Data C100/C200, Midterm 2

Spring 2022

Name: _____

Email: _____@berkeley.edu

Student ID: _____

Exam Room: _____

Name and SID of left neighbor: _____

Name and SID of right neighbor: _____

Instructions:

This midterm exam consists of **55 points** spread out over **7 questions** and the Honor Code and must be completed in the **80 minute** time period ending at **8:30**, unless you have accommodations supported by a DSP letter.

Note that some questions have circular bubbles to select a choice. This means that you should only **select one choice**. Other questions have boxes. This means you should **select all that apply**. Please shade in the box/circle to mark your answer.

Q0 [1 Pt]: Honor Code

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I am the person whose name is on the exam and I completed this exam in accordance with the Honor Code.

Signature: _____

This page has been intentionally left blank.

1 Regular Ice, SQLite Sugar [6 Pts]

Marshall Mathers' Data 100 study group is holding a mid-semester social on Memorial Glade, and he is tasked to get bubble tea (i.e., boba). Since it's a Data 100 event, he surveys the students and staff for some information about the boba shops they've been to in Berkeley, and compiles the data into a table named `BobaReview`. Each row is a person's review of a boba shop; the first few rows are shown to the right.

reviewer	role	shop_name	rating	price
Kelly	staff	Boba_Ninja	2	5
Andrew	staff	Asha	4.5	5
Parth	student	Asha	5	5
Kunal	staff	One_Plus	4.5	4.5

(a) [2 Pts] Marshall wants to find all the reviews that rated a shop at least 4 (out of a maximum rating of 5). Help Marshall by writing a one-line SQL query that outputs all rows of `BobaReview` where a boba shop was rated **at least** 4 out of 5.

_____;

Solution: `SELECT * FROM BobaReview WHERE rating >= 4;`

(b) [4 Pts] Marshall now wants to find the **5 top-rated boba shops** in Berkeley. Help Marshall by completing the below SQL query, which selects the 5 boba shops from `BobaReview` with the highest average rating. Your query result should have three columns labeled `shop_name`, `avg_rating`, and `avg_price`, which are the shop name, average rating, and average price paid for boba, respectively. Rows should be sorted by the average rating, with the highest rated boba shop as the first row. The first few rows of the query result are shown below:

shop_name	avg_rating	avg_price
Asha	4.75	5.0
One_Plus	4.5	4.5

```
SELECT ____(i)____ FROM BobaReview ____(ii)____;
```

(i) What goes in the blank denoted by `(i)`?

`SELECT` _____

Solution:
```
shop_name,
AVG(rating) AS avg_rating,
AVG(price) AS avg_price
```

(ii) What goes in the blank denoted by `(ii)`? Use as few/many lines as you need.

Solution:
```
GROUP BY shop_name ORDER BY avg_rating DESC LIMIT 5;
```

2 Expectation and Variance [6 Pts]

Manak's is a store which sells pencils and miniature wax replicas of the head of legendary 1980s singer Kuldeep Manak. Suppose that the pencils are each \$2 and the wax heads are \$10. By carefully studying its customer records, Manak's has determined that the chance of a customer buying a pencil is 50 percent, and the chance of a customer buying a wax head is 10 percent. A summary of the probabilities and cost per item are given below.

Probability of Purchase	Price Per Item
$p_{\text{pencil}} = 0.50$	\$2
$p_{\text{head}} = 0.10$	\$10

Furthermore, the store observes that these purchases are independent, and limits purchases to at most one of each item at Manak's. So for example, the chance that a customer buys both a pencil and a wax head is $0.5 \times 0.1 = 0.05$, for which they spend \$12.

Suppose 10 customers enter the store. Define the random variable X as the **total purchase value** in dollars for the 10-person group. Assume all customers act independently of each other.

(a) [3 Pts] Compute $E[X]$, the expected total purchase value. Show your work.

Solution: Define random variables P and H as the number of pencils and wax heads, respectively, purchased by the 10-customer group. Then P is a Binomial random variable with $n = 10$ and $p = 0.5$; H is a Binomial random variable with $n = 10$ and $p = 0.1$.

Then, we know that $X = 2P + 10H$, corresponding to the product of the price and the number of items purchased. Then:

$$E[X] = 2E[P] + 10E[H] = 2(10 \cdot 0.5) + 10(10 \cdot 0.1) = 10 + 10 = 20.$$

(b) [3 Pts] Compute $SD(X)$, the standard deviation of the total purchase value. Show your work.

Solution: We use the definitions of P, H and X from the previous part. Recall that if we have independence, the variance of a sum is the sum of the variances. We also know that $\text{Var}(kP) = k^2\text{Var}(P)$, and that a Binomial random variable with parameters n and p has variance $np(1-p)$.

$$\begin{aligned}\text{Var}(X) &= \text{Var}(2P + 10H) \\ &= 4\text{Var}(P) + 100\text{Var}(H) \\ &= 4(10 \cdot 0.5 \cdot 0.5) + 100(10 \cdot 0.1 \cdot 0.9) \\ &= 10 + 90 \\ &= 100\end{aligned}$$

Then, $SD(X) = \sqrt{\text{Var}(X)} = 10$.

3 Welcome to the Linear Regression Multiverse [6 Pts]

Consider the multiple linear regression model, where $\mathbb{X}$ is an arbitrary, full column rank design matrix of size $n \times p$, and $\mathbb{Y}$ is the true response vector (i.e., outcome vector). Let $\hat{\mathbb{Y}} = \mathbb{X}\hat{\theta}$ be the predicted response vector, where the optimal parameter vector $\hat{\theta}$ is calculated using the least squares estimate.

(a) [1 Pt] What is the dimension of $\hat{\theta}$?

◯ A. $n \times 1$ ◯ **B.** $p \times 1$ ◯ C. $1 \times n$ ◯ D. $1 \times p$

Solution: If $\mathbb{X}$ is of size $n \times p$, then $\hat{\theta}$ must be $p \times 1$ by the rules of matrix multiplication.

(b) [2 Pts] Define the residual vector $e = \mathbb{Y} - \hat{\mathbb{Y}}$. Which of the following is true about the vector $v = \mathbb{X}^T e$? Select the best answer.

