MA477: Data Science

Lesson 16 Outline — 19 February 2026

United States Military Academy, West Point
Instructor: MAJ Patrick Kuiper

---

1 Administrative

- Student review
- Work Problem Set 2 (due next lesson)
- KNN Lecture

2 K-Nearest Neighbors Lesson Objectives

- Understand the strengths and limitations of k nearest neighbors (p.164 of ISLP).
- Understand how the bias-variance tradeoff applies when selecting k.
- Use modules in sklearn to assess k nearest neighbors classifiers.

3 Discussion: K-Nearest Neighbors (KNN)

Discussion Question 1

What is the fundamental idea behind the K-Nearest Neighbors algorithm, and how does it make a prediction?

Answer: KNN predicts a response by identifying the K training observations closest to a new point in predictor space. For classification, it assigns the class that appears most frequently among those K neighbors. The method is local and does not estimate a global model.

$$\Pr(Y = j \mid X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} \mathbf{1}(y_i = j).$$

$$\mathcal{N}_0 = \{i \in \{1, \ldots, n\} : \ d(x_0, x_i) \text{ is among the } K \text{ smallest distances}\}$$

Discussion Question 2

How is "closeness" defined in KNN, and why does scaling matter?

Answer: Closeness is typically measured using Euclidean distance, computed as the square root of the sum of squared differences across predictors. Variables on larger scales dominate the distance unless predictors are standardized, so scaling is usually necessary.

$$d(x_0, x_i) = \sqrt{\sum_{k=1}^{p} (x_{0k} - x_{ik})^2}$$

$p =$ the number of predictor variables (features), i.e., the dimension of the predictor space.

Discussion Question 3

How does the choice of K affect flexibility, bias, and variance?

Answer: Small K produces a highly flexible model with low bias and high variance (risk of overfitting). Large K produces a smoother model with higher bias and lower variance (risk of underfitting). Thus, K controls the bias–variance tradeoff.

Discussion Question 4

How does KNN differ from parametric models, and what are its limitations?

Answer: Unlike parametric models, KNN does not assume a fixed functional form; it uses the training data directly for prediction. While flexible, it can be computationally expensive, sensitive to scaling, and less effective in high dimensions due to the curse of dimensionality.

Discussion Question 5

What is the curse of dimensionality, and can you demonstrate it with a simple numerical example?

Answer: The curse of dimensionality refers to the phenomenon that as the number of predictors increases, data points become sparse and distances between points become less informative.

Simple example: Suppose predictors are scaled between 0 and 1.

In 1 dimension, consider a neighborhood of radius 0.1. That region covers 10% of the space.

In 2 dimensions, a square with side length 0.1 covers $0.1 \times 0.1 = 0.01$, or 1% of the space.

In 10 dimensions, a hypercube with side length 0.1 covers $0.1^{10} = 0.0000000001$ of the space.

Thus, to capture the same fraction of observations in higher dimensions, the neighborhood must expand dramatically. This makes "nearest" neighbors less meaningful and requires far more data.

Discussion Question 6

Why is KNN considered a non-parametric method, and how is this similar to SVM in high dimensions?

Answer: KNN is non-parametric because it does not assume a fixed functional form (such as a linear equation). It stores the data and makes predictions locally using nearby points.

Similarly, Support Vector Machines (SVMs), especially with kernels, do not assume a simple global linear relationship in the original feature space. Both methods rely on geometric structure in predictor space rather than estimating a small fixed set of parameters. However, KNN suffers more severely from the curse of dimensionality because it depends directly on distance calculations across all features.

Parameter	**Where (SVC)**	**Effect if Increased**	**Bias–Variance Interpretation (Concise)**
C	All kernels	Fewer training errors; narrower margin; more support vectors can become "active"	**Variance up, bias down.** Larger C fits training data more tightly (risk of overfitting). Smaller C allows more violations (more regularization).
`kernel`	Chooses model family (e.g., `linear`, `rbf`, `poly`)	More flexible kernels can fit more complex boundaries	**Flexibility affects bias–variance.** Linear tends to higher bias / lower variance; RBF and higher-degree polynomial tend to lower bias / higher variance.
γ	RBF, Polynomial, Sigmoid	More local / sharper influence of points (RBF); stronger scaling of dot products (Poly)	**Variance up, bias down.** Large γ yields very flexible boundaries (overfitting risk). Small γ yields smoother, more global boundaries (underfitting risk).
`degree` (d)	Polynomial kernel only	More complex polynomial interactions; more curvature	**Variance up, bias down.** Higher degree increases model complexity quickly; lower degree is smoother and more biased.
`coef0` (r)	Polynomial (also Sigmoid)	In polynomial kernels, increases the influence of lower-order terms relative to pure high-order interactions	**Often bias up, variance down (but data-dependent).** Larger `coef0` typically makes the kernel behave less "purely high-degree," which can stabilize fits; very small `coef0` can emphasize high-order interactions and increase variance.
`gamma = "scale"`	Default choice for γ (RBF/Poly)	Sets γ based on data variance and number of features	**Stabilizes variance across datasets.** Helps prevent extreme γ values when features are not standardized; still may need tuning.
`gamma = "auto"`	Alternative choice for γ (RBF/Poly)	Sets γ to $1/p$ (p = number of features)	**Simpler, can miscalibrate complexity.** May be too flexible or too smooth depending on feature scaling; often less robust than `"scale"`.

Table 1: SVM (scikit-learn `SVC`) tuning parameters and their bias–variance interpretations.