- Intuitively:
  - The **loss surface** for just the mean squared error component is in blue.
  - The constraint, $\sum_{j=1}^{d} w_j^2 \le Q$, is in green.

    The larger Q is, the larger the radius of the ball is.

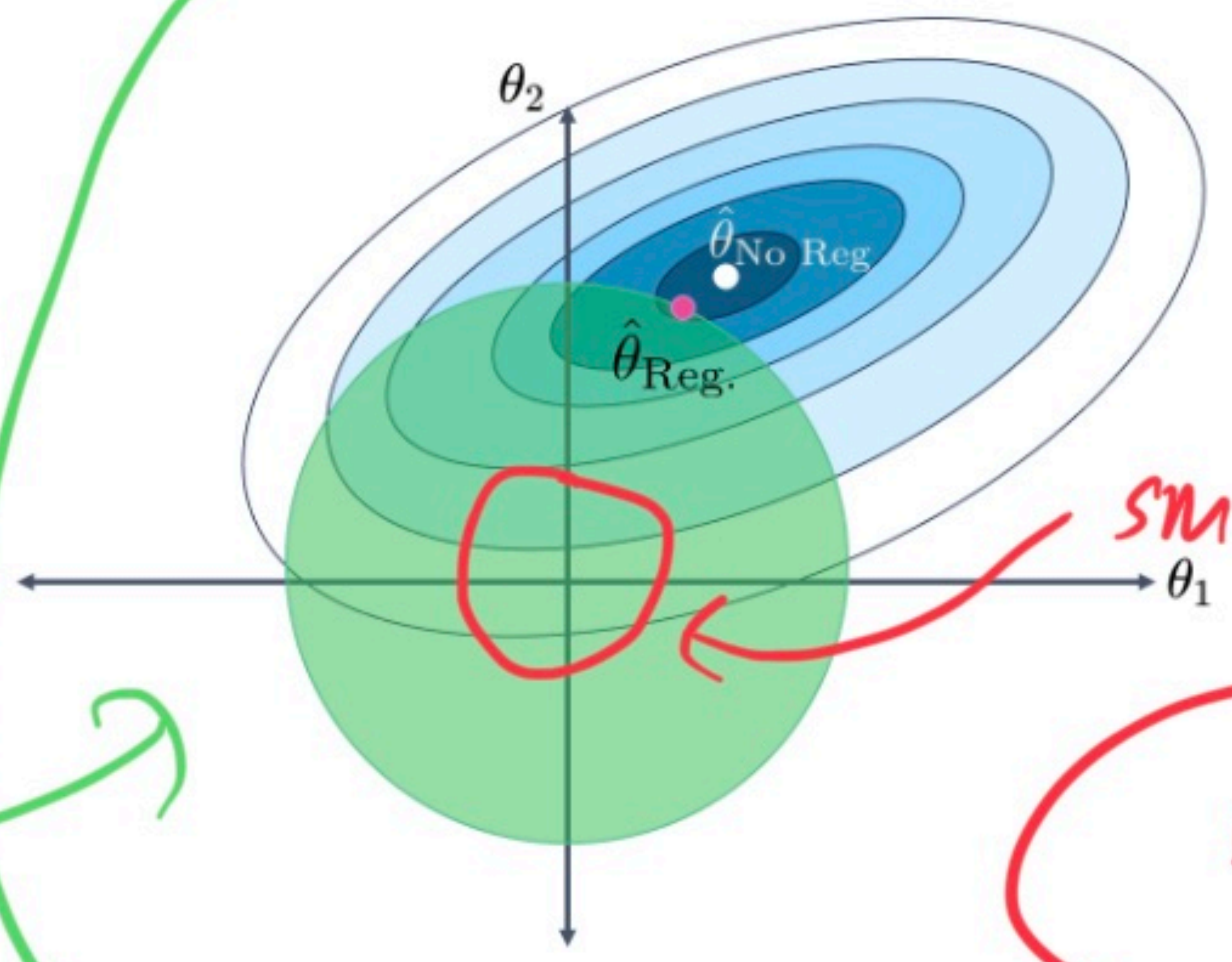
small λ

big Q:

$\vec{w}^*_{ridge} = \vec{w}^*_{OLS}$!

a circle with radius $\sqrt{Q}$

$w_1^2 + w_2^2 \le Q$



(source)

small Q: $\vec{w}^*_{ridge}$ all very close to 0, very constraining

big λ

localhost

- Sometimes, $\vec{w}^*_{\text{OLS}}$ is unique, and sometimes there are infinitely many possible $\vec{w}^*_{\text{OLS}}$.
  There are infinitely many possible $\vec{w}^*_{\text{OLS}}$ when the design matrix, X, is not full rank! All of these infinitely many solutions minimize mean squared error.

- Which vector $\vec{w}^*_{\text{ridge}}$ minimizes the ridge regression objective function?

$$R_{\text{ridge}}(\vec{w}) = \frac{1}{n}\|\vec{y} - X\vec{w}\|^2 + \lambda \sum_{j=1}^{d} w_j^2$$

- It turns out there is **always** a unique solution for $\vec{w}^*_{\text{ridge}}$, even if X is not full rank. It is:

$$\vec{w}^*_{\text{ridge}} = (X^TX + n\lambda I)^{-1}X^T\vec{y}$$

adds $n\lambda$ to the diagonal elements of X^TX

The proof is outside of the scope of the class, and requires vector calculus.

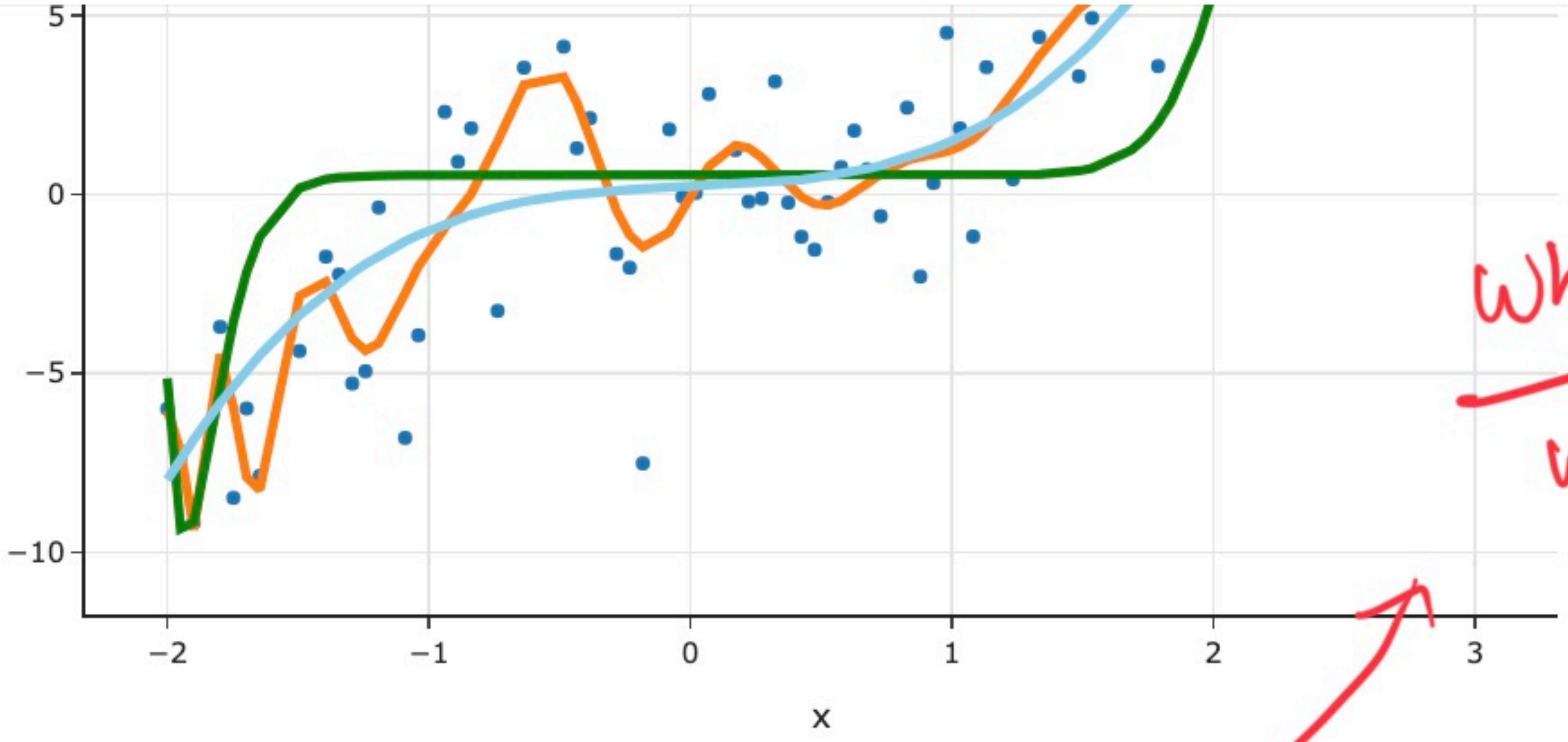
- Since there is **always** a unique solution, ridge regression is often used in the presence of multicollinearity!

