Least Mean Squares Regression
Lecture Overview

• Linear classifiers

• What functions do linear classifiers express?

• Least Squares Method for Regression
Where are we?

• Linear classifiers

• What functions do linear classifiers express?

• Least Squares Method for regression
  – Examples
  – The LMS objective
  – Gradient descent
  – stochastic gradient descent
What’s the mileage?

Suppose we want to predict the mileage of a car from its weight and age

<table>
<thead>
<tr>
<th>Weight (x 100 lb)</th>
<th>Age (years)</th>
<th>Mileage</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>(x_2)</td>
<td></td>
</tr>
<tr>
<td>31.5</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>36.2</td>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>43.1</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>27.6</td>
<td>2</td>
<td>30</td>
</tr>
</tbody>
</table>

What we want: A function that can predict mileage using \(x_1\) and \(x_2\).
Linear regression: The strategy

Predicting continuous values using a linear model

Assumption: The output is a linear function of the inputs
Mileage = \( w_0 + w_1 x_1 + w_2 x_2 \)

Learning: Using the training data to find the best possible value of \( w \)

Prediction: Given the values for \( x_1, x_2 \) for a new car, use the learned \( w \) to predict the Mileage for the new car
Linear regression: The strategy

Predicting continuous values using a linear model

**Assumption:** The output is a linear function of the inputs

$$\text{Mileage} = w_0 + w_1 x_1 + w_2 x_2$$

**Learning:** Using the training data to find the *best* possible value of $w$

**Prediction:** Given the values for $x_1, x_2$ for a new car, use the learned $w$ to predict the **Mileage** for the new car
Linear regression: The strategy

- **Inputs** are vectors: $\mathbf{x} \in \mathbb{R}^d$
- **Outputs** are real numbers: $y \in \mathbb{R}$

We have a training set
$$D = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \}$$

We want to approximate $y$ as
$$y = f_w(x) = w_1 + w_2 x_2 + \cdots + w_n x_n = \mathbf{w}^T \mathbf{x}$$

For simplicity, we will assume that $x_1$ is always 1.

That is $x = [1 \ x_2 \ x_3 \ \cdots \ x_d]^T$

This lets makes notation easier

$\mathbf{w}$ is the learned weight vector in $\mathbb{R}^d$
Examples

One dimensional input
Examples

Predict using $y = w_1 + w_2 x_2$

One dimensional input
Examples

Predict using $y = w_1 + w_2 x_2$

One dimensional input

Predict using $y = w_1 + w_2 x_2 + w_3 x_3$

Two dimensional input
What is the best weight vector?

*Question*: How do we know which weight vector is the *best* one for a training set?
What is the best weight vector?

*Question*: How do we know which weight vector is the *best* one for a training set?

For an example $(x_i, y_i)$ in the training set, the *cost* of a mistake is

$$ |y_i - w^T x_i| $$
What is the best weight vector?

*Question*: How do we know which weight vector is the *best* one for a training set?

For an input \((x_i, y_i)\) in the training set, the *cost* of a mistake is

\[
|y_i - w^T x_i|
\]

Define the cost (or *loss*) for a particular weight vector \(w\) to be

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]
What is the best weight vector?

*Question:* How do we know which weight vector is the *best* one for a training set?

For an input \((x_i, y_i)\) in the training set, the *cost* of a mistake is

\[
|y_i - w^T x_i|
\]

Define the cost (or *loss*) for a particular weight vector \(w\) to be

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]  

Sum of squared costs over the training set.
What is the best weight vector?

