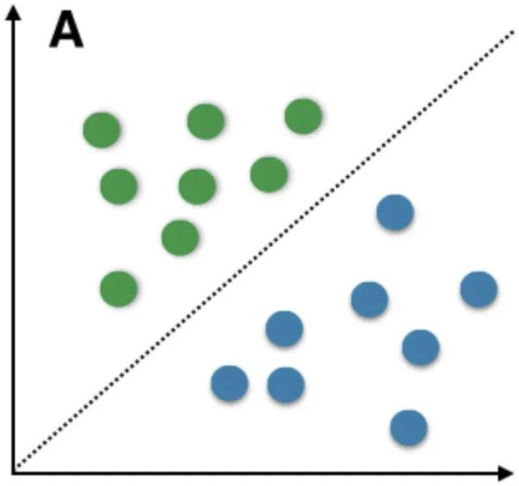
16ML: Polynomial Regression & Non-linear Boundaries

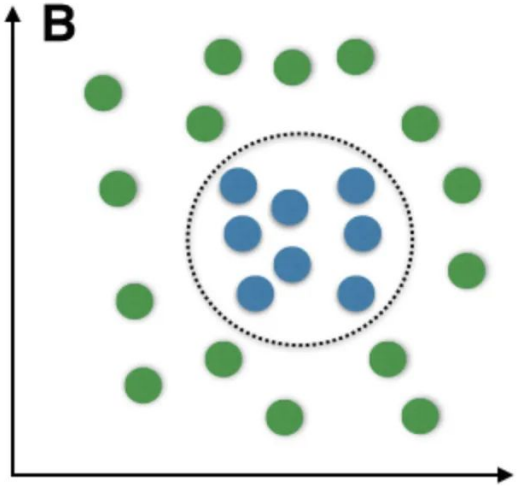
Team CCMJ: Angela Chiang, Christina Lu, Jianheng Luo, Lu Yang, Minan Wu

When linear regression fails...

Linear regression is an easy way to generate a boundary when you have linearly separable data e.g. plot A below. But what about plot B?

Can we use linear regression for plot B? Why or why not?

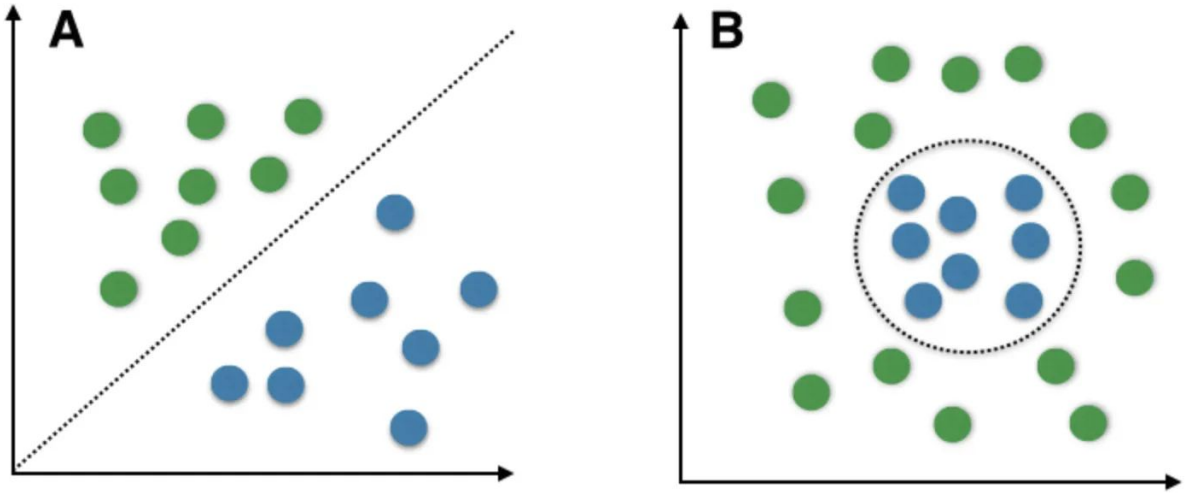




...try polynomial regression!