Taking a step back

- $\vec{w}_{\text{ridge}}^*$ **doesn't** minimize mean squared error – it minimizes a slightly different objective function.

- So, why would we use ever use ridge regression?

we hope the resulting model will peform better on unseen test data than if we don't regulaize!



Why null?
we didn't use cross-validation because there were no hyperparameters to choose!

In [26]:
```
display(HTML(results_df_str))
```

	Unregularized (Degree 25)	Regularized (Degree 25) Used cross-validation to choose λ	Regularized (Degree 3) Used cross-validation to choose λ and degree
training MSE	4.72	10.33	7.11
average validation MSE (across all folds)	NaN	17.60	7.40
test MSE	14.21	17.17	10.52

cv for λ cv for λ, degree

- It seems that the regularized polynomial, in which we used cross-validation to choose both the regularization penalty λ **and** degree, generalizes best to unseen data!

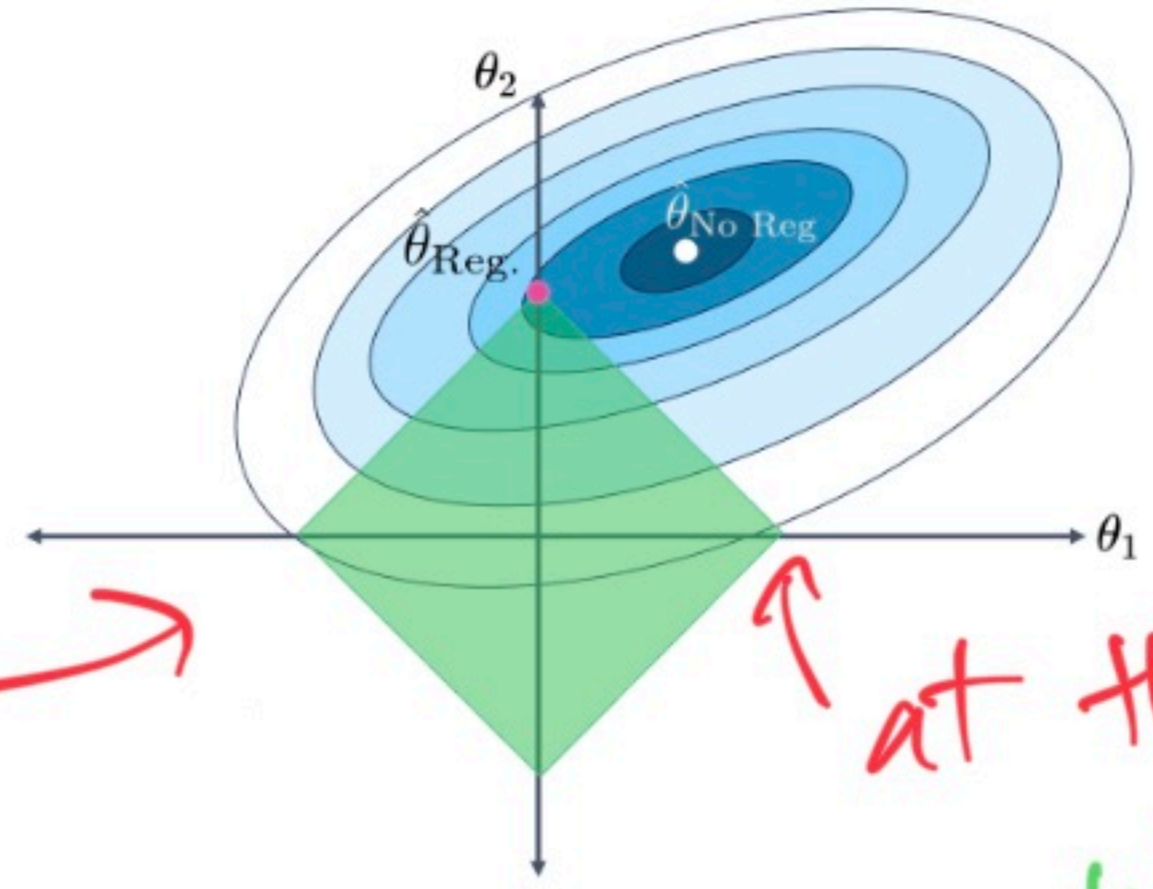
- Again:
  - The **loss surface** for just the mean squared error component is in **blue**.
  - The constraint, $\sum_{j=1}^{d} |w_j| \leq Q$, is in **green**.

