Predictions and Inference for Logistic Regression

Last modified: 2026-11-19

```
# The Python chunks import these packages; reticulate provisions them.
library(reticulate) # nolint: undesirable_function_linter
py_packages <- c(
  "numpy", "pandas", "statsmodels", "scipy", "scikit-learn", "matplotlib",
  "plotly"
)
py_require(py_packages)
```

Source

This material was a section of the logistic regression page. It carries the same sources: Brian Hutchinson's Fall 2025 lecture notes, Kameron Decker Harris' 2022 weekly notes and Logan Sizemore's Spring 2025 lectures, as described there.

1 Quantifying the uncertainty of the coefficients

A fitted coefficient is an estimate[1]: another sample[2] from the same population[3] would give a different value, and its sampling distribution[4] describes how much it varies. The matrix[5] $\mathbf{D}^\top W \mathbf{D}$ that Newton's method[6] uses to compute its step also gives the size of that variation. Where the log-likelihood[7] curves sharply at $\hat{\vec{\beta}}$, $\mathbf{D}^\top W \mathbf{D}$ is large and the standard errors[8] are small; where it is nearly flat, the standard errors are large.

Remark (Inference for logistic regression)

Remark 1.1 (Inference for logistic regression). The statistics notes' page Inference for Logistic Regression[a] covers the deviance[b], the null deviance[c], the likelihood ratio test[d] of the slopes[e], and the Wald interval for a fitted logistic coefficient[f]. It shows that the drop from the null deviance to the deviance is the likelihood ratio statistic[g]. It also shows that, for a large sample[h], the fitted coefficients $\hat{\vec{\beta}}$ are approximately Gaussian with mean[i] $\vec{\beta}$ and covariance[j] $(\mathbf{D}^\top W \mathbf{D})^{-1}$ (its approximate distribution theorem[k]), where W is the weight matrix[l] in the logistic Hessian theorem. In that result the weights $\pi_i(1-\pi_i)$ are computed from the true probabilities; the standard errors[m], like the Newton step, use the weights computed from the fitted probabilities instead.

[a] https://morrison-lab.github.io/sds/logistic-inference.html
[b] https://morrison-lab.github.io/sds/logistic-inference.html#def-deviance
[c] https://morrison-lab.github.io/sds/logistic-inference.html#def-null-deviance
[d] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat

[1] https://morrison-lab.github.io/sds/estimation.html#def-estimate
[2] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample
[3] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-population
[4] https://morrison-lab.github.io/sds/estimation.html#def-sampling-distribution
[5] https://morrison-lab.github.io/mds/linear-algebra-matrices.html#def-matrix
[6] https://morrison-lab.github.io/mds/optimization-newton.html#def-newton-method
[7] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[8] https://morrison-lab.github.io/sds/estimation.html#def-SE

[e] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[f] https://morrison-lab.github.io/sds/logistic-inference.html#cor-logistic-wald
[g] https://morrison-lab.github.io/sds/logistic-inference.html#thm-deviance-drop
[h] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample
[i] https://morrison-lab.github.io/pds/expectation.html#def-expectation
[j] https://morrison-lab.github.io/pds/variance-covariance.html#def-cov
[k] https://morrison-lab.github.io/sds/logistic-inference.html#thm-logistic-mle-distribution
[l] https://morrison-lab.github.io/mds/linear-algebra-matrices.html#def-matrix
[m] https://morrison-lab.github.io/sds/estimation.html#def-SE

2 Fitting on real data

Statistical software fits logistic regression by the Newton iteration of part 2 of the Newton and IRLS exercise. This section uses it on stock market data, to predict whether the market rises on a given day.

Daily percentage returns of the S&P 500 stock index on 1250 trading days from 2001 to 2005 make up the `Smarket` data [1, Sec. 4.7]. The data are in `data/smarket.csv`[9], one row per trading day, copied from the `ISLP` Python package. The columns this course uses are:

- `Direction`: the outcome[10], whether the index rose (`Up`) or fell (`Down`) that day.
- `Lag1` to `Lag5`: the percentage returns on each of the five previous trading days.
- `Volume`: the number of shares traded on the previous day, in billions.
- `Year`: the year of the trading day.

The `Today` column, the day's own return, is not a predictor, because `Direction` is `Down` exactly when `Today` is negative.

The `Date` column, the calendar date of each trading day, is not used by the exercises; it names the day that a row describes, from 2001-01-10 to 2005-12-30. The `ISLP` copy has no dates. The dates were recovered by matching `Today` against the S&P 500 daily returns in the `sp500ret` data of the `rugarch` package [2], after converting those log returns to percentage returns. The script `tools/add_smarket_dates.R`[11] does the match. It stops unless exactly one run of trading days agrees with all 1250 values of `Today`, to within 0.0005 percentage points, the rounding of `Today`.

Dataset 1: The stock market (`Smarket`) data

2.0.1 R

```
smarket_preview <- readr::read_csv("data/smarket.csv", show_col_types = FALSE)
smarket_preview
```

```
# A tibble: 1,250 x 10
    Year   Lag1   Lag2   Lag3   Lag4   Lag5 Volume  Today Direction Date
   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <chr>     <date>
 1  2001  0.381 -0.192 -2.62  -1.06   5.01    1.19  0.959 Up        2001-01-10
 2  2001  0.959  0.381 -0.192 -2.62  -1.06    1.30  1.03  Up        2001-01-11
 3  2001  1.03   0.959  0.381 -0.192 -2.62    1.41 -0.623 Down      2001-01-12
 4  2001 -0.623  1.03   0.959  0.381 -0.192   1.28  0.614 Up        2001-01-16
 5  2001  0.614 -0.623  1.03   0.959  0.381   1.21  0.213 Up        2001-01-17
 6  2001  0.213  0.614 -0.623  1.03   0.959   1.35  1.39  Up        2001-01-18
 7  2001  1.39   0.213  0.614 -0.623  1.03    1.44 -0.403 Down      2001-01-19
 8  2001 -0.403  1.39   0.213  0.614 -0.623   1.41  0.027 Up        2001-01-22
 9  2001  0.027 -0.403  1.39   0.213  0.614   1.16  1.30  Up        2001-01-23
10  2001  1.30   0.027 -0.403  1.39   0.213   1.23  0.287 Up        2001-01-24
# i 1,240 more rows
```

```
dim(smarket_preview)
```

```
[1] 1250   10
```

```
skimr::skim_without_charts(smarket_preview)
```

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Direction	0	1	2	4	0	2	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
Date	0	1	2001-01-10	2005-12-30	2003-07-10	1250

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
Year	0	1	2003.02	1.41	2001.00	2002.00	2003.00	2004.00	2005.00
Lag1	0	1	0.00	1.14	-4.92	-0.64	0.04	0.60	5.73
Lag2	0	1	0.00	1.14	-4.92	-0.64	0.04	0.60	5.73
Lag3	0	1	0.00	1.14	-4.92	-0.64	0.04	0.60	5.73
Lag4	0	1	0.00	1.14	-4.92	-0.64	0.04	0.60	5.73
Lag5	0	1	0.01	1.15	-4.92	-0.64	0.04	0.60	5.73
Volume	0	1	1.48	0.36	0.36	1.26	1.42	1.64	3.15
Today	0	1	0.00	1.14	-4.92	-0.64	0.04	0.60	5.73

2.0.2 Python

```
import pandas as pd

pd.set_option("display.max_columns", None)
pd.set_option("display.width", 120)

smarket_preview = pd.read_csv("data/smarket.csv")
smarket_preview
```

```
      Year   Lag1   Lag2   Lag3   Lag4   Lag5   Volume  Today Direction        Date
0     2001  0.381 -0.192 -2.624 -1.055  5.010  1.19130  0.959        Up  2001-01-10
1     2001  0.959  0.381 -0.192 -2.624 -1.055  1.29650  1.032        Up  2001-01-11
2     2001  1.032  0.959  0.381 -0.192 -2.624  1.41120 -0.623      Down  2001-01-12
3     2001 -0.623  1.032  0.959  0.381 -0.192  1.27600  0.614        Up  2001-01-16
4     2001  0.614 -0.623  1.032  0.959  0.381  1.20570  0.213        Up  2001-01-17
...    ...    ...    ...    ...    ...    ...      ...    ...       ...         ...
1245  2005  0.422  0.252 -0.024 -0.584 -0.285  1.88850  0.043        Up  2005-12-23
1246  2005  0.043  0.422  0.252 -0.024 -0.584  1.28581 -0.955      Down  2005-12-27
1247  2005 -0.955  0.043  0.422  0.252 -0.024  1.54047  0.130        Up  2005-12-28
1248  2005  0.130 -0.955  0.043  0.422  0.252  1.42236 -0.298      Down  2005-12-29
1249  2005 -0.298  0.130 -0.955  0.043  0.422  1.38254 -0.489      Down  2005-12-30

[1250 rows x 10 columns]
```

```
smarket_preview.info()
```

```
<class 'pandas.DataFrame'>
RangeIndex: 1250 entries, 0 to 1249
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Year       1250 non-null   int64
 1   Lag1       1250 non-null   float64
 2   Lag2       1250 non-null   float64
 3   Lag3       1250 non-null   float64
 4   Lag4       1250 non-null   float64
 5   Lag5       1250 non-null   float64
 6   Volume     1250 non-null   float64
 7   Today      1250 non-null   float64
 8   Direction  1250 non-null   str
 9   Date       1250 non-null   str
dtypes: float64(7), int64(1), str(2)
memory usage: 97.8 KB
```

```
print(smarket_preview.describe())
```

```
              Year         Lag1         Lag2         Lag3         Lag4         Lag5       Volume         T
count  1250.000000  1250.000000  1250.000000  1250.000000  1250.000000  1250.00000  1250.000000  1250.00
mean   2003.016000     0.003834     0.003919     0.001716     0.001636     0.00561     1.478305     0.00
std       1.409018     1.136299     1.136280     1.138703     1.138774     1.14755     0.360357     1.13
min    2001.000000    -4.922000    -4.922000    -4.922000    -4.922000    -4.92200     0.356070    -
4.922000
25%    2002.000000    -0.639500    -0.639500    -0.640000    -0.640000    -0.64000     1.257400    -
0.639500
50%    2003.000000     0.039000     0.039000     0.038500     0.038500     0.03850     1.422950     0.03
75%    2004.000000     0.596750     0.596750     0.596750     0.596750     0.59700     1.641675     0.59
max    2005.000000     5.733000     5.733000     5.733000     5.733000     5.73300     3.152470     5.73
```

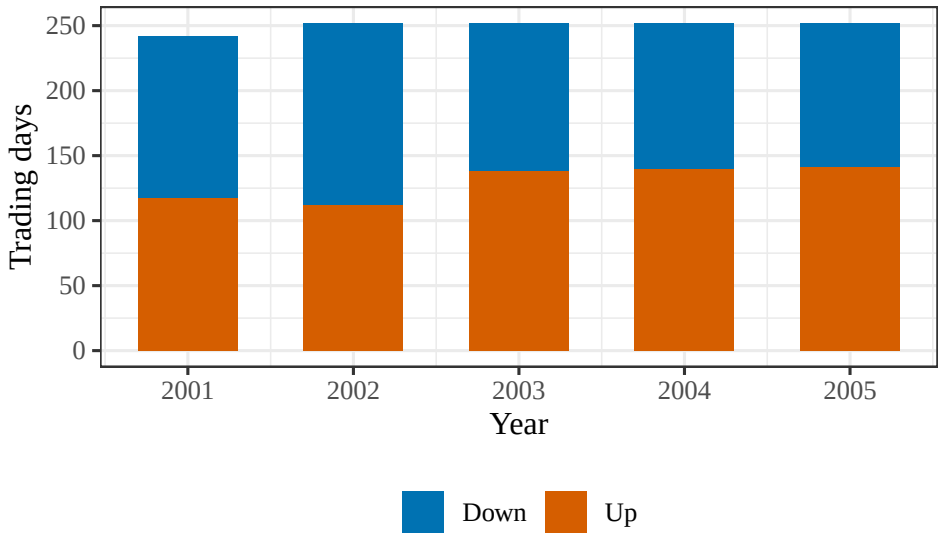
```
print(smarket_preview.isna().sum())
```

```
Year         0
Lag1         0
Lag2         0
Lag3         0
Lag4         0
Lag5         0
Volume       0
Today        0
Direction    0
Date         0
dtype: int64
```

Exercise 2.1 (Predicting the direction of the stock market)**.** Use the `Smarket` data. Fit a logistic regression for the probability[a] that `Direction` is `Up`, with the five lagged returns and `Volume` as predictors, on the days before 2005 only.
Figure 1 plots the number of trading days in each year by `Direction`, including the days of 2005 that the fit leaves out.

2.0.3 R

```
sf_data <- readr::read_csv("data/smarket.csv", show_col_types = FALSE)
sf_fills <- c(Down = "#0072B2", Up = "#D55E00")
sf_data |>
  ggplot2::ggplot(ggplot2::aes(Year, fill = Direction)) +
  ggplot2::geom_bar(width = 0.6) +
  ggplot2::scale_fill_manual(values = sf_fills) +
  ggplot2::labs(x = "Year", y = "Trading days", fill = NULL)
```

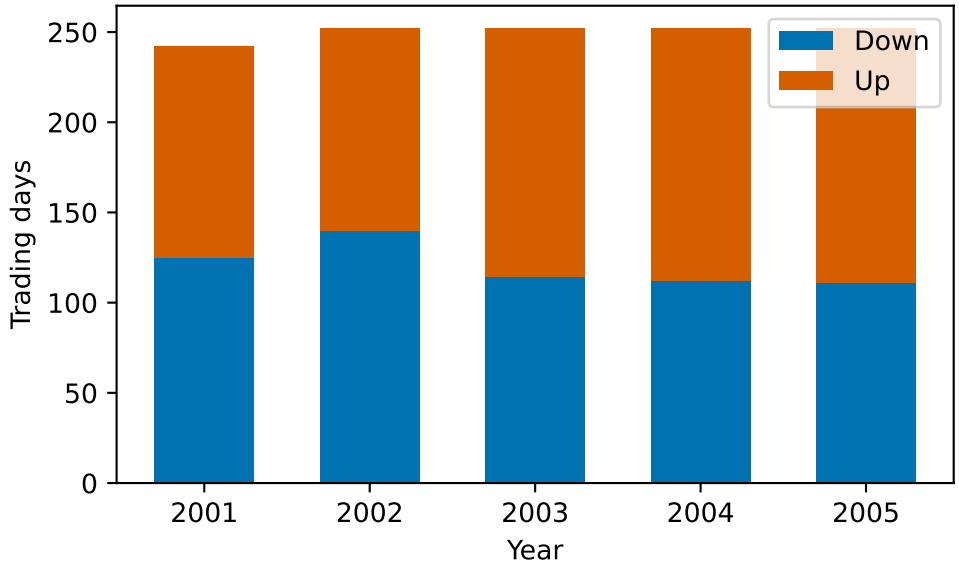


(a)

2.0.4 Python

```
import matplotlib.pyplot as plt
import pandas as pd

smarket_sf = pd.read_csv("data/smarket.csv")
counts_sf = smarket_sf.groupby(["Year", "Direction"]).size().unstack()
fig, ax = plt.subplots(figsize=(5, 3), layout="constrained")
ax.bar(counts_sf.index, counts_sf["Down"], color="#0072B2", width=0.6,
       label="Down");
ax.bar(counts_sf.index, counts_sf["Up"], bottom=counts_sf["Down"],
       color="#D55E00", width=0.6, label="Up");
ax.set_xlabel("Year");
ax.set_ylabel("Trading days");
ax.legend();
plt.show()
```



(b)

Figure 1: The number of trading days in each year of the `Smarket` data, split by `Direction`.

