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 <u>average loss</u> (a.k.a. <u>risk minimization</u>)

- Remember linear regression objective:

$$\theta^* = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f_\theta(\boldsymbol{x}_i)\right)^2$$

- We can rewrite this as:

$$\theta^* = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell\left(y_i, f_\theta(\boldsymbol{x}_i)\right)$$

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

Many supervised ML can be written minimizing the average loss

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

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

- Logistic regression uses **logistic loss**

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

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

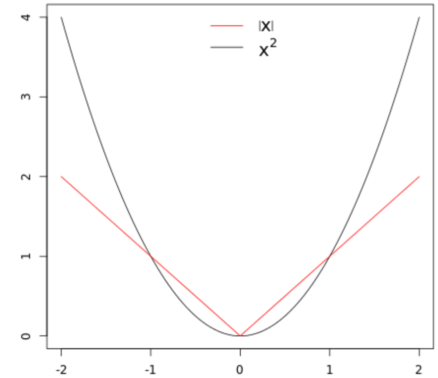
- Classification error is known as **0-1 loss**

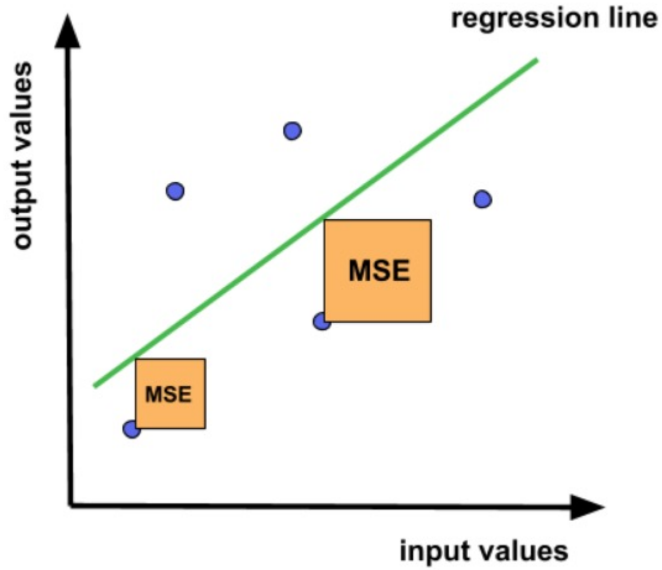
$$\ell(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y} \\ 1, & \text{otherwise} \end{cases}$$

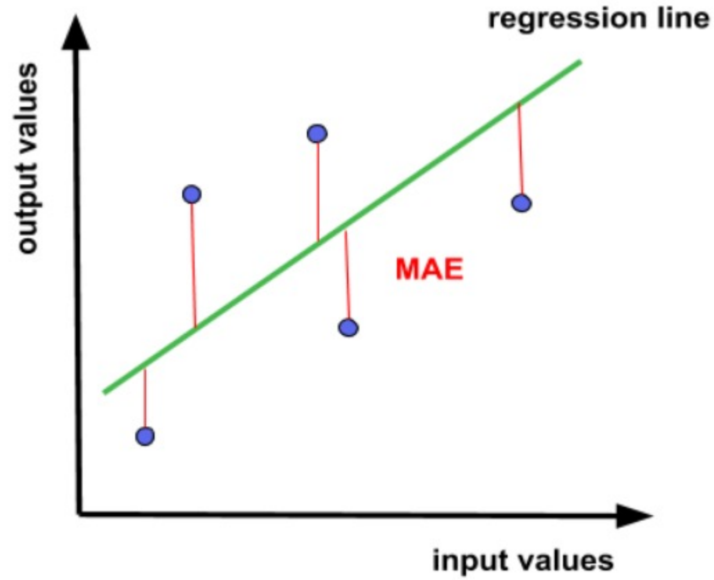
Example: <u>**Absolute error**</u> is less sensitive to outliers but is harder to optimize

- Absolute error loss is:

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







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

Example: The <u>hinge loss</u> is used for learning support vector machine (SVM) classifiers

- <u>**Hinge loss**</u> is defined as:

$$\ell(y, \hat{z}) = \max\{0, 1 - y\hat{z}\}$$

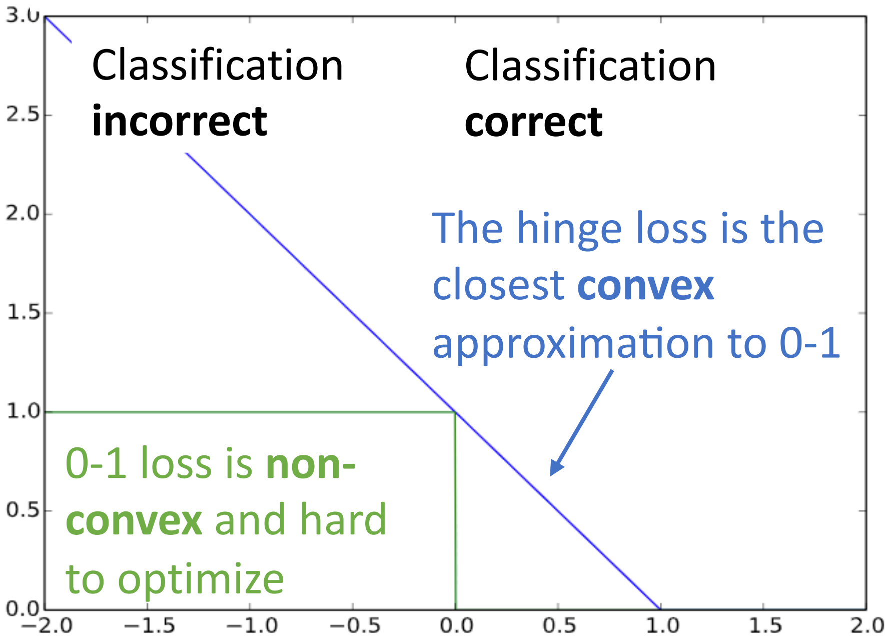
$$(\text{Note: } y \in \{-1, 1\})$$

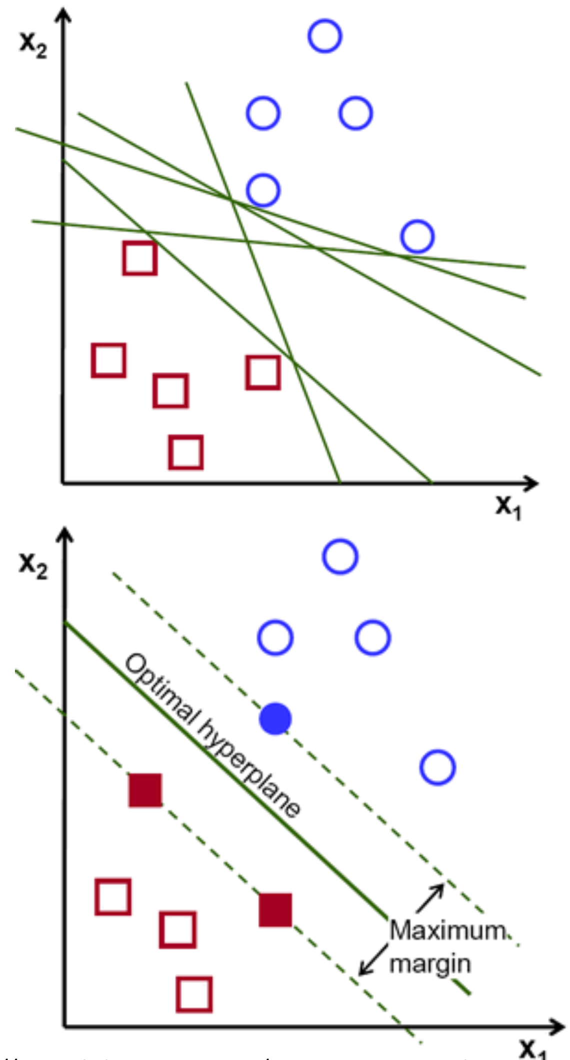
$$\ell(y, \hat{z}) = \begin{cases} \begin{cases} 1 - \hat{z}, & \hat{z} \le 1 \\ 0, & \hat{z} > 1 \end{cases} & y = 1 \\ \begin{cases} 1 + \hat{z}, & \hat{z} \ge -1 \\ 0, & \hat{z} < -1 \end{cases} & y = -1 \end{cases}$$

Example: The <u>**hinge loss**</u> is used for learning support vector machine (SVM) classifiers

- <u>**Hinge loss**</u> is defined as:
  $$\ell(y, \hat{z}) = \max\{0, 1 - y\hat{z}\}$$
  (Note: $y \in \{-1, 1\}$)

(Assume $\mathrm{y} = 1$ below)





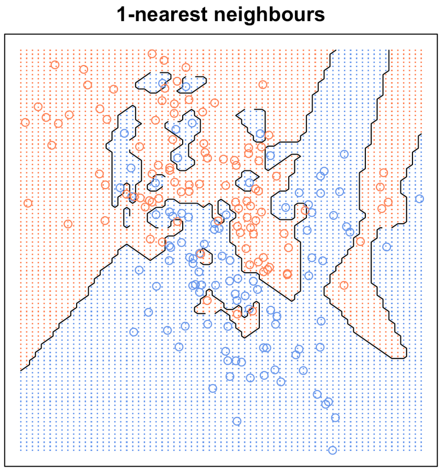
https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

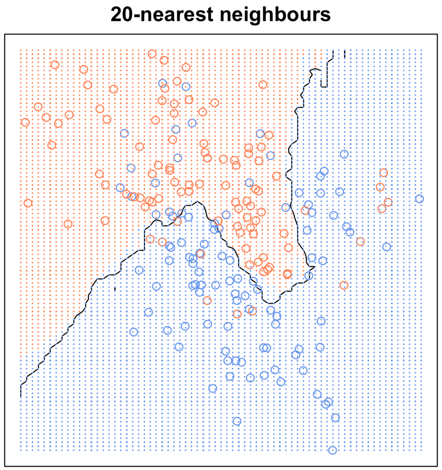
Regularization is a common method to improve *generalization* by **reducing the complexity of a model**

- k in KNN can be seen as an *implicit* regularization technique

- We can use *explicit* regularization for parametric models by adding a **regularizer** $R(\theta)$

$$\min_{\theta} \sum_i \ell(y_i, f_\theta(\boldsymbol{x}_i)) + \lambda R(\theta)$$





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

Brief aside: 1D polynomial regression can be computed by creating polynomial “pseudo” features