    The larger Q is, the larger the side length of the diamond is.

spiky constraints: more likely to hit a spike!!!

example

$|x| + |y| \leq 5$



at this corner, $w_1^* = 5$, $w_2^* = 0$

(source)

- Notice that the **constraint set** has clearly defined "corners," which lie on the axes. The axes are where the parameter values, w_1 and w_2 here, are 0.

- Due to the shape of the constraint set, it's likely that the minimum value of **mean squared error**, among all options in the **green diamond**, will occur at a corner, where some of the parameter values are 0.

Aside: average validation error vs. λ

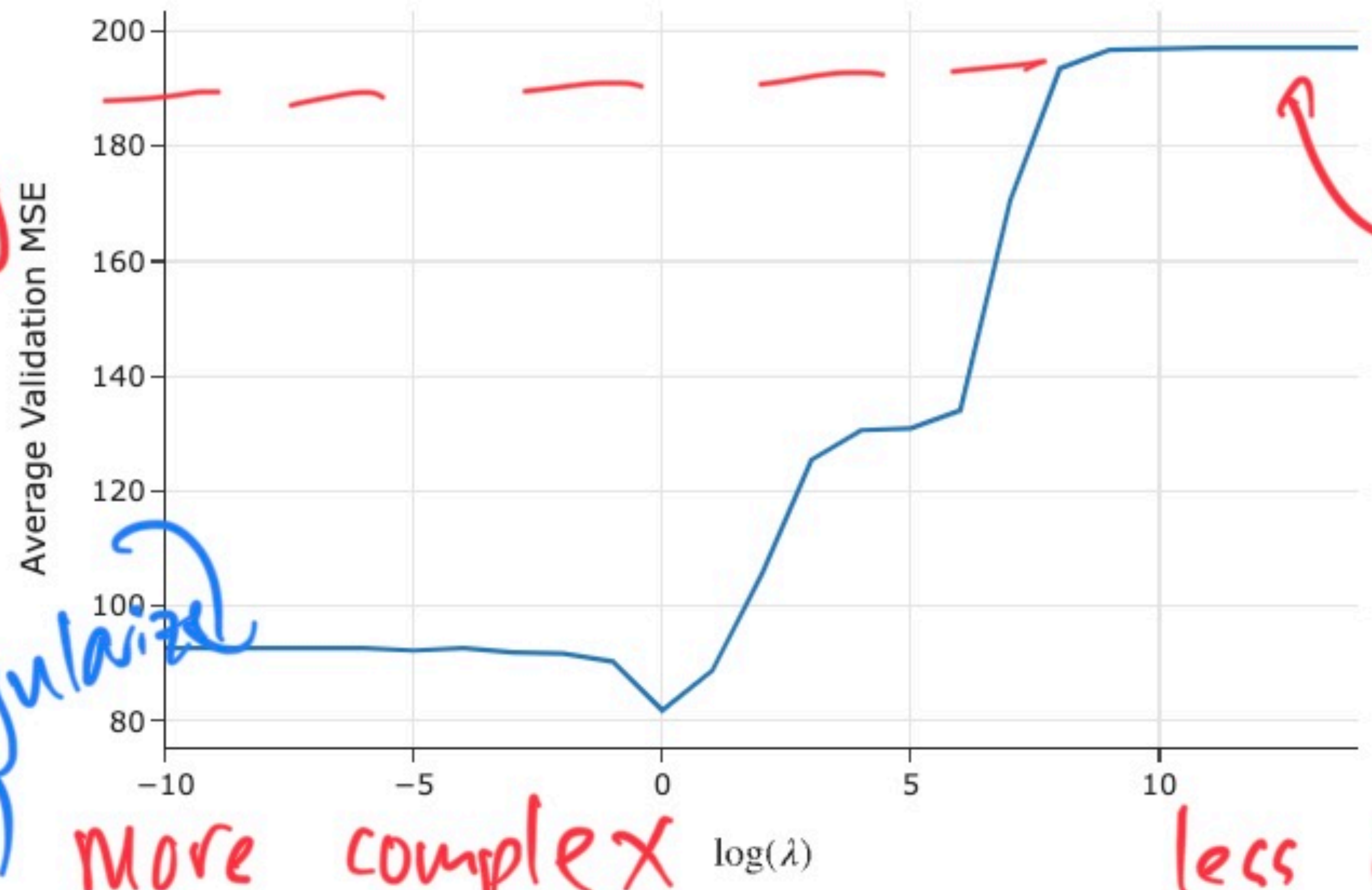
- How did the average validation MSE change with λ?