[a] https://morrison-lab.github.io/pds/probability-basics.html#def-probability

Solution

Solution 2.1.

2.0.5 R

```
smarket <- read.csv("data/smarket.csv")
str(smarket)
```

```
'data.frame':   1250 obs. of  10 variables:
 $ Year     : int  2001 2001 2001 2001 2001 2001 2001 2001 2001 2001 ...
 $ Lag1     : num  0.381 0.959 1.032 -0.623 0.614 ...
 $ Lag2     : num  -0.192 0.381 0.959 1.032 -0.623 ...
 $ Lag3     : num  -2.624 -0.192 0.381 0.959 1.032 ...
 $ Lag4     : num  -1.055 -2.624 -0.192 0.381 0.959 ...
```

```
 $ Lag5     : num  5.01 -1.055 -2.624 -0.192 0.381 ...
 $ Volume   : num  1.19 1.3 1.41 1.28 1.21 ...
 $ Today    : num  0.959 1.032 -0.623 0.614 0.213 ...
 $ Direction: chr  "Up" "Up" "Down" "Up" ...
 $ Date     : chr  "2001-01-10" "2001-01-11" "2001-01-12" "2001-01-16" ...
```

```r
smarket <- dplyr::mutate(smarket, up = as.integer(Direction == "Up"))
train <- smarket$Year < 2005

fit <- glm(
  up ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
  data = dplyr::filter(smarket, train),
  family = binomial
)
coef(fit)
```

```
 (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5
 0.191212621 -0.054178292 -0.045805334  0.007200118  0.006440875 -0.004222672
      Volume
-0.116256960
```

2.0.6 Python

```python
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf

smarket_py = pd.read_csv("data/smarket.csv")
print(smarket_py.head(3))
```

```
   Year   Lag1   Lag2   Lag3   Lag4   Lag5  Volume  Today Direction        Date
0  2001  0.381 -0.192 -2.624 -1.055  5.010  1.1913  0.959        Up  2001-01-10
1  2001  0.959  0.381 -0.192 -2.624 -1.055  1.2965  1.032        Up  2001-01-11
2  2001  1.032  0.959  0.381 -0.192 -2.624  1.4112 -0.623      Down  2001-01-12
```

```python
smarket_py["up"] = (smarket_py["Direction"] == "Up").astype(int)
train_py = smarket_py["Year"] < 2005

fit_py = smf.glm(
    "up ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume",
    data=smarket_py[train_py],
    family=sm.families.Binomial(),
).fit()
print(fit_py.params.round(3))
```

```
Intercept    0.191
Lag1        -0.054
Lag2        -0.046
Lag3         0.007
Lag4         0.006
Lag5        -0.004
Volume      -0.116
dtype: float64
```

Exercise 2.2 (Predicting the direction of the stock market (part 1b))**.** Use the setup of Exercise 2.1. Predict `Direction` for each day in 2005 from the fit of Exercise 2.1, predicting `Up` when the predicted

probability is above 0.5.

Solution

Solution 2.2. This continues Solution 2.0.6.

2.0.7 R

```
test_days <- dplyr::filter(smarket, !train)
p <- predict(fit, test_days, type = "response")
pred <- ifelse(p > 0.5, "Up", "Down")
actual <- smarket$Direction[!train]

table(predicted = pred, actual = actual)
```

```
         actual
predicted Down Up
     Down   77 97
     Up     34 44
```

2.0.8 Python

```
p_py = fit_py.predict(smarket_py[~train_py])
pred_py = np.where(p_py > 0.5, "Up", "Down")
actual_py = smarket_py.loc[~train_py, "Direction"]

print(pd.crosstab(pred_py, actual_py, rownames=["predicted"], colnames=["actual"]))
```

```
actual     Down  Up
predicted
Down         77  97
Up           34  44
```

Exercise 2.3 (Predicting the direction of the stock market (part 1c))**.** Use the setup of Exercise 2.1. On what proportion of the 2005 days is the prediction of Exercise 2.2 correct?

Solution

Solution 2.3. This continues Solution 2.0.6 and Solution 2.0.8.

2.0.9 R

```
accuracy <- mean(pred == actual)
accuracy
```

```
[1] 0.4801587
```

```
# A 95% interval for an accuracy: the proportion plus or minus
# 1.96 standard errors, which is accurate when the number of test days
# is in the hundreds.
n_test <- length(actual)
bounds <- c(lower = -1, upper = 1)
z_crit <- qnorm(0.975)
se_accuracy <- sqrt(accuracy * (1 - accuracy) / n_test)
ci_accuracy <- accuracy + bounds * z_crit * se_accuracy
```

2.0.10 Python

```
print("accuracy:", round(float(np.mean(pred_py == actual_py)), 3))
```

```
accuracy: 0.48
```

```
def wald_ci_py(p, n):
    # A 95% interval for a proportion: p plus or minus 1.96 standard errors.
    half = 1.96 * np.sqrt(p * (1 - p) / n)
    return round(float(p - half), 3), round(float(p + half), 3)

print(wald_ci_py(float(np.mean(pred_py == actual_py)), len(actual_py)))
```

```
(0.418, 0.542)
```

The fitted logistic regression predicts the correct label on 48% of the 252 days in 2005 (95% interval [41.8, 54.2]%).

Exercise 2.4 (Predicting the direction of the stock market (part 2))**.** Use the setup of Exercise 2.1. Compare the proportion correct of Exercise 2.3 with that of the rule that predicts `Up` on every day.

Solution

Solution 2.4.

2.0.11 R

```
always_up <- mean(actual == "Up")
se_always_up <- sqrt(always_up * (1 - always_up) / n_test)
ci_always_up <- always_up + bounds * z_crit * se_always_up
# Both rules are scored on the same days, so the paired difference in
# correctness, day by day, has a t interval for its mean.
hit_model <- as.integer(pred == actual)
hit_up <- as.integer(actual == "Up")
diff_hit <- hit_model - hit_up
t_crit <- qt(0.975, n_test - 1)
se_diff <- sd(diff_hit) / sqrt(n_test)
ci_vs_up <- mean(diff_hit) + bounds * t_crit * se_diff
```

2.0.12 Python

```
print("always Up:", round(float(np.mean(actual_py == "Up")), 3))
```

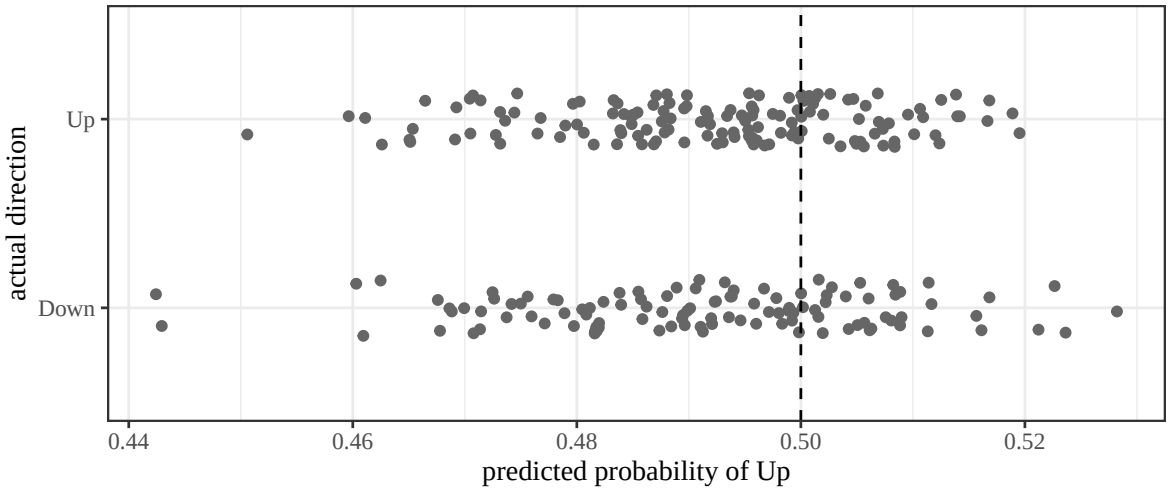
```
always Up: 0.56
```

Predicting `Up` on every day is right on 56% of the 2005 days (95% interval [49.8, 62.1]%). Each interval is the proportion plus or minus 1.96 standard errors[a], $\sqrt{\hat{\pi}(1-\hat{\pi})/n}$. The two rules are scored on the same days, so their difference is taken day by day: the accuracy of the logistic regression minus that of the constant[b] rule is -7.9 percentage points (95% interval [-18.2, 2.3]). The intervals describe sampling variation across the test days in 2005, not across training sets.

The fitted model is right less often than the rule that ignores every predictor, but the interval for the difference includes 0, so these 252 days do not show that the fitted model is worse. Its predicted probabilities are all between 0.44 and 0.53: no day in 2005 gets a probability of `Up` far from 0.5. Figure 2 shows that the days that went up and the days that went down get nearly the same predicted probabilities.

2.0.13 R

```r
library(ggplot2)
ggplot(tibble::tibble(p = p, actual = actual)) +
  aes(p, actual) +
  geom_jitter(height = 0.15, width = 0, colour = "grey40") +
  geom_vline(xintercept = 0.5, linetype = "dashed") +
  labs(x = "predicted probability of Up", y = "actual direction")
```



2.0.14 Python

```python
import matplotlib.pyplot as plt

rng_strip = np.random.default_rng(571)
row = np.where(actual_py == "Up", 1, 0) + rng_strip.uniform(-0.15, 0.15, len(actual_py))
fig, ax = plt.subplots(figsize=(7, 3), layout="constrained")
ax.scatter(p_py, row, color="0.25", s=12);
ax.axvline(0.5, color="black", linestyle="--");
ax.set_yticks([0, 1], ["Down", "Up"]);
ax.set_xlabel("predicted probability of Up");
ax.set_ylabel("actual direction");
plt.show()
```

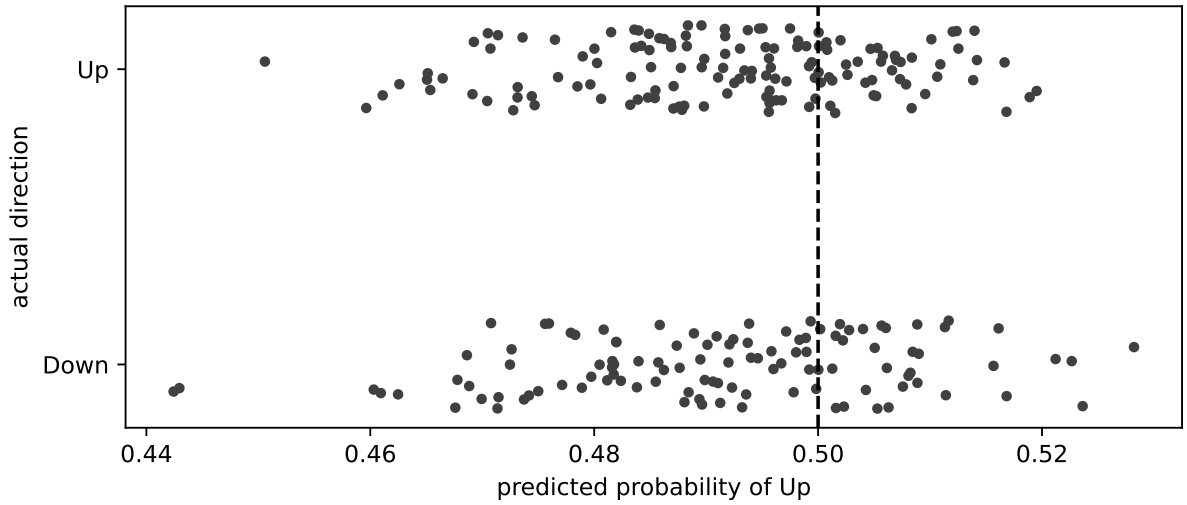


Figure 2: Predicted probability that the market goes up, for each trading day in 2005, by what the market actually did. The dashed line is the threshold of 0.5.

[a] https://morrison-lab.github.io/sds/estimation.html#def-SE
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-constant-function

Exercise 2.5 (Fitting the stock market model by gradient descent)**.** Exercise 2.1 fitted the 2001 to 2004 `Smarket` days with R's `glm()` and statsmodels' `GLM`. Fit the same logistic regression with the gradient descent[a] of the exercise on fitting by gradient descent, standardizing the six predictors with the training days' means[b] and standard deviations[c], starting from $\vec{\beta} = \vec{0}$, with step size $\alpha = 0.5$ for 2,000 steps.

[a] https://morrison-lab.github.io/mds/optimization-gradient-descent.html#def-gradient-descent
[b] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample-mean
[c] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample-sd

Solution

Solution 2.5.

