Introduction to Machine Learning (Fall 2022)

Note: Due date extended to Sept. 21, 2022 at midnight

Related to earlier catsoop outages, the Homework 2 due date (usually 11pm) is extended to Sept. 21, 2022 at midnight.

Recitation attendance check

Type in your section passcode to get attendance credit (within the first fifteen minutes of your scheduled section).
Passcode: Enter

100.00%

Section 0 is for those without a section assigned yet from the registrar. 1-7 correspond to the list on [homepage](COURSE).

{0: 'holidays', 1: 'april_fools', 2: 'bon_festival', 3: 'chun_jie', 4: 'diwali', 5: 'eid_al_fitr', 6: 'flag_day', 7: 'groundhog_day', 8: 'holi', 9: 'isra'}

Homework 2 -- Regression

Due: Wednesday, September 21, 2022 at 11:59 PM

Please study the lecture notes on Regression before you address these questions.

For debugging the coding parts of this assignment, we suggest using this HW02 Colab Notebook.

1) A taste of regularization

You are given the following data, where $d = 1$, $n = 4$.

$$D = \{[[1], 2], [[2], 7], [[3], -3], [[4], 1]\}$$

You want to use analytic linear regression to solve the problem.

1.1)

What is the X matrix? Remember to include an extra input dimension that always has the value 1.
Provide a list of lists that would be an argument to `np.array()`, where each data point is a column.
X = [[1,2,3,4], [1,1,1,1]]

Check Formatting Submit View Answer **100.00%**

You have infinitely many submissions remaining.

1.2)

What is the Y vector? Provide a list of lists that would be an argument to `np.array()`

Y = [[2,7,-3, 1]]

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

1.3)

Using Python and numpy (you might want to fire up a co-lab session), compute the θ, θ_0 that minimize MSE on this data. Provide your answer as a list.

As a reminder, you can use np.linalg.inv to take inverses in numpy.

$\theta, \theta_0 =$

[-1.3, 5]

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

1.4)

What is the MSE of the hypothesis you found on the data (any answer within the right order of magnitude will be fine)?

10.6

Check Formatting | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

1.5)

Now, if we change the data to have a small amount of noise so that

$$D = \{[[1, \epsilon, \epsilon], 2], [[2, \epsilon, \epsilon], 7], [[3, \epsilon, \epsilon], -3], [[4, \epsilon, \epsilon], 1]\}$$

where the ϵ values are all different, randomly chosen independently in the range $(-1 \times 10^{-5}, +1 \times 10^{-5})$, we get the following parameters:

$$\theta = \begin{bmatrix} 4.70697 \times 10^{0} \\ -1.28107 \times 10^{6} \\ 1.10581 \times 10^{7} \end{bmatrix}, \theta_0 = -1.03437 \times 10^{2}$$

This hypothesis has the following MSE on the training data: 1.6970×10^{-24}.

Consider a "testing" set, which is *very* similar to the training set:

$$D = \{[[1, 0, 0], 2], [[2, 0, 0], 7], [[3, 0, 0], -3], [[4, 0, 0], 1]\}.$$

What's the MSE of `th, th0` on the testing set?

8556.48

Check Formatting | Submit | View Answer 100.00%

You have infinitely many submissions remaining.

1.6)

Using the same data as in the previous question (with two added noise dimensions) but using ridge regression with $\lambda = 1 \times 10^{-10}$.

We get the following parameters:

$$\theta = \begin{bmatrix} -1.377711 \times 10^{0} \\ -2.574581 \times 10^{5} \\ -5.070563 \times 10^{4} \end{bmatrix}, \theta_0 = 7.260269 \times 10^{0}$$

Does this hypothesis have higher or lower training set error than the result without ridge regression?

- (•) Higher
- ( ) Lower

Submit | View Answer 100.00%

You have infinitely many submissions remaining.

Does this hypothesis have higher or lower testing set error than the result without ridge regression?

