Try your best, on #1.

#2 is hard.

Ridge Regression

Alvin Wan

alvinwan.com/cs189

Quote of the day

"Give me your tired, your poor, your huddled masses yearning to breathe free, the wretched refuse of your teeming shore."

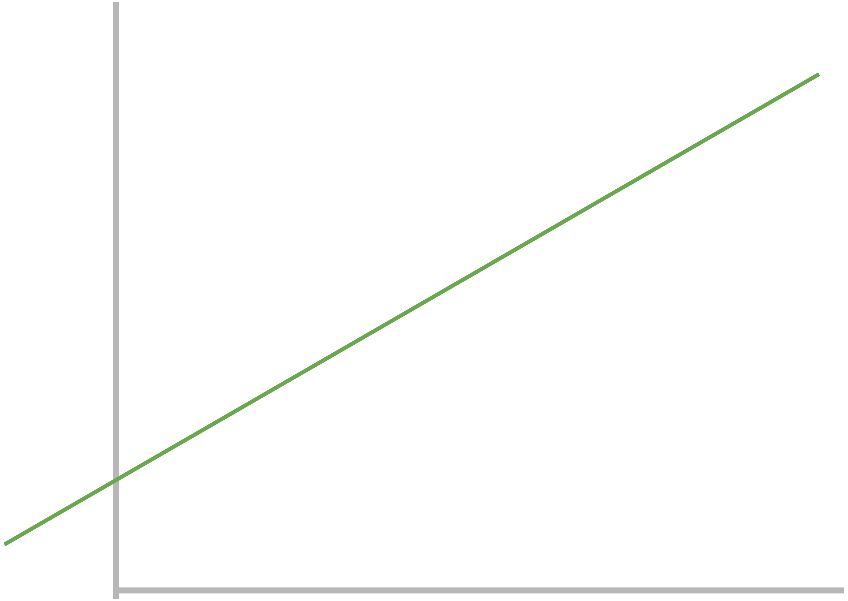
-- Sinho, EE 126 (to me)

Let’s talk content.

