Lecture 4: Bias-Variance Tradeoff & Cross-Validation

ECE 2410 – Introduction to Machine Learning

Farzad Farnoud

University of Virginia

Spring 2026

Outline

Last Time: Implementing kNN

Key concepts from L03:

- Train/Test split for evaluation
- Images as vectors (flattening)
- Data normalization (Min-Max, Z-score)
- Classification performance metrics: accuracy, precision, recall

What we can do now:

1. Load real datasets (MNIST, NBA)
2. Split into train/test
3. Normalize features
4. Classify with kNN
5. Measure performance

The Big Question

🤔 How do we choose k?

- We've been using $k = 3$ or $k = 5$... why?
- Different k values give different accuracies
- What makes one model "better" than another?

Today's Goals:

1. Understand **why** some models fail (overfitting/underfitting)
2. Learn to **diagnose** issues (bias-variance tradeoff)
3. Know **how** to choose hyperparameters (cross-validation)

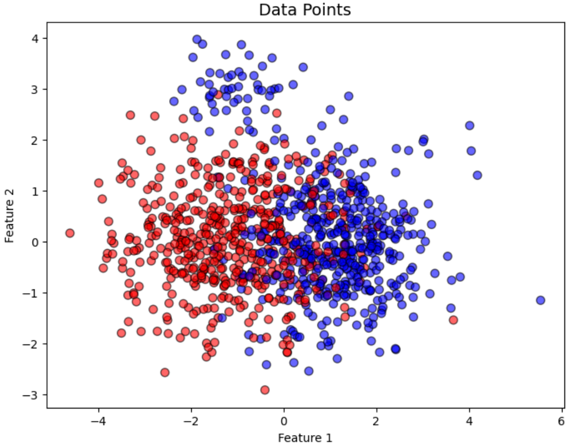
The Setup: Building a Classifier

The Goal:

- We have a dataset (Red vs Blue points).
- We want to separate them.
- The data clearly has some structure (a curved boundary), but also noise.

The Experiment:

- Let's divide this data into **4 disjoint subsets**.
- We will train a kNN model on each subset.
- **Question:** How much does the decision boundary change between subsets?



Possible Outcomes: What are we looking for?

Before we run the models, let's predict what **could** happen:

Scenario A: Stability

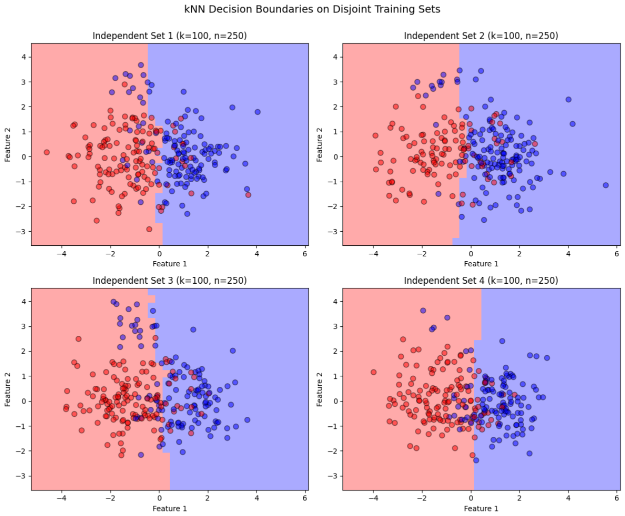
- The 4 models look **similar**.
- Good? Possibly yes, if they are capturing true patterns.
- Risk: Maybe they are all *wrong in the same way*? This is **High Bias**.

Scenario B: Variability

- The 4 models look **quite different**.
- Bad! At least some of them are wrong. Probably all of them (no reason to believe they perform differently).
- This is **High Variance**.

Ideally, training with different datasets from the same population (same underlying patterns) should give **similar** models that **capture the true patterns**.