2.0.15 R

```
cols <- c("Lag1", "Lag2", "Lag3", "Lag4", "Lag5", "Volume")
smarket_train <- smarket |>
  dplyr::filter(train) |>
  dplyr::select(dplyr::all_of(cols))
smarket_test <- smarket |>
  dplyr::filter(!train) |>
  dplyr::select(dplyr::all_of(cols))
mu <- colMeans(smarket_train)
sdev <- apply(smarket_train, 2, sd)
# nolint start: object_name_linter.
Xs <- cbind(1, scale(smarket_train, mu, sdev))
Xte <- cbind(1, scale(smarket_test, mu, sdev))
# nolint end
y_tr <- smarket$up[train]
sig <- function(z) 0.5 * (1 + tanh(0.5 * z))

b <- rep(0, ncol(Xs))
for (i in 1:2000) {
  b <- b - 0.5 * drop(t(Xs) %*% (sig(drop(Xs %*% b)) - y_tr)) / length(y_tr)
}
b
```

```
                   Lag1         Lag2         Lag3         Lag4         Lag5
 0.032104744 -0.066621178 -0.056325934  0.008874033  0.007939830 -0.005248037
      Volume
-0.031112330
```

2.0.16 Python

```
cols = ["Lag1", "Lag2", "Lag3", "Lag4", "Lag5", "Volume"]
tr = smarket_py[train_py]
mu, sd = tr[cols].mean(), tr[cols].std()
Xs_py = np.column_stack([np.ones(len(tr)), ((tr[cols] - mu) / sd).to_numpy()])
Xte_py = np.column_stack(
    [np.ones((~train_py).sum()), ((smarket_py.loc[~train_py, cols] - mu) / sd).to_numpy()]
)
y_tr_py = tr["up"].to_numpy()
sig = lambda z: 0.5 * (1 + np.tanh(0.5 * z))

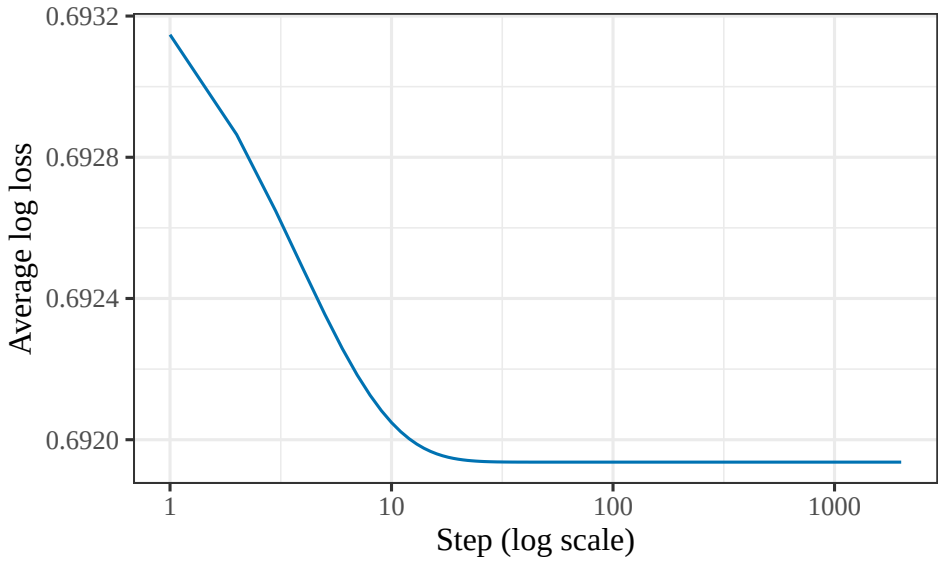
b_py = np.zeros(Xs_py.shape[1])
for _ in range(2000):
    b_py = b_py - 0.5 * Xs_py.T @ (sig(Xs_py @ b_py) - y_tr_py) / len(y_tr_py)
print("standardized:", np.round(b_py, 3))
```

```
standardized: [ 0.032 -0.067 -0.056  0.009  0.008 -0.005 -0.031]
```

Figure 3 plots the average[a] log loss at each step of the loop above.

2.0.17 R

```
gdl_steps <- 2000
gdl_loss <- numeric(gdl_steps)
gdl_ncol <- ncol(Xs)
gdl_b <- rep(0, gdl_ncol)
for (i in seq_len(gdl_steps)) {
  gdl_z <- drop(Xs %*% gdl_b)
  gdl_p <- sig(gdl_z)
  gdl_pt <- ifelse(y_tr == 1, gdl_p, 1 - gdl_p)
  gdl_loss[i] <- mean(-log(gdl_pt))
  gdl_grad <- drop(t(Xs) %*% (gdl_p - y_tr)) / length(y_tr)
  gdl_b <- gdl_b - 0.5 * gdl_grad
}
gdl_seq <- seq_len(gdl_steps)
gdl_path <- tibble::tibble(step = gdl_seq, loss = gdl_loss)
gdl_path |>
  ggplot2::ggplot(ggplot2::aes(step, loss)) +
  ggplot2::geom_line(colour = "#0072B2") +
  ggplot2::scale_x_log10() +
  ggplot2::labs(x = "Step (log scale)", y = "Average log loss")
```



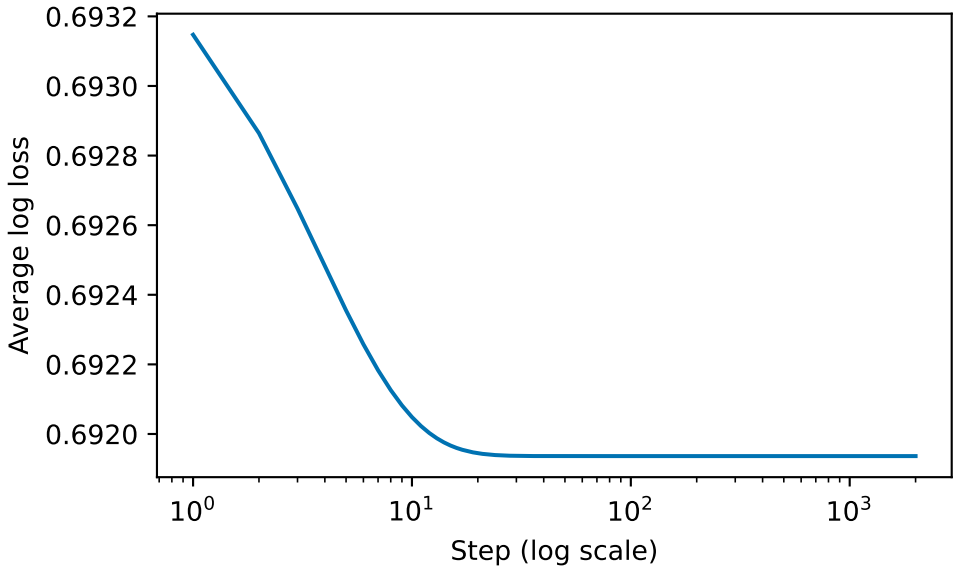
(a)

2.0.18 Python

```
import matplotlib.pyplot as plt
import numpy as np

b_gdl = np.zeros(Xs_py.shape[1])
loss_gdl = []
for _ in range(2000):
    p_gdl = sig(Xs_py @ b_gdl)
    p_true = np.where(y_tr_py == 1, p_gdl, 1 - p_gdl)
    loss_gdl.append(np.mean(-np.log(p_true)))
    grad_gdl = Xs_py.T @ (p_gdl - y_tr_py) / len(y_tr_py)
    b_gdl = b_gdl - 0.5 * grad_gdl

fig, ax = plt.subplots(figsize=(5, 3), layout="constrained")
ax.plot(np.arange(1, 2001), loss_gdl, color="#0072B2");
ax.set_xscale("log");
ax.set_xlabel("Step (log scale)");
ax.set_ylabel("Average log loss");
plt.show()
```



(b)

Figure 3: The average[b] log loss of the training days at each step of gradient descent[c].

[a] https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-mean
[b] https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-mean
[c] https://morrison-lab.github.io/mds/optimization-gradient-descent.html#def-gradient-descent

Exercise 2.6 (Fitting the stock market model by gradient descent (part 1b))**.** Use the setup of Exercise 2.5.
Convert the coefficients of the fit of Exercise 2.5 back to the original units of the predictors.

Solution

Solution 2.6. This continues Solution 2.0.16.

2.0.19 R

```
slopes <- b[-1] / sdev
coef_gd <- c(b[1] - sum(slopes * mu), slopes)
coef_lib <- coef(fit)
rbind( # nolint: undesirable_function_linter.
  gradient_descent = coef_gd, library_fit = coef_lib
)
```

```
                                 Lag1        Lag2        Lag3        Lag4
gradient_descent 0.1912126 -0.05417829 -0.04580533 0.007200118 0.006440875
library_fit      0.1912126 -0.05417829 -0.04580533 0.007200118 0.006440875
                         Lag5    Volume
gradient_descent -0.004222672 -0.116257
library_fit      -0.004222672 -0.116257
```

```
log_odds <- drop(Xte %*% b)
same_class <- (sig(log_odds) > 0.5) == (p > 0.5)
same <- mean(same_class)
same
```

```
[1] 1
```

2.0.20 Python

```
slopes_py = b_py[1:] / sd.to_numpy()
intercept_py = b_py[0] - np.sum(slopes_py * mu.to_numpy())
print("gradient descent:", np.round(np.r_[intercept_py, slopes_py], 3))
```

```
gradient descent: [ 0.191 -0.054 -0.046  0.007  0.006 -0.004 -0.116]
```

```
print("library fit:     ", fit_py.params.round(3).to_numpy())
```

```
library fit:      [ 0.191 -0.054 -0.046  0.007  0.006 -0.004 -0.116]
```

```
same_py = (sig(Xte_py @ b_py) > 0.5) == (p_py.to_numpy() > 0.5)
print("same predictions:", same_py.mean())
```

```
same predictions: 1.0
```

Exercise 2.7 (Fitting the stock market model by gradient descent: how close?)**.** Use the setup of Exercise 2.5.
What is the largest absolute difference between the gradient descent[a] coefficients and those of Exercise 2.1?

[a] https://morrison-lab.github.io/mds/optimization-gradient-descent.html#def-gradient-descent

Solution

Solution 2.7. The largest absolute difference between the two sets of coefficients is 4.3e-10.

Exercise 2.8 (Fitting the stock market model by gradient descent (part 2))**.** Use the setup of Exercise 2.5.
On what proportion of the 2005 days do the two fits make the same prediction?

Solution

Solution 2.8. The two fits make the same prediction on 100% of the 2005 days. Gradient descent[a] ends near the `glm()` fit for three reasons.

```
# The curvature bound L of the average log loss for the standardized design.
gd_lip <- max(eigen(crossprod(Xs) / nrow(Xs), only.values = TRUE)$values) / 4
c(L = gd_lip, step_bound = 2 / gd_lip)
```

```
         L step_bound
 0.2823152  7.0842810
```

- The maximum likelihood[b] fit exists: `glm()` converged, with every fitted probability between 0.4 and 0.6.
- The total log loss is convex[c] (the logistic Hessian theorem), and strictly convex when the design matrix has full column rank[d] (the solution to part 2 of the Newton and IRLS exercise), so the maximum likelihood fit is its only minimizer.
- The step size is small enough. Every $\pi_i(1-\pi_i) \le \frac{1}{4}$, so the Hessian[e] of the average[f] log loss is at most $L = \frac{1}{4}\lambda_{\max}(\mathbf{D}_s{}^\top \mathbf{D}_s/n)$, where $\mathbf{D}_s$ is the standardized design matrix, and here $2/L = 7.08$, above the step size 0.5.

Standardizing the predictors puts the coefficients on similar scales, so one step size suits all of them.

[a] https://morrison-lab.github.io/mds/optimization-gradient-descent.html#def-gradient-descent
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-mle
[c] https://morrison-lab.github.io/mds/algebra-sums.html#def-convex-function
[d] https://morrison-lab.github.io/mds/linear-algebra-subspaces.html#def-full-column-rank
[e] https://morrison-lab.github.io/mds/vector-calculus.html#def-hessian
[f] https://morrison-lab.github.io/mds/linear-algebra-vectors.html#def-mean

Watch

Hastie and Tibshirani fit logistic regression in code in their Python lab[a] [3].

[a] https://www.youtube.com/watch?v=wOGBlLLuc4I

Read

The Chapter 4 lab of James et al. [1], `Ch04-classification-lab.ipynb` in ISLP_labs[a], fits the same logistic regression to the `Smarket` data, and then compares it with four other classifiers[b] on the same data [1, Sec. 4.7]. The discriminant analysis chapter names three of them. The classification lab works the logistic regression part.

[a] https://github.com/intro-stat-learning/ISLP_labs/blob/main/Ch04-classification-lab.ipynb
[b] https://morrison-lab.github.io/sds/classification.html#def-classification-rule

3 When the coefficients diverge

The fits so far assumed that the likelihood[12] has a finite maximizer. When a hyperplane puts every observation of one class[13] on one side and every observation of the other class on the other side, the likelihood has no finite maximizer [4]. Any $p+1$ points in general position in $\mathbb{R}^p$ can be split this way, whatever their classes[14], so such a split is common when the sample[15] is small relative to the number of predictors. Fitting software still stops and reports finite coefficients, and the Wald test[16] and the likelihood ratio test[17] reach opposite conclusions.

[12] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik
[13] https://morrison-lab.github.io/sds/classification.html#def-class
[14] https://morrison-lab.github.io/sds/classification.html#def-class
[15] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample
[16] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test
[17] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat

Exercise 3.1 (When the data separate perfectly)**.** A second trial, of six patients, records a dose x and recovery y:

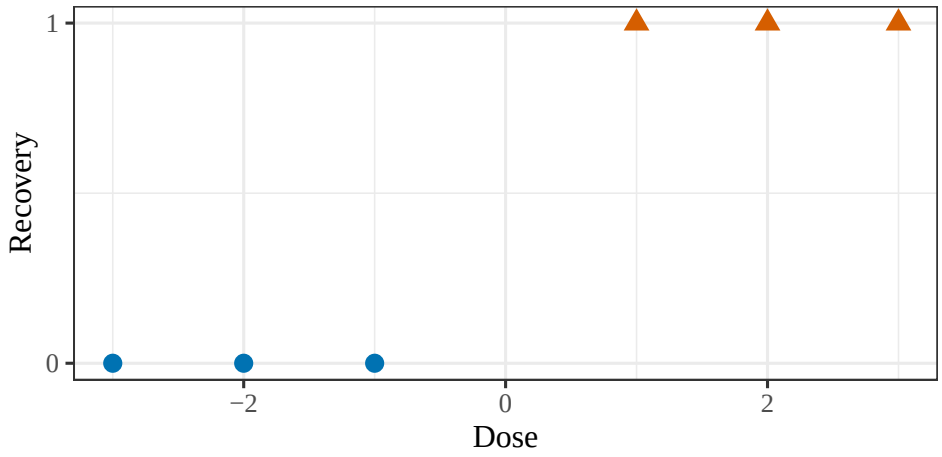
Table 1: The dose and recovery of each of six patients.

x	-3	-2	-1	$+1$	$+2$	$+3$
y	0	0	0	1	1	1

Every patient with a negative dose failed to recover, and every patient with a positive dose recovered. Show, as in part 3 of the worked fit exercise, that the score for β_0 is 0 at $\beta_0 = 0$ for every β_1. Figure 4 plots the six patients.