- ◯ **A. All of the entries of v are always zero.**
- ◯ B. All of the entries of v are zero only if $\mathbb{X}$ contains a bias column of all ones, $\mathbb{1}$.
- ◯ C. Some of the entries of v are zero.
- ◯ D. Some of the entries of v are zero only if $\mathbb{X}$ contains a bias column of all ones, $\mathbb{1}$.

Solution: This is from the geometric interpretation of least squares and the derivation of the normal equation. Recall that all vectors in the column space of $\mathbb{X}$ are orthogonal to the residuals e. Then, all vectors x_i in $\mathbb{X}$ are also orthogonal to e.

Expressed as a matrix multiply, this is $X^T e = 0$.

(c) [2 Pts] Which of the following is true about $(\mathbb{I} - \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T)\mathbb{Y}$? Define $\mathbb{I}$ as the identity matrix with appropriate dimensions. Select all that apply.

- ☐ A. It is in the span of $\mathbb{X}$.
- ☐ **B. It is orthogonal to the first column of $\mathbb{X}$.**
- ☐ C. It has dimension $n \times p$.
- ☐ **D. It is the residual vector $e = \mathbb{Y} - \hat{\mathbb{Y}}$.**

Solution: If we expand this quantity (as in discussion):

$$(\mathbb{I} - \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T)\mathbb{Y} = \mathbb{Y} - \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y} = \mathbb{Y} - X\hat{\theta}$$

. Recall that $e = \mathbb{Y} - X\hat{\theta}$. Based on the geometric properties of least squares, we know that all column vectors of $\mathbb{X}$ are orthogonal to the residual vector; this also means that it isn't within the span of $\mathbb{X}$.

(d) [1 Pt] Suppose we scale up every element in our design matrix $\mathbb{X}$ by a scalar a. What is the new least squares estimate if we use this new design matrix?

- ◯ A. $a^{np}\hat{\theta}$
- ◯ B. $\frac{1}{a^{np}}\hat{\theta}$
- ◯ C. $a\hat{\theta}$
- ◯ D. $\frac{1}{a\hat{\theta}}$
- ◯ **E.** $\frac{1}{a}\hat{\theta}$

Solution: Approach 1: If we let the new least squares estimate $\hat{\theta}_a = \frac{1}{a}\hat{\theta}$, then

$$\hat{\mathbb{Y}} = a\mathbb{X}\frac{1}{a}\hat{\theta} = \mathbb{X}\hat{\theta}.$$

Approach 2: Define the new design matrix as $\mathbb{X}_a = a\mathbb{X}$. The new least estimate is then

$$\begin{aligned}\hat{\theta}_a &= (\mathbb{X}_a^T\mathbb{X}_a)^{-1}\mathbb{X}_a^T\mathbb{Y} = ((a\mathbb{X})^T(a\mathbb{X}))^{-1}(a\mathbb{X})^T\mathbb{Y} \\ &= \left(\frac{1}{a^2}a\right)((\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}) = \frac{1}{a}\hat{\theta}.\end{aligned}$$

Recall that for an invertible matrix M and a scalar c, $(cM)^{-1} = \frac{1}{c}M$; for any matrix Q, $(cQ)^T = c(Q^T)$.

4 Hit or Miss [12 Pts]

Belcalis is using gradient descent for a constant model with $f_\theta(x) = \theta$. The empirical risk that she is trying to optimize is mean squared error:

$$R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f_\theta(x_i))^2$$

Belcalis uses gradient descent with an initialization of $\theta^{(0)}$ and a learning rate of α. Our dataset contains three data points (x_i, y_i) as shown below.

x	y
1	2
9	4
4	6

(a) [3 Pts] Derive the explicit gradient update for $\theta^{(t+1)}$ at time $t+1$ given $\theta^{(t)}$ at time t **in terms of only** y_i, assuming Belicalis chooses a **learning rate** $\alpha = 0.5$. **Make sure to fully simplify your answer** by multiplying out all expressions and collecting all terms.

Solution: We derive $\frac{dR}{d\theta}$ below. Note that the -2 term is from the chain rule.

$$\frac{dR}{d\theta} = \frac{1}{n}\sum_{i=1}^{n} -2(y_i - \theta)$$

Then, the gradient update is:

$$\begin{aligned}\theta^{(t+1)} &= \theta^{(t)} - \alpha\frac{1}{n}\sum_{i=1}^{n} -2(y_i - \theta^{(t)}) \\ &= \theta^{(t)} + \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^{(t)}) \\ &= \bar{y}\end{aligned}$$

Note: we should probably also include the solution worked out with numbers rather than symbols, since that's how I did it. In this version, the final answer is 4.

(b) [3 Pts] Suppose that instead of choosing $\alpha = 0.5$, Belicalis chooses a **learning rate of** $\alpha = 1$ with a random initialization of our parameter $\theta^{(0)}$. Unfortunately, with this configuration Belicalis discovers that rather than converging very quickly (as in part (a)), batch gradient descent never converges for this learning rate! Which of the following reasons describes why? Justify your answer on the following page.

Hint: What is the gradient update corresponding to $\alpha = 1$? Use the update to find $\theta^{(1)}$ and $\theta^{(2)}$ in terms of $\theta^{(0)}$.

- ◯ A. Gradient descent oscillates between growing negative and positive parameter values forever.
- ◯ **B. Gradient descent oscillates between two parameter values $\theta^{(0)}$ and $\theta^{(1)}$ forever.**
- ◯ C. Gradient descent returns an unbounded update since the gradient is infinity.
- ◯ D. Gradient descent returns an unbounded update since the gradient is always 0.

Justify your answer to part (b).

Solution: The gradient update corresponding to $\alpha = 1$ is:

$$\begin{aligned}\theta^{(t+1)} &= \theta^{(t)} + 2\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta^{(t)}) \\ &= \theta^{(t)} + 2\bar{y} - 2\theta^{(t)} \\ &= 2\bar{y} - \theta^{(t)}\end{aligned}$$

Thus, $\theta^{(1)} = 2\bar{y} - \theta^{(0)}$.

Applying the next gradient step, we notice that we return to the initialization: $\theta^{(2)} = 2\bar{y} - \theta^{(1)} = 2\bar{y} - (2\bar{y} - \theta^{(0)}) = \theta^{(0)}$!

Similar story continues for the next step where we return to $\theta^{(1)}$: $\theta^{(3)} = 2\bar{y} - \theta^{(2)} = 2\bar{y} - \theta^{(0)} = \theta^{(1)}$.

