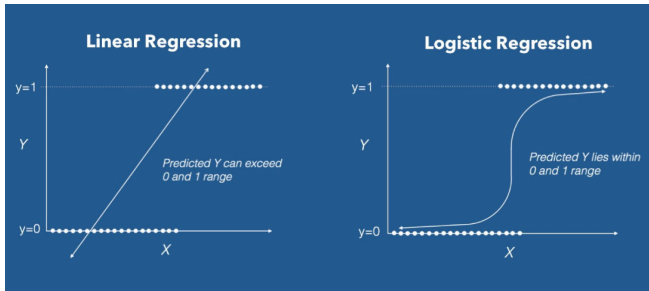


Regression

PSY-GS 8875 Behavioral Data Science



Overview: Week 2

Readings

- ESL Chapters: 3.1, 3.2, 4.4, and 4.4.1
- HML Chapters: 4.1-4.5 and 5.1-5.5
- [Yarkoni and Westfall - 2017](#)

Optional

- none

- Linear regression and **regression** metrics
- Logistic regression and **classification** metrics
- Gradient descent-style regression
- Activity: regression – new wine in old bottles

Linear Regression

Goal: predict some outcome Y using some features X

- known as a “supervised learning” problem
- **supervised:** outcome is known and used in “learning”
- **learning:** estimation of model parameters

Linear Regression

What relationship does *linear* regression learn between Y and X ?

Solves a *regression* problem

Linear Regression

What relationship does *linear* regression learn between Y and X ?

Solves a *regression* problem

But how does it “learn” to solve this problem?

Linear Regression

Formally:

$$y_i = \sum_{i=1}^n X_i \beta_i + \epsilon_i$$

or

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

ordinary least squares (OLS): estimate a line that minimizes the distance between \mathbf{y} and $\hat{\mathbf{y}}$

$$\hat{\beta} = \arg \min_{\hat{\beta}} \sum (\hat{\mathbf{y}} - \mathbf{y})^2$$

The OLS solution is a *closed-form* solution meaning can be solved analytically without any numerical optimization procedures such as Newton's method

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

Linear Regression

LINEAR REGRESSION

The thing we want to explain

DEPENDENT VARIABLE

y

i.e 77% of the variance in y is explained by x. Below c.30% means they're hardly connected. Above 95% and they're practically the same.

$$R^2 = 0.77$$

If you only had data on x, this line provides your best estimate of y. If the fit is strong and no major outliers, x could be used as a surrogate or forecast of y.

LINE OF BEST FIT

DATA POINT

95% CONFIDENCE BAND

If a data point falls outside these lines, you're 95% sure there is something special about it causing it to do better or worse than others - an 'outlier' worth understanding

OUTLIER

INDEPENDENT VARIABLE

x

The factor we think might influence the dependent variable

Common Usage

- $R^2 = \frac{\sum(\hat{y}_i - y)^2}{\sum(y_i - \bar{y})^2}$
- $p < 0.05$ using $\hat{se}(\hat{b}) = \sqrt{\frac{n\hat{\sigma}^2}{n \sum x_i^2 - (\sum x_i)^2}}$
- Assumptions?

Assumptions

- continuous outcome
- linear relationship between \mathbf{y} and $\hat{\mathbf{y}}$
- normality distributed errors (methods to check?)
- homoskedasticity or equal error variance (methods to check?)
- (lack of) multicollinearity (variance inflation factor or $VIF > 5$)

$$VIF_i = \frac{1}{1 - R_i^2},$$

where R_i^2 is each predictor treated as an outcome and predictor by all other predictors

R Example

Dataset

- Student math success in two Portegeuse schools [[source](#)]
- 395 students

```
# Load data  
math <- read.csv("../data/student_math/student_math_clean.csv")
```

Variables of Interest

Dependent Variable

- `final_grade`: math grade earned (0-20)

Independent Variables

- `study_time`: weekly study time (1 = < 2 hours, 2 = 2-5 hours, 3 = 5-10 hours, 4 = > 10 hours)
- `class_failures`: number of past class failures (1-3; otherwise, 4)
- `school_support`: extra educational support (yes/no)
- `family_support`: extra family support (yes/no)
- `extra_paid_classes`: extra (outside of school) classes that were paid for (yes/no)
- `higher_ed`: wants to go to higher education (yes/no)
- `internet_access`: whether they had internet access at home (yes/no)
- `absences`: number of days absent from school (0-93)

