# CS5489
# Lecture 6.1: Robust and Non-Linear Regression

Kede Ma

City University of Hong Kong (Dongguan)

香港城市大學（東莞）
City University of Hong Kong
(Dongguan)

Slide template by courtesy of Benjamin M. Marlin

## Outline

## Regression

### Definition: The Regression Task

Given a feature vector $\mathbf{x} \in \mathbb{R}^N$, predict its output value $y \in \mathbb{R}$

### Definition: Regression Learning Problem

Given a data set of example pairs $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \ldots, M\}$ where $\mathbf{x}^{(i)} \in \mathbb{R}^N$ is a feature vector and $y^{(i)} \in \mathbb{R}$ is the output, learn a function $f : \mathbb{R}^N \mapsto \mathbb{R}$ that accurately predicts $y$ for any feature vector $\mathbf{x}$

### Error Measure: Mean Squared Error (MSE)

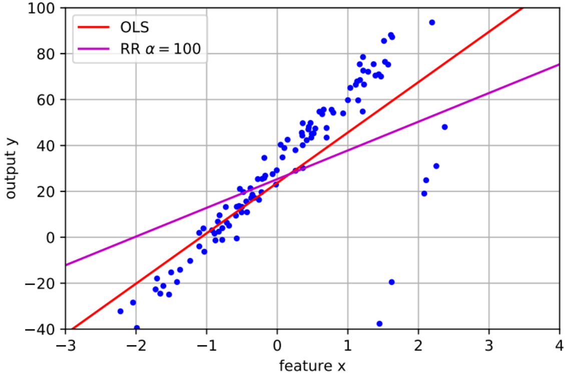
Given a function $f : \mathbb{R}^N \mapsto \mathbb{R}$, the MSE of $f$ on $\mathcal{D}$ is

$$\text{MSE}(f, \mathcal{D}) = \frac{1}{M} \sum_{i=1}^{M} (y^{(i)} - f(\mathbf{x}^{(i)}))^2$$

## Outline

## Outliers

- Outliers in the data can affect the squared-error term
  - Regression function will try to reduce large prediction errors for outliers, at the expense of worsening predictions for other points

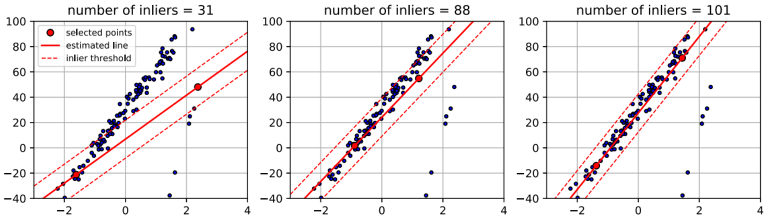


## RANSAC

- **RAN**dom **SA**mple **C**onsensus
  - Attempts to robustly fit a regression model in the presence of corrupted data (outliers)
  - Works with any regression model
- **Idea**:
  - Split the data into inliers (good data) and outliers (bad data)
  - Learn the model only from the inliers

## Random Sampling

- Repeat many times...
  - Randomly sample a subset of points from the data. Typically just enough to learn the regression model
  - Fit a model to the subset
  - Determine all data as inlier or outlier by calculating the residuals (prediction errors) and comparing to a threshold. The set of inliers is called the consensus set
  - Save the model with the highest number of inliers
- Finally, use the largest consensus set to learn the final model



## How to Choose Parameters?

- More repeats increase the chance to find the correct function
  - Higher probability to select a subset of points contains all inliers
- Mathematically,
  - $T$ - number of iterations (what we would like to determine)
  - $s$ - number of points (just enough) to fit the model
  - $e$ - proportion of outliers (so % inliers $= 1 - e$)
  - $p(\text{training subset with all inliers}) = (1-e)^s$
  - $p(\text{training subset with at least one outlier}) = 1 - (1-e)^s$
  - $p(\text{all } T \text{ training repeats have outliers}) = (1 - (1-e)^s)^T$
  - $\delta$ - probability of success (so probability of failure $= 1 - \delta$)
  - We want