- ( ) Higher
- (•) Lower

Submit | View Answer 100.00%

You have infinitely many submissions remaining.

2) Linear Regression

We are interested in performing ordinary least squares regression given data X, Y to find parameters θ, θ_0 that minimize the mean squared error objective:

$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} L_s(x^{(i)}, y^{(i)}; \theta, \theta_0) \ ,$$

where the squared loss is

$$L_s(x^{(i)}, y^{(i)}; \theta, \theta_0) = (\hat{y}^{(i)} - y^{(i)})^2 = (\theta^T x^{(i)} + \theta_0 - y^{(i)})^2 \ .$$

Note that the $L_s(x^{(i)}, y^{(i)}; \theta, \theta_0)$ notation here is used to emphasize that the loss depends on both the data sample $(x^{(i)}, y^{(i)})$, and

the parameters, θ and θ_0. Compared to the $\mathcal{L}_s$ notation as used in some part of the notes and the lab, we see that $L_s(x^{(i)}, y^{(i)}; \theta, \theta_0) = \mathcal{L}_s(h(x^{(i)}; \theta, \theta_0), y^{(i)})$, where $h(x^{(i)}; \theta, \theta_0) = \theta^T x^{(i)} + \theta_0$.

2.1)

If X is d by n and Y is 1 by n, what is the dimension of θ? d by 1

Submit View Answer 100.00%

You have infinitely many submissions remaining.

2.2)

If X is d by n and Y is 1 by n, what is the dimension of $\nabla_\theta J(\theta, \theta_0)$? d by 1

Submit View Answer 100.00%

You have infinitely many submissions remaining.

2.3)

What is the gradient of the objective with respect to θ? We can see that it is of the form:

$$\nabla_\theta J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L_s(x^{(i)}, y^{(i)}; \theta, \theta_0)$$

where $x^{(i)}, y^{(i)}$ are the i^{th} data point and its label. Note that $x^{(i)}$ is a column vector.

Write an expression for $\nabla_\theta L_s(x^{(i)}, y^{(i)}; \theta, \theta_0)$ using the symbols: `x_i`, `y_i`, `theta` and `theta_0`, where $\nabla_\theta L_s(x^{(i)}, y^{(i)}; \theta, \theta_0)$ is the gradient of the L_s function described above with respect to θ. Remember that you can use `@` for matrix product, and you can use `transpose(v)` to transpose a vector. Note that this $\nabla_\theta L_s(\cdot)$ function is *just* the gradient with respect to a single data point $x^{(i)}, y^{(i)}$. We'll build up to the gradient of J in a moment. If you get stuck, refer to the notes.

$\nabla_\theta L_s(x^{(i)}, y^{(i)}; \theta, \theta_0) =$ 2*(transpose(theta)@x_i + theta_0 - y_i)*x_i

Check Syntax Submit View Answer 100.00%

You have infinitely many submissions remaining.

2.4)

What is the gradient of the objective J **now with respect to** θ_0? We can see that it is of a similar form:

$$\nabla_{\theta_0} J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\theta_0} L_s(x^{(i)}, y^{(i)}; \theta, \theta_0) \ .$$

Write an expression for $\nabla_{\theta_0} L_s(x^{(i)}, y^{(i)}; \theta, \theta_0)$ using the symbols: `x_i`, `y_i`, `theta` and `theta_0`.

$\nabla_{\theta_0} L_s(x^{(i)}, y^{(i)}; \theta, \theta_0) =$ 2*(transpose(theta)@x_i + theta_0 - y_i)

Check Syntax | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

2.5)

Next we're interested in the gradient of the objective with respect to θ, written $\nabla_\theta J$, but now for a whole data set X (of dimensions d by n).

Write an expression for $\nabla_\theta J$ using the symbols: `X` and `Y` for the full set of data, `theta`, `theta_0`, and `n`. Here `X` has dimensions d by n, and `Y` has dimensions 1 by n. `theta_0` is a scalar that needs broadcasting, similar to the problem in Homework 1. Remember that you can use `@` for matrix product, and you can use `transpose(a)` to transpose a vector or array.

