MA289 Mathematics of AI
Lesson 4 Outline — 03 February 2026
United States Military Academy, West Point
Instructor: MAJ Patrick Kuiper

1 One-Hot Encoding Example

One-hot encoding + real-valued feature

Let $C \in \{\text{Red}, \text{Blue}, \text{Green}\}$ and define the one-hot vector

$$(c_{\text{red}}, c_{\text{blue}}, c_{\text{green}}) \in \{0,1\}^3, \qquad c_{\text{red}} + c_{\text{blue}} + c_{\text{green}} = 1.$$

Let $x \in \mathbb{R}$ be a real-valued feature. Define

$$\mathbf{x} = (x, c_{\text{red}}, c_{\text{blue}}, c_{\text{green}}), \qquad \mathbf{w} = (w_x, w_{\text{red}}, w_{\text{blue}}, w_{\text{green}}).$$

Then the linear model prediction is

$$\hat{y} = \mathbf{w}^\top \mathbf{x} = w_x x + w_{\text{red}} c_{\text{red}} + w_{\text{blue}} c_{\text{blue}} + w_{\text{green}} c_{\text{green}}.$$

If $C = \text{Blue}$, then $(c_{\text{red}}, c_{\text{blue}}, c_{\text{green}}) = (0,1,0)$ and

$$\hat{y} = w_x x + w_{\text{blue}}.$$

Step	Student checklist for a prediction project
1	**Clarify the question:** Define the target, task type (regression vs. classification), and what "good" performance means in context.
2	**EDA (Exploratory Data Analysis):** Inspect data types, summary statistics, missingness, outliers, class balance, and simple plots/relationships to understand what you have.
3	**Baseline + preprocessing:** Build a simple baseline model; decide if features need **scaling/standardization** (especially for distance/gradient-based models) and encode categorical variables as needed.
4	**Evaluation + regularization:** Choose an appropriate metric/loss; use a validation plan (train/test split or cross-validation); apply **regularization** (e.g., L1/L2) and tune hyperparameters to reduce overfitting.
5	**Overfitting check:** Compare training vs. validation performance; look for instability across folds/splits; simplify model or increase regularization if needed.
6	**Interpret + sanity-check:** Confirm predictions make sense; check feature effects; verify there is no data leakage; summarize limitations and next steps.

Error Metrics and Their Interpretation

1. **Most Common Error Metric**
   What is the most commonly used error metric in regression problems, and why is it so widely adopted in practice?
   The most commonly used error metric is mean squared error (MSE). It is widely adopted because it is mathematically convenient, differentiable everywhere, and leads to closed-form solutions or efficient gradient-based optimization methods.

2. **Alternative Error Metric**
   What is another commonly used error metric, and in what situations might it be preferred over the most common choice?
   Mean absolute error (MAE) is another common error metric. It is often preferred when robustness to outliers is important, since it penalizes errors linearly rather than quadratically.

3. **Tradeoffs Between Metrics**
   What are the primary advantages and disadvantages of these two error metrics, particularly with respect to sensitivity to large errors and robustness to outliers?
   MSE strongly penalizes large errors, making it sensitive to outliers but effective when large deviations are especially costly. MAE treats all errors proportionally, making it more robust to outliers but less sensitive to rare large errors.

4. **Analytical Demonstration**
   How can the difference between these two error metrics be demonstrated analytically, for example by comparing their geometric interpretations or their level sets in two dimensions?
   Analytically, MAE corresponds to the L1 norm and produces diamond-shaped level sets in two dimensions, while the square root of MSE corresponds to the L2 norm and produces circular level sets. These different geometries explain their distinct optimization and robustness properties.

For a vector $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, the most common norms are:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \quad \text{(L1 norm: sum of absolute values)},$$

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \quad \text{(L2 norm: Euclidean length)},$$

$$\|x\|_\infty = \max_{1 \le i \le n} |x_i| \quad \text{(L-infinity norm: maximum component)}.$$

K-Fold Cross-Validation (CV)

Idea. Cross-validation estimates how well a model will generalize to new data. Instead of training once on a single train/test split, we repeatedly train and validate on different parts of the data.

Procedure (K-fold CV)

Assume we have n labeled examples and choose an integer K (often 5 or 10).

1. Randomly shuffle the dataset (optional but common).
2. Split the data into K **folds** of (roughly) equal size.
3. For each round $i = 1, 2, \ldots, K$:
   - Use fold i as the **validation set**.
   - Use the other $K - 1$ folds as the **training set**.
   - Train the model on the training set and compute a validation score on fold i.
4. Average the K validation scores to get the cross-validation estimate:

$$\text{CV score} = \frac{1}{K} \sum_{i=1}^{K} s_i,$$

   where s_i is the chosen metric (accuracy, MSE, etc.) on validation fold i.

Why it helps. Each example is used for validation exactly once and for training $K - 1$ times, so the estimate is typically more stable than a single split.

A simple visual (5-fold CV)

Legend: **T** = training fold, **V** = validation fold

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Round 1:	V	T	T	T	T
Round 2:	T	V	T	T	T
Round 3:	T	T	V	T	T
Round 4:	T	T	T	V	T
Round 5:	T	T	T	T	V

Common variations

- **Stratified K-fold:** keeps class proportions similar in each fold (important for classification).
- **Repeated K-fold:** repeats K-fold CV multiple times with different shuffles.
- **Leave-one-out (LOOCV):** $K = n$ (each fold has one example); can be expensive.