Regression Tasks - Linear Regression

Astrini Sie

asie@seattleu.edu

Week 2

ECEGR4750 - Introduction to Machine Learning
Seattle University

October 12, 2023

Recap and Updates

Tuesday

- Stable Diffusion demo

Thursday

- OLS derivation, notation, and x_0
- Logistical updates: office hours, homework/lab, next week
- Communication method

Recap and Updates

1. Office Hours
   - T, Th 12-1p at Bannan 224
   - W 7-9p via Zoom
   - F 9-9.45a via Zoom
2. Syllabus to be posted this week
3. First lab homework to be posted by Friday
4. Communication: enable Canvas notifications
5. Next Week: Guest Lecture

Overview

1. Supervised Learning
   - Definition
   - Terminologies
2. Regression
   - Case Study: Input, Model, Cost Function
   - Optimization: Ordinary Least Squares (OLS)
   - Optimization: Least Mean Squares (LMS)
     - Gradient Descent
     - Batch Gradient Descent
     - Stochastic Gradient Descent
     - Stochastic Minibatch Gradient Descent
3. Case Study, Revisited
4. Summary of Linear Regression

Supervised Learning

Goal of supervised learning

Given training set with known features and labels, produce a prediction function

Supervised Learning

Goal of supervised learning

Given training set with known features and labels, produce a prediction function

- During training: given input data ('**training set**') with **features** and **labels**, learn the relationship ('**prediction function**') between them
- During inference: given a brand new data with **features**, use the **prediction function** to predict the **labels**

Supervised Learning

Terminologies:

- Input / **Training Set** consists of feature and label pairs:

  Features: $x^{(1)}, x^{(2)}, \cdots, x^{(n)}$. $x^{(i)} \in X$.
  Labels: $y^{(1)}, y^{(2)}, \cdots, y^{(n)}$. $y^{(i)} \in Y$.
  for $i = 1, \cdots, n$, where n is the total number of training samples.

Supervised Learning

Terminologies:

- Input / **Training Set** consists of feature and label pairs:

  Features: $x^{(1)}, x^{(2)}, \cdots, x^{(n)}$. $x^{(i)} \in X$.
  Labels: $y^{(1)}, y^{(2)}, \cdots, y^{(n)}$. $y^{(i)} \in Y$.
  for $i = 1, \cdots, n$, where n is the total number of training samples.

- Model / **Prediction Function**:

  A function $h_\theta(x) : X \rightarrow Y$ that maps the input features X to the output values Y, where θ is the **parameter** or **weight** of the model. In the training process, we *learn* the values of θ that results in good predictions.

Supervised Learning

Terminologies:

- Input / **Training Set** consists of feature and label pairs:

  Features: $x^{(1)}, x^{(2)}, \cdots, x^{(n)}$. $x^{(i)} \in X$.
  Labels: $y^{(1)}, y^{(2)}, \cdots, y^{(n)}$. $y^{(i)} \in Y$.
  for $i = 1, \cdots, n$, where n is the total number of training samples.

- Model / **Prediction Function**:

  A function $h_\theta(x) : X \rightarrow Y$ that maps the input features X to the output values Y, where θ is the **parameter** or **weight** of the model. In the training process, we *learn* the values of θ that results in good predictions.

- Output / **Prediction**:

  Given an unseen data point $x^{(k)}$, predict the output $y^{(k)}$ based on prediction function h.

Supervised Learning

Terminologies

- Input / **Training Set** consists of feature and label pairs:
  Features: $x^{(1)}, x^{(2)}, \cdots, x^{(n)}$. $x^{(i)} \in X$.
  Labels: $y^{(1)}, y^{(2)}, \cdots, y^{(n)}$. $y^{(i)} \in Y$.
  for $i = 1, \cdots, n$, where n is the total number of training samples.
- Model / **Prediction Function**:
  A function $h_\theta(x) : X \rightarrow Y$ that maps the input features X to the output values Y, where θ is the **parameter** or **weight** of the model. In the training process, we *learn* the values of θ that results in good predictions.
- Output / **Prediction**:
  Given an unseen data point $x^{(k)}$, predict the output $y^{(k)}$ based on prediction function h.

- if Y is continuous, it will be a regression task
- if Y is discrete, it will be a classification task

Supervised Learning

There are two types of supervised learning tasks:

Regression

if Y is continuous

- Estimating the relationships between a dependent variable ('**label**') and one or more independent variables ('**features**').
- Example: X is data of house dimensions and locations, predict Y the price of the house.