We can’t use linear regression for plot B! There does not exist an affine hyperplane that separates the data.

For plot B, we ideally want to model a circular boundary i.e. $x^2 + y^2 = r^2$.



“Lifting” raw data to features

To get those x^2 and y^2 terms, we need to transform our raw data. This is called “lifting”.

“Lifting” means taking our raw features (denoted as X) and applying a function to them to get a different set of features (denoted as Φ). An example of “lifting” x to a set of 2-degree polynomial features:

x	y
2	5
5	26

1	x	x^2	y
1	2	4	5
1	5	25	26

“Lifting” raw data to features

After lifting, we can run regression (or any other machine learning model) on the new set of features to get a decision boundary.

In the case for polynomial regression, we lift the raw features so that they form the set of monomials in a d-degree polynomial.

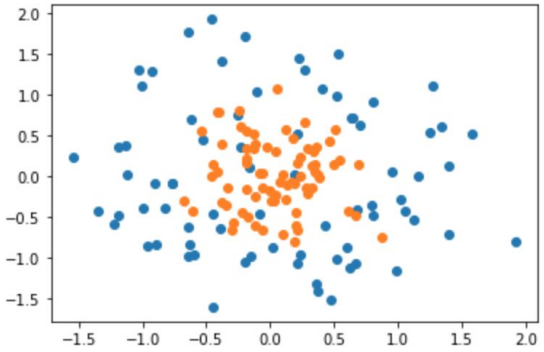
x	y
2	5
5	26

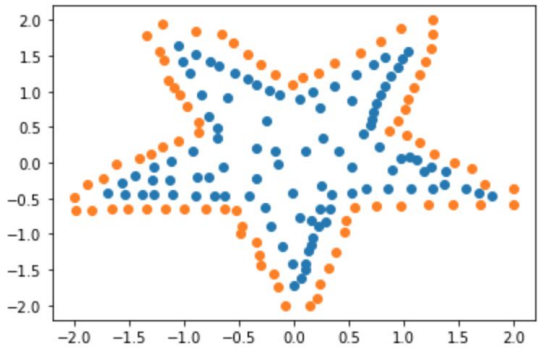
1	x	x^2	y
1	2	4	5
1	5	25	26

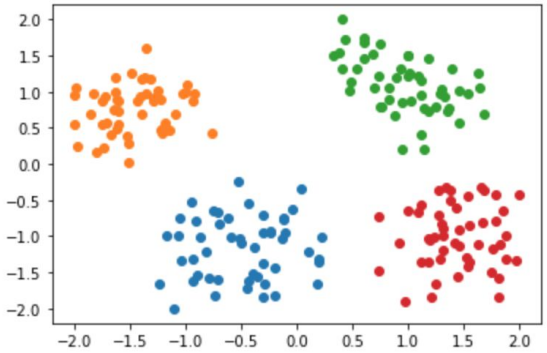
Other examples of non-linear boundaries

In the previous slides, we had a simple example with a circular boundary. In reality, the decision boundary can be very complicated. Below are some more examples of non-linear boundaries between different classes.

In these cases, we will need to lift to even higher degree polynomials.







Polynomial regression algorithm

(1) Pick a degree (d) for the polynomial

(2) Lift raw features (X) to d-degree polynomial features (Φ)

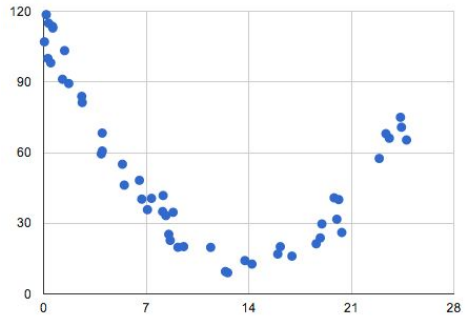
You need to create all monomials from the raw features i.e. all the powers and cross-terms