By recursive properties, we oscillate between $\theta^{(0)}$ and $\theta^{(1)}$ forever.

(c) [2 Pts] Suppose that Belcalis runs gradient descent with the same objective $R(\theta)$ and dataset above, with a chosen initialization $\theta^{(0)} = 1$ and an unknown fixed learning rate $\alpha > 0$. Which of the following may occur given that batch gradient descent converges? Select all that apply.

- ☐ A. Batch gradient descent converges to a local minimum that is not a global minimum.
- ☐ **B. Batch gradient descent converges to a global minimum.**
- ☐ C. Batch gradient descent converges to a local maximum that is not a global maximum.
- ☐ D. Batch gradient descent converges to a global maximum.

Solution: There are no local minima since this function (MSE) is convex, so there is a unique global minimum. Depending on α (which could be appropriate or not), we could either converge or not converge to that global minimum, so option 2 is correct.

There exist no local minima and no local or global maxima for this function since it is convex (its "maximum" is unboundedly large).

(d) [2 Pts] Belcalis decides to implement stochastic gradient descent (SGD), where she chooses a **batch size of 1** and samples the dataset randomly without replacement. If she uses learning rate $\alpha = 0.5$, which of the following are **possible** values of the parameter $\theta^{(t)}$ at some arbitrary time step $t > 0$ of applying SGD? Select all that apply.

☐ A. $\theta^{(t)} = 0$

☐ **B.** $\theta^{(t)} = 2$

☐ C. $\theta^{(t)} = 3$

☐ **D.** $\theta^{(t)} = 4$

☐ E. $\theta^{(t)} = 5$

☐ **F.** $\theta^{(t)} = 6$

Solution: Using SGD with batch size 1 and $\alpha = 0.5$, each update considers only one y_i. Recall that the update for $\alpha = 0.5$ is

$$\theta^{(t+1)} = \bar{y} = \frac{1}{|B|} \sum_{i=1}^{|B|} y_i$$

Since we only have one y_i, $\bar{y} = y_i$. Then, the only possible θ values are values in our dataset - 2, 4, and 6!

(e) [2 Pts] Suppose that Belcalis changes the empirical risk function to mean absolute error (MAE):

$$R_{\text{new}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} |y_i - f_\theta(x_i))|$$

Which of the following are true if Belcalis now minimizes this new empirical risk on the constant model using gradient descent with learning rate $\alpha = 0.01$? Select all that apply.

☐ A. It is impossible to use gradient descent with this empirical risk since it is non-differentiable.

☐ B. Batch gradient descent will always find an optimal θ that minimizes this empirical risk.

☐ **C. Batch gradient descent may not find an optimal θ that minimizes this empirical risk.**

☐ **D. The derivative of this empirical risk with respect to θ is undefined when $y_i = f_\theta(x_i)$ for any i.**

Solution: Recall from discussion that we can use gradient descent even when functions are non-differentiable at certain points. This is differentiable at all points except for when $f_\theta(x_i) = y_i$ for some i. However, there is no guarantee that we pick an appropriate learning rate, so batch/stochastic gradient descent may or may not converge.

The derivative is undefined at the minimizing value when $y_i = f_\theta(x_i)$ if the absolute value

function is graphed out (i.e. the ”kinks” in the function are when an absolute value of 0 are taken for each of the n losses).

5 Exponential Regularization [10 Pts]

Namjoon is experimenting with regularization functions, and he wants to explore an exponential regularization function $\text{Pow}(\theta)$:

$$\text{Pow}(\theta) = \lambda \sum_{i=1}^{p} 10^{\theta_i}$$

(a) [2 Pts] Suppose Namjoon fits a multiple linear regression model $\hat{\mathbb{Y}} = \mathbb{X}\theta$ with L_2 loss and the exponential regularization function specified above, so that the objective function being minimized is $\frac{1}{n}||\mathbb{Y} - \mathbb{X}\theta||_2^2 + \lambda \sum_{i=1}^{p} 10^{\theta_i}$. Which of the following are true? Select all that apply.

- ☐ A. The optimal solution is the least squares estimate, $(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$.
- ☐ B. The optimal solution is $(\mathbb{X}^{-1} + \lambda\mathbb{I})\mathbb{X}^T\mathbb{Y}$.
- ☐ **C. Gradient descent can be used to solve for the optimal solution with exponential regularization.**
- ☐ **D. λ represents the regularization coefficient.**

Solution: The optimal solution for OLS without regularization will not hold. A simple example is to take $\lambda = \infty$, where we would expect the optimal $\hat{\theta}$ to change under the effect of regularization.

Option B violates rules of inversion since $\mathbb{X}$ need not be square (in fact, in almost all cases, it isn't). Even if it is square, we can easily use a counterexample of a constant model with one data point where $y_1 = 0$ to show that this isn't optimal.

Gradient descent can be used since $R(\theta)$ is differentiable, and like with all regularization functions, λ represents the regularization coefficient.

(b) [2 Pts] For the model described in part (a), which of the following is true about the entries of the optimal $\hat{\theta}$ vector as λ approaches positive infinity (i.e. $\lambda \to \infty$)?

- ◯ **A. $\hat{\theta} \to -\infty$**
- ◯ B. $\hat{\theta} \to 0$
- ◯ C. $\hat{\theta} \to \infty$

Justify your answer (2–3 sentences maximum):

Solution: There are a few ways of approaching this question. Some are described below. Note that the solution is extremely long to provide multiple view points and approaches, and you were not expected to go through these during the exams.

As a prelude to answering the question, it may be worth noting that for any choice of $\hat{\theta}$, whether it's bounded or unbounded, the objective function value at that point will be unbounded (unlike for ridge regression). For $\hat{\theta} = 0$, the exponential regularization will make the objective unbounded and for the other values, the loss function will make the objective unbounded. However, the optimal $\hat{\theta}$ does approach one of these values even if the objective function becomes unbounded. This observation is not required to answer the question, but it is provided for overall understanding.

Intuitive Approach

Similarly to ridge regression, as λ grows, the loss function value starts "mattering" less and less to the minimization of the objective function. In other words, the dominant term is the regularization term, and the optimizer (whether it is gradient descent or some other algorithm) will try its best to focus on minimizing that term.

