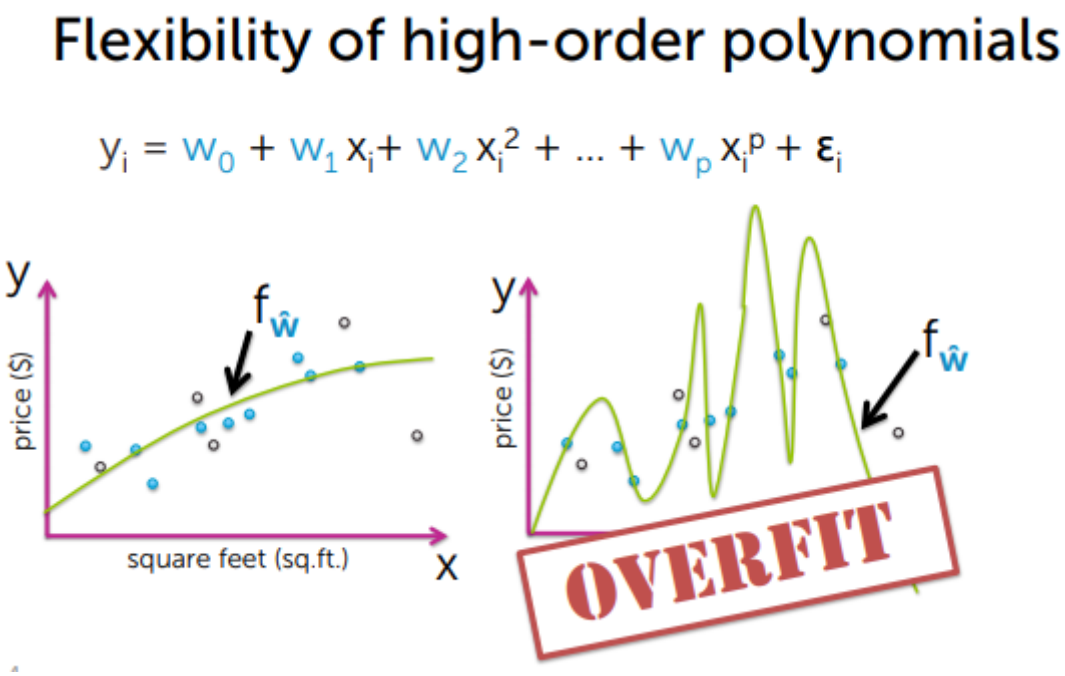
Ridge Regression

- High complexity model -> **Low Bias** and **High Variance** and vice-versa.
- Bias-Variance trade-off - required to achieve good predictive performance.
- **Ridge Regression** - automatically balance between Bias and Variance.

Overfit:

- When a model is highly specific to the training data and does not generalize well - then it is **Overfit**.
- Training error of model w' > training error of another model w1
- True error of model w' < True error of another model w1

Polynomial -> features are power of an input.



When models become overfit the estimated parameters (w-hat) become very large in magnitude.

Overfiting is generic issue with complex models

- It occurs with polynomial regression.
- Occurs in complex models
  - lots of inputs (d large).
  - lots of features(D large).y(j) = Σ(j=0--D) w(j)h(j) + ε(i)

How does # of observations influence overfitting?

- Few observations **(N small)** -> **rapidly overfit** as model **complexity increases**. With few points, as the order of the polynomial increases, it becomes easy to hit all the datapoints - hence overfit the dataset.
- Many observations **(N is large)** -> **harder to overfit**.
- It doesn't overfit easily since there are dense observations to overfit the input. It's not able to hit all the datapoints in the observations.

How does # of inputs influence overfitting?

- 1 input (e.g., sqft):
- The dataset must be very dense - must include representative examples of all possible (sqft, 'dollars') pairs to avoid overfitting.
- This is hard, to have all possible (sqft, 'dollars') pairs values.
- d inputs (e.g., sqft, #bath, #bed, lot size, year,...):
- The data must include examples of all possible (sqft, #bath, #bed, lot size, year,...,'$') combos to avoid overfitting.
- This is even harder to cover all possible scenarios.

Balancing the fit and magnitude of coefficients

- Overfitting increases the magnitude of the coefficient.

Quality Metrics -> Adding term to cost-of-fit to prefer small coefficients.

- Thus far -> quality metrics - depended on the ***actual sales price / actual output*** and the ***predicted sales price / predicted output*** - **RSS** - was used to measure of fit.
- Now The Quality Metrics - is also gonna incorporate the **complexity of the model**.

Desired total cost format

Want to balance :

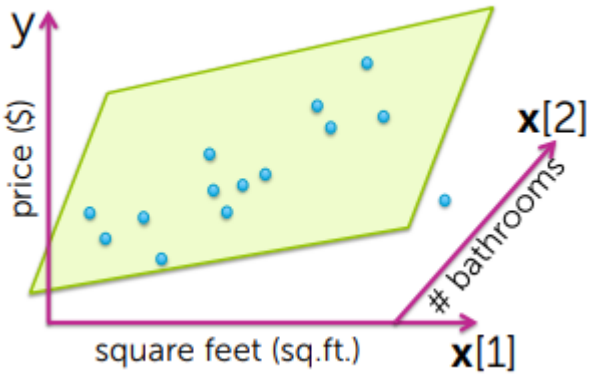
1. How well the function fits the data.
2. The complexity of the model - through 'Magniude of the estimated coefficients'.

Previous cost = RSS = (Actual - Predicted)^2.-> measure of fit

New Cost = Total cost = measure of fit + measure of magnitude of coefficients

- Measure of fit -> (small # = good fit to training data).
- Measure of magnitude of coefficients -> (small # = not overfit).