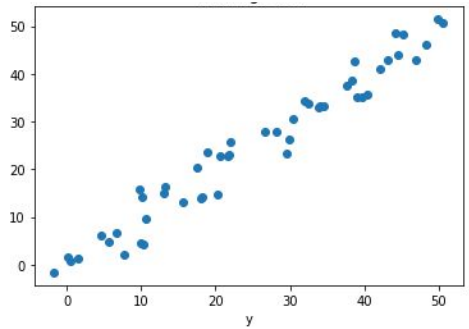
(3) Run OLS on these new features

Recall for linear regression, we want to solve $y = w^T * X$ and the closed form OLS solution is $w = (X^T @ X)^{-1} @ X^T @ y$. For polynomial, we use Φ instead of X

Quiz Questions

#1 In the previous slide we talked about the classification setting but the same holds true for the regression setting. Consider the two plots below:





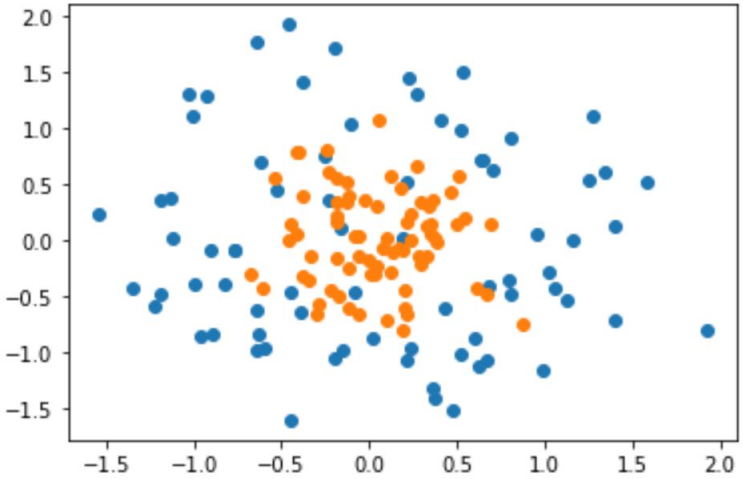
In a regression setting, would you use linear regression or polynomial regression for dataset on the left? How about the dataset on the right?

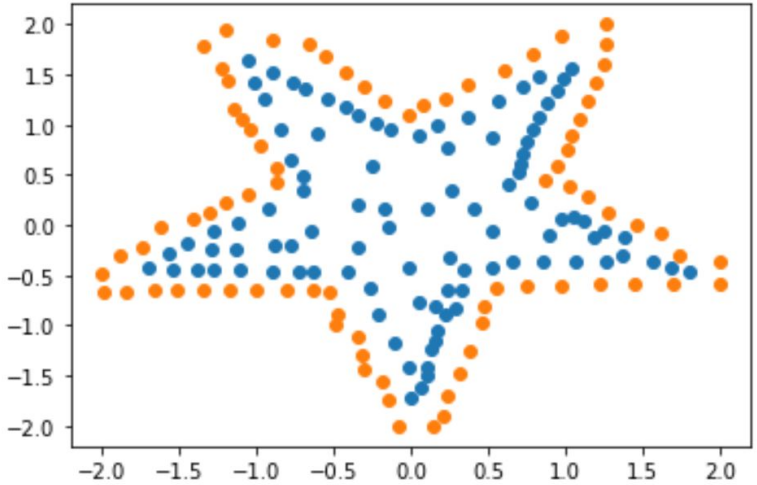
Sources: https://www.statisticshowto.com/quadratic-regression/, https://www.geeksforgeeks.org/linear-regression-using-tensorflow/

How to pick the degree?

Some cases are obvious and others are not. What do we do when it's not?