**Question**: How do we know which weight vector is the best one for a training set?

For an input \((x_i, y_i)\) in the training set, the **cost** of a mistake is

\[
\left| y_i - w^T x_i \right|
\]

Define the cost (or **loss**) for a particular weight vector \(w\) to be

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]

One strategy for learning: **Find the \(w\) with least cost on this data**
Least Mean Squares (LMS) Regression

Learning: minimizing mean squared error

\[
\min_w \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]
Least Mean Squares (LMS) Regression

\[
\min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^{m} (y_i - \mathbf{w}^T \mathbf{x}_i)^2
\]

Learning: minimizing mean squared error

Different strategies exist for \textit{learning by optimization}

- Gradient descent is a popular algorithm
  - (For this minimization objective, there is also an analytical solution)
Gradient descent

General strategy for minimizing a function $J(w)$

- Start with an initial guess for $w$, say $w^0$
- Iterate till convergence:
  - Compute the gradient of the gradient of $J$ at $w^t$
  - Update $w^t$ to get $w^{t+1}$ by taking a step in the opposite direction of the gradient

We are trying to minimize

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$

**Intuition**: The gradient is the direction of increase in the function. To get to the minimum, go in the opposite direction
Gradient descent

General strategy for minimizing a function $J(w)$

• Start with an initial guess for $w$, say $w^0$

• Iterate till convergence:
  – Compute the gradient of the gradient of $J$ at $w^t$
  – Update $w^t$ to get $w^{t+1}$ by taking a step in the opposite direction of the gradient

Intuition: The gradient is the direction of increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$
Gradient descent

General strategy for minimizing a function $J(w)$

• Start with an initial guess for $w$, say $w^0$

• Iterate till convergence:
  – Compute the gradient of the gradient of $J$ at $w^t$
  – Update $w^t$ to get $w^{t+1}$ by taking a step in the opposite direction of the gradient

Intuition: The gradient is the direction of increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$
Gradient descent

General strategy for minimizing a function $J(w)$

• Start with an initial guess for $w$, say $w^0$

• Iterate till convergence:
  
  – Compute the gradient of the gradient of $J$ at $w^t$
  – Update $w^t$ to get $w^{t+1}$ by taking a step in the opposite direction of the gradient

Intuition: The gradient is the direction of increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$
Gradient descent

General strategy for minimizing a function $J(w)$

- Start with an initial guess for $w$, say $w^0$

- Iterate till convergence:
  - Compute the gradient of the gradient of $J$ at $w^t$
  - Update $w^t$ to get $w^{t+1}$ by taking a step in the opposite direction of the gradient

Intuition: The gradient is the direction of increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$
Gradient descent

General strategy for minimizing a function $J(w)$

- Start with an initial guess for $w$, say $w^0$
- Iterate till convergence:
  - Compute the gradient of the gradient of $J$ at $w^t$
  - Update $w^t$ to get $w^{t+1}$ by taking a step in the opposite direction of the gradient

Intuition: The gradient is the direction of increase in the function. To get to the minimum, go in the opposite direction

We are trying to minimize

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$
Gradient descent for LMS

1. Initialize $w^0$

2. For $t = 0, 1, 2, \ldots$
   1. Compute gradient of $J(w)$ at $w^t$. Call it $\nabla J(w^t)$

2. Update $w$ as follows:

   $$w^{t+1} = w^t - r\nabla J(w^t)$$

   $r$: Called the learning rate
   (For now, a small constant. We will get to this later)
Gradient descent for LMS

We are trying to minimize

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$

1. Initialize $w^0$

2. For $t = 0, 1, 2, \ldots$
   
   1. Compute gradient of $J(w)$ at $w^t$. Call it $\nabla J(w^t)$

   2. Update $w$ as follows:

      $$w^{t+1} = w^t - r \nabla J(w^t)$$

      $r$: Called the **learning rate**
      (For now, a small constant. We will get to this later)
Gradient of the cost

- The gradient is of the form \( \nabla J(w^t) = \left[ \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d} \right] \)

- Remember that \( w \) is a vector with \( d \) elements
  - \( w = [w_1, w_2, w_3, \ldots w_j, \ldots, w_d] \)

We are trying to minimize

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]
Gradient of the cost

• The gradient is of the form $\nabla J(w^t) = \left[ \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d} \right]$.

$$\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$

We are trying to minimize

$$J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$$
Gradient of the cost

• The gradient is of the form \( \nabla J(w) = \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d} \)

\[
\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]

\[
= \frac{1}{2} \sum_{i=1}^{m} \frac{\partial}{\partial w_j} (y_i - w^T x_i)^2
\]
Gradient of the cost