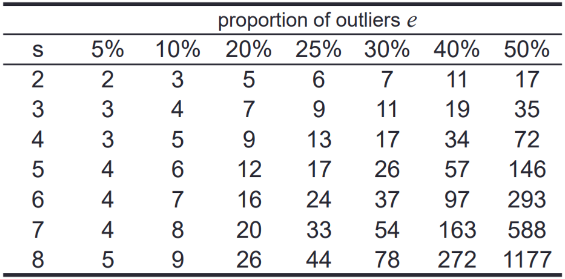
$$(1 - (1-e)^s)^T < 1 - \delta$$

$$T > \frac{\log(1-\delta)}{\log(1-(1-e)^s)}$$

- Threshold typically set as the *median* absolute deviation of $y$

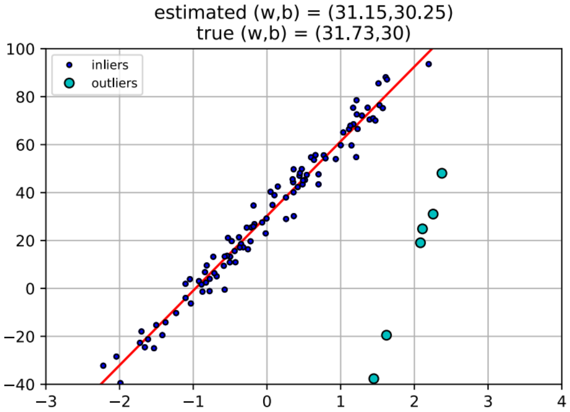
## Number of Iterations $T$

- $T$ increases steeply with $s$
  - Setting $\delta = 0.99$...



|  | proportion of outliers $e$ |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| s | 5% | 10% | 20% | 25% | 30% | 40% | 50% |
| 2 | 2 | 3 | 5 | 6 | 7 | 11 | 17 |
| 3 | 3 | 4 | 7 | 9 | 11 | 19 | 35 |
| 4 | 3 | 5 | 9 | 13 | 17 | 34 | 72 |
| 5 | 4 | 6 | 12 | 17 | 26 | 57 | 146 |
| 6 | 4 | 7 | 16 | 24 | 37 | 97 | 293 |
| 7 | 4 | 8 | 20 | 33 | 54 | 163 | 588 |
| 8 | 5 | 9 | 26 | 44 | 78 | 272 | 1177 |

## Example


estimated (w,b) = (31.15,30.25)
true (w,b) = (31.73,30)
● inliers
● outliers
100
80
60
40
20
0
−20
−40
−3
−2
−1
0
1
2
3
4

## Outline

## Non-Linear Regression

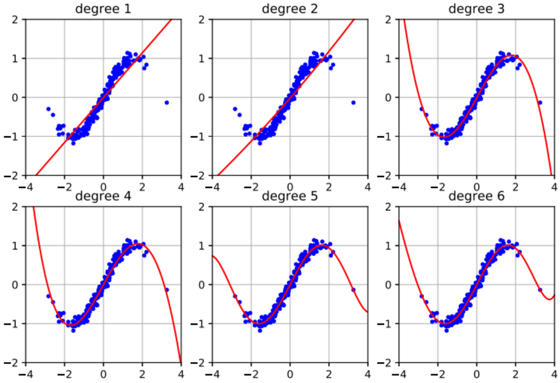
- So far we have only considered linear regression $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$
- Similar to classification, we can do non-linear regression by performing feature mapping $\phi(\mathbf{x})$, followed by linear regression on the new feature vector

## Polynomial Regression

- $p$-th order polynomial function
  - $f(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_p x^p$
- Collect the terms into a vector
  - $$f(x) = \begin{bmatrix} w_0 & w_1 & w_2 & \cdots & w_p \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^p \end{bmatrix} = \mathbf{w}^T \phi(x)$$
    - Weight vector $\mathbf{w} = [w_0, w_1, w_2, \ldots, w_p]^T$
    - Polynomial feature vector $\phi(x) = [1, x, x^2, \ldots, x^p]^T$
- Now it's a linear function, so we can use the same linear regression!

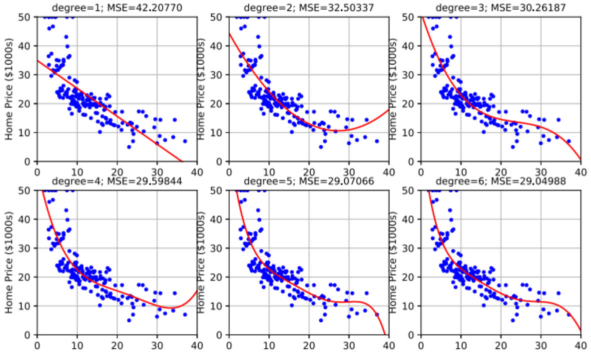
## Example

- Consider $y = \sin(x) + 0.1\epsilon$, where $x, \epsilon \sim \mathcal{N}(0, 1)$
  - Fit 1st to 6th order polynomials



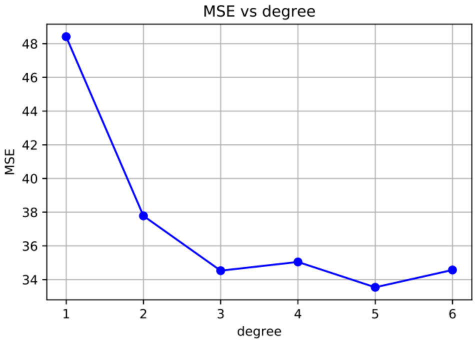
## Example: Boston Data

- Use "percentage of lower-status" feature
- Increase polynomial degree *d* will decrease MSE on training data
    - More complicated model always fits data better
    - But, it could overfit



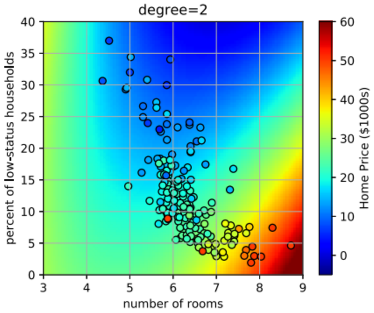
## Select Degree Using Cross-Validation

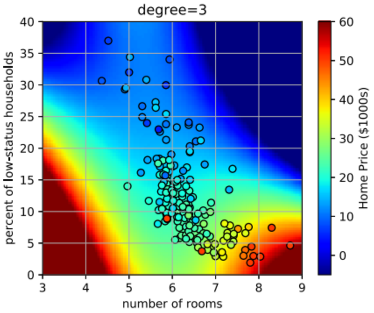
- Minimizing the MSE on the training set will overfit
  - More complex function always has lower MSE on training set
- Use cross-validation to select the proper model
  - The parameter to adjust is in feature transformation step



## Polynomial Features: 2D Example

- 2D feature: $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$
- Degree 2: $\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 & x_1 x_2 & x_2^2 \end{bmatrix}^T$
- Degree 3: $\phi(\mathbf{x}) = \begin{bmatrix} x_1^3 & x_1^2 x_2 & x_1 x_2^2 & x_2^3 \end{bmatrix}^T$





## Kernel Trick for Machine Learning Methods

- **Kernel trick** revisited
  - Kernel function $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$
    - Much less expensive to compute than explicitly calculating the high dimensional feature vector and the inner product
- Many learning methods depend on computing inner products between features (e.g., linear regression)
- For those methods, we can use a kernel function in place of inner product (i.e., "kernelizing" the methods), and thus introduce nonlinearity
- Instead of first transforming the original features into the new feature space and then computing the inner product, we can compute the inner product in the new feature space directly through the kernel function

## Kernel Ridge Regression

- Recall ridge regression: $\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|_2^2 + \frac{\alpha}{2}\|\mathbf{w}\|_2^2$

$$\mathbf{w}^\star = (\mathbf{X}^T\mathbf{X}+\alpha\mathbf{I}_{N+1})^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T+\alpha\mathbf{I}_M)^{-1}\mathbf{y},$$

- This is achieved by making use of the matrix identity

$$(\mathbf{P}^{-1}+\mathbf{B}^T\mathbf{R}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{R}^{-1} = \mathbf{P}\mathbf{B}^T(\mathbf{B}\mathbf{P}\mathbf{B}^T+\mathbf{R})^{-1}$$

  - $\mathbf{P} = \frac{1}{\alpha}\mathbf{I}_{N+1}$, $\mathbf{B} = \mathbf{X}$, and $\mathbf{R} = \mathbf{I}_M$
- This equation can be rewritten as $\mathbf{w}^\star = \sum_{i=1}^{M}\lambda_i\mathbf{x}^{(i)}$ with $\boldsymbol{\lambda} = (\mathbf{X}\mathbf{X}^T+\alpha\mathbf{I}_M)^{-1}\mathbf{y}$.
  - I.e., the solution $\mathbf{w}^\star$ must lie in the span of the data cases
- We can arrive at the same solution using Lagrange duality, similar as SVM

## Kernel Ridge Regression

- For any new point $\mathbf{x}^*$, we make prediction by

$$y^* = (\mathbf{x}^*)^T\mathbf{w} = (\mathbf{x}^*)^T\mathbf{X}^T(\mathbf{X}\mathbf{X}^T+\alpha\mathbf{I}_M)^{-1}\mathbf{y} = (\mathbf{X}\mathbf{x}^*)^T(\mathbf{X}\mathbf{X}^T+\alpha\mathbf{I}_M)^{-1}\mathbf{y}$$

- Kernelizing the method, we have

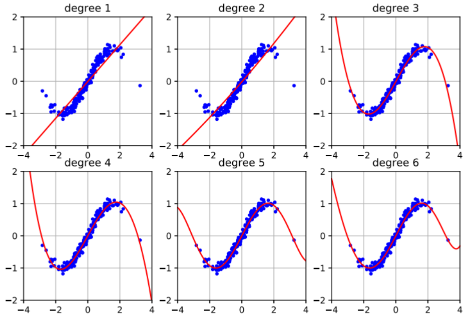
$$y^* = (\mathbf{x}^*)^T\mathbf{w} = (\mathbf{k}^*)^T(\mathbf{K}+\alpha\mathbf{I}_M)^{-1}\mathbf{y}$$

  - $K_{ij} = \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^T\phi(\mathbf{x}^{(j)})$ - the kernel matrix ($M \times M$)
  - $k_i^* = \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^*)$ - vector containing the kernel values between $\mathbf{x}^*$ and all training points $\mathbf{x}^{(i)}$

- No extra computational burden is incurred, but it now becomes a (much more powerful) nonlinear regression model

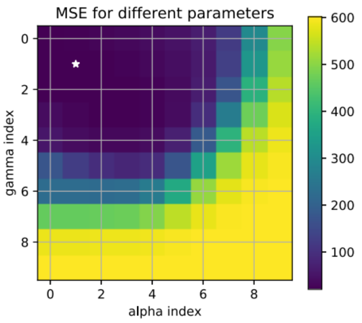
## Example: Polynomial Kernel

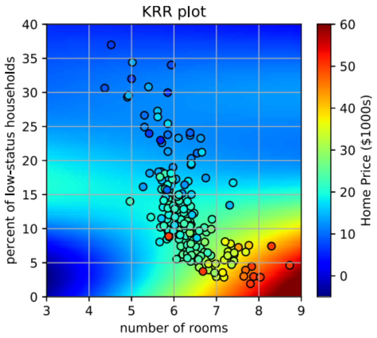
- Note: it's the same as using polynomial features and linear regression!
  - Using the kernel, we don't need to explicitly calculate the polynomial features
  - But, we do need to calculate the kernel function between all pairs of training points



## Boston Data: Cross-Validation

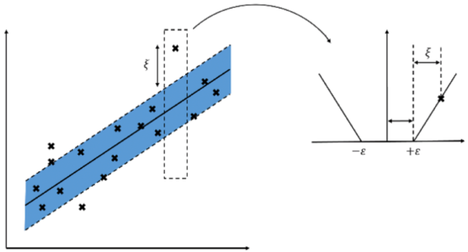
- RBF kernel
  - Cross-validation to select $\alpha$ and $\gamma$





## Support Vector Regression

- Borrow ideas from classification
  - Suppose we form a "tube" around the function:
    - If a point is inside, then it is "correctly" predicted
    - If a point is outside, then it is "incorrectly" predicted
  - Allow some points to be outside the "tube"
    - Penalty of point outside tube can be controlled by the hyperparameter $C$



## SVR (Primal Form, Soft-Margin)

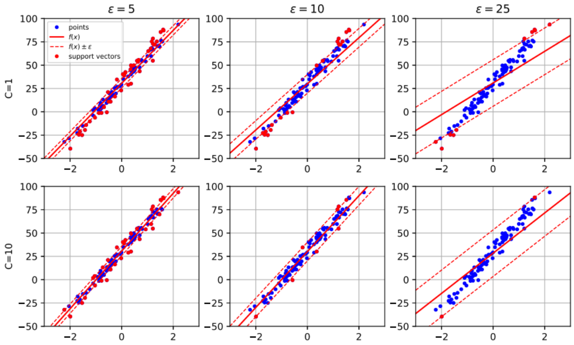
$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{M}\left|y^{(i)} - (\mathbf{w}^T\mathbf{x}^{(i)} + b)\right|_\epsilon$$

- Epsilon-insensitive error: $|z|_\epsilon = \begin{cases} 0, & |z| \leq \epsilon \\ |z| - \epsilon, & |z| > \epsilon \end{cases}$
- Equivalently, we have the SVR objective function:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad & \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{M}(\xi_i + \xi_i^*) \\ \text{subject to} \quad & y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)} - b \leq \epsilon + \xi_i,\ i = 1, \ldots, M \\ & \mathbf{w}^T\mathbf{x}^{(i)} + b - y^{(i)} \leq \epsilon + \xi_i^*,\ i = 1, \ldots, M \\ & \xi_i, \xi_i^* \geq 0,\ i = 1, \ldots, M \end{aligned}$$

## Different Tube Widths

- Similar to SVM classifier, the points on the tube or outside the tube are the support vectors



## Kernel SVR

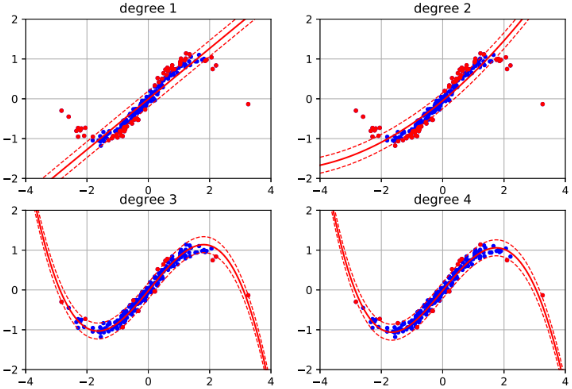
- SVR can also be kernelized through Lagrange duality
  - Turn linear regression to non-linear regression
- SVR (dual form):

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -\frac{1}{2}\sum_{i,j=1}^{M}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)(\mathbf{x}^{(i)})^T\mathbf{x}^{(j)} \\
& -\epsilon\sum_{i=1}^{M}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{M} y^{(i)}(\alpha_i - \alpha_i^*) \\
\text{subject to} \quad & \sum_{i=1}^{M}(\alpha_i - \alpha_i^*) = 0 \\
& 0 \leq \alpha_i, \alpha_i^* \leq C, \ i = 1, \ldots, M
\end{aligned}
$$