I. Measure of fit



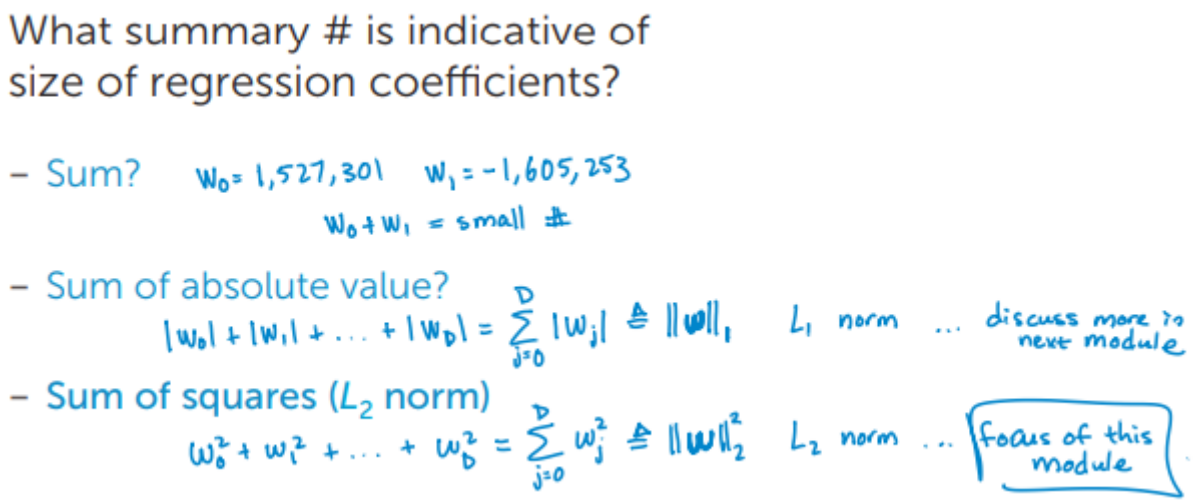
$$RSS(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T\mathbf{w})^2$$

$$= \sum_{i=1}^{N} (y_i - \hat{y}_i(\mathbf{w}))^2$$

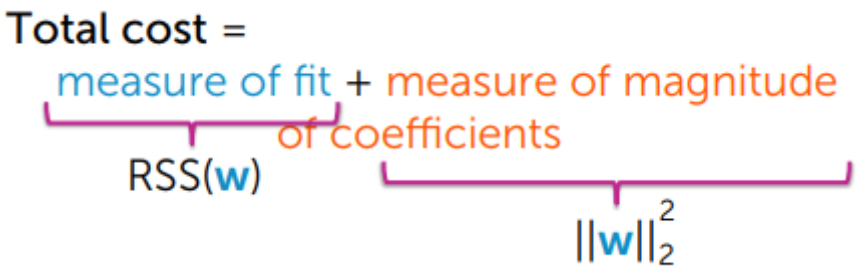
pred. value using w

small RSS → model fitting training data well

II. Measure of magnitude of the coeffients

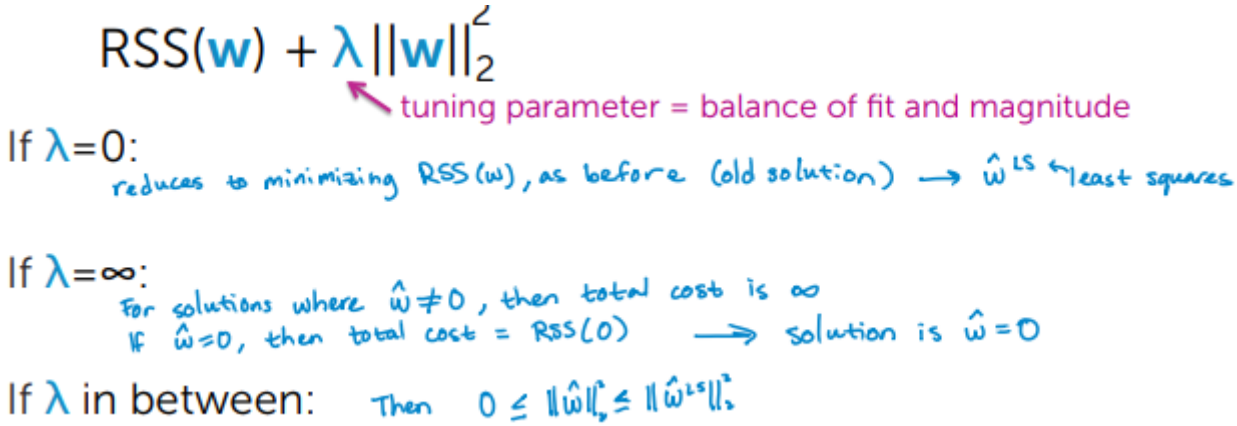


Total Cost

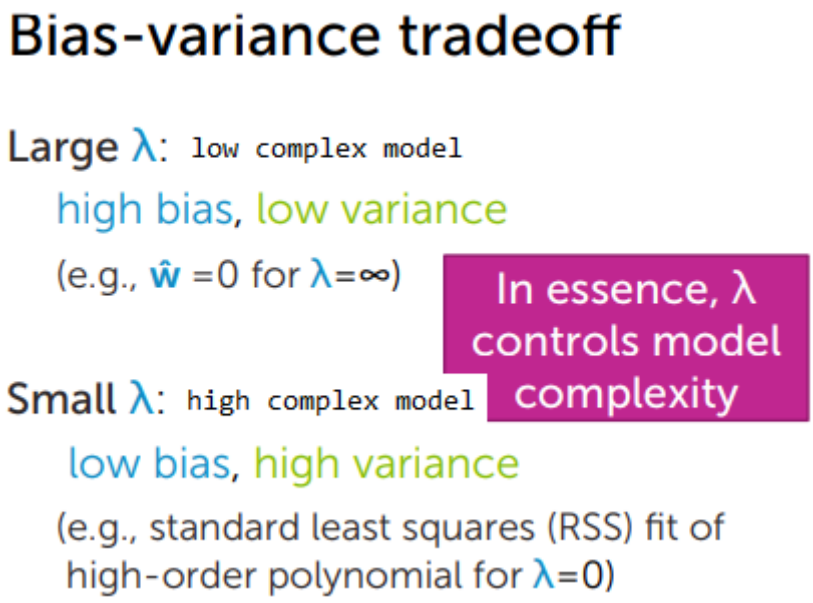


Resulting Objective

- Find the estimated parameters (w-hat) that minimizes the **total cost**.
- A tuning parameter **λ** is introduced inorder to balance of fit and magnitude.

