Machine Learning

Lecture 02: Linear Regression and Basic ML Issues

Nevin L. Zhang
lzhang@cse.ust.hk

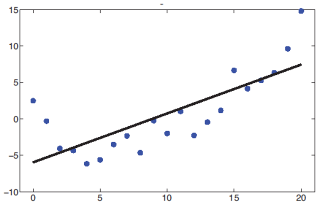
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

This set of notes is based on internet resources and
KP Murphy (2012). Machine learning: a probabilistic perspective. MIT Press. (Chapter 7)
Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT press. www.deeplearningbook.org. (Chapter 5)
Andrew Ng. Lecture Notes on Machine Learning. Stanford.

Outline

Linear Regression: Problem Statement

- Given: A **training set** $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$
  - Each $\mathbf{x}_i$ is a D-dimensional real-valued column vector: $\mathbf{x} = (x_1, \ldots, x_D)^\top$.
  - Each y_i is a real number
- To Learn: $$y = f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} = \sum_{j=0}^{D} w_j x_j$$
  - The **weights** $\mathbf{w} = (w_0, w_1, \ldots, w_D)^\top$ determine how important the features $(x_1, \ldots, x_D)$ are in predicting the response y.
  - Always set $x_0 = 1$, and w_0 is the **bias** term. Often it is denoted by b.

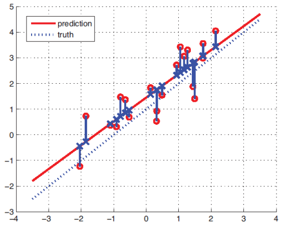


Linear Regression: Examples

Here are several examples from
`http://people.sc.fsu.edu/∼jburkardt/datasets/regression/regression.html`

- Predict brain weight of mammals based their body weight (x01.txt)
- Predict blood fat content based on age and weight (x09.txt)
- Predict death rate from cirrhosis based on a number of other factors (x20.txt)
- Predict selling price of houses based on a number of factors (X27.txt)

Linear Regression: Mean Squared Error

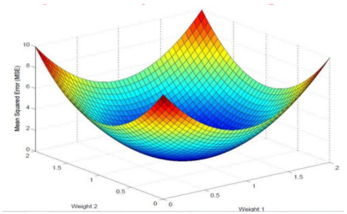


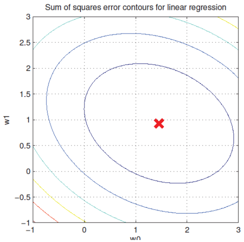
- How to determine the weights **w**? We want the predicted response values $f(\mathbf{x}_i)$ to be close to the observed response values. So, we want to minimize the following objective function:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2.$$

- This is called **mean squared error (MSE)**.

Linear Regression: Mean Squared Error





- As a function of the weights **w**, MSE is a quadratic "bowl" with a unique minimum.
- We can minimizing it by setting its gradient to 0

$$\nabla J(\mathbf{w}) = 0$$

Linear Regression: Matrix Representation

- We can represent the **training set** $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ using the **design matrix** $\mathbf{X}$ and a column vector $\mathbf{y}$.

$$\mathbf{X} = \begin{bmatrix} x_{1,0} & x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,0} & x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{N,0} & x_{N,2} & x_{N,2} & \cdots & x_{N,D} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \cdots \\ \mathbf{x}_N^\top \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}$$

- Then, the MSE can be written as follows

$$J(\mathbf{w}) = \frac{1}{N}||\mathbf{y}-\mathbf{X}\mathbf{w}||_2^2 = \frac{1}{N}(\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) = \frac{1}{N}(\mathbf{w}^\top(\mathbf{X}^\top\mathbf{X})\mathbf{w}-2\mathbf{w}^\top(\mathbf{X}^\top\mathbf{y})+\mathbf{y}^\top\mathbf{y})$$

Linear Regression: The Normal Equation

- From the equation of the previous slide, we get (see Murphy Chapter 7) that

$$\nabla J(\mathbf{w}) = \frac{1}{N}(2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y})$$

- Setting the gradient to zero, we get the **normal equation**

$$\mathbf{X}^\top \mathbf{X}\mathbf{w} = \mathbf{X}^\top \mathbf{y}$$

- The value of **w** that minimizes $J(\mathbf{w})$ is

$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

  This is called the **ordinary least squares (OLS)** solution.

Outline

Probabilistic Interpretation

- Next, we show that least squares regression can be derived from a probabilistic model.
- We assert

$$y = \mathbf{w}^\top \mathbf{x} + \epsilon = \sum_{j=0}^{D} w_j x_j + \epsilon$$

  where the error term ϵ captures unmodeled effects and random noise.
- We also assume that ϵ follow the Gaussian distribution with zero mean and variance σ: $\epsilon \sim \mathcal{N}(0, \sigma^2)$.
- The model parameters θ include **w** and σ. The conditional distribution of y given input **x** and parameters θ is a Gaussian

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(y|\mu(\mathbf{x}), \sigma^2)$$

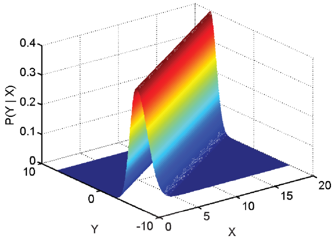
  where $\mu(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$

Probabilistic Interpretation

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(y|\mu(\mathbf{x}), \sigma^2)$$

- For each input $\mathbf{x}$, we get a distribution of y, which is a Gaussian distribution.
- To get a **point estimation** of y, we can use the mean, i.e.,

$$\hat{y} = \mu(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$



$$p(y|x, \boldsymbol{\theta}) = \mathcal{N}(y|w_0 + w_1 x, \sigma^2)$$

Parameter Estimation

- Determine $\theta = (\mathbf{w}, \sigma)$ by minimizing the cross entropy:

$$\begin{aligned} -\frac{1}{N}\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \theta) &= -\frac{1}{N}\sum_{i=1}^{N} \log[\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma^2})] \\ &= \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{N2\sigma^2}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2 \end{aligned}$$

- Assume σ is fixed. This is the same as minimizing the MSE

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2$$

- **Summary**: Under some assumptions, least-squares regression can be justified as a very natural method that minimizes cross entropy, or maximize likelihood.

Outline

Beyond Linear Regression

- Here again is the linear regression model:

$$y = f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} = \sum_{j=0}^{D} w_j x_j$$

- Linear regression can be made to model non-linear relationships by replacing $\mathbf{x}$ with some non-linear function of the inputs, $\phi(\mathbf{x})$. That is, we use

$$y = f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x})$$

  This is known as **basis function expansion** and ϕ is called **feature mapping**.

Polynomial Regression

- For $\mathbf{x} = [1, x_1, x_2]^\top$, we can use the following **polynomial feature mapping**

$$\phi(\mathbf{x}) = [1, x_1, x_2, \ldots, x_D, x_1^2, x_1x_2, \ldots, x_D^d]^\top$$

- When and $d = 2$, we get **polynomial regression**.

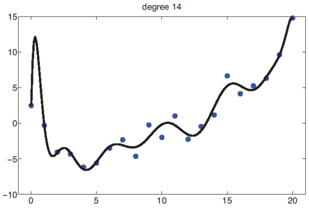
$$y = \mathbf{w}^\top \phi(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_1x_2 + w_5x_2^2$$

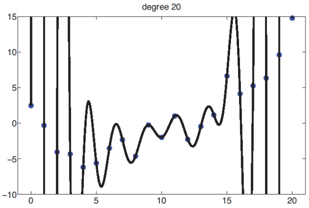
- **Model selection**: What d to choose? What is the impact of d?

Outline

Hypothesis Space and Capacity

- The **hypothesis space** of a machine learning algorithm/model is the set of functions that it is allowed to select as being the solution.
- The "size" of the hypothesis space is called the **capacity** of the model.
- For polynomial regression, the larger the d, the higher the model capacity.
- Higher model capacity implies better fit to *training data*.
    - Two examples with $d = 14$ and 20 and one feature $\mathbf{x} = (x)$.





Generalization Error

- An machine learning model is trained to perform well on the training example. But it is not really what we care about.
- What we really care about is that it must perform well on new and previously unseen examples. This is called **generalization**.
- We use a the error on a **test set** to measure how well a model generalize:

$$J^{(test)}(\mathbf{w}) = \frac{1}{N^{(test)}}||\mathbf{y}^{(test)} - \mathbf{X}^{(test)}\mathbf{w}||_2^2$$

  This is called the **test error** or the **generalization error**

- In contrast, here is the **training error** we have been talking about so far:

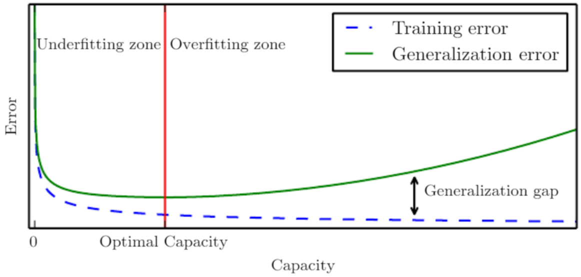
$$J^{(train)}(\mathbf{w}) = \frac{1}{N^{(train)}}||\mathbf{y}^{(train)} - \mathbf{X}^{(train)}\mathbf{w}||_2^2$$

Test and Training Error

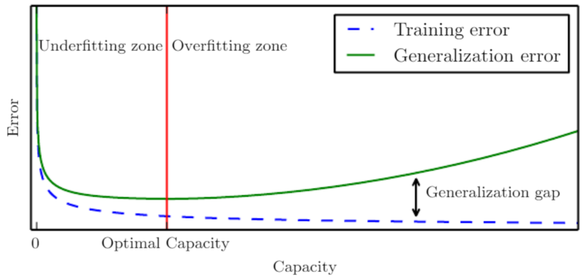
- The test and training errors are related because we assume both training and test data are iid samples of an underlining data generation process $p(\mathbf{x}, y)$.
- However, **small training error does not always imply small generalization error.**
- The generalization error is usually larger than training error because the model parameters are selected to minimizing the training error.
- So, we need to
    - Make the training error small, and
    - Make the gap between the test and training error small.

Overfitting and Underfitting

- Training and test error behave differently as model capacity increases.
- At the left end of the graph, training error and generalization error are both high. This is the **underfitting regime**.
- As we increase capacity, training error decreases, but the gap between training and generalization error increases. Eventually,the size of this gap outweighs the decrease in training error, and we enter the **overfitting regime**, where capacity is too large.



Overfitting and Underfitting



- **Choosing a model with the appropriate capacity is important**.
  - This can be achieved by either validation or regularization.

Validation

- Model capacity is usually determined by **hyperparameters** such as the order d of polynomial in polynomial regression.
- **Validation** is a common method for determining the values of hyperparameters such as d:
  - Randomly divide the training set into two disjoint subsets.
  - One subset is still called the training set, and the other called the **validation set** or **held-out set**.
  - To determine the value of d:
    - Try a set of possible values.
    - For each possible value of d, train the model on the training set, and measure the error on the validation set. This is called the **validation error**.
    - Pick the value that has the minimum validation error.

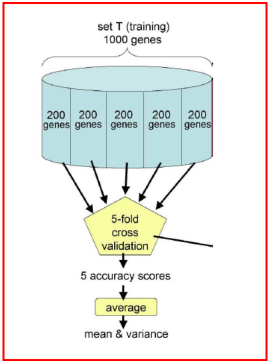
Validation

How to divide training data into training set and validation set?

- Generally,
  - the larger the training set, the better the hypothesis (i.e., $y = f(\mathbf{x})$)
  - the larger the validation set, the more accurate the validation error. estimation
- Typically, withhold 20% of the available examples for the validation set, using the other two-thirds for training

Cross Validation

- When data is limited, withholding part of it for validation set reduces even further the number of examples available for training, and error estimates can have large variance.
- An alternative is to use **cross validation**



1. The N available examples are partitioned into k disjoint subsets, each of size N/k
2. The learning procedure is then run k times, each time
   - using one of these subsets as the validation set, and
   - combining the other subsets for the training set
3. Average the performance on the validation sets over the k runs
4. Typically $k = 10$.

Regularization

- Instead of using validation to pick an appropriate value for the order d of polynomial, we can start with a large d, and hence a large hypothesis space.
- Then we use **regularization** to pick an **appropriate** solution from that space so as to avoid overfitting.
- Some setup:
  - Suppose the non-linear transformation $\phi(\mathbf{x})$ has K components
    - Example: In $y = f(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_1x_2 + w_5x_2^2$, we have $k = 5$ and $\phi = (x_1, x_2, x_1^2, x_1x_2, x^2)$.
  - Let $\mathbf{w} = (w_1, w_2, \ldots, w_K)^\top$. The bias w_0 is separated from $\mathbf{w}$.

Regularization

- The error function without regularization is

$$J(\mathbf{w}, w_0) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + \mathbf{w}^\top \phi(\mathbf{x}_i)))^2$$

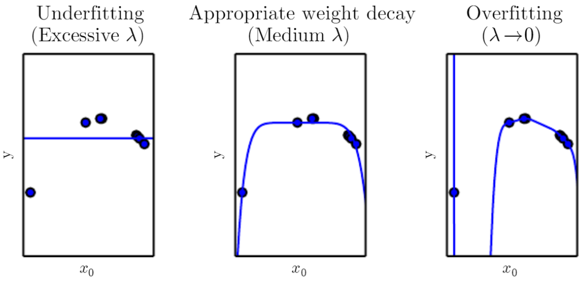
- The error function **with regularization** is

$$J(\mathbf{w}, w_0) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + \mathbf{w}^\top \phi(\mathbf{x}_i)))^2 + \lambda||\mathbf{w}||_2^2$$

- $\lambda \geq 0$ is a hyperparameter chosen ahead of time that controls the strength of our preference for smaller weights.
- Minimizing $J(\mathbf{w}, w_0)$ gives us a solution that puts **significant weights** on a small number of features. This is called **weight decay**.
- This way, we get a solution that **effectively** uses a small number of features and hence does not suffer from overfitting.
- Note that w_0 is not regularized as it does not influence model complexity.

Regularization: Example

- The true function is quadratic, and we polynomials with degree 9.
- RIGHT: With λ approaching zero, the degree-9 polynomial overfits significantly.
- LEFT: With very large λ, we can force the model to learn a function with no slope at all.
- CENTER: With a medium value of λ, the learning algorithm recovers a curve with the right general shape.



Regularization: Solution

- Regression using the following error function is called **ridge regression** or **penalized least squares**.

$$J(\mathbf{w}, w_0) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + \mathbf{w}^\top \phi(\mathbf{x}_i)))^2 + \lambda||\mathbf{w}||_2^2$$

- The **penalized least squares** solution is

$$\hat{\mathbf{w}}_{ridge} = (\lambda \mathbf{I}_K + \mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y},$$

  where $\mathbf{I}_K$ is the k-dimensional identity matrix. The larger the regularization constant λ, the smaller the weights.

- Compare this with the ordinary least squares solution:

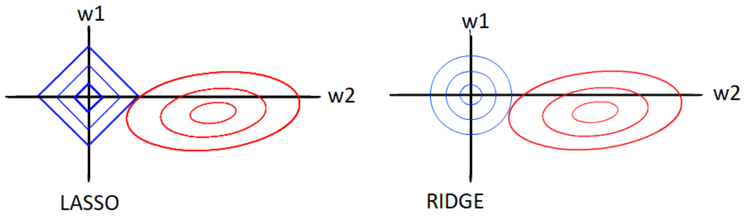
$$\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$$

Regularization: LASSO

- By using L2 regularization, Ridge regression **shrinks large regression coefficients** in order to reduce overfitting.
- In contrast, LASSO (least absolute shrinkage and selection operator) **forces certain coefficients to zero**, and thereby chooses a **sparser model** that uses only a subset of the features.
- LASSO uses L1 regularization:

$$J(\mathbf{w}, w_0) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + \mathbf{w}^\top \phi(\mathbf{x}_i)))^2 + \lambda||\mathbf{w}||_1$$

LASSO vs Ridge



- Red circles represent contours of the error function *error*($\mathbf{w}$); Blue lines represent contours of the regularization term *regularization*($\mathbf{w}$); Minimum is at the center.
- With Lasso, the sum *error*($\mathbf{w}$) + *regularization*($\mathbf{w}$) usually achieves minimum at some corners, which lie on the axes.This means some of the weights are set to 0, and the corresponding features not used.
- With Ridge regression, the minimum is usually not achieved on the axes. Weights are seldom 0.