Classification

if Y is discrete

- Categorizing a given set of input data into categories ('**classes**') based on one or more variables ('**features**').
- Example: X is an image, predict if Y is a "cat" or a "dog".

Supervised Learning

There are two types of supervised learning tasks:

Regression

if Y is continuous

- Estimating the relationships between a dependent variable ('**label**') and one or more independent variables ('**features**').
- Example: X is data of house dimensions and locations, predict Y the price of the house.

Classification

if Y is discrete

- Categorizing a given set of input data into categories ('**classes**') based on one or more variables ('**features**').
- Example: X is an image, predict if Y is a "cat" or a "dog".

Input

Regression Sample Case: Predicting House Data

1) Input: feature $x = \textit{size}$ and label $y = \textit{price}$, with $n = 3$ training samples.

	Size
$x^{(1)}$	2104
$x^{(2)}$	2500
$x^{(3)}$	1600

	Price
$y^{(1)}$	400K
$y^{(2)}$	900K
$y^{(3)}$	330K

Input

Regression Sample Case: Predicting House Data

1) Input: feature $x = \textit{size}$ and label $y = \textit{price}$, with $n = 3$ training samples.

	Size
$x^{(1)}$	2104
$x^{(2)}$	2500
$x^{(3)}$	1600

	Price
$y^{(1)}$	400K
$y^{(2)}$	900K
$y^{(3)}$	330K

2) Model / Prediction Function: a linear model

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)}$$

2) Model / Prediction Function: a linear model

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)}$$

Using the convention of $x_0 = 1$, we can rewrite the prediction function as:

$$h_\theta(x^{(i)}) = \sum_{j=0}^{d} \theta_j^{(i)} x_j^{(i)}$$

A note on notation: the superscript (i) denotes the index of the training sample, and the subscript j denotes index of the feature. For example, $x_3^{(1)}$ means feature 3 from training sample 1.

2) Model / Prediction Function: a linear model

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)}$$

Using the convention of $x_0 = 1$, we can rewrite the prediction function as:

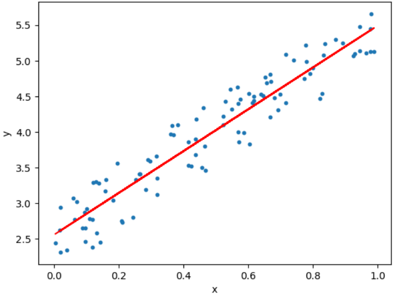
$$h_\theta(x^{(i)}) = \sum_{j=0}^{d} \theta_j^{(i)} x_j^{(i)}$$

Goal

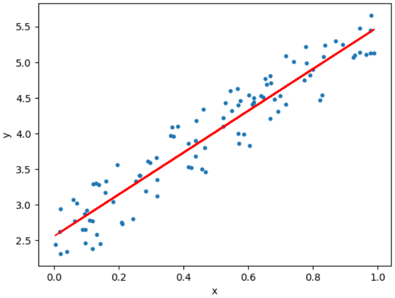
Learn the values of θ that results in **good prediction**.

What is a good prediction?

What is a good prediction?



What is a good prediction?



Calculate the distance from each data point (label) $y^{(i)}$ to the regression line (prediction) $h_\theta(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)}$, square it, and sum all of the squared errors together. We call this the **cost function** $J(\theta)$.

Cost Function

Regression Sample Case: Predicting House Data

What is a good prediction? A good prediction is the one that minimizes the cost function $J(\theta)$.

How do you define a cost function?

Cost Function

Regression Sample Case: Predicting House Data

What is a good prediction? A good prediction is the one that minimizes the cost function $J(\theta)$.

How do you define a cost function? Case by case basis. In this example, let's choose **sum of squared errors** as our cost function.

Cost Function

Regression Sample Case: Predicting House Data

What is a good prediction? A good prediction is the one that minimizes the cost function $J(\theta)$.

How do you define a cost function? Case by case basis. In this example, let's choose **sum of squared errors** as our cost function.

2a) Cost Function:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

Where n is the total number of training samples. Choose θ such that:

$$\theta = \underset{\theta}{\operatorname{argmin}} J(\theta)$$

Solving the Cost Function

Regression Sample Case: Predicting House Data

How do we solve for θ that minimizes this cost function?
How do we solve for this minimization / optimization problem?