Linear Regression | R Example

```
# Select variables
math_voi <- math[,c(
  "final_grade", "study_time", "class_failures",
  "school_support", "family_support", "extra_paid_classes",
  "higher_ed", "internet_access", "absences"
)]

# Set study time
replace_values <- 1:4
names(replace_values) <- unique(math_voi$study_time)
math_voi$study_time <- replace_values[math_voi$study_time]

# Set "yes/no" responses to numeric
math_voi[,4:8] <- ifelse(math_voi[,4:8] == "yes", 1, 0)

# Separate out outcome and predictors
Y <- as.matrix(math_voi[, 1])
X <- as.matrix(math_voi[, -1])
X <- cbind(1, X) # add intercept

# Compute betas (transposed for print)
t(solve(t(X) %*% X) %*% t(X) %*% Y)
```

	study_time	class_failures	school_support	family_support
[1,]	8.386779	0.1957801	-2.036978	-1.071571
	extra_paid_classes	higher_ed	internet_access	absences
[1,]	0.3211003	1.945091	0.8606776	0.03396231

Linear Regression | R Example

```
# Linear model function
math_lm <- lm(final_grade ~ ., data = math_voi)

# Print summary
summary(math_lm)
```

Call:

```
lm(formula = final_grade ~ ., data = math_voi)
```

Residuals:

Min	1Q	Median	3Q	Max
-12.2968	-1.9497	0.2201	2.8957	8.1426

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	8.38678	1.24186	6.753	0.0000000000533 ***
study_time	0.19578	0.21229	0.922	0.3570
class_failures	-2.03698	0.30520	-6.674	0.0000000000865 ***
school_support	-1.07157	0.64273	-1.667	0.0963 .
family_support	-0.70063	0.46339	-1.512	0.1314
extra_paid_classes	0.32110	0.46439	0.691	0.4897
higher_ed	1.94509	1.03675	1.876	0.0614 .
internet_access	0.86068	0.58458	1.472	0.1418
absences	0.03396	0.02701	1.257	0.2094

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.244 on 386 degrees of freedom

Multiple R-squared: 0.1594, Adjusted R-squared: 0.142

F-statistic: 9.15 on 8 and 386 DF, p-value: 0.0000000001514

Model Evaluation

- Usual suspects: R^2 and statistical significance
- More common in data science to use (root) mean square error

$$RMSE = \sqrt{\frac{\sum(\hat{y}_i - y)^2}{n}}$$

```
# Compute RMSE
```

```
sqrt(mean((predict(math_lm) - math_voi$final_grade)^2))
```

```
[1] 4.195114
```

Is that good?

Linear Regression

RMSE from the [paper](#):

- neural networks: 4.41
- support vector machines: 4.37
- decision trees: 4.46
- random forest: 3.90

Seems like we did pretty good, right?

Linear Regression

RMSE from the [paper](#):

- neural networks: 4.41
- support vector machines: 4.37
- decision trees: 4.46
- random forest: 3.90

Seems like we did pretty good, right?

There's a catch. . .

Linear Regression

- they held out data and were computing RMSE on the data *held out* (more on that next week)
- we had the knowledge of their better predictors and used those only (there were a lot of irrelevant predictors in the dataset)

Logistic Regression

Logistic Regression

Goal: predict some outcome Y using some features X

- known as a “supervised learning” problem
- **supervised:** outcome is known and used in “learning”
- **learning:** estimation of model parameters

Logistic Regression

What relationship does *logistic* regression learn between Y and X ?

Logistic Regression

What relationship does *logistic* regression learn between Y and X ?

Solves a *classification* problem

- classification: categorizing observations into two (or more) classes

Logistic Regression

Formally:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\sum_{i=1}^n X_i \beta_i)}}$$

Conversely, it could be:

$$\sigma(z) = \frac{1}{1 + e^{-z}},$$

where

$$z = \sum_{i=1}^n X_i \beta_i$$

More on this notation when we cover neural networks. . .

Logistic Regression

Unlike linear regression, there isn't a closed-form solution for logistic regression

Instead, a numerical optimization method is used such as Newton's method:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

or iteratively reweighted least squares (IRLS):

$$\mathbf{w}_{k+1} = (\mathbf{X}^T \mathbf{S}_k \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{S}_k \mathbf{X} \mathbf{w}_k + \mathbf{y} - \mu_k),$$

where $\mathbf{S} = \text{diag}(\mu(i)(1 - \mu(i)))$ is a diagonal weighting matrix

Logistic Regression

The end goal is to seek a minimize of the difference between the predicted class (0 or 1) and the actual class

$$\hat{\beta} = \arg \min_{\hat{\beta}} \sum \log(1 + e^{(-y\mathbf{x}\beta)})$$

Assumptions?