3.0.1 R

```
sep_dose <- c(-3, -2, -1, 1, 2, 3)
sep_y <- c(0, 0, 0, 1, 1, 1)
sep_recovery <- factor(sep_y)
sep_fig <- tibble::tibble(dose = sep_dose, y = sep_y, recovery = sep_recovery)
sep_colours <- c("0" = "#0072B2", "1" = "#D55E00")
sep_shapes <- c("0" = 16, "1" = 17)
sep_breaks <- c(0, 1)
sep_fig |>
  ggplot2::ggplot(ggplot2::aes(dose, y, colour = recovery, shape = recovery)) +
  ggplot2::geom_point(size = 3) +
  ggplot2::scale_colour_manual(values = sep_colours, guide = "none") +
  ggplot2::scale_shape_manual(values = sep_shapes, guide = "none") +
  ggplot2::scale_y_continuous(breaks = sep_breaks) +
  ggplot2::labs(x = "Dose", y = "Recovery")
```

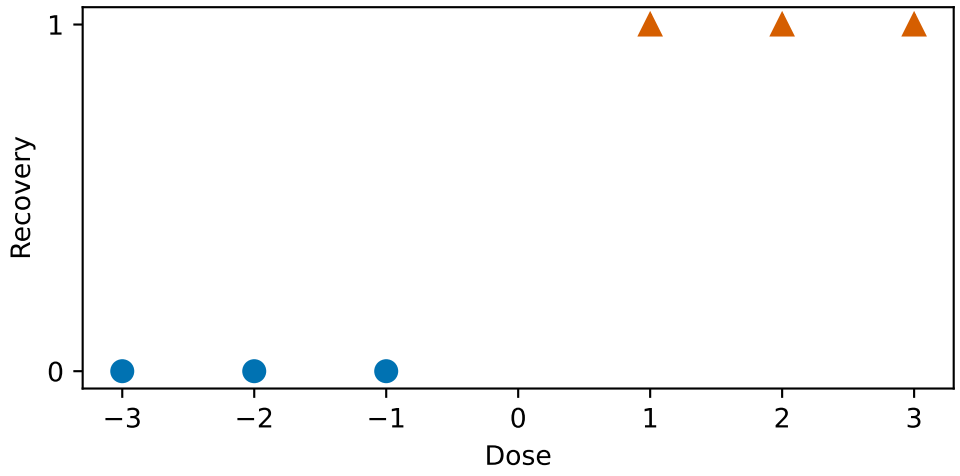


(a)

3.0.2 Python

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(5, 2.5), layout="constrained")
dose_sep = [-3, -2, -1]
ax.plot(dose_sep, [0, 0, 0], "o", color="#0072B2", markersize=8);
ax.plot([1, 2, 3], [1, 1, 1], "^", color="#D55E00", markersize=8);
ax.set_yticks([0, 1]);
ax.set_xlabel("Dose");
ax.set_ylabel("Recovery");
plt.show()
```



(b)

Figure 4: The six patients' doses and recoveries. Every patient with a negative dose has recovery 0, and every patient with a positive dose has recovery 1.

Solution

Solution 3.1. In Exercise 3.1, the doses of the three patients with $y = 0$ are the negatives of the doses of the three with $y = 1$. At $\beta_0 = 0$, $\pi_i = \text{expit}(\beta_1 x_i)$, and $\text{expit}(-z) + \text{expit}(z) = 1$ (part 2 of the sigmoid properties exercise), so

$$\begin{aligned}\sum_{i=1}^{n} \pi_i &= \sum_{k=1}^{3} [\text{expit}(-k\beta_1) + \text{expit}(k\beta_1)] && \text{(pair dose } -k \text{ with dose } k\text{)} \\ &= 3 && \text{(each pair sums to 1)} \\ &= \sum_{i=1}^{n} y_i.\end{aligned}$$

So the score for β_0, $\sum_{i=1}^{n}(y_i - \pi_i)$, is 0 for every β_1.

Exercise 3.2 (When the data separate perfectly: the log-likelihood)**.** Use the setup of Exercise 3.1. Show that with $\beta_0 = 0$ the log-likelihood[a] is

$$\ell(\beta_1) = 2\log\text{expit}(\beta_1) + 2\log\text{expit}(2\beta_1) + 2\log\text{expit}(3\beta_1).$$

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Solution

Solution 3.2. With $\beta_0 = 0$, a patient with $y_i = 1$ and dose k contributes $\log \operatorname{expit}(k\beta_1)$. A patient with $y_i = 0$ and dose $-k$ contributes

$$\log[1 - \operatorname{expit}(-k\beta_1)] = \log \operatorname{expit}(k\beta_1). \quad (1 - \operatorname{expit}(-z) = \operatorname{expit}(z))$$

Each $k \in \{1, 2, 3\}$ occurs once in each group, so each $\log \operatorname{expit}(k\beta_1)$ appears twice, which gives the stated ℓ.

Exercise 3.3 (When the data separate perfectly: an increasing log-likelihood)**.** Use the setup of Exercise 3.2.
Show that $\ell(\beta_1)$ is strictly increasing.

Solution

Solution 3.3. For each k, the derivative of $\log \operatorname{expit}(kz)$ with respect to z is $k[1 - \operatorname{expit}(kz)]$, so

$$\begin{aligned} \ell'(\beta_1) &= 2[1 - \operatorname{expit}(\beta_1)] + 4[1 - \operatorname{expit}(2\beta_1)] + 6[1 - \operatorname{expit}(3\beta_1)] && \text{(differentiate each term)} \\ &= 2\operatorname{expit}(-\beta_1) + 4\operatorname{expit}(-2\beta_1) + 6\operatorname{expit}(-3\beta_1) && (1 - \operatorname{expit}(z) = \operatorname{expit}(-z)) \\ &> 0 && (\operatorname{expit} > 0) \end{aligned}$$

for every β_1, so ℓ is strictly increasing.

Exercise 3.4 (When the data separate perfectly: the supremum)**.** Use the setup of Exercise 3.3.
Find $\sup_{\beta_1} \ell(\beta_1)$.

Solution

Solution 3.4. As $\beta_1 \to \infty$, each $\operatorname{expit}(k\beta_1) \to 1$, so each $\log \operatorname{expit}(k\beta_1) \to 0$. So $\sup_{\beta_1} \ell(\beta_1) = 0$, and the supremum of the likelihood[a] is 1.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik

Exercise 3.5 (When the data separate perfectly: is the supremum attained?)**.** Use the setup of Exercise 3.3.
Is the supremum of the log-likelihood[a] attained at any (β_0, β_1)?

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Solution

Solution 3.5. No. At any finite (β_0, β_1), every fitted probability is strictly between 0 and 1, so each of the six terms of the log-likelihood[a] is negative, and $\ell(\beta_0, \beta_1) < 0$. So no (β_0, β_1) maximizes the likelihood[b], and the maximum likelihood estimate[c] does not exist.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik
[c] https://morrison-lab.github.io/sds/intro-MLEs.html#def-mle

Definition 3.1 (Separated data (completely separated data))**.** A dataset $(\underset{\sim}{x}_1, y_1), \ldots, (\underset{\sim}{x}_n, y_n)$ with $y_i \in \{0, 1\}$, where each $\underset{\sim}{x}_i$ includes the leading 1 for the intercept[a], is **separated**, also called **completely separated** [4], when some $\vec{\beta}$ has $\vec{\beta} \cdot \underset{\sim}{x}_i > 0$ for every i with $y_i = 1$ and $\vec{\beta} \cdot \underset{\sim}{x}_i < 0$ for every i with $y_i = 0$.

Such a $\vec{\beta}$ is said to separate the data.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-intercept

Example 3.1 (A separated dataset and one that is not)**.**
The six patients of Exercise 3.1 are separated: $\vec{\beta} = (0, 1)^\top$ gives $\vec{\beta} \cdot \underset{\sim}{x}_i = x_i$, which is 1, 2 or 3 for the patients who recovered and -1, -2 or -3 for those who did not.
The six patients of the worked fit exercise are not separated. Each group contains a patient who recovered and a patient who did not, and those two patients have the same $\underset{\sim}{x}_i = (1, x_i)^\top$, so $\vec{\beta} \cdot \underset{\sim}{x}_i$ cannot be positive for one and negative for the other.

Definition 3.2 (Quasi-completely separated data)**.** A dataset with $y_i \in \{0, 1\}$ is **quasi-completely separated** [4] when it is not separated, but some $\vec{\beta} \neq \vec{0}$ has $\vec{\beta} \cdot \underset{\sim}{x}_i \geq 0$ for every i with $y_i = 1$ and $\vec{\beta} \cdot \underset{\sim}{x}_i \leq 0$ for every i with $y_i = 0$.

Example 3.2 (Four patients, quasi-completely separated)**.** Take four patients with doses $x = -1, 0, 0, 1$ and recoveries $y = 0, 0, 1, 1$. They are not separated: the two patients with dose 0 have the same $\underset{\sim}{x}_i$ and different outcomes. The vector $\vec{\beta} = (0, 1)^\top$ gives $\vec{\beta} \cdot \underset{\sim}{x}_i = -1, 0, 0, 1$, which is ≥ 0 for both patients who recovered and ≤ 0 for both who did not, so the data are quasi-completely separated.

```
quasi <- data.frame(x = c(-1, 0, 0, 1), y = c(0, 0, 1, 1))
quasi_fit <- glm(y ~ x, family = binomial, data = quasi)
summary(quasi_fit)$coefficients
```

```
               Estimate  Std. Error      z value Pr(>|z|)
(Intercept) 8.146903e-18    1.414214 5.760730e-18 1.000000
x           1.956607e+01 7604.235498 2.573049e-03 0.997947
```

`glm()` reports a slope[a] of 19.6, with a standard error[b] of 7604: the same symptoms as for the separated trial of Exercise 3.1. For quasi-completely separated data, too, the maximum likelihood estimate[c] does not exist [4].

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/sds/estimation.html#def-SE
[c] https://morrison-lab.github.io/sds/intro-MLEs.html#def-mle

Exercise 3.6 (When the data separate perfectly (part 2))**.** Use the setup of Exercise 3.1.
Now take any separated dataset, and suppose $\vec{\beta}$ separates it. Show that replacing $\vec{\beta}$ by $c\vec{\beta}$, for any $c > 1$, strictly increases the likelihood[a].

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik

Solution

Solution 3.6.
By the margin form theorem, each observation's contribution to the log-likelihood[a] is $-\log(1 + \mathrm{e}^{-m_i})$, with the margin $m_i = \tilde{y}_i \vec{\beta} \cdot \underset{\sim}{x}_i$ and $\tilde{y}_i = 2y_i - 1 \in \{-1, +1\}$, so

$$\ell(\vec{\beta}) = -\sum_{i=1}^{n} \log(1 + \mathrm{e}^{-m_i}).$$

This margin of an observation is not the margin of a separating hyperplane, which does not change when $\vec{\beta}$ is scaled.
Separation says that every margin is positive: $m_i > 0$ for all i. Scaling $\vec{\beta}$ to $c\vec{\beta}$ multiplies every m_i by c, so for $c > 1$ every m_i increases. Each term $-\log(1 + \mathrm{e}^{-m})$ is strictly increasing in m, so every term

increases. Hence $\ell(c\vec{\beta}) > \ell(\vec{\beta})$.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Exercise 3.7 (When the data separate perfectly: the supremum in general)**.** Use the setup of Exercise 3.6.
Find the supremum of the log-likelihood[a].

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Solution

Solution 3.7. Every term $-\log(1+\mathrm{e}^{-m_i})$ of Solution 3.6 is negative, so $\ell < 0$ at every finite parameter vector. Along the ray $c\vec{\beta}$, each term tends to 0 as $c \to \infty$, because $cm_i \to \infty$. So the supremum of the log-likelihood[a] is 0, and the supremum of the likelihood[b] is 1.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik

Exercise 3.8 (When the data separate perfectly: no maximizer in general)**.** Use the setup of Exercise 3.7.
Show that no finite parameter vector attains that supremum.

Solution

Solution 3.8. In Solution 3.7, every term of ℓ is negative, so $\ell < 0$ at every finite parameter vector, while the supremum is 0. So no finite parameter vector attains the supremum of the log-likelihood[a], or that of the likelihood[b]. Solution 3.4 and Solution 3.5 were this argument for one six-patient dataset.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik

Exercise 3.9 (When the data separate perfectly (part 3))**.** Use the setup of Exercise 3.1.
At $\beta_0 = 0$, the observed information[a] $\mathbf{D}^\top W \mathbf{D}$ is diagonal[b], and its entry for β_1 is

$$I(\beta_1) = \sum_{i=1}^{n} x_i^2 \, \pi_i (1 - \pi_i).$$

Evaluate $I(\beta_1)$ at $\beta_1 = 1, 2, 5, 10, 20$, and the standard error[c] $1/\sqrt{I(\beta_1)}$ that a fitting routine would report there.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-oinf
[b] https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-diagonal-matrix
[c] https://morrison-lab.github.io/sds/estimation.html#def-SE

Solution

Solution 3.9. In Exercise 3.9, the off-diagonal entry of $\mathbf{D}^\top W \mathbf{D}$ is $\sum_{i=1}^{n} x_i \pi_i (1-\pi_i)$, which is 0 at $\beta_0 = 0$ because the doses k and $-k$ have the same weight $\mathrm{expit}(k\beta_1)\mathrm{expit}(-k\beta_1)$. So the standard error[a] of β_1, the square root of the corresponding diagonal[b] entry of $(\mathbf{D}^\top W \mathbf{D})^{-1}$, is $1/\sqrt{I(\beta_1)}$. At the five values:

```r
sep <- tibble::tibble(x = c(-3, -2, -1, 1, 2, 3), y = c(0, 0, 0, 1, 1, 1))
sep_info <- function(b1) {
  p <- plogis(b1 * sep$x)
  sum(sep$x^2 * p * (1 - p))
}
sep_b1 <- c(1, 2, 5, 10, 20)
sep_i <- vapply(sep_b1, sep_info, numeric(1))
sep_se <- 1 / sqrt(sep_i)

# Typeset a number for the page: plain if it is not tiny,
# and as $m \times 10^{e}$ when its exponent is -4 or lower.
sep_num <- function(v, digits = 3) {
  v <- signif(v, digits)
  e <- floor(log10(abs(v)))
  if (e > -4) {
    return(format(v))
  }
  sprintf("$%s \\times 10^{%d}$", format(signif(v / 10^e, digits)), e)
}
sep_nums <- function(v, digits = 3) {
  vapply(v, sep_num, "", digits = digits)
}

knitr::kable(
  tibble::tibble(
    b1 = sep_b1,
    info = sep_nums(sep_i),
    se = sep_nums(sep_se)
  ),
  col.names = c(
    "$\\coef{1}$", "$\\oinf(\\coef{1})$", "$\\mathrm{SE} = 1/\\sqrt{\\oinf}$"
  ),
  align = "r"
)
```

β_1	$I(\beta_1)$	$\mathrm{SE} = 1/\sqrt{I}$
1	2.05	0.699
2	0.396	1.59
5	0.0137	8.55
10	9.08×10^{-5}	105
20	4.12×10^{-9}	15600

The information tends to 0 as β_1 grows, because every π_i tends to 0 or 1, where $\pi_i(1-\pi_i)$ is 0, so the standard error grows without bound. The slope[c] of ℓ, the score of Solution 3.3, also tends to 0: far out along β_1 the log-likelihood[d] is nearly flat.

[a] https://morrison-lab.github.io/sds/estimation.html#def-SE
[b] https://morrison-lab.github.io/mds/linear-algebra-special-matrices.html#def-diagonal-matrix
[c] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[d] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Example 3.3 (When the data separate perfectly: what software reports)**.**
Fitting the six patients of Exercise 3.1 with R's `glm()`:

```r
sep_fit <- glm(y ~ x, family = binomial, data = sep)
```

```r
sep_b1_hat <- coef(sep_fit)[["x"]]
sep_p_hat <- summary(sep_fit)$coefficients["x", "Pr(>|z|)"]
# The slope's p-value, formatted by insight::format_p()
sep_p_printed <- insight::format_p(sep_p_hat)
parameters::parameters(sep_fit) |> parameters::print_md(digits = 3)
```

Parameter	Log-Odds	SE	95% CI	z	p
(Intercept)	-9.013e-08	48419.140	(-94899.770, 94899.770)	-1.862e-12	> .999
x	23.268	48418.863	(-94875.958, 94922.495)	4.806e-04	> .999

R warns, "fitted probabilities numerically 0 or 1 occurred". That is, some fitted probabilities are within rounding error of 0 or 1.

Example 3.4 (When the data separate perfectly: why the fit stops)**.**
In Example 3.3, `glm()` stops and reports a slope[a]. Far out along β_1 the log-likelihood[b] is nearly flat (Solution 3.9). There the gradient and the change in fit are both small, so a fitting routine stops there, at a β_1 set by its tolerance, without having found a maximum. `glm()` stops after 24 iterations, when the relative change in the deviance[c] falls below its tolerance, `epsilon = 1e-8` by default. The reported slope 23.27 is set by that tolerance; it is not an estimate[d] of a finite parameter. Running the same fit with a tighter tolerance:

```r
sep_tight <- suppressWarnings(
  glm(y ~ x, family = binomial, data = sep,
      control = glm.control(epsilon = 1e-14, maxit = 100))
)
c(slope = coef(sep_tight)[["x"]], iterations = sep_tight$iter)
```

```
    slope iterations
 30.68281   32.00000
```

returns a larger slope, 30.68, after 32 iterations.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[c] https://morrison-lab.github.io/sds/logistic-inference.html#def-deviance
[d] https://morrison-lab.github.io/sds/estimation.html#def-estimate

Exercise 3.10 (When the data separate perfectly (part 4))**.** Use the setup of Exercise 3.9.
The Wald test[a] of $H_0 : \beta_1 = 0$ uses $z = \beta_1 \sqrt{I(\beta_1)}$, the slope[b] divided by its standard error[c], evaluated at the fitted slope. Evaluate z and its two-sided p-value at the five values of Exercise 3.9.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[c] https://morrison-lab.github.io/sds/estimation.html#def-SE

Solution

Solution 3.10. In Exercise 3.10, the Wald statistic $z = \beta_1 \sqrt{I(\beta_1)}$ first rises and then falls as β_1 grows:

```r
sep_z <- sep_b1 * sqrt(sep_i)
sep_p <- 2 * pnorm(-sep_z)
sep_wald <- rbind(
  "$z$" = sep_nums(sep_z),
  "$p$" = sep_nums(sep_p)
)
colnames(sep_wald) <- paste0("$\\coef{1} = ", sep_b1, "$")
sep_peak <- optimize(
  \(b1) b1 * sqrt(sep_info(b1)),
  interval = c(0, 10),
  maximum = TRUE
)
sep_g2 <- 12 * log(2)
sep_g2_p <- pchisq(sep_g2, df = 1, lower.tail = FALSE)
knitr::kable(sep_wald, align = "r")
```

	$\beta_1 = 1$	$\beta_1 = 2$	$\beta_1 = 5$	$\beta_1 = 10$	$\beta_1 = 20$
z	1.43	1.26	0.584	0.0953	0.00128
p	0.153	0.208	0.559	0.924	0.999

Its largest value is about 1.44, at $\beta_1 \approx 1.12$, which is below 1.96, the two-sided 5% critical value, and after that it decreases toward 0. So beyond $\beta_1 \approx 1.12$, the further the fit moves toward the supremum of the likelihood[a], the larger the Wald p-value. R's fit in Example 3.3 gives the slope[b] a p-value of 0.9996, printed as $p > .999$, on data where the sign of the dose predicts the outcome without error.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope

Exercise 3.11 (When the data separate perfectly: the likelihood ratio test)**.** Use the setup of Exercise 3.10.
Compute the likelihood ratio statistic[a] $G^2 = 2[\sup \ell - \ell(0,0)]$ for $H_0 : \beta_1 = 0$, and its p-value from the χ^2_1 distribution[b].

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat
[b] https://morrison-lab.github.io/pds/random-variables.html#def-cdf

Solution

Solution 3.11. Under H_0 the slope[a] is 0, and the maximum likelihood estimate[b] of the intercept[c] is $\text{logit}\{3/6\} = 0$, so the null fit has log-likelihood[d]

$$\begin{aligned} \ell(0,0) &= 6 \log \tfrac{1}{2} \\ &= -6 \log 2. \end{aligned}$$

The unrestricted fit has no maximizer, so the supremum $\sup \ell = 0$ takes the place of the maximized log-likelihood in the likelihood ratio statistic[e]:

$$\begin{aligned} G^2 &= 2[0 - (-6 \log 2)] \\ &= 12 \log 2 \\ &= 8.3178. \end{aligned}$$

Against the χ^2_1 distribution[f], G^2 gives $p = 0.0039$. G^2 is the null deviance[g] minus the deviance[h] of the fit, and R's fit reports a null deviance of 8.3178 and a deviance of 3.1×10^{-10}.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope

[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-mle
[c] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-intercept
[d] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[e] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat
[f] https://morrison-lab.github.io/pds/random-variables.html#def-cdf
[g] https://morrison-lab.github.io/sds/logistic-inference.html#def-null-deviance
[h] https://morrison-lab.github.io/sds/logistic-inference.html#def-deviance

Exercise 3.12 (When the data separate perfectly: which test to use)**.** Use the setup of Exercise 3.11. The Wald test[a] of Exercise 3.10 and the likelihood ratio test[b] of Exercise 3.11 reach opposite conclusions. Which of the two should you use here?

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat

Solution

Solution 3.12. Use the likelihood ratio test[a]. The Wald statistic is the slope[b] divided by its standard error[c], and both grow without bound as $\beta_1 \to \infty$. The standard error $1/\sqrt{I(\beta_1)}$ grows exponentially, faster than the slope, so their ratio tends to 0. The Wald test[d] uses the curvature of ℓ at the fitted slope, far out, where ℓ is nearly flat. The likelihood ratio test[e] uses the change in log-likelihood[f] between $\beta_1 = 0$ and the fit, which measures how much better the separated fit describes the data.
On a separated or nearly separated dataset, then, a large reported standard error is a symptom of separation, not a sign of weak evidence.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[c] https://morrison-lab.github.io/sds/estimation.html#def-SE
[d] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test
[e] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat
[f] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Exercise 3.13 (When the data separate perfectly: how far to trust the p-value)**.** Use the setup of Exercise 3.11.
The p-value of Solution 3.11 relies on the χ^2_1 approximation. With only six patients, compute an exact p-value instead, by conditioning on the number of recoveries.

Solution

Solution 3.13. The trial of Exercise 3.1 has 3 recoveries. Under H_0, each of the $\binom{6}{3} = 20$ ways to choose which three patients recover is equally likely. Only the observed choice gives the largest possible $\sum_{i=1}^{n} x_i y_i$, so the one-sided exact p-value is $1/20 = 0.05$. The two-sided one is 0.1; it also counts the mirror-image outcome. Both are weaker evidence than the χ^2_1 value of Solution 3.11 suggests, but they point the same way.

Definition 3.3 (Hauck–Donner effect)**.** In logistic regression, the **Hauck–Donner effect** is the decrease of the Wald statistic $z = \hat{\beta}_j / \widehat{\mathrm{SE}}(\hat{\beta}_j)$ for $H_0 : \beta_j = 0$ toward 0 as $|\hat{\beta}_j|$ grows large. It happens because the standard error[a] grows faster than $|\hat{\beta}_j|$ [5]. As a result, the Wald test[b] can fail to reject H_0 when the coefficient is far from 0.

[a] https://morrison-lab.github.io/sds/estimation.html#def-SE
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test

Example 3.5 (The Hauck–Donner effect in the separated trial)**.** Take the six separated patients of Exercise 3.1, with $\beta_0 = 0$. The Wald statistic for the slope[a] is $\beta_1\sqrt{I(\beta_1)}$ (Exercise 3.10).

```
# The Wald z for the slope of the six separated patients, at beta_0 = 0.
hd_z <- function(b1) {
  x <- c(-3, -2, -1, 1, 2, 3)
  p <- plogis(b1 * x)
  b1 * sqrt(sum(x^2 * p * (1 - p)))
}
hd_z_values <- c(slope_2 = hd_z(2), slope_20 = hd_z(20))
hd_p_values <- 2 * pnorm(-hd_z_values)
rbind(z = hd_z_values, p = hd_p_values)
```

```
    slope_2    slope_20
z 1.2580715 0.001284104
p 0.2083659 0.998975434
```

A fitting routine that stopped at the estimate[b] $\hat{\beta}_1 = 2$ would report a Wald statistic of 1.26, with $p = 0.208$. One that stopped at $\hat{\beta}_1 = 20$ would report 0.0013, with $p = 0.999$. The estimate ten times further from 0 gets the smaller z and the larger p-value.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/sds/estimation.html#def-estimate

Exercise 3.14 (When the data separate perfectly (part 5))**.** Use the setup of Exercise 3.2. Consider the ridge-penalized log-likelihood[a] $f(\beta_1) = \ell(\beta_1) - \frac{\lambda}{2}\beta_1^2$, with $\beta_0 = 0$ as before. Show that f has a unique maximizer for every $\lambda > 0$.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Solution

Solution 3.14. Every $\log\text{expit}$ is negative, so $\ell < 0$ everywhere, and therefore $f(\beta_1) < -\frac{\lambda}{2}\beta_1^2$, which tends to $-\infty$ as $|\beta_1| \to \infty$. A continuous function that tends to $-\infty$ in both directions attains a maximum, so a maximizer exists. It is unique because f is strictly concave: $f''(\beta_1) = -I(\beta_1) - \lambda$, and $I(\beta_1) > 0$ (Exercise 3.9).

Exercise 3.15 (When the data separate perfectly: the penalized slope)**.** Use the setup of Exercise 3.14. Find the maximizer $\hat{\beta}_1$ of f for $\lambda = 1$, 0.1 and 0.01.

Solution

Solution 3.15. Setting $f'(\beta_1) = 0$, with the score of Solution 3.3, gives

$$2\text{expit}(-\beta_1) + 4\text{expit}(-2\beta_1) + 6\text{expit}(-3\beta_1) = \lambda\beta_1.$$

The left-hand side is positive, so the root is positive. For $\beta_1 \geq 0$ the left-hand side decreases from 6 at $\beta_1 = 0$ toward 0, while the right-hand side increases from 0, so the two sides are equal at exactly one point. Solving numerically:

```r
sep_lambda <- c(1, 0.1, 0.01)
sep_ridge_fit <- function(lambda) {
  uniroot(
    \(b1) 2 * plogis(-b1) + 4 * plogis(-2 * b1) + 6 * plogis(-3 * b1) - lambda * b1,
    interval = c(0, 100),
    tol = 1e-10
  )$root
}
sep_ridge <- vapply(sep_lambda, sep_ridge_fit, numeric(1))
sep_ridge_tbl <- rbind("$\\hcoef{1}$" = round(sep_ridge, 3))
colnames(sep_ridge_tbl) <- paste0("$\\lambda = ", sep_lambda, "$")
knitr::kable(sep_ridge_tbl, align = "r")
```

	$\lambda = 1$	$\lambda = 0.1$	$\lambda = 0.01$
$\hat{\beta}_1$	1.104	2.295	3.946

Exercise 3.16 (When the data separate perfectly: what the penalty costs)**.** Use the setup of Exercise 3.15.
What does the penalty achieve, and at what cost?

Solution

Solution 3.16. In Solution 3.14, the ridge penalty gives a finite, unique maximizer.
The cost is that the maximizer depends strongly on λ. It is 3.6 times larger at $\lambda = 0.01$ than at $\lambda = 1$, and it tends to infinity as $\lambda \to 0$, because the likelihood[a] alone puts no upper bound on the slope[b]. The penalty adds no information about the slope from the data; it adds a preference for smaller coefficients, and its maximizer reflects that preference as much as the data.
That regularization suits a problem where prior knowledge rules out very large coefficients, and a report of the penalized fit should state λ. What these six patients show is not that $\hat{\beta}_1 = 2.295$. They show that the dose separates the outcomes, that the evidence against $\beta_1 = 0$ is modest (exact one-sided $p = 0.05$, Solution 3.13), and that the data put no upper bound on the slope.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope

Theorem 3.1 (Separated data have no maximum likelihood estimate)**.** *Suppose $\vec{\beta}$ separates the data (Definition 3.1). Then $\ell(c\vec{\beta}) > \ell(\vec{\beta})$ for every $c > 1$. The supremum of the log-likelihood[a] ℓ is 0, approached as $c \to \infty$ along the ray $c\vec{\beta}$, and no finite parameter vector attains it, so the maximum likelihood estimate[b] of the logistic regression does not exist.*

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-mle

Proof

Proof. See Solution 3.6, Solution 3.7 and Solution 3.8. □

Example 3.6 (The likelihood keeps rising along a ray)**.** Consider the six patients of Exercise 3.1, with doses $-3, -2, -1, 1, 2, 3$ and recoveries $0, 0, 0, 1, 1, 1$. The vector $\vec{\beta} = (0, 1)^\top$ separates them (Example 3.1). Scale it by $c = 1, 2, 5, 10$, which gives $\beta_0 = 0$ and $\beta_1 = c$.

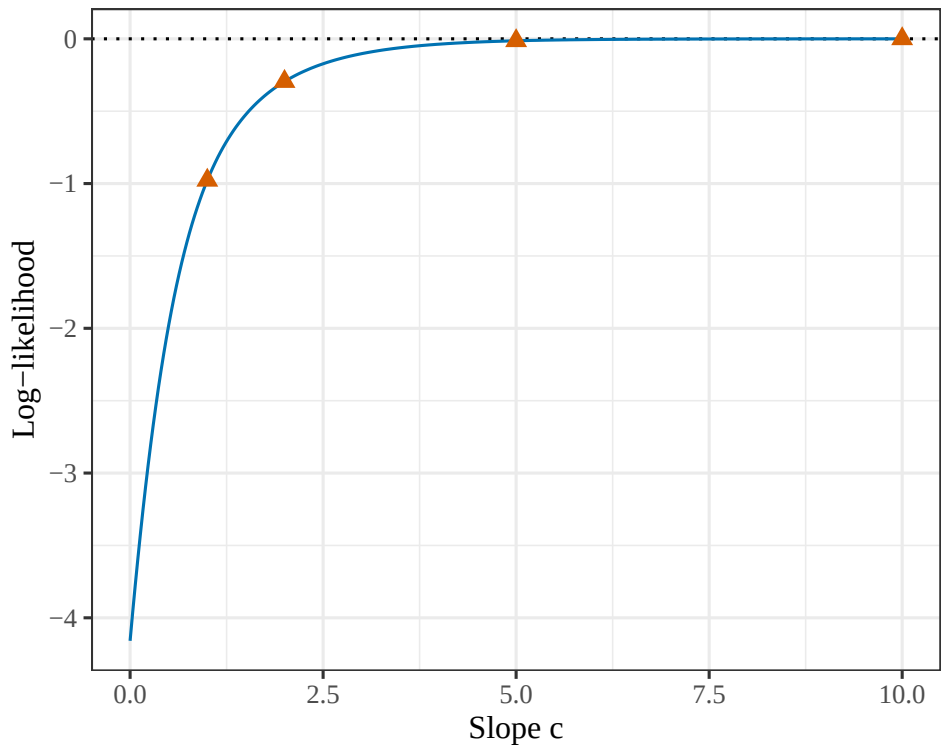
```r
sp_x <- c(-3, -2, -1, 1, 2, 3)
sp_y <- c(0, 0, 0, 1, 1, 1)
sp_llik <- function(b1) sum(dbinom(sp_y, 1, plogis(b1 * sp_x), log = TRUE))
sp_c <- c(1, 2, 5, 10)
data.frame(c = sp_c, loglik = vapply(sp_c, sp_llik, numeric(1)))
```