The same holds in the case of exponential regularization, and in fact, it is even more pronounced. Exponential functions blow up very quickly - and it is even more so a priority for the optimizer as λ grows large, to keep the exponentiated θ values as small as possible. In other words, we ideally want each $10^{\theta_i} \to 0$, which means that $\hat{\theta} \to -\infty$.

By minimizing this more dominant term as λ grows big, we obtain the optimal $\hat{\theta}$ vector that results.

Rigorous Approach (somewhat)

Some may not be convinced that the optimal $\hat{\theta}$ vector approaches $-\infty$ since all the options listed have an unboundedly large objective value. What makes one of them more correct than the other?

To answer this more concretely, consider a simple example of a constant model fit on one datapoint ($y_1 = 0$) using exponential regularization. The objective is:

$$\arg\min_{\theta} \theta^2 + \lambda 10^{\theta}$$

Consider $\lambda = 0$, where there is no regularization. In this case, the clear optimal answer is $\hat{\theta} = 0$ as expected.

Consider $\lambda = 10^{100}$. While this function can't be minimized through calculus very easily, we will use intuition to approximate an optimum. For $\theta = 0$, we incur an enormous regularization penalty of 10^{100}. For values of θ between -95 and 0, we incur loss values on the order of magnitude of $(90)^2 \approx 8 \cdot 10^3$ at worst, but our regularization penalty is between $10^{100} \cdot 10^{-95} = 10^5$ and 10^{100}. Clearly, none of these minimize the dominant term once λ becomes large.

In this case, consider that $\hat{\theta}$ would be near -100, which makes sense since the regularization penalty is bounded by fairly small values given $\theta \approx 100$ (around 1). We could apply similar analyses to conclude that for any large $\lambda = 10^k$ for $k > 0$, the optimal $\hat{\theta} \approx -k$. Note that this is just an illustrative example for a simple constant model!

Here are the optimal $\hat{\theta}$ values for various values of λ (obtained from Desmos). You should notice that for the larger λ values, the optimal value matches our approximation quite well!

- $\lambda = 1, \hat{\theta} = -0.429$
- $\lambda = 10, \hat{\theta} = -1.043$
- $\lambda = 10^{10}, \hat{\theta} = -9.102$
- $\lambda = 10^{100}, \hat{\theta} = -98.070$
- $\lambda = 10^{300}, \hat{\theta} = -297.588$

(c) [2 Pts] Which of the following are true if we compare our exponential regularization with L_1 and L_2 regularization if we use the same λ for all three? Select all that apply.

☐ **A. Exponential regularization always penalizes large positive coefficients more than L_2 regularization.**

☐ B. Exponential regularization always penalizes large negative coefficients more than L_2 regularization.

☐ **C. L_2 regularization is more effective than exponential regularization when we want parameters close to zero.**

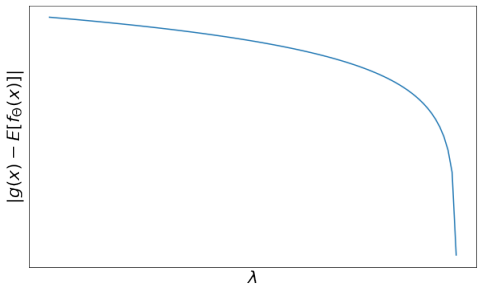
☐ **D. L_1 regularization often performs feature selection by setting parameters corresponding to non-contributing features to 0, but exponential regularization cannot be used for feature selection.**

Solution: Exponential regularization as shown in the equation $\text{Pow}(\theta)$ will penalize large positive coefficients since 10^k for $k > 0$ grows quickly, but it will not penalize negative coefficients since 10^{-k} can be bounded by 1 (i.e. it is always less than 1). As a result, exponential regularization encourages parameters to be negative, and ridge (as studied) encourages parameters to be zero, so the third option is correct. The final option is also correct since as shown in the previous part, the parameters do not even tend to 0 as λ grows; in general, exponential regularization cannot be used for feature selection.

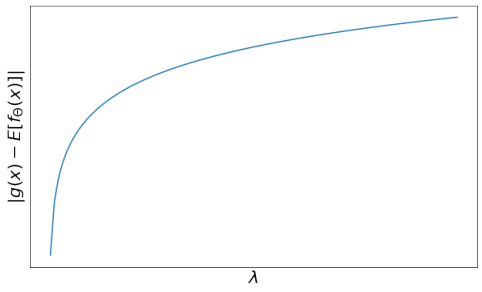
(d) [2 Pts] Which of the following could plausibly be a plot of the *magnitude* (i.e., absolute value) of model bias as a function of λ if we use multiple linear regression with exponential regularization? Assume that values on the x-axis are linear from 0 to asymptotically large λ.

Circle the letter corresponding to the most plausible plot.

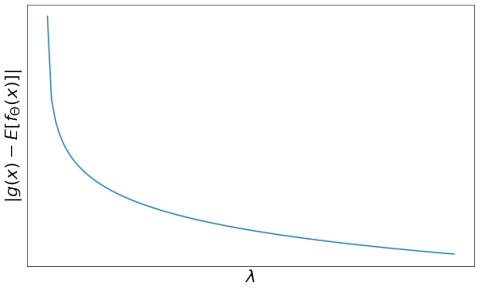
A.



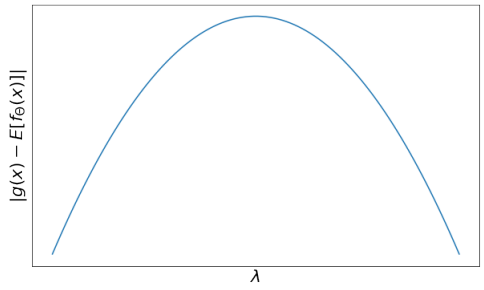
B.



C.



D.



Solution: The solution is B. The bias must grow as the λ increases since our estimates of the ground truth function $g(x)$ will become poorer.

Consider the scenario from part (b), where as λ grows, the optimal parameters shrink towards $-\infty$. With this, we can conclude that our estimates of $g(x)$ become more inaccurate as λ grows.

(e) [2 Pts] To choose the most optimal λ, Namjoon uses 5-fold cross-validation to train 5 models and stores the appropriate root-mean square error for each fold and λ. Given the collected statistics below about the **cross-validation error for each model**, help Namjoon choose the values of λ (among the 5 listed) that is the **best** to use with exponential regularization.

	λ = 0.001000	λ = 0.010000	λ = 0.100000	λ = 1.000000	λ = 10.000000
count	5.000000	5.000000	5.000000	5.000000	5.0
mean	105.017370	62.983541	126.653039	2873.545309	inf
std	89.465170	57.638969	46.808695	1070.162327	NaN
min	8.027953	7.550870	56.968206	1380.204219	inf
25%	60.837863	18.134202	102.327504	2381.351981	NaN
50%	71.527759	39.551492	142.738819	2807.642621	NaN
75%	146.070646	123.592860	162.326315	3891.280147	NaN
max	238.622629	126.088281	168.904348	3907.247576	inf

◯ A. $\lambda = 0.001$

◯ **B.** $\lambda = 0.01$

◯ C. $\lambda = 0.1$

◯ D. $\lambda = 1$

◯ E. $\lambda = 10$

Solution: We wish to use the hyperparameter associated with the smallest mean cross-validated error. In this case, that is $\lambda = 0.01$.

6 Food Features [13 Pts]

Crossroads, the Cal Dining Hall, wants to discover more about what is in a Cal student's diet. They collect data on what 10,000 Cal students ate for lunch over 100 consecutive days, along with the total number of calories of each lunch.

The first few rows of the dataset are shown below, where each row represents the daily lunch diet of an individual; since we record 100 lunches for each person, there are 1,000,000 = 10^6 rows. You may assume that there are only 4 foods: rice, pizza, orange juice, and sushi.

	name	day	foods	servings	total_calories
0	Morgan	6	Rice, Pizza, Orange Juice	2, 1, 5	3000
1	Ashley	3	Rice	4	1000
2	Kevin	4	Orange Juice	2	250
3	Aditya	2	Pizza	1	800
4	Ashley	10	Sushi, Pizza	5, 1	1200

(a) [2 Pts] Suppose Crossroads uses the following boolean featurization for encoding an individual's lunch, where they create "dummy" features corresponding to all foods. They set these dummy features of **all** foods for an individual's lunch on a particular day to 1 and not eaten to 0.

A featurized sample of the dataset is shown to the right (corresponding to the same first few rows above).

		Rice	Pizza	Orange Juice	Sushi
name	**day**				
Morgan	**6**	1	1	1	0
Ashley	**3**	1	0	0	0
Kevin	**4**	0	0	1	0
Aditya	**2**	0	1	0	0
Ashley	**10**	0	1	0	1

Note that the name and day are not included as features. The index (name, day) is simply present to identify the rows.

We construct a design matrix $\mathbb{X}$ of shape $(10^6, 4)$ that contains **only** the featurization from the above DataFrame to predict the total calories in an individual's lunch using ordinary least squares. Which of the following is true?

☐ A. The optimal coefficients $\hat{\theta}$ must be zero.

☐ **B. The design matrix $\mathbb{X}$ must be full column rank.**

☐ C. The sum of each feature vector represents the number of servings eaten of each food.

☐ **D. The sum of each row vector represents the number of unique foods eaten by a person for a particular lunch.**

Solution: The optimal coefficients need not be zero. To see this, consider a simple example where the calories for all the meals were all zero except the third row. Then, the coefficient for orange juice must be positive.

The design matrix must be full column rank even based on this sample because regardless of the remaining rows, these vectors are linearly independent by observation.

The sum of each row represents the total number of unique foods eaten by a person and the sum of each column represents the total number of unique (people, day) combinations that ate each food.

(b) [2 Pts] Jennie modifies the featurization from the previous part slightly; instead of setting all dummy features of all foods eaten by an individual on that day to 1 and not eaten to 0, we set the dummy features of all foods eaten to the **number of servings** they ate.

A featurized sample of the dataset is shown to the right.

		Rice	Pizza	Orange Juice	Sushi
name	**day**				
Morgan	**6**	2	1	5	0
Ashley	**3**	4	0	0	0
Kevin	**4**	0	0	2	0
Aditya	**2**	0	1	0	0
Ashley	**10**	0	1	0	5

We construct a design matrix $\mathbb{X}$ of shape 10^6 by 4 that contains **only** the featurization from the above DataFrame to predict the total calories in an individual's lunch using ordinary least squares. What does the optimal coefficient $\hat{\theta}_j$ represent?

- ◯ A. Typical calories consumed through food j per day
- ◯ **B. Typical calories consumed through food j per serving size**
- ◯ C. Typical calories consumed by person j per day
- ◯ D. Typical calories consumed by person j per serving size

Solution: Options C and D do not work since there are exactly 4 entries in our θ vector, each corresponding to a food. otherwise, the matrix multiplication between our design matrix and the parameter vector cannot work.

Out of the remaining options, we are trying to estimate the total number of calories based on the number of servings of each food. Hence, we express our response variable as a linear

combination of our features:

$$\text{meal calories} = \text{food 1 servings} \cdot \frac{\text{food 1 calories}}{\text{food 1 servings}} + ... + \text{food 4 servings} \cdot \frac{\text{food 4 calories}}{\text{food 4 servings}}$$

Note that the number of servings for each food are features, whereas the ratio of calories to servings per each food are our learned parameters to be able to generate the total number of calories.

(c) [6 Pts] For this part, suppose we have a DataFrame `lunch_mat` as shown below:

	person	calories_1d_ago	calories_2d_ago	calories
0	Morgan	1200	850	500
1	Ashley	900	1300	900

Jennie wants to predict the number of calories eaten for lunch today (`calories`) using the number of calories for the two previous days' lunches. The linear model now has 3 parameters $\theta = (\theta_0, \theta_1, \theta_2)$, respectively corresponding to an **intercept term** and calories from 1 and 2 days ago, respectively. To see whether she can interpret her models' parameters, Jennie decides to construct bootstrapped 95% confidence interval for the new model parameters.

Below we show the code Jennie writes to bootstrap the least squares estimate 5,000 times and create the confidence interval for θ_1. Some notes:

- Note the helper function `get_param1` takes in a design matrix `X` and response vector `y` and returns the least squares estimate for $\hat{\theta}_1$ using `sklearn`.
- `df.sample(n, replace)` draws a random sample (with/without replacement, specified by boolean `replace`) of `n` rows from a Pandas dataframe `df`.
- `np.percentile(a, q)` computes the `q`-th percentile of the array/list `a`.

Complete Jennie's code on the next page.

Fill in the blanks for part (c).

```
from sklearn.linear_model import LinearRegression
def get_param1(X, y):
    model = LinearRegression(fit_intercept = __________)
    model.fit(X, y)
    theta1_estimate = model.coef_[____]
    return theta1_estimate

estimates = []
num_rows, num_cols = lunch_mat.shape
for i in range(5000):
    boot_sample = lunch_mat.sample(____________, ____________)
    boot_X = boot_sample[_____________________________________________]
    boot_y = boot_sample[_____________________________________________]
    boot_param1 = get_param1(boot_X, boot_y)
    estimates.append(boot_param1)

lower = np.percentile(estimates, ____________)
upper = np.percentile(estimates, ____________)
confidence_interval = (lower, upper)
```

Solution:

```
from sklearn.linear_model import LinearRegression

def get_param1(X, y):
    model = LinearRegression(fit_intercept=True)
    model.fit(X, y)
    param1_estimate = model.coef_[0]
    return param1_estimate

estimates = []
num_rows, num_cols = lunch_mat.shape
for i in range(5000):
    bootstrap_sample = lunch_mat.sample(num_rows, True)
    boot_X = bootstrap_sample[["calories_1d_ago", "calories_2d_ago"]]
    boot_Y = bootstrap_sample["calories"]
    boot_param1 = get_param1(boot_X, boot_Y)
```

```
    estimates.append(boot_param1)


lower = np.percentile(estimates, 2.5)
upper = np.percentile(estimates, 97.5)
confidence_interval = (lower, upper)
```

(d) [3 Pts] Jennie extends the code you wrote in part (c) to generate bootstrapped 95% confidence intervals for *all* parameters:

$$\theta_0 : [44.808, 45.013], \qquad \theta_1 : [-0.026, 0.342], \qquad \theta_2 : [-0.142, 1.324]$$

What conclusions can we draw from **these confidence intervals**? Select all that apply.

☐ A. Given Morgan's lunch calories for the last two days, we cannot predict Morgan's lunch calories today because the true thetas might be zero.

☐ **B. The features of the design matrix might be collinear (i.e., multicollinear).**

☐ **C. A person's lunch calories yesterday may have no true relationship with lunch calories today.**

☐ D. Given Morgan's lunch calories for the last 2 days, we can make a reasonable prediction for Ashley's lunch calories today.

Solution: A is false: Even if the confidence intervals include zero, our model is still useful for prediction, though not necessarily inference.

B is true: One possible way that we can get confidence intervals that include zero is if the features are multicollinear.

C is true: One possible way that we can get confidence intervals that include zero is if the true relationship is null.

D is false: Our model doesn't even try to predict person X's calories today from person Y's calories in the last two days.

7 Following Instructions [1 Pt]

If you are taking your exam in-person, you can earn this point by writing your Student ID in the top right corner of each page.

If you are taking your exam online, you can earn this point by doing both of the following:

1. Write the answer to each question on a different page.
2. Assign the pages to the correct question subpart when submitting on Gradescope.

Reminder: Complete the Honor Code question (Q0) on the front of the exam for another point.

[Optional] Do you like cats or dogs? Draw your favorite of the two pets in the box below to cast your vote!

Congratulations on finishing the exam!

This page has been intentionally left blank.

Spring 2022 Data 100/200 Midterm 1 Reference Sheet

Pandas

Suppose `df` is a DataFrame; `s` is a Series. `pd` is the Pandas package.

Function	Description
`df[col]`	Returns the column labeled `col` from `df` as a Series.
`df[[col1, col2]]`	Returns a DataFrame containing the columns labeled `col1` and `col2`.
`s.loc[rows] / df.loc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their index values.
`s.iloc[rows] / df.iloc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their positions.
`s.isnull() / df.isnull()`	Returns boolean Series/DataFrame identifying missing values
`s.fillna(value) / df.fillna(value)`	Returns a Series/DataFrame where missing values are replaced by `value`
`df.drop(labels, axis)`	Returns a DataFrame without the rows or columns named `labels` along `axis` (either 0 or 1)
`df.rename(index=None, columns=None)`	Returns a DataFrame with renamed columns from a dictionary `index` and/or `columns`
`df.sort_values(by, ascending=True)`	Returns a DataFrame where rows are sorted by the values in columns `by`
`s.sort_values(ascending=True)`	Returns a sorted Series.
`s.unique()`	Returns a NumPy array of the unique values
`s.value_counts()`	Returns the number of times each unique value appears in a Series
`pd.merge(left, right, how='inner', on='a')`	Returns a DataFrame joining DataFrames `left` and `right` on the column labeled `a`; the join is of type `inner`
`left.merge(right, left_on=col1, right_on=col2)`	Returns a DataFrame joining DataFrames `left` and `right` on columns labeled `col1` and `col2`.
`df.pivot_table(index, columns, values=None, aggfunc='mean')`	Returns a DataFrame pivot table where columns are unique values from `columns` (column name or list), and rows are unique values from `index` (column name or list); cells are collected `values` using `aggfunc`. If `values` is not provided, cells are collected for each remaining column with multi-level column indexing.
`df.set_index(col)`	Returns a DataFrame that uses the values in the column labeled `col` as the row index.
`df.reset_index()`	Returns a DataFrame that has row index 0, 1, etc., and adds the current index as a column.

Let `grouped = df.groupby(by)` where `by` can be a column label or a list of labels.

Function	Description
`grouped.count()`	Return a Series containing the size of each group, excluding missing values
`grouped.size()`	Return a Series containing size of each group, including missing values
`grouped.mean()`/`grouped.min()`/`grouped.max()`	Return a Series/DataFrame containing mean/min/max of each group for each column, excluding missing values
`grouped.filter(f)` `grouped.agg(f)`	Filters or aggregates using the given function f

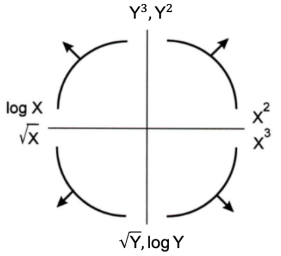
Function	Description
`s.str.len()`	Returns a Series containing length of each string
`s.str.lower()`/`s.str.upper()`	Returns a Series containing lowercase/uppercase version of each string
`s.str.replace(pat, repl)`	Returns a Series after replacing occurences of substrings matching regular expression `pat` with string `repl`
`s.str.contains(pat)`	Returns a boolean Series indicating whether a substring matching the regular expression `pat` is contained in each string
`s.str.extract(pat)`	Returns a Series of the first subsequence of each string that matches the regular expression `pat`. If `pat` contains one group, then only the substring matching the group is extracted

Visualization

Matplotlib: `x` and `y` are sequences of values.

Function	Description
`plt.plot(x, y)`	Creates a line plot of `x` against `y`
`plt.scatter(x, y)`	Creates a scatter plot of `x` against `y`
`plt.hist(x, bins=None)`	Creates a histogram of `x`; `bins` can be an integer or a sequence
`plt.bar(x, height)`	Creates a bar plot of categories `x` and corresponding heights `height`

Tukey-Mosteller Bulge Diagram.



Seaborn: `x` and `y` are column names in a DataFrame `data`.

Function	Description
`sns.countplot(data, x)`	Create a barplot of value counts of variable `x` from `data`
`sns.histplot(data, x, kde=False)` `sns.displot(x, data, rug = True, kde = True)`	Creates a histogram of `x` from `data`; optionally overlay a kernel density estimator. `displot` is similar but can optionally overlay a rug plot.
`sns.boxplot(data, x=None, y)` `sns.violinplot(data, x=None, y)`	Create a boxplot of y, optionally factoring by categorical `x`, from `data`. `violinplot` is similar but also draws a kernel density estimator of `y`.
`sns.scatterplot(data, x, y)`	Create a scatterplot of `x` versus `y` from `data`
`sns.lmplot(x, y, data, fit_reg=True)`	Create a scatterplot of `x` versus `y` from `data`, and by default overlay a least-squares regression line
`sns.jointplot(x, y, data, kind)`	Combine a bivariate scatterplot of `x` versus `y` from `data`, with univariate density plots of each variable overlaid on the axes; `kind` determines the visualization type for the distribution plot, can be `scatter`, `kde` or `hist`

Regular Expressions

List of all metacharacters: `. ^ $ * + ? ] [ \ | ( ) { }`

Operator	Description	Operator	Description
`.`	Matches any character except **\n**	`*`	Matches preceding character/group zero or more times
`\\`	Escapes metacharacters	`?`	Matches preceding character/group zero or one times
`|`	Matches expression on either side of expression; has lowest priority of any operator	`+`	Matches preceding character/group one or more times
`\d`, `\w`, `\s`	Predefined character group of digits (0–9), alphanumerics (a–z, A–Z, 0–9, and underscore), or whitespace, respectively	`^`, `$`	Matches the beginning and end of the line, respectively
`\D`, `\W`, `\S`	Inverse sets of `\d`, `\w`, `\s`, respectively	`( )`	Capturing group used to create a sub-expression
`{m}`	Matches preceding character/group exactly `m` times	`[ ]`	Character class used to match any of the specified characters or range (e.g. `[abcde]` is equivalent to `[a-e]`)
`{m, n}`	Matches preceding character/group at least `m` times and at most `n` times if either `m` or `n` are omitted, set lower/upper bounds to 0 and ∞, respectively	`[^ ]`	Invert character class; e.g. `[^a-c]` matches all characters except `a`, `b`, `c`

Function	Description
`re.match(pattern, string)`	Returns a match if zero or more characters at beginning of `string` matches `pattern`, else None
`re.search(pattern, string)`	Returns a match if zero or more characters anywhere in `string` matches `pattern`, else None
`re.findall(pattern, string)`	Returns a list of all non-overlapping matches of `pattern` in `string` (if none, returns empty list)
`re.sub(pattern, repl, string)`	Returns `string` after replacing all occurrences of `pattern` with `repl`

Modified lecture example for a single capturing group:

```
lines = '169.237.46.168 - - [26/Jan/2014:10:47:58 -0800] "GET ... HTTP/1.1"'
re.findall(r'\[\d+\/(\w+)\/\d+:\d+:\d+:\d+ .+\]', line) # returns ['Jan']
```

Modeling

Concept	Formula	Concept	Formula
L_1 loss	$L_1(y, \hat{y}) = \mid y - \hat{y} \mid$	Correlation r	$r = \frac{1}{n}\sum_{i=1}^{n} \frac{x_i - \bar{x}}{\sigma_x}\frac{y_i - \bar{y}}{\sigma_y}$
L_2 loss	$L_2(y, \hat{y}) = (y - \hat{y})^2$	Linear regression prediction of y	$\hat{y} = a + bx$
Empirical risk with loss L	$R(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i)$	Least squares linear regression, slope $\hat{b}$	$\hat{b} = r\frac{\sigma_y}{\sigma_x}$
		Least squares linear regression, intercept $\hat{a}$	$\hat{a} = \bar{y} - \hat{b}\bar{x}$

Spring 2022 Data 100/200 Midterm 2 Reference Sheet

Ordinary Least Squares

Multiple Linear Regression Model: $\hat{\mathbb{Y}} = \mathbb{X}\theta$ with design matrix $\mathbb{X}$, response vector $\mathbb{Y}$, and predicted vector $\hat{\mathbb{Y}}$. If there are p features plus a bias/intercept, then the vector of parameters $\theta = [\theta_0, \theta_1, \ldots, \theta_p]^T \in \mathbb{R}^{p+1}$. The vector of estimates $\hat{\theta}$ is obtained from fitting the model to the sample $(\mathbb{X}, \mathbb{Y})$.

Concept	Formula	Concept	Formula
Mean squared error	$R(\theta) = \frac{1}{n}\|\|Y - X\theta\|\|_2^2$	Normal equation	$\mathbb{X}^T\mathbb{X}\hat{\theta} = \mathbb{X}^T\mathbb{Y}$
Least squares estimate, if $\mathbb{X}$ is full rank	$\hat{\theta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$	Residual vector, e	$e = \mathbb{Y} - \hat{\mathbb{Y}}$
		Multiple R^2 (coefficient of determination)	$R^2 = \frac{\text{variance of fitted values}}{\text{variance of } y}$
Ridge Regression L2 Regularization	$\frac{1}{n}\|\|Y - X\theta\|\|_2^2 + \lambda\|\|\theta\|\|_2^2$	Squared L2 Norm of $\theta \in \mathbb{R}^d$	$\|\|\theta\|\|_2^2 = \sum_{j=1}^d \theta_j^2$
Ridge regression estimate (closed form)	$\hat{\theta}_{\text{ridge}} = (\mathbb{X}^T\mathbb{X} + n\lambda I)^{-1}\mathbb{X}^T\mathbb{Y}$		
LASSO Regression L1 Regularization	$\frac{1}{n}\|\|Y - X\theta\|\|_2^2 + \lambda\|\|\theta\|\|_1$	L1 Norm of $\theta \in \mathbb{R}^d$	$\|\|\theta\|\|_1 = \sum_{j=1}^d \|\theta_j\|$

Scikit-Learn

Suppose `sklearn.model_selection` and `sklearn.linear_model` are both imported packages.

Package	Function(s)	Description
`sklearn.linear_model`	`LinearRegression()`	Returns an ordinary least squares Linear Regression model.
	`LassoCV()`, `RidgeCV()`	Returns a Lasso (L1 Regularization) or Ridge (L2 regularization) linear model, respectively, and picks the best model by cross validation.
	`model.fit(X, y)`	Fits the scikit-learn `model` to the provided `X` and `y`.
	`model.predict(X)`	Returns predictions for the X passed in according to the fitted `model`.
`sklearn.model_selection`	`train_test_split(*arrays, test_size=0.2)`	Returns two random subsets of each array passed in, with 0.8 of the array in the first subset and 0.2 in the second subset.

Probability

Let X have a discrete probability distribution $P(X = x)$. X has expectation $\mathbb{E}[X] = \sum_x xP(X = x)$ over all possible values x, variance $\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$, and standard deviation $\text{SD}(X) = \sqrt{\text{Var}(X)}$.

The covariance of two random variables X and Y is $\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$. If X and Y are independent, then $\text{Cov}(X, Y) = 0$.

Notes	Property of Expectation	Property of Variance
X is a random variable. $a, b \in \mathbb{R}$ are scalars.	$\mathbb{E}[aX + b] = a\mathbb{E}[X] + b$	$\text{Var}(aX + b) = a^2\text{Var}$
X, Y are random variables.	$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$	$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y)$
X is a Bernoulli random variable that takes on value 1 with probability p and 0 otherwise.	$\mathbb{E}[X] = p$	$\text{Var}(X) = p(1 - p)$
Y is a Binomial random variable representing the number of ones in n independent Bernoulli trials with probability p of 1.	$E[Y] = np$	$\text{Var}(Y) = np(1 - p)$

Central Limit Theorem

Let $(X_1, \ldots, X_n)$ be a sample of independent and identically distributed random variables drawn from a population with mean μ and standard deviation σ. The sample mean $\overline{X}_n = \sum_{i=1}^{n} X_i$ is normally distributed, where $\mathbb{E}[\overline{X}_n] = \mu$ and $\text{SD}(\overline{X}_n) = \sigma/\sqrt{n}$.

Parameter Estimation

Suppose for each individual with fixed input x, we observe a random response $Y = g(x) + \epsilon$, where g is the true relationship and ϵ is random noise with zero mean and variance σ^2.

For a new individual with fixed input x, define our random prediction $\hat{Y}(x)$ based on a model fit to our observed sample $(\mathbb{X}, \mathbb{Y})$. The model risk is the mean squared prediction error between Y and $\hat{Y}(x)$:

$$\mathbb{E}[(Y - \hat{Y}(x))^2] = \sigma^2 + \Big(\mathbb{E}[\hat{Y}(x)] - g(x)\Big)^2 + \text{Var}(\hat{Y}(x)).$$

Suppose that input x has p features and the true relationship g is linear with parameter $\theta \in \mathbb{R}^{p+1}$.
Then $Y = f_\theta(x) = \theta_0 + \sum_{j=1}^{p} \theta_j x_j + \epsilon$ and $\hat{Y} = f_{\hat{\theta}}(x)$ for a parameter estimate $\hat{\theta}$ fit to the observed sample $(\mathbb{X}, \mathbb{Y})$.

Gradient Descent

Let $L(\theta, \mathbb{X}, \mathbb{Y})$ be an objective function to minimize over θ, with some optimal $\hat{\theta}$. Suppose $\theta^{(0)}$ is some starting estimate at $t = 0$, and $\theta^{(t)}$ is the estimate at step t. Then for a learning rate α, the gradient update step to compute $\theta^{(t+1)}$ is

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_\theta L(\theta^{(t)}, \mathbb{X}, \mathbb{Y}),$$

where $\nabla_\theta L(\theta^{(t)}, \mathbb{X}, \mathbb{Y})$ is the partial derivative/gradient of L with respect to θ, evaluated at $\theta^{(t)}$.

SQL

SQLite syntax:

```
SELECT [DISTINCT]
    {* | expr [[AS] c_alias]
    {,expr [[AS] c_alias] ...}}
FROM tableref {, tableref}
[[INNER | LEFT ] JOIN table_name
    ON qualification_list]
[WHERE search_condition]
[GROUP BY colname {,colname...}]
[HAVING search_condition]
[ORDER BY column_list]
[LIMIT number]
[OFFSET number of rows];
```

Syntax	Description
`SELECT column_expression_list`	List is comma-separated. Column expressions may include aggregation functions (`MAX`, `FIRST`, `COUNT`, etc). `AS` renames columns. `DISTINCT` selects only unique rows.
`FROM s INNER JOIN t ON cond`	Inner join tables `s` and `t` using `cond` to filter rows; the `INNER` keyword is optional.
`FROM s LEFT JOIN t ON cond`	Left outer join of tables `s` and `t` using `cond` to filter rows.
`FROM s, t`	Cross join of tables `s` and `t`: all pairs of a row from `s` and a row from `t`
`WHERE a IN cons_list`	Select rows for which the value in column `a` is among the values in a `cons_list`.
`ORDER BY RANDOM LIMIT n`	Draw a simple random sample of `n` rows.
`ORDER BY a, b DESC`	Order by column `a` (ascending by default) , then `b` (descending).
`CASE WHEN pred THEN cons ELSE alt END`	Evaluates to `cons` if `pred` is true and `alt` otherwise. Multiple `WHEN`/`THEN` pairs can be included, and `ELSE` is optional.
`WHERE s.a LIKE 'p'`	Matches each entry in the column `a` of table `s` to the text pattern `p`. The wildcard `%` matches at least zero characters.
`LIMIT number`	Keep only the first `number` rows in the return result.
`OFFSET number`	Skip the first `number` rows in the return result.