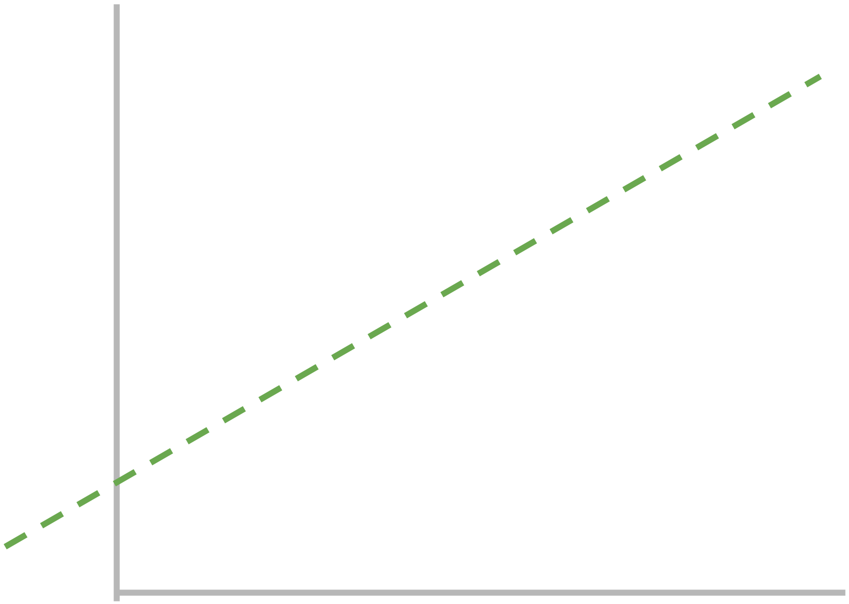
1 - Proxy for “True” Error

How well our model represents the true model

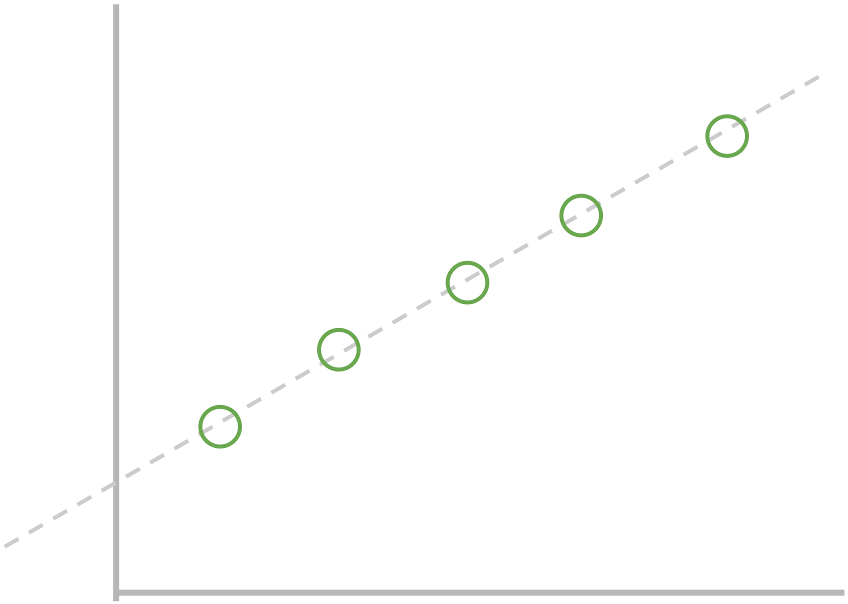
True error?