Logistic Regression

Assumptions?

Fewer than linear regression but...

- (lack of) multicollinearity
- **balanced outcome** (i.e, $1s \approx 0s$)

R Example

Logistic Regression | R Example

Predicting whether a student had extra paid classes outside of school

```
# Logistic regression function (don't forget to set 'family')
math_log <- glm(extra_paid_classes ~ ., data = math_voi, family = "binomial")

# Print summary
summary(math_log)
```

Call:

```
glm(formula = extra_paid_classes ~ ., family = "binomial", data = math_voi)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	-3.646498	1.145916	-3.182	0.00146	**
final_grade	0.014419	0.026773	0.539	0.59019	
study_time	-0.138488	0.108597	-1.275	0.20222	
class_failures	-0.463546	0.191990	-2.414	0.01576	*
school_support	-0.354765	0.325253	-1.091	0.27539	
family_support	1.234305	0.234664	5.260	0.000000144	***
higher_ed	2.405163	1.064476	2.259	0.02385	*
internet_access	0.755183	0.317798	2.376	0.01749	*
absences	0.002583	0.014899	0.173	0.86236	

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 544.83 on 394 degrees of freedom
Residual deviance: 477.10 on 386 degrees of freedom
AIC: 495.1

Number of Fisher Scoring iterations: 5

Model Evaluation

- Usual suspects: pseudo- R^2 and statistical significance
- More common in data science to use confusion matrix metrics

Logistic Regression | Model Evaluation

		Predicted condition		Sources: [21][22][23][24][25][26][27][28][29] view · talk · edit	
Actual condition	Total population = P + N	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) = TPR + TNR - 1	Prevalence threshold (PT) = $\frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power = $\frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate = $\frac{FN}{P} = 1 - TPR$
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out = $\frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity = $\frac{TN}{N} = 1 - FPR$
	Prevalence = $\frac{P}{P + N}$	Positive predictive value (PPV), precision = $\frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) = $\frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) = $\frac{TPR}{FPR}$	Negative likelihood ratio (LR-) = $\frac{FNR}{TNR}$
	Accuracy (ACC) = $\frac{TP + TN}{P + N}$	False discovery rate (FDR) = $\frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) = $\frac{TN}{PN}$ = 1 - FOR	Markedness (MK), deltaP (Δp) = PPV + NPV - 1	Diagnostic odds ratio (DOR) = $\frac{LR+}{LR-}$
	Balanced accuracy (BA) = $\frac{TPR + TNR}{2}$	F ₁ score = $\frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) = $\sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) = $\sqrt{TPR \times TNR \times PPV \times NPV}$ - $\sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index = $\frac{TP}{TP + FN + FP}$

Wikipedia

Common Metrics

sensitivity: $\frac{TP}{TP+FP}$

specificity: $\frac{TN}{TN+FN}$

Accuracy: $\frac{TP+TN}{TP+FP+TN+FN}$

F1: $\frac{2TP}{2TP+FP+FN}$

Matthew's Correlation Coefficient (MCC or phi coefficient):

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Meet `{caret}`, your new best friend

Classification and Regression Training

Logistic Regression | Model Evaluation

```
# Load {caret}
library(caret)

# Get predicted probabilities from model
math_probabilities <- predict(
  math_log, type = "response"
  # don't forget to set `type = "response"`
)

# Convert to classes
math_classes <- factor( # ensure 'factor' mode!
  ifelse(math_probabilities > 0.50, "yes", "no")
)

# Ensure 'factor' mode for actual!
math_actual <- factor( # ensure 'factor' mode!
  ifelse(math_voi$extra_paid_classes == 1, "yes", "no")
)

