CS-470: Machine Learning

Week 3 — Polynomial Regression and Regularization

Instructor: Dr. Sajjad Hussain

Department of Electrical and Computer Engineering
SEECS, NUST

September 25, 2025

Outline

Recap of Week-02

- Explored **Linear Regression**:
  - Simple linear regression (one variable).
  - Multiple linear regression with many features.
  - Matrix notation for compact representation.
- Learned about **Cost Function**:
  - Mean Squared Error (MSE) as a measure of fit.
  - Goal: minimize cost by finding best parameters.
- Discussed **Gradient Descent**:
  - Update rule: move parameters in direction of steepest descent.
  - Role of learning rate α: too small = slow, too large = unstable.
  - Variants: Batch, Stochastic, and Mini-batch GD.

Polynomial Regression

- Real-world data is often more complex than a simple straight line.
- Surprisingly, we can still use a **linear model** to fit nonlinear data.
- Idea: Add powers of each feature as new features, then train a linear model on this extended dataset.
- This approach is called **Polynomial Regression**.

How Polynomial Regression Works

- Example with one feature x:

$$y \approx \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \dots$$

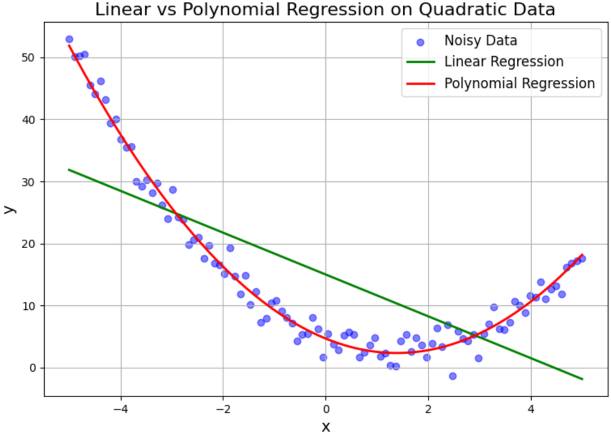
- The model is still linear in parameters θ, but can capture nonlinear patterns.
- By adding higher-order terms, the model becomes more flexible.

Polynomial Regression with Multiple Features

- With multiple features, Polynomial Regression captures interactions between features.
- We can use **PolynomialFeatures** class from Scikit-Learn library to generates all combinations of features up to the chosen degree.
- Example: With features a and b, and degree $= 3$:
  - Individual powers: a^2, a^3, b^2, b^3
  - Combinations: ab, a^2b, ab^2
- This makes Polynomial Regression much richer than plain Linear Regression.

Linear vs Polynomial Regression

- As shown, the linear model underfits the quadratic dataset, while polynomial regression captures the curvature much better.



Learning Curves

- A learning curve plots the **training error** and **validation error** as the number of training examples increases.
- They provide insight into whether the model is:
  - **Underfitting**: both training and validation errors are high.
  - **Overfitting**: training error is low but validation error is high.
- By analyzing learning curves, we can decide whether to use a more complex model, gather more data, or adjust regularization.

Learning Curves: Underfitting vs Overfitting

- **Underfitting:** Model is too simple, both errors remain high and close to each other.
- **Overfitting:** Training error becomes very small, but validation error remains high with a large gap.

Bias, Variance, and Irreducible Error

- **Bias:** Error due to wrong assumptions about the data (e.g., assuming linear when it is quadratic).
  - High bias $\rightarrow$ underfitting
- **Variance:** Error due to sensitivity to small variations in the training set.
  - High variance $\rightarrow$ overfitting
- **Irreducible error:** Caused by inherent noise in the data.
  - Can only be reduced by improving data quality (fix sensors, remove outliers, etc.)

The Bias–Variance Tradeoff

- Increasing model complexity:
  - $\uparrow$ Variance, $\downarrow$ Bias
  - Risk of overfitting
- Reducing model complexity:
  - $\uparrow$ Bias, $\downarrow$ Variance
  - Risk of underfitting

- **Tradeoff:** Find the right balance of bias and variance to minimize generalization error.