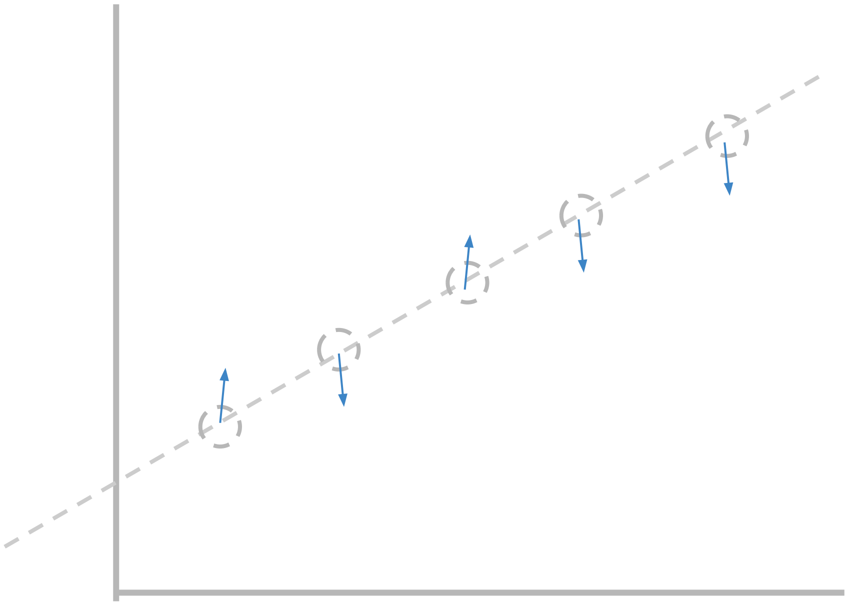
True model



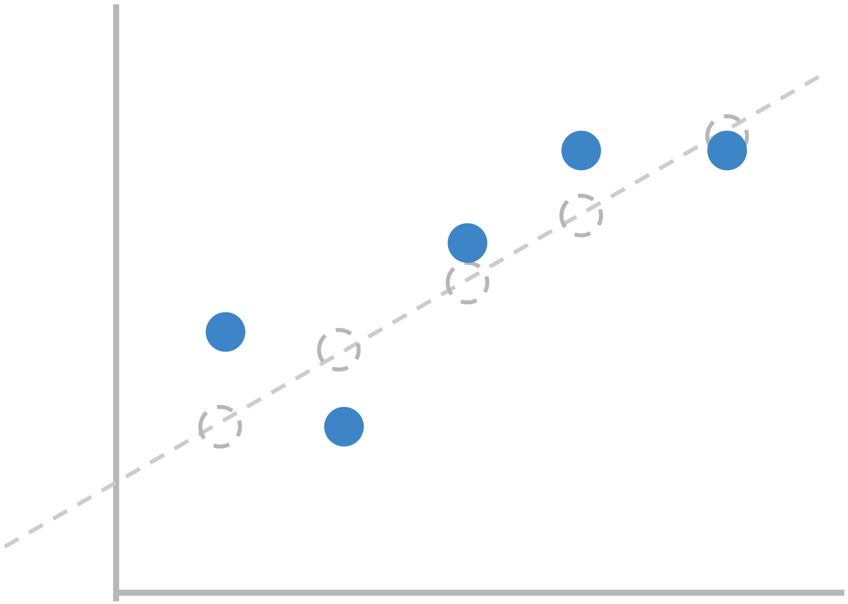
True model



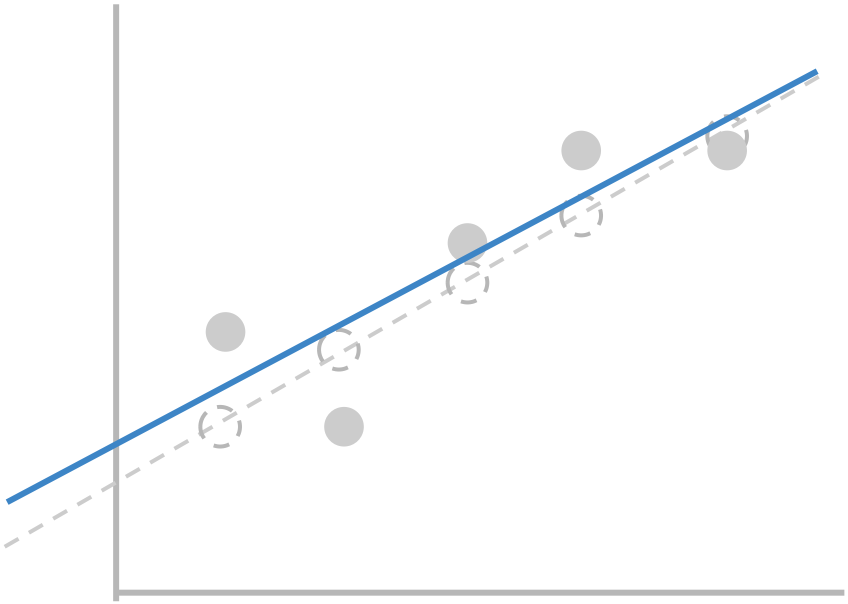
True data



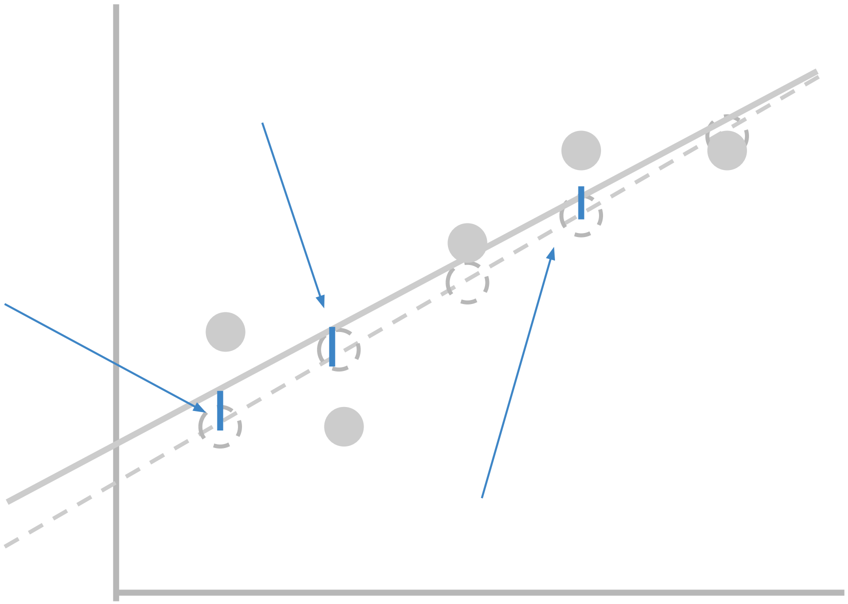
Noise



Observations



Proposed model



True error

Some of that error is irreducible. Inherent noise in observations.

Why is it irreducible??

Math

Math

Math

Math

Math

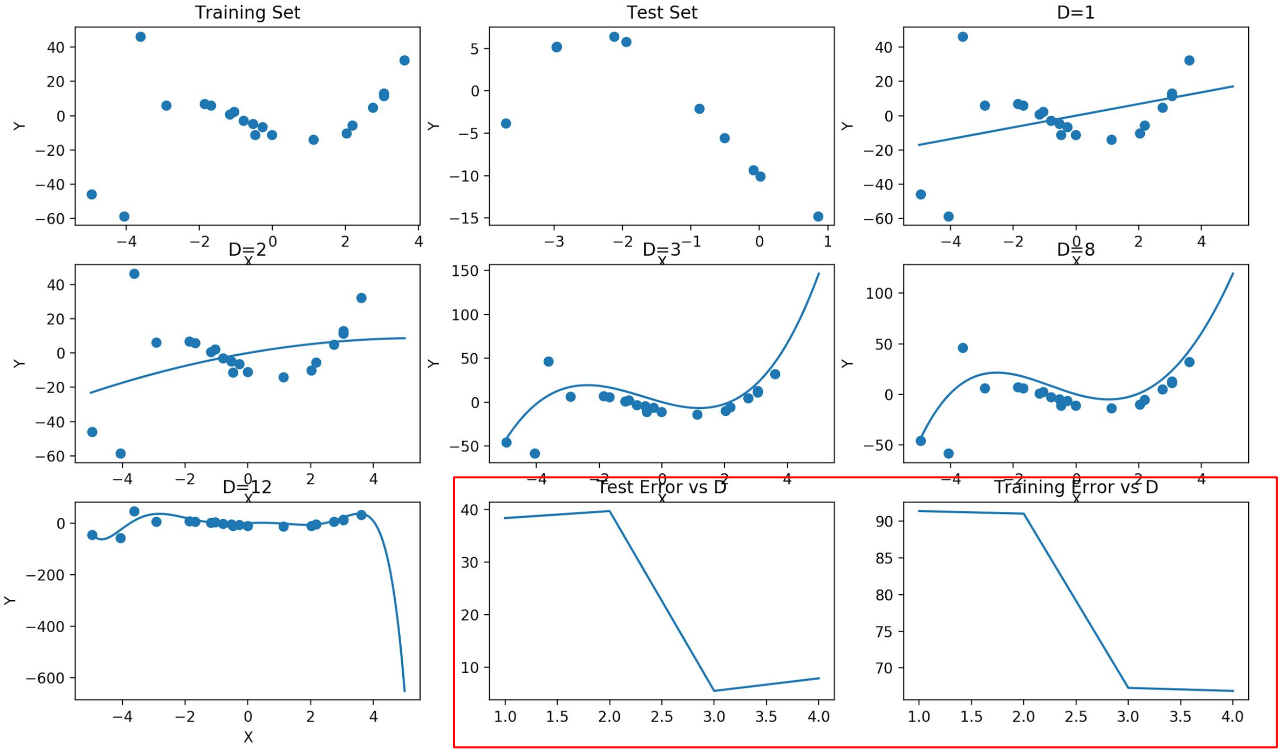
Math

Math

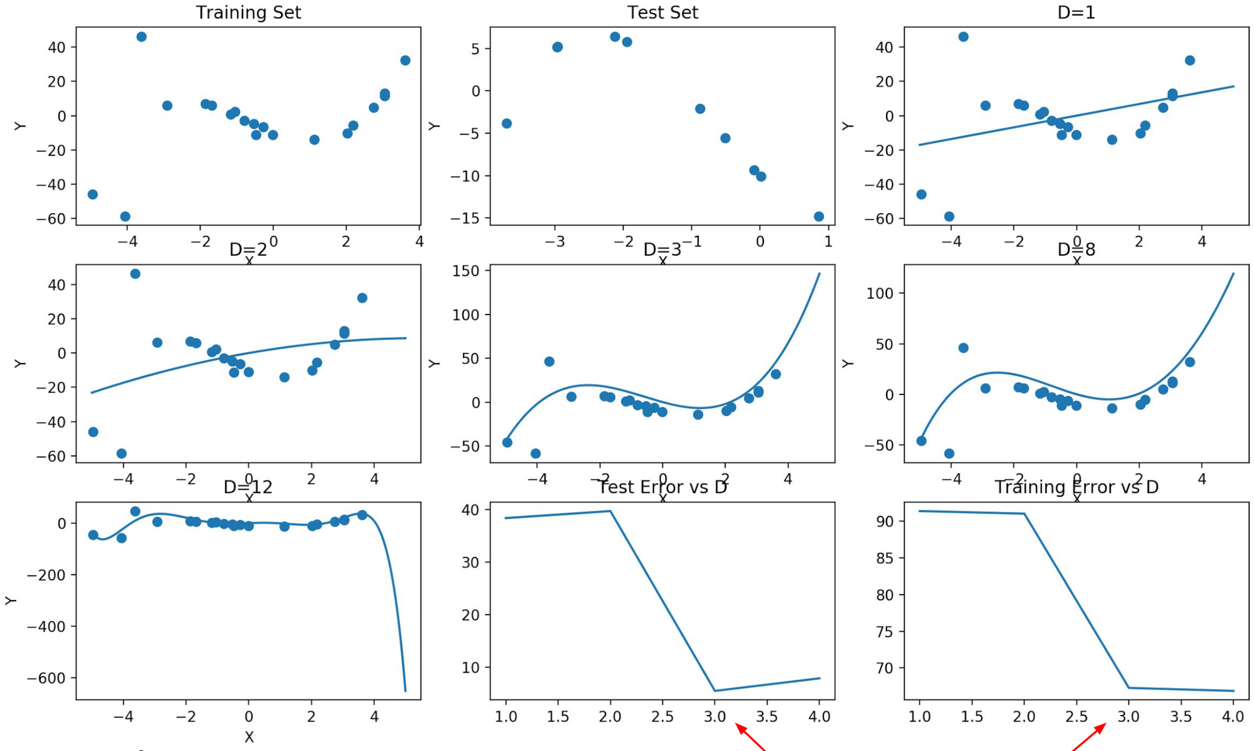
Math

Next week.

So, our model can suck.