  - $\alpha_i$ and $\alpha_i^*$ are dual variables

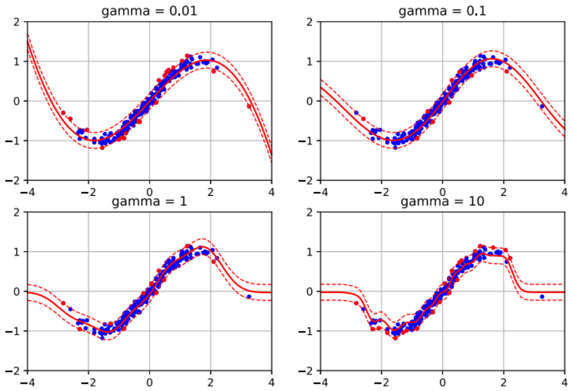
## Example: Polynomial Kernel

- Consider $y = \sin(x) + \epsilon$



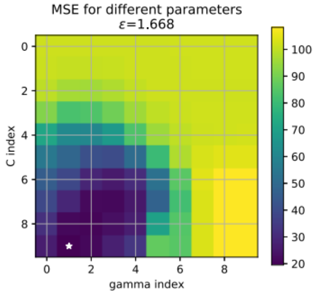
## Example: RBF Kernel

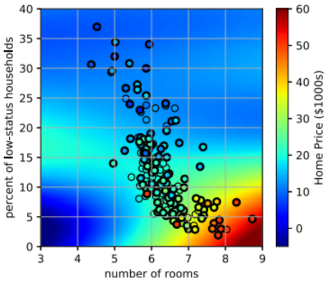
- Consider $y = \sin(x) + \epsilon$



## Example: Boston Data

- Cross-validation to select 3 parameters
  - $C$, $\gamma$, $\epsilon$





## Outline

## Regression Summary

- **Regression task**:
  - Observation $\mathbf{x}$: typically a real vector of feature values, $\mathbf{x} \in \mathbb{R}^N$
  - $y \in \mathbb{R}$: a real number
  - **Goal**: given an observation $\mathbf{x}$, predict $y$
- **Ordinary least squares**
  - **Function**: linear
  - **Training**: minimize squared error
  - **Pros**: closed-form solution
  - **Cons**: sensitive to outliers and overfitting
- **Ridge regression**
  - **Function**: linear
  - **Training**: minimize squared error with $\|\mathbf{w}\|_2^2$ regularization term
  - **Pros**: closed-form solution; shrinkage to prevent overfitting
  - **Cons**: sensitive to outliers

## Regression Summary

- **LASSO**
  - **Function**: linear
  - **Training**: minimize squared error with $\|\mathbf{w}\|_1$ regularization term
  - **Pros**: feature selection (by forcing weights to zero)
  - **Cons**: sensitive to outliers
- **RANSAC**
  - **Function**: same as the base model
  - **Training**: randomly sample subset of training data and fit model; keep model with most inliers
  - **Pros**: ignore outliers
  - **Cons**: require enough repeats to find good consensus set

## Regression Summary

- **Kernel ridge regression**
  - **Function**: non-linear (kernel function)
  - **Training**: apply “kernel trick” to ridge regression
  - **Pros**: non-linear regression; closed-form solution
  - **Cons**: require calculating kernel matrix ($\mathcal{O}(M^2)$); cross-validation to select hyperparameters
- **Kernel support vector regression**
  - **Function**: non-linear (kernel function)
  - **Training**: minimize epsilon-error
  - **Pros**: non-linear regression; faster predictions than kernel ridge regression
  - **Cons**: require calculating kernel matrix ($\mathcal{O}(M^2)$); iterative solution (slow); cross-validation to select hyperparameters

## Other Things

- **Feature normalization**
  - Feature normalization is typically required for regression methods with regularization
  - Makes ordering of weights more interpretable (LASSO, RR)
- **Output transformations**
  - Sometimes the output values $y$ have a large dynamic range (e.g., $10^{-1}$ to $10^5$)
    - Large output values will have large error, which will dominate the training error
  - In this case, it is better to transform the output values using the logarithm function
    - $\hat{y} = \log_{10}(y)$