Loss Functions and Regularization

David I. Inouye

Outline

- Loss functions
  - Regression losses
  - Classification losses
- Regularization
  - "Implicit regularization" by changing k in KNN
  - L2 regularization
  - L1 regularization and feature selection
- Caveat: Very brief introduction to these concepts
  - If you want to learn more, take ECE50024 Machine Learning I

Many Machine Learning Methods Minimize the **Average Loss** (a.k.a. **Risk Minimization**)

- Remember linear regression objective:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f_\theta(x_i))^2$$

- We can rewrite this as:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} l(y_i, f_\theta(x_i))$$

  - where $l(y, \hat{y}) = (y - \hat{y})^2$ is the **loss function**
  - Many supervised ML can be written as above

Many Supervised ML Can Be Written Minimizing the Average Loss

- Ordinary least squares uses **squared loss**:

$$l(y, \hat{y}) = (y - \hat{y})^2$$

- Logistic regression uses **logistic loss**:

$$l(y, \hat{p} \in [0, 1]) = -y \log \hat{p} - (1 - y) \log(1 - \hat{p})$$

$$l(y, \hat{z} \in \mathbb{R}) = -y \log \sigma(\hat{z}) - (1 - y) \log(1 - \sigma(\hat{z}))$$

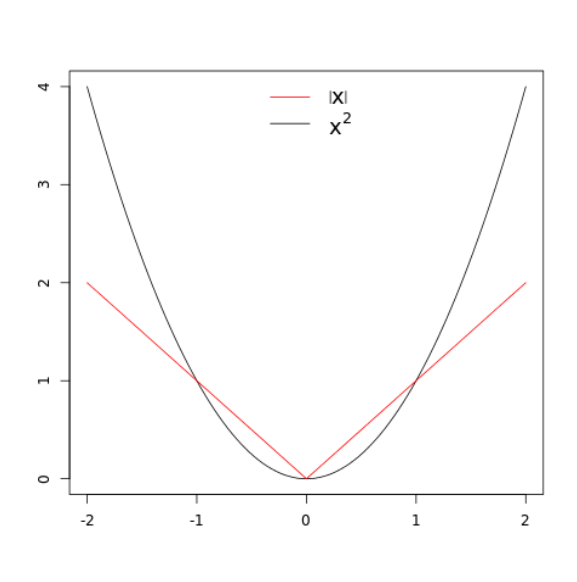
- Classification error is known as **0-1 loss**:

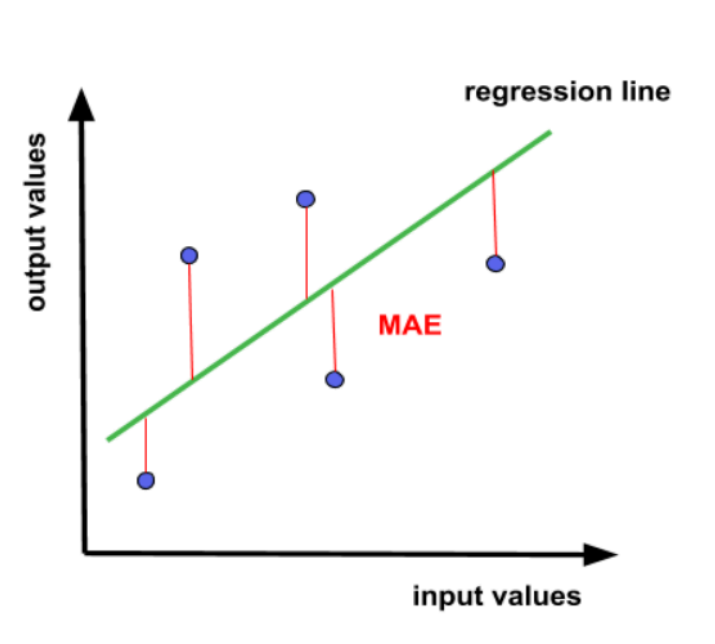
$$l(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y} \\ 1, & \text{otherwise} \end{cases}$$

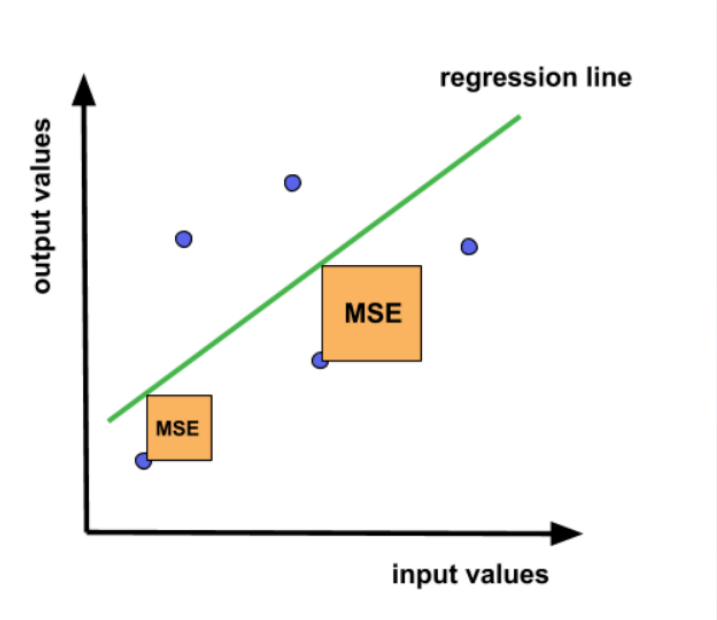
Example: **Absolute Error** Is Less Sensitive to Outliers but Is Harder to Optimize

Absolute error loss is:

$$l(y, \hat{y}) = |y - \hat{y}|$$





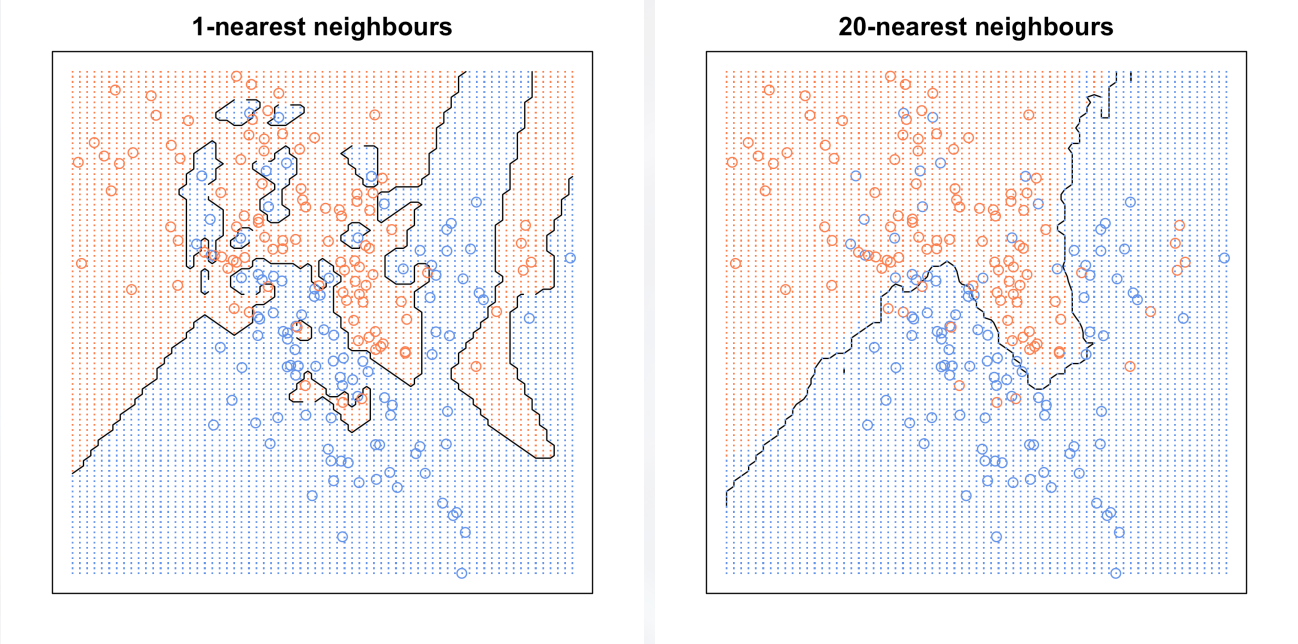


https://www.datacourses.com/evaluation-of-regression-models-in-scikit-learn-846/

Regularization Is a Common Method to Improve Generalization by Reducing the Complexity of a Model

- k in KNN can be seen as an **implicit regularization** technique
- We can use **explicit regularization** for parametric models by adding a **regularizer** $R(\theta)$

$$\min_{\theta} \sum_{i} l(y_i, f_\theta(x_i)) + \lambda R(\theta)$$



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Brief Aside: 1D Polynomial Regression Can Be Computed by Creating Polynomial "Pseudo" Features

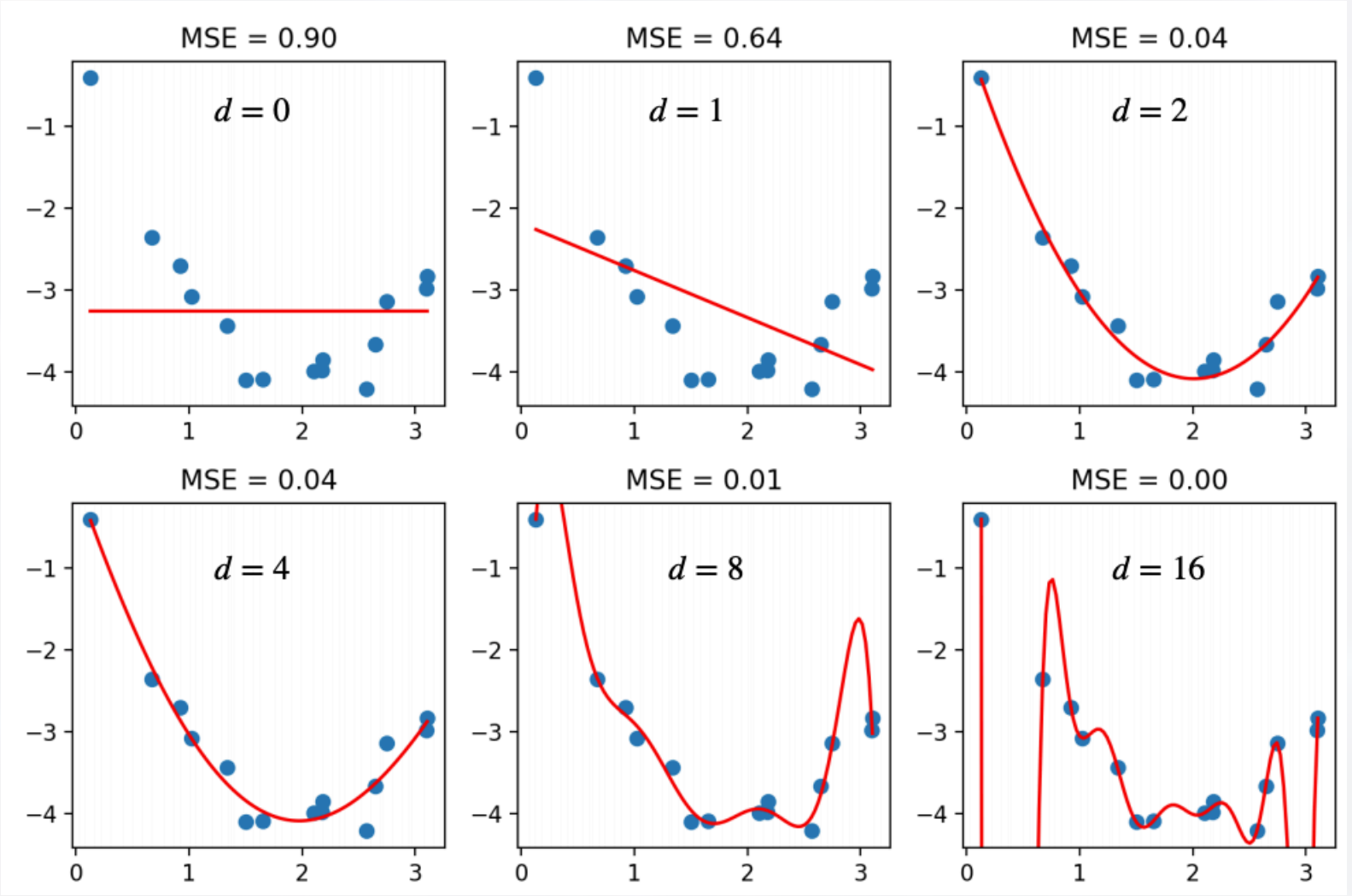
- Suppose we have 1D input data, i.e., $X \in \mathbb{R}^{n \times 1}$
- We can create pseudo polynomial features, e.g.

$$X' = \begin{bmatrix} x_1 & x_1^2 & x_1^3 \\ x_2 & x_2^2 & x_2^3 \\ x_3 & x_3^2 & x_3^3 \end{bmatrix} \in \mathbb{R}^{n \times 3}$$

- Linear regression can then be used to fit a polynomial model

$$y_i = \theta_1 x_i + \theta_2 (x_i^2) + \theta_3 (x_i^3) \ldots$$

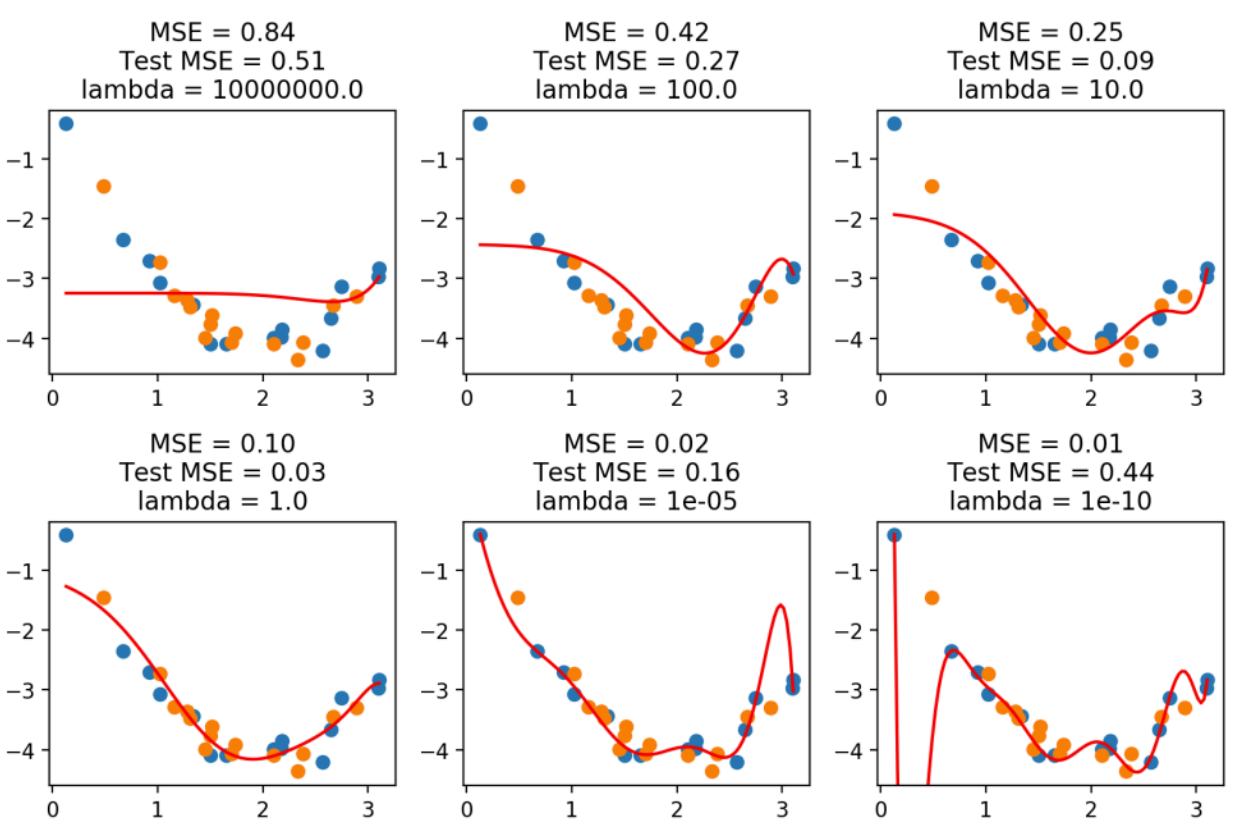
Brief Aside: 1D Polynomial Regression Can Be Computed by Creating Polynomial "Pseudo" Features



Ridge Regression: A Squared Norm Regularizer Encourages Small Parameter Values

- **Ridge regression** is defined as:

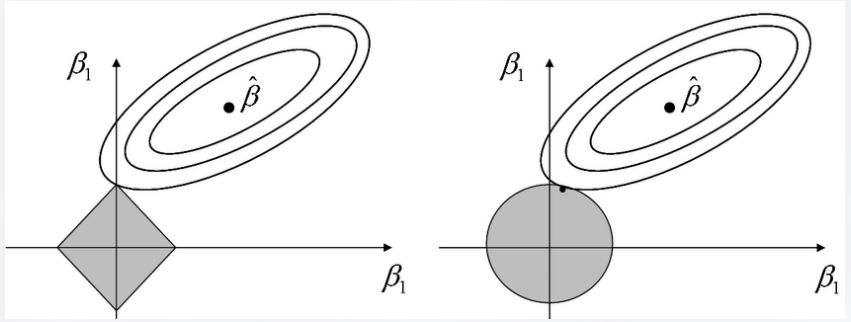
$$\min_{\theta} ||\, y - X\theta||_2^2 + \lambda ||\theta||_2^2$$



Regularizing the parameters of 1D polynomial regression helps to improve test MSE if chosen appropriately.

Lasso Regression: An L_1 Norm Regularizer Encourages **Sparsity** in the Parameters (i.e., Zeros)

- **Lasso regression** is defined as: $\min_\theta ||\, y - X\theta||_2^2 + \lambda||\theta||_1$
- Because lasso encourages **exact zeros**, lasso can be used for *feature selection*.
  - $f_\theta(x) = \theta_1 x_1 + \theta_2 x_2 = (0)x_1 + \theta_2 x_2 = \theta_2 x_2$



▶ Code