  Here, large values of λ mean **less complex models**, not more complex.

In [44]:
```
(
    pd.Series(-commute_model_ridge.cv_results_['mean_test_score'],
              index=np.log10(lambdas))
    .to_frame()
    .reset_index()
    .plot(kind='line', x='index', y=0)
    .update_layout(xaxis_title='$\log(\lambda)$', yaxis_title='Average Validation MSE')
)
```

as $\lambda \to \infty$, predictions $\to$ constant model

variance of y



height = MSE of the constant model = variance of y!!!

MSE if you don't regularize (OLS)

more complex   less complex

feature	ols	ridge	lasso
intercept	460.31	214.15	2.54e+02
polynomialfeatures__departure_hour	-94.79	-0.71	-2.10e+01
polynomialfeatures__departure_hour^2	6.80	-4.63	-1.70e+00
polynomialfeatures__departure_hour^3	-0.14	0.31	1.81e-01
onehotencoder__day_Mon	-0.61	-5.74	-2.70e+00
onehotencoder__day_Thu	13.30	6.04	9.00e+00
onehotencoder__day_Tue	11.19	5.52	8.68e+00
onehotencoder__day_Wed	5.73	-0.46	0.00e+00
onehotencoder__month_December	8.90	2.82	4.06e+00
onehotencoder__month_February	-5.33	-7.14	-5.81e+00
onehotencoder__month_January	1.93	0.39	0.00e+00
onehotencoder__month_July	2.46	0.44	0.00e+00
onehotencoder__month_June	6.28	4.45	5.14e+00
onehotencoder__month_March	-0.76	-1.70	-8.17e-01
onehotencoder__month_May	9.36	4.95	5.57e+00
onehotencoder__month_November	1.40	-1.81	-0.00e+00
onehotencoder__month_October	2.06	0.22	0.00e+00
onehotencoder__month_September	-3.20	0.05	-0.00e+00
pipeline__day_of_month_Week 2	0.91	1.39	3.23e-01
pipeline__day_of_month_Week 3	6.30	4.70	4.57e+00
pipeline__day_of_month_Week 4	0.28	-0.20	-0.00e+00
pipeline__day_of_month_Week 5	2.09	0.76	4.78e-03

What do you notice?

got larger in magnitude!

lots of 0 coefficients with LASSO!

all small, but none 0!