Regression Tasks

Astrini Sie

asie@seattleu.edu

Week 2

ECEGR4750 - Introduction to Machine Learning
Seattle University

September 26, 2023

Recap

- Stable Diffusion demo

Overview

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 $h_\theta(x) : X \to Y$ 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 $h_\theta(x) : X \to Y$ 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 $h_\theta(x) : X \to Y$ 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 $n = 2$ 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 $x = size$ and label $y = price$, with $n = 2$ 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

$$h_\theta(x) = \theta_0 + \theta_1 x_1$$

2) Model / Prediction Function: a linear model

$$h_\theta(x) = \theta_0 + \theta_1 x_1$$

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

$$h(x) = \sum_{i=0}^{n} \theta_i x_i$$

2) Model / Prediction Function: a linear model

$$h_\theta(x) = \theta_0 + \theta_1 x_1$$

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

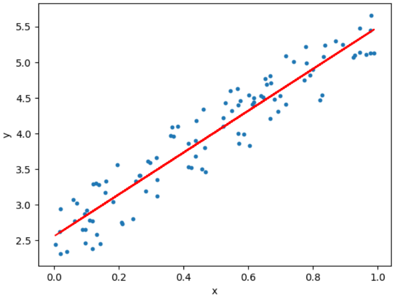
$$h(x) = \sum_{i=0}^{n} \theta_i x_i$$

Goal

Learn the values of θ that results in **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** $J(\theta)$.

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

2a) Cost Function:

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

Choose θ such that:

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

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

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 $z^T z = \sum_i z_i^2$:

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

$$J(\theta) = \frac{1}{2}(H - Y)^T(H - Y) \tag{3}$$

Ordinary Least Squares (OLS)

Optimization

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

$$x^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_n^{(1)} \end{bmatrix}, \qquad x \in X, \qquad X = \begin{bmatrix} \left(x^{(1)}\right)^T \\ \left(x^{(2)}\right)^T \\ \vdots \\ \left(x^{(n)}\right)^T \end{bmatrix}$$

$$\mathrm{y} \in Y, \qquad Y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

Ordinary Least Squares (OLS)

Optimization

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

$$h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i$$

In matrix form, this is equivalent to:

$$h_\theta(x^{(i)}) = (x^{(i)})^T \theta \tag{4}$$
$$H = X\Theta \tag{5}$$

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.

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

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

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

Using $(ab)^T = b^T a^T$ and $a^T b = b^T a$:

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

Ordinary Least Squares (OLS)

Optimization

continued...

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

$$\text{Using } \nabla_x b^T x = b \text{ and } \nabla_x x^T Ax = 2Ax \text{ for symmetric matrix } A:$$

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

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

Solving for Θ:

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

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

Ordinary Least Squares (OLS)

Optimization

Ordinary Least Squares (OLS) Solution

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

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

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) / 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 $J(\theta)$.

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

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

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

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

Optimization

Stochastic Gradient Descent

Optimization

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

	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.

References

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