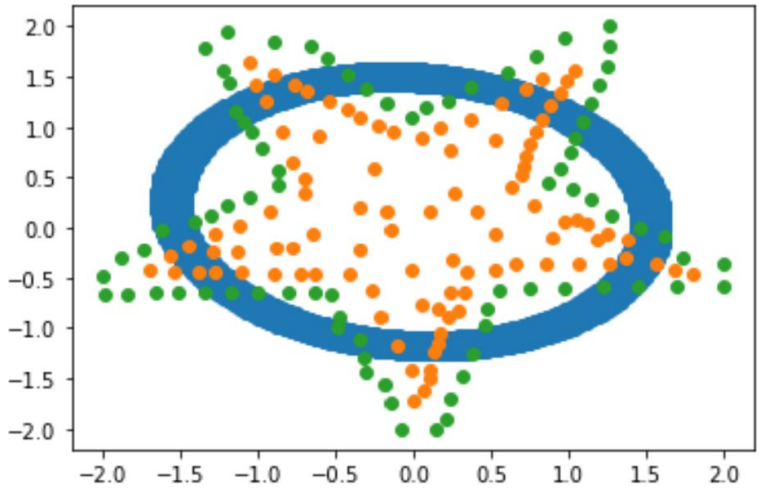
The degree is called a "hyperparameter." It is a parameter in the learning process that we need to pick and feed in. It isn't trained by the learning process.

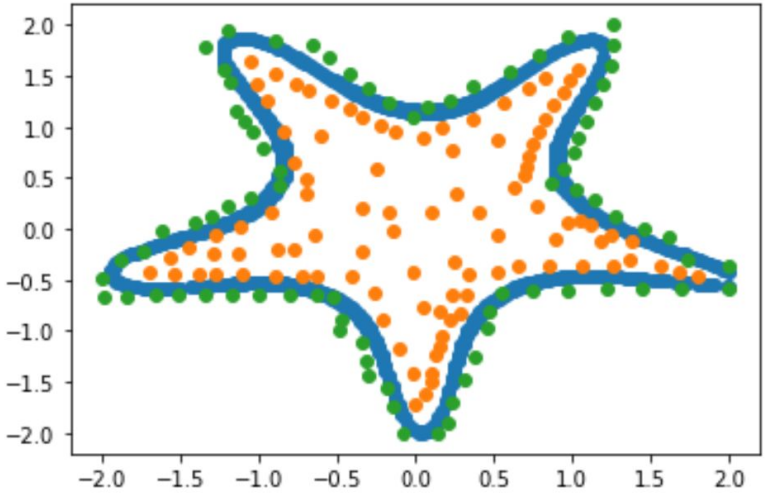




Overfitting and Underfitting

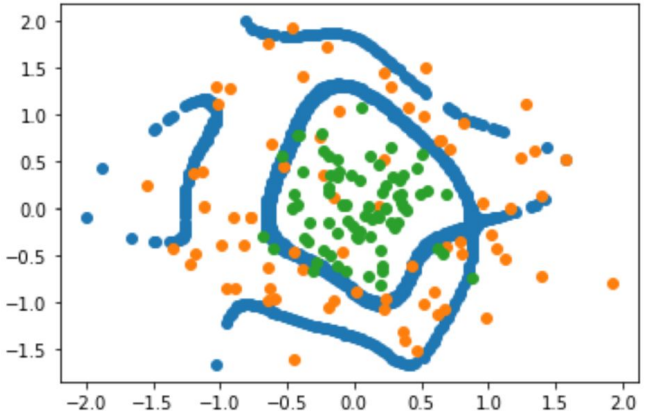
- If we pick a degree too small i.e. **don't have enough features**, our model would potentially **underfit** because it is not able to capture the details in our data.

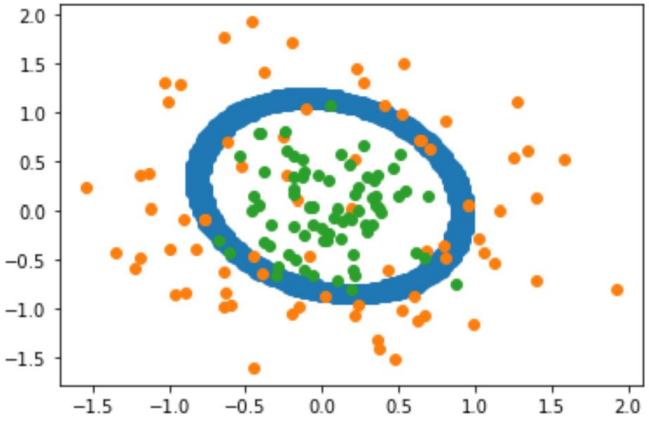




Overfitting and Underfitting

- If we pick a degree too large i.e. **have too many features**, our model would potentially **overfit** and do poorly on the test set.
- This is because the the large number of polynomials we create would fit the training data too particularly , making it not generalizable to a different set of data.





Hyperparameter tuning via validation

We use a process called “validation” to solve this:

1. Split data into train, validation, test sets. Set the test set aside.
2. Chose a range of hyperparameters to validate
3. While in range of hyperparameters:
    3.1. Pick a hyperparameter and train model using train set
    3.2. Predict using validation set and calculate validation error
4. Compare validation errors across all hyperparameters tested. Pick hyperparameter with smallest error
5. Train model using train+validation set using picked hyperparameter
6. Predict using test set and calculate error to evaluate model

Quiz Questions

#2 How should you split your data for hyperparameter tuning?

Answer: Three sets of data: train, validation, and test

#3 Which sets of data should you train your data when it comes test time?

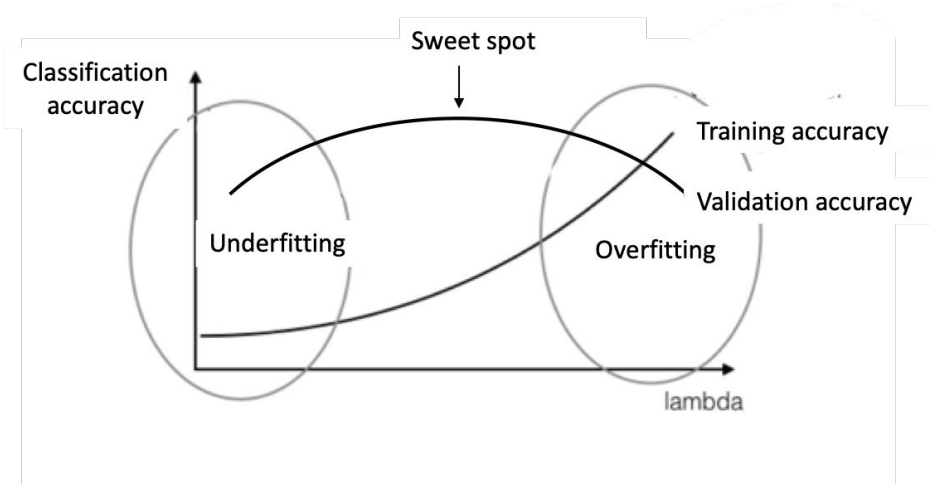
Answer: Use both train and validation sets

Why is it important to use validation error?

The error on the training set will always go down when you add more features to the model

But adding more features makes the model inflexible to handling “new” points and this is why we see validation error goes up

We want to be at the sweet spot!



#3 Quiz Question

You trained a classifier to recognize handwritten digits. When training your model, you noticed that the training accuracy is very high. However, upon validation, you noticed that your model has much lower accuracy on the validation set. Please answer the following question:

(1) Which of the following is true?
    (a) Your model is underfitting
    (b) Your model is overfitting
(2) Utilizing the validation set, what should you do to address the issue above?

Answer: (1) (b); (2) hyperparameter tuning

Bias-Variance Tradeoff

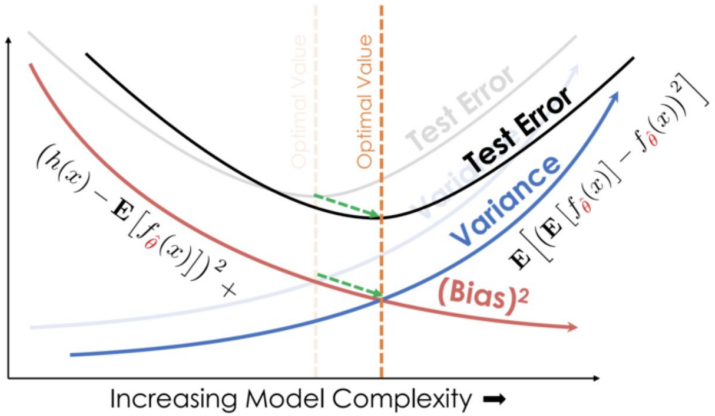
This is closely related to the idea of bias-variance tradeoff.