The Appendix on matrix derivatives in the course notes might help.

$\nabla_\theta J =$ 2/n*X@(transpose(X)@theta + theta_0- transpose(Y))

Check Syntax | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

We will now look at how to find an analytical solution to the linear regression optimization problem.

As described in the notes, we will work with the transpose of our original data matrices. Let data matrix $\tilde{X} = X^T$ be n by d, let target output vector $\tilde{Y} = Y^T$ be n by 1, and recall that θ is d by 1. Then we can write the whole linear regression prediction as $\tilde{X}\theta$.

2.6)

$\tilde{Y}$ is the n by 1 vector of target output values. Write an equation expressing the mean squared loss of θ in terms of $\tilde{X}$, $\tilde{Y}$, n, and θ.

Enter your answer as a Python expression. You can use symbols `X_tilde`, `Y_tilde`, `n` and `theta`. Recall that our expression syntax includes `transpose(x)` for transpose of an array, `inverse(x)` for the inverse of an array, and `x@y` to indicate a matrix product of two arrays.

$J(\theta) =$ 1/n*transpose(X_tilde@theta - Y_tilde)@(X_tilde@theta - Y_tilde)

Check Syntax | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

2.7)

Now, how can we find the minimizing θ, given $\tilde{X}$ and $\tilde{Y}$? Take the gradient (yes, even with a matrix expression), set it to zero(s) and solve for θ.

What if you set the gradient of $J(\theta)$ to 0 and solve for θ^*, the optimal θ?

$\theta^* =$

- $(\tilde{X}^T\tilde{Y})^{-1}(\tilde{X}^T\tilde{X})$
- (•) $(\tilde{X}^T\tilde{X})^{-1}\tilde{X}^T\tilde{Y}$
- $(\tilde{X}\tilde{X}^T)^{-1}\tilde{X}\tilde{Y}^T$

Submit View Answer 100.00%

You have infinitely many submissions remaining.

2.8)

Just converting back to the data matrix format we have been using (not transposed), we have

$\theta^* =$

- $(XY^T)^{-1}(XX^T)$
- $(X^TX)^{-1}X^TY$
- (•) $(XX^T)^{-1}XY^T$

Submit View Answer 100.00%

You have infinitely many submissions remaining.

2.9)

Now implement θ^* as found in the previous problem, using symbols `X` and `Y` for the data matrix and outputs, and `np.dot`(or `@` shorthand), `np.transpose` (or `.T` shorthand), `np.linalg.inv`.

```
# Enter an expression to compute and set th to the optimal theta
th = np.linalg.inv(X@X.T)@X@Y.T
```

Run Code Submit View Answer 100.00%

You have infinitely many submissions remaining.

3) Sources of Error

Recall that *structural* error arises when the hypothesis class cannot represent a hypothesis that performs well on the test data and *estimation* error arises when the parameters of a hypothesis cannot be estimated well based on the training data. (You can also refer to the notes on Regression.)

Following is a collection of potential cures for a situation in which your learning algorithm generates a hypothesis with a high test error.

For each one, indicate whether it **can reduce** structural error, estimation error, both, or neither.

3.1)

Penalize $||\theta||^2$ during training.

Can reduce:

- structural error
- estimation error
- both
- neither

Submit View Answer 100.00%

You have infinitely many submissions remaining.

3.2)

Penalize $||\theta||^2$ during testing.

Can reduce:

- structural error
- estimation error
- both
- neither

Submit View Answer 100.00%

You have infinitely many submissions remaining.

3.3)

Increase the amount of training data.

Can reduce:

- structural error
- estimation error
- both
- neither

Submit View Answer 100.00%

You have infinitely many submissions remaining.

3.4)

Increase the complexity/power of the hypothesis class.

Can reduce:

- structural error
- estimation error
- both
- neither

Submit View Answer 100.00%

You have infinitely many submissions remaining.