Regularized Linear Models

- A good way to reduce **overfitting** is to **regularize the model** (i.e., to constrain it).
- Fewer degrees of freedom $\rightarrow$ harder for the model to overfit.
- Example: In Polynomial Regression, reduce the polynomial degree.
- In Linear Models, regularization is achieved by **constraining the weights**.
- We will study:
  - Ridge Regression
  - Lasso Regression
  - Elastic Net

Ridge Regression

- Ridge Regression (a.k.a. Tikhonov regularization) is a **regularized version of Linear Regression**.
- It adds a **penalty term** to the cost function to keep weights small:

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

- Only weights $\theta_1, \ldots, \theta_n$ are regularized, not the bias θ_0.
- **Effect:** The model must balance fitting the data and keeping weights small (reduces overfitting).
- Hyperparameter α:
  - $\alpha = 0 \Rightarrow$ Ordinary Linear Regression.
  - Large $\alpha \Rightarrow$ weights ≈ 0, prediction $\approx$ flat line at data mean.

Ridge Regression – Training Notes

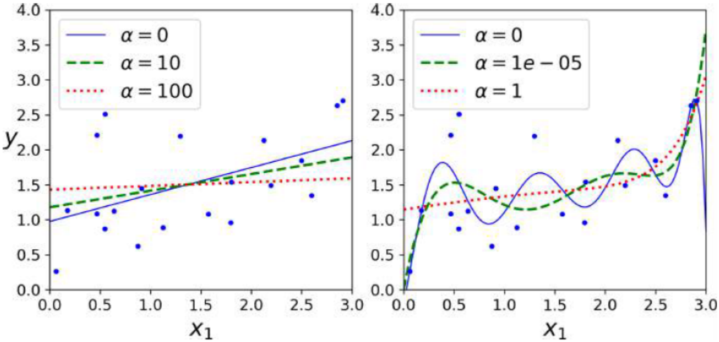
- The cost function used in training may differ from the evaluation metric:
  - Training: add regularization term for optimization.
  - Testing: evaluate using standard performance measures (e.g., MSE).
- For Gradient Descent:

$$\nabla J(\theta) = \nabla \text{MSE}(\theta) + \alpha w$$

  where $w = (\theta_1, \ldots, \theta_n)$.
- Important: **Scale input features** (e.g., StandardScaler) before applying Ridge Regression.

Ridge Regression – Effect of α

- Ridge models with different α values:
  - Left: Ridge applied directly to linear data.
  - Right: Polynomial features (degree = 10) + StandardScaler + Ridge.
- Increasing α makes predictions **flatter and less extreme**.
- **Bias–variance tradeoff:** larger α reduces variance but increases bias.



Lasso Regression

- **Lasso Regression** (Least Absolute Shrinkage and Selection Operator):
  - Adds an ℓ_1 penalty to the cost function.
  - Cost function:

$$J(\theta) = \mathsf{MSE}(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$$

- **Key property:** Lasso tends to set some weights exactly to zero.
  - Performs **automatic feature selection**.
  - Produces **sparse models** (few nonzero weights).
- Hyperparameter α:
  - $\alpha = 0$: plain Linear Regression.
  - Large α: stronger regularization $\Rightarrow$ more weights forced to zero.

Elastic Net

- **Elastic Net:** Combines Ridge (ℓ_2) and Lasso (ℓ_1) penalties.
- Cost function:

$$J(\theta) = \text{MSE}(\theta) + r\alpha \sum_{i=1}^{n} |\theta_i| + \frac{1-r}{2}\alpha \sum_{i=1}^{n} \theta_i^2$$

- **Mixing ratio r:**
  - $r = 0 \quad \Rightarrow \quad$ Ridge Regression
  - $r = 1 \quad \Rightarrow \quad$ Lasso Regression
- **When to use:**
  - **Ridge:** good default (all features contribute).
  - **Lasso:** if only a few features are useful.
  - **Elastic Net:** preferred over Lasso when
    - features $>$ training instances, or
    - features are strongly correlated.
- In practice, **always use some regularization** (avoid plain Linear Regression).

Week-03 Summary

- **Underfitting vs. Overfitting:**
  - Underfitting: model too simple $\rightarrow$ high bias.
  - Overfitting: model too complex $\rightarrow$ high variance.
  - Best model balances bias and variance.
- **Learning Curves:**
  - Train vs. validation error on same graph.
  - Diagnose underfitting (both errors high) and overfitting (gap between train and validation).
- **Regularized Linear Models:**
  - Ridge Regression: ℓ_2 penalty (shrinks weights).
  - Lasso Regression: ℓ_1 penalty (performs feature selection).
  - Elastic Net: combination of ℓ_1 and ℓ_2.
- **Key takeaway:** Regularization reduces overfitting and improves generalization.