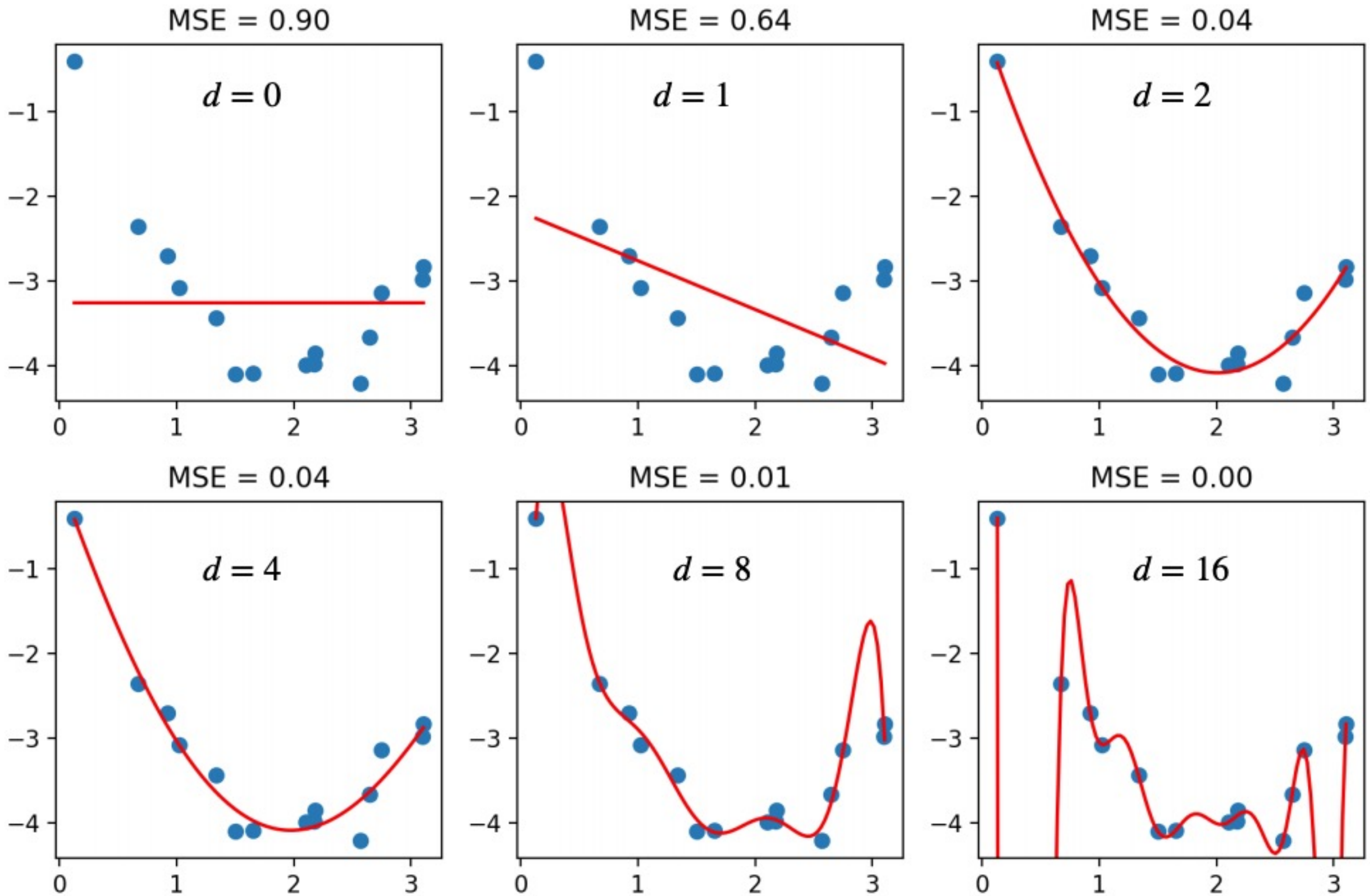
- Suppose we have 1D input data, i.e., $X \in \mathcal{R}^{n\times 1}$
- We can create pseudo polynomial features, e.g.

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

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

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

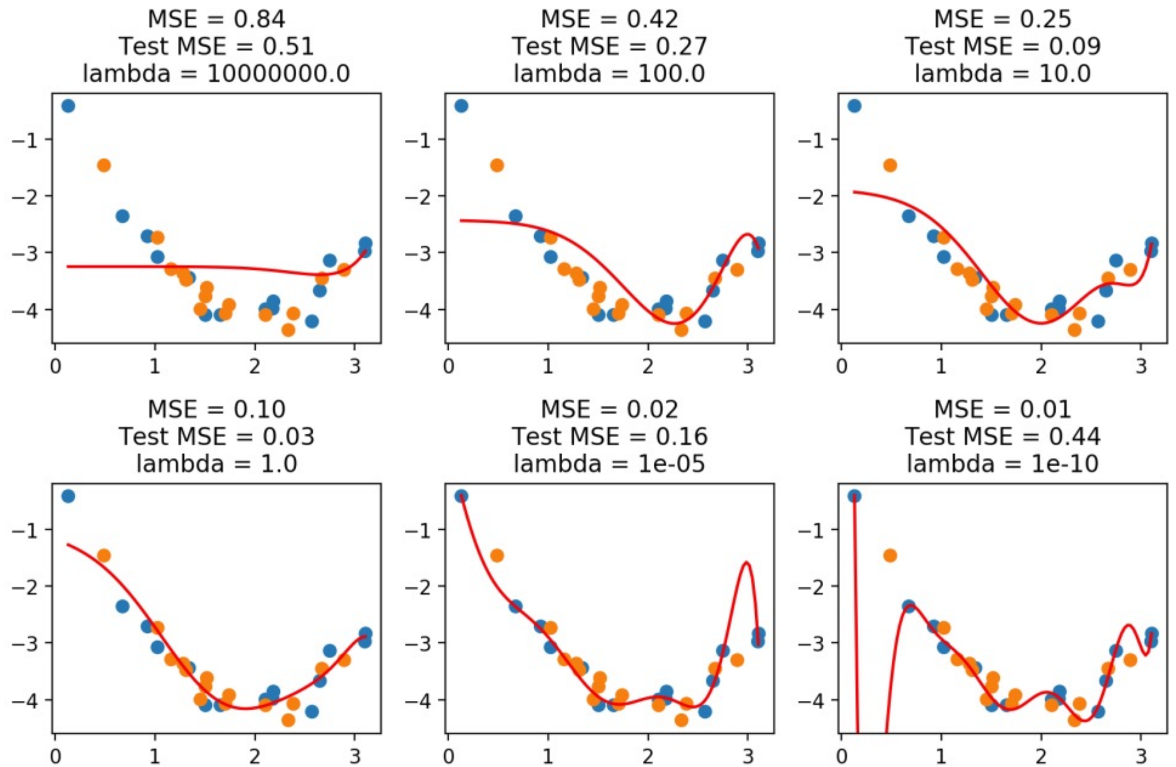
Brief aside: 1D polynomial regression can be computed by creating polynomial "pseudo" features