# Get confusion matrix
confusionMatrix(
  data = math_classes, # predicted
  reference = math_actual, # actual
  positive = "yes" # class = 1
)
```


Logistic Regression | Model Evaluation

Confusion Matrix and Statistics

	Reference	
Prediction	no	yes
no	145	60
yes	69	121

Accuracy : 0.6734

95% CI : (0.6247, 0.7195)

No Information Rate : 0.5418

P-Value [Acc > NIR] : 0.00000006868

Kappa : 0.3448

Mcnemar's Test P-Value : 0.4812

Sensitivity : 0.6685

Specificity : 0.6776

Pos Pred Value : 0.6368

Neg Pred Value : 0.7073

Prevalence : 0.4582

Detection Rate : 0.3063

Detection Prevalence : 0.4810

Balanced Accuracy : 0.6730

'Positive' Class : yes

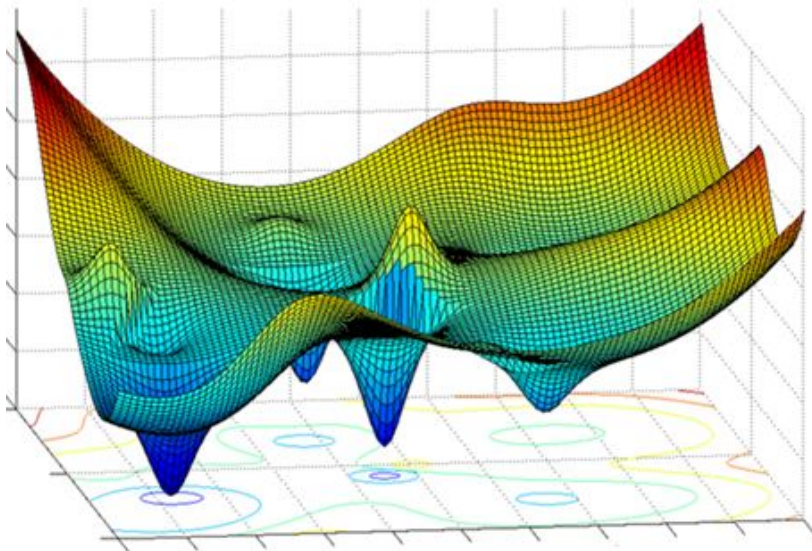
Kappa

- Can be used to account for class imbalances
- Ranges from -1 (complete discordance) to 1 (complete concordance)
 - ≤ 0 : no agreement
 - > 0.00 : slight agreement
 - > 0.20 : fair agreement
 - > 0.40 : moderate agreement
 - > 0.60 : substantial agreement
 - > 0.80 : almost perfect agreement

✓ kappa = 0.345

Gradient Descent

Gradient Descent



Gradient Descent

- Numeric optimization method like Newton's method (or IRLS)
- Relatively simple to implement
- Computationally efficient for complex problems but inefficient relative to closed-form solutions or solutions amenable to Newton's method

Gradient Descent

Formally:

$$p_{n+1} = p_n - \eta \nabla f(p_n),$$

where

- η = learning rate (magnitude of “step” in direction of descent)
- p_n = current values (e.g., β s in regression)
- $\nabla f(p_n)$ = gradient $\frac{\mathbf{X}^T(\hat{\mathbf{y}} - \mathbf{y})}{n}$

Gradient Descent for Linear Regression

$$p_{n+1} = p_n - \eta \nabla f(p_n),$$

$$p_n: B_0 = 0 \text{ and } B_1 = 0$$

$$\eta: 0.1$$

$$\nabla f(p_n): B_0 = -2.851 \text{ and } B_1 = -8.265$$

$$p_{n+1}: B_0 = 0.285 \text{ and } B_1 = 0.826$$

$$\text{OLS: } B_0 = 2.970 \text{ and } B_1 = 1.014$$

Did we get closer or further away?

Gradient Descent for Linear Regression

$$p_{n+1} = p_n - \eta \nabla f(p_n),$$

$$p_n: B_0 = 0.285 \text{ and } B_1 = 0.826$$

$$\eta: 0.1$$