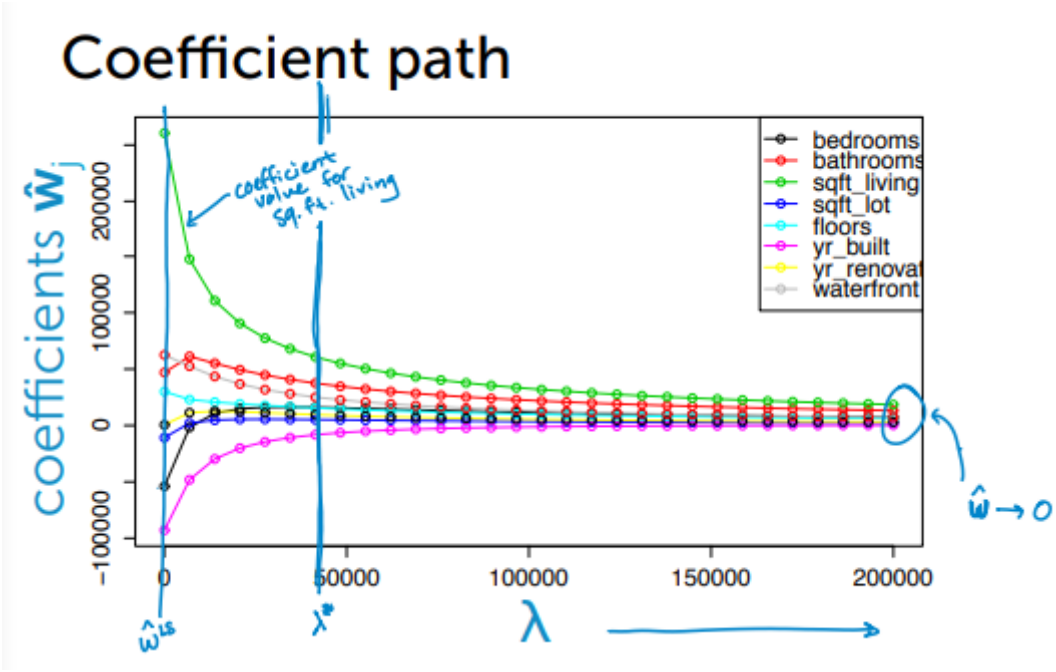


Ridge Regression - a.k.a L(2) regularization = RSS(w) + λ||w||(2) ^2



The ridge coefficient path:

- In general when **λ** is small, the coefficient magnitude is large.
- When **λ** is large -> infinity, the coefficient magnitude is small -> 0;



ML Algorithm -> ML Block Diagram

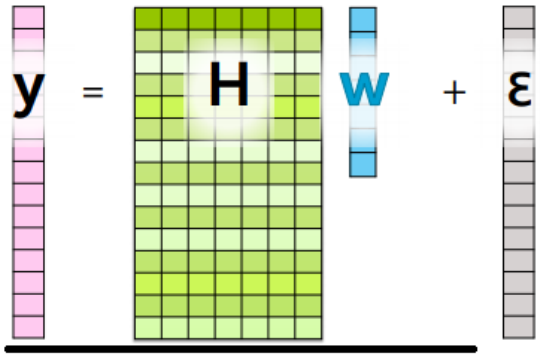
Computing the gradient of the Ridge Objective

Step 1: Rewrite total cost in matrix notation

- Model for all N observations together. Y = H * w + ε

Recall matrix form of RSS

Model for all N observations together



$$RSS(\mathbf{w}) = \sum_{i=1}^{N} (y_i - h(\mathbf{x}_i)^T\mathbf{w})^2$$
$$= (\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})$$

Rewrite magnitude of coefficients in vector notation

$$||\mathbf{w}||_2^2 = w_0^2 + w_1^2 + w_2^2 + \ldots + w_D^2$$
$$= \begin{bmatrix} w_0 & w_1 & w_2 & \cdots & w_D \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}$$
$$= \mathbf{w}^T\mathbf{w}$$

Putting it all together

In matrix form, ridge regression cost is:

$$RSS(\mathbf{w}) + \lambda||\mathbf{w}||_2^2$$
$$= (\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w}$$

Step 2: Compute the gradient

Gradient of ridge regression cost

$$\nabla[RSS(\mathbf{w}) + \lambda||\mathbf{w}||_2^2] = \nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w}]$$
$$= \underbrace{\nabla[(\mathbf{y}-\mathbf{H}\mathbf{w})^T(\mathbf{y}-\mathbf{H}\mathbf{w})]}_{-2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w})} + \lambda \underbrace{\nabla[\mathbf{w}^T\mathbf{w}]}_{2\mathbf{w}}$$

Why? By analogy to 1d case...

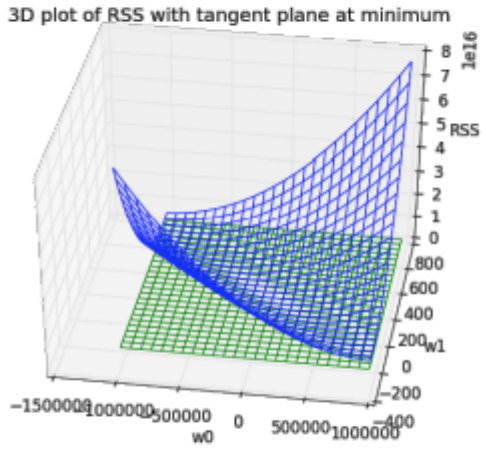
$\mathbf{w}^T\mathbf{w}$ analogous to w^2 and derivative of $w^2=2w$

Step 3: Approach 1 : Set the gradient = 0 | Closed-form solution

Ridge closed-form solution

$$\nabla cost(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + 2\lambda\mathbf{w} = -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + 2\lambda\mathbf{I}\mathbf{w}$$

equivalent



$$\nabla cost(\mathbf{w}) = -2\mathbf{H}^T(\mathbf{y}-\mathbf{H}\mathbf{w}) + 2\lambda\mathbf{I}\mathbf{w} = 0$$

Solve for $\hat{w}$:

$$-H^T y + H^T H\hat{w} + \lambda I\hat{w} = 0$$

$$H^T H\hat{w} + \lambda I\hat{w} = H^T y$$

$$(H^T H + \lambda I)\hat{w} = H^T y$$

$$\hat{w}^{ridge} = (H^T H + \lambda I)^{-1} H^T y$$

Interpreting ridge closed-form solution

$$\hat{w}^{ridge} = (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T\mathbf{y}$$

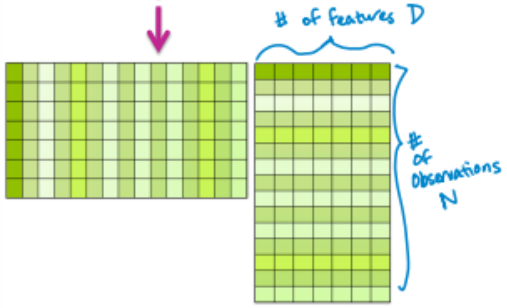
If $\lambda=0$: $\hat{w}^{ridge} = (H^T H)^{-1} H^T y = \hat{w}^{LS}$ ← old solution!