Result 1: kNN with Large k ($k = 100$)



Result 1: Underfitting ($k = 100$)

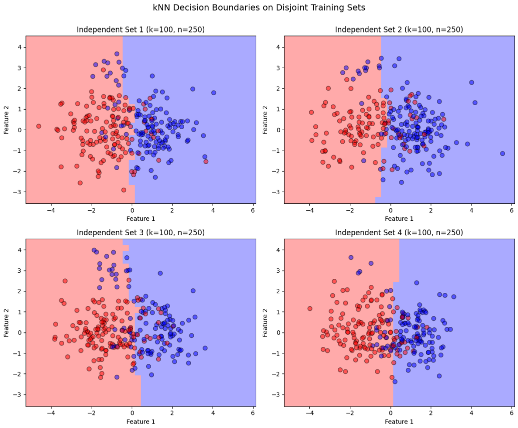
Observations:

- Boundaries are very smooth and nearly identical across subsets
- Low Variance (Stable)
- But wrong in the same way (missing the "ear"): High Bias

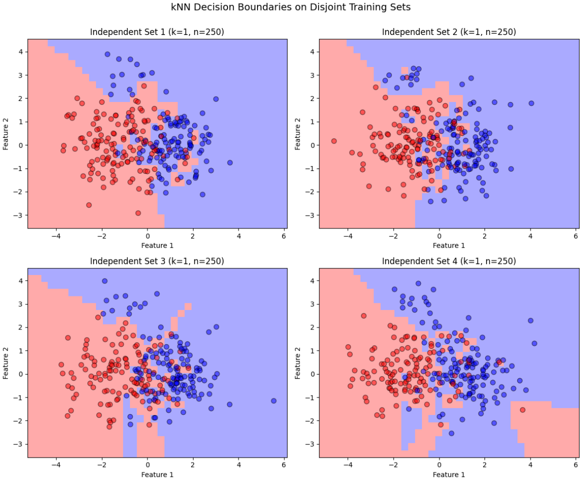
What happens:

- Model is too simple
- Fails to capture complex patterns

In kNN: $k = N$ always predicts majority class



Result 2: kNN with Small k ($k = 1$)



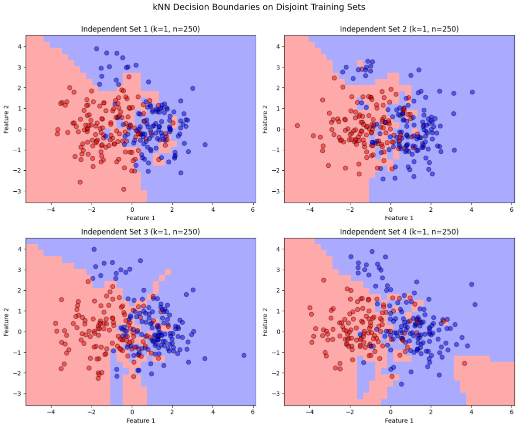
Observations:

- Boundaries are jagged
- Change wildly between subsets: High Variance
- There is no systematic or consistent error pattern: Low Bias

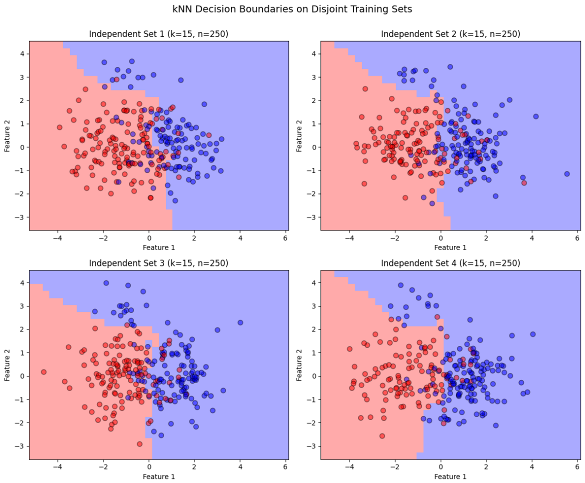
What happens:

- Model is too complex
- Too sensitive to any noise or overlap
- Memorizes training data including noise

In kNN: $k = 1$ means training accuracy $= 100\%$!

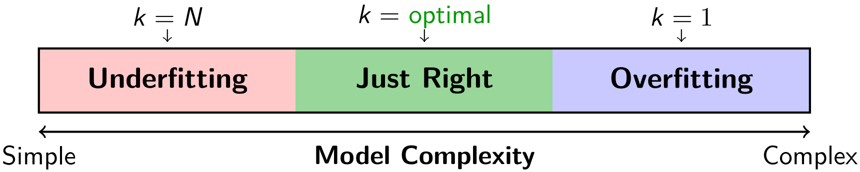


Result 3: The Sweet Spot ($k = 15$)



Result 3: The Sweet Spot ($k = 15$)

Observation: Boundaries capture the curve, but are relatively stable.
Diagnosis: Good balance between Bias and Variance $\rightarrow$ **Generalization!**



🎯 **Goal:** Find a model complex enough to learn patterns, but simple enough to generalize

What is Bias?

Definition: Bias

Bias measures how far off the model's *average prediction* is from the true value.

Intuition:

- Train many models on different training sets
- Average all their predictions
- Compare average prediction to truth

High Bias means:

- Model systematically misses the target
- Too simple to capture the pattern
- Underfitting

What is Variance?

Definition: Variance

Variance measures how much predictions *vary* when trained on different datasets.

Intuition:

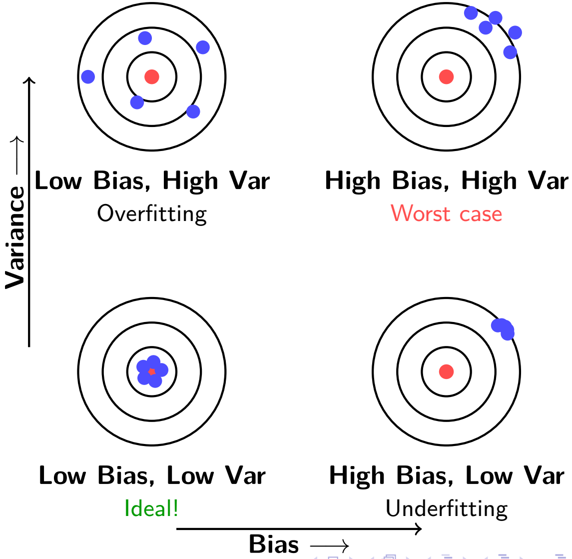
- Train model on different random subsets
- See how much predictions change
- High variance = predictions unstable

High Variance means:

- Model is very sensitive to training data
- Small changes $\rightarrow$ different predictions
- Overfitting

The Bullseye Analogy

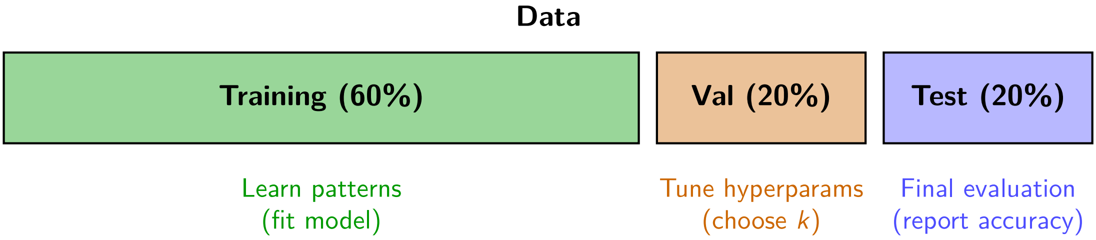
- Each blue dot represents a model trained on a different dataset (different sample from the same population)
- The **spread** of dots shows **variance**
- The **distance from center** shows **bias**



The Three-Way Split

Problem with Two-Way Split

- We need to choose k using *some* data
- But we can't use test data (that's cheating!)
- Solution: Split into **three** parts



Why Not Just Use Test Data?

The "Golden Rule" of ML Evaluation

The test set is for **final evaluation only**!
Never use it to make any decisions about your model.

Why this matters:

- If you pick k based on test performance, you're "overfitting to the test set"
- Your reported accuracy will be **overoptimistic**
- This is called **data snooping** or **data leakage**

Proper Workflow

1. Try different k values on validation set
2. Pick k^* with lowest validation error
3. Report final accuracy on test set (only once!)

Problem: Validation Set Wastes Data

Issue with fixed validation split:

1. We "lose" 20% of training data
2. With small datasets, this is a big problem!
3. With small datasets, validation results depend on which samples end up where

Solution: Cross-Validation

- Use *all* data for both training and validation
- Rotate which part is validation
- Average results over all rotations

Fixed Split Problem

Which one is right?

K-Fold Cross-Validation

5-Fold Cross-Validation

Val	Train	Train	Train	Train	$\rightarrow$ Error$_1$
Train	Val	Train	Train	Train	$\rightarrow$ Error$_2$
Train	Train	Val	Train	Train	$\rightarrow$ Error$_3$
Train	Train	Train	Val	Train	$\rightarrow$ Error$_4$
Train	Train	Train	Train	Val	$\rightarrow$ Error$_5$

$\longrightarrow$ **CV Error** $= \frac{1}{5}\sum_{i=1}^{5}$ Error$_i$

- Split data into K equal “folds” (common: $K = 5$ or $K = 10$)
- Each fold takes a turn as the validation set
- Average the K error estimates

Benefits of Cross-Validation

Advantages:

1. **Uses all data**: Every point is used for both training and validation
2. **More stable estimate**: Averaging reduces variance
3. **Better for small datasets**: Don't waste data on fixed validation set

Common choices:

- $K = 5$: Fast, works well in practice
- $K = 10$: Standard choice, good balance
- $K = N$ (LOOCV): "Leave-one-out" — most expensive but uses maximum data

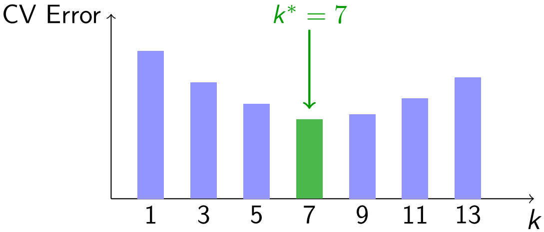
Leave-One-Out CV (LOOCV)

$K = N$ (number of samples): Each sample is its own validation fold.
⚠️ Very expensive for large datasets! (N training runs)

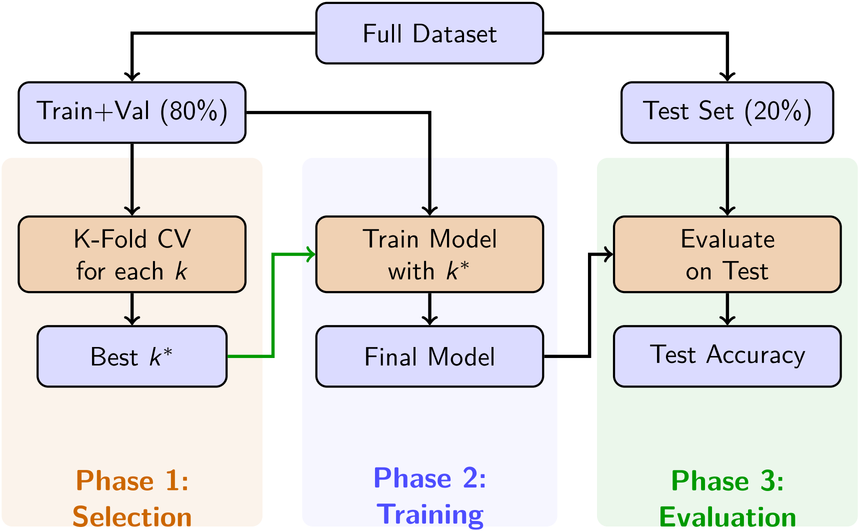
Grid Search: Finding the Best k

Algorithm: Grid Search with Cross-Validation

1. Define a grid of k values to try: $\{1, 3, 5, 7, 9, ...\}$
2. For each k:
   - Compute K-fold CV error
3. Pick $k^* = \text{argmin(CV error)}$
4. Evaluate final model (k^*) on test set



Complete Workflow



Key Takeaways

1. **Overfitting vs Underfitting**:
   - Overfitting: Too complex, memorizes noise (kNN: small k)
   - Underfitting: Too simple, misses patterns (kNN: large k)
2. **Bias-Variance Tradeoff**:
   - Bias = systematic error (underfitting)
   - Variance = sensitivity to training data (overfitting)
   - Goal: minimize total error = $\text{Bias}^2 + \text{Variance}$
3. **Cross-Validation**:
   - Use all data for training and validation
   - K-fold: rotate validation set, average results
4. **Hyperparameter Selection**:
   - Grid search: try many values, pick best by CV error
   - Only use test set for final evaluation!

Next Time: Cost Functions & Mathematical Formulations

Notebook Activities

Today's Hands-On Work

1. **Visualize Variance**: See how decision boundaries change with training data
2. **Implement K-Fold CV**: Write cross-validation from scratch
3. **Grid Search**: Find optimal k for kNN on real data
4. **Plot**: Training error vs CV error as k changes

📓 Open `L04-2026-01-26-Bias-Variance.ipynb`