1. **Ordinary Least Squares (OLS)**. Analytical solution (closed form).
2. **Least Mean Squares (LMS)**. Numerical solution (approximation).

Ordinary Least Squares (OLS)

Optimization

Choose θ that minimizes $J(\theta)$ using OLS solution. This solution provides a closed form solution of θ in terms of the known variables x and y.

$$\text{Cost Function: } J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2 \tag{1}$$

We can simplify the expression of the cost function in matrix from using the rule $z^T z = \sum_i z_i^2$:

Ordinary Least Squares (OLS)

Optimization

Choose θ that minimizes $J(\theta)$ using OLS solution. This solution provides a closed form solution of θ in terms of the known variables x and y.

$$\text{Cost Function: } J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2 \tag{2}$$

We can simplify the expression of the cost function in matrix from using the rule $z^T z = \sum_i z_i^2$:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2 \tag{3}$$

$$J(\theta) = \frac{1}{2} (H - Y)^T (H - Y) \tag{4}$$

Ordinary Least Squares (OLS)

Optimization

Let's define x, y, and $h_\theta(x)$ in matrix forms:

Let's define $\boldsymbol{x}$, y, and $h_\theta(x)$ in matrix forms:

	Size
$x^{(1)}$	2104
$x^{(2)}$	2500
$x^{(3)}$	1600

$$x_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1^{(1)} \\ x_1^{(2)} \\ x_1^{(3)} \end{bmatrix} \longrightarrow \begin{bmatrix} x_1^{(1)} \\ x_1^{(2)} \\ \vdots \\ x_1^{(n)} \end{bmatrix}, \qquad x_1 = \begin{bmatrix} 2104 \\ 2500 \\ 1600 \end{bmatrix} = \begin{bmatrix} x_1^{(1)} \\ x_1^{(2)} \\ x_1^{(3)} \end{bmatrix} \longrightarrow \begin{bmatrix} x_1^{(1)} \\ x_1^{(2)} \\ \vdots \\ x_1^{(n)} \end{bmatrix}$$

Let's define $\boldsymbol{x}$, y, and $h_\theta(x)$ in matrix forms:

	Size	Bedrooms	Lot Size
$x^{(1)}$	2104	4	4500
$x^{(2)}$	2500	3	3000
$x^{(3)}$	1600	3	3000

$$x_2 = \begin{bmatrix} 4 \\ 3 \\ 3 \end{bmatrix} = \begin{bmatrix} x_2^{(1)} \\ x_2^{(2)} \\ x_2^{(3)} \end{bmatrix} \longrightarrow \begin{bmatrix} x_2^{(1)} \\ x_2^{(2)} \\ \vdots \\ x_2^{(n)} \end{bmatrix}, \qquad x_3 = \begin{bmatrix} 4500 \\ 3000 \\ 3000 \end{bmatrix} = \begin{bmatrix} x_3^{(1)} \\ x_3^{(2)} \\ x_3^{(3)} \end{bmatrix} \longrightarrow \begin{bmatrix} x_3^{(1)} \\ x_3^{(2)} \\ \vdots \\ x_3^{(n)} \end{bmatrix}$$

Ordinary Least Squares (OLS)

Optimization

Let's define $\boldsymbol{x}$, y, and $h_\theta(x)$ in matrix forms:

Combining the all the x vectors:

$$X = \begin{bmatrix} | & | & & | \\ x_0 & x_1 & \dots & x_d \\ | & | & & | \end{bmatrix} = \begin{bmatrix} — & x^{(1)} & — \\ — & x^{(2)} & — \\ & \vdots & \\ — & x^{(n)} & — \end{bmatrix} \in \mathbb{R}^{nx(d+1)}$$

where:

n = number of training samples

d = number of features

Let's define x, $\boldsymbol{y}$, and $h_\theta(x)$ in matrix forms:

	Price
$y^{(1)}$	400K
$y^{(2)}$	900K
$y^{(3)}$	890K

$$Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} \in \mathbb{R}^{nx1}$$

where:

$n =$ number of training samples

Ordinary Least Squares (OLS)

Optimization

Let's define x, y, and $\boldsymbol{h_\theta(x)}$ in matrix forms:

$$h_\theta(x^{(i)}) = \sum_{j=0}^{d} \theta_j^{(i)} x_j^{(i)}$$

In matrix form, this is equivalent to:

$$h_\theta(x^{(i)}) = (x^{(i)})^T \theta \quad (5)$$

$$H = X\Theta \quad (6)$$

Now, let's plug in H (5) into the cost function (3):

Cost Function

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

Ordinary Least Squares (OLS)

Optimization

Now, let's plug in H (5) into the cost function (3):

Cost Function

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$J(\theta) = \frac{1}{2}(H - Y)^T(H - Y)$$

$$J(\theta) = \frac{1}{2}(X\Theta - Y)^T(X\Theta - Y)$$

Ordinary Least Squares (OLS)

Optimization

Finally, let's find θ that minimizes J. To do this, find the derivative of J with respect to θ, and set it to zero.

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(X\Theta - Y)^T(X\Theta - Y)$$

Ordinary Least Squares (OLS)

Optimization

Finally, let's find θ that minimizes J. To do this, find the derivative of J with respect to θ, and set it to zero.

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(X\Theta - Y)^T(X\Theta - Y)$$

$$0 = \frac{1}{2}\nabla_\theta((X\Theta)^T - Y^T)(X\Theta - Y)$$

$$0 = \frac{1}{2}\nabla_\theta((X\Theta)^T X\Theta - (X\Theta)^T Y - Y^T(X\Theta) + Y^T Y)$$

$(ab)^T = b^T a^T$ gives us $(X\Theta)^T = \Theta^T X^T$

$a^T b = b^T a$ gives us $(X\Theta)^T Y = Y^T(X\Theta)$

$$0 = \frac{1}{2}\nabla_\theta(\Theta^T X^T X\Theta - Y^T X\Theta - Y^T X\Theta + Y^T Y)$$

Ordinary Least Squares (OLS)

Optimization

continued...

$$0 = \frac{1}{2}\nabla_\theta(\Theta^T X^T X\Theta - 2Y^T X\Theta + Y^T Y)$$

$\nabla_x b^T x = b$ gives us $\nabla_x 2Y^T X\Theta = 2(Y^T X)^T$

$\nabla_x x^T Ax = 2Ax$ for symmetric matrix A gives us $\nabla_x \Theta^T(X^T X)\Theta = 2(X^T X)\Theta$

$$0 = \frac{1}{2}(2X^T X\Theta - 2(Y^T X)^T)$$

$$0 = \frac{1}{2}(2X^T X\Theta - 2XY^T)$$

$$0 = X^T X\Theta - X^T Y$$

Solving for Θ:

$$X^T X\Theta = X^T Y$$

$$\Theta = (X^T X)^{-1} X^T Y$$

Ordinary Least Squares (OLS) Solution

The value of θ that minimizes $J(\theta)$ is given in closed form by the equation:

$$\Theta = (X^T X)^{-1} X^T Y$$

$$\Theta \in \mathbb{R}^{(1x(d+1)}$$

Ordinary Least Squares (OLS) Solution

The value of θ that minimizes $J(\theta)$ is given in closed form by the equation:

$$\Theta = (X^T X)^{-1} X^T Y$$

$$\Theta \in \mathbb{R}^{(1x(d+1))}$$

Caveat:

- This closed form solution exists only if $(X^T X)^{-1}$ exists.

Choose θ that minimizes $J(\theta)$ using LMS algorithm. This is a search algorithm in which:

1. We start with an "initial guess" θ
2. Repeatedly change θ to make $J(\theta)$ smaller
3. Converge to a value of θ that minimizes $J(\theta)$

This is called the **Gradient Descent** algorithm.

Least Mean Squares (LMS) and Gradient Descent

Optimization

Choose θ that minimizes $J(\theta)$ using LMS algorithm. This is a search algorithm in which:

1. We start with an "initial guess" θ
2. Repeatedly performs an update:
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$
   - α is the **learning rate**
   - The update is simultaneously performed $\forall j = 0, \ldots, d$, where j is the number of training samples.
   - Repeatedly takes a step in the direction of steepest decrease of J.
3. Converge to a value of θ that minimizes $J(\theta)$

This is called the **Gradient Descent** algorithm.

Least Mean Squares (LMS) and Gradient Descent

Optimization

Least Mean Squares (LMS) / Widrow-Hoff Learning Rule:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Let's solve for this rule by expanding the partial derivative, considering the case of only one training sample (x, y), thereby neglecting the sum notation in the cost function $J(\theta)$.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(h_\theta \left(x^{(i)} \right) - y^{(i)} \right)^2$$

$$J(\theta) = \frac{1}{2} \left(h_\theta \left(x \right) - y \right)^2$$

Least Mean Squares (LMS) and Gradient Descent

Optimization

continued...

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta \left(x \right) - y \right)^2$$

continued...

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta \left(x \right) - y \right)^2 \\
&= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{d} \theta_i x_i - y \right) \\
&= (h_\theta(x) - y) x_j
\end{aligned}
$$

Substituting this into the Learning Rule results in:

$$
\theta_j := \theta_j - \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}
$$

Least Mean Squares (LMS) and Gradient Descent

Optimization

Least Mean Squares (LMS) / Widrow-Hoff Learning Rule for one training sample

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\theta_j := \theta_j - \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

A couple interesting notes:

- The magnitude of the update is proportional to the error term $(y^{(i)} - h_\theta(x^{(i)}))$.
- If our prediction $h_\theta(x^{(i)})$ nearly matches the label $y^{(i)}$, there is little need to change the parameters θ.
- If the prediction $h_\theta(x^{(i)})$ has a larger error from the label $y^{(i)}$, then a larger change to the parameters θ is needed.

Batch Gradient Descent

$$\theta_j^{(t+1)} := \theta_j^{(t)} - \alpha \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

- Within a single update, all the n training samples are examined.
- In modern day applications, the number of training samples could be in the order of billions or trillions.
- Updating all training samples at a time might not be desirable or computationally possible (can't load all data in memory!)
- Use **Stochastic Gradient Descent**

Stochastic Gradient Descent

Optimization

Stochastic Gradient Descent

$$\theta_j^{(t+1)} := \theta_j^{(t)} - \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

- One training sample is updated in one iteration.
- For n training samples, we needed to perform n iterations (θ is updated n times).
- This could be painfully slow! There should be a middle ground where we can update a *batch* B of training samples together. In practice B can be a few to a few hundreds or thousands training samples, depending on the size of each of your training sample.
- This is called **Stochastic Minibatch Gradient Descent**

Stochastic Minibatch Gradient Descent

Optimization

Stochastic Minibatch Gradient Descent

$$\theta^{(t+1)} := \theta^{(t)} - \alpha_B \sum_{j \in B}^{n} \left(y^{(j)} - h_\theta(x^{(j)}) \right) x^{(j)}$$

- if $|B| = 1, \ldots, n$ (the whole training samples), then this becomes Batch Gradient Descent.
- if $B = 1$, this becomes Stochastic Gradient Descent.
- In practice, choose B proportional to what works well on modern GPUs.

More Complex Data

Regression Sample Case: Predicting House Data, Revisited

Input / Training Set: In the previous example, we only had 1 feature $x_1 = \textit{price}$. Now we are adding 2 more features $x_2 = \textit{bedrooms}$ and $x_3 = \textit{lotsize}$.

	Size	Bedrooms	Lot Size
$x^{(1)}$	2104	4	45K
$x^{(2)}$	2500	3	30K

	Price
$y^{(1)}$	400K
$y^{(2)}$	900K

Model / Prediction Function: a linear model

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

Output / Prediction: Given a new house with size $x_1^{(k)} = 2250$, bedrooms $x_2^{(k)} = 3$, and lot size $x_3^{(k)} = 39K$, predict the price $y^{(k)}$ of this house.

Summary

1. Gather the training set training set $\{(x^{(i)}, y^{(i)}$ for $i = 1, \ldots, n\}$ where $y^{(i)} \in \mathbb{R}$.
2. Define the prediction as $h_\theta(x) = \sum_j \theta_j x_j$, or in vector form $H(\Theta) = X\Theta$.
3. Solve for θ. There are two methods for solving θ
   1. Ordinary Least Squares (OLS): $\Theta = (X^T X)^{-1} X^T Y$. This is a closed form solution
   2. Least Mean Squares (LMS): $\theta_j := \theta_j - \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$. This is a numerical solution
4. In most cases, the OLS solution does not exist due to the inverse of the matrix not existing. As such, solve using the LMS approach.
5. To iterate for θ, use the gradient descent (GD) approach:
   1. Batch GD: weights for all training samples are updated at once.
   2. Stochastic GD: weight for one training sample is updated at a time.
6. Given new data, calculate prediction using the final values of θ: $Y_{pred} = H(\Theta) = X\Theta$.

References

Chris Re, Andrew Ng, and Tengyu Ma (2023)
CSE229 Machine Learning
Stanford University