If $\lambda=\infty$: $\hat{w}^{ridge} = 0$ ← because it's like dividing by ∞

Closed-form solution : w.r.t Ridge regression -> the λ term is multiplied with I (identity matrix) -> λ is a scalar -> identity matrix with λ along the diagonal and the rest are 0 is formed.

previous closed-form solution

$$\hat{w}^{LS} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y}$$

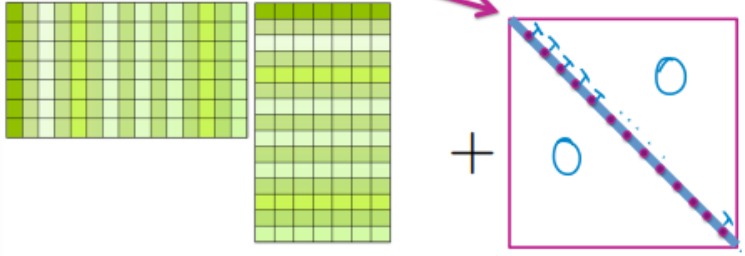


Invertible if:
- In general,
- (# linearly independent obs)
- N > D

Complexity of inverse:
- $O(D^3)$

ridge closed-form solution

$$\hat{w} = (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T\mathbf{y}$$



Invertible if:
- Always if λ>0, even if N < D

really important for large D (lots of features)

Complexity of inverse:
- $O(D^3)$...
- big for large D!

λI is making $H^T H + \lambda I$ more "regular" → "regularized"

- The closed-form solution is computatively expensive.

Step 3: Approach 2 : Gradient Descent

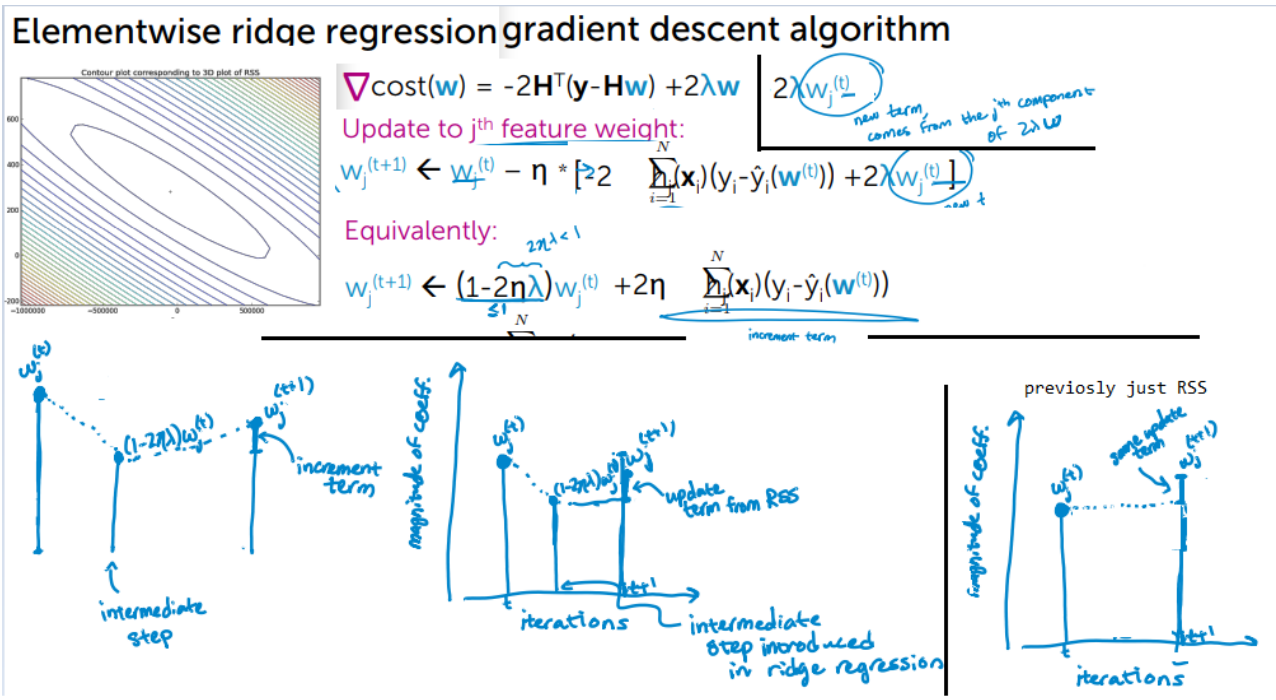
- The new coefficient

```
w(j)(t+1) = (1-2ηλ) * w(j)(t) + 2ηΣ(i=1...N) hj(x) (yi - y-hati(w-hat))
```

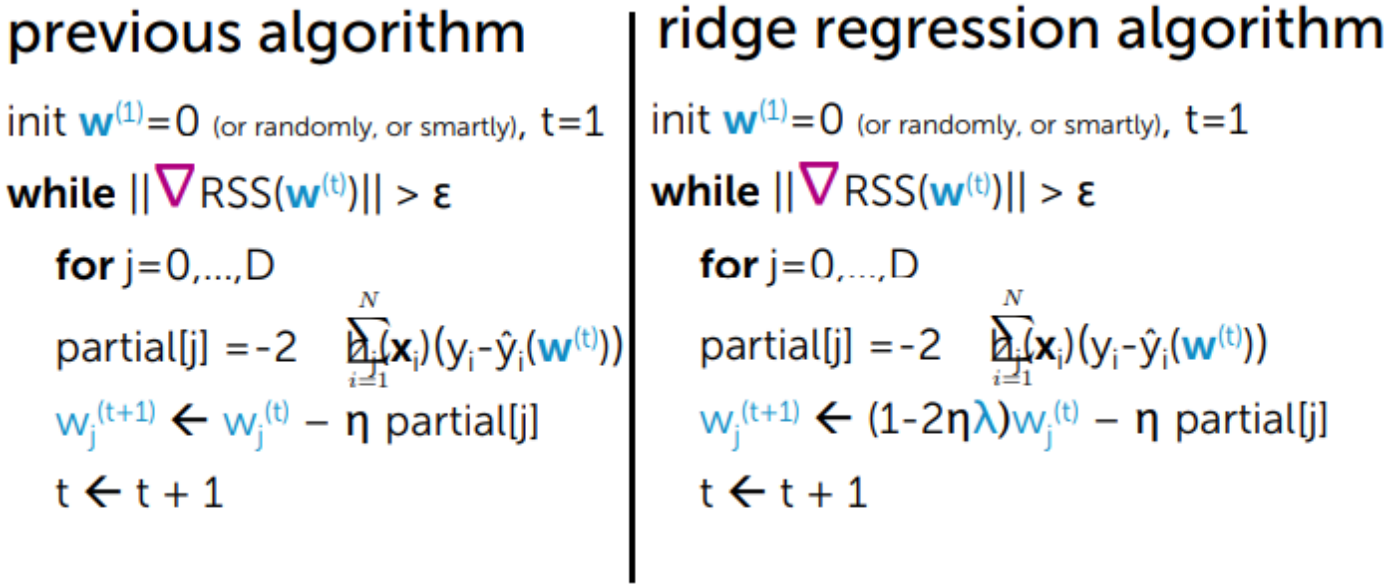
- Step i: With every new interation -> (1-2ηλ) reduces the w(j)(t) term since **η** & **λ** are > 0.
- Step ii: At t+1 iteration, the update term for RSS is added to the previously shrink term. This is the new **w(j)(t+1) - coefficient**.