$$\nabla f(p_n): B_0 = -2.663 \text{ and } B_1 = -1.274$$

$$p_{n+1}: B_0 = 0.551 \text{ and } B_1 = 0.954$$

$$\text{OLS: } B_0 = 2.970 \text{ and } B_1 = 1.014$$

Did we get closer or further away?

Gradient Descent for Linear Regression

$$p_{n+1} = p_n - \eta \nabla f(p_n),$$

$$p_n: B_0 = 0.551 \text{ and } B_1 = 0.954$$

$$\eta: 0.1$$

$$\nabla f(p_n): B_0 = -2.412 \text{ and } B_1 = -0.223$$

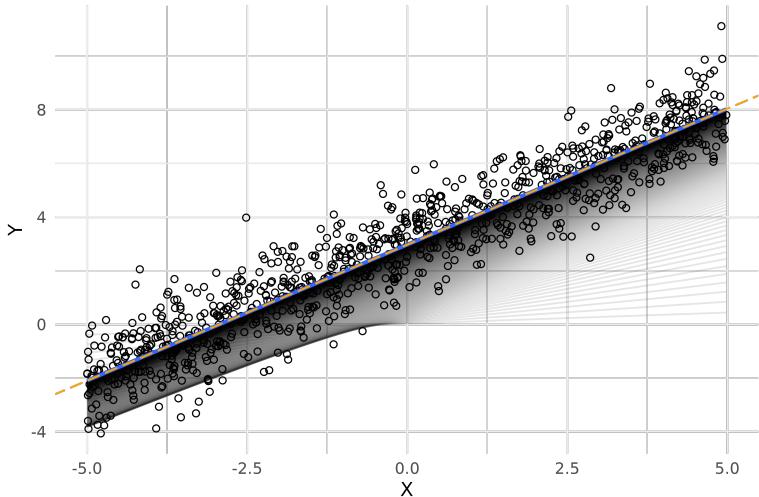
$$p_{n+1}: B_0 = 0.793 \text{ and } B_1 = 0.976$$

$$\text{OLS: } B_0 = 2.970 \text{ and } B_1 = 1.014$$

Did we get closer or further away?

Gradient Descent

Repeat for N iterations



How many iterations is enough?

- more complicated functions won't have closed-form solutions to check against (e.g., OLS)
- some criterion needs to be consulted for convergence
- change in criterion should stabilize (e.g., $\Delta_{\text{criterion}} < 0.001$ for n iterations)

Cost Functions

Mean square error (regression):

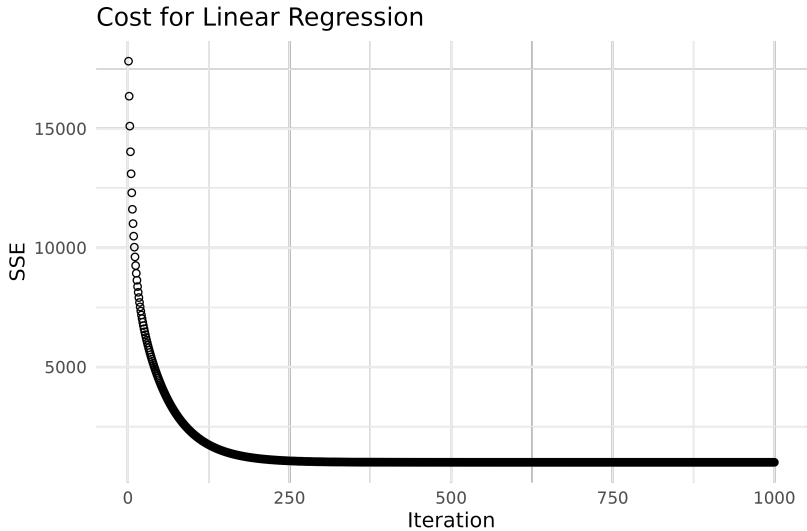
$$\frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{n}$$

(Binary) cross-entropy loss (classification):

$$\frac{\sum_{i=1}^n -(y_i \log(\hat{y}_i) + (1 - y_i)(1 - \log(\hat{y}_i)))}{n}$$

These functions are sought to be minimized

Gradient Descent



Gradient Descent

OLS: $B_0 = 2.970053$ and $B_1 = 1.013516$

GD: $B_0 = 2.969922$ and $B_1 = 1.013514$

Using 1000 iterations and $\eta = 0.01$

Iterations can be increased (21,401) and η (0.001) can be decreased to get closer and match OLS solution

In-class Activity

regression.R: new wine in old bottles

At Home Activity

Using the `student_math_clean.csv` data with the variables of interest used in these slides, get gradient descent linear and logistic regression parameters to match those presented in the slides

- Adjust `max_iter` and `learning_rate` as necessary
- Turn in final values of `max_iter` and `learning_rate` for the linear and logistic regression models to Brightspace

Readings for Next Week

Readings

- ESL Chapters: 7.10 and 7.11
- HML: Chapter 2
- [Yarkoni - 2022](#)

Optional

- [Shen et al. - 2017](#)