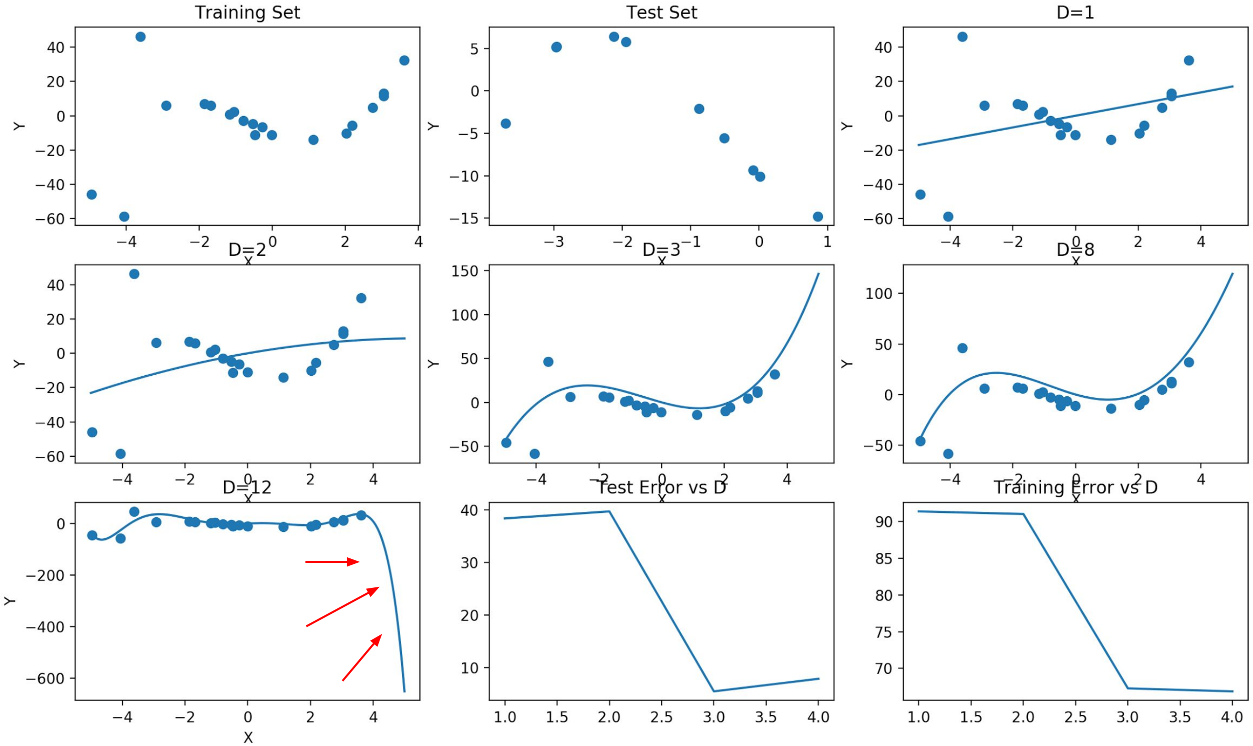


Model order complexity

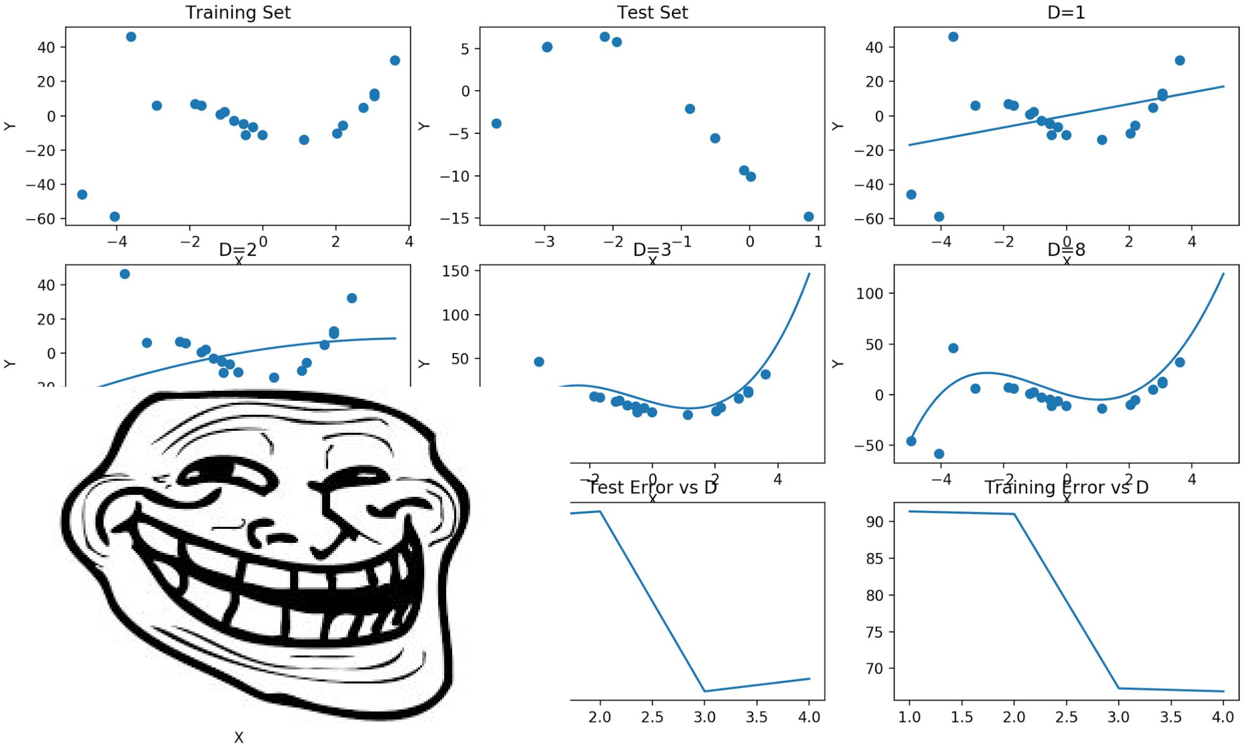


Model order complexity

Sweet spot



Model order complexity



Model order complexity

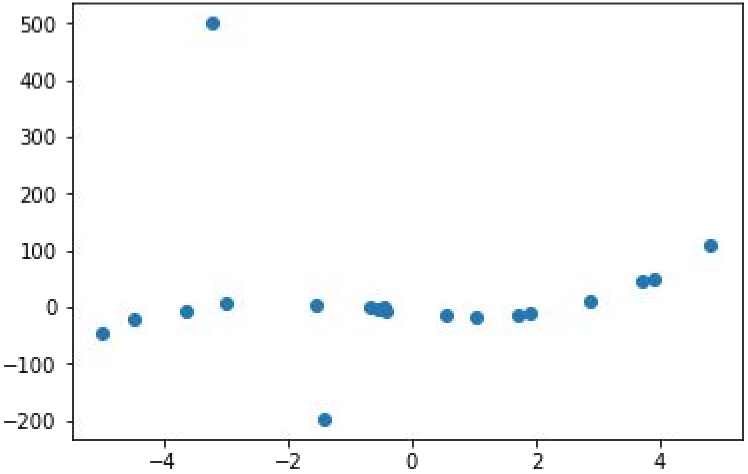
Overfitting is the enemy…

2 - Tricks of the Trade

Intuition behind regularization

Step 2

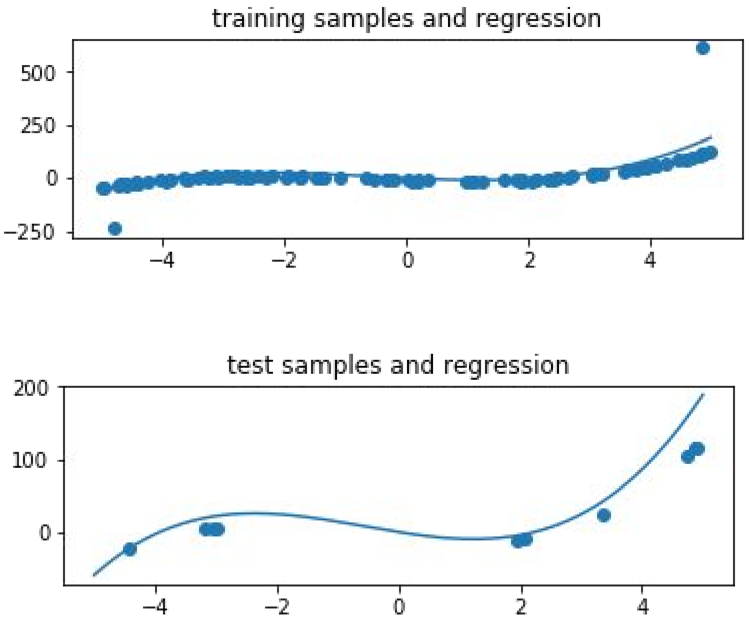
Try some matplotlib. Use `gen_data`



```
data = gen_data(1, 20, 10)
plt.scatter(data['Xtrain'], data['Ytrain'])
```

Step 3

Use `my_regression`. Play with values.



```
my_regression(data,3,0.1, plot=True)
```

Step 4

Plot as a function of lambda.

Step 5

Trick 1: Make your train-val split

```
indices = list(range(n_total))
random.shuffle(indices)
all_x = all_x[indices]
all_y = all_y[indices]
arr_train_x = all_x[: Ntrain]
arr_train_y = all_y[: Ntrain]
arr_val_x = all_x[Ntrain :]
arr_val_y = all_y[Ntrain :]
```

Step 6

Play with different splits, parameters.

Step 7 & 8

Plot as function of model complexity, lambda.

Feedback? aaalv.in/survey