Previously - Just RSS

- Here the **w(j)(t)** term was taken and for the new **w(j)(t+1)** term the **w(j)(t)** was added with **update term from RSS**.

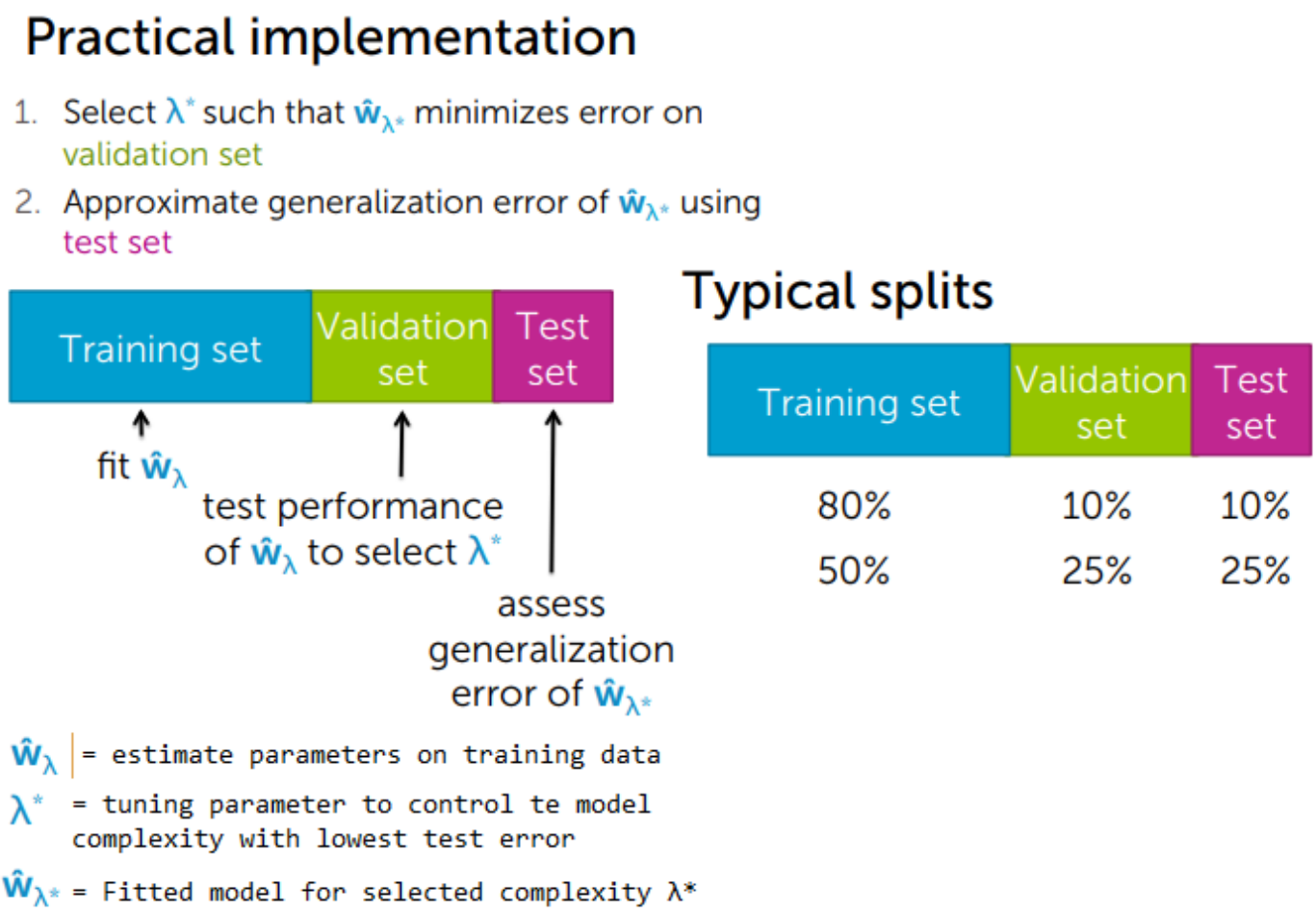


Gradient Descent - implemented w.r.t 'Multiple Regression' v/s 'Ridge Regression'



Selecting tuning parameters λ via cross validation

Case 1 : Sufficient amount of data



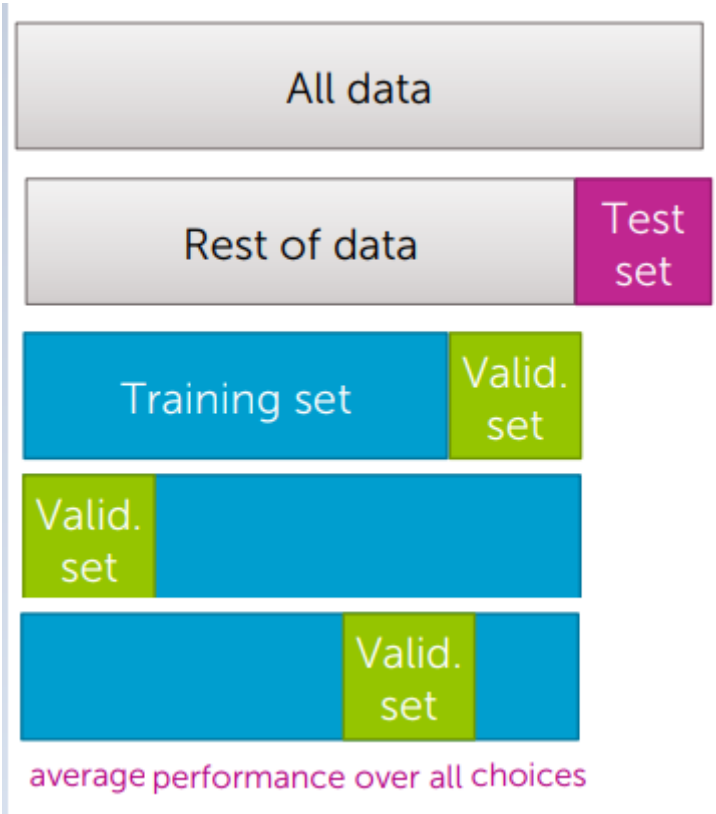
- For every value of that tuning parameter we can fit our model in the training data.
- Then assess the performance of the fitted model on a validation set, and can tabulate all values of 'lambda' that we can consider and choose the specific model complexity accord to error on validation set.
- Assess the performance of the selected model on the test set.
- In the presence of **sufficient data** the above splitting process can be applied even to 'Ridge Regression'.

