ECEN 377: Engineering Applications of AI

Dr. Mahmoud Nabil Mahmoud
mnmahmoud@ncat.edu

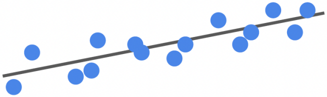
North Carolina A & T State University

September 25, 2024

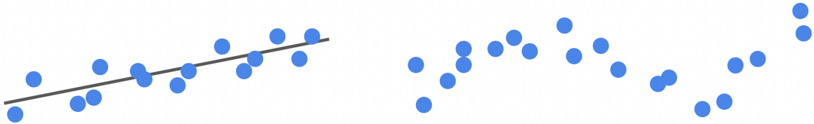
Outline

Polynomial Regression

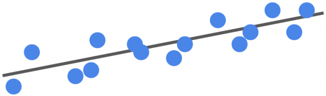
Polynomial Regression

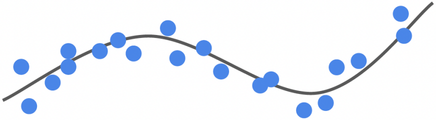


Polynomial Regression



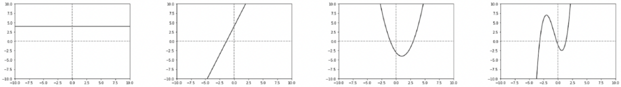
Polynomial Regression





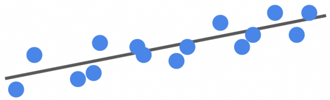
Examples of polynomials:

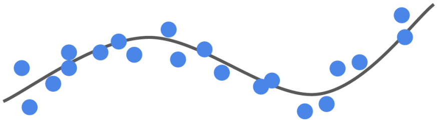
- $y = 4$
- $y = 3x + 2$
- $y = x^2 - 2x + 5$
- $y = 2x^3 + 8x^2 - 40$



degree 0 $y = 4$ | degree 1 $y = 3x + 4$ | degree 2 $y = x^2 - 2x - 3$ | degree 3 $y = x^3 + 2x^2 - 4x - 1$

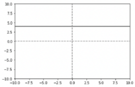
Polynomial Regression



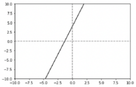


Examples of polynomials:

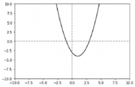
- $y = 4$
- $y = 3x + 2$
- $y = x^2 - 2x + 5$
- $y = 2x^3 + 8x^2 - 40$



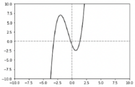
degree 0
$y = 4$



degree 1
$y = 3x + 4$



degree 2
$y = x^2 - 2x - 3$



degree 3
$y = x^3 + 2x^2 - 4x - 1$

Note: What degree of polynomial needed?

From Linear to non-Linear (Feature Crosses)

- Our basic equation for linear regression with one feature is:

$$y' = w_1 x_1 + w_0$$

From Linear to non-Linear (Feature Crosses)

- Our basic equation for linear regression with one feature is:

$$y' = w_1 x_1 + w_0$$

- To fit a quadratic relationship, we can use:

$$y' = w_2 x_1^2 + w_1 x_1 + w_0$$

From Linear to non-Linear (Feature Crosses)

- Our basic equation for linear regression with one feature is:

$$y' = w_1 x_1 + w_0$$

- To fit a quadratic relationship, we can use:

$$y' = w_2 x_1^2 + w_1 x_1 + w_0$$

- This technique of adding polynomial terms is called feature crosses.

From Linear to non-Linear (Feature Crosses)

- Our basic equation for linear regression with one feature is:

$$y' = w_1 x_1 + w_0$$

- To fit a quadratic relationship, we can use:

$$y' = w_2 x_1^2 + w_1 x_1 + w_0$$

- This technique of adding polynomial terms is called feature crosses.
- By adding higher-degree terms, we can increase the model complexity:

$$y' = w_n x_1^n + w_{n-1} x_1^{n-1} + \cdots + w_1 x_1 + w_0$$

From Linear to non-Linear (Feature Crosses)