4) Buy 'n' Large

We have been hired by Buy 'n' Large to deliver a predictor of change in sales volume from last year, for each of their stores. We have a machine-learning algorithm that can be used with regularization parameter λ. Our overall objective is to deliver a predictor that minimizes squared loss on predictions when actually used by the company. There are three relevant data sets: training dataset D_{train}, validation dataset D_{val} and test dataset D_{test}, each of size n. You have D_{train} and D_{val}; D_{test} is owned by the company and not shared with you..

We will use a linear predictor with parameters θ (without the offset parameter θ_0 for simplicity) and use regularizer $\lambda||\theta||^2$. There are several phases of the training process (as represented in the subproblems below), and we need to select the appropriate objective for each of those tasks. Each question provides the template of an answer, and asks you to fill in choices for different "slots" in the template that specify

- A - what the minimization is over,
- B - the dataset used,
- C - the predictor used, and
- D - whether regularization is added.

Fill in the slots (A,B,C,D) for each phase by choosing the expressions for the indicated slots.

Note that $\theta_{best}(\lambda)$ is a value of θ that is a function of λ; θ^*, λ^* are specific values found as described below; θ and λ are variables that range over d-dimensional column vectors and positive reals, respectively.

4.1) Training

Selecting the best hypothesis (parameters θ) for some fixed value of the regularization parameter λ. Call this $\theta_{best}(\lambda)$.

$$\theta_{best}(\lambda) = \arg\min_{A} \frac{1}{|B|} \sum_{(x,y)\in B} (C^T x - y)^2/2 + D$$

4.1.1)

What should we minimize over? This is denoted by the variable A.

- θ
- $\theta_{best}(\lambda)$
- θ^*

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

4.1.2)

What dataset should we use? This is denoted by the variable B.

- D_{train}
- D_{val}
- $D_{train} \cup D_{val}$
- D_{test}

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

4.1.3)

What predictor should we use? This is denoted by the variable C.

- θ
- $\theta_{best}(\lambda)$
- θ^*

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

4.1.4)

What regularizer should we use? This is denoted by the variable D.

- 0
- λ
- λ^*
- $\lambda\|\theta\|^2$
- $\lambda^*\|\theta\|^2$

Submit View Answer 100.00%

You have infinitely many submissions remaining.

4.2) Hyperparameter selection

Selecting the best value of the regularization parameter λ. We will call this best value λ^*.

$$\lambda^* = \arg\min_A \frac{1}{|B|} \sum_{(x,y)\in B} (C^T x - y)^2/2 + D$$

4.2.1)

What should we minimize over? This is denoted by the variable A.

- λ
- λ^*
- $\lambda\|\theta\|^2$
- $\lambda^*\|\theta\|^2$

Submit View Answer 100.00%

You have infinitely many submissions remaining.

4.2.2)

What dataset should we use? This is denoted by the variable B.

- D_{train}
- D_{val}
- $D_{train} \cup D_{val}$
- D_{test}

Submit View Answer 100.00%

You have infinitely many submissions remaining.

4.2.3)

What predictor should we use? This is denoted by the variable C.

- θ
- $\theta_{best}(\lambda)$
- θ^*

Submit View Answer 100.00%

You have infinitely many submissions remaining.

4.2.4)

What regularizer should we use? This is denoted by the variable D.

- 0
- λ
- λ^*
- $\lambda\|\theta\|^2$
- $\lambda^*\|\theta\|^2$

Submit View Answer 100.00%

You have infinitely many submissions remaining.

4.3) Shipping a predictor

Selecting the hypothesis (parameters θ) to deliver to the company. Call this θ^*.

$$\theta^* = \arg\min_A \frac{1}{|B|} \sum_{(x,y)\in B} (C^T x - y)^2/2 + D$$

4.3.1)

What should we minimize over? This is denoted by the variable A.

- (•) θ
- ( ) $\theta_{best}(\lambda)$
- ( ) θ^*

100.00%

You have infinitely many submissions remaining.