Case 1 : Insufficient amount of data

```
------------------------------
|            DATA            |
------------------------------
              |
             \ /
------------------------------
|    Rest of data    |Test Set|
------------------------------

----------------------
|    Rest of data    |
----------------------
          |
         \ /
----------------------
| Training     |Valida|
|              |-tion |
| set          |set   |
----------------------
```

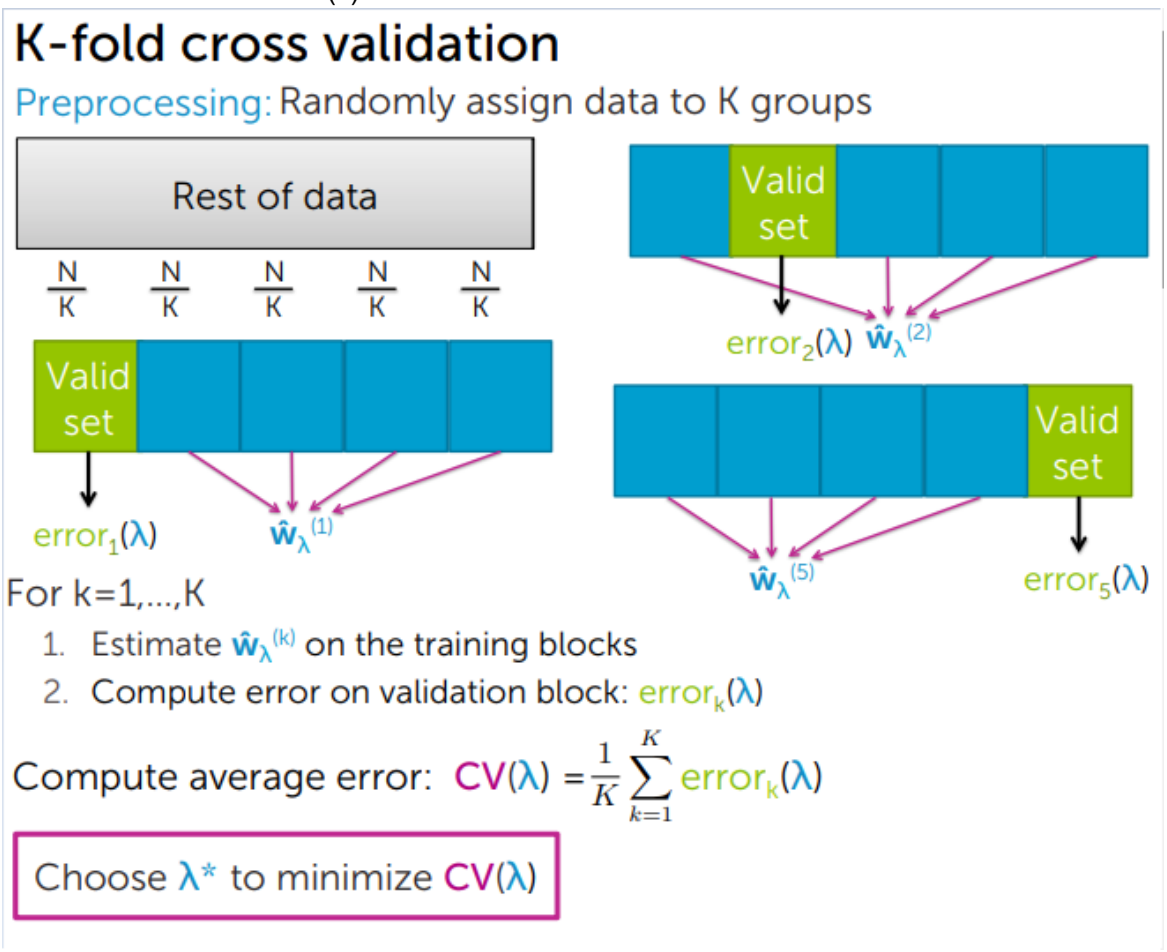
- Is the validation set enough to compare performance of estimated paraters for a given tuning parameter 'lambda' **(w-hat)λ** across all 'lambda' **λ** values?
- It is not necessary to use the last data points tabulated to form the valiation set. Rather can use any subset.
- In case of insufficin=ent data. Can use the entire training dataset in subsets. **Then Average the performance over all choices of the subset**.



K-fold cross validation

- Step 1 : Preprocessing : Randomly assign the data to K groups.

- Take the Rest of the data (data apart from the test set) and divide into into k groups. There are N total observations so every block is gonna have **N/K** observations and are assigned randomly to each block.
- Step 2: For k = 1,...,K
  - i. Estimate (w-hat)λ(k) on the training blocks (except the k -subset);
  - ii. Compute the error on the validation block (K) : error-k(λ)
- Step 3: Compute the average error : CV(λ) = 1/K Σ(k = 1-K) error-k(λ).
- Step 4: Repeat the above procedure for each choice of λ.
  - Choose λ* to minimize CV(λ).



What value of K?

- Formally, the best approximation occurs for validation sets of size 1 (K=N)

```
 ------------------------------
|          Rest of data        |
 ------------------------------
 ------------------------------
|   |   |   |   |   |   |   |   |
 ------------------------------
 1/K 1/K 1/K 1/K 1/K 1/K 1/K 1/K
```

- This is the **leave-out-one cross validation** (LOO Cross Validation).
- It is computationally intensive - requires computing N fits of model per λ.
- Typically , K = 5 **(5-fold CV)** or k = 10 **(10-fold CV)**.