- Our basic equation for linear regression with one feature is:

$$y' = w_1 x_1 + w_0$$

- To fit a quadratic relationship, we can use:

$$y' = w_2 x_1^2 + w_1 x_1 + w_0$$

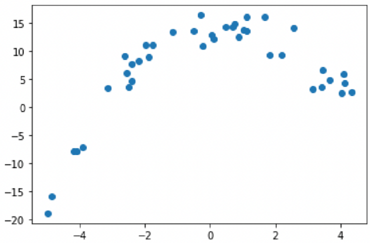
- This technique of adding polynomial terms is called feature crosses.
- By adding higher-degree terms, we can increase the model complexity:

$$y' = w_n x_1^n + w_{n-1} x_1^{n-1} + \cdots + w_1 x_1 + w_0$$

- The degree of the polynomial determines the flexibility of the model.

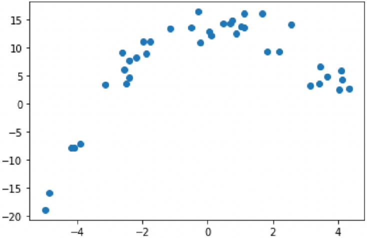
Feature Crosses Example

x	y
3.4442185152504816	6.685961311021467
-2.4108324970703663	4.690236225597948
0.11274721368608542	12.205789026637378
-1.9668727392107255	11.133217991032268



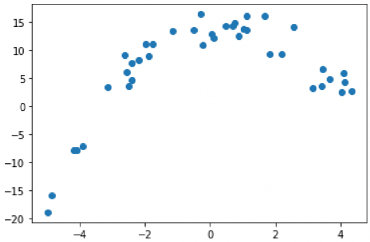
Feature Crosses Example

x	y	x^2	x^3	x^4
3.4442185152504816	6.685961311021467	11.862641180794233	40.85752839466433	140.72225578427518
-2.4108324970703663	4.690236225597948	5.812113328930538	-14.012031690041567	33.78066134833202
0.11274721368608542	12.205789026637378	0.012711934193975809	0.0014332351609316464	0.00016159327095197139
-1.9668727392107255	11.133217991032268	3.8685883722503025	-7.609021008606714	14.965975993910245



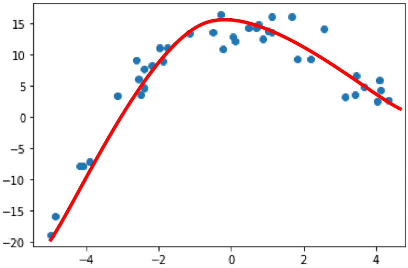
Feature Crosses Example

x	y	x^2	x^3	x^4
3.4442185152504816	6.685961311021467	11.862641180794233	40.85752839466433	140.72225578427518
-2.4108324970703663	4.690236225597948	5.812113328930538	-14.012031690041567	33.78066134833202
0.11274721368608542	12.205789026637378	0.012711934193975809	0.0014332351609316464	0.00016159327095197139
-1.9668727392107255	11.133217991032268	3.8685883722503025	-7.609021008606714	14.965975993910245



Feature Crosses Example

x	y	x^2	x^3	x^4
3.4442185152504816	6.685961311021467	11.862641180794233	40.85752839466433	140.72225578427518
-2.4108324970703663	4.690236225597948	5.812113328930538	-14.012031690041567	33.78066134833202
0.11274721368608542	12.205789026637378	0.012711934193975809	0.0014332351609316464	0.00016159327095197139
-1.9668727392107255	11.133217991032268	3.8685883722503025	-7.609021008606714	14.965975993910245



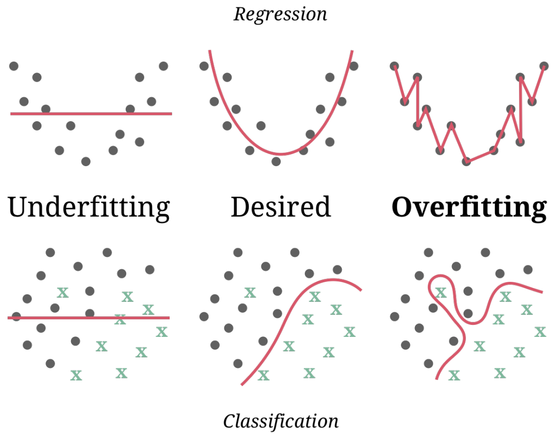
Outline

Underfitting and Overfitting problems

Which polynomial degree is suitable for this data? In the previous example, we saw that the model with $d = 2$ is a good fit for the data.

How do machines know that?!!

Overfitting and Underfitting



- Wait, how can model 2 have a larger error than model 3, yet still be better for our data?!
- How we supposed to figure out if our model is overfitting or underfitting?

Overfitting and Underfitting

Note

A good model (best fit) should be able to generalize to new (unseen) data. How?

- **Over-fitting:**
  - Model too complex (flexible)
  - Fits "noise" in the training data
  - High error is expected on the test data.
- **Under-fitting:**
  - Model too simplistic (too rigid)
  - Not powerful enough to capture salient patterns in training data and test data.

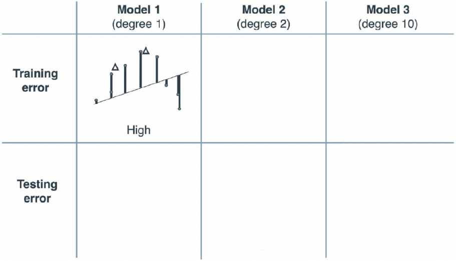
Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
  - Training set
  - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that

	Model 1 (degree 1)	Model 2 (degree 2)	Model 3 (degree 10)
Training error			
Testing error			

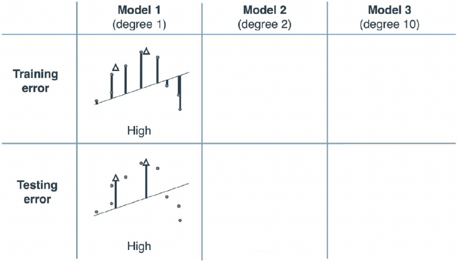
Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
  - Training set
  - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that



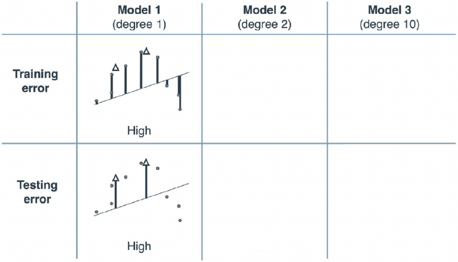
Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
  - Training set
  - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that



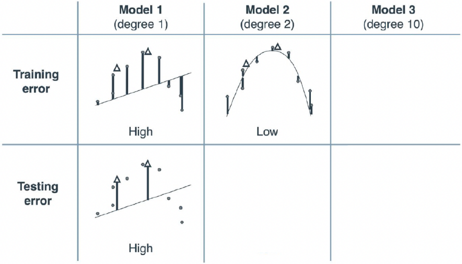
Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
    - Training set
    - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that



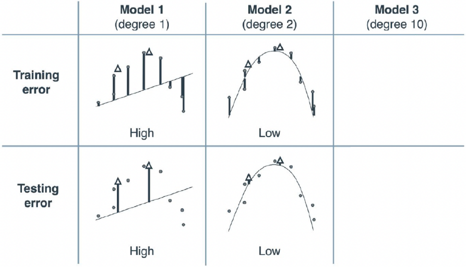
Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
  - Training set
  - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that



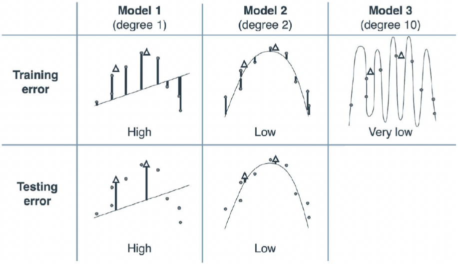
Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
  - Training set
  - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that



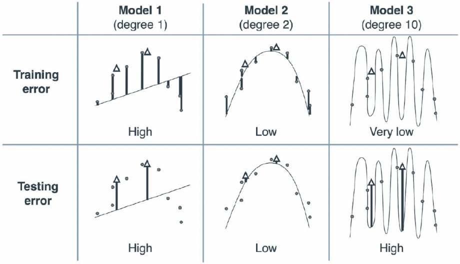
Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
  - Training set
  - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that



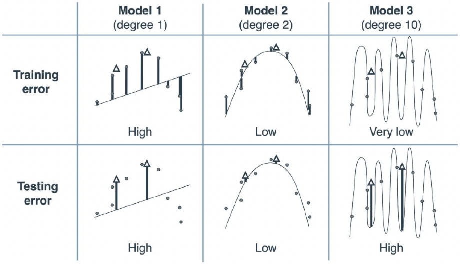
Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
    - Training set
    - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that



Catching Overfitting/Underfitting Test Data Solution

- We divide the data into 2 sets:
  - Training set
  - Testing set
- We can know if there is underfit or overfit problem from train and test errors
- We can tune our hyper-parameters based on that



Best Model: Lowest error on the test data

Outline

Can we still overfit while using the test data?

- Yes, we can still overfit even when using test data

Can we still overfit while using the test data?

- Yes, we can still overfit even when using test data
- This happens when we use the test set too many times to tune hyperparameters of our model (e.g., degree of polynomial, learning rate, training epochs, etc.)

Can we still overfit while using the test data?

- Yes, we can still overfit even when using test data
- This happens when we use the test set too many times to tune hyperparameters of our model (e.g., degree of polynomial, learning rate, training epochs, etc.)
- The test set becomes part of the training process We lose the ability to generalize

Can we still overfit while using the test data?

- Yes, we can still overfit even when using test data
- This happens when we use the test set too many times to tune hyperparameters of our model (e.g., degree of polynomial, learning rate, training epochs, etc.)
- The test set becomes part of the training process We lose the ability to generalize
- **Solution:** Use a validation set

Can we still overfit while using the test data?

- Yes, we can still overfit even when using test data
- This happens when we use the test set too many times to tune hyperparameters of our model (e.g., degree of polynomial, learning rate, training epochs, etc.)
- The test set becomes part of the training process We lose the ability to generalize
- **Solution:** Use a validation set
  - Train set: Used to train the model

Can we still overfit while using the test data?

- Yes, we can still overfit even when using test data
- This happens when we use the test set too many times to tune hyperparameters of our model (e.g., degree of polynomial, learning rate, training epochs, etc.)
- The test set becomes part of the training process We lose the ability to generalize
- **Solution:** Use a validation set
  - Train set: Used to train the model
  - Validation set: Used to tune hyperparameters

Can we still overfit while using the test data?

- Yes, we can still overfit even when using test data
- This happens when we use the test set too many times to tune hyperparameters of our model (e.g., degree of polynomial, learning rate, training epochs, etc.)
- The test set becomes part of the training process We lose the ability to generalize
- **Solution:** Use a validation set
  - Train set: Used to train the model
  - Validation set: Used to tune hyperparameters
  - Test set: Used **once** only for final evaluation

Can we still overfit while using the test data?

- Yes, we can still overfit even when using test data
- This happens when we use the test set too many times to tune hyperparameters of our model (e.g., degree of polynomial, learning rate, training epochs, etc.)
- The test set becomes part of the training process We lose the ability to generalize
- **Solution:** Use a validation set
  - Train set: Used to train the model
  - Validation set: Used to tune hyperparameters
  - Test set: Used **once** only for final evaluation
- This three-way split helps prevent overfitting on the test data

Outline

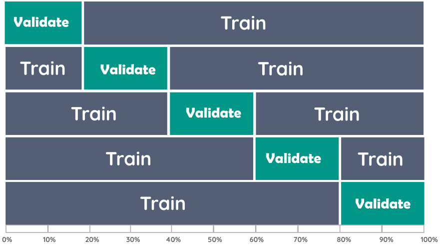
K-Cross Validation

Why?

- We can be exposed to the test set only once.
- We need to estimate future error as accurately as possible.

Ex.

- Randomly split the training into k sets.
- Validate on one in each turn (train on 4 others)
- Average the results over 5 folds



5-fold cross validation

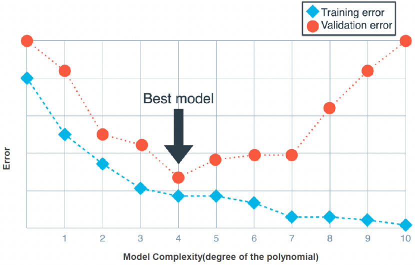
Underfitting and Overfitting problems [**The Validation set solution**]

- This is called **Simple (Holdout) Cross Validation**
- Note: We can use more sophisticated cross-validation techniques for better model evaluation.

Outline

Model Complexity

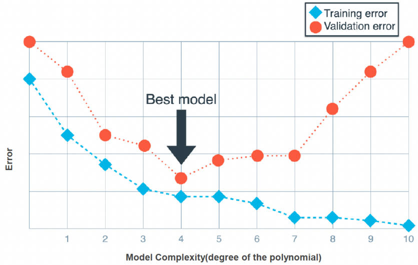
- As we increase the degree of the polynomial, the model becomes more complex.
- A complex model can fit the training data very well, but it may not generalize well to new, unseen data.
- This is where the concept of **model complexity** comes into play.



Outline

Solving overfitting/underfitting problem **[Early Stopping]**

- Early stopping is a technique to prevent overfitting
- Monitor the model's performance on a validation set during training
- Training is stopped when the validation error starts to increase
- This helps find the optimal point between underfitting and overfitting



Outline

Understanding Validation Error

- **Why is validation error usually larger than training error?**

Understanding Validation Error

- **Why is validation error usually larger than training error?**
  - The model is optimized on the training data

Understanding Validation Error

- **Why is validation error usually larger than training error?**
  - The model is optimized on the training data
  - Validation data is unseen, so performance is typically worse

Understanding Validation Error

- **Why is validation error usually larger than training error?**
  - The model is optimized on the training data
  - Validation data is unseen, so performance is typically worse
  - This difference helps assess the model's generalization ability

Understanding Validation Error

- **Why is validation error usually larger than training error?**
  - The model is optimized on the training data
  - Validation data is unseen, so performance is typically worse
  - This difference helps assess the model's generalization ability

Understanding Validation Error

- **Why is validation error usually larger than training error?**
  - The model is optimized on the training data
  - Validation data is unseen, so performance is typically worse
  - This difference helps assess the model's generalization ability

- **Should we always pick the model with the least validation error?**

Understanding Validation Error

- **Why is validation error usually larger than training error?**
    - The model is optimized on the training data
    - Validation data is unseen, so performance is typically worse
    - This difference helps assess the model's generalization ability
- **Should we always pick the model with the least validation error?**
    - Not necessarily - consider the following:

Understanding Validation Error

- **Why is validation error usually larger than training error?**
  - The model is optimized on the training data
  - Validation data is unseen, so performance is typically worse
  - This difference helps assess the model's generalization ability
- **Should we always pick the model with the least validation error?**
  - Not necessarily - consider the following:
  - Balance between performance and complexity

Understanding Validation Error

- **Why is validation error usually larger than training error?**
  - The model is optimized on the training data
  - Validation data is unseen, so performance is typically worse
  - This difference helps assess the model's generalization ability
- **Should we always pick the model with the least validation error?**
  - Not necessarily - consider the following:
  - Balance between performance and complexity
  - Practical considerations (e.g., computational resources)

Understanding Validation Error

- **Why is validation error usually larger than training error?**
  - The model is optimized on the training data
  - Validation data is unseen, so performance is typically worse
  - This difference helps assess the model's generalization ability
- **Should we always pick the model with the least validation error?**
  - Not necessarily - consider the following:
  - Balance between performance and complexity
  - Practical considerations (e.g., computational resources)
  - Sometimes a simpler model with slightly higher error is preferable

Outline

Solving overfitting/underfitting problem **[Regularization]**

- Higher coefficient values (weights) ⇒ Higher complexity
- Multiple regression example:

$$y = w_0 + w_1 x^1 + w_2 x^2 + w_3 x^3 + \cdots + w_n x^n$$

- Larger w_i values indicate more complex model
- Regularization aims to keep these coefficients **small**
- This helps prevent overfitting by **reducing model complexity**

Regularization: Example (L1 and L2)

- L1 Regularization (Lasso) - Encourages sparsity (encourages some weights to be zero):

$$L(\mathbf{W}) = \underbrace{(y' - y)^2}_{\text{Old Loss term}} + \underbrace{\lambda \sum |w_i|}_{\text{L1 regularization term}}$$

- L2 Regularization (Ridge) - Shrinks coefficients:

$$L(\mathbf{W}) = \underbrace{(y' - y)^2}_{\text{Old Loss term}} + \underbrace{\lambda \sum w_i^2}_{\text{L2 regularization term}}$$

- Note: λ is a hyperparameter that controls the strength of regularization

Regularization: Numeric Examples

- Consider the following models:
  - Model 1: $y = 2x$
  - Model 2: $y = x + 6$
  - Model 3: $y = x + 4x^2 + 9x^3 + 3x^4 + 14x^5 + 2x^6 + 9x^7 + x^8 + 6x^9$
- L1 Norm (sum of absolute values of coefficients):
  - Model 1: $|2| = 2$
  - Model 2: $|1| + |6| = 7$
  - Model 3: $|1| + |4| + |9| + |3| + |14| + |2| + |9| + |1| + |6| = 49$
- L2 Norm (square root of sum of squared coefficients):
  - Model 1: $2^2 = 4$
  - Model 2: $1^2 + 6^2 = 37$
  - Model 3: $1^2 + 4^2 + 9^2 + 3^2 + 14^2 + 2^2 + 9^2 + 1^2 + 6^2 = 425$

Regularization: How sparsity and shrinkage?

- L1 regularization gradient:

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\underbrace{(y' - y)^2}_{\text{L2 Error}} + \underbrace{\lambda \sum |w_i|}_{\text{L1 regularization}}\right)$$

Regularization: How sparsity and shrinkage?

- L1 regularization gradient:

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\underbrace{(y' - y)^2}_{\text{L2 Error}} + \underbrace{\lambda \sum |w_i|}_{\text{L1 regularization}}\right)$$

$$\textbf{New Update Rule} \quad \Rightarrow \quad w_i = w_i - \eta \cdot [2x_i(y' - y) + \underbrace{\lambda \cdot \text{sign}(w_i)}_{\text{Constant term}}]$$

Regularization: How sparsity and shrinkage?

- L1 regularization gradient:

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\underbrace{(y' - y)^2}_{\text{L2 Error}} + \underbrace{\lambda \sum |w_i|}_{\text{L1 regularization}}\right)$$

New Update Rule $\Rightarrow$ $w_i = w_i - \eta \cdot [2x_i(y' - y) + \underbrace{\lambda \cdot \text{sign}(w_i)]}_{\text{Constant term}}$

- L2 regularization gradient:

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\underbrace{(y' - y)^2}_{\text{L2 Error}} + \underbrace{\lambda \sum w_i^2}_{\text{L2 regularization}}\right)$$

Regularization: How sparsity and shrinkage?

- L1 regularization gradient:

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\underbrace{(y'-y)^2}_{\text{L2 Error}} + \underbrace{\lambda \sum |w_i|}_{\text{L1 regularization}}\right)$$

New Update Rule $\Rightarrow$ $w_i = w_i - \eta \cdot [2x_i(y'-y) + \underbrace{\lambda \cdot \text{sign}(w_i)}_{\text{Constant term}}]$

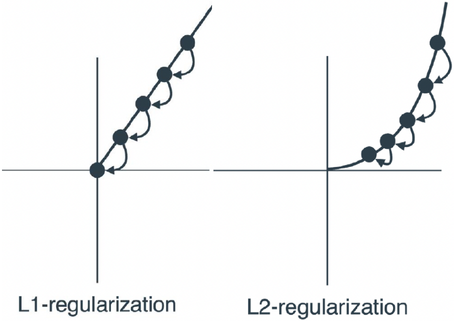
- L2 regularization gradient:

$$\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i}\left(\underbrace{(y'-y)^2}_{\text{L2 Error}} + \underbrace{\lambda \sum w_i^2}_{\text{L2 regularization}}\right)$$

New Update Rule $\Rightarrow$ $w_i = w_i - \eta \cdot [2x_i(y'-y) + \underbrace{2\lambda \cdot w_i}_{\text{Ratio of weight to its value}}]$

Regularization: How sparsity and shrinkage?

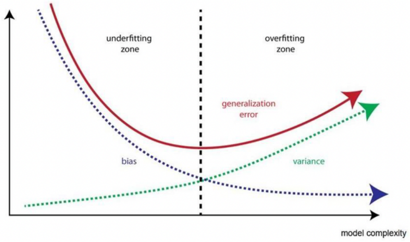
- L1: The constant $\lambda \cdot \text{sign}(w_i)$ term pushes small weights to **exactly zero** (**sparsity**)
- L2: The $2\lambda w_i$ term **shrinks weights proportionally** to their magnitude (**shrinkage**)



Outline

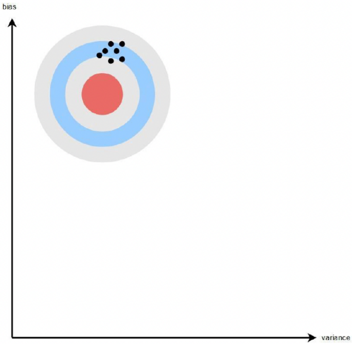
Variance-Bias Tradeoff

- **Variance:** How much the model's predictions vary with different type of data.
  - Overfit model: High Variance model
- **Bias:** How much the model's predictions deviate from the true value.
  - Underfit model: High Bias model
- **Tradeoff:** Lower bias often results in higher variance, and vice versa.



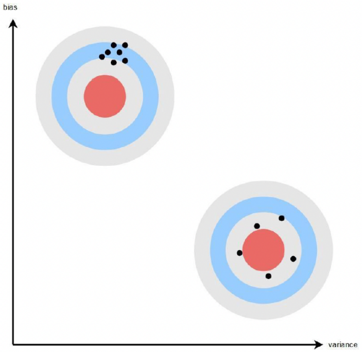
Low Variance and High Bias

- **Characteristics:**
  - Underfits the data
  - Poor performance on both training and test sets
  - Example: Linear model for complex, non-linear data



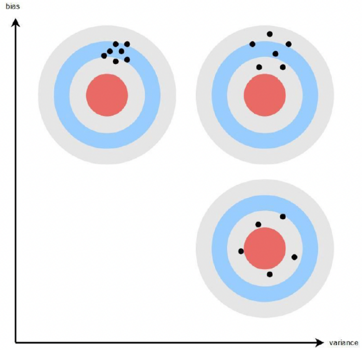
High Variance and Low Bias

- **Characteristics:**
  - Overfits the data
  - Excellent performance on training set, poor on test set
  - Sensitive to small fluctuations in the training data
  - Example: High-degree polynomial for simple, nearly linear data



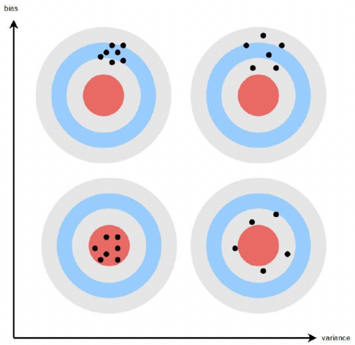
High Variance and High Bias

- **Characteristics:**
  - Poor performance on both training and test sets
  - Example: Linear model for complex, non-linear data



Low Variance and Low Bias

- **Characteristics:**
  - Good performance on both training and test sets
  - Balances between underfitting and overfitting
  - Generalizes well to new, unseen data
  - Example: Appropriate complexity model for the given data



Factors Affecting Bias and Variance

Factors contributing to high bias:

- Model simplicity
- Insufficient features
- Incorrect assumptions
- Limited training data

Factors contributing to high variance:

- Model complexity
- Too many features
- Small training set
- High sensitivity to training data

Exercise

We have trained four models in the same dataset with different hyperparameters. In the following table, we have recorded the training and testing errors for each of the models.

Model	Training Error	Testing Error
1	0.1	1.8
2	0.4	1.2
3	0.6	0.8
4	1.9	2.3

Questions:

(a) Which model would you select for this dataset?

(b) Which model looks like it's underfitting the data?

Questions