Solution: θ

Explanation:

To find the value θ^* that minimizes our objective function, we want to search over all θ values.

4.3.2)

What dataset should we use? This is denoted by the variable B.

- ( ) D_{train}
- ( ) D_{val}
- (•) $D_{train} \cup D_{val}$
- ( ) D_{test}

100.00%

You have infinitely many submissions remaining.

Solution: $D_{train} \cup D_{val}$

Explanation:

Before delivering the classifier to the company, we'd like to train on all the data which we have and which represents the underlying distribution that we are trying to capture. $D_{train} \cup D_{val}$ allows us to use all the data we have.

You may argue that D_{train} is the actual training data that was used to find the $\theta_{best}(\lambda)$ values in the first place. But technically, having more data (i.e., including D_{val}) should not cause problems -- it should only help the hypothesis get better.

4.3.3)

What predictor should we use? This is denoted by the variable C.

- (•) θ
- ( ) $\theta_{best}(\lambda)$
- ( ) θ^*

100.00%

You have infinitely many submissions remaining.

Solution: θ

Explanation:

We are optimizing over values of θ in order to minimize our objective function, so we must use θ.

4.3.4)

What regularizer should we use? This is denoted by the variable D.

- ( ) 0
- ( ) λ
- ( ) λ^*
- ( ) $\lambda\|\theta\|^2$
- (•) $\lambda^*\|\theta\|^2$

100.00%

You have infinitely many submissions remaining.

Solution: $\lambda^*\|\theta\|^2$

Explanation:

In the regularization term, we should use our best regularization parameter λ^*, which we found in the previous step.

4.4) Evaluation in practice

Evaluating the actual on-the-job performance ϵ^* of the selected hypothesis θ^*.

$$\epsilon^* = \frac{1}{|B|}\sum_{(x,y)\in B}(C^T x - y)^2/2 + D$$

4.4.1)

What dataset should we use? This is denoted by the variable B.

- D_{train}
- D_{val}
- $D_{train} \cup D_{val}$
- D_{test}

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

4.4.2)

What predictor should we use? This is denoted by the variable C.

- θ
- $\theta_{best}(\lambda)$
- θ^*

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

4.4.3)

What regularizer should we use? This is denoted by the variable D.

- 0
- λ
- λ^*
- $\lambda \|\theta\|^2$
- $\lambda^* \|\theta\|^2$

100.00%

You have infinitely many submissions remaining.

Solution: 0

Explanation:

Because we are performing evaluation rather than training, we have no need for regularization and want only loss, so D is 0.

5) Regression and Matrix Invertibility

In the question below, we consider how invertibility of a matrix X relates to our ability to calculate the analytical solution to a regression problem.

There is a good reason to *regularize* or put pressure on the coefficient vector θ to prevent the model from fitting the training data too closely, especially in cases where we have few data points and many features.

And, as it happens, this same regularization will help address a problem that you might have anticipated when finding the analytical solution for θ, which is that XX^T might not be invertible (where we are using the definition of X as earlier, where each column of X is a d-length vector representing a d-feature sample point and there are n columns in X (or equivalently, there are n training examples)).

If you need some help, you can watch a 3B1B video about inverse matrices.

Consider this matrix:

```
X = np.array([[1, 2], [2, 3], [3, 5], [1, 4]])
```

$$\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 5 \\ 1 & 4 \end{bmatrix}$$

Is XX^T invertible? If not, what's the problem? Mark all that are true.

- [ ] It is invertible
- [ ] It is not invertible because X is not square
- [ ] It is not invertible because two columns of X are linearly dependent
- [x] It is not invertible because the rows of XX^T are linearly dependent
- [x] It is not invertible because n is smaller than d
- [ ] We cannot compute the transpose of X

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

We can demonstrate this using the DataCommons dataset on obesity rates in U.S. cities from Lab 2. For example, the issue of matrix noninvertibility comes to play when we have fewer data points than features.

As a reminder, the original 500 cities dataset is separated into three training datasets:

- `Train_small` contains data from 10 big cities (SF, NYC, Atlanta, etc.)
- `Train_big` contains data from 200 cities
- `Train_tiny` contains data from 5 cities

And for each city, the dataset has the following data:

- `Percent_Person_Obesity`
- `Count_Person`
- `Median_Income_Person`
- `Percent_NoHealthInsurance`
- `Percent_Person_PhysicalInactivity`
- `Percent_Person_SleepLessThan7Hours`
- `Commute_Time`
- `Percent_Person_WithHighBloodPressure`
- `Percent_Person_WithMentalHealthNotGood`
- `Percent_Person_WithHighCholesterol`

The following code box performs the analytic regression on a given dataset using the specified features. Try some different combinations of datasets and features lists; you should notice that regression on `Train_tiny` with 5 or more features fails, while other combinations succeed.

```
def run():
    return analytic_regress_hw(train_data="Train_tiny",
                               features=["Count_Person",
                                         "Median_Income_Person",
                                         "Percent_NoHealthInsurance"])
```

Run Code Submit 100.00%

You have infinitely many submissions remaining.

6) Regularization and Cross Validation

We will now try to synthesize what we've learned in order to perform ridge regression on the DataCommons public health dataset that we explored in Lab 2. Unlike in Lab 2, where we did some simple linear regressions, here we now employ and explore regularization, with the goal of building a model which generalizes better (than without regularization) to unseen data.

The overall objective function for ridge regression is

$$J_{\text{ridge}}(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} \left(\theta^T x^{(i)} + \theta_0 - y^{(i)} \right)^2 + \lambda \|\theta\|^2$$

Remarkably, there is an analytical function giving $\Theta = (\theta, \theta_0)$ which minimizes this objective, given X, Y, and λ. But how should we choose λ?

To choose an optimum λ, we can use the following approach. Each particular value of λ gives us a different linear regression model.

And we want the best model: one which balances providing good predictions (fitting well to given training data) with generalizing well (avoiding over-fitting training data). And as we saw in the notes on Regression, we can employ **cross-validation** to evaluate and compare different models.

Implementation of cross-validation algorithm

Let us begin by implementing this algorithm for cross-validation:

$\textsc{Cross-Validate}(\mathcal{D}, k)$
1 divide $\mathcal{D}$ into k chunks $\mathcal{D}_1, \mathcal{D}_2, \ldots \mathcal{D}_k$ (of roughly equal size)
2 **for** $i = 1$ **to** k
3 train h_i on $\mathcal{D} \setminus \mathcal{D}_i$ (withholding chunk $\mathcal{D}_i$)
4 compute "test" error $\mathcal{E}_i(h_i)$ on withheld data $\mathcal{D}_i$
5 **return** $\frac{1}{k} \sum_{i=1}^{k} \mathcal{E}_i(h_i)$

6.1)

Let's implement the cross-validation algorithm as the procedure `cross_validate`, which takes the following input arguments:

- `X`: the list of data points ($d \times n$)
- `Y`: the true values of the responders ($1 \times n$)
- `n_splits`: the number of chunks to divide the dataset into
- `lam`: the regularization parameter
- `learning_algorithm`: a function that takes `X`, `Y`, and `lam`, and returns `th`, `th0`
- `loss_function`: a function that takes `X`, `Y`, `th`, and `th0`, and returns a 1x1 array

`cross_validate` should return a scalar, the cross-validation error of applying the learning algorithm on the list of data points.

Note that this is a generic version of cross-validation, that can be applied to any learning algorithm and any loss function. Later in this problem, we will use cross-validation specifically for ridge regression and mean square loss.