How to handle the intercept

- w0 -> first coefficient of the model is usually the intercept, i.e where x is 0. h0(x) = contant (1).
- Therefore the H matrix first column will be 1.
- Since w0 is the intercept.
- The cost of Ridge Regression -> RSS(w) + λ||w||2^2;

```
- where λ - strength of penalty.
- since w0 is the intercept multipled by 1, squaring it will encourage the
 intercept w0 to also be small.
- This operation will not be indicative of overfitting. (Overfitting is ind
icated bt the magnitude of the coefficients.)
```

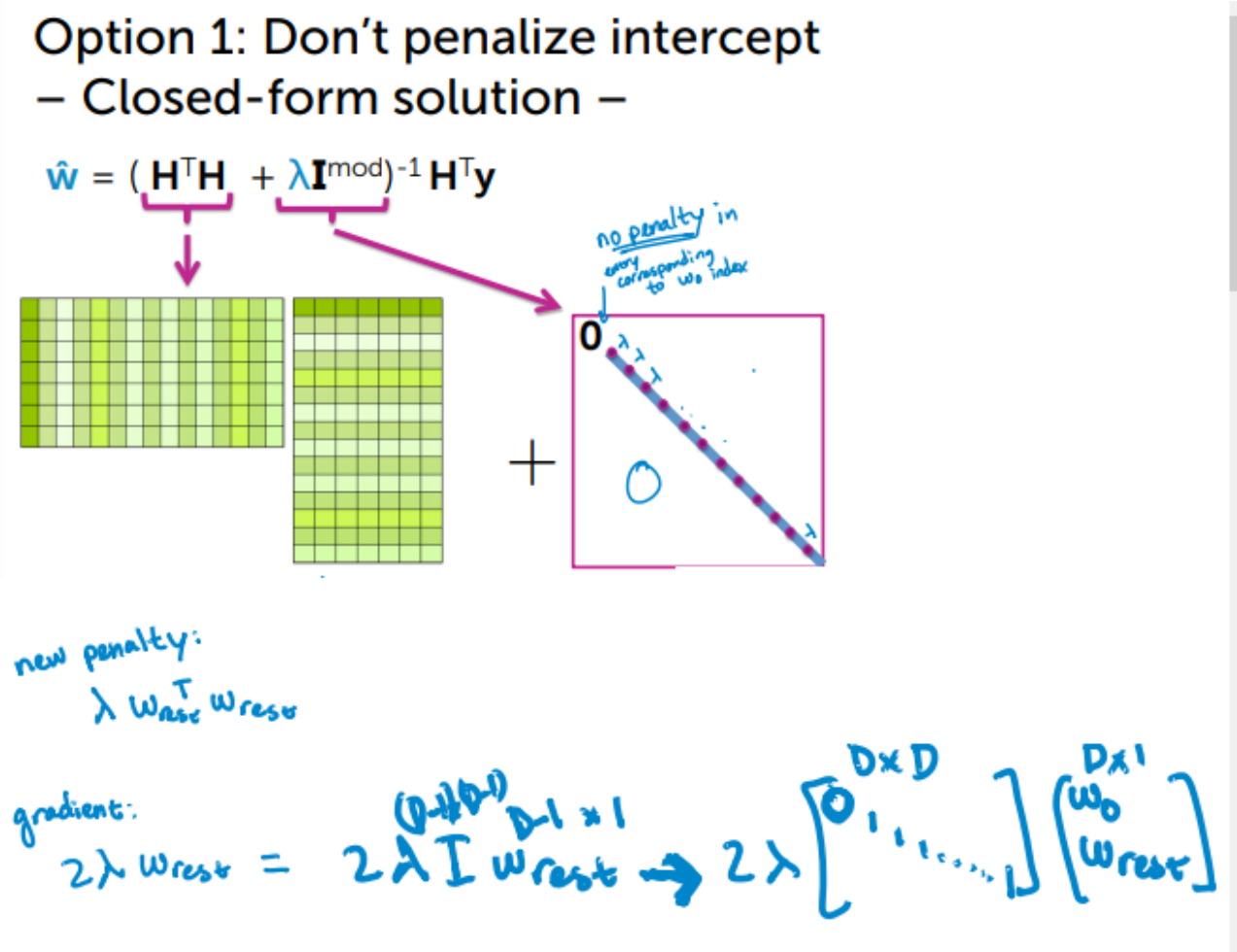
Avoid the intercept term from being squared.

Option 1: Don't penalize the intercept

- Previous Ridge Regression Cost : RSS(w) + λ||w||2^2;
- New Ridge Regression Cost : RSS(w0,w-rest) + λ||w-rest||2^2;

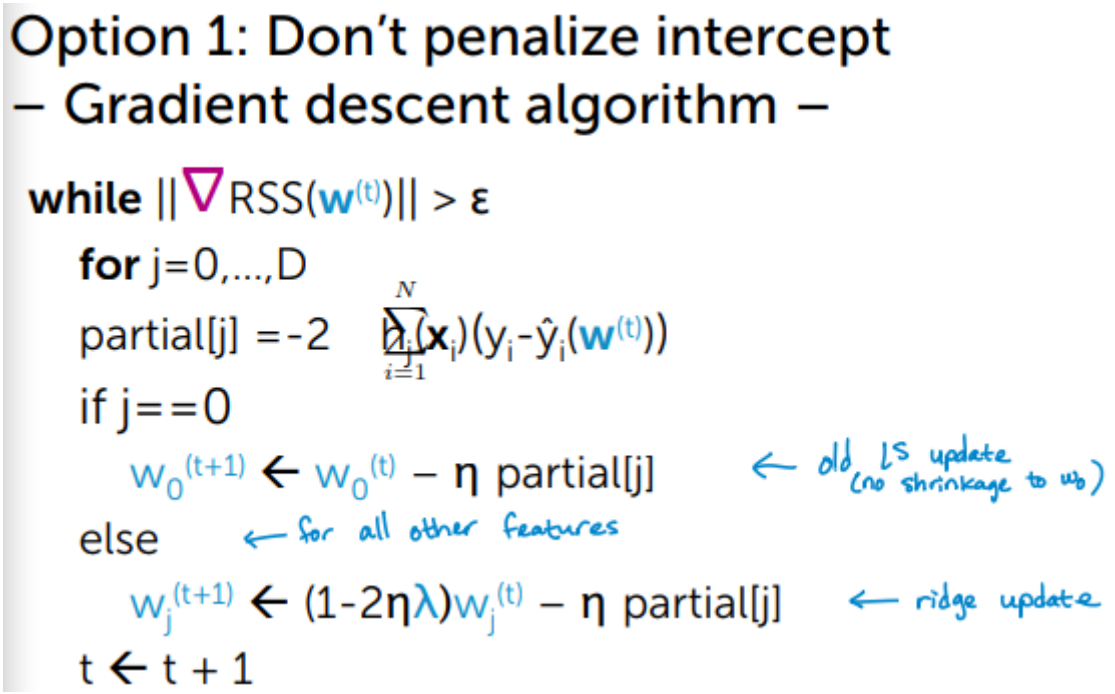
Closed-form solution

- Here the Identity matrix 'λI' gets modified to 'λI(mod)' - the λI(mod) matrix has the first element as 0, and the rest of the diagonal is filled with λ and the rest of the matrix is filled with 0's.



