dtype=torch_float32()) h <- function(u, v) { (torch_dot(u, v) ^ 3) } result <- h(u, v) result$backward() print(u$grad)

1.3 (5 points)

Consider the following function $$ f(z) = z^4 - 6z^2 - 3z + 4 $$

Derive the expression for $$ f'(z_0) = \frac{df}{dz}\Bigg|_{z=z_0} $$ and evaluate $f'(z_0)$ when $z_0 = -3.5$.

Define $f(z)$ as a function in R, and using the `torch` library compute $f'(-3.5)$.

{R} f <- function(z) { z^4 - 6*z^2 - 3*z + 4 } z <- torch_tensor(-3.5, dtype = torch_float64(), requires_grad = TRUE) result <- f(z) result$backward() print(z$grad)

1.4 (5 points)

For the same function f, initialize $z[1] = -3.5$, and perform $n=100$ iterations of **gradient descent**, i.e.,

$z[{k+1}] = z[k] - \eta f'(z[k]) \quad$ for $k = 1, 2, \dots, 100$

Plot the curve f and add taking $\eta = 0.02$, add the points $\{z_0, z_1, z_2, \dots z_{100}\}$ obtained using gradient descent to the plot. What do you observe?

{R} library(ggplot2) f <- function(z) { z^4 - 6*z^2 - 3*z + 4 } df_dz <- function(z) { 4*z^3 - 12*z - 3 } z <- c(-3.5) eta <- 0.02 n_iterations <- 100 for (k in 1:n_iterations) { z_next <- z[length(z)] - eta * df_dz(z[length(z)]) z <- c(z, z_next) } z_plot <- seq(-4.5, 4.5, length.out = 400) f_plot <- f(z_plot) plot_data <- data.frame(z = z_plot, f = f_plot) points_data <- data.frame(z = z, f = f(z)) ggplot() + geom_line(data = plot_data, aes(x = z, y = f), color = 'blue') + geom_point(data = points_data, aes(x = z, y = f), color = 'red') + labs(title = 'Function f(z) and Points Obtained by Gradient Descent', x = 'z', y = 'f(z)') + theme_minimal()

1.5 (5 points)

Redo the same analysis as **Question 1.4**, but this time using $\eta = 0.03$. What do you observe? What can you conclude from this analysis

With a higher learning rate the points obtained through gradient descent converge to a different location on the function. This suggests that the choice of learning rate significantly impacts the convergence behavior of gradient descent.

<br><br><br><br> <br><br><br><br> —

Question 2

50 points

Logistic regression and interpretation of effect sizes

For this question we will use the **Titanic** dataset from the Stanford data archive. This dataset contains information about passengers aboard the Titanic and whether or not they survived.

2.1 (5 points)

Read the data from the following URL as a tibble in R. Preprocess the data such that the variables are of the right data type, e.g., binary variables are encoded as factors, and convert all column names to lower case for consistency. Let's also rename the response variable `Survival` to `y` for convenience.

{R} url <- "https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv" df <- read.csv(url) #Pre-Processing df <- df %>% mutate_at("Survived", factor) %>% rename(y = Survived) %>% mutate_at("Pclass", factor) names(df) <- tolower(names(df)) df

---

2.2 (5 points)

Visualize the correlation matrix of all numeric columns in `df` using `corrplot()`

{R} df_numeric <- df %>% select(age, siblings.spouses.aboard, parents.children.aboard, fare) corrplot(cor(df_numeric))

2.3 (10 points)

Fit a logistic regression model to predict the probability of surviving the titanic as a function of:

- `pclass`
- `sex`
- `age`
- `fare`
- `# siblings`
- `# parents`

{R} df_logistic <- df %>% select(-name) full_model <- glm(y ~ ., data = df_logistic, family = "binomial") summary(full_model)

2.4 (30 points)

Provide an interpretation for the slope and intercept terms estimated in `full_model` in terms of the log-odds of survival in the titanic and in terms of the odds-ratio (if the covariate is also categorical).

Recall the definition of logistic regression from the lecture notes, and also recall how we interpreted the slope in the linear regression model (particularly when the covariate was categorical).

In logistic regression, the intercept represents the log-odds of survival on the Titanic when all predictors are zero, while the slope coefficients indicate how the log-odds change with a one-unit increase in continuous predictors or when comparing categories in categorical predictors, with these effects expressed as odds ratios for easier interpretation.

<br><br><br><br> <br><br><br><br> —

Question 3

70 points

Variable selection and logistic regression in `torch`

3.1 (15 points)

Complete the following function `overview` which takes in two categorical vectors (`predicted` and `expected`) and outputs:

- The prediction accuracy
- The prediction error
- The false positive rate, and
- The false negative rate

```
{R} overview <- function(predicted, expected) { total_true_positives <- sum(predicted == 1 &
expected == 1) total_true_negatives <- sum(predicted == 0 & expected == 0)
total_false_positives <- sum(predicted == 1 & expected == 0) total_false_negatives <-
sum(predicted == 0 & expected == 1) accuracy <- (total_true_positives + total_true_negatives) /
length(expected) error <- (total_false_positives + total_false_negatives) / length(expected)
false_positive_rate <- total_false_positives / (total_false_positives + total_true_negatives)
false_negative_rate <- total_false_negatives / (total_true_positives + total_false_negatives)
return( data.frame( accuracy = accuracy, error=error, false_positive_rate = false_positive_rate,
false_negative_rate = false_negative_rate ) ) }
```

You can check if your function is doing what it's supposed to do by evaluating

```
{R} overview(df$y, df$y)
```

and making sure that the accuracy is $100\%$ while the errors are $0\%$.

3.2 (5 points)

Display an overview of the key performance metrics of `full_model`

```
{R} predicted_probabilities <- predict(full_model, df, type = "response") predicted_outcomes <-
ifelse(predicted_probabilities > 0.5, 1, 0) performance_metrics <- overview(predicted =
predicted_outcomes, expected = df$y) performance_metrics
```

3.3 (5 points)

Using backward-stepwise logistic regression, find a parsimonious altenative to `full_model`, and print its `overview`

```
{R} step_model <- step(full_model, direction = "backward") summary(step_model)
{R} step_pred_prob <- predict(step_model, df, type = "response") step_predictions <-
ifelse(step_pred_prob > 0.5, 1, 0) overview(step_predictions, df$y)
```

3.4 (15 points)

Using the `caret` package, setup a 5-fold cross-validation training method using the `caret::trainConrol()` function

{R} controls <- trainControl(method = "cv", number = 5, classProbs = TRUE)

Now, using `control`, perform 5-fold cross validation using `caret::train()` to select the optimal λ parameter for LASSO with logistic regression.

Take the search grid for λ to be in $\{ 2^{-20}, 2^{-19.5}, 2^{-19}, \dots, 2^{-0.5}, 2^{0} \}$.

{R} lasso_fit <- train(x = df[, -which(names(df) == "y")], # Assuming 'df' is your dataframe and 'y' is the response variable y = df$y, method = "glmnet", trControl = controls, tuneGrid = expand.grid(alpha = 1, # LASSO regression lambda = 2^seq(-20, 0, by = 0.5)), family = "binomial" # Assuming a binary classification problem; use "gaussian" for regression)

Using the information stored in `lasso_fit$results`, plot the results for cross-validation accuracy vs. $log_2(\lambda)$. Choose the optimal λ^*, and report your results for this value of λ^*.

3.5 (25 points)

First, use the `model.matrix()` function to convert the covariates of `df` to a matrix format

{R} covariate_matrix <- model.matrix(full_model)[, -1]

Now, initialize the covariates X and the response y as `torch` tensors

{R} X <- ... # Insert your code here y <- ... # Insert your code here

Using the `torch` library, initialize an `nn_module` which performs logistic regression for this dataset. (Remember that we have 6 different covariates)

{R} logistic <- nn_module(initialize = function() { self$f <- nn_linear(in_features = 6, out_features = 1) self$g <- nn_sigmoid() }, forward = function(x) { x %>% self$f() %>% self$g() }) f <- logistic()

You can verify that your code is right by checking that the output to the following code is a vector of probabilities:

{R} f(X)

Now, define the loss function `Loss()` which takes in two tensors `X` and `y` and a function `Fun`, and outputs the **Binary cross Entropy loss** between `Fun(X)` and `y`.

{R} Loss <- function(X, y, Fun){ predicted <- Fun(X) loss <- nnf_binary_cross_entropy(predicted, y$view(c(-1, 1))) return(loss) }

Initialize an optimizer using `optim_adam()` and perform $n=1000$ steps of gradient descent in order to fit logistic regression using `torch`.

{R} f <- logistic() optimizer <- optim_adam(f$parameters) n <- 1000 for (i in 1:n) { optimizer$zero_grad() loss <- Loss(X, y, f) loss$backward() optimizer$step() if (i %% 100 == 0) { cat("Iteration:", i, "Loss:", loss$item(), "\n") } }

Using the final, optimized parameters of `f`, compute the compute the predicted results on `X`

{R} predicted_probabilities <- f(X) %>% as_array() torch_predictions <- ifelse(predicted_probabilities > 0.5, 1, 0) overview(torch_predictions, df$y)

3.6 (5 points)

Create a summary table of the `overview()` summary statistics for each of the $4$ models we have looked at in this assignment, and comment on their relative strengths and drawbacks.

\pagebreak

<br><br><br><br> <br><br><br><br> —

Session Information

Print your `R` session information using the following command

{R} sessionInfo()