You have the following function available to you:

```
def make_splits(X, Y, n_splits):
    '''
    Splits the dataset into n_split chunks, creating n_split sets of
    cross-validation data.
    Returns a list of n_split tuples (X_train, Y_train, X_test, Y_test).
    For the ith returned tuple:
    *X_train and Y_train include all data except the ith chunk, and
    * X_test and Y_test are the ith chunk.

    X : d x n numpy array (d = #features, n = #data points)
    Y : 1 x n numpy array
    n_splits : integer
    '''
```

It's a good idea to write and test your code elsewhere (e.g., in the colab notebook linked at the top of this homework), then paste it below and submit. The code package in the colab notebook includes a simple test case.

```
def cross_validate(X, Y, n_splits, lam,
                   learning_algorithm, loss_function):
    data_sets = make_splits(X, Y, n_splits)
    test_error = []
    for i in range(n_splits):
        th, th0 = learning_algorithm(data_sets[i][0], data_sets[i][1], lam)
        test_error.append(loss_function(data_sets[i][2], data_sets[i][3], th, th0))
    return 1/n_splits*(np.sum(np.array(test_error)))
```

Run Code | Submit | View Answer **100.00%**

You have infinitely many submissions remaining.

Optimization of regularization parameter

Now that you have implemented a cross-validation algorithm, we want you to think about finding an optimum value of λ for predicting obesity rates.

6.2)

Below, `X` and `Y` are sample data, and `lams` is a list of possible values of `lambda`. Write code to set `errors` as a list of corresponding cross-validation errors. Use the `cross_validate` function above to run cross-validation with three splits. Use the following functions (which we implement for you, per the specifications below) as the learning algorithm and loss function:

```
def ridge_analytic(X_train, Y_train, lam):
    '''Applies analytic ridge regression on the given training data.
    Returns th, th0.

    X : d x n numpy array (d = # features, n = # data points)
    Y : 1 x n numpy array
    lam : (float) regularization strength parameter
    th : d x 1 numpy array
    th0 : 1 x 1 numpy array'''

def mse(x, y, th, th0):
    '''Calculates the mean-squared loss of a linear regression.
    Returns a scalar.

    x : d x n numpy array
    y : 1 x n numpy array
    th : d x 1 numpy array
    th0 : 1 x 1 numpy array'''
```

```
X = np.array([[4, 6, 8, 2, 9, 10, 11, 17],
              [1, 1, 6, 0, 5, 8, 7, 9],
              [2, 2, 2, 6, 7, 4, 9, 8],
              [1, 2, 3, 4, 5, 6, 7, 8]])
Y = np.array([[1, 3, 3, 4, 7, 6, 7, 7]])
lams = [0, 0.01, 0.02, 0.1]
errors = [cross_validate(X, Y, 3, lam, ridge_analytic, mse) for lam in lams] # your coc

```

Run Code Submit View Answer 100.00%

You have infinitely many submissions remaining.

6.3)

We have prepared two datasets, `cv_small` and `cv_big`, for you to perform 10-fold cross-validation on using different values of λ and test which is best. Both of these are subsets of the datacomms dataset, and a working implementation of `cross_validate_data` is provided for you.

Try both and note what you observe:

```
def run():
    # replace with cv_big
    return cross_validate_data(data="cv_big")

```

Run Code Submit 100.00%

You have infinitely many submissions remaining.

6.4)

As the number of data points increases,

We tend to see:

- [ ] The optimal λ increase
- [x] The optimal λ decrease
- [ ] The minimum cross-validation error increase
- [x] The minimum cross-validation error decrease

100.00%

You have infinitely many submissions remaining.

Solution:

✗ The optimal λ increase

✓ The optimal λ decrease

✗ The minimum cross-validation error increase

✓ The minimum cross-validation error decrease

Explanation:

With less data, we want the regularization to be stronger, in order to generalize better. With more data, we don't need as much regularization (optimum lambda decreases), and our model generalizes better (minimum cross-validation error drops).

6.5)

Based on our observations, does regularization have a larger effect when we have more or less data?

Regularization is much more important when we have:

- more data
- (selected) less data
- around the same for both

100.00%

You have infinitely many submissions remaining.

Solution: less data

Explanation:

As explained in the previous section, regularization is much more important when we have less data.