Gradient Descent algorithm

- While performing the update, incase j = 0, w0(intercept - coefficient) - no shrinking operation is performed, only the fit to data - **update term from RSS in added*.
- For all other cases -> the previous coefficient is shrunk and then appended to form the next coefficient of the polynomial regression term.



Option 2 : Center the data first

- If data are first centered about 0, then favoring small intercept not so worrisome.
- Step 1 : Transform y to have 0 mean.
- Step 2 : Run ridge regression as normal (closed-form or gradient descent algorithm).

Quiz

1 point

1. **Which of the following is NOT a valid measure of overfitting?**

- (●) Sum of parameters ($w_1 + w_2 + \ldots + w_n$)
- ( ) Sum of squares of parameters ($w_1^2 + w_2^2 + \ldots + w_n^2$)
- ( ) Range of parameters, i.e., difference between maximum and minimum parameters
- ( ) Sum of absolute values of parameters ($|w_1| + |w_2| + \ldots + |w_n|$)

- Explaination : Take weight vector in 2D like the following one : w = = From the value of the weights, we might suspect overfitting. Now: If you consider the sum of weights, you get -200. So this is not a good measure for overfitting detection. On the other hand, if you consider range, you get 2500 - (-2700) = 5200, which seems good measure for overfitting detection.

---

1 point

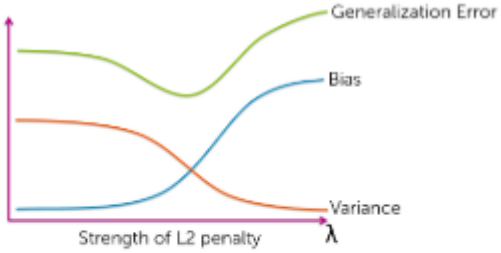
2. **In ridge regression, choosing a large penalty strength λ tends to lead to a model with (choose all that apply):**

- [■] High bias
- [ ] Low bias
- [ ] High variance
- [■] Low variance

* When lambda is large (model complexity is low) - Bias is high and variance is low.

1 point

3. Which of the following plots best characterize the trend of bias, variance, and generalization error (all plotted over λ)?



- The bias increases with 'lambda'.
- The variancec decreases with 'lambda'.
- The Generalization error does not portoprtionally vary with model complexity. It decrease when the bias meets the variance. Increases else where.

---

1 point

4. In ridge regression using unnormalized features, if you double the value of a given feature (i.e., a specific column of the feature matrix), what happens to the estimated coefficients for every other feature? They:

- Double
- Half
- Stay the same
- Impossible to tell from the information provided

- Suppose, w0 + w1 *x1*+ *w2* x2 = y.
- Let x1 = 2 and x2 = 4 for y = 7, then one set of likely values of (w0, w1, w2) will be (1,1,1).
- If x2 = 8, w2 will have to be half of previous value and will be less than other two parameters.
- Since ridge regression penalizes larger parameters more, using lambda, values of other parameters should change.
- Hence the estimated parameters may or may not change.

---

1 point

5. If we only have a small number of observations, K-fold cross validation provides a better estimate of the generalization error than the validation set method.

- (●) True
- ( ) False

---

1 point

6. 10-fold cross validation is more computationally intensive than leave-one-out (LOO) cross validation.

- ( ) True
- (●) False

1 point

7. Assume you have a training dataset consisting of N observations and D features. You use the closed-form solution to fit a multiple linear regression model using ridge regression. To choose the penalty strength λ, you run leave-one-out (LOO) cross validation searching over L values of λ. Let $\text{Cost}(N, D)$ be the computational cost of running ridge regression with N data points and D features. Assume the prediction cost is negligible compared to the computational cost of training the model. Which of the following represents the computational cost of your LOO cross validation procedure?

- ( ) $LN \cdot \text{Cost}(N, D)$
- (●) $LN \cdot \text{Cost}(N-1, D)$
- ( ) $LD \cdot \text{Cost}(N-1, D)$
- ( ) $LD \cdot \text{Cost}(N, D)$
- ( ) $L \cdot \text{Cost}(N-1, D)$
- ( ) $L \cdot \text{Cost}(N, D)$

- Since it is LOO, each block is 1 observation wide. therefore 1 observation is a validation set , while the rest (N-1) observations are a part of the training set.
- Therefore computationaal cost = L *N* Cost(N-1, D)

---

1 point

8. Assume you have a training dataset consisting of 1 million observations. Suppose running the closed-form solution to fit a multiple linear regression model using ridge regression on this data takes 1 second. Suppose you want to choose the penalty strength λ by searching over 100 possible values. How long will it take to run leave-one-out (LOO) cross-validation for this selection task?

- ( ) About 3 hours
- ( ) About 3 days
- (●) About 3 years
- ( ) About 3 decades

- N = # of observation = 10^6;
- L = # of lambda = 100;
- Cost(N-1, D) operation = 1 second
- T = L *N* Cost(N-1, D) = 100 *10^6* 1 second
- T = 100 *10^6* 1 second / (60 *60* 24 * 365) ~ 3 years.

---

1 point

9. **Assume you have a training dataset consisting of 1 million observations. Suppose running the closed-form solution to fit a multiple linear regression model using ridge regression on this data takes 1 second. Suppose you want to choose the penalty strength λ by searching over 100 possible values. If you only want to spend about 1 hour to select λ, what value of k should you use for k-fold cross-validation?**

- ○ k=6
- ◉ k=36
- ○ k=600
- ○ k=3600

- k - folds cross validaions.
- L = # lambda = 100;
- t = time to select lamba* over all lambda = 1 hour
- k = t/ L = 1 hour / 100 = 60 * 60 / 100 = 36;

In [ ]: