DATA SOCIETY:

Model Performance And Fit - 4

One should look for what is and not what he thinks should be. (Albert Einstein)

Module completion checklist

Objective	Complete
Compare methods to assess fit of a neural network	
Discuss methods to improve the fit of a neural network	

What is fit?

- Model fitting measures how well a machine learning model generalizes to data similar to that on which it was trained
- Fit is important because a well-fit model produces more accurate outcomes
    - **Overfitting** is when the model performs great on training data but not as well when fitting the model to test data
    - **Underfitting** is when the model can neither perform well on training or test data

What causes poor fit?

- **Bias**
  - Assumptions can prevent us from learning true relationships between features and output

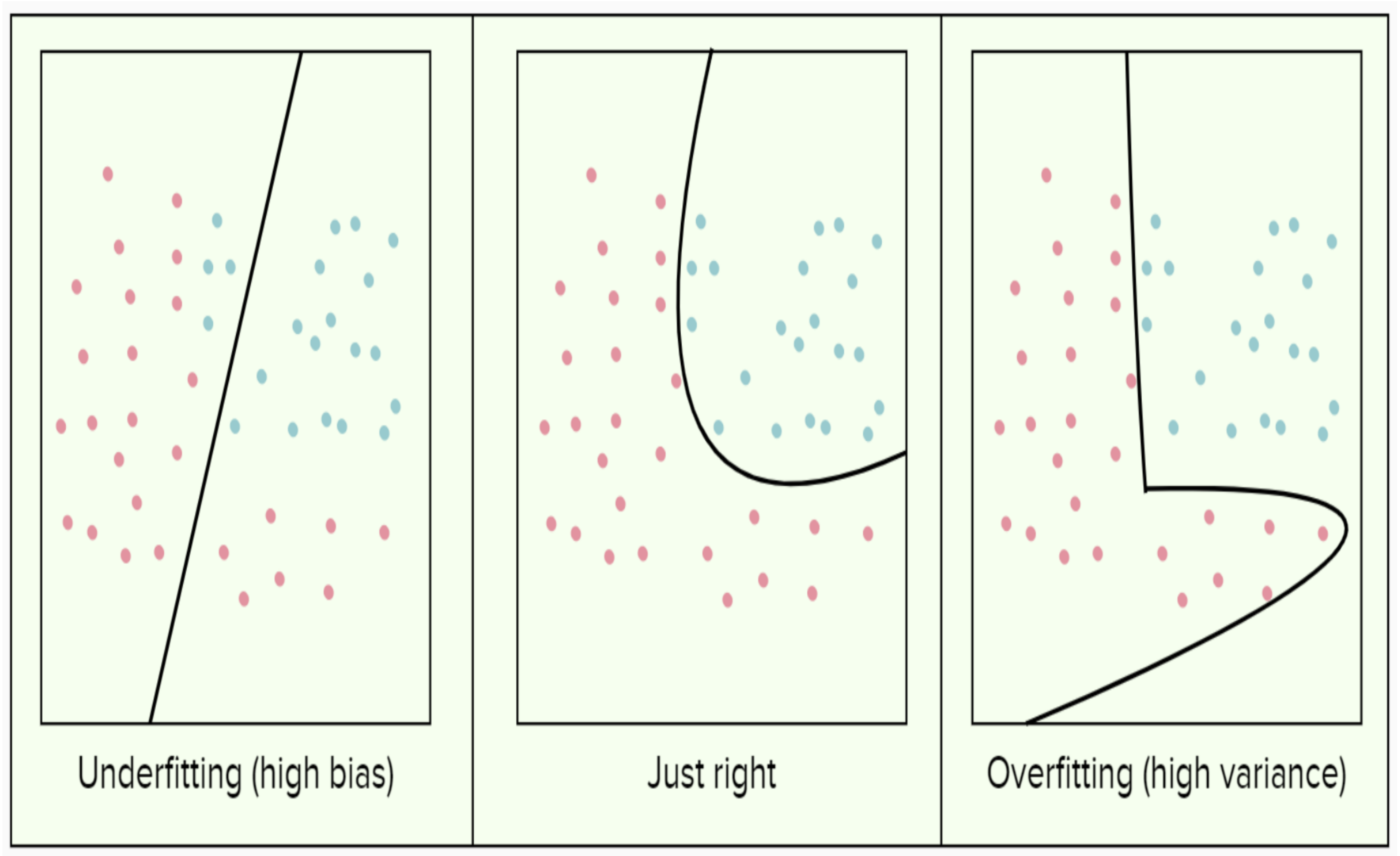
- **Variance**
  - Sensitivity to intricate details of training set can lead to modeling noise in data

- **Irreducible error**
  - Noise in the problem itself

Bias-variance tradeoff

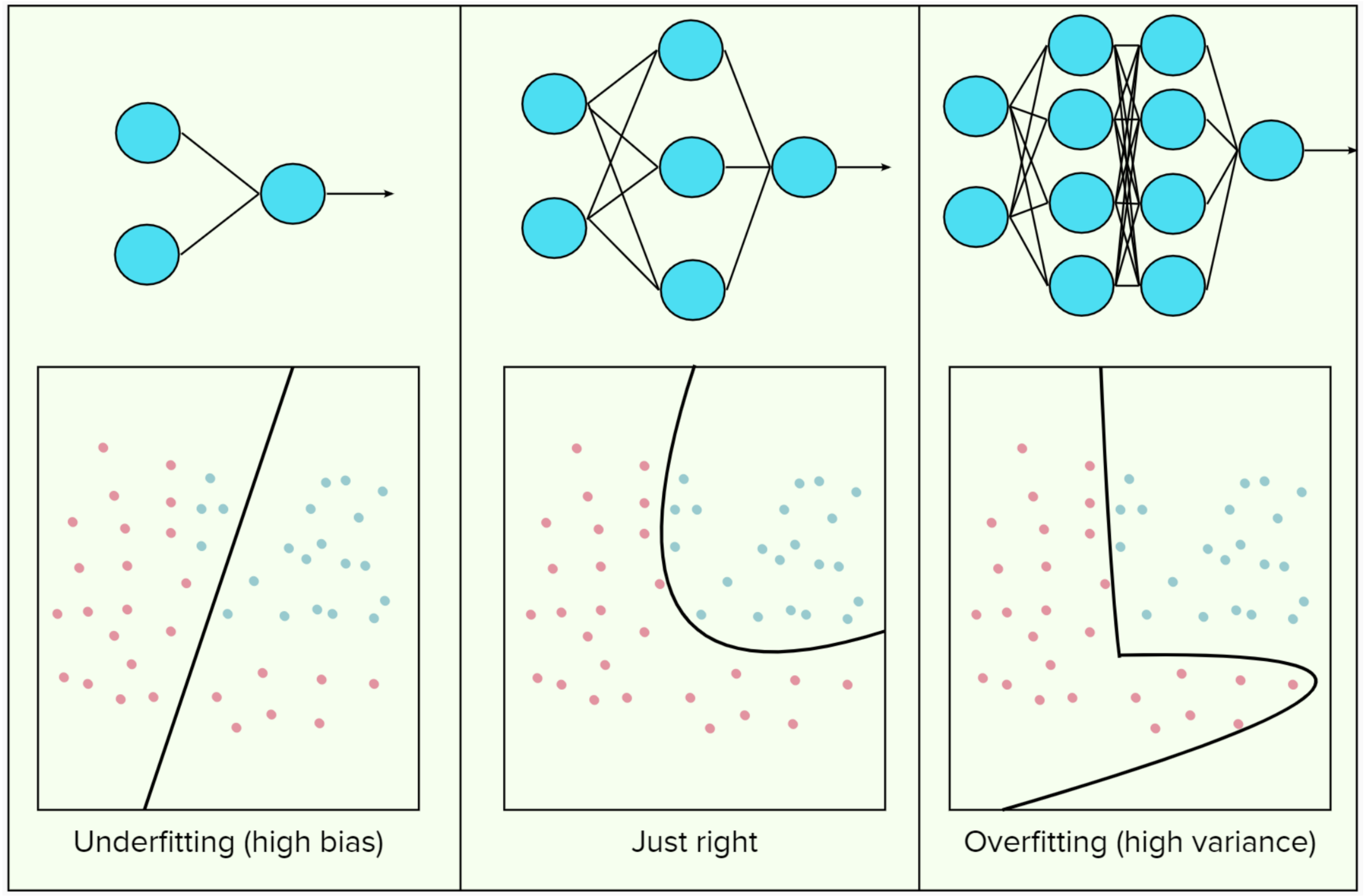
- Models that perform well at capturing the intricate features of the training set but may easily overfit are called **high-variance models**

- Models that assume away essential data features, usually leading to underfitting, are called **high-bias models**



Overfitting in neural networks

- Neural networks with more hidden layers and neurons will generally give better results
- However, since neural networks are such good learners, they may capture the intricate details of the training data too well
- Neural networks can fail at generalizing to unseen data, which leads to overfitting



Module completion checklist

Objective	Complete
Compare methods to assess fit of a neural network	✔
Methods to improve the fit of a neural network	

How to combat overfitting?

- Neural networks have a host of different methods to prevent overfitting
- There are no specific reasons to choose a particular method over the other, and you may choose to use more than one
- We will cover the most common ones:
    - regularization
    - early stopping
    - dropout
    - weight constraints

Regularization

- **Regularization** is a crucial method for preventing overfitting in neural networks
- If you have done *logistic regression*,you may already be familiar with its parameters
- In short, regularization **softens the classifier margins** and lets in some misclassified observations for generalization

Regularization techniques

- Any machine learning algorithm that optimizes some *cost function* $f(x)$
- `l1` (*Lasso*) adds a term to that function like so:

$$f(x) + C\sum_{j=1}^{n} |b_j|$$

- While `l2` (*Ridge*) adds a term like so:

$$f(x) + C\sum_{j=1}^{n} b_j^2$$

- You can see that *Lasso* uses the absolute value - $|b_j|$, while *Ridge* uses a squared value - b_j^2
- That term, when added to the original *cost function*, **dampens the margins** of our classifier, making it more **forgiving** of the misclassifying some points that might be noise

Lasso vs. Ridge

Lasso (l1)

$$C\sum_{j=1}^{n}|b_j|$$

- Stands for **L**east **A**bsolute **S**hrinkage and **S**election **O**perator
- It adds the `absolute value of magnitude` of the coefficient as a penalty term to the loss function
- Shrinks (as the name suggests) the less important features' coefficients to zero, which leads to **removal** of some features

Ridge (l2)

$$C\sum_{j=1}^{n}b_j^2$$

- Adds `squared magnitude` of coefficient as penalty term to the loss function
- Dampens the less important features' coefficients making them less significant, which leads to **weighting** of the features according to their importance

What is the role of C?

There are 4 scenarios that might happen with a classifier with respect to C:

- $C = 0$
  - The classifier becomes an **OLS** problem (i.e., Ordinary Least Squares, or just a strict regression without any penalization)
  - Since $0 \times anything = 0$, we are just left with optimizing $f(x)$, which is a definite overfitting problem
- $C = small$
  - We still run into an overfitting problem
  - Since C will not “magnify” the effect of the penalty term enough

What is the role of C? (cont'd)

- $C = large$
  - We run into an **underfitting** problem, where we've weighted and dampened the coefficients too much and we made the model too general
- $C = optimal$
  - We have a **good, robust, and generalizable model** that works well with new data
  - Ignores most of the noise while preserving the main pattern in data

- To pick the right combination of parameters, we need to tune our model to find the right combination of those parameters

Early stopping

- **Early stopping** is another method to prevent overfitting
- It automatically terminates training when the monitored metric is not improving by some value for a certain number of iterations (i.e., `n_iter_no_change` in TensorFlow)
- The value is known as **tolerance for the optimization** (i.e., `tol` in TensorFlow)

- For more information, visit the TensorFlow website: ***Link***

Dropout

- **Dropout** is a common technique to address overfitting
- Theoretically, dropout **randomly drops neurons** *(along with their connections)* from the neural network during training
- It prevents neurons in layers from co-adapting too much
- It takes 2 arguments:
  - `rate`: a proportion of neurons to be dropped (`~0.1-0.4`)
  - `seed`: “locks” the random number generator for reproduceable our results
- For more information, visit the TensorFlow website: ***Link***

Weight constraint

- Weight constraint is another common technique to avoid overfitting
- It is an update to the neural net that checks the magnitude of the weights
    - If the size exceeds the predefined limit, the weights are rescaled so that the size is within the range
- Unlike adding a penalty to the loss function, weight constraint ensures the weights of the network are small
- For more information, visit the TensorFlow website: ***Link***

Early stopping strategies

Technique	Strategy
Early stopping	Always expect during hyper parameter optimization
Early stopping + dropout	Small training data or large network
Early stopping + weight decay	Large network
Early stopping + weight constraint	Large network + large learning rate

Knowledge check

Module completion checklist

Objective	Complete
Compare methods to assess fit of a neural network	✔
Discuss methods to improve the fit of a neural network	✔

Model Performance and Fit: Topic summary

In this part of the course, we have covered:

- Implement a custom neural network to demonstrate model fit with different learning rates, epochs and batch sizes
- Understand loss functions and math behind gradient descent
- Assess and discuss methods to improve the fit of a neural network

Congratulations on completing this module!