Ridge Regression: A squared norm regularizer encourages small parameter values

- Ridge regression is defined as:

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

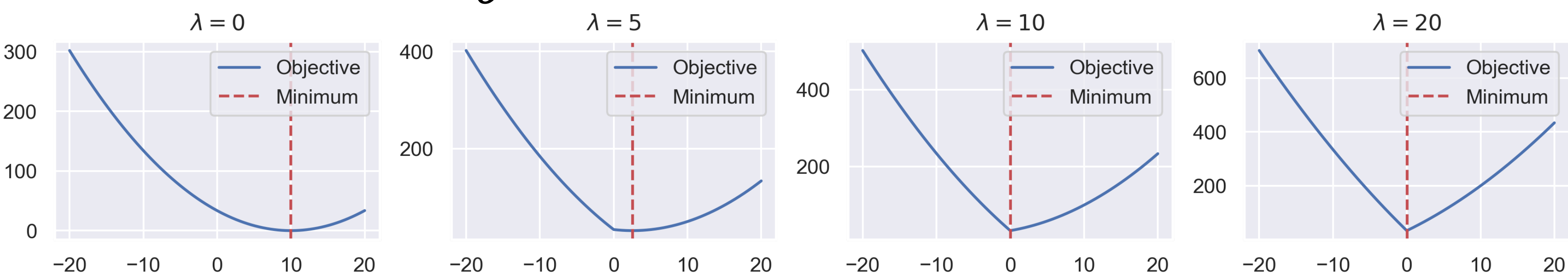


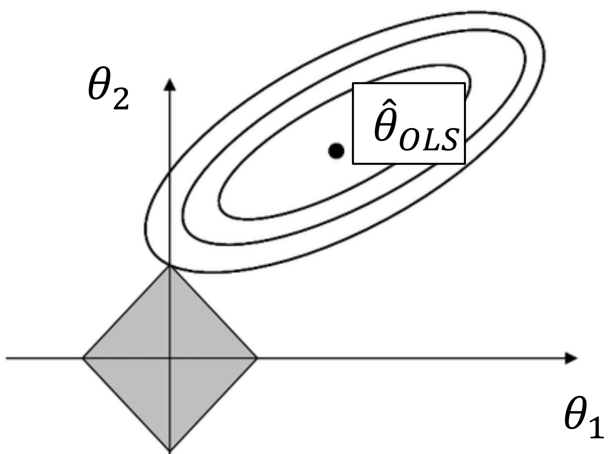
Regularizing the parameters of 1D polynomial regression helps to improve test MSE if chosen appropriately.

<u>Lasso Regression</u>: An L_1 norm regularizer encourages <u>sparsity</u> in the parameters (i.e., zeros)

- Lasso regression is defined as:

$$\min_{\theta} \|\boldsymbol{y} - X\theta\|_2^2 + \lambda \|\theta\|_1$$





Because lasso encourages **exact zeros**, lasso can be used for <u>**feature selection**</u>.

$$f_\theta(x) = \theta_1 x_1 + \theta_2 x_2$$
$$= (0)x_1 + \theta_2 x_2$$
$$= \theta_2 x_2$$