```
# A tibble: 4 x 2
      c     loglik
  <dbl>      <dbl>
1     1 -0.978
2     2 -0.295
3     5 -0.0135
4    10 -0.0000908
```

As Theorem 3.1 states, the log-likelihood[a] increases with c, from -0.978 at $c = 1$ to -9.08e-05 at $c = 10$. It approaches its supremum 0 and does not reach it, so the likelihood[b] approaches its supremum 1 and does not reach it either.
Figure 5 plots the curve against c.

3.0.3 R

```r
library(ggplot2)
sep_grid <- data.frame(c = seq(0, 10, length.out = 300))
sep_grid$loglik <- vapply(sep_grid$c, sp_llik, numeric(1))
sep_marks <- data.frame(c = sp_c, loglik = vapply(sp_c, sp_llik, numeric(1)))
ggplot(sep_grid) +
  aes(c, loglik) +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_line(colour = "#0072B2") +
  geom_point(data = sep_marks, colour = "#D55E00", shape = 17, size = 2.5) +
  labs(x = "Slope c", y = "Log-likelihood")
```



(a)

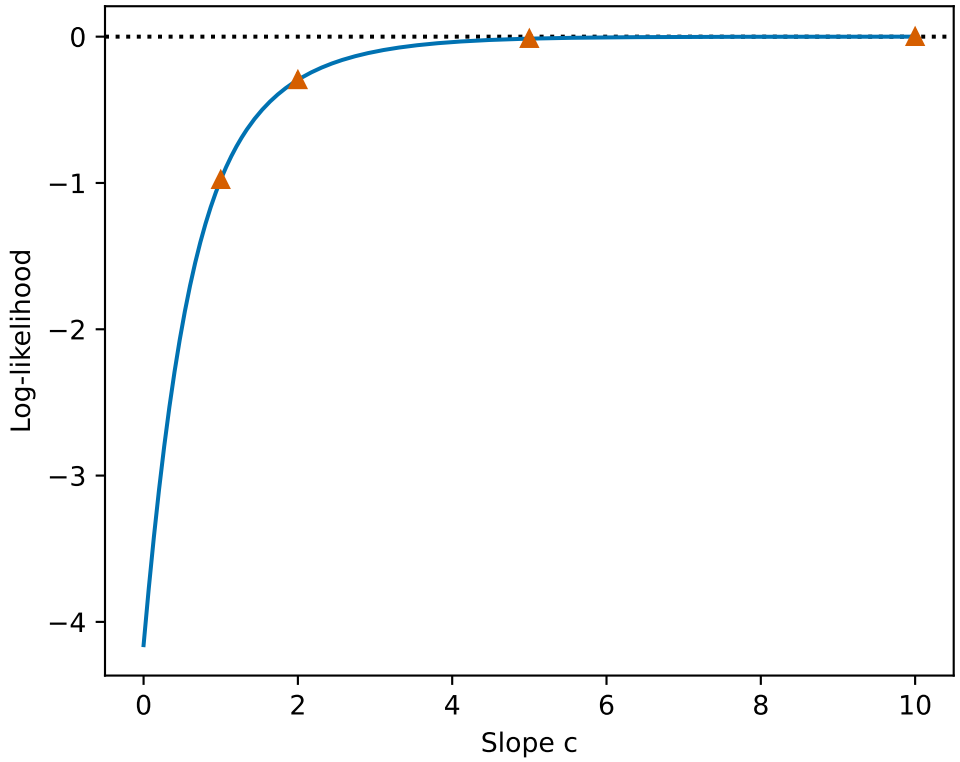
3.0.4 Python

```
import matplotlib.pyplot as plt
import numpy as np

x_sep = np.array([-3, -2, -1, 1, 2, 3])
y_sep = np.array([0, 0, 0, 1, 1, 1])

def llik_sep(b1):
    p = 1 / (1 + np.exp(-b1 * x_sep))
    return np.sum(y_sep * np.log(p) + (1 - y_sep) * np.log(1 - p))

c_sep = np.linspace(0, 10, 300)
marks_sep = np.array([1, 2, 5, 10])
fig, ax = plt.subplots(figsize=(5, 4), layout="constrained")
ax.axhline(0, color="black", linestyle=":");
ax.plot(c_sep, [llik_sep(c) for c in c_sep], color="#0072B2");
ax.plot(marks_sep, [llik_sep(c) for c in marks_sep], color="#D55E00", marker="^", linestyle="");
ax.set_xlabel("Slope c");
ax.set_ylabel("Log-likelihood");
plt.show()
```



(b)

Figure 5: The log-likelihood[c] of Example 3.6 at $\beta_0 = 0$ and $\beta_1 = c$ (solid), with the four values of c from the example marked. The curve rises toward its supremum 0 (dotted) and does not reach it.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik
[c] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Example 3.7 (Gradient descent on separated data has nowhere to stop)**.** Table 2 lists four observations[a] with one feature x_i, so $\underset{\sim}{x}_i = (1, x_i)^\top$ and $\vec{\beta} = (\beta_0, \beta_1)^\top$.

Table 2: Four observations[b] whose labels switch from 0 to 1 between $x = 2$ and $x = 4$.

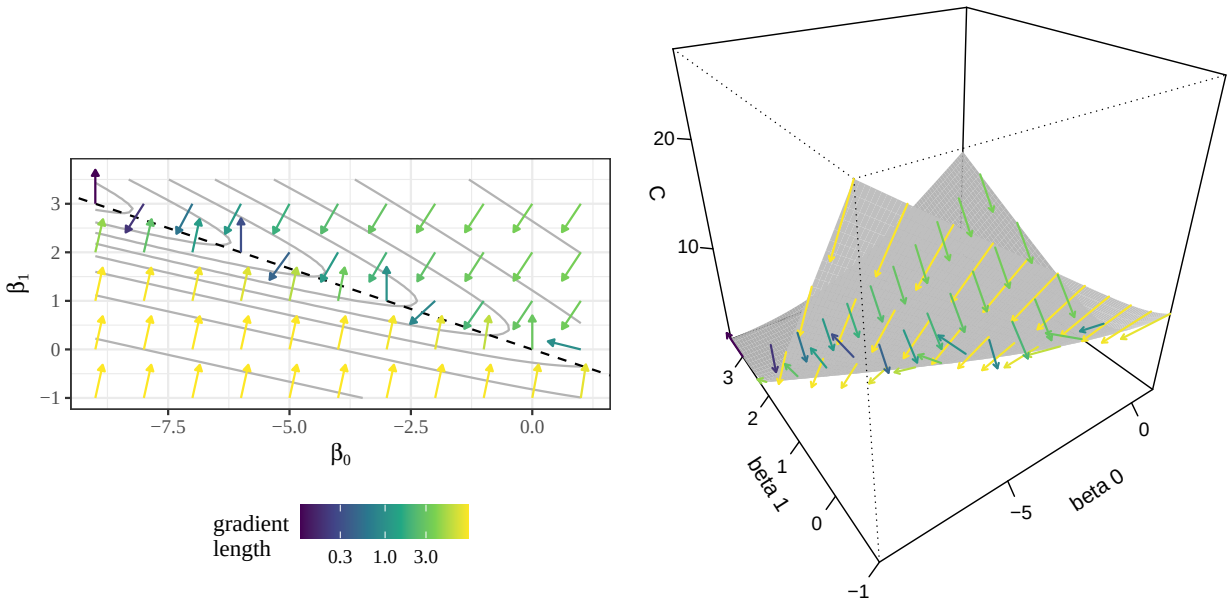
i	x_i	y_i
1	1	0
2	2	0
3	4	1
4	5	1

The vector $\vec{\beta} = (-3, 1)^\top$ separates these observations[c]: $\vec{\beta} \cdot \underset{\sim}{x}_i = x_i - 3$ is negative for the two with $y_i = 0$ and positive for the two with $y_i = 1$. So, by Theorem 3.1, the total log loss $C(\vec{\beta})$ has no minimizer[d]. The log-likelihood[e] $\ell = -C$ is concave, so C is convex[f], and any point where the gradient $\nabla C(\vec{\beta})$ is $\vec{0}$ would minimize C[g]. No point does, so $\nabla C(\vec{\beta})$ is never $\vec{0}$, and gradient descent[h] never reaches a point where it stops moving.
Figure 6a draws the descent direction[i] $-\nabla C(\vec{\beta})$ at a grid of points, on the contours of C, and Figure 6b draws the same arrows on its surface. By the logistic gradient theorem, $\nabla C(\vec{\beta}) = \sum_{i=1}^{n} (\pi_i - y_i) \underset{\sim}{x}_i$, so

$$-\nabla C(\vec{\beta}) \cdot (-3, 1)^\top = \sum_{i=1}^{n} (\pi_i - y_i)(3 - x_i).$$

Each term is positive: an observation with $y_i = 0$ has $\pi_i > 0$ and $x_i < 3$, and one with $y_i = 1$ has $\pi_i < 1$ and $x_i > 3$. So every arrow has a part pointing along the dashed line $\beta_0 + 3\beta_1 = 0$ toward the upper left, where C keeps falling toward 0 without reaching it.

[a] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-observation
[b] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-observation
[c] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-observation
[d] https://morrison-lab.github.io/mds/algebra-basics.html#def-global-minimizer
[e] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[f] https://morrison-lab.github.io/mds/algebra-sums.html#def-convex-function
[g] https://morrison-lab.github.io/mds/optimization-convexity.html#cor-convex-stationary
[h] https://morrison-lab.github.io/mds/optimization-gradient-descent.html#def-gradient-descent
[i] https://morrison-lab.github.io/mds/optimization-gradient-descent.html#def-descent-direction



(a) Contours of C with the descent direction[a] at each grid point.

(b) Surface of C with the same arrows, each drawn in the tangent plane[a] where it starts.

[a] https://morrison-lab.github.io/mds/optimization-gradient-descent.html#def-descent-direction

[a] https://morrison-lab.github.io/mds/optimization-convexity.html#def-tangent-plane

Figure 6: The descent direction[18] $-\nabla C(\vec{\beta})$ (arrows, all drawn equally long) for the total log loss C of the data in Table 2. An arrow's color shows the length[19] $\left\|\nabla C(\vec{\beta})\right\|$ of the gradient, on a logarithmic scale. The dashed line is $\beta_0 + 3\beta_1 = 0$, and every nonzero $\vec{\beta}$ on it has $\vec{\beta} \cdot \underset{\sim}{x} = 0$ exactly where $x = 3$.

ℹ Remark (Why the contour levels double)

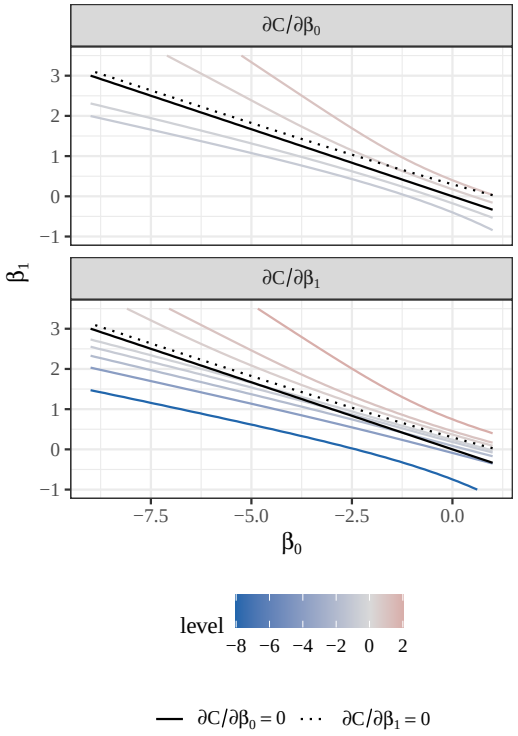
Remark 3.1 (Why the contour levels double). The contours in Figure 6a are drawn at values of C that double from 1/8 to 16, because over the plotted window C runs from 0.096, slightly above the dashed line at the upper left, to 27, at the lower left. Evenly spaced levels would crowd into the lower left and leave none where the arrows head.

Example 3.8 (The zero curves of the two partial derivatives never meet)**.** For the four observations[a] of Table 2, the gradient $\nabla C(\vec{\beta})$ of the total log loss is $\vec{0}$ only where both partial derivatives[b] are 0, so such a point would lie on the zero curve of each. Example 3.7 showed that

$$3\frac{\partial C}{\partial \beta_0} - \frac{\partial C}{\partial \beta_1} = \sum_{i=1}^{n} (\pi_i - y_i)(3 - x_i) > 0$$

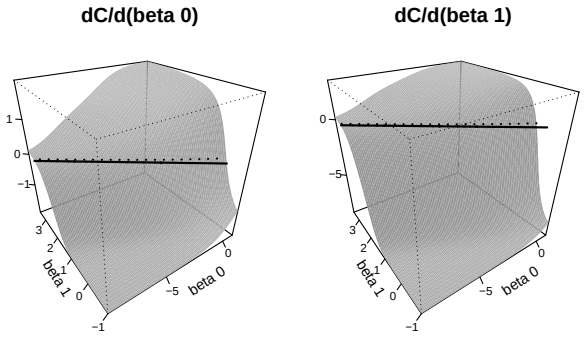
for every $\vec{\beta}$. So wherever $\frac{\partial C}{\partial \beta_0} = 0$, $\frac{\partial C}{\partial \beta_1}$ is negative, and the two zero curves never meet.
Figure 7a draws each partial derivative as level curves[c] and Figure 7b as a surface, with both zero curves in every panel. The zero curve of $\frac{\partial C}{\partial \beta_0}$ (solid) is the line $\beta_0 + 3\beta_1 = 0$, because $\frac{\partial C}{\partial \beta_0} = \sum_{i=1}^{n} (\pi_i - y_i)$ and on that line the four predicted probabilities[d] are $\text{expit}\{-2\beta_1\}$, $\text{expit}\{-\beta_1\}$, $\text{expit}\{\beta_1\}$ and $\text{expit}\{2\beta_1\}$, which pair up as $\text{expit}\{-t\} + \text{expit}\{t\} = 1$ and so add up to 2, the number of observations with $y_i = 1$. Off the line, $\frac{\partial C}{\partial \beta_0}$ is not 0, because it increases with β_0 for each fixed β_1. The zero curve of $\frac{\partial C}{\partial \beta_1}$ (dotted) lies above the solid curve: on the solid curve $\frac{\partial C}{\partial \beta_1}$ is negative, for the reason given at the start of this example, and $\frac{\partial C}{\partial \beta_1}$ increases with β_1 for each fixed β_0, because the partial derivative of $\frac{\partial C}{\partial \beta_1}$ in β_1 is $\sum_{i=1}^{n} \pi_i(1 - \pi_i)x_i^2 > 0$.

[a] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-observation
[b] https://morrison-lab.github.io/mds/vector-calculus.html#def-partial-derivative
[c] https://morrison-lab.github.io/mds/vector-calculus.html#def-level-curve
[d] https://morrison-lab.github.io/sds/classification.html#def-predicted-probability



(a) Level curves[a] of each partial derivative[b], with both zero curves.

[a] https://morrison-lab.github.io/mds/vector-calculus.html#def-level-curve
[b] https://morrison-lab.github.io/mds/vector-calculus.html#def-partial-derivative



(b) Each partial derivative[a] as a surface, with both zero curves at height 0.

[a] https://morrison-lab.github.io/mds/vector-calculus.html#def-partial-derivative

Figure 7: The partial derivatives[20] of the total log loss C for the data in Table 2, colored from blue (negative) through gray to red (positive). The solid black curve is where $\frac{\partial C}{\partial \beta_0} = 0$, and the dotted black curve is where $\frac{\partial C}{\partial \beta_1} = 0$. The solid curve is the dashed line $\beta_0 + 3\beta_1 = 0$ of Figure 6.

Remark (Why these level curves)

Remark 3.2 (Why these level curves). The level curves[a] in Figure 7a are drawn at ± 0.5, ± 1, ± 2, -4 and -8, because over the plotted window $\frac{\partial C}{\partial \beta_0}$ runs from -2 to 2 and $\frac{\partial C}{\partial \beta_1}$ from -9 to 3. Levels that double away from 0 cover both of them with the same colors, and leave room near 0 for the zero curves.

[a] https://morrison-lab.github.io/mds/vector-calculus.html#def-level-curve

3.0.5 R

