Avoiding Overfitting

CMPUT 261: Introduction to Artificial Intelligence

P&M §7.4

Logistics & Assignment #2

- **Assignment #2 is** due **today** at **11:59pm**
  - Submit via eclass
  - Late submission deadline **Monday, February 20** at 11:59pm
  - 20% deduction for late submissions
- Next week is **reading week**
  - No classes
- **Midterm** is Thursday March 2
  - Covers everything up to and including Bayesian inference

Recap: Supervised Learning

Definition: A **supervised learning task** consists of

- A set of **input features** $X_1, \ldots, X_n$
- A set of **target features** $Y_1, \ldots, Y_k$
- A set of **training examples**, for which both input and target features are given
- A set of **test examples**, for which only the input features are given

The goal is to **predict** the values of the **target features** given the **input features**; i.e., **learn** a function $h(x)$ that will map features X to a prediction of Y

- We want to predict **new, unseen data** well; this is called **generalization**
- Can **estimate generalization** performance by reserving separate **test examples**

Lecture Outline

1. Recap & Logistics
2. Causes of Overfitting
3. Avoiding Overfitting

After this lecture, you should be able to:

- define overfitting, bias, and noise
- explain how to avoid overfitting using pseudocounts, regularization, and cross-validation

Overfitting

Overfitting: The learner makes predictions based on regularities that occur in the **training data** but **not** in the **underlying population,** causing failure to **generalize**

1. Learning **spurious correlations**: In any training data there may be coincidental associations that are not reflective of the process being learned
   - *Example:* More pictures of tanks taken on sunny days, more pictures without tanks taken on cloudy days. Learning agent learns that sunny pictures are predictive of tanks.
2. **Overconfidence** in the learned model. The unseen data is assumed to be more **exactly like** the training data than is plausible.
   - *Example:* Just because my training data doesn't contain the word "squeegee" doesn't mean there is a literally **zero percent** chance of encountering it!

Example: Restaurant Ratings

- Suppose a website collects **ratings for restaurants** on a scale of 1 to 5 stars
- The website wants to display the **best** restaurants
  - *Definition:* Restaurants that future diners will like most
  - I.e., based on **observations** (ratings from past diners), predict "true" **rating** (average ratings from the population of diners)
- **Question:** What rating **prediction** for a given restaurant optimizes the **squared loss** on the training data?
- **Question:** What would happen if the website just listed the restaurants with the highest rating predicted in this way?

Reversion to the Mean

Reversion to the mean: Extreme predictions often generalize worse

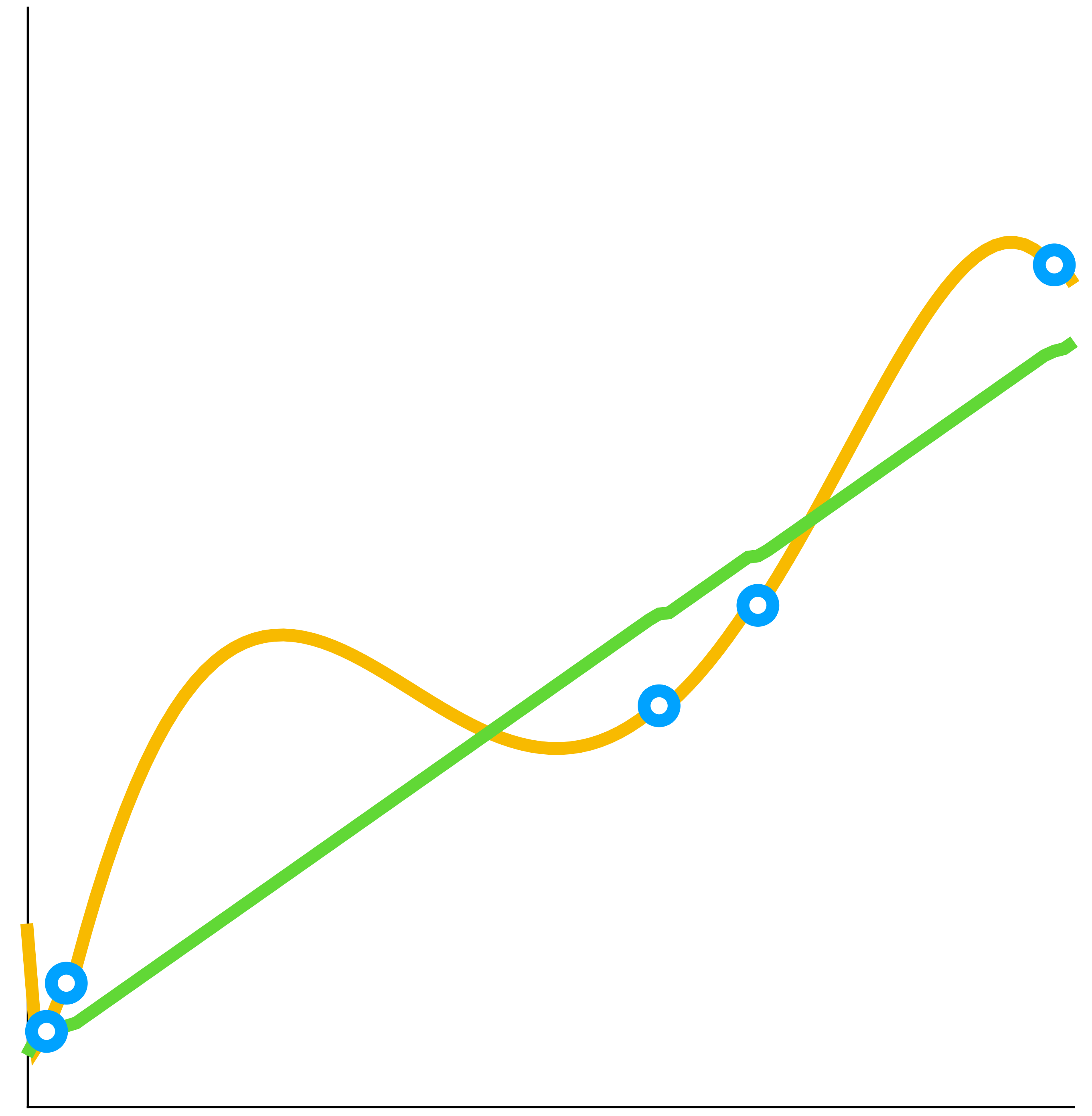
1. Children of very tall parents are likely to be shorter than either parent
2. The Sports Illustrated Cover curse: Players who have just appeared on the cover of Sports Illustrated often perform much worse subsequently
3. If the first few ratings are five stars, subsequent ratings are likely to be lower
    - Even if it's "really" a 5-star restaurant! (**why?**)

Model Complexity

- Adding **more parameters** to a model can usually fit the training data better
  - Especially when the larger model is a **generalization** of the smaller model; it is then **mathematically inevitable**
- *Intuition:*
  - Simple models can't represent much, so they are forced to prioritize the largest/**most important** effects
  - Complex models can represent more effects, including small, **unimportant**, and or **spurious** effects

Example: Fitting Polynomials

- A linear fit **won't hit** every observation exactly
- A sufficiently high-degree polynomial **will**
- **Question:** Which model's predictions are more credible?



Big Data

- More **training examples** usually lead to better **predictions** (i.e., better generalization) (**why**?)
- But this is not a cure-all
- Often when we have access to more **examples**, we also have access to more **features** of the examples
    - More features require more examples for efficient learning (**why?**)

Bias

What causes **test set error**? **Bias** + variance + noise

- **Bias** is error from systematically finding an **imperfect model**
  - **Representation bias:** Hypothesis space does not **contain** a model close enough to the **ground truth**
  - **Search bias:** Algorithm was not able to **find** a good enough hypothesis
- *Example:* **Decision trees** can represent **any function** of categorical variables, so they have **low representational bias**
  - The space of decision trees is too large to search exhaustively, so they can have a **high search bias**
- *Example:* **Linear regression** is a very simple class of models, so it has **high representation bias**
  - But the optimal linear model can be found analytically, so it has **zero search bias**

Variance

What causes **test set error**? Bias + **variance** + noise

- The smaller the training dataset, the more **different** we can expect our model estimates to be
  - *Restaurant Example:* how different would the estimates be from two training sets of **1 rating each**? How different would they be from two training sets of **100,000 ratings each**? (**why**?)
- **Variance** is the error from having **too little data** to train from
  - or (equivalently), from having **too complex** a model for the amount of data that we have
  - More complex models require more data to fit
- **Bias-variance tradeoff** (for a given fixed amount of data):
  - Complicated models will contain better hypotheses, but be harder to estimate
  - Simple models will be easier to estimate, but not as accurate (due to representational bias)

Noise

What causes **test set error**? Bias + variance + **noise**

- Sometimes the underlying process that generates our data is **inherently random**
  - In this case, we **cannot predict exactly** no matter how many we have
  - *Example:* Biased coin toss
- Sometimes the underlying process is not random, but we are **missing measurements** for important features
  - In this case, we also cannot predict exactly
  - The missing features make the process **appear** random
  - *Example:* Ice cream trucks only come out when it's sunny, but our dataset doesn't record the weather

Avoiding Overfitting

There are multiple approaches to avoiding overfitting:

1. **Pseudocounts**: Explicitly account for **regression to the mean**
2. **Regularization**: Explicitly **trade off** between fitting the data and model complexity
3. **Cross-validation**: **Detect** overfitting using some of the training data

Pseudocounts

- When we have not observed all the **values** of a variable, those variables should not be assigned **probability zero**
- If we don't have very much **data**, we should not be making very extreme predictions
- Solution: artificially add some "pretend" observations for each value of a variable (**pseudocounts**)
  - When there is not much data, predictions will tend to be less extreme as a result (**why**?)
  - When there is more data, the pseudocounts will have less effect on the predictions

Regularization

- We shouldn't choose a **complicated** model unless there is **clear evidence** for it
- Instead of optimizing directly for training error, optimize training error plus a **penalty** for complexity:

$$\arg\min_{h \in \mathcal{H}} \sum_{e} \mathrm{error}(e, h) + \lambda \times \mathrm{regularizer}(h)$$

- $regularizer$ measures the **complexity** of the hypothesis
- λ is the **regularization parameter**: indicates how important hypothesis complexity is compared to fit
    - Larger λ means complexity is more important

Types of Regularizer

- Number of **parameters**
- **Degree** of polynomial
- **L2** regularizer ("ridge regularizer"): sum of squares of weights
  - Prefers models with **smaller** weights
- **L1** regularizer ("lasso regularizer"): sum of absolute values of weights
  - Prefers models with **fewer nonzero** weights
  - Often used for **feature selection**: only features with nonzero weights are used

Cross-Validation

- Previous methods require us to already know how simple a model "should" be:
  - How many **pseudocounts** to add?
  - What should **regularization parameter** be?
  - What **degree of polynomial** should we use?
- Ideally we would like to be able to answer these questions **from the data**
- **Question:** Can we use the **test data** to see which of these work best?
- **Idea:** Use some of the **training data** as an **estimate** of the test data

Cross-Validation Procedure

Cross-validation can be used to estimate most bias-control parameters (**hyperparameters**)

1. **Randomly remove** some datapoints from the training set; these examples are the **validation set**
2. **Train** the model on the training set using some values of hyperparameters (pseudocounts, polynomial degree, regression parameter, etc.)
3. **Evaluate** the results on the validation set
4. **Update** values of hyperparameters
5. Repeat

k-Fold Cross-Validation

- We want our **training set** to be as large as possible, so we get better models
- We want our **validation set** to be as large as possible, so that it is an accurate estimation of test performance
- When one is larger, the other must be **smaller**
- **k-fold cross-validation** lets us use every one of our examples for both validation and training

k-Fold Cross-Validation Procedure

1. **Randomly partition** training data into k approximately equal-sized sets (folds)
2. **Train** k times, each time using all the folds but one; **remaining fold** is used for **validation**
3. **Optimize** hyperparameters based on **validation errors**

- Each example is used exactly **once** for **validation** and $k - 1$ **times** for **training**
- **Extreme case**: $k = n$ is called **leave-one-out** cross-validation

Summary

- **Overfitting** is when a learned model fails to **generalize** due to **overconfidence** and/or learning **spurious regularities**
- **Bias-variance tradeoff**: More **complex** models can be more **accurate**, but also require more **data** to train
- Techniques for avoiding overfitting:
  1. **Pseudocounts**: Add **imaginary** observations
  2. **Regularization**: **Penalize** model complexity
  3. **Cross-validation**: Reserve **validation data** to estimate test error