• The gradient is of the form

\[ \nabla J(w^t) = \left[ \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d} \right] \]

\[
\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2 \\
= \frac{1}{2} \sum_{i=1}^{m} \frac{\partial}{\partial w_j} (y_i - w^T x_i)^2 \\
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i) \frac{\partial}{\partial w_j} (y_i - w_1 x_{i1} - \cdots w_j x_{ij} - \cdots)
\]

We are trying to minimize

\[ J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2 \]
Gradient of the cost

• The gradient is of the form \( \nabla J(w^t) = \left[ \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d} \right] \)

\[
\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2 \\
= \frac{1}{2} \sum_{i=1}^{m} \frac{\partial}{\partial w_j} (y_i - w^T x_i)^2 \\
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i) \frac{\partial}{\partial w_j} (y_i - w_1 x_1 - \cdots w_j x_{ij} - \cdots) \\
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i)(-x_{ij})
\]
Gradient of the cost

The gradient is of the form $\nabla J(w^t) = \left[ \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d} \right]$

\[
\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2 \\
= \frac{1}{2} \sum_{i=1}^{m} \frac{\partial}{\partial w_j} (y_i - w^T x_i)^2 \\
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i) \frac{\partial}{\partial w_j} (y_i - w_1 x_{i1} - \cdots w_j x_{ij} - \cdots) \\
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i) (-x_{ij}) \\
= - \sum_{i=1}^{m} (y_i - w^T x_i) x_{ij}
\]

We are trying to minimize $J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2$
Gradient of the cost

- The gradient is of the form \( \nabla J(w^t) = [\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d}] \)

\[
\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]

\[
= \frac{1}{2} \sum_{i=1}^{m} \frac{\partial}{\partial w_j} (y_i - w^T x_i)^2
\]

\[
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i) \frac{\partial}{\partial w_j} (y_i - w_1 x_{i1} - \cdots w_j x_{ij} - \cdots)
\]

\[
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i)(-x_{ij})
\]

\[
= -\sum_{i=1}^{m} (y_i - w^T x_i)x_{ij}
\]

We are trying to minimize

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]
Gradient of the cost

- The gradient is of the form \( \nabla J(w) = \left[ \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_d} \right] \)

\[
\frac{\partial J}{\partial w_j} = \frac{\partial}{\partial w_j} \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
= \frac{1}{2} \sum_{i=1}^{m} \frac{\partial}{\partial w_j} (y_i - w^T x_i)^2
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i) \frac{\partial}{\partial w_j} (y_i - w_1 x_{i1} - \ldots w_j x_{ij} - \ldots)
= \frac{1}{2} \sum_{i=1}^{m} 2(y_i - w^T x_i)(-x_{ij})
= -\sum_{i=1}^{m} (y_i - w^T x_i) x_{ij}
\]

We are trying to minimize

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]
Gradient descent for LMS

1. Initialize $w^0$

2. For $t = 0, 1, 2, \ldots$
   1. Compute gradient of $J(w)$ at $w^t$. Call it $\nabla J(w^t)$
      
      Evaluate the function for **each** training example to compute the error and construct the gradient vector
      
      $$
      \frac{\partial J}{\partial w_j} = - \sum_{i=1}^{m} (y_i - w^T x_i) x_{ij}
      $$

We are trying to minimize

$$
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
$$
Gradient descent for LMS

1. Initialize $\mathbf{w}^0$

2. For $t = 0, 1, 2, \ldots$
   1. Compute gradient of $J(\mathbf{w})$ at $\mathbf{w}^t$. Call it $\nabla J(\mathbf{w}^t)$
      
      Evaluate the function for each training example to compute the error and construct the gradient vector
      
      $\frac{\partial J}{\partial w_j} = - \sum_{i=1}^{m} (y_i - \mathbf{w}^T \mathbf{x}_i) x_{ij}$

      One element of $\nabla J$

We are trying to minimize

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{m} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$
Gradient descent for LMS

1. Initialize $\mathbf{w}^0$

2. For $t = 0, 1, 2, \ldots$
   1. Compute gradient of $J(\mathbf{w})$ at $\mathbf{w}^t$. Call it $\nabla J(\mathbf{w}^t)$
      Evaluate the function for each training example to compute the error and construct the gradient vector
      \[
      \frac{\partial J}{\partial w_j} = -\sum_{i=1}^{m} (y_i - \mathbf{w}^T \mathbf{x}_i) x_{ij}
      \]
      One element of $\nabla J$
   2. Update $\mathbf{w}$ as follows:
      \[
      \mathbf{w}^{t+1} = \mathbf{w}^t - r \nabla J(\mathbf{w}^t)
      \]

We are trying to minimize
\[
J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{m} (y_i - \mathbf{w}^T \mathbf{x}_i)^2
\]
Gradient descent for LMS

1. Initialize \( w^0 \)

2. For \( t = 0, 1, 2, \ldots \) (until total error is below a threshold)
   1. Compute gradient of \( J(w) \) at \( w^t \). Call it \( \nabla J(w^t) \)
   
      Evaluate the function for each training example to compute the error and construct the gradient vector
      
      \[
      \frac{\partial J}{\partial w_j} = - \sum_{i=1}^{m} (y_i - w^T x_i) x_{ij}
      \]

   2. Update \( w \) as follows: 
      
      \[
      w^{t+1} = w^t - r \nabla J(w^t)
      \]

We are trying to minimize

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]
Gradient descent for LMS

We are trying to minimize

\[ J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2 \]

1. Initialize \( w^0 \)

2. For \( t = 0, 1, 2, \ldots \) (until total error is below a threshold)
   
   1. Compute gradient of \( J(w) \) at \( w^t \). Call it \( \nabla J(w^t) \)
      
      Evaluate the function for each training example to compute the error and construct the gradient vector
      
      \[
      \frac{\partial J}{\partial w_j} = -\sum_{i=1}^{m} (y_i - w^T x_i) x_{ij}
      \]
      
      One element of \( \nabla J \)

   2. Update \( w \) as follows:
      
      \[
      w^{t+1} = w^t - r \nabla J(w^t)
      \]
      
      \( r \): Called the learning rate
      (For now, a small constant. We will get to this later)
Gradient descent for LMS

1. Initialize $w^0$

2. For $t = 0, 1, 2, \ldots$ (until total error is below a threshold)
   
   1. Compute gradient of $J(w)$ at $w^t$. Call it $\nabla J(w^t)$

   Evaluate the function for each training example to compute the error and construct the gradient vector

   
   $$\frac{\partial J}{\partial w_j} = -\sum_{i=1}^{m} (y_i - w^T x_i) x_{ij}$$

   One element of $\nabla J$

   2. Update $w$ as follows: $w^{t+1} = w^t - r \nabla J(w^t)$

   $r$: Called the learning rate
   (For now, a small constant. We will get to this later)

   This algorithm is guaranteed to converge to the minimum of $J$ if $r$ is small enough.
   Why? The objective $J$ is a convex function
Gradient descent for LMS

1. **Initialize** \( w^0 \)

2. **For** \( t = 0, 1, 2, \ldots \) **(until total error is below a threshold)**
   
   1. **Compute gradient of** \( J(w) \) **at** \( w^t \). **Call it** \( \nabla J(w^t) \)
      
      Evaluate the function for *each* training example to compute the error and construct the gradient vector
      
      \[
      \frac{\partial J}{\partial w_j} = -\sum_{i=1}^{m} (y_i - w^T x_i) x_{ij}
      \]

   2. **Update** \( w \) **as follows:**
      
      \[
      w^{t+1} = w^t - r \nabla J(w^t)
      \]

We are trying to minimize

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]
Gradient descent for LMS

1. Initialize \( w^0 \)

2. For \( t = 0, 1, 2, \ldots \) (until total error is below a threshold)
   1. Compute gradient of \( J(w) \) at \( w^t \). Call it \( \nabla J(w^t) \)
      Evaluate the function for each training example to compute the error and construct the gradient vector
      \[
      \frac{\partial J}{\partial w_j} = -\sum_{i=1}^{m} (y_i - w^T x_i) x_{ij}
      \]
   2. Update \( w \) as follows:
      \[
      w^{t+1} = w^t - r \nabla J(w^t)
      \]

   The weight vector is not updated until all errors are calculated

Why not make early updates to the weight vector as soon as we encounter errors instead of waiting for a full pass over the data?

We are trying to minimize

\[
J(w) = \frac{1}{2} \sum_{i=1}^{m} (y_i - w^T x_i)^2
\]
Stochastic gradient descent

• Repeat for each example \((x_i, y_i)\)
  – Pretend that the entire training set is represented by this single example
  – Use this example to calculate the gradient and update the model

• Contrast with *batch gradient descent* which makes one update to the weight vector for every pass over the data
Stochastic gradient descent

1. Initialize $\mathbf{w}$

2. For $t = 0, 1, 2, \ldots$ (until error below some threshold)
   - For each training example $(\mathbf{x}_i, y_i)$:
     - Update $\mathbf{w}$. For each element of the weight vector ($w_j$):
       $$ w_j^{t+1} = w_j^t + r(y_i - \mathbf{w}^T \mathbf{x}_i)x_{ij} $$
Stochastic gradient descent

1. Initialize \( \mathbf{w} \)

2. For \( t = 0, 1, 2, \ldots \) (until error below some threshold)
   
   - For each training example \((\mathbf{x}_i, y_i)\):
     
     • Update \( \mathbf{w} \). For each element of the weight vector \((w_j)\):
       
       \[
       w_{j}^{t+1} = w_{j}^{t} + r(y_{i} - \mathbf{w}^{T}\mathbf{x}_i)x_{ij}
       \]

Contrast with the previous method, where the weights are updated only after all examples are processed once.
Stochastic gradient descent

1. Initialize $w$

2. For $t = 0, 1, 2, \ldots$ (until error below some threshold)
   - For each training example $(x_i, y_i)$:
     • Update $w$. For each element of the weight vector ($w_j$):
       $$w_j^{t+1} = w_j^t + r(y_i - w^T x_i)x_{ij}$$

This update rule is also called the Widrow-Hoff rule in the neural networks literature.
Stochastic gradient descent

1. Initialize $w$

2. For $t = 0, 1, 2, \ldots$ (until error below some threshold)
   - For each training example $(x_i, y_i)$:
     - Update $w$. For each element of the weight vector ($w_j$):
       $$ w_j^{t+1} = w_j^t + r(y_i - w^T x_i) x_{ij} $$

This update rule is also called the Widrow-Hoff rule in the neural networks literature

Online/stochastic algorithms are often preferred when the training set is very large

May get close to optimum much faster than the batch version
Learning Rates and Convergence

• In the general case the learning rate $r$ must decrease to zero to guarantee convergence.

• The learning rate is called the *step size*.
  – More sophisticated algorithms choose the step size automatically and converge faster.

• Choosing a better starting point can also have impact.

• Gradient descent and its stochastic version are very simple algorithms.
  – Yet, almost all the algorithms we will learn in the class can be traced back to gradient decent algorithms for different loss functions and different hypotheses spaces.
Linear regression: Summary

• **What we want**: Predict a real valued output using a feature representation of the input

• **Assumption**: Output is a linear function of the inputs

• Learning by minimizing total cost
  – Gradient descent and stochastic gradient descent to find the *best* weight vector
  – This particular optimization can be computed directly by framing the problem as a matrix problem
Exercises

1. LMS regression can be solved analytically. Given a dataset \( D = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m) \} \), define matrix \( X \) and vector \( Y \) as follows:

\[
X = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix}_{d \times m}
\]

\[
Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{m \times 1}
\]

Show that the optimization problem we saw earlier is equivalent to

\[
\min_w (X^T w - Y)^T (X^T w - Y)
\]

This can be solved analytically. Show that the solution \( w^* \) is

\[
w^* = (XX^T)^{-1} XY
\]

**Hint:** You have to take the derivative of the objective with respect to the vector \( w \) and set it to zero.