```r
library(ggplot2)

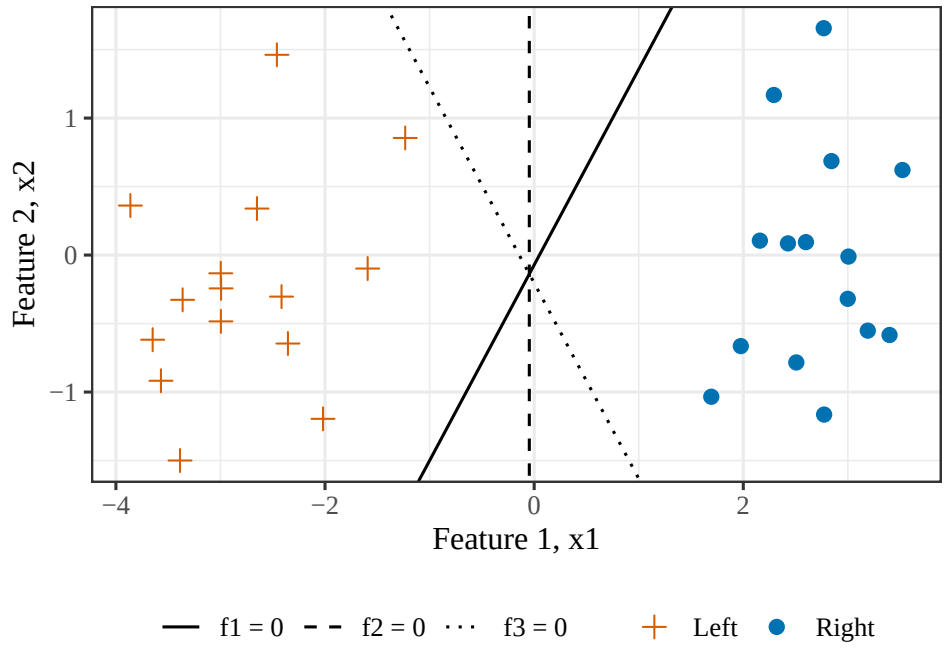
set.seed(571)
n_per_class <- 15
sim_df <- data.frame(
  x1 = c(rnorm(n_per_class, -2.5, 0.6), rnorm(n_per_class, 2.5, 0.6)),
  x2 = rnorm(2 * n_per_class, 0, 0.9),
  class = rep(c("Left", "Right"), each = n_per_class)
)

# Each boundary passes through the midpoint of the class means,
# at a different angle (in degrees) from the horizontal axis.
pivot <- c(mean(sim_df$x1), mean(sim_df$x2))
angles <- c(f1 = 55, f2 = 90, f3 = 125)
boundary_df <- data.frame(
  boundary = paste(names(angles), "= 0"),
  angle = angles * pi / 180
)

# Check that every boundary separates the two classes.
side <- function(angle) {
  (sim_df$x1 - pivot[1]) * sin(angle) - (sim_df$x2 - pivot[2]) * cos(angle) > 0
}
stopifnot(all(vapply(
  boundary_df$angle,
  function(a) nrow(unique(data.frame(side(a), sim_df$class))) == 2,
  logical(1)
)))

# geom_abline cannot draw a vertical line, so draw each boundary as a segment.
boundary_df$x_start <- pivot[1] - 10 * cos(boundary_df$angle)
boundary_df$x_end <- pivot[1] + 10 * cos(boundary_df$angle)
boundary_df$y_start <- pivot[2] - 10 * sin(boundary_df$angle)
boundary_df$y_end <- pivot[2] + 10 * sin(boundary_df$angle)

ggplot(sim_df) +
  aes(x1, x2) +
  geom_point(aes(colour = class, shape = class), size = 2.5) +
  geom_segment(
    data = boundary_df,
    aes(x = x_start, y = y_start, xend = x_end, yend = y_end, linetype = boundary)
  ) +
  scale_colour_manual(values = c(Left = "#D55E00", Right = "#0072B2")) +
  scale_shape_manual(values = c(Left = 3, Right = 16)) +
  scale_linetype_manual(values = c(
    "f1 = 0" = "solid", "f2 = 0" = "dashed", "f3 = 0" = "dotted"
  )) +
  coord_cartesian(xlim = range(sim_df$x1), ylim = range(sim_df$x2)) +
  labs(x = "Feature 1, x1", y = "Feature 2, x2", colour = NULL, shape = NULL, linetype = NULL)
```



(a)

3.0.6 Python

```python
import matplotlib.pyplot as plt
import numpy as np

rng = np.random.default_rng(571)
n_per_class = 15
left = np.column_stack([rng.normal(-2.5, 0.6, n_per_class),
                        rng.normal(0, 0.9, n_per_class)])
right = np.column_stack([rng.normal(2.5, 0.6, n_per_class),
                         rng.normal(0, 0.9, n_per_class)])
points = np.vstack([left, right])
is_right = np.repeat([False, True], n_per_class)

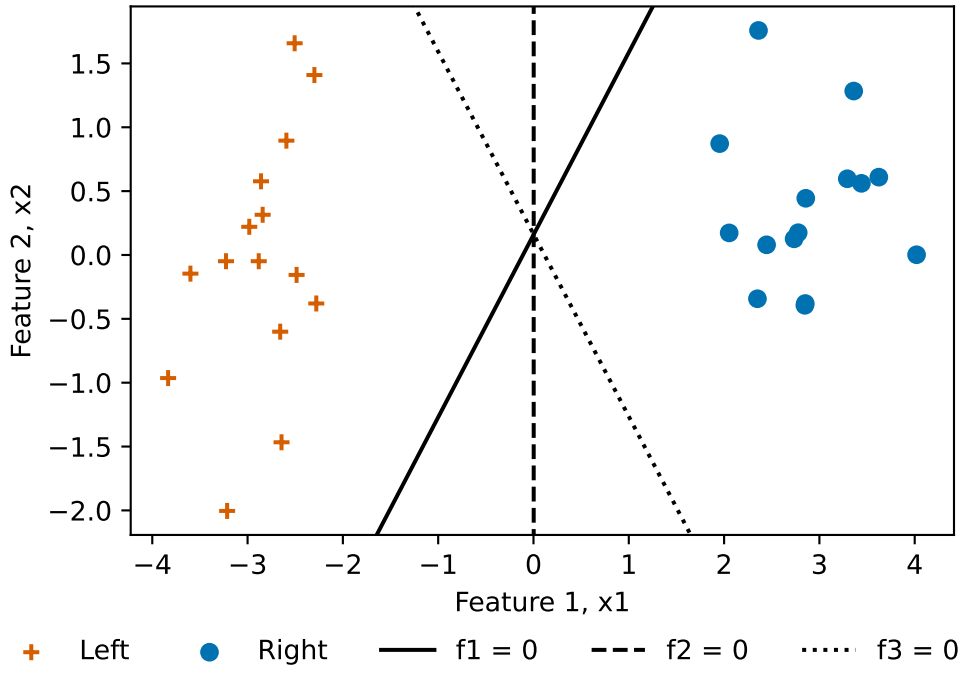
# Each boundary passes through the midpoint of the class means,
# at a different angle (in degrees) from the horizontal axis.
pivot = points.mean(axis=0)
angles = {"f1": 55, "f2": 90, "f3": 125}
styles = {"f1": "-", "f2": "--", "f3": ":"}


def side(angle):
    offset = points - pivot
    return offset[:, 0] * np.sin(angle) - offset[:, 1] * np.cos(angle) > 0


# Check that every boundary separates the two classes.
for degrees in angles.values():
    assert len(set(zip(side(np.radians(degrees)), is_right))) == 2

fig, ax = plt.subplots(figsize=(5, 3.6), layout="constrained")
ax.scatter(*left.T, marker="+", color="#D55E00", label="Left");
```

```
ax.scatter(*right.T, marker="o", color="#0072B2", label="Right");
x_lim, y_lim = ax.get_xlim(), ax.get_ylim()
for name, degrees in angles.items():
    direction = np.array([np.cos(np.radians(degrees)), np.sin(np.radians(degrees))])
    ends = pivot + np.outer([-10, 10], direction)
    ax.plot(*ends.T, styles[name], color="black", label=f"{name} = 0");
ax.set_xlim(x_lim);
ax.set_ylim(y_lim);
ax.set_xlabel("Feature 1, x1");
ax.set_ylabel("Feature 2, x2");
fig.legend(loc="outside lower center", ncol=5, frameon=False);
plt.show()
```



(b)

Figure 8: Three linear (strictly, affine) boundaries $f(\underset{\sim}{x}) = 0$, where $f(\underset{\sim}{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$, that all separate the same two classes.

Every boundary in Figure 8 classifies the simulated points without error, and the one farthest from the nearest point of either class[21] has the largest margin. Maximizing the logistic likelihood[22] does not give that boundary. The log loss of an observation[23], $\log(1 + e^{-m})$, decreases exponentially in its margin $m = \tilde{y}\vec{\beta}\cdot\underset{\sim}{x}$ and never reaches 0, so on separated data the likelihood keeps increasing as $\|\vec{\beta}\|$ grows along a separating direction, and there is no finite maximizer (Theorem 3.1). A ridge penalty gives a finite maximizer (Solution 3.14). As the penalty weight tends to 0, the direction of the penalized fit approaches that of the maximum-margin boundary [6]. Support vector machines maximize the margin of the separating hyperplane directly, and their solution is finite.

4 Separation in real data

The six patients of Exercise 3.1 were made up so the algebra could be done by hand. Separation also turns up in published studies, and fitting software reports it no more clearly there.

[21] https://morrison-lab.github.io/sds/classification.html#def-class
[22] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik
[23] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-observation

Heinze and Schemper analyzed endometrial cancer data on 79 patients as an example of separation [7]. Asseryanis, of the Medical University of Vienna, provided the data. The copy here comes from the `endometrial` table in the R package brglm2[24]. The data are in `data/endometrial.csv`[25], one patient per line:

- `HG`: the outcome, the histology grade of the tumor, `1` for high grade and `0` for low grade.
- `NV`: neovascularization, `1` if new blood vessels have formed in the tumor and `0` if not.
- `PI`: the pulsatility index of the uterine artery.
- `EH`: the height of the endometrium.

Dataset 2: The endometrial cancer data

Exercise 4.1 (Separation in the endometrial cancer data: what software reports)**.** Data from published studies can be quasi-completely separated too. In the endometrial cancer data, tabulating the histology grade `HG` against neovascularization `NV` shows that every patient with `NV` = 1 has a high-grade tumor:

```
# Read the local copy, then count the patients in each NV and HG cell,
# listing the empty cell with a count of 0 rather than dropping it.
endo <- readr::read_csv("data/endometrial.csv", show_col_types = FALSE)
empty_cell_count <- list(n = 0L)
endo |>
  dplyr::count(NV, HG) |>
  tidyr::complete(NV, HG, fill = empty_cell_count)
```

```
# A tibble: 4 x 3
     NV    HG     n
  <dbl> <dbl> <int>
1     0     0    49
2     0     1    17
3     1     0     0
4     1     1    13
```

Fit the logistic regression of `HG` on `NV` with R's `glm()`. What does it report?

Hint

Compare the report with the quasi-completely separated example of Example 3.2.

Solution

Solution 4.1. Fitting the endometrial cancer data with R's `glm()`:

```
# Fit HG on NV, then tabulate the coefficients with parameters.
endo_fit <- glm(HG ~ NV, family = binomial, data = endo)
endo_coefs <- summary(endo_fit)$coefficients
endo_b1_hat <- coef(endo_fit)[["NV"]]
endo_se_hat <- endo_coefs["NV", "Std. Error"]
endo_p_hat <- endo_coefs["NV", "Pr(>|z|)"]
parameters::parameters(endo_fit) |> parameters::print_md(digits = 3)
```

Parameter	Log-Odds	SE	95% CI	z	p
(Intercept)	-1.059	0.281	(-1.639, -0.528)	-3.761	< .001
NV	19.625	1809.054	(-104.915,)	0.011	0.991

R reports a slope[a] of 19.6 for NV, with a standard error[b] of 1809 and a p-value of 0.991. It gives no warning. An odds ratio[c] of $\exp\{19.6\}$ is not a plausible value for any predictor.
NV quasi-completely separates the data. The 13 patients with NV = 1 all have HG = 1, while the 66 patients with NV = 0 include both grades. So the maximum likelihood estimate[d] of the slope does not exist [4], [7], and the only sign of trouble in the output is the large standard error.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/sds/estimation.html#def-SE
[c] https://morrison-lab.github.io/sds/categorical-tests.html#def-odds-ratio
[d] https://morrison-lab.github.io/sds/intro-MLEs.html#def-mle

Exercise 4.2 (Separation in the endometrial cancer data: why the fit stops)**.** R's `glm()` reports a slope[a] for NV when it fits the logistic regression of the histology grade HG on NV in the endometrial cancer data (Solution 4.1).
Why does `glm()` stop at that slope?

Hint

Compare the stopped fit for the six patients of Exercise 3.1 in Example 3.4.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope

Solution

Solution 4.2. The log-likelihood[a] of the endometrial cancer data keeps increasing as the slope[b] of NV grows. Each increase moves the fitted probability[c] for the patients with NV = 1 closer to their observed 1. Far out, the log-likelihood changes very little. There the gradient and the change in fit are both small, so a fitting routine stops there, at a slope that its tolerance determines, without having found a maximum. `glm()` stops after 17 iterations, when the relative change in the deviance[d] falls below its tolerance, `epsilon = 1e-8` by default. The reported slope 19.62 depends on that tolerance; it is not an estimate[e] of a finite parameter. Running the same fit with a tighter tolerance:

```
# Tighten glm's convergence tolerance and allow more iterations.
endo_control <- glm.control(epsilon = 1e-14, maxit = 100)
endo_tight <- glm(
  HG ~ NV,
  family = binomial,
  data = endo,
  control = endo_control
)
c(slope = coef(endo_tight)[["NV"]], iterations = endo_tight$iter)
```

```
    slope iterations
 32.62489   30.00000
```

returns a larger slope, 32.62, after 30 iterations. Only this fit makes R warn, "fitted probabilities numerically 0 or 1 occurred". At the default tolerance the fit stopped before any fitted probability rounded to 1.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[c] https://morrison-lab.github.io/pds/probability-basics.html#def-probability
[d] https://morrison-lab.github.io/sds/logistic-inference.html#def-deviance
[e] https://morrison-lab.github.io/sds/estimation.html#def-estimate

Exercise 4.3 (Separation in the endometrial cancer data: the likelihood ratio test)**.** R's `glm()` gives the slope[a] of NV a Wald test[b] p-value of 0.991 in the logistic regression of the histology grade HG on NV in the endometrial cancer data (Solution 4.1).
Compute the likelihood ratio statistic[c] G^2 for H_0: the slope of NV is 0, and its p-value from the χ^2_1 distribution[d].

Hint

G^2 is the null deviance[e] minus the deviance[f] of the fit, as in Solution 3.11. In R, `anova()` with `test = "LRT"` compares two nested[g] fits.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test
[c] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat
[d] https://morrison-lab.github.io/pds/random-variables.html#def-cdf
[e] https://morrison-lab.github.io/sds/logistic-inference.html#def-null-deviance
[f] https://morrison-lab.github.io/sds/logistic-inference.html#def-deviance
[g] https://morrison-lab.github.io/sds/correlation-regression.html#def-nested-regression-models

Solution

Solution 4.3. Under H_0 the slope[a] of `NV` is 0, so the null fit has an intercept[b] alone. The unrestricted fit has no maximizer, so the log-likelihood[c] at the slope where `glm()` stopped takes the place of the maximized log-likelihood in the likelihood ratio statistic[d]. That log-likelihood is less than 10^{-6} below its supremum[e], because the log-likelihood changes very little that far out (Solution 4.2). Comparing the two fits of the endometrial cancer data:

```
# Fit the intercept-only model, then compare it with the fit on NV.
endo_null <- glm(HG ~ 1, family = binomial, data = endo)
endo_lrt <- anova(endo_null, endo_fit, test = "LRT")
endo_g2 <- endo_lrt$Deviance[[2]]
endo_g2_p <- endo_lrt[["Pr(>Chi)"]][[2]]
endo_lrt
```

```
# A tibble: 2 x 5
  `Resid. Df` `Resid. Dev`    Df Deviance    `Pr(>Chi)`
        <dbl>        <dbl> <dbl>    <dbl>         <dbl>
1          78        105.     NA     NA   NA
2          77         75.3     1     29.6  0.0000000532
```

G^2 is the null deviance[f] minus the deviance[g] of the fit:

$$\begin{aligned} G^2 &= 104.9 - 75.31 \\ &= 29.6. \end{aligned}$$

Against the χ^2_1 distribution[h], G^2 gives $p = 5.3 \times 10^{-8}$, while the Wald test[i] of the same hypothesis gave $p = 0.991$.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-intercept
[c] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[d] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat
[e] https://morrison-lab.github.io/mds/algebra-sums.html#def-supremum
[f] https://morrison-lab.github.io/sds/logistic-inference.html#def-null-deviance
[g] https://morrison-lab.github.io/sds/logistic-inference.html#def-deviance
[h] https://morrison-lab.github.io/pds/random-variables.html#def-cdf
[i] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test

Exercise 4.4 (Separation in the endometrial cancer data: which test to use)**.** For the slope[a] of `NV` in the logistic regression of the histology grade `HG` on `NV` in the endometrial cancer data, the Wald test[b] gives $p = 0.991$ (Solution 4.1) and the likelihood ratio test[c] gives $p = 5.3 \times 10^{-8}$ (Solution 4.3). Which of the two should you use here?

Hint

Look again at the Hauck–Donner effect.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test
[c] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat

Solution

Solution 4.4. Use the likelihood ratio test[a]: it finds a strong association between `NV` and the histology grade `HG`. The Wald statistic is the slope[b] divided by its standard error[c], and `glm()` stopped where both are large and the standard error has grown faster than the slope, so the Wald statistic is near 0. That decrease is the Hauck–Donner effect. The Wald test[d] uses the curvature of the log-likelihood[e] at the fitted slope, far out, where the log-likelihood changes very little. The likelihood ratio test uses the change in log-likelihood between a slope of 0 and the fit, which shows how much better the quasi-completely separated fit describes the endometrial cancer data.

[a] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[c] https://morrison-lab.github.io/sds/estimation.html#def-SE
[d] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test
[e] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik

Exercise 4.5 (Separation in the endometrial cancer data: how far to trust the p-value)**.** For the slope[a] of `NV` in the logistic regression of the histology grade `HG` on `NV` in the endometrial cancer data, the likelihood ratio test[b] p-value of Solution 4.3 relies on the χ^2_1 approximation. Only 13 patients have `NV` $= 1$.

Compute a p-value that needs no approximation, by conditioning on the number of high-grade tumors.

Hint

With one binary predictor, conditioning on the number of patients with `NV` $= 1$ as well as the number of high-grade tumors gives Fisher's exact test[c], `fisher.test()` in R. Compare the six patients' test in Solution 3.13.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lrt-stat
[c] https://morrison-lab.github.io/sds/categorical-tests.html#def-fishers-exact

Solution

Solution 4.5. The endometrial cancer data have 30 high-grade tumors among 79 patients. Under H_0, given that count and the 13 patients with `NV` $= 1$, each way to choose which patients have high-grade tumors is equally likely. The p-value is the probability[a] of a table at least as extreme as the observed one, which is what Fisher's exact test[b] computes:

```
# Condition on the row and column totals of the NV by HG table.
endo_exact <- fisher.test(endo_tab)
endo_exact$p.value
```

```
[1] 4.537619e-07
```

The two-sided p-value from `fisher.test()` is 4.5×10^{-7}. It is larger than the χ^2_1 value of Solution 4.3, 5.3×10^{-8}, but both reject H_0 at any usual level, while the Wald test[c] gave 0.991.

[a] https://morrison-lab.github.io/pds/probability-basics.html#def-probability
[b] https://morrison-lab.github.io/sds/categorical-tests.html#def-fishers-exact
[c] https://morrison-lab.github.io/sds/intro-MLEs.html#def-wald-test

Exercise 4.6 (Separation in the endometrial cancer data: the penalized slope)**.** In the logistic regression of the histology grade `HG` on `NV` in the endometrial cancer data, let β_0 be the intercept[a], β_1 the slope[b] of `NV`, and $\ell(\beta_0, \beta_1)$ the log-likelihood[c]. The ridge-penalized log-likelihood penalizes the slope alone:

$$f(\beta_0, \beta_1) = \ell(\beta_0, \beta_1) - \tfrac{\lambda}{2}\beta_1^2.$$

Find the slope $\hat{\beta}_1$ of the maximizer of f for $\lambda = 1$, 0.1 and 0.01.

Hint

Maximize f numerically, for example with R's `optim()` on $-f$. Compare the six patients'

penalized slope[d]s in Solution 3.15.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-intercept
[b] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[c] https://morrison-lab.github.io/sds/intro-MLEs.html#def-loglik
[d] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope

Solution

Solution 4.6. Maximizing f numerically for the endometrial cancer data, from a start at $\beta_0 = \beta_1 = 0$:

```
# Minus the ridge-penalized log-likelihood.
# The penalty leaves the intercept free.
endo_neg_pen <- function(b, lambda, data) { # nolint: function_location_linter
  eta <- b[[1]] + b[[2]] * data$NV
  p <- plogis(eta)
  loglik_i <- dbinom(data$HG, size = 1, prob = p, log = TRUE)
  loglik <- sum(loglik_i)
  lambda / 2 * b[[2]]^2 - loglik
}
# A tight tolerance, so the reported slopes do not depend on optim's default.
endo_optim_control <- list(reltol = 1e-14)
endo_ridge_slope <- function(lambda, data) { # nolint: function_location_linter
  start <- c(0, 0)
  fit <- optim(
    start, endo_neg_pen,
    lambda = lambda, data = data,
    method = "BFGS", control = endo_optim_control
  )
  fit$par[[2]]
}
endo_lambda <- c(1, 0.1, 0.01)
endo_ridge <- endo_lambda |> purrr::map_dbl(endo_ridge_slope, data = endo)
endo_ridge_cols <- c("$\\regpar$", "$\\hcoef{1}$")
tibble::tibble(lambda = endo_lambda, slope = endo_ridge) |>
  knitr::kable(
    col.names = endo_ridge_cols,
    digits = 3,
    align = "r"
  )
```

λ	$\widehat{\beta}_1$
1.00	2.376
0.10	4.380
0.01	6.368

The penalized slope[a] is 2.38 at $\lambda = 1$ and grows to 6.37 at $\lambda = 0.01$.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope

Exercise 4.7 (Separation in the endometrial cancer data: what the penalty costs)**.** The ridge-penalized fits of the endometrial cancer data in Solution 4.6 give a slope[a] for `NV` at each of $\lambda = 1$, 0.1 and 0.01. What does the penalty achieve, and at what cost?

Hint

Compare the six patients' answer in Solution 3.16.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope

Solution

Solution 4.7. For the endometrial cancer data, the ridge penalty gives f a maximizer at every $\lambda > 0$, so the slope[a] for `NV` is a number that the fit can report.
The cost is that the slope depends on λ. It is 2.7 times larger at $\lambda = 0.01$ than at $\lambda = 1$, and it keeps growing as $\lambda \to 0$, because the likelihood[b] alone keeps increasing as the slope grows. The penalty adds a preference for smaller coefficients, not information about `NV` from the data, and each penalized slope reflects that preference as much as the data. A report of a penalized fit should therefore state λ.
What the 79 patients show is not that $\hat{\beta}_1 = 4.38$. They show that `NV` quasi-completely separates the histology grades, that a slope of 0 is implausible ($p = 4.5 \times 10^{-7}$, Solution 4.5), and that the data alone cannot say how large the slope is.

[a] https://morrison-lab.github.io/mds/algebra-exponentials.html#def-slope
[b] https://morrison-lab.github.io/sds/intro-MLEs.html#def-lik

[1] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, *An introduction to statistical learning: With applications in Python*. Springer, 2023. doi: 10.1007/978-3-031-38747-0[26].

[2] A. Galanos, *rugarch: Univariate GARCH models*. 2026. doi: 10.32614/CRAN.package.rugarch[27].

[3] T. Hastie, R. Tibshirani, and J. Taylor, "Statistical learning with python." Stanford Online, lecture videos on YouTube, 2023. Available: https://www.youtube.com/playlist?list=PLoROMvodv4rPP6braWoRt5UCXYZ71GZIQ

[4] A. Albert and J. A. Anderson, "On the existence of maximum likelihood estimates in logistic regression models," *Biometrika*, vol. 71, no. 1, pp. 1–10, 1984, doi: 10.1093/biomet/71.1.1[28].

[5] Jr. Hauck Walter W. and A. Donner, "Wald's test as applied to hypotheses in logit analysis," *Journal of the American Statistical Association*, vol. 72, no. 360, pp. 851–853, 1977, doi: 10.1080/01621459.1977.10479969[29].

[6] S. Rosset, J. Zhu, and T. Hastie, "Boosting as a regularized path to a maximum margin classifier," *Journal of Machine Learning Research*, vol. 5, pp. 941–973, 2004, Available: https://www.jmlr.org/papers/volume5/rosset04a/rosset04a.pdf

[7] G. Heinze and M. Schemper, "A solution to the problem of separation in logistic regression," *Statistics in Medicine*, vol. 21, no. 16, pp. 2409–2419, 2002, doi: 10.1002/sim.1047[30].

[26] https://doi.org/10.1007/978-3-031-38747-0
[27] https://doi.org/10.32614/CRAN.package.rugarch
[28] https://doi.org/10.1093/biomet/71.1.1
[29] https://doi.org/10.1080/01621459.1977.10479969
[30] https://doi.org/10.1002/sim.1047