Bias is error caused by wrong assumptions in our learning algorithm.

Variance is error caused by the algorithm's sensitivity to noise.

Variance can be viewed as **inversely proportional** to bias.

Finding the balance where test error is at its lowest, which is also where variance and bias intersect, as seen in the image.



Source:
http://www.textbook.ds100.org/ch/15/bias_modeling.html

Bias-Variance Tradeoff

There will always be a bias-variance tradeoff when trying to fit models as they are inversely related.

Low variance algorithms tend to be less complex, with a simple or rigid underlying structure.

- Drawback: they sometimes can’t learn the signal from the data

Low bias algorithms tend to be more complex, with flexible underlying structure.

- Drawback: they tend to memorize noise in training set and bring it into the test set.

#4 Quiz Question

When doing hyperparameter tuning, if we increase the number of features for a polynomial function, what is the behavior of bias and variance?

Answer: The bias will decrease, the variance will increase.

Computation time

What happens when we try to use a very high degree polynomial?

We get a lot of features! Remember that if we want to lift n variables to a d-degree polynomial, we get n+d choose d number of monomials i.e. polynomial features

This is computationally expensive because of matrix inversion:

$$w = (\Phi^T \mathbin{@} \Phi)^{-1} \mathbin{@} \Phi^T \mathbin{@} y$$

The kernel trick can improve computation speed

You should be familiar with kernels from the previous lesson and this is the perfect application! The polynomial kernel for a d-degree polynomial is:

$$K(x,y)=(x^Ty+c)^d$$

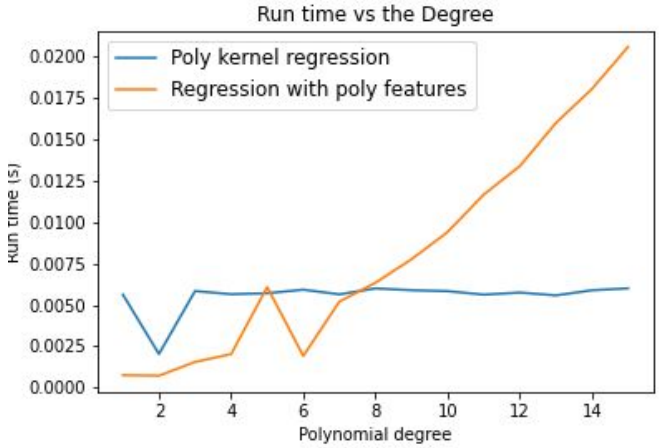
Example: Let's consider $(x+y)^3$. If we compute this expression directly, we only need one addition and two multiplications. However, if we choose expand it, The expression becomes $x^3+3x^2y+3xy^2+y^3$, which is far more computationally expensive.

Takeaway: Using a polynomial kernel relies on the raw feature space size not the lifted feature space size i.e. reduced computation time for large degree polynomial

One exception...

While kernel trick will help when we have a lot of features, it doesn't necessarily help when we have few features.

This is because calculating the gram matrix K(x,y) is a fixed cost dependent on the size of our dataset (since we are computing the inner products between raw data points)



#5 Quiz Questions

- When using the kernel trick, can one get non-linear decision boundaries? Try to argue why.

  Answer: True. Because the result of some kernel functions is effectively equal to the sum of each term in a higher degree polynomial.

- Comparing using polynomial features versus a polynomial kernel for polynomial regression, which method would be computationally faster when the number of features is a lot larger than the number of